RX Family

Debugging with RI600/4

R20AN0092ES0100
Rev.1.00
Oct 01, 2010

Introduction

Real-time operating system was often employed with the notion of reducing development time, improving response time and integrity of the embedded system. However, new problems are created with the improper use of RTOS. Existing standard debuggers can only display information such as local variables, registers and memory contents. To use standard debugger to monitor the transition, communications and synchronizations of the multiple tasks, semaphore, event flag and data queues etc would require one to have in-depth knowledge in the mechanisms of the RTOS. RTOS aware tools are therefore required to enhance users' ability in debugging such problems.

This document explains how Renesas real-time OS aware debugging utility can be used to debug these problems.

Target Device

Applicable MCU: RX Family

Contents

1. Guide in using this Document

This document discusses few common problems faced in using real-time operating system (RTOS) and explains how the various Renesas OS aware debugging facilities can be employed to identify the problems.

Table 1 Explanation of Document Topics

Topic	Objective	Pre-requisite
Common Debugging Issues of RTOS	Provide listing of common issues with multi-task debugging. Explanations on how Renesas real-time OS aware debugging utility can be used for identifying the issues are also given	Knowledge in embedded system debugging, RI600/4 and E100 emulator
Reference Documents	Listing of documents that equip users with knowledge in the pre-requisite requirements	None

2. Common Debugging Issues of RTOS

Often, an RTOS user will encounter the following issues with multi-task debugging:

1. Deadlock
2. Stack overflow
3. Memory corruption
4. Memory leaks
5. Task starvation

These problems usually arise due to unfamiliarity with resources of RTOS used, incomplete execution of software development process and inadequate knowledge on limitations/specifications of device used.

2.1 Deadlock

2.1.1 What is it?

Deadlock is a special case of nested resource locks. It occurs when two or more tasks got “locked” up as they failed to gain access to one another’s resources required to resume their execution. Semaphore, a RTOS object that allows tasks to lock/unlock resources facilitates the occurrence of a deadlock when it is improperly used. Figure 1 depicts an example of a deadlock occurrence.

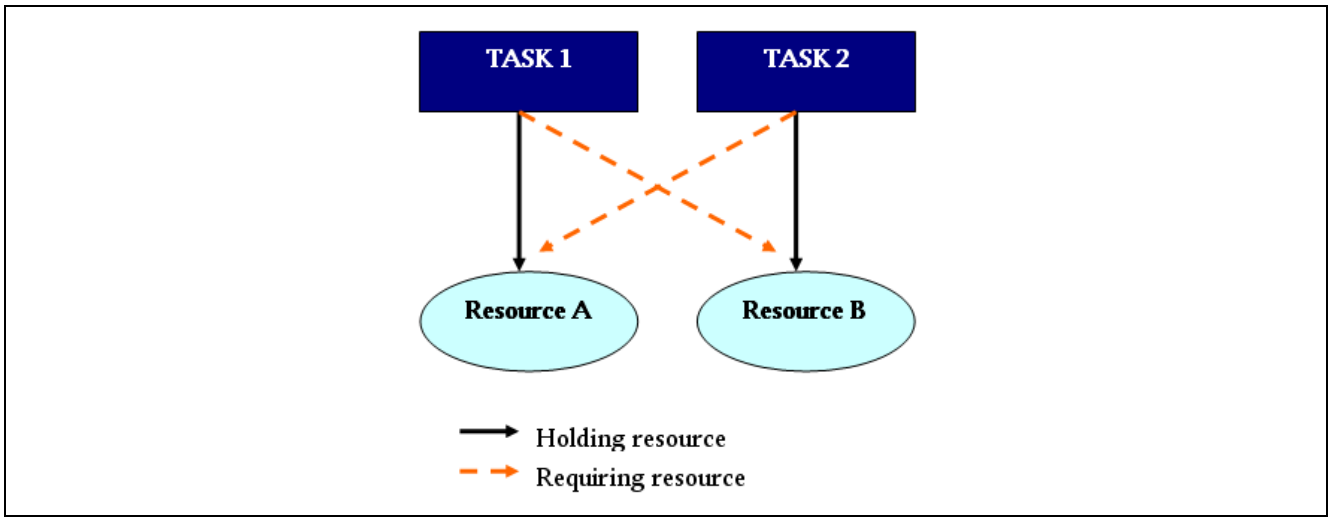

Figure 1 A Deadlock Example

In the example, “TASK 1” and “TASK 2” are engaged in a deadlock. “TASK 2” locked “TASK 1” as it is holding on to “Resource A” that “TASK 1” requires to process. Likewise, “TASK 1” is holding on to “Resource B” required by “TASK 2”, thereby locks it.

