RX Family

Using the Simple Flash API for RX without the r_bsp Module

R01AN1890EU0100
Rev.1.00
Feb 25, 2014

Introduction

Starting with version 2.20 of the Simple Flash API for RX the Renesas Board Support Package (r_bsp) was introduced as a software dependency. The r_bsp provides many features to the user and is a requirement to use any Firmware Integration Technology (FIT) software module. The Simple Flash API for RX uses the r_bsp to determine information about the MCU being used and to ensure that only one flash operation is ongoing at any given time. Though not recommended, some users will wish to use the Simple Flash API for RX without the r_bsp and this document describes the steps that will need to be performed to do so.

Target Device

The following is a list of devices that are currently supported by r_bsp and the Simple Flash API for RX:

- **RX210 Group**
- **RX610 Group**
- **RX621, RX62N, RX62T, RX62G Groups**
- **RX630, RX631, RX63N, RX63T Groups**

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Related Documents

This document will assume that the user is familiar with the basic concepts of the Simple Flash API for RX, FIT, and the r_bsp. If you need more information on these topics please read the documents listed below.

- Simple Flash API for RX600 & RX200 (R01AN0544EU0240)
- Firmware Integration Technology User's Manual (R01AN1833EU0100)
- Board Support Package Module Using Firmware Integration Technology (R01AN1685EU0230)

Contents

1. Overview

By default the Simple Flash API for RX (Flash API for short) is intended to be used as shown in Figure 1. In this setup the user's code has the ability to use both the Flash API and the r_bsp. The Flash API requires the r_bsp to be present.

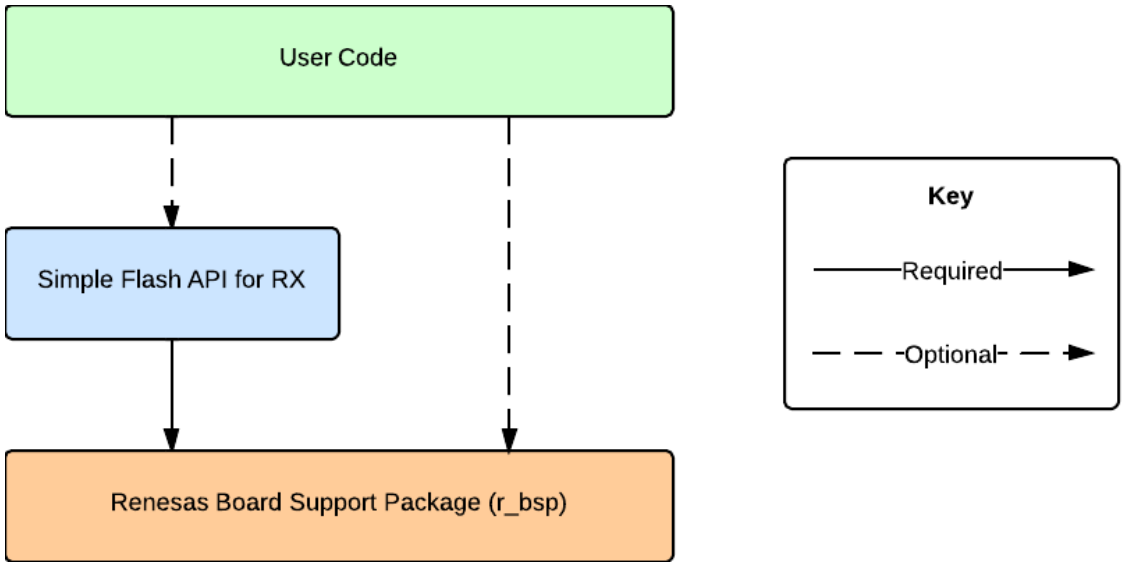

Figure 1: Default Project Setup

In order for the Flash API to work without the r_bsp the user must replace the r_bsp Module with their own custom board support package (BSP) code. In this document the user's custom BSP code will be referred to as *u_bsp* (user BSP). The u_bsp must provide the same information and API calls to the Flash API as provided by the r_bsp. If some r_bsp features are not used by the user then they may choose to omit that functionality from their u_bsp.

