

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.

32170/32174 Group, 32171 Group, 32176 Group

Example of Startup Program

1. Overview

The following article shows sample of general-purpose startup program used into 32170/32174 group, 32171 group, 32176 group.

2. Introduction

The explanation of this issue is applied to the following condition:

- Microcomputer: 32170/32174 Group
32171 Group
32176 Group
- Operation frequency: 20 to 40 MHz

3. Summary of the startup program

For an embedded application program to be run on the target system, you must have a startup program to initialize the target system, in addition to the user program and an EIT (Exception, Interrupt, Trap) handler routine.

EIT handler is detailed in chapter 4, “The Outline of EIT”

3.1 Method for Creating a Startup Program

To run an embedded application on the target system, you must have a program known as the “startup program” which calls up the user program (main function). The startup program performs initialization and processing, items (1) through (12) below.

- (1) EIT Vector Entry
- (2) Set Protect ID
- (3) ICU Vector Table
- (4) Allocate memory for the SFR area
- (5) EIT handler processing
- (6) Allocate memory for the stack area
- (7) Set microprocessor operation mode
- (8) Set the stack pointer
- (9) Set the data area sections (B, D)
- (10) Set Base register
- (11) Call the main function
- (12) IE bit Operation Function

The need for these processing varies with the contents of the actual application program.

Additional processing may be required, or some of the above processing may not be necessary.

In case the C standard library is used, it may be required to incorporate initializing and terminating.

For details refer to the “CC32R User's Manual C Compiler”.

3.1.1 EIT Vector Entry

(Lines 10 through 63 in Chapter 5, “Sample Programming Code”)

EIT vector entry is placed at the head of an internal ROM area or an extended area, and allocates a branch instruction to the processing handler start address (Note) of each EIT phenomenon. (Note: It is not branch address)
For details about EIT phenomenon refer to chapter 4, “The Outline of EIT”.

3.1.2 Setting Protect ID

(Lines 65 through 67 in Chapter 5, “Sample Programming Code”)

As general-purpose serial programmer, writing and erasing tools of internal flash memory in emulators and so on are in used, it checks out ID inputted with the tool against ID in an internal flash memory. Unless it is inputted right ID, writing and erasing operation are not accepted. (In some tools, after erasing all areas, it becomes possible to use tool operation and to write in internal flash memory.)

If it is not necessary to protect internal flash memory, set all checking ID areas for protection of an internal flash memory (H'0000 0084 to H'0000 0093) up as H'FF. It is assumed that protect ID is not necessary in this sample program.

3.1.3 ICU Vector Table

(Lines 70 through 109 in Chapter 5, “Sample Programming Code”)

Set ICU Vector Table (in the case of 32170, it is the address from H'0000 0094 to H'0000 010F) up as the start address of the interrupt handler of each internal peripheral I/O. As an interrupt request is accepted, set address of low 16 bits for ICU vector table that corresponds to sources of acceptance for it up as an Interrupt Vector Register (IVECT). As for EIT handler, the address of an ICU vector table is acquired by reading out the contents of this IVECT register under LDH command.

3.1.4 Allocating memory for the SFR area

(Lines 111 through 116 in Chapter 5, “Sample Programming Code”)

The register of internal peripheral I/O is allocated from address H'0080 0000. 16K bytes of SFR (Special Function Register) area is allocated here.

3.1.5 EIT handler processing

(Lines 120 through 197 in Chapter 5, “Sample Programming Code”)

If an EIT phenomenon occurs, the program that was being run till then is interrupted, and after running a hardware preprocessing (after-mentioned), it will branch to EIT handler processing.

In the case of this sample program, the EIT handler of external interrupt (EI) is realized.

For details refer to chapter 4.3, “Procedure for Processing of External Interrupts (EI) by EIT Handler”.

As system break interruption, exception and trap are occurred, handler processing is infinite loop processing.

In those cases, please describe the start address (label name) of the processing in the part where an EIT vector entry corresponds.

3.1.6 Allocating memory for the stack area (Lines 218 through 227 in Chapter 5, “Sample Programming Code”)

Allocate memory for the stack area required for the operation of the application. This generally is accomplished by declaring a stack area section with the pseudo-instruction, [.SECTION] and allocating memory for the stack area with the pseudo-instruction [.RES] in an assembly program.

The stack size required for the operation of the application is determined by considering the relationship between function calls and the stack size used by each function.

However, because it is difficult to find the exact stack size, a more practical method is adopted in which the necessary amount of stack is examined by actually running the application in the debug and evaluation processes. (For example, an ample stack area is allocated at the initial stage and then the necessary amount of stack is examined using a debugger.)

In addition, there is also the method of using a stack size calculation utility (stk32R). Stk32R is able to calculate stack size for operating program with adding “-stack” option and processing outputted stack quantity consumed indication file when compile. For details refer to the “CC32R User's Manual C Compiler”.