2.1.2 Why did it happen?

For a deadlock to occur, four conditions are required.

- Resources are mutually exclusive. When resources are mutually exclusive, only one task can acquire it at any one time. Other tasks that request for the resources will be placed in WAITING state.
- Tasks are allowed to hold on to resource/s, request and wait for other resource/s at the same time. Competition among multiple tasks for individual resource will stifle and it increases the likelihood of tasks being interlocked.
- Tasks cannot be forced to release resources. This restriction eliminates the possibility of releasing the interlocks among tasks.
- Circular wait condition where each task waits for a resource currently in the holding of the next task.

2.1.3 How to Identify?

In the occurrence of a deadlock, tasks involved will be kept in a WAITING state. To identify its occurrence, user may monitor the status of individual task by referring to the OS object window of Renesas real-time OS aware debugging extension as shown in Figure 2 and Figure 3. Figure 2 of task analyze window displays the execution time of respective tasks. As can be seen, execution duration of tasks 1 and 2 are very short as compared to tasks 3 and 4. Therefore, user will be able to identify the need to investigate both tasks 3 and 4.

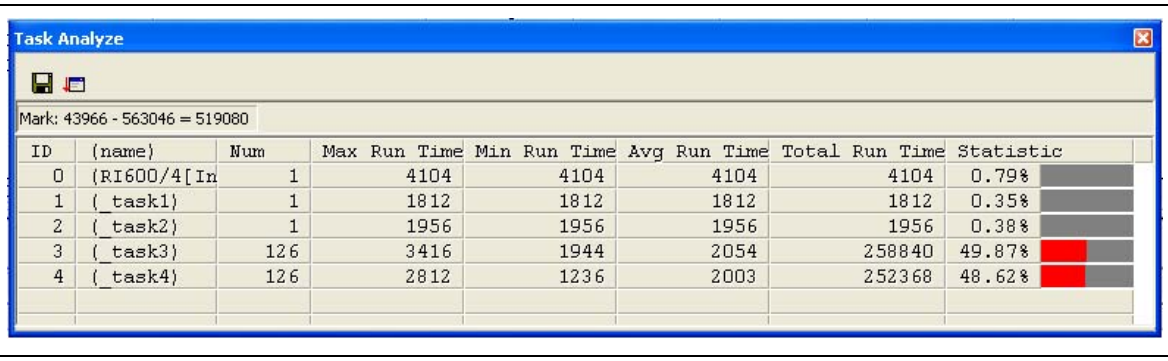

Figure 2 Task Analysis Window

Figure 3 shows the OS object window for task object. The OS object window displays the status of individual task. From Figure 3, user will be able to identify the reason for the low execution duration of tasks 1 and 2. Both tasks are in a "WAITING" for respective semaphore resource. Tasks 1 and 2 are therefore, engaged in a deadlock.

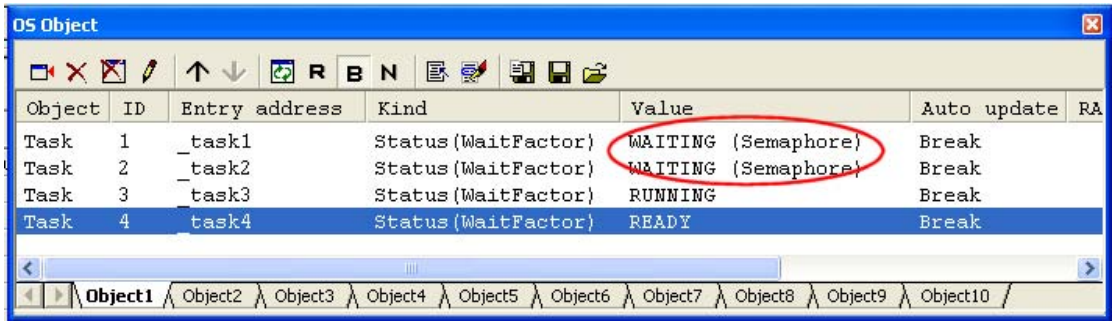

Figure 3 OS Object Window (Task Objects)