This document will only cover creating a u_bsp that supports the Flash API. If the user's code uses the r_bsp, or if other FIT Modules are used, then the user will be responsible for implementing that functionality in their u_bsp.

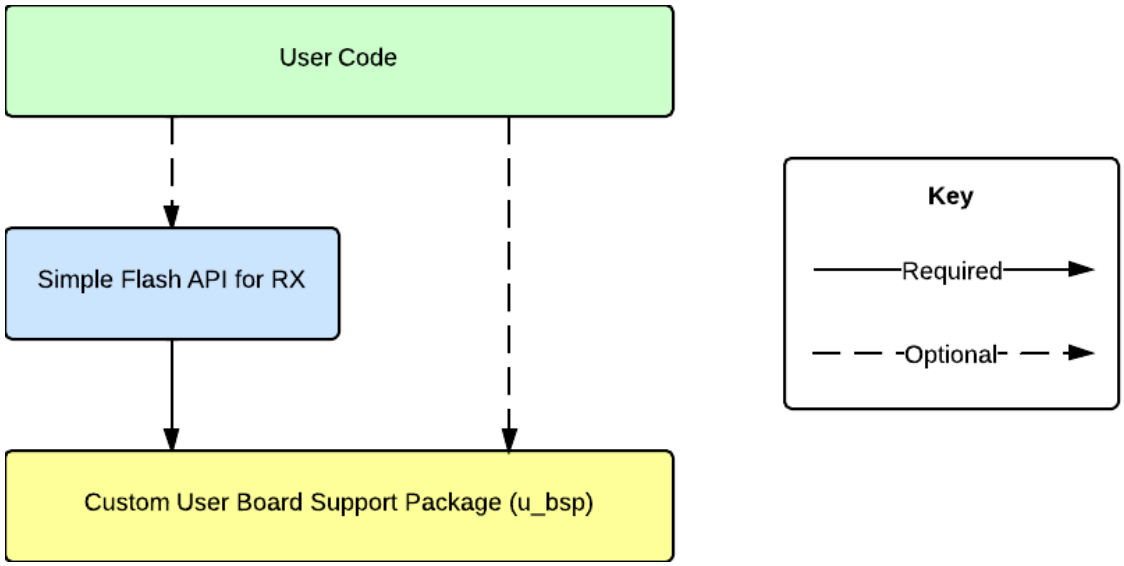

Figure 2: Project with Custom User BSP

2. Creating a u_bsp

Each subsection below will detail the pieces of the r_bsp that will need to be implemented in the u_bsp. The box below shows the default files for the RDKRX63N r_bsp. The highlighted files are the ones that will be mentioned in this section. The example below is for the RDKRX63N but the same files are available for other RX boards and MCUs.

```
r_bsp
 |    platform.h
 |    readme.txt
 |
 ├──────board
 |     └──────rdkrx63n
 |             dbsct.c
 |             hwsetup.c
 |             hwsetup.h
 |             lowlvl.c
 |             lowsrc.c
 |             lowsrc.h
 |             resetprg.c
 |             r_bsp.h
 |             r_bsp_config_reference.h
 |             sbrk.c
 |             vecttbl.c
 |             vecttbl.h
 |             yrdkrx63n.h
 |
 ├──────doc
 |        r01an1685eu0230_rx.pdf
 |
 └──────mcu
      ├──────all
      |        r_bsp_common.c
      |        r_bsp_common.h
      |        r_typedefs.h
      |
      └──────rx63n
           |    cpu.c
           |    cpu.h
           |    locking.c
           |    locking.h
           |    mcu_info.h
           |    mcu_init.c
           |    mcu_init.h
           |    mcu_interrupts.c
           |    mcu_interrupts.h
           |    mcu_locks.c
           |    mcu_locks.h
           |
           └──────register_access
                  iodefine.h
```

2.1 MCU Information

The Flash API supports many RX MCUs. Across these MCUs the ROM and Data Flash specifications differ. The Flash API is designed such that it will automatically configure its own code at compile-time based upon which MCU is being used. The Flash API gets information about the MCU from the r_bsp. More specifically, this information is found in the file *mcu_info.h*. This header file uses information provided by the user in *r_bsp_config.h* to create macros that detail the MCU.

