# RZ/A1LU Group

## QSPI Flash Boot Loader

R11AN0084EG0300
Rev.3.00
May 10, 2018

## Introduction

This application note describes the technical details of configuring and accessing the QSPI flash memory fitted on the Stream it! - RZ V2 board using e² studio.

Please refer to the Stream it! - RZ Kit User's Manual (R20UT3823EG) for more details of the Stream it! - RZ V2 board.

Please refer to the RZ/A1LU Group, RZ/A1LU Group Hardware Device Manual (R01UH0437EJ) for more details of the RZ/A1LU MCU.

## Target Device

RZ/A1LU

## Contents

# 1. Overview

QSPI flash is a serial peripheral interface with multiple data lines. A key feature of this interface is the ability to connect two serial flash memories to a channel. The data bus size for each channel can be specified as 1-bit, 2-bits or 4-bits. The RZ/A1LU device supports a single channel of QSPI memory.

## 1.1 Address Space

For the address map of the RZ/A1LU device please refer to section 5 of the Device Hardware Manual.

## 1.2 Summary

A boot-loader is the first program executed in the microcontroller following a system reset. It is used to configure the device to a known state; load a new boot program and, if necessary, copy the main program to the allocated program memory (RAM) of the microcontroller.

The RZ/A1LU has a single QSPI channel, and allows for one or two serial flash memories to be directly connected. The number of connected memories is specified by writing to the BSZ bits of register CMNCR of the QSPI bus controller. Booting from QSPI is performed using Single Channel Single Bit SPI mode. On the Stream it! - RZ V2, two QSPI flash devices are connected to the QSPI channel.

The RZ/A1LU supports four boot modes:

- Boot mode 0: Boots the LSI from the memory (bus width: 16 bits) connected to the CS0 space
- Boot mode 1: Boots the LSI from the serial flash memory connected to the SPI multi I/O bus space
- Boot mode 2: Boots the LSI from the NAND flash memory with the SD controller enabled
- Boot mode 3: Boots the LSI from the NAND flash memory with the MMC controller enabled

| QSPI Signal Name | Stream-IT v2 Signal Name | Function | Comment |
|---|---|---|---|
| SPBSSL | SPBSSL_0 | Slave Select | Common to both QSPI devices (U4 and U16) |
| SPBCLK | SPBCLK_0 | Clock | Common to both QSPI devices (U4 and U16) |
| SPBIO00 | SPBIO00_0 | Data | SI |
| SPBIO10 | SPBIO10_0 | Data | SO |
| SPBIO20 | SPBIO20_0 | Data | Not configured |
| SPBIO30 | SPBIO30_0 | Data | Not configured |
| SPBIO01 | SPBIO01_0 | Data | Not configured |
| SPBIO11 | SPBIO11_0 | Data | Not configured |
| SPBIO21 | SPBIO21_0 | Data | Not configured |
| SPBIO31 | SPBIO31_0 | Data | Not configured |

**Table 1 – QSPI Connections at boot**

Figures 1 and 2 are a graphical representation of the boot sequence between the RZ/A1LU and the QSPI.

Key: Red Text Boot process stages workflow (stage 1->2->3->4)

