To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

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.

SH2 Series

VDE Certified IEC60730 Self Test Code for SH2 Series MCU

INTRODUCTION

Today, as automatic electronic controls systems continue to expand into many diverse applications, the requirement of reliability and safety are becoming an ever increasing factor in system design.

For example, the introduction of the IEC60730 safety standard for household appliances requires manufactures to design automatic electronic controls that ensure safe and reliable operation of their products.

The IEC60730 standard covers all aspects of product design but Annex H is of key importance for design of Microcontroller based control systems. This provides three software classifications for automatic electronic controls:

1. Class A: Control functions, which are not intended to be relied upon for the safety of the equipment.
   Examples: Room thermostats, humidity controls, lighting controls, timers, and switches.

2. Class B: Control functions, which are intended to prevent unsafe operation of the controlled equipment.
   Examples: Thermal cut-offs and door locks for laundry equipment.

3. Class C: Control functions, which are intended to prevent special hazards
   Examples: Automatic burner controls and thermal cut-outs for closed.

Appliances such as washing machines, dishwashers, dryers, refrigerators, freezers, and Cookers / Stoves will tend to fall under the classification of Class B.

This Application Note provides guidelines of how to use flexible sample software routines to assist with compliance with IEC60730 class B safety standards. These routines have been certified by VDE Test and Certification Institute GmbH and a copy of the Test Certificate is available in the download package for this Application Note (See Note 1 below)

Although these routines were developed using IEC60730 compliance as a basis, they can be implemented in any system for self testing of Renesas MCUs.

These three key components are:
1. CPU
2. ROM / Flash memory
3. RAM

The software routines provided for CPU, ROM and RAM testing can be used after reset and also during the program execution. The end user has the flexibility of how to integrate these routines into their overall system design.

Note 1. This document is based on the European Norm EN60335-1:2002/A1:2004 Annex R, in which the Norm IEC 60730-1 (EN60730-1:2000) is used in some points. The Annex R of the mentioned Norm contains just a single sheet that jumps to the IEC 60730-1 for definitions, information and applicable paragraphs.

TABLE OF CONTENTS

1 TESTS

1.1 CPU TEST

This section describes the CPU tests routines. Reference IEC 60730: 1999+A1:2003 Annex H - Table H.11.12.1 CPU

The CPU test covers testing of CPU Registers by writing test values (like 0x55, 0xAA) into them and then reading them back. This can't be done using 'C' language so inline assembly code has been used.

These tests are testing such fundamental aspects of the CPU operation; the API functions do not have return values to indicate the result of a test. Instead the user of these tests must provide an error handling function with the following declaration:-

```
extern void CPU_Test_ErrorHandler(void);
```

This will be called by the CPU test if an error is detected. This function must not return back to the test code.

The test functions all follow the rules of register preservation following a C function call as specified in the Renesas tool chain manual. Therefore the user can call these functions like any normal C function without any additional responsibilities for saving register values beforehand.

Specifically CPU registers R0-R15, MACH, MACL, PR, GBR, VBR and SR are tested.
The source files 'CPU_Test.c' and 'CPU_Test_Coupling.c' provide the implementation of the CPU test using "C" language functions that contain the inline assembly. The source file 'CPU_Test.h' provides the interface to the function CPU test. The file 'MisraTypes.h' includes definitions of MISRA compliant standard data types.

IMPORTANT NOTE: Please keep the "Optimisation" option "OFF" for the 'CPU_Test.c' and 'CPU_Test_Coupling.c' files, to prevent modification of the test code.

The CPU test is split into several functions or a single function can be called to run all tests. See the Software API for details.

1.1.1 Software API

Syntax
`void CPU_TestAll(void)`
Description
Runs through all the tests detailed below in the following order:- 1. If using Coupling GPR Tests (*See below):- CPU_Test_GPRsCouplingPartA CPU_Test_GPRsCouplingPartB If not using Coupling GPR test:- CPU_Test_TestR0toR7 CPU_Test_TestR0toR7 2. CPU_Test_TestMACH_MACL_PR 3. CPU_Test_TestGBR_VBR_SR 4. CPU_Test_PC It is the calling function's responsibility to ensure no interrupts occur during this test. If an error is detected then external function 'CPU_Test_ErrorHandler' will be called. See the individual tests for a full description. *A #define 'USE_TestGPRsCoupling' in the code is used to select which functions will be used to test the General Purpose Registers.

Input Parameters	
NONE	N/A
Output Parameters	
NONE	N/A
Return Values	
NONE	N/A

Syntax
`void CPU_Test_GPRsCouplingPartA(void)`
Description
Tests general purpose registers R0 to R15.Coupling faults between the registers are detected. This is PartA of a complete GPR test, use function CPU_Test_GPRsCouplingPartB to complete the test. If an error is detected then external function '`CPU_Test_ErrorHandler`' will be called.