Because the stack is used in units of WORD (4 bytes), the stack area must be allocated in units of 4 bytes.

In this sample program 2048 bytes of stack area is secured. (It is necessary to secure a suitable stack area by the program in fact)

3.1.7 Setting microprocessor operation mode (Lines 234,235,297 through 299 in Chapter 5, “Sample Programming Code”)

Specify which stack to use and whether or not to enable interrupts for the target microprocessor by using the Processor Status Word (PSW) register.

The Processor Status Word (PSW) register indicates the status of the CPU, and consists of a normally used PSW field and a BPSW field used to save the PSW field when an EIT occurs.

In this sample program, settings are made to “disable interrupts” immediately after program execution, and to “use the interrupt stack” and “enable interrupts” immediately before calling the main function.

3.1.8 Setting the stack pointer (Lines 240 through 243 in Chapter 5, “Sample Programming Code”)

Set the most significant address of the allocated stack area in the stack pointer.

Because it has described that setting of interrupt stack is only used (stack mode bit (SM) is always “0”) on the assumption, stack for users is not set up.

3.1.9 Setting the data area sections (B, D)

Embedded applications require initializing the data area without initial values (section B) and the data area with initial values (section D).

Perform the following operations when linking the application and when executing the startup program:

(1) When linking the application

- Locate sections D and B in the RAM area. (Only allocate memory, without data output.)
- Locate the initial value data area for section D (ROM_D) in the ROM area.

(2) When initializing the startup program

- Initialize section B in the RAM area by clearing it to “0” (filling the area with 0s).
(Lines 246 through 257 in Chapter 5, “Sample Programming Code”)
- Initialize the data located in the ROM_D section of the ROM area by transferring it into section D in the RAM area.
(Lines 260 through 273 in Chapter 5, “Sample Programming Code”)

3.1.10 Setting base register

(Lines 276 through 292 in Chapter 5, “Sample Programming Code”)

If the “-rel16” option is added when compiling, R11, R12, R13-based register relative indirect instruction is outputted for access to symbols for sections D and B.

Here, the program sets the R12 register and defines “__REL_BASE12” required for it. Because in this sample program, H'0080 8000 is defined for “__REL_BASE12”, the register relative indirect instruction is output for the area $\pm 32\text{KB}$ from H'0080 8000. And in case of adding “-access” option, an access control file is required in addition to setup the above-mentioned base register.

For details refer to the “CC32R User’s Manual C Compiler”.

3.1.11 Calling the main function

(Lines 301 in Chapter 5, “Sample Programming Code”)

A subroutine call is set at the start address of the main function. For programs comprised not to return from the main function, a simple jump instruction can suffice.

For details, refer to the “CC32R User’s Manual C Compiler”.

3.1.12 IE bit operation function

(Lines 309 through 323 in Chapter 5, “Sample Programming Code”)

- Interrupt enable function (EnInt)
 Interruption is enabled by setting the interrupt enable bit (IE) of PSW to “1”.
- Interrupt disable function (DisInt)
 Interruption is disabled by clearing IE bit of PSW to “0”.

4. The Outline of EIT

In process of CPU is running the usual program, if a certain phenomenon occurs, it may be necessary to interrupt the program and run another program. Such a phenomenon is generically called an EIT (Exception, Interrupt, and Trap) phenomenon.

4.1 Causes of EIT

There are following causes of EIT for the M32R/ECU.

(1) Exception

Exception occurs for errors or violations encountered during instruction execution, and include the following.

- Reserved Instruction Exception (RIE)
This Exception occurs when an unimplemented instruction is detected.
- Address Exception (AE)
This Exception occurs when an attempt is made to access non-aligned addresses in load or store instructions.

(2) Interrupt

Interrupts are generated for hardware causes or sources of the microcomputer, and include the following:

- Reset Interrupt (RI)
It is generated when a reset signal is accepted.
Reset interrupt has the highest priority.
- System Break Interrupt (SBI)
This is an emergency interrupt generated when power-down condition is detected or when a faulty state is notified by an external watchdog timer, etc. This interrupt can only be used in cases where the interrupt handler does not return to the main routine.
- External Interrupt (EI)
Interrupt request from each internal peripheral I/Os controlled by interrupt controller. The interrupt controller controls interrupt by 8 level priority order include interrupt disable.

(3) Trap

This refers to a software interrupt generated by executing the TRAP instruction in the program.

4.2 The Outline of EIT processing

As EIT phenomenon occurs, EIT processing takes place. In EIT processing there are areas which the hardware processes automatically or which the user routine program (EIT handler) processes.

EIT Processing Procedure except for Interrupt Reset is below.