Purple Text Highlights RAM memory

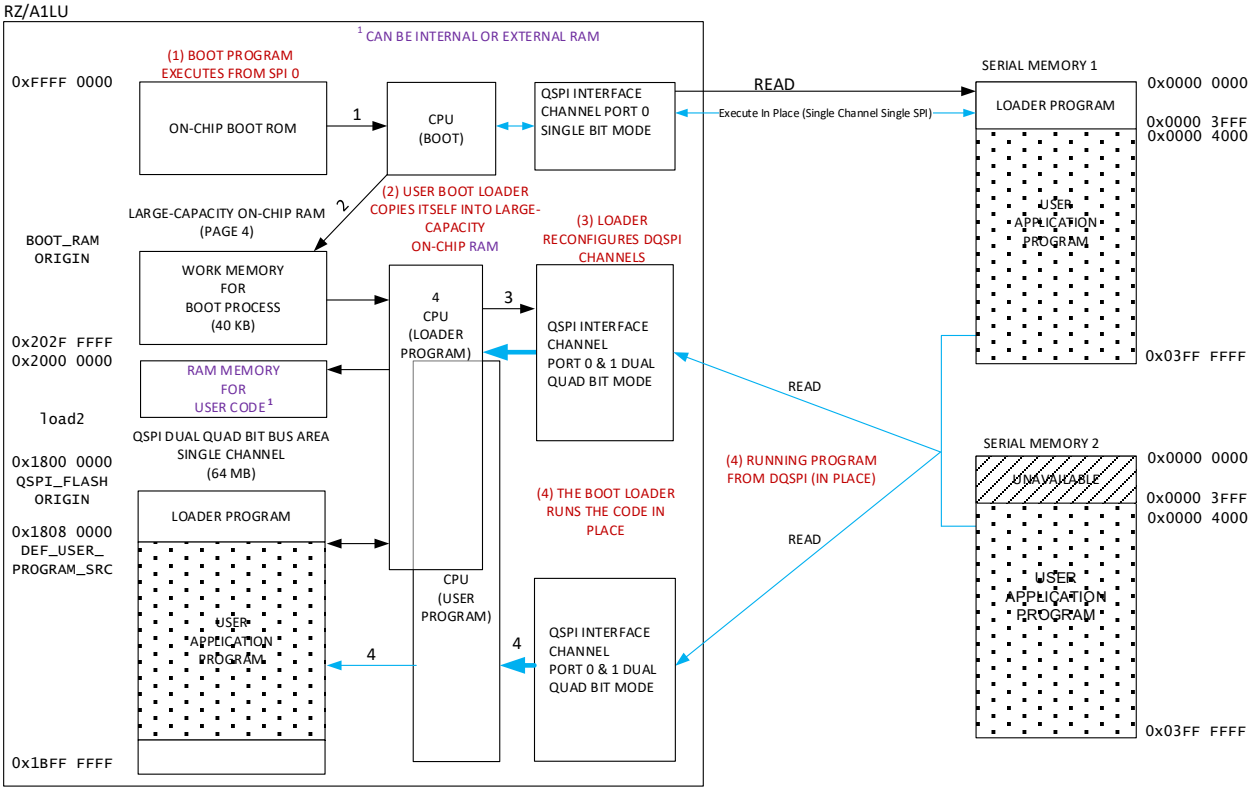


**Figure 1 Boot process of user code stored in QSPI devices and executed in place**

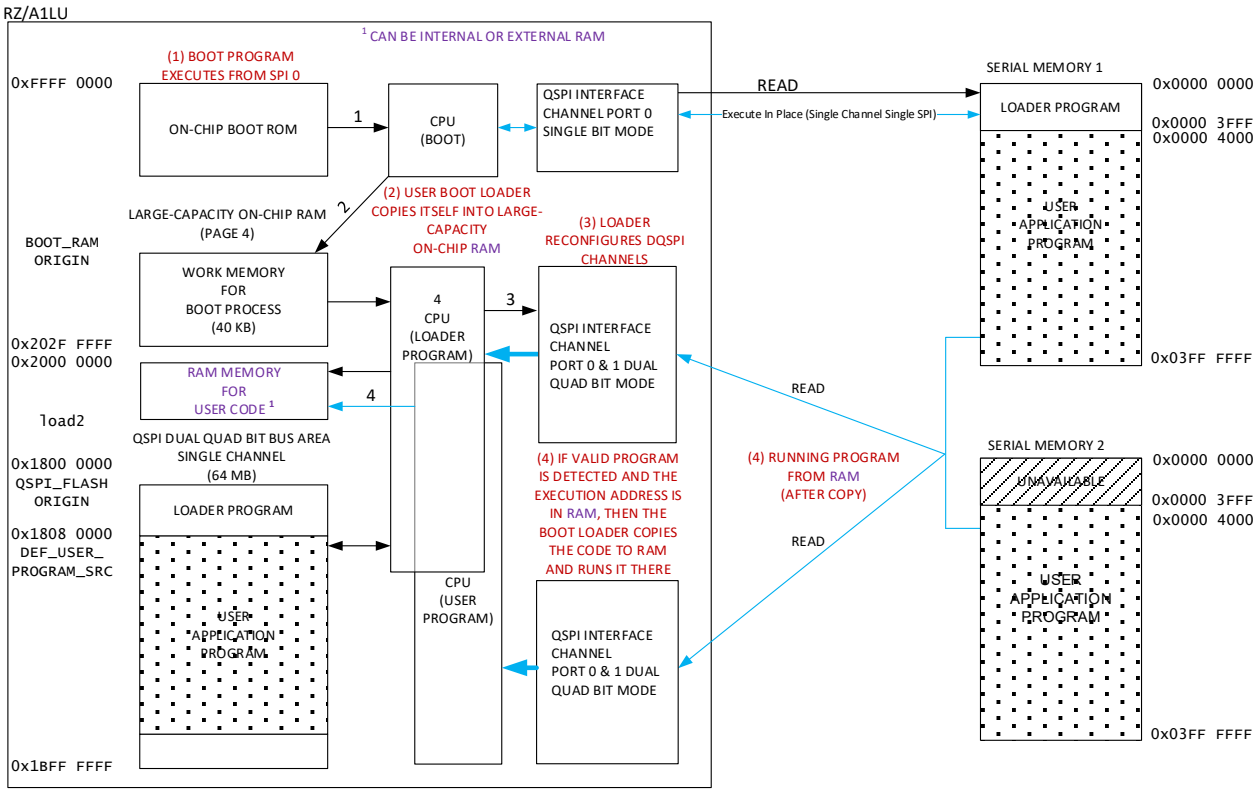


**Figure 2 Boot process of user code copied from QSPI devices to RAM, and executed there**

# 2. QSPI Boot Process

## 2.1 Initiation of the QSPI Channel in Single Bit Mode

Following a reset, the RZ/A1LU executes the Boot Program located in the high exception vector address 0xFFFF 0000; which then configures the QSPI bus Channel Port 0 only in Single Bit mode and external address space read mode, ready to read directly from the connected serial flash memory.

Execution of user code in this configuration is possible, but will be slower than necessary. Therefore a small User Boot Loader is provided.

## 2.2 Transfer of the Loader Program

The User Boot Loader Program copies itself to internal RAM and executes here. The Loader program can then disable the QSPI channel, reconfigure it to the Dual port Quad Bit SPI mode for maximum speed, and look for a user application program.

Note: The external address space for the QSPI channel is mapped internally to the address range 0x1800 0000 to 0x1BFF FFFF, giving an address space of 64MB. However, the RZ/A1LU supports up to two 4GB memory devices giving a total of eight gigabytes of memory. This memory is accessed in 64MB chunks using a paging scheme via the DREAR (Data Read Extended Address setting Register) and DRENR (Data Read Enable setting Register) registers. Please see the Hardware Device Manual for further details. The RZ/A1LU Boot Program uses 40kB of work RAM (from BOOT_RAM ORIGIN to 0x202F FFFF).

## 2.3 Boot Loader Transfer of a User Application Program

After configuring the external address space, the User Boot Loader Program inspects the user application program in QSPI memory to validate its configuration. There should be four items specified there. These are the start and end addresses of the user application program, the execution address, and a boot validation string. They can be found in the *start.s* file of the sample e² studio project. These labels must be correctly placed with accurate data directly after your reset vectors. Failure to do this will render the bootloader unable to decipher your application, and your code won't start. A snippet of the contents of the *start.s* file is shown below:

```
    .text
    .code 32

    .global start
    .func   start
start:
    LDR pc, =reset_handler
    LDR pc, =undefined_handler