Input Parameters	
NONE	N/A
Output Parameters	
NONE	N/A
Return Values	
NONE	N/A

Syntax
`void CPU_Test_GPRsCouplingPartB(void)`
Description
Tests general purpose registers R0 to R15.Coupling faults between the registers are detected. This is PartB of a complete GPR test, use function CPU_Test_GPRsCouplingPartA to complete the test. If an error is detected then external function '`CPU_Test_ErrorHandler`' will be called.
Input Parameters
`NONE` — N/A
Output Parameters
`NONE` — N/A
Return Values
`NONE` — N/A

Syntax
`void CPU_Test_R0toR7(void)`
Description
Test registers R0 to R7. Registers are tested in pairs. For each pair of registers: 1. Write h'55555555 to both. 2. Read both and check they are equal. 3. Write h'AAAAAAAA to both. 4. Read both and check they are equal. If an error is detected then external function '`CPU_Test_ErrorHandler`' will be called
Input Parameters
`NONE` — N/A
Output Parameters
`NONE` — N/A
Return Values
`NONE` — N/A

Syntax
`void CPU_Test_R8toR15(void)`
Description
Test registers R8 to R15. Registers are tested in pairs. For each pair of registers: 1. Write h'55555555 to both. 2. Read both and check they are equal. 3. Write h'AAAAAAAA to both. 4. Read both and check they are equal. If an error is detected then external function '`CPU_Test_ErrorHandler`' will be called

Input Parameters	
NONE	N/A
Output Parameters	
NONE	N/A
Return Values	
NONE	N/A

Syntax
`void CPU_Test_MACH_MACL_PR(void)`
Description
Tests registers MACH, MACL and PR. This test assumes registers R0 to R7 are working. For each register:- 1. Write h'55555555 to. 2. Read back and check value equals h'55555555. 3. Write h'AAAAAAAA to. 4. Read back and check value equals h'AAAAAAAA. If an error is detected then external function CPU_Test_ErrorHandler will be called.

Input Parameters	
NONE	N/A
Output Parameters	
NONE	N/A
Return Values	
NONE	N/A

Syntax
`void CPU_Test_GBR_VBR_SR(void)`
Description
Tests registers GBR, VBR and SR. This test assumes registers R0 to R7 are working. For GBR and VBR:- 1. Write h'55555555 to. 2. Read back and check value equals h'55555555. 3. Write h'AAAAAAAA to. 4. Read back and check value equals h'AAAAAAAA. For SR not all bits are Read/Write so other test values have been chosen. If an error is detected then external function CPU_Test_ErrorHandler will be called.

Input Parameters	
NONE	N/A
Output Parameters	

NONE	N/A
Return Values	
NONE	N/A

Syntax
`void TestPCReg(void)`
Description
This function provides the Program Counter (PC) register test. This provides a confidence check that the PC is working. It tests that the PC is working by calling a function that is located in its own section so that it can be located away from this function, so that when it is called more of the PC Register bits are required for it to work. So that this function can be sure that the function has actually been executed it returns the inverse of the supplied parameter. This return value is checked for correctness. If an error is detected then external function '`CPU_Test_ErrorHandler`' will be called.

Input Parameters	
NONE	N/A
Output Parameters	
NONE	N/A
Return Values	
NONE	N/A

1.2 ROM / FLASH MEMORY TEST

This section describes the ROM / Flash memory test using CRC routines. Reference IEC 60730: 1999+A1:2003 Annex H – H2.19.4.1 CRC – Single Word.

CRC is a fault / error control technique which generates a single word or checksum to represent the contents of memory. A CRC checksum is the remainder of a binary division with no bit carry (XOR used instead of subtraction), of the message bit stream, by a predefined (short) bit stream of length n + 1, which represents the coefficients of a polynomial with degree n. Before the division, n zeros are appended to the message stream. CRCs are popular because they are simple to implement in binary hardware and are easy to analyse mathematically.

The ROM test can be achieved by generating a CRC value for the contents of the ROM and saving it.
During the memory self test the same CRC algorithm is used to generate another CRC value, which is compared with the saved CRC value. The technique recognises all one-bit errors and a high percentage of multi-bit errors.

The complicated part of using CRCs is if you need to generate a CRC value that will then be compared with other CRC values produced by other CRC generators. This proves difficult because there are a number of factors that can change the resulting CRC value even if the basic CRC algorithm is the same. This includes the combination of the order that the data is supplied to the algorithm, the assumed bit order in any look-up table used and the required order of the bits of the actual CRC value. This complication has arisen because big and little endian systems were developed to work together that employed serial data transfers where bit order became important. In a closed system, where the same implementation of CRC is used to check memory contents haven't changed, these complications can be ignored.

1.2.1 Algorithms implemented

1.2.1.1 CRC16-CCITT

The 16-bit CRC16-CCITT specification is:

- Polynomial = 0x1021 ($x^{16} + x^{12} + x^5 + 1$)
- Width = 16 bits
- Initial value = 0xFFFF
- Input data is NOT reflected
- Output CRC is NOT reflected
- No XOR operation is performed on the output CRC

Advantage of using the 16-bit CRC16-CCITT:

- It is a straightforward 16-bit CRC implementation in that it does not involve:
  - reflection of data
  - reflection of the final CRC value
- Starts with a non-zero initial value – leading zero bits cannot affect the CRC16 used by LHA, ARC, etc., because the initial value is zero.
- It requires no additional XOR operation after everything else is done.