Below is an excerpt from the *mcu_info.h* header file for the RX63N. The macro BSP_CFG_MCU_PART_PACKAGE is defined in *r_bsp_config.h* and set by the user. In this example, if the user set BSP_CFG_MCU_PART_PACKAGE to 3 in their *r_bsp_config.h* file then *mcu_info.h* would define these two macros:

1. BSP_PACKAGE_LQFP144 defined as '1'
2. BSP_PACKAGE_PINS defined as '144'

```
/* Package. */
#if   BSP_CFG_MCU_PART_PACKAGE == 0x0
    #define BSP_PACKAGE_LQFP176     (1)
    #define BSP_PACKAGE_PINS        (176)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x1
    #define BSP_PACKAGE_LFBGA176    (1)
    #define BSP_PACKAGE_PINS        (176)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x2
    #define BSP_PACKAGE_TFLGA177    (1)
    #define BSP_PACKAGE_PINS        (177)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x3
    #define BSP_PACKAGE_LQFP144     (1)
    #define BSP_PACKAGE_PINS        (144)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x4
    #define BSP_PACKAGE_TFLGA145    (1)
    #define BSP_PACKAGE_PINS        (145)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x5
    #define BSP_PACKAGE_LQFP100     (1)
    #define BSP_PACKAGE_PINS        (100)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x6
    #define BSP_PACKAGE_TFLGA100    (1)
    #define BSP_PACKAGE_PINS        (100)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x7
    #define BSP_PACKAGE_LQFP80      (1)
    #define BSP_PACKAGE_PINS        (80)
#elif BSP_CFG_MCU_PART_PACKAGE == 0x8
    #define BSP PACKAGE LQFP64      (1)
    #define BSP PACKAGE PINS        (64)
#elif BSP CFG MCU PART PACKAGE == 0x9
    #define BSP PACKAGE LQFP48      (1)
    #define BSP PACKAGE PINS        (48)
#else
    #error "ERROR - Unknown package chosen in r bsp config.h"
#endif
```

There are many ways the user can bring this functionality to the u_bsp. Here are 2 examples:

1. Copy the *r_bsp_config.h* and *mcu_info.h* files from the r_bsp to the u_bsp. Some options in *r_bsp_config.h* will be ignored due to the features being removed in the u_bsp, and others will remain valid. The user will still need to fill out the information about their MCU in *r_bsp_config.h*. After that, the *mcu_info.h* file will work just as it did in the r_bsp.
2. Copy the *mcu_info.h* file from the r_bsp to the u_bsp. Go through the *mcu_info.h* file and examine all of the preprocessor conditional code (i.e. #if, #elif, #endif). For each group of conditional statements, find the macros that apply to your MCU, and delete the rest (i.e. same macro name but different definition). Please note that all macros that are defined in *mcu_info.h* must still be available to the Flash API. Using the example excerpt from

before, if we used this method then the code in *mcu_info.h* would now look like this:

```
/* Package. */
#define BSP_PACKAGE_LQFP144     (1)
#define BSP_PACKAGE_PINS        (144)
```

Now go through the macros that are left and replace any macros within macros with hard coded values. For example, this:

```
/* FlashIF clock speed in Hz. */
#define BSP_FCLK_HZ             (BSP_SELECTED_CLOCK_HZ / BSP_CFG_FCK_DIV)
```

would be replaced with this:

```
/* FlashIF clock speed in Hz. */
#define BSP_FCLK_HZ             (48000000)
```

2.2 Atomic Locking

The Flash API uses the atomic locking functions of the r_bsp to protect against reentrancy. Using this feature ensures that two flash operations can never be active at the same time. There are two C source files and 2 accompanying header files that are associated with the atomic locking feature: *locking.c*, *locking.h*, *mcu_locks.c*, and *mcu_locks.h*. The *locking.c* and *locking.h* files implement the atomic locking functionality. The *mcu_locks.c* and *mcu_locks.h* files allocate locks for all the peripherals on the MCU. The globally defined locks in *mcu_locks.h* are provided by the r_bsp so that multiple FIT Modules can use the same peripheral safely. One of the globally defined locks is for the MCU's flash (i.e. BSP_LOCK_FLASH) and is used by the Flash API.