2.2 Stack Overflow

2.2.1 What is it?

In RI600/4, each task is being allocated with a fixed stack size statically in the configurator file (Figure 4). This block of memory denotes the worst-case memory consumption of the individual task during run-time. A stack overflow occurs when the stack exceeds its capacity and no more data can be inserted into it. When stack overflow occurred, application instability or even breakdown will happened.

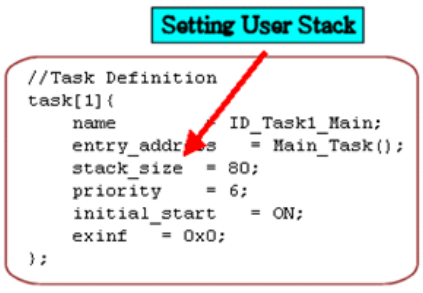

Figure 4 Defining User Stack in Configurator File

2.2.2 Why did it happen?

Stack overflow happened when the required stack size exceeded the allocated stack size. An insufficient stack allocation is largely due to two common causes.

The stack is a region of memory where local variables are created. Therefore, sizes of the local variables declared are directly proportional to the amount of stack memory consumed. When local variables that are far too large are created, stack overflow will occur. Figure 5 shows an example of a large local variable that consume more stack memory than being allocated for a task.

```
24  //Task Definition
25  task[]{
26      name            = ID_TASK1;
27      entry_address   = task1();
28      initial_start   = ON;
29      stack_size      = 512;
30      priority        = 1;
31  //  stack_section   = STK1;
32      exinf           = 1;
33  };
```

Task1 allocated 512 bytes stack size

```
15  /*****************************
16   task #1
17  *****************************
18  void task1( VP_INT exinf)
19  {
20      ER  lErcd, lNumber;
21
22      double variable_y[200];
23
```

variable_y require 200 X 4 bytes
= 800 bytes of memory

Figure 5 Stack Overflow due to Large Local Variable

The other major cause of a stack overflow results from infinite recursion. Infinite recursion is a design error that causes the functions to be executed in an infinite loop as shown in Figure 6.

```
162  void check_output(int var)
163  {
164    int output;
165    ...
166    ...
167    get_output(output);
168  }
169
170  void get_output(int var)
171  {
172    int output;
173    ...
174    ...
175    check_output(output);
176  }
```

Figure 6 Stack Overflow due to Infinite Recursion

2.2.3 How to identify?

In Renesas E100 emulator, task stack access violation facility is provided. By activating this option, error is detected when one task attempts writing to the task stack of another task. Users are able to define the actions to be taken when a task stack access violation is detected. Figure 7 shows the setting of hardware break as the action for task stack violation.

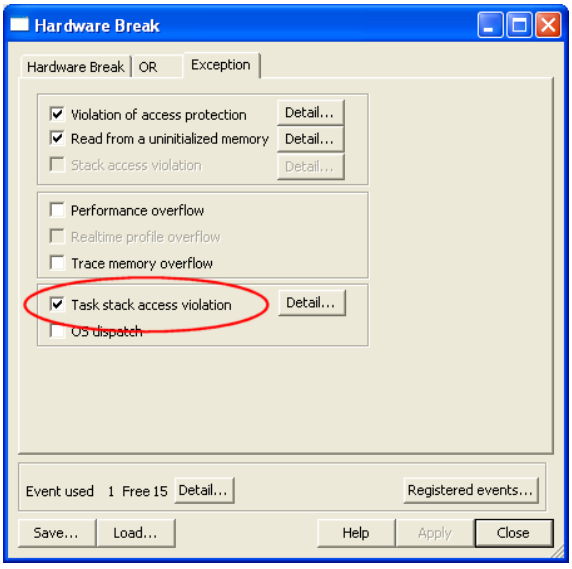

Figure 7 Setting Hardware Break for Stack Overflow Detection

Alternatively, user may refer to the Data Coverage Window for the access rate of individual task stack. If a particular task stack access rate hit 100%, stack overflow is likely to have occurred as shown in Figure 8.