(1) Processing in hardware when an EIT is accepted

- Register transfer: PC => BPC
- Register transfer: PSW => BPSW

(2) EIT handler

- Executes the instruction in the appropriate EIT vector entry for the cause of occurrence (branch instruction to the appropriate EIT handler, Lines 10 through 63 in Chapter 5, "Sample Programming Code")
- After executing the EIT handler, the processor exits from it with the RTE instruction. (except SBI)

(3) Processing in hardware after Command RTE

- Register transfer: BPSW => PSW
- Register transfer: BPC => PC

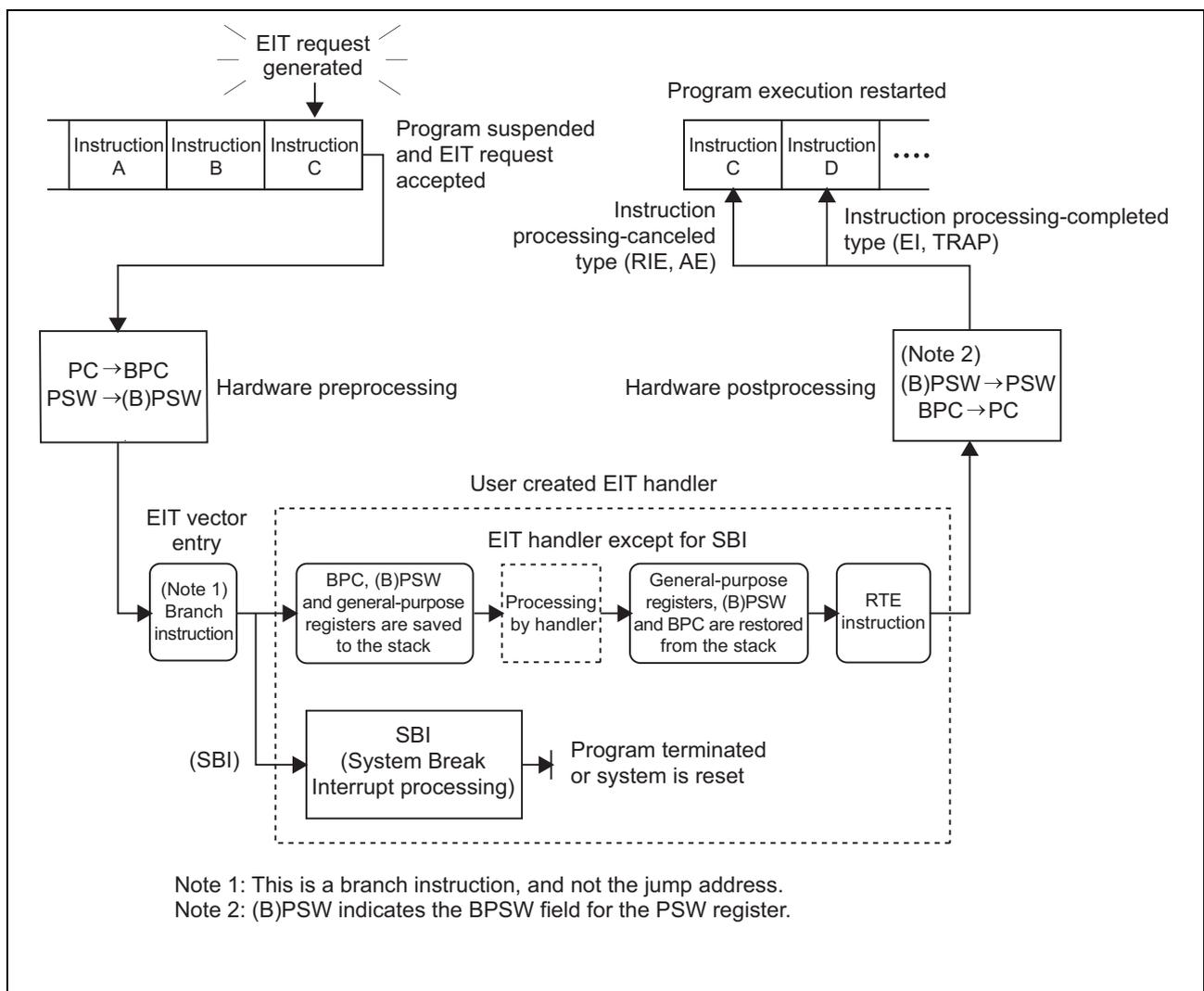


Figure 4.2.1 Outline of EIT Processing Procedure

4.3 Procedure for Processing of External Interrupts (EI) by EIT Handler

The EIT handler processing procedure of external interrupts (EI) is shown below and also it is shown line 120 though 197 in chapter 5, "Sample Programming Code".

It has optimized so that it can run at the maximum high speed considering a motion of a pipeline and so on.