The recommended way for the user to implement atomic locking in their u_bsp is to copy over the four files listed earlier. The locking code is very small and should not impact the user's project. If the Flash API and r_bsp are the only FIT Modules the user has in their project then they can cut down the RAM requirements for the locking code by removing all unneeded global locks. The user would do this by modifying the *mcu_lock_t* enumerator in *mcu_locks.h* to only include the flash lock and the sizing entry. After making this modification the Flash API code will work as usual.

2.3 platform.h

Any file that uses the r_bsp must include the *platform.h* header file. The Flash API includes *platform.h* to get access to MCU information macros, the atomic locking API, MCU registers, and C99 standard types. The user will need to create their own *platform.h* in their u_bsp that will satisfy the requirements of the Flash API. If the user's toolchain does not support C99 then instead of including the *stdint.h* and *stdbool.h* header files the user could instead use *r_typedefs.h* which can be found in the *r_bsp >> mcu >> all* directory. Some C89 compilers still offer *stdint.h* and *stdbool.h* in which case *r_typedefs.h* will not need to be used.

3. Demo

This section will detail the steps taken to make a u_bsp for a RX63N based board following the subsections from Section 2. The specifics of the RX63N we are using are:

- 80-pin LQFP Package
- CAN included
- 512KB ROM, 64KB RAM, 32KB Data Flash
- 12MHz external crystal, PLL at 192MHz, FCLK at 48MHz, PLL is the selected clock

For this demo the following code is used:

- Renesas Board Support Package (r_bsp) FIT Module v2.30 (R01AN1685EU)
- Simple Flash API for RX600 & RX200 v2.40 (R01AN0544EU)

The Renesas RX v2.01 toolchain was used when creating the demo. Before starting these steps the user should download the latest r_bsp and create a folder named u_bsp.

3.1 MCU Information Steps

1. In the r_bsp, open up the board folder of a board that uses your MCU. For the RX63N we could use the RDKRX63N or RSKRX63N board folders. In this example we will use the RDKRX63N board for our reference. Copy the file *r_bsp >> board >> rdkrx63n >> r_bsp_config_reference.h* to the root of your u_bsp folder and rename it to *r_bsp_config.h*.
2. In the r_bsp, open up the MCU folder for your MCU. In this example we will use the *r_bsp >> mcu >> rx63n* folder. Copy the file *r_bsp >> mcu >> rx63n >> mcu_info.h* to the root of your u_bsp folder.
3. Open up your newly copied *r_bsp_config.h* file and review all of the macros that start with 'BSP_CFG_MCU_PART_'. The comments above each macro explain how to set the macro. If the default macro setting is not correct for MCU, change it to the appropriate value. Using the RX63N MCU described earlier our settings would be the following:

```
/* Package type. Set the macro definition based on values below:
   Character(s) = Value for macro = Package Type/Number of Pins/Pin Pitch
   FC           = 0x0             = LQFP/176/0.50
   BG           = 0x1             = LFBGA/176/0.80
   LC           = 0x2             = TFLGA/177/0.50
   FB           = 0x3             = LQFP/144/0.50
   LK           = 0x4             = TFLGA/145/0.50
   FP           = 0x5             = LQFP/100/0.50
   LA           = 0x6             = TFLGA/100/0.50
   FN           = 0x7             = LQFP/80/0.50
   FM           = 0x8             = LQFP/64/0.50
   FL           = 0x9             = LQFP/48/0.50
*/
#define BSP_CFG_MCU_PART_PACKAGE        (0x7)

/* Whether CAN is included or not.
   Character(s) = Value for macro = Description
   C            = false           = CAN not included
   D            = true            = CAN included
   E            =                 = 3V included (RX63T). Ignore.
*/
#define BSP_CFG_MCU_PART_CAN_INCLUDED   (true)

/* ROM, RAM, and Data Flash Capacity.
   Character(s) = Value for macro = ROM Size/Ram Size/Data Flash Size
   E            = 0xE             = 2MB/128KB/32KB
   D            = 0xD             = 1.5MB/128KB/32KB
   B            = 0xB             = 1MB/128KB/32KB
   A            = 0xA             = 768KB/128KB/32KB
   8            = 0x8             = 512KB/64KB/32KB
```

```
   7            = 0x7             = 384KB/64KB/32KB
   6            = 0x6             = 64KB/8KB/8KB
   5            = 0x5             = 48KB/8KB/8KB
   4            = 0x4             = 32KB/8KB/8KB
   0            = 0x0             = 0/128KB/0
*/
#define BSP_CFG_MCU_PART_MEMORY_SIZE    (0x8)

/* Group name.
   Character(s) = Value for macro = Description
   30           = 0x0             = RX630 Group
   31           = 0x1             = RX631 Group
   3N           = 0x2             = RX63N Group
   3T           = 0x3             = RX63T Group
*/
#define BSP_CFG_MCU_PART_GROUP          (0x2)

/* Series name.
   Character(s) = Value for macro = Description
   56           = 0x0             = RX600 Series
*/
#define BSP_CFG_MCU_PART_SERIES         (0x0)

/* Memory type.
   Character(s) = Value for macro = Description
   F            = 0x0             = Flash memory version
   S            = 0x1             = ROMless version
*/
#define BSP_CFG_MCU_PART_MEMORY_TYPE    (0x0)
```