1.2.1.1.1 CRC16-CCITT Software Calculation

The following three methods have been implemented:

1. No look-up table.
   This requires the least ROM but requires the most CPU cycles.
2. Large look-up table.
   This requires the most ROM (512 bytes) but requires the least CPU cycles.
3. Small look-up table.
   This provides a compromise between speed and size. It uses a 32 byte look-up table.

1.2.1.2 CRC16

The CRC16 specification implemented is:

- Polynomial = 0x8005 ($x^{16} + x^{15} + x^2 + 1$)
- Width = 16 bits
- Initial value = 0xFFFF
- Input data is NOT reflected
- Output CRC is reflected
- No XOR operation is performed on the output CRC

This algorithm has been implemented using a small static lookup table requiring only 32 bytes.

1.2.2 CRC Software API

All software is written in ANSI C.
'MisraTypes.h' includes definitions of MISRA-compliant standard data types.

The functions in the remainder of this section are used to calculate a CRC value. After calculating a new CRC value it should be compared against a reference CRC value that has been stored in ROM. To do this use the following function that is implemented in files CRC_Verify.h and CRC_Verify.c:

Syntax	
`bool_t CRC_Verify(const uint16_t ui16_NewCRCValue, const uint32_t ui32_AddrRefCRC)`	
Description	
This function compares a new CRC value with a reference CRC.	
Input Parameters	
`uint16_t ui16_NewCRCValue`	Value of calculated new CRC value.
`uint32_t ui32_AddrRefCRC`	Address where 16 bit reference CRC value is stored.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	Test result: TRUE = Passed, FALSE = Failed

1.2.2.1 CRC16-CCITT Software API

The implementation of the 'No look-up table' and the 'Large look-up table' share the same source files: CRC16-CCITT.h and CRC16-CCITT.c.
A compiler conditional '#define _USE_STATIC_TABLE' is used to select between the two.
The 'Small look-up table' is implemented in files:
CRC16_CCITT_Small_LT1Func.h and CRC16_CCITT_Small_LT1Func.c.

Syntax	
`uint16_t CRC16_CCITT(uint8_t* pui8_DataBuf, uint32_t ui32_DataBufSize)` `uint16_t CRC16_CCITT_Small_LT(uint8_t* pui8_DataBuf, uint32_t ui32_DataBufSize)`	
Description	
This function calculates the CRC16–CCITT.	
Input Parameters	
`uint8_t* pui8_DataBuf`	Pointer to start of data buffer / memory. This should be unsigned integer pointer to the data.
`Uint32_t ui32_DataBufSize`	Length of the data buffer / memory. This should be a 32-bit value.
Output Parameters	
`NONE`	N/A
Return Values	
`uint16_t`	16-bit calculated CRC16 CCITT value.

1.2.2.2 CRC16 Software API

This is implemented in files CRC16_Small_LT.h and CRC16_Small_LT.c.

Syntax	
`uint16_t CRC16_Small_LT(uint8_t* pui8_DataBuf, uint32_t ui32_DataBufSize)`	
Description	
This function calculates the CRC16 using a small look-up table.	
Input Parameters	
`uint8_t* pui8_DataBuf`	Pointer to start of data buffer / memory. This should be unsigned integer pointer to the data.
`Uint32_t ui32_DataBufSize`	Length of the data buffer / memory. This should be a 32-bit value.
Output Parameters	
`NONE`	N/A
Return Values	
`uint16_t`	16-bit calculated CRC16 value.

1.3 RAM TEST

March Tests are a family of tests that are well recognised as an effective way of testing RAM.
A March test consists of a finite sequence of March elements, while a March element is a finite sequence of operations applied to every cell in the memory array before proceeding to the next cell.
In general the more March elements the algorithm consists of the better will be its fault coverage but at the expense of a slower execution time.

The algorithms themselves are destructive (they do not preserve the current RAM values) but the supplied test functions provide a non-destructive option so that memory contents can be preserved. This is achieved by copying the memory to a supplied buffer before running the actual algorithm and then restoring the memory from the buffer at the end of the test. The API includes an option for automatically testing the buffer as well as the RAM test area.

The area of RAM being tested can not be used for anything else while it is being tested. This makes the testing of RAM used for the stack particularly difficult. To help with this problem the API includes functions which can be used for testing the stack.

The following section introduces the specific March Tests. Following that is the specification of the software APIs.

1.3.1 Algorithms

1.3.1.1 March C

The March C algorithm (van de Goor 1991) consists of 6 March elements with a total of 10 operations. It detects the following faults:

1. Stuck At Faults (SAF)
   - The logic value of a cell or a line is always 0 or 1.

2. Transition Faults (TF)
   - A cell or a line that fails to undergo a 0→1 or a 1→0 transition.

3. Coupling Faults (CF)
   - A write operation to one cell changes the content of a second cell.

4. Address Decoder Faults (AF)
   - Any fault that affects address decoder:
   - With a certain address, no cell will be accessed.
   - A certain cell is never accessed.
   - With a certain address, multiple cells are accessed simultaneously.
   - A certain cell can be accessed by multiple addresses.