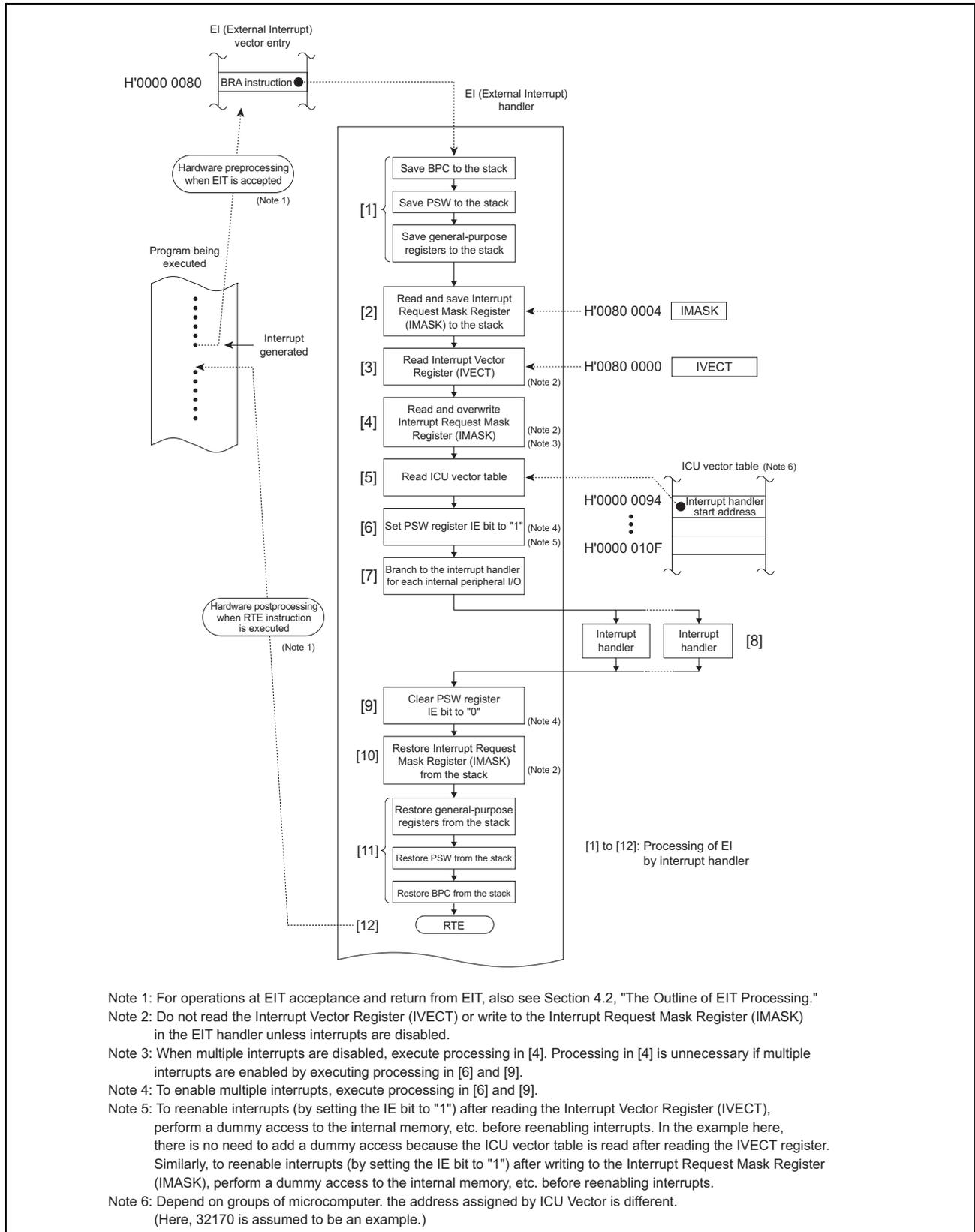


Figure 4.3.1 The example of EI handler processing operation from internal peripheral I/O of M32R/ECU

The External Interrupt (EI) in Figure 4.3.1 is explained here.

(1) Saving each registers to the stack

(BPC, PSW, general-purpose registers, and accumulator)

- Backup PC (BPC) evacuates the value of a program counter (PC) when EIT occurs.
At that time, the value of PC just before that or next commanded is set, and the value of BPC is returned to PC when “RTE command” runs.
(Lines 147,162 in Chapter 5, “Sample Programming Code”)
- PSW is the register that displays the status of M32R, and consists of PSW field usually used and BPSW field for evacuating PSW field when EIT occurs. SM, IE, and C in PSW register are evacuated respectively to BPSW, when EIT occurs.
(Lines 145,161 in Chapter 5, “Sample Programming Code”)
- General-purpose registers (R0 - R15) that are used in the EIT handler must always be saved. The registers used in user processing executed from the EIT handler also need to be saved.
(Lines 129 through 132,134,144,146,148,156,157 in Chapter 5, “Sample Programming Code”)
- The accumulator (ACC) must always be saved if DSP feature instructions or the multiply instruction (MUL) are used in the EIT handler.
(Lines 137 through 140 in Chapter 5, “Sample Programming Code”)