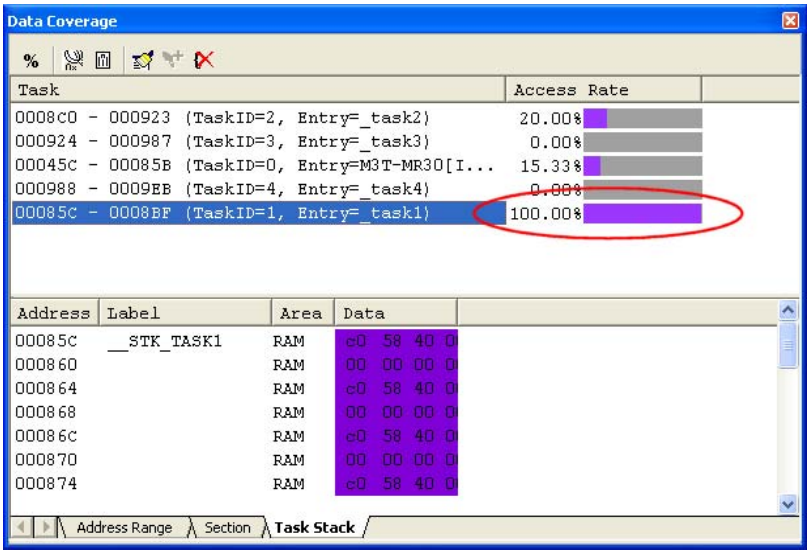

Figure 8 Data Coverage Window showing Task Stack Usage

2.3 Memory Corruption

2.3.1 What is it?

When contents of memory location are unintentionally modified, memory corruption is said to have occurred. When the corrupted memory contents are accessed, it will leads to bizarre application behavior or even system crash.

2.3.2 Why did it happen?

Memory corruptions are largely a result of programming errors. Memory corruption errors may occur in one of the following ways.

- Accessing un-initialized memory that may lead to unpredictable application behavior
- Accessing out of bound memory. With the improper use of pointers, memory location outside of current task stack boundary may result in a system crash.
- Writing data beyond buffer capacity.

2.3.3 How to identify?

There are three types of memory protection available for users (Figure 9). The protection available includes:

- Violation of access protection
- Reading from non-initialize memory
- Stack Access violation

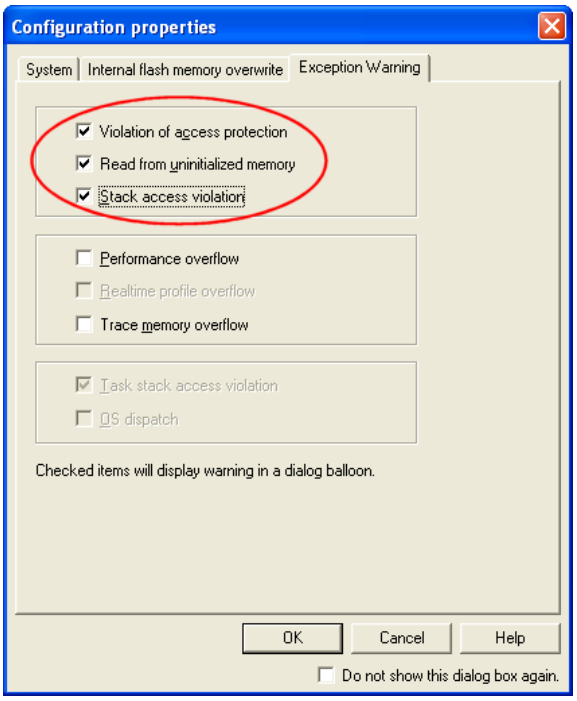

Figure 9 Configuration Settings for Memory Protection

Violation of access protection allows the detection of an incorrect access to an unused area or writing to a ROM area.

Reading from a non-initialized area can also be detected. The actions correspond to the detection can be in the form of a warning display or color highlight in the RAM Monitor window.

Stack access violation detects an error when one task attempts to write to the stack of another task.