These are the 6 March elements:-

I. Write all zeros to array
II. Starting at lowest address, read zeros, write ones, increment up array bit by bit.
III. Starting at lowest address, read ones, write zeros, increment up array bit by bit.
IV. Starting at highest address, read zeros, write ones, decrement down array bit by bit.
V. Starting at highest address, read ones, write zeros, decrement down array bit by bit.
VI. Read all zeros from array.

1.3.1.2 March X

Note: This algorithm has not been implemented for SH7124 and is only presented here for information as it relates to the March X WOM version below.

The March X algorithm consists of 4 March elements with a total of 6 operations. It detects the following faults:

1. Stuck At Faults (SAF)
2. Transition Faults (TF)
3. Inversion Coupling Faults (Cfin)
4. Address Decoder Faults (AF)

These are the 4 March elements:-

I. Write all zeros to array
II. Starting at lowest address, read zeros, write ones, increment up array bit by bit.
III. Starting at highest address, read ones, write zeros, decrement down array bit by bit.
IV. Read all zeros from array.

1.3.1.3 March X (Word-Oriented Memory version)

The March X Word-Oriented Memory (WOM) algorithm has been created from a standard March X algorithm in two stages. First the standard March X is converted from using a single bit data pattern to using a data pattern equal to the memory access width. At this stage the test is primarily detecting inter word faults including Address Decoder faults. The second stage is to add an additional two March elements. The first using a data pattern of alternating high/low bits then the second using the inverse. The addition of these elements is to detect intra-word coupling faults.

These are the 6 March elements:-

I. Write all zeros to array
II. Starting at lowest address, read zeros, write ones, increment up array word by word.
III. Starting at highest address, read ones, write zeros, decrement down word by word.
IV. Starting at lowest address, read zeros, write h'Aas, increment up array word by word.
V. Starting at highest address, read h'Aas, write h'55s, decrement down word by word.
VI. Read all h'55s from array.

1.3.2 Software API

1.3.2.1 March C API

This test can be configured to use 8, 16 or 32 bit RAM accesses.
This is achieved by #defining RAMTEST_MARCH_C_ACCESS_SIZE in the header file to be one of the following:

- RAMTEST_MARCH_C_ACCESS_SIZE_8BIT
- RAMTEST_MARCH_C_ACCESS_SIZE_16BIT
- RAMTEST_MARCH_C_ACCESS_SIZE_32BIT

Sometimes limiting the maximum size of RAM that can be tested with a single function call can speed the test up as well as reducing stack and code size. This is done by limiting the size of the variable used to hold the number of 'words' that the test area contains. The 'word' size is the selected access width.
This is achieved by #defining RAMTEST_MARCH_C_MAX_WORDS in the header file to be one of the following:

- RAMTEST_MARCH_C_MAX_WORDS_8BIT (Max words in test area is 0xFF)
- RAMTEST_MARCH_C_MAX_WORDS_16BIT (Max words in test area is 0xFFFF)
- RAMTEST_MARCH_C_MAX_WORDS_32BIT (Max words in test area is 0xFFFFFFFF)

NOTE: The SH7124 benchmarking showed that there was no benefit in reducing the max words count. So it is recommended to leave this as RAMTEST_MARCH_C_MAX_WORDS_32BIT.

Table 1: Source files:

File name
ramtest_march_c.h
ramtest_march_c.c

The source is written in ANSI C and uses MISRA-compliant data types as declared in file MisraTypes.h.

Declaration	
`bool_t RamTest_March_C(uint32_t ui32_StartAddr, uint32_t ui32_EndAddr, void* p_RAMSafe);`	
Description	
RAM memory test using March C (Goor 1991) algorithm.	
Input Parameters	
`ui32_StartAddr`	The address of the first word of RAM to be tested. This must be aligned with the selected memory access width.
`Ui32_EndAddr`	The address of the last word of RAM to be tested. This must be aligned with the selected memory access width and be a value greater or equal to ui32_StartAddr.
`P_RAMSafe`	For a destructive memory test set to NULL. For a non-destructive memory test, set to the start of a buffer that is large enough to copy the contents of the test area into it and that is aligned with the selected memory access width.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	TRUE = Test passed. FALSE = Test or parameter check failed.

Declaration	
`bool_t RamTest_March_C_Extra(uint32_t ui32_StartAddr, uint32_t ui32_EndAddr, void* p_RAMSafe);`	
Description	
Non Destructive RAM memory test using March C (Goor 1991) algorithm. This function differs from the RamTest_March_C function by testing the 'RAMSafe' buffer before using it. If the test of the 'RAMSafe' buffer fails then the test will be aborted and the function will return FALSE.	
Input Parameters	
`ui32_StartAddr`	The address of the first word of RAM to be tested. This must be aligned with the selected memory access width.
`Ui32_EndAddr`	The address of the last word of RAM to be tested. This must be aligned with the selected memory access width and be a value greater or equal to ui32_StartAddr.
`P_RAMSafe`	Set to the start of a buffer that is large enough to copy the contents of the test area into it and that is aligned with the selected memory access width.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	TRUE = Test passed. FALSE = Test or parameter check failed.