4. Also in *r_bsp_config.h*, find the macro BSP_CFG_CLOCK_SOURCE. This is the first macro in a series that defines the clock setup for your MCU. Set these macros to reflect the clock setup for your system. The Flash API needs the frequency of the clock supplied to the flash for configuration purposes. On RX610 and RX62x MCUs this is the PCLK. On RX63x and RX200 MCUs this is the FCLK. Here are the macros filled out for our demo setup. These macro settings will end up setting BSP_FCLK_HZ in *mcu_info.h* to 48MHz. Please note that once these macros are set the user can also use the other clock macros defined in *mcu_info.h* in their own code if needed.

```
/* Clock source select (CKSEL).
   0 = Low Speed On-Chip Oscillator  (LOCO)
   1 = High Speed On-Chip Oscillator (HOCO)
   2 = Main Clock Oscillator
   3 = Sub-Clock Oscillator
   4 = PLL Circuit
*/
#define BSP_CFG_CLOCK_SOURCE            (4)

/* Clock configuration options. */

/* XTAL - Input clock frequency in Hz */
#define BSP_CFG_XTAL_HZ                 (12000000)

/* PLL Input Frequency Divider Select (PLIDIV).
   Available divisors = /1 (no division), /2, /4
*/
#define BSP_CFG_PLL_DIV                 (1)

/* PLL Frequency Multiplication Factor Select (STC).
   Available multipliers = x8, x10, x12, x16, x20, x24, x25, x50
*/
#define BSP_CFG_PLL_MUL                 (16)
```

```
/* System Clock Divider (ICK).
   Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
*/
#define BSP_CFG_ICK_DIV                 (2)

/* Peripheral Module Clock A Divider (PCKA).
   Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
*/
#define BSP_CFG_PCKA_DIV                (4)

/* Peripheral Module Clock B Divider (PCKB).
   Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
*/
#define BSP_CFG_PCKB_DIV                (4)

/* External Bus Clock Divider (BCK).
   Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
*/
#define BSP_CFG_BCK_DIV                 (8)

/* Flash IF Clock Divider (FCK).
   Available divisors = /1 (no division), /2, /4, /8, /16, /32, /64
*/
#define BSP_CFG_FCK_DIV                 (4)

/* IEBUS Clock Divider Select.
   Available divisors = /1 (no division), /2, /4, /6, /8, /16, /32, /64
*/
#define BSP_CFG_IEBCK_DIV               (8)

/* USB Clock Divider Select.
   Available divisors = /3, /4
*/
#define BSP_CFG_UCK_DIV                 (4)

/* Configure BCLK output pin (only effective when external bus enabled)
   Values 0=no output, 1 = BCK frequency, 2= BCK/2 frequency */
#define BSP_CFG_BCLK_OUTPUT             (0)

/* Configure SDCLK output pin (only effective when external bus enabled)
   Values 0=no output, 1 = BCK frequency */
#define BSP_CFG_SDCLK_OUTPUT            (0)
```