    LDR pc, =svc_handler
    LDR pc, =prefetch_handler
    LDR pc, =abort_handler
    LDR pc, =reserved_handler
    LDR pc, =irq_handler
    LDR pc, =fiq_handler
code_start:
    .word    start
code_end:
    .word    end
code_execute:
    .word    execute
    .string ".BootLoad_ValidProgramTest."
    .align 4
    .end
```

The 'start' is a function/label for loading the user application code's vector table. The 'code_start' and 'code_end' labels contain variables specifying the start and end addresses of the entire user application code, including the vector addresses.

The 'code_execute' label contains the 'execute' variable used to indicate the execution start address.

The '.string' variable is a signature marker used by the Loader Program to validate the user application code, whether to load the code or not.

If the location constants and .string variable in the *start.s* file are not found immediately after the vector table as shown above, the configuration is deemed invalid and so that the error can be recognized, user LED (D13) will flash in a continually repeating sequence of long (~2 second) flashes with a delay of ~0.5 seconds between flashes. If valid, the User Boot Loader Program checks the start address. If this matches the current location, the execute address is used and the program is launched and executes from QSPI.

Otherwise the start and end addresses are used to copy the program to the required destination in RAM. On completion the program is launched at the provided execute address.

## 2.4 Boot Loader Sections

The boot loader code is arranged in sections, separate from the user application code. The memory map from the previous page is explained below:

**Section 1: reset**

This section contains the reset vectors and initialization code.

**Section 2: load2**

This section contains code to speed up the QSPI and set I/O ports (rza_io_regrw.c). This code is executed in RAM as it cannot change the QSPI access speed when executing from it.

**Section 3: load3**

This section contains code to set QSPI into quad bit mode, using both the devices. This code is executed in RAM as it cannot configure the QSPI when executing from it. It then checks if there is an application in the start location (DEF_USER_PROGRAM_SRC 0x1808 0000). If there is, it checks if this application should be executed from QSPI or from RAM. For QSPI, it jumps to the start location; if not, it copies the code into RAM at the location specified 'code_start', and then jumps to it.