1.3.2.2 March X WOM API

This test can be configured to use 8, 16 or 32 bit RAM accesses.
This is achieved by #defining RAMTEST_MARCH_X_WOM_ACCESS_SIZE in the header file to be one of the following:

- RAMTEST_MARCH_ X_WOM_ACCESS_SIZE_8BIT
- RAMTEST_MARCH_ X_WOM_ACCESS_SIZE_16BIT
- RAMTEST_MARCH_ X_WOM_ACCESS_SIZE_32BIT

In order to speed up the run time of the test you can choose to limit the maximum size of RAM that can be tested with a single function call. This is done by limiting the size of the variable used to hold the number of 'words' that the test area contains. The 'word' size is the same as the selected access width.
This is achieved by #defining RAMTEST_MARCH_ X_WOM_MAX_WORDS in the header file to be one of the following:

- RAMTEST_MARCH_ X_WOM_MAX_WORDS_8BIT (Max words in test area is 0xFF)
- RAMTEST_MARCH_ X_WOM_MAX_WORDS_16BIT (Max words in test area is 0xFFFF)
- RAMTEST_MARCH_ X_WOM_MAX_WORDS_32BIT (Max words in test area is 0xFFFFFFFF)

Table 2: Source files:

File name
ramtest_march_x_wom.h
ramtest_march_x_wom.c

The source is written in ANSI C and uses MISRA-compliant data types as declared in file MisraTypes.h.

Declaration	
`bool_t RamTest_March_X_WOM(uint32_t ui32_StartAddr, uint32_t ui32_EndAddr, void* p_RAMSafe);`	
Description	
RAM memory test based on March X algorithm converted for WOM.	
Input Parameters	
`ui32_StartAddr`	Address of the first word of RAM to be tested. This must be aligned with the selected memory access width.
`Ui32_EndAddr`	Address of the last word of RAM to be tested. This must be aligned with the selected memory access width and be a value greater or equal to ui32_StartAddr.
`P_RAMSafe`	For a destructive memory test set to NULL. For a non-destructive memory test, set to the start of a buffer that is large enough to copy the contents of the test area into it and that is aligned with the selected memory access width.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	TRUE = Test passed. FALSE = Test or parameter check failed.

Declaration	
`bool_t RamTest_March_X_WOM_Extra(uint32_t ui32_StartAddr, uint32_t ui32_EndAddr, void* p_RAMSafe);`	
Description	
Non Destructive RAM memory test based on March X algorithm converted for WOM. This function differs from the RamTest_March_X_WOM_XXBit function by testing the 'RAMSafe' buffer before using it. If the test of the 'RAMSafe' buffer fails then the test will be aborted and the function will return FALSE.	
Input Parameters	
`ui32_StartAddr`	The address of the first word of RAM to be tested. This must be aligned with the selected memory access width.
`Ui32_EndAddr`	The address of the last word of RAM to be tested. This must be aligned with the selected memory access width and be a value greater or equal to ui32_StartAddr.
`P_RAMSafe`	Set to the start of a buffer that is large enough to copy the contents of the test area into it and that is aligned with the selected memory access width.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	TRUE = Test passed. FALSE = Test or parameter check failed.

1.3.2.3 RAM Test Stack

Table 3: Source files:

File name
ramtest_stack.h
ramtest_stack.c

This function enables a RAM test to be performed on an area of RAM that includes the stack. As the function that actually performs the RAM test requires a stack, this function will re-locate the stack to a supplied new RAM area allowing the original stack area to be tested.
This function makes use of another function, of the user's choice, to actually perform the memory test.

Declaration	

`bool_t RamTest_Stack(const uint32_t ui32_StartAddr,`
`const uint32_t ui32_EndAddr,`
`void* const p_RAMSafe,`
`const uint32_t ui32_NewSP,`
`const TEST_FUNC fpTest_Func);`	

Description	
RAM test of an area that includes the Stack.	
Input Parameters	
`ui32_StartAddr`	The address of the first word of RAM to be tested. This must be compatible with the requirements of the fpTest_Func.
`Ui32_EndAddr`	The address of the last word of RAM to be tested. This must be compatible with the requirements of the fpTest_Func.
`P_RAMSafe`	Set to the start of a buffer that is the same size as the test RAM area. This must be compatible with the requirements of the fpTest_Func.
`Ui32_NewUSP`	New Stack pointer value for the stack to be re-located to.
`fpTest_Func`	Function pointer of type TEST_FUNC to the actual memory test to be used. Typedef bool_t(*TEST_FUNC)(uint32_t, uint32_t, void*); For example 'RamTest_March_X_WOM'.
Output Parameters	
`NONE`	N/A
Return Values	
`bool_t`	TRUE = Test passed. FALSE = Test or parameter check failed.