2.4 Memory Leaks

2.4.1 What is it?

A memory leaks occurs when acquired memory is not released, that causes system to run out of memory eventually. It is one of the most common problems found in multi-task applications that involve high frequency of memory passing among the tasks. A memory leak can diminish the performance of the device as it reduces the amount of available memory. When too much memory leaks occur, part or whole device might stops working correctly which may eventually lead to a system failure due to memory exhaustion.

2.4.2 Why did it happen?

Each resource declared/used takes up space in memory. When an application runs, tasks can acquire memory or gain access to resources using various kernel objects (e.g. semaphore). If a task is incorrectly deleted, it might not get to release the acquired resources. This abrupt deletion of the operating task can result in a corrupt data structure, unreleased resource and/or inaccessible data structure (due to the unreleased resource) thereby resulting in memory leak.

Figure 10 illustrates an example of a memory leak due to abrupt termination of a task. In Figure 10 , task2 terminates task1 before it releases the acquired memory pool “ID_MPF1”. If the process repeats, device will eventually run out of memory for allocation.

```
28  void task1( VP_INT exinf)
29  {
30      ER  lErcd, lNumber;
31
32      lErcd = get_mpf(ID_MPF1, &p_blk1_1);
33
34      lErcd = sta_tsk(ID_TASK2,2);
35
36      lErcd = wai_sem(ID_SEM1);
37  }
38
39  void task2( VP_INT exinf)
40  {
41      ER  lErcd, lNumber;
42
43      lErcd = ter_tsk(ID_TASK1);
44
45      lErcd = sta_tsk(ID_TASK3,3);
46  }
```

Figure 10 Memory Leak

2.4.3 How to identify?

In the event of a memory leak, large part (if not all) of the memory are likely to be consumed for a duration of time. To identify this phenomenon, user may refer to the data coverage window to view the access rate of individual memory block as shown in Figure 11.

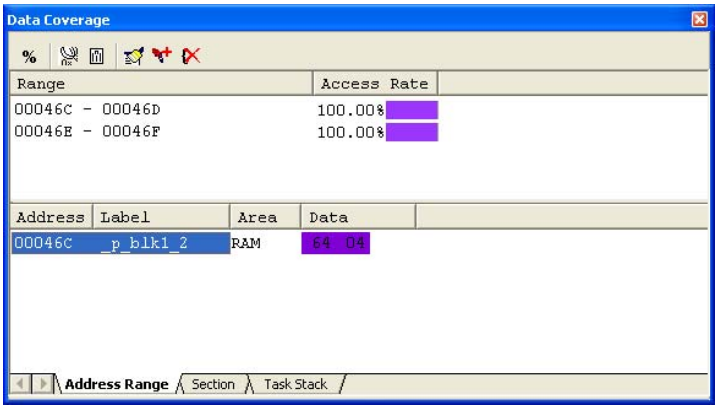

Figure 11 Data Coverage Window showing Memory Block Access Rate

To identify the task/s that results in the memory leak, user may refer to the track window. Figure 12 shows “Task1” is culprit for acquiring all the memory blocks.