**Note:**

The boot loader can be installed by first building the Release configuration of the StreamIt2_QSPI_Loader sample project. This can be programmed in to the QSPI devices via the J-Link debugger by either using the e$^2$ studio 'Debug' launch configuration or by running the Program_QSPI_Loader.bat file located in the project directory.

The board is shipped with the boot loader pre-installed.

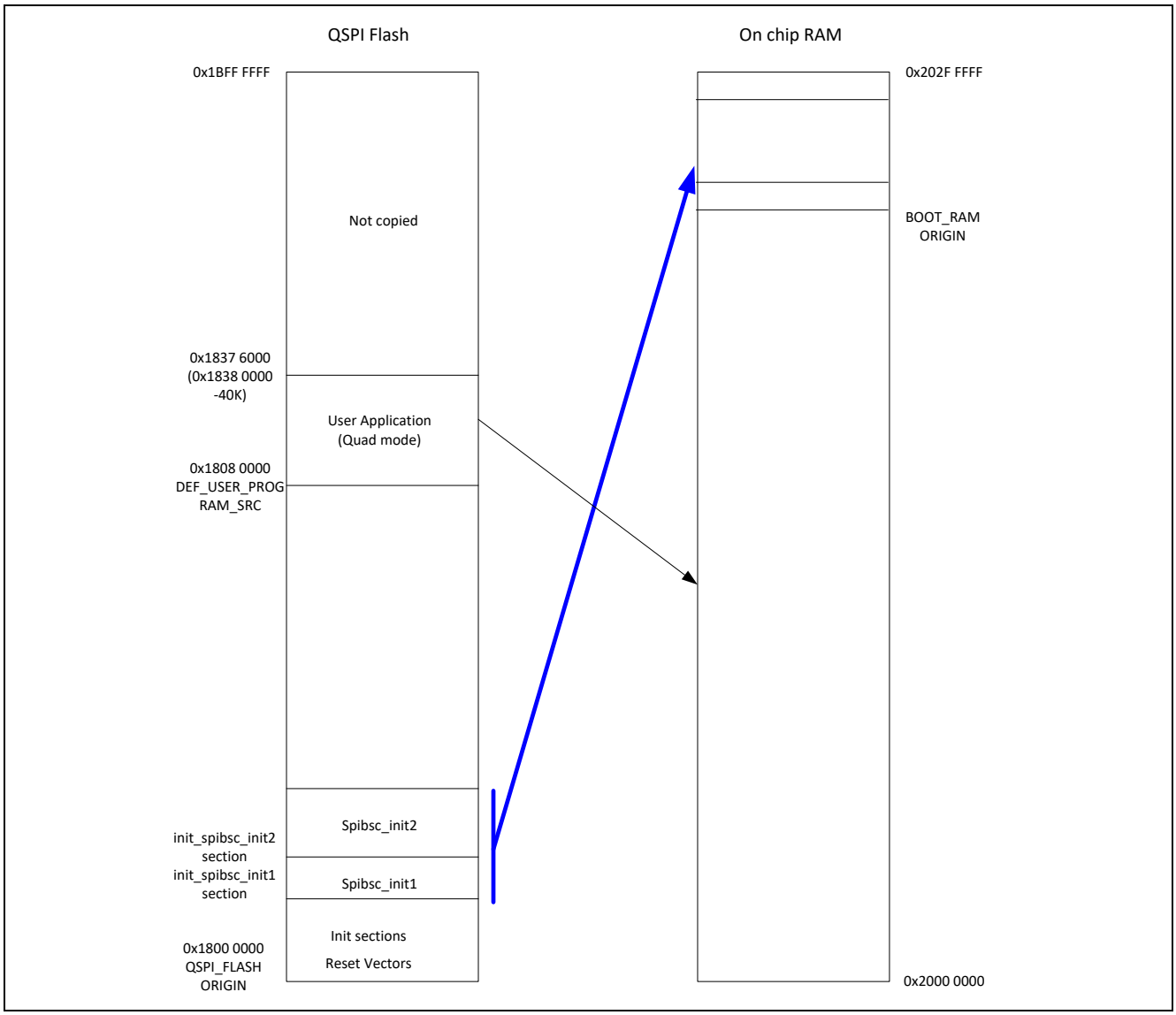


**Figure 3 Transfer of QSPI device data to the RZ/A1LU On-chip RAM**

The QSPI Flash device is shown mapped to the RZ/A1LU's QSPI bus area. The start address of the user application in SPI is determined by the size of the lowest block in the QSPI device, this is 0x40000 in each QSPI device, resulting in a Dual QSPI start location for the first available block being 0x80000.

## 2.5 Destination RAM Details

The RAM area from 0x2000 0000 to 0x2020 0000 is on-chip Data Retention RAM.