2 TEST ENVIRONMENT

Development board: RSKSH7124, 10MHz external clock.
MCU: R5F71243V
Tool chain: SuperH RISC engine Standard Toolchain 9.3.0.0
In-circuit debugger: E10A.

2.1 TOOL CHAIN SETTINGS

2.1.1 No optimisation

Compiler:
-cpu=sh2 -include="$(PROJDIR)\Tests" -debug -optimize=0 -noinline -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo

Linker:
-noprelink -rom=D=R -nomessage -list="$(CONFIGDIR)\$(PROJECTNAME).map" -show=all -nooptimize
-start=DVECTTBL,DINTTBL/00,PIntPRG/0400,PResetPRG,P,C,C$DSEC,C$BSEC,D,PRAM_TEST_MarchC,PRAM_TEST_MarchXWOM,PRAM_TEST_Stack/0800,RAM_TEST_AREA/0FFFFA000,B,R,BRAM_TEST_Stack/0FFFFA500,S/0FFFFBE00 -nologo -stack
-output="$(CONFIGDIR)\$(PROJECTNAME).abs" -end
-input="$(CONFIGDIR)\$(PROJECTNAME).abs" -form=stype
-output="$(CONFIGDIR)\$(PROJECTNAME).mot" -exit

2.1.2 Minimal ROM size

Compiler:
-cpu=sh2 -include="$(PROJDIR)\Tests" -debug -size -noinline -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 –nologo

Linker:
-noprelink -rom=D=R -nomessage -list="$(CONFIGDIR)\$(PROJECTNAME).map" -show=all -nooptimize
-start=DVECTTBL,DINTTBL/00,PIntPRG/0400,PResetPRG,P,C,C$DSEC,C$BSEC,D,PRAM_TEST_MarchC,PRAM_TEST_MarchXWOM,PRAM_TEST_Stack/0800,RAM_TEST_AREA/0FFFFA000,B,R,BRAM_TEST_Stack/0FFFFA500,S/0FFFFBE00 -nologo -stack
-output="$(CONFIGDIR)\$(PROJECTNAME).abs" -end
-input="$(CONFIGDIR)\$(PROJECTNAME).abs" -form=stype
-output="$(CONFIGDIR)\$(PROJECTNAME).mot" -exit

2.1.3 Maximum speed

Compiler:
-cpu=sh2 -include="$(PROJDIR)\Tests" -debug -speed -noinline -gbr=auto -chgincpath -errorpath -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1 -nologo

Linker:
-noprelink -rom=D=R -nomessage -list="$(CONFIGDIR)\$(PROJECTNAME).map" -show=all -optimize=speed
-start=DVECTTBL,DINTTBL/00,PIntPRG/0400,PResetPRG,P,C,C$DSEC,C$BSEC,D,PRAM_TEST_MarchC,PRAM_TEST_MarchXWOM,PRAM_TEST_Stack/0800,RAM_TEST_AREA/0FFFFA000,B,R,BRAM_TEST_Stack/0FFFFA500,S/0FFFFBE00 -nologo -stack
-output="$(CONFIGDIR)\$(PROJECTNAME).abs" -end
-input="$(CONFIGDIR)\$(PROJECTNAME).abs" -form=stype
-output="$(CONFIGDIR)\$(PROJECTNAME).mot" -exit

3 BENCHMARKING RESULTS

The function execution time was measured using the pulse-width measurement function on a TDS3034B digital oscilloscope. A port pin was set low at function entry and high at function exit.

The clock cycle count was calculated using the following equation:

$$Clock\ cycles = f_{CPU} \times t_{FUNCTION}$$

where : f_{CPU} is the CPU clock frequency (Hz)
$t_{FUNCTION}$ is the function execution time (seconds)

Code Size is the size of all functions in the specific file.

3.1 CPU TEST RESULTS

Note: Optimisation cannot be used for these tests.

Table 4: SH2 CPU test results

Measurement	Result
Code size (bytes) including Coupling GPR Test.	3400
Code size (bytes) excluding	904
Stack usage (bytes) for CPU_TestAll	4
Clock cycle count to execute function CPU_TestAll when it uses Coupling GPR tests.	1564
Clock cycle count to execute function CPU_TestAll when it doesn't use the Coupling GPR tests.	304

3.2 RAM TEST RESULTS

The tests were executed in 8-, 16- and 32- bit access width configurations. The 32-bit word limit was always used as it was found that using a smaller limit did not improve performance.

The name 'Extra' refers to the function that includes the automatic safe buffer test.