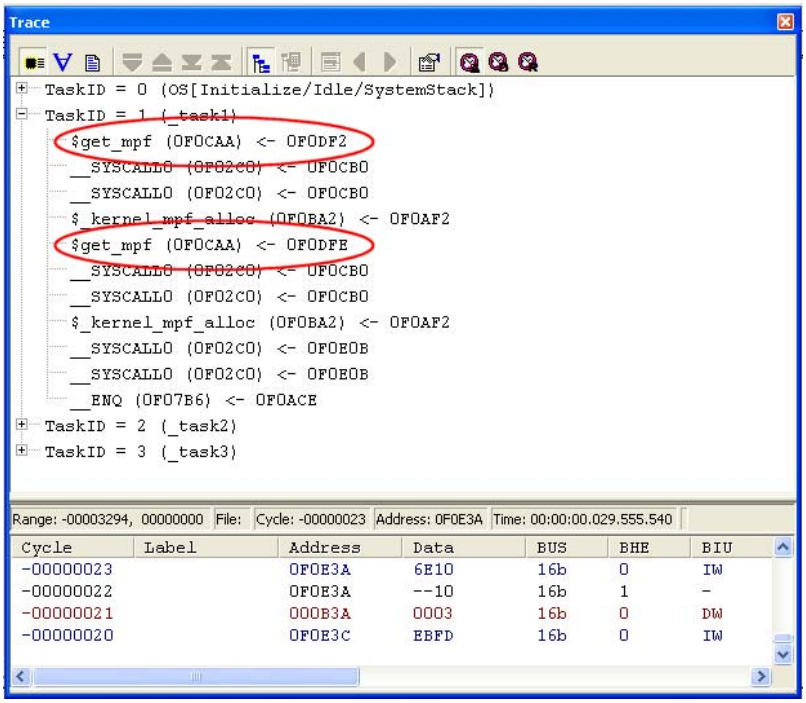

Figure 12 Data Coverage Window showing Memory Block Access Rate

2.5 Task Starvation

2.5.1 What is it?

In an application, every task is assigned a priority level and designed to be activated at an interval or within a time frame. When task/s is unable to activate due to other task/s hogging on to the processor time, task starvation happens.

Task starvation may results in intermittent and unexpected system behavior rather than hard failures.

2.5.2 Why did it happen?

For instance, a watchdog task at higher priority than other tasks can possibly result in starvations of other tasks if it is being activated at regular close intervals. Alternatively, deadlocks between tasks can result in task starvation as well. Task starvation is largely due to poor implementation of scheduling algorithm.

2.5.3 How to identify?

Using the conventional method of building, testing, locating and fixing to deal with task starvation is time consuming and expensive. Using real-time profile window of real-time OS aware debugging tool, real-time profile performance of individual task may be viewed. If a task's execution count, time, average and ratio fall below expectations, it can be suspected to be suffering from starvation. Figure 13 indicates "task4" was not executed (based on zero percentage under statistic). Thus, "task4" may be suspected to be suffering from task starvation.

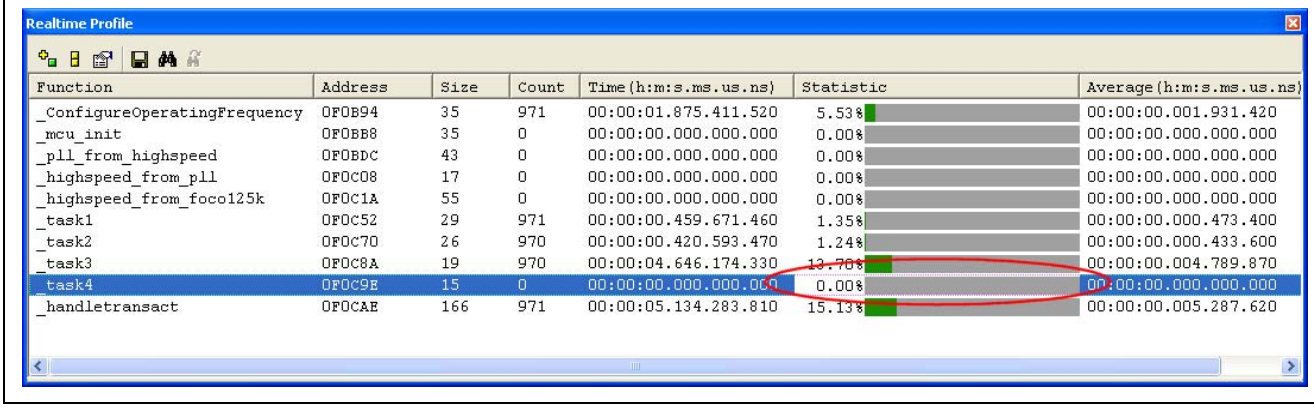

Figure 13 Real-time Profile Window showing Task Starvation

3. Reference Documents

User's Manual

- RI600/4 V.1.00 User's Manual
- RX Family Hardware Manual
- E100 Emulator User's Manual
- High-performance Embedded Workshop (Real-time OS aware debugging) Application Note

The latest version can be downloaded from the Renesas Electronics website

Website and Support

Renesas Electronics Website

- http://www.renesas.com/

Inquiries

- http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Oct.01.10	—	First edition issued

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 accord 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. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. 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.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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.
5. When exporting the 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. You should not use Renesas Electronics products or the 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. 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.
6. 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.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. 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 categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. 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 an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "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.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. 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.
9. 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 system 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. 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.
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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, 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.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

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: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

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.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0