This area is not write enabled when the RZ/A1LU initially starts up, requiring bits in SYSCR3 to be set before allowing writes. These bits are set in the boot loader; allowing the user's application access to the full address range: up to BOOT_RAM ORIGIN.

Note: without the relevant bits set in SYSCR3, only RAM from 0x2002 0000 is writable. Many applications start at this address, allowing downloading to RAM from e$^2$ studio.

## 2.6 Code Description

In order to understand the boot loader code it helps to know where the code is located.

Start of QSPI memory (QSPI_FLASH : ORIGIN = 0x1800 0000):

**start.s**
**vbar_init.s**
**reset_handler.s**

**resetprg.c**
**init_spibsc_init1_section.c**
**init_spibsc_init2_section.c**

LOAD_MODULE2 (QSPI g_ld_load_module2_addr):

**spibsc_init1.c**
**rza_io_regrw.c**

LOAD_MODULE3 (QSPI g_ld_load_module3_addr):

**spibsc_init2.c**
**user_prog_jmp.s**
**spibsc_flash_api.c**
**spibsc_flash_userdef.c**
**spibsc_ioset_api.c**
**spibsc_ioset_drv.c**
**spibsc_ioset_userdef.c**

The first C function to be executed is resetprg() in resetprg.c at the start of QSPI memory. This function manages the entire boot loader process by calling other functions as described below. Initially QSPI memory is running at low speed.

1) resetprg() first calls init_spibsc_init1_section() in **init_spibsc_init1_section.c** (executed in QSPI memory at slow speed) to copy LOAD_MODULE2 from QSPI memory to RAM.
2) Secondly, it calls spibsc_init1() in **spibsc_init1.c** (now in RAM) to change the QSPI interface to high speed with cache and bus mode on. When this function returns, resetprg() will be running from QSPI memory at full speed.
3) Thirdly, it calls init_spibsc_init2_section() in **init_spibsc_init2_section.c** (executed from QSPI memory) to copy LOAD_MODULE3 from QSPI memory to RAM.
4) Finally, it calls spibsc_init2() in **spibsc_init2.c** (executed in RAM) to check the address of the user program. If it's inside internal RAM, then the program is copied from QSPI memory to RAM, before being executed. Otherwise it will be executed in place.

## 2.7 Error Indication

In the event of an error, the red LED (D13) will flash continually to indicate that an error has occurred. The different error conditions are summarised in the table below.

| Error | Red LED (D13) flash sequence |
|---|---|
| User program too big to fit in available RAM | ~0.25s on, ~0.25s off |
| SPI memory initialisation failure | ~2s on, ~0.5s off, ~0.5s on, ~0.5s off |
| No error | Off |

**Table 2 – LED error indications**

Please note that the LED timings above are approximate. For simplicity, a simple delay loop is used for timing, but due to the effect of the code optimiser, the delays generated are not the same for successive calls to the routine.

# 3. QSPI Data Reading

Reading of QSPI data can be configured for single or dual device read, depending on the number of serial memories connected to a channel. Differences between the two modes are clearly explained in the next two diagrams. From the diagrams it is clear to see the advantages of dual read mode over single read. With two devices connected, using dual mode doubles the storage space and reduces the number cycles taken to read the same amount of data in single mode. In this case, the application code is stored interleaved between the two devices. The serial flash memory connected to the pin SPBIO30-SPBIO00 has the address 2n and the serial flash memory connected to the pin SPBIO31-SPBIO01 has the address 2n + 1. The data will be accessed in word or larger units - it cannot be accessed in byte units.