3.2.1 March C

Table 5: SH/7124 March C test results (8-bit access, 32-bit word limit)

Measurement			Optimisation		
			None	Size	Speed
Code size (bytes)			826	522	1272
Stack usage (bytes)			88	64	68
Stack usage Extra (bytes)			112	80	84
Clock cycle count (/ 1000)	Destructive	1024 bytes	1434	940	628.4
		500 bytes	700	460	306.8
		100 bytes	140	92	61.6
	Non-destructive	1024 bytes	1456	956	638.8
		500 bytes	711.2	468	312
		100 bytes	142.4	93.6	62.4
	Extra	1024 bytes	2888	1896	1267.2
		500 bytes	1412	924	619.2
		100 bytes	282.8	185.6	124.4
Time Measured (ms)	Destructive	1024 bytes	35.85	23.5	15.71
		500 bytes	17.5	11.5	7.67
		100 bytes	3.5	2.3	1.54
	Non-destructive	1024 bytes	36.4	23.9	15.97
		500 bytes	17.78	11.7	7.8
		100 bytes	3.56	2.34	1.56
	Extra	1024 bytes	72.2	47.4	31.68
		500 bytes	35.3	23.1	15.48
		100 bytes	7.07	4.64	3.11

Table 6: SH/7124 March C test results (16-bit access, 32-bit word limit)

Measurement			Optimisation		
			None	Size	Speed
Code size (bytes)			872	584	1412
Stack usage (bytes)			88	68	72
Stack usage Extra (bytes)			112	84	88
Clock cycle count (/ 1000)	Destructive	1024 bytes	1492	916	608
		500 bytes	728	448	297.2
		100 bytes	146	90	59.6
	Non-destructive	1024 bytes	1504	928	612
		500 bytes	736	452	300
		100 bytes	147.2	90.4	60.4
	Extra	1024 bytes	2996	1844	1224
		500 bytes	1464	900	596
		100 bytes	293.2	180.4	120
Time Measured (ms)	Destructive	1024 bytes	37.3	22.9	15.2
		500 bytes	18.2	11.2	7.43
		100 bytes	3.65	2.25	1.49
	Non-destructive	1024 bytes	37.6	23.2	15.3
		500 bytes	18.4	11.3	7.5
		100 bytes	3.68	2.26	1.51
	Extra	1024 bytes	74.9	46.1	30.6
		500 bytes	36.6	22.5	14.9
		100 bytes	7.33	4.51	3

Table 7: SH/7124 March C test results (32-bit access, 32-bit word limit)

Measurement			Optimisation None	Size	Speed
Code size (bytes)			818	564	1308
Stack usage (bytes)			88	64	68
Stack usage Extra (bytes)			112	80	84
Clock cycle count (/ 1000)	Destructive	1024 bytes	1200	852	570.8
		500 bytes	588	416	278.8
		100 bytes	117.6	84	56
	Non-destructive	1024 bytes	1208	856	572
		500 bytes	592	420	280.4
		100 bytes	118.4	84	56.4
	Extra	1024 bytes	2412	1708	1144
		500 bytes	1176	836	560
		100 bytes	236	168	112.4
Time Measured (ms)	Destructive	1024 bytes	30	21.3	14.27
		500 bytes	14.7	10.4	6.97
		100 bytes	2.94	2.1	1.4
	Non-destructive	1024 bytes	30.2	21.4	14.3
		500 bytes	14.8	10.5	7.01
		100 bytes	2.96	2.1	1.41
	Extra	1024 bytes	60.3	42.7	28.6
		500 bytes	29.4	20.9	14
		100 bytes	5.9	4.2	2.81

3.2.2 March X WOM

Table 8: SH/7124 March X WOM test results (8-bit access, 32-bit word limit)

Measurement			Optimisation None	Size	Speed
Code size (bytes)			626	390	606
Stack usage (bytes)			72	44	48
Stack usage Extra (bytes)			96	60	64
Clock cycle count (/ 1000)	Destructive	1024 bytes	105.6	86.4	83.2
		500 bytes	51.6	42	40.4
		100 bytes	10.4	8.8	8.4
	Non-destructive	1024 bytes	128	102.8	92.4
		500 bytes	62.8	50	45.2
		100 bytes	15.2	10.4	9.2
	Extra	1024 bytes	234	188.8	175.6
		500 bytes	114.8	92.4	86
		100 bytes	23.6	18.8	17.6
Time Measured (ms)	Destructive	1024 bytes	2.64	2.16	2.08
		500 bytes	1.29	1.05	1.01
		100 bytes	0.26	0.22	0.21
	Non-destructive	1024 bytes	3.2	2.57	2.31
		500 bytes	1.57	1.25	1.13
		100 bytes	0.38	0.26	0.23
	Extra	1024 bytes	5.85	4.72	4.39
		500 bytes	2.87	2.31	2.15
		100 bytes	0.59	0.47	0.44

Table 9: SH/7124 March X WOM test results (16-bit access, 32-bit word limit)

Measurement			Optimisation None	Size	Speed
Code size (bytes)			676	452	702
Stack usage (bytes)			72	44	52
Stack usage Extra (bytes)			96	60	68
Clock cycle count (/ 1000)	Destructive	1024 bytes	64	50.4	44
		500 bytes	31.2	24.8	21.2
		100 bytes	6.4	5.2	4.4
	Non-destructive	1024 bytes	77.6	59.6	48.8
		500 bytes	38	29.2	24
		100 bytes	7.6	6	4.8
	Extra	1024 bytes	142	110	92.4
		500 bytes	69.2	54	45.6
		100 bytes	14.4	11.2	9.6
Time Measured (ms)	Destructive	1024 bytes	1.6	1.26	1.1
		500 bytes	0.78	0.62	0.53
		100 bytes	0.16	0.13	0.11
	Non-destructive	1024 bytes	1.94	1.49	1.22
		500 bytes	0.95	0.73	0.6
		100 bytes	0.19	0.15	0.12
	Extra	1024 bytes	3.55	2.75	2.31
		500 bytes	1.73	1.35	1.14
		100 bytes	0.36	0.28	0.24