3.2 Atomic Locking Steps

1. In the r_bsp, open up the MCU folder for your MCU. In this example we will use the *r_bsp* >> *mcu* >> *rx63n* folder. Copy the following files from the *r_bsp* >> *mcu* >> *rx63n* directory to the root of your u_bsp folder:
   1.1. *locking.c*
   1.2. *locking.h*
   1.3. *mcu_locks.c*
   1.4. *mcu_locks.h*
2. In this demo the Flash API is the only FIT Module that is being used. To reduce RAM usage we will delete all locks except the one for the flash. Open up the *mcu_locks.h* file that you just copied to your u_bsp folder.

3. Modify the *mcu_lock_t* enumerator so that only the BSP_LOCK_FLASH and BSP_NUM_LOCKS members remain.

```
typedef enum
{
    BSP_LOCK_FLASH = 0,
    BSP_NUM_LOCKS       //This entry is not a valid lock. It is used for
                        //sizing g_bsp_Locks[] array below. Do not touch!
} mcu_lock_t;
```

3.3 platform.h Steps

1. We are going to replace all of the contents of the *platform.h* file so there is no real benefit of using the existing file. Create a file named *platform.h* in your u_bsp directory.
2. Add #includes for the *stdint.h* and *stdbool.h* header files.
3. Add a #include for *machine.h*. This header file is provided by the Renesas RXC toolchain and provides intrinsic functions. One of these functions is xchg() which is used to implement atomic locking.
4. Add a #include for your register access file. By default this is usually named *iodefine.h* for RX MCUs.
5. Add #includes for all of the header files in your u_bsp directory except *platform.h*. Make sure to put the #include for *r_bsp_config.h* before the others.
6. Open up the file *r_bsp* >> *mcu* >> *all* >> *r_bsp_common.h*. Find the R_BSP_VERSION_MAJOR and R_BSP_VERSION_MINOR macros. Copy these macros to your *platform.h* file. An example of what your file should look like is shown below. Preprocessor conditionals protecting against multiple inclusion were added at the top and bottom.

```
#ifndef PLATFORM_CUSTOM_H
#define PLATFORM_CUSTOM_H

/***********************************************************************
Includes   <System Includes> , "Project Includes"
***********************************************************************/
#include <stdint.h>
#include <stdbool.h>
/* Access to intrinsic functions. */
#include <machine.h>
/* Register access. */
#include "iodefine.h"
/* Not all configuration items in r_bsp_config.h are being used. */
#include "r_bsp_config.h"
#include "mcu_info.h"
/* Locking used by some FIT Modules. */
#include "locking.h"
#include "mcu_locks.h"

/***********************************************************************
Macro definitions
***********************************************************************/
/* This takes care of FIT Modules that check for the version of the r_bsp */
#define R_BSP_VERSION_MAJOR  (2)
#define R_BSP_VERSION_MINOR  (30)

#endif /* PLATFORM_CUSTOM_H */
```

3.4 Project Configuration

1. Setup your project so that the u_bsp source files will be compiled.
2. Add an include path to your u_bsp folder.
3. Make sure that an include path is setup so that *iodefine.h* can be found by the Flash API.
4. Setup the Flash API as you normally would following the directions in its documentation.
5. You can now use the Flash API as you normally would.

Website and Support

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

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

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Feb 25, 2014	—	First Release

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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 an MPU or MCU 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of 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. 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 should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended 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; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor 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 damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may 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 should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be 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, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

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

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

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 906, 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 Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0