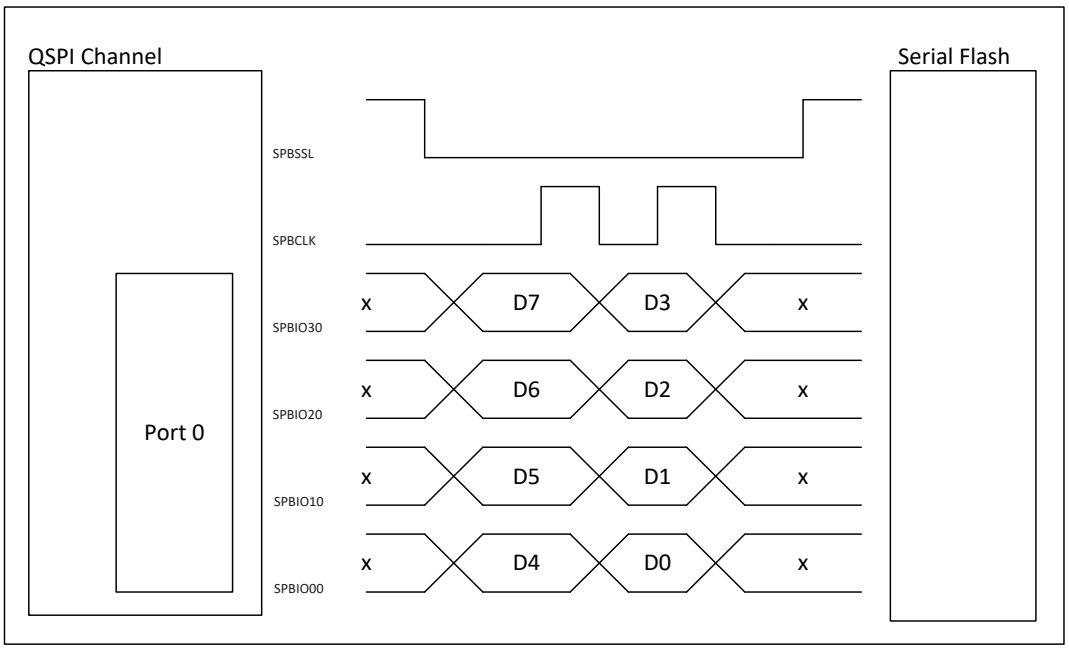


**Figure 4 Example of a 4-bit data size using one serial flash memory**

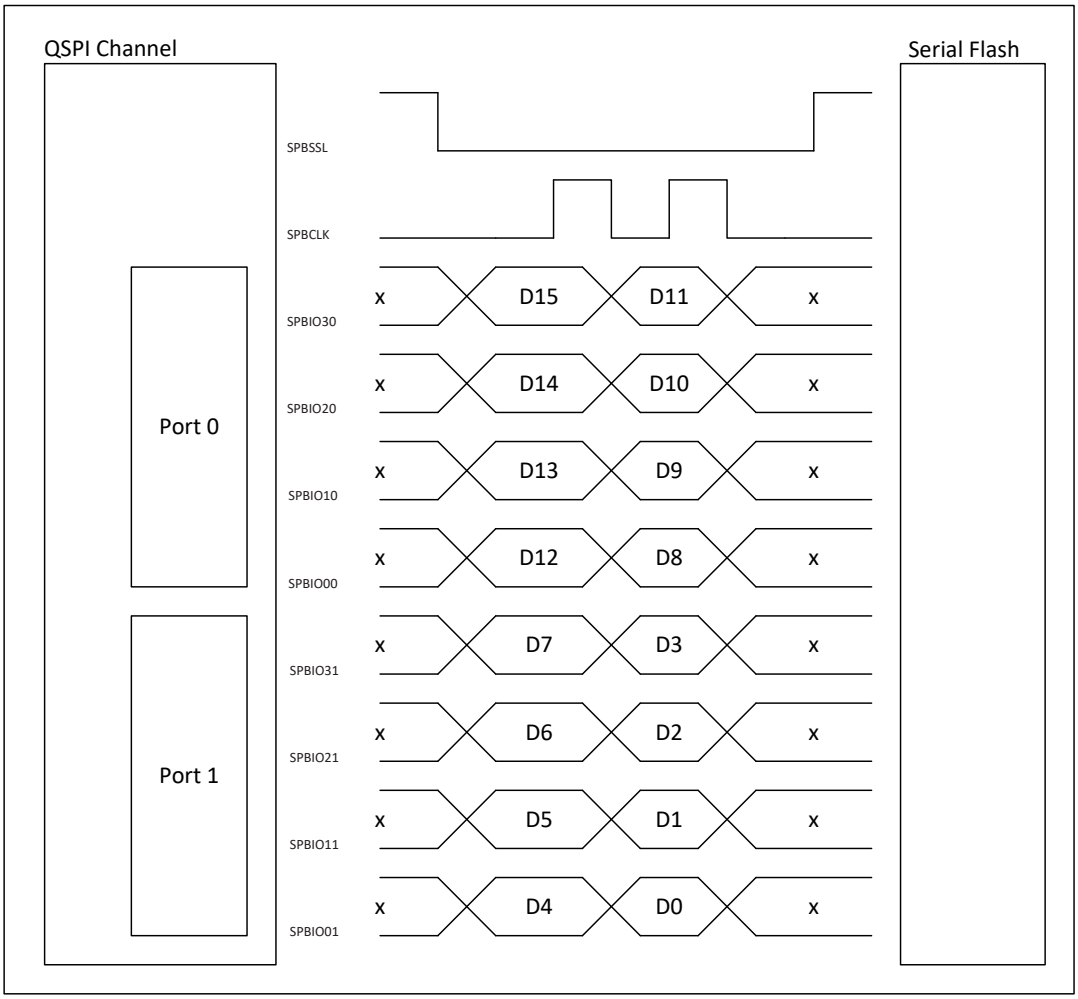


**Figure 5 Example of a 4-bit data size using two serial flash memories**

# 4. Sample project

You can download one of the sample applications from the Stream it! - RZ product page.