Table 10: SH/7124 March X WOM test results (32-bit access, 32-bit word limit)

Measurement			Optimisation		
			None	Size	Speed
Code size (bytes)			652	440	660
Stack usage (bytes)			68	44	48
Stack usage Extra (bytes)			92	60	64
Clock cycle count (/ 1000)	Destructive	1024 bytes	32.4	22.4	20
		500 bytes	16	11.08	9.92
		100 bytes	3.6	2.36	2.08
	Non-destructive	1024 bytes	40	27.12	22.72
		500 bytes	19.68	13.36	11.2
		100 bytes	4	2.84	2.4
	Extra	1024 bytes	72	49.6	42.8
		500 bytes	35.52	24.4	21.12
		100 bytes	7.52	5.2	4.64
Time Measured (ms)	Destructive	1024 bytes	0.81	0.56	0.5
		500 bytes	0.4	0.277	0.248
		100 bytes	0.09	0.059	0.052
	Non-destructive	1024 bytes	1	0.678	0.568
		500 bytes	0.492	0.334	0.28
		100 bytes	0.1	0.071	0.06
	Extra	1024 bytes	1.8	1.24	1.07
		500 bytes	0.888	0.61	0.528
		100 bytes	0.188	0.13	0.116

3.2.3 Stack Test

Note: This does not contain timing information as that depends upon the specific algorithm used. The time to move the stack is negligible compared with the memory test.

	Optimisation		
Measurement	**None**	**Size**	**Speed**
Code size (bytes) Program	180	564	156
Code size (bytes) RAM	28	28	28
Stack usage (bytes)	4	4	4

3.3 ROM TEST RESULTS

Table 11: SH/7124 test results CRC16-CCITT using a static table

		Optimisation		
Measurement		**None**	**Size**	**Speed**
Code size / bytes		112	60	64
Constant size / bytes		512	512	512
Stack usage / bytes		24	0	4
Clock cycle count (/ 1000)	1k bytes	38.4	22.4	20
	4k bytes	152	90.4	77.6
	16k bytes	604	360	311.2
Time Measured (ms)	1k bytes	0.96	0.56	0.5
	16k bytes	3.8	2.26	1.94
	64k bytes	15.1	9	7.78

Table 12: SH/7124 test results CRC16-CCITT using no table (bit shifting)

		Optimisation		
Measurement		**None**	**Size**	**Speed**
Code size / bytes		180	74	74
Constant size / bytes		0	0	0
Stack usage / bytes		20	0	0
Clock cycle count (/ 1000)	1k bytes	75.2	32	30.4
	4k bytes	298.4	126.4	124
	16k bytes	1196	508	492
Time Measured (ms)	1k bytes	1.88	0.8	0.76
	16k bytes	7.46	3.16	3.1
	64k bytes	29.9	12.7	12.3

Table 13: SH/7124 test results CRC16-CCITT with small lookup table

		Optimisation		
Measurement		**None**	**Size**	**Speed**
Code size / bytes		276	172	176
Constant size / bytes		32	32	32
Stack usage / bytes		44	20	20
Clock cycle count (/ 1000)	1k bytes	116	64.8	60.8
	4k bytes	468	258.4	245.6
	16k bytes	1868	1032	984
Time Measured (ms)	1k bytes	2.9	1.62	1.52
	16k bytes	11.7	6.46	6.14
	64k bytes	46.7	25.8	24.6

Table 14: SH/7124 test results CRC16 (NOT CCITT)with small lookup table

		Optimisation		
Measurement		**None**	**Size**	**Speed**
Code size / bytes		208	112	124
Constant size / bytes		32	32	32
Stack usage / bytes		32	4	8
Clock cycle count (/ 1000)	1k bytes	86.4	45.6	45.6
	4k bytes	348	184.8	180
	16k bytes	1392	736	720
Time Measured (ms)	1k bytes	2.16	1.14	1.14
	16k bytes	8.7	4.62	4.5
	64k bytes	34.8	18.4	18

4 ABBREVIATIONS

API	Application Programming Interface
CCITT	Comité Consultatif International Téléphonique et Télégraphique, an organisation that sets international communications standards.
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
IEC60730	International Electronics Commission 60730 safety standards
MCU	Micro Controller Unit
MISRA	Motor Industry Software Reliability Association
RAM	Random Access Memory
ROM	Read-Only Memory
WOM	Word Oriented Memory
XOR	Exclusive-OR

5 WEBSITE AND SUPPORT

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

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Dec.12.09	—	First edition issued
2.00	Mar.15.10		All CPU Cycle count values have been doubled. This was necessary because the conversion from time to CPU cycles was done incorrectly. The benchmarking timing values remain valid.

6 CAUTIONS

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life

   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.