(2) Reading out the Interrupt Mask Register (IMASK) and saving to the stack

(Lines 133,142,163 in Chapter 5, “Sample Programming Code”)

- Reading out the Interrupt Mask Register and saving to the stack

(3) Reading out the Interrupt Vector Register (IVECT)

(Lines 143,150 in Chapter 5, “Sample Programming Code”)

- Reading out the IVECT.
- By reading out this IVECT, hardware set the interrupt priority level (ILEVEL) of the interrupt demand sources received automatically to IMASK register as a new value.

(4) Reading out and overwrite IMASK

(Lines 152 to 154 in Chapter 5, “Sample Programming Code”)

- Reading out IMASK and overwrite IMASK with the read value.
- This processing is not necessary in case of permitting multiple interruptions.

(5) Reading out the ICU vector table

(Lines 159 in Chapter 5, “Sample Programming Code”)

- Read out the ICU vector table received interrupt demand sources. The address corresponding the ICU vector table is obtained by zero expansion that expands the contents read in (3) above (address low 16 bits of the ICU vector table which receives interrupt demand sources) of the interrupt vector table register.

(6) Permission of multiplex interrupt

(Lines 135,165 in Chapter 5, “Sample Programming Code”)

- In case of permitting higher interrupt of a priority level during interrupt processing (permission of multiplex interrupt), it sets IE bit of PSW to “1”. In this sample programming code, multiplex interrupt is set as permission.

- (7) Branching to Interrupt handler of each Internal Peripheral I/Os
(Lines 167 in Chapter 5, "Sample Programming Code")
- It branches to the start address of the interrupt handler read in (5) above.
- (8) Processing Interrupt handler of each Internal Peripheral I/Os
- (9) Prohibiting interrupt
(Lines 169,170 in Chapter 5, "Sample Programming Code")
- By clearing IE bit to "0", Interrupt is prohibited.
- (10) Restoring the Interrupt Mask Register
(Lines 172,173,178 in Chapter 5, "Sample Programming Code")
- Restore the Interrupt Mask Register that is saved in (2).
- (11) Restoring each registers from the stack
(Accumulator, general-purpose registers, PSW, BPC.)
(Lines 171,172,174 through 177, 179 through 195 in Chapter 5, "Sample Programming Code")
- Restore the Register that is saved in (1).
- (12) Completion of external interrupt processing
(Lines 197 in Chapter 5, "Sample Programming Code")
- Complete external interrupt processing by RTE command.

4.4 Interrupts from Internal Peripheral I/Os

Interrupts from the internal peripheral I/Os are taken care of by the Interrupt Controller (ICU) along with system break interrupts, and are notified as external interrupt (EI) to the M32R CPU.

The 32170 has a total of 31 sources for interrupts from the internal peripheral I/Os, which are assigned priority in levels up to 8 (including interrupts disabled) for management purposes. When multiple interrupts with the same priority occur simultaneously, they are resolved by predetermined hardware priority. The source for an interrupt request generated from internal peripheral I/Os can be identified by reading the internal peripheral I/Os' interrupt status register.