For the purposes of this document we are using the tutorial project.

## 4.1 Linker Files

The sample projects provide a linker (.ld) file (sometimes called a 'load file') used to specify where the user code is to be located and executed. The program is loaded into QSPI and then is copied at start-up by the User Boot Loader and runs from RAM. The linker file(s) can be found in the project folder under the Project Explorer view.

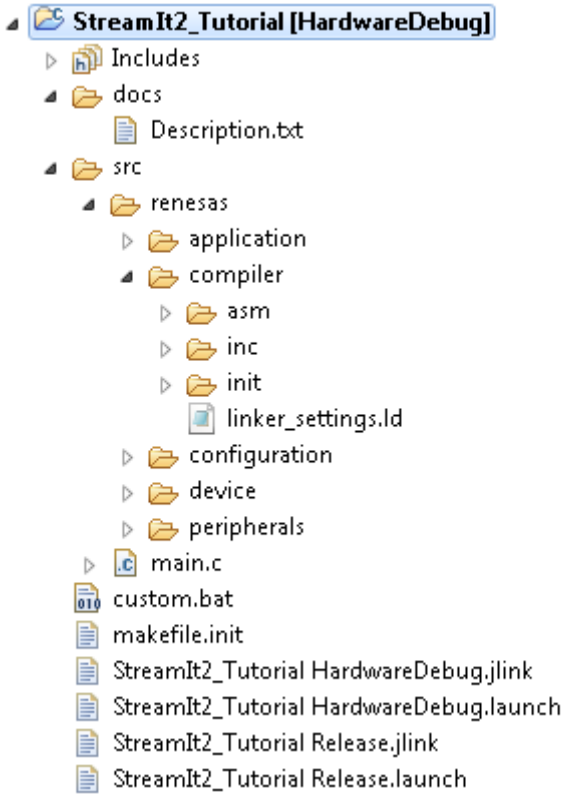


**Figure 6 Locating files in e² studio's Project Explorer**

The linker file contains the following lines:

```
EXEC_BASE               = 0x20040000;
```

The EXEC_BASE variable specifies where in memory the user application code finally resides and executed from.

The linker file is selected to override the project settings in the compiler options. This can be found by:

Select File > Properties > C/C++ Build > Settings > Linker > Other

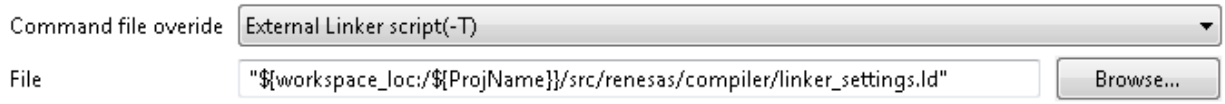


**Figure 7 Setting the project's linker file**

The RAM linker file is configured in all samples as supplied.

## 4.2 Running Code in Place

By default most Renesas sample projects will be configured to run code in RAM. However, it may be necessary to run your code in place in QSPI memory. This could be because your program is too big to fit into RAM, or because it uses the RAM for other purposes. Note that code running in place will not run as fast as code executing from RAM.

In order to run the code in place, changes need to be made to the project's linker (.ld) file. The example below shows snippets from a sample project where the file needs to be changed.

Firstly, change the setting of EXEC_BASE to EXEC_BASE_QSPI:

```
/* Set EXEC_BASE to run in RAM or run in place in QSPI */
/* EXEC_BASE = EXEC_BASE_RAM; */            /* Use this to run from RAM */
/* EXEC_BASE = EXEC_BASE_QSPI; */           /* Use this to run from QSPI */
EXEC_BASE = EXEC_BASE_QSPI;
```

Secondly, change all lines that tell the linker that a section is in USERRAM (but only those that have the option to put that section in QSPI). Some sections (like stack and data) have to stay in RAM.

For example:

```
/* } > USERRAM */     /* uncomment this if running from RAM / comment this out if
                         running from QSPI */
} > QSPI              /* uncomment this if running from QSPI / comment this out if
                         running from RAM */
```

## 4.3 Generating a Binary File

Programming a user application requires the program file to be in binary format (.bin). By default the HardwareDebug and Release builds are configured to generate ELF files with extensions .x and .mot. These two builds are configured to generate a binary file with extension .bin. This is achieved by the following post-build instruction in e$^2$ studio:

1. In e$^2$ studio, click on the desired project under the 'Project Explorer' view.
2. From the menu bar select 'File > Properties'.
3. In the Properties dialog select 'C/C++ Build > Settings'
4. Select the desired build 'Configuration'; either 'HardwareDebug' or 'Release'.
5. Select the 'Build Steps' tab.
6. Verify the 'Post-build steps' is as follows:
   `arm-none-eabi-objcopy -O srec  ${ProjName}.x  ${ProjName}.mot & arm-none-eabi-objcopy -O binary ${ProjName}.x  ${ProjName}.bin`
7. Click 'OK'.