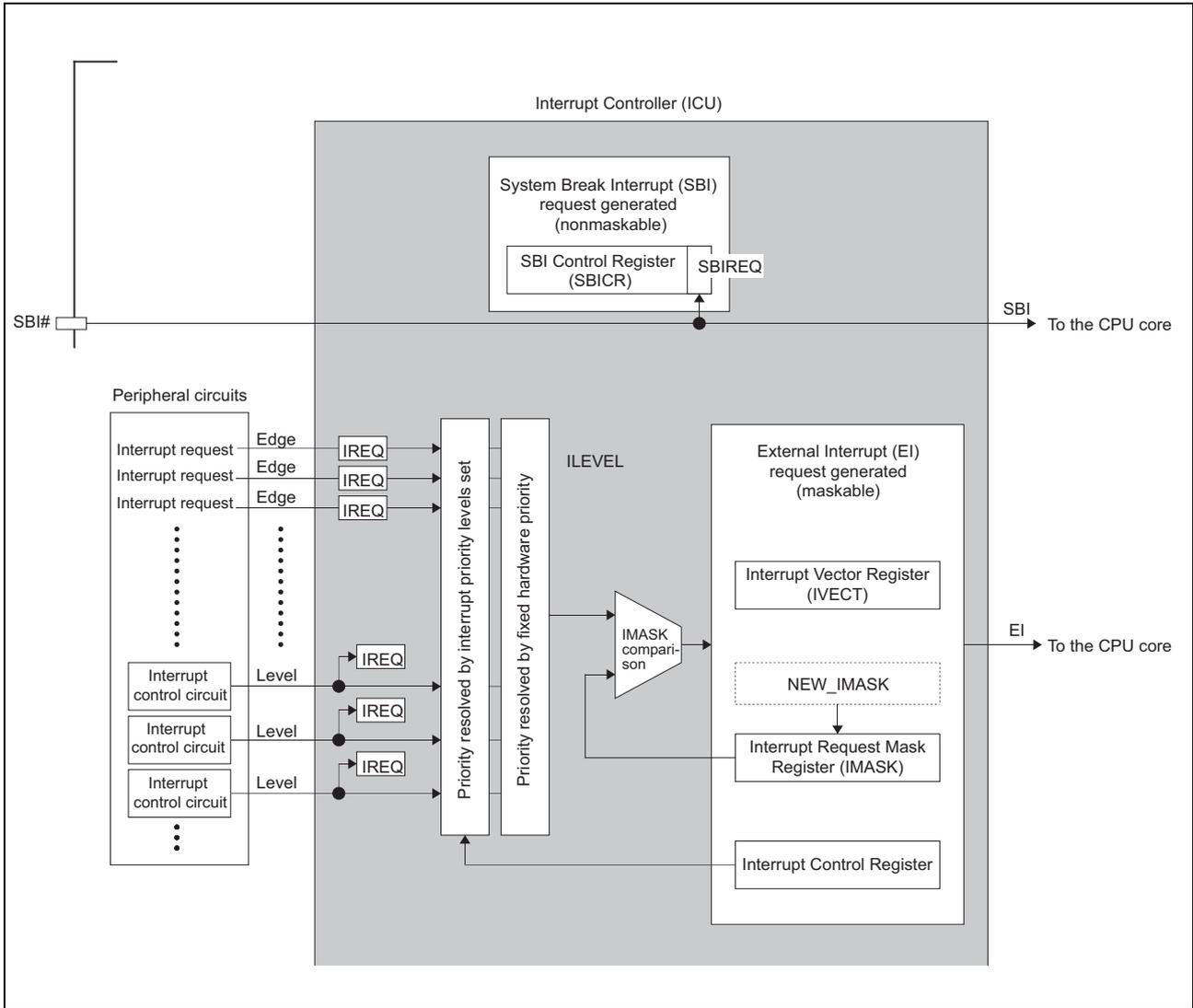


Figure 4.4.1 Block Diagram of the Interrupt Controller

4.4.1 Interrupt Sources of Internal Peripheral I/Os

The interrupt controller receives as its inputs the interrupt requests from MJT (multijunction timer), DMAC, serial interface, A/D converter, RTD, and CAN.

For details about these interrupts, refer each section at user's manual in which the relevant internal peripheral I/O is described.

Table 4.4.1 32170 Interrupt Sources of Internal Peripheral I/Os

Interrupt source	Contents	Number of Input Source	ICU Type of Input Source (Note 1)
A/D0 converter interrupt	Single-shot operation of A/D0 converter's scan mode completed, single mode completed, or comparator mode completed	1	Edge-type
A/D1 converter interrupt	Single-shot operation of A/D1 converter's scan mode completed, single mode completed, or comparator mode completed	1	Edge-type
SIO0 transmit interrupt	SIO0 transmit buffer empty interrupt	1	Edge-type
SIO0 receive interrupt	SIO0 reception completed or receive error interrupt	1	Edge-type
SIO1 transmit interrupt	SIO1 transmit buffer empty interrupt	1	Edge-type
SIO1 receive interrupt	SIO1 reception completed or receive error interrupt	1	Edge-type
SIO2, 3 transmit/ receive interrupt	SIO2, 3 reception completed or receive error interrupt, Transmit buffer empty	4	Level-type
SIO4, 5 transmit/ receive interrupt	SIO4, 5 reception completed or receive error interrupt, Transmit buffer empty	4	Level-type
TID0 output interrupt	TID0 output	1	Edge-type
TID1 output interrupt	TID1 output	1	Edge-type
TID2 output interrupt	TID2 output	1	Edge-type
TOD0 output interrupt	TOD0_0 - TOD0_7 outputs	8	Level-type
TOD1+TOM0 output Interrupt	TOD1_0 - TOD1_7 outputs + TOM0_0 - TOM0_7 outputs	16	Level-type
TML1 input interrupt	TML1 input (TIN30 - TIN33 inputs)	4	Level-type
RTD interrupt	RTD interrupt generation command	1	Edge-type
DMA transfer interrupt 0	DMA0 - 4 transfers completed	5	Level-type
DMA transfer interrupt 1	DMA5 - 9 transfers completed	5	Level-type
CAN0 transmit/receive & error interrupt	CAN0 transmission completed, CAN0 reception completed CAN0 error passive, CAN0 error bus off, CAN0 bus error	19	Level-type
MJT output interrupt 7	MJT output interrupt group 7 (TMS0 and TMS1 outputs)	2	Level-type
MJT output interrupt 6	MJT output interrupt group 6 (TOP8 and TOP9 outputs)	2	Level-type
MJT output interrupt 5	MJT output interrupt group 5 (TOP10 outputs)	1	Edge-type
MJT output interrupt 4	MJT output interrupt group 4 (TIO4 - TIO7 outputs)	4	Level-type
MJT output interrupt 3	MJT output interrupt group 3 (TIO8 and TIO9 outputs)	2	Level-type
MJT output interrupt 2	MJT output interrupt group 2 (TOP0 - TOP5 outputs)	6	Level-type
MJT output interrupt 1	MJT output interrupt group 1 (TOP6 and TOP7 outputs)	2	Level-type
MJT output interrupt 0	MJT output interrupt group 0 (TIO0 - TIO3 outputs)	4	Level-type
MJT input interrupt 4	MJT input interrupt group 4 (TIN3 - TIN6 inputs)	4	Level-type
MJT input interrupt 3	MJT input interrupt group 3 (TIN20 - TIN23 inputs)	4	Level-type
MJT input interrupt 2	MJT input interrupt group 2 (TIN12 - TIN19 inputs)	8	Level-type
MJT input interrupt 1	MJT input interrupt group 1 (TIN0 - TIN2 inputs)	3	Level-type
MJT input interrupt 0	MJT input interrupt group 0 (TIN7 - TIN11 inputs)	5	Level-type

Note 1: ICU type of input source

Edge-type: Interrupt requests are generated on a falling edge of the interrupt signal applied to the ICU.

Level-type: Interrupt requests are generated when the interrupt signal applied to the ICU is held low.

For these level-recognized interrupts, the ICU's Interrupt Control Register IRQ bit cannot be set or cleared in software.

Interrupts signaled to the ICU are recognized by either edge or level. Each type of interrupt is schematically explained below.

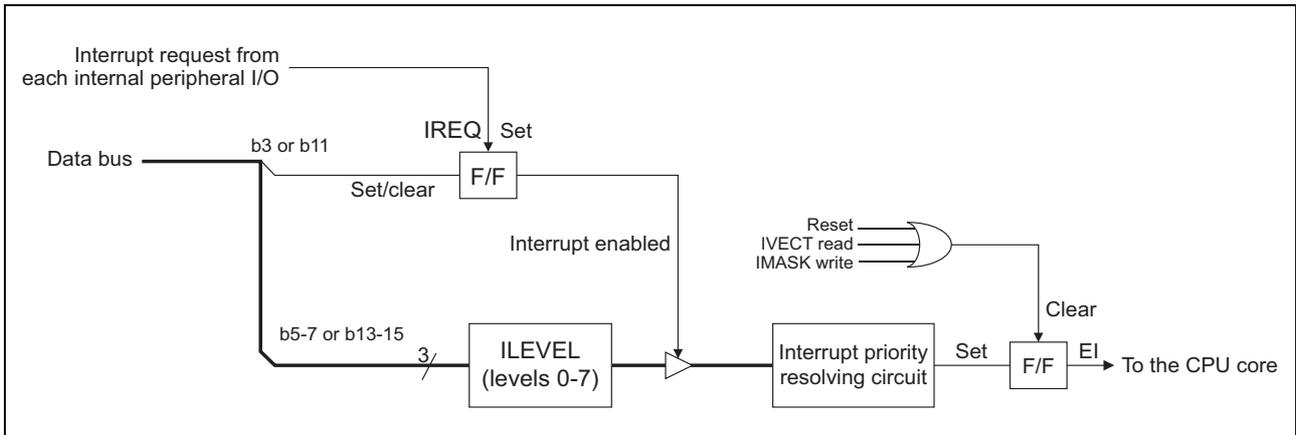


Figure 4.4.2 Interrupt Sources Signaled to the ICU (Edge Type)

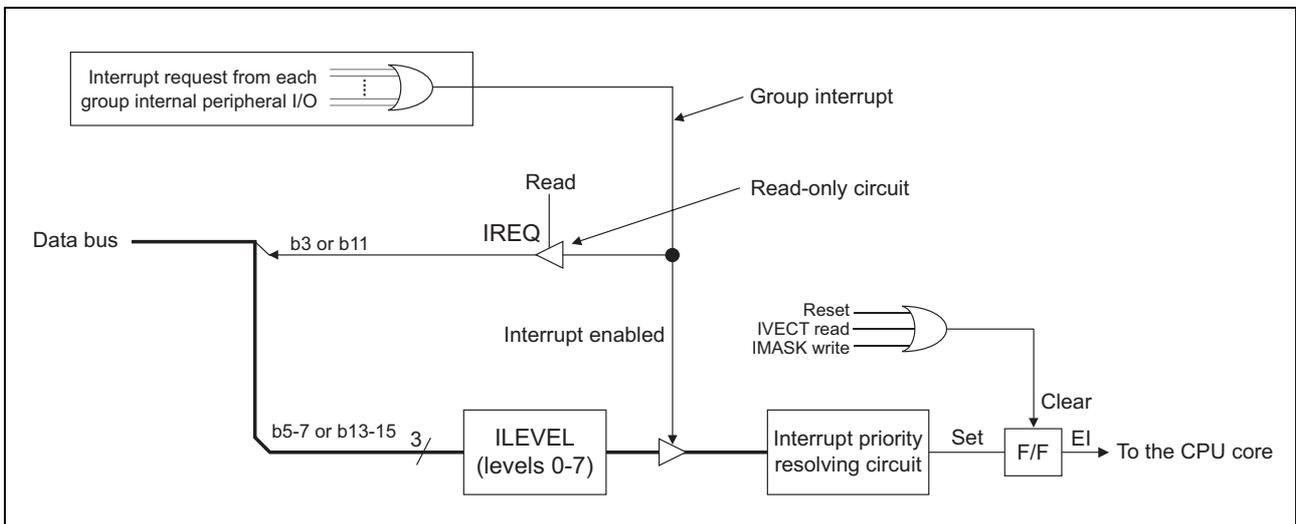


Figure 4.4.3 Interrupt Sources Signaled to the ICU (Level Type)

(1) Edge-type interrupt sources

For edge-type interrupt sources, the ICU's internal Interrupt Control Register interrupt request bit can be set and cleared by writing to the register in software. The bit can also be cleared by reading out the Interrupt Vector Register (IVECT).

(2) Level-type interrupt sources

The level-type interrupt sources differ from the other type in that two or more interrupt sources are grouped into a single source as they are mapped to the interrupt vector table. These interrupt sources are controlled for interrupt request and resolved for interrupt input by an interrupt control register in each internal peripheral I/O before being fed into the ICU. Therefore, the ICU's internal interrupt request bits function only as an interrupt-enabled interrupt request judgment bit and cannot be set or reset by writing in software. The diagram below shows example configuration of Interrupt request status register and Mask register.

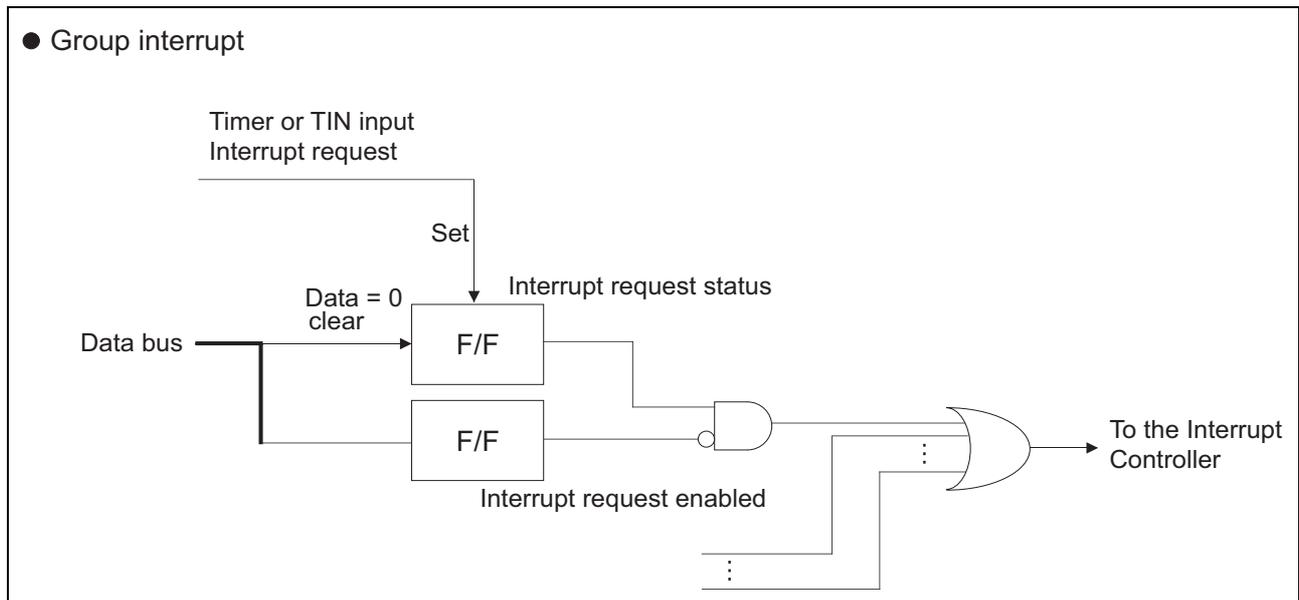