## 4.4 Programming a User Application Program

Applications can be programmed onto the board by using the batch file 'Program_QSPI_Loader_Application.bat' found in the 'scripts' folder of the project. The batch (.bat) file calls the Segger jLink executable and passes a command file. This command file selects the debugger and device to select, and performs the programming operation for the binary file specified in the command.

Note that the 'exec SetSkipProgOnCRCMatch=0' instruction in the command files checks if the boot loader program to be loaded matches the existing boot loader code in the QSPI device. The 'loadbin' instruction will skip programming if they match. A power cycle to the Stream it! - RZ is required following a successful loading of the user application.

Alternatively, the application can be programmed onto the device using e$^2$ studio. When doing this please change the Debug Configuration connection setting for 'Reset before run' to 'Yes' to allow the Boot Loader to operate.

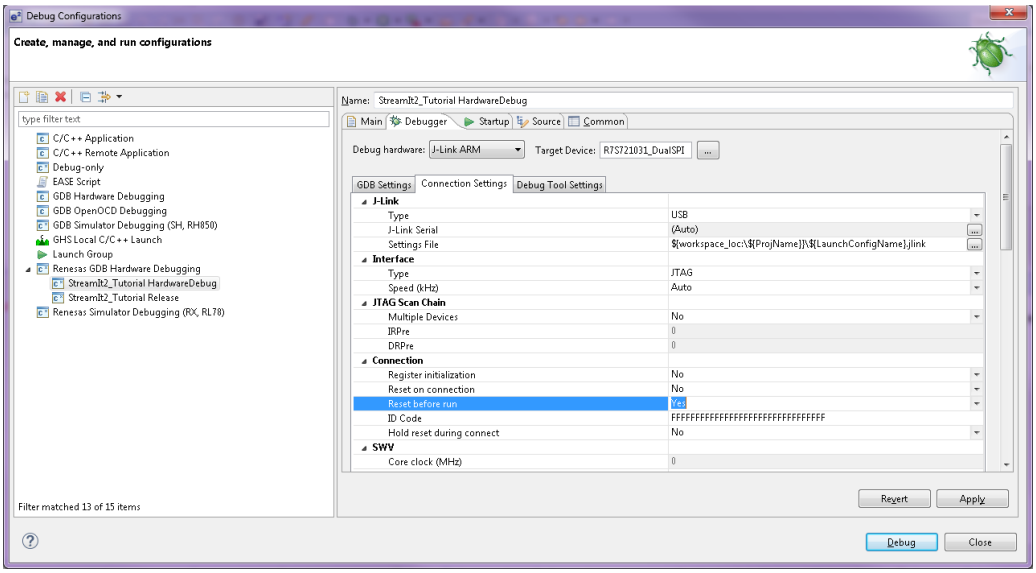


**Figure 8 The ‘Reset before run’ setting**

# 5. Further Reading

**Technical Support**

For details on how to use e$^2$ studio, refer to the help file by opening e$^2$ studio, then selecting Help > Help Contents from the menu bar.


For information about the RZA1L series microcontrollers refer to the RZA1L Group Hardware Manual.

**Technical Contact Details**

***Please refer to the contact details listed in section 5 of the Stream it! - RZ* "Quick Start Guide" (r12qs0013eg0100-rza1lu.pdf).**

Renesas Electronics Website:
https://www.renesas.com/

Inquiries:
https://www.renesas.com/contact/

This product's homepage, where additional documentation and source code can be found, is located at:
https://www.renesas.com/en-eu/solutions/key-technology/human-interface/rz-stream-it.html

## Revision History

| | | Description | |
|---|---|---|---|
| **Rev.** | **Date** | **Page** | **Summary** |
| 1.00 | Mar 21, 2017 | All | Original release |
| 2.00 | Jan 25, 2018 | -- | Code change. Bootloader optimized for DDR |
| 3.00 | May 10, 2018 | -- | Bootloader code change for alternative QSPI timings<br>Refer to R12TU0053EG Release Note for information |

## 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. 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 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. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment 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 moment 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 moment 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 moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has 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. Moreover, 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.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the 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 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

## SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics America Inc.**
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

**Renesas Electronics Canada Limited**
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338




Colophon 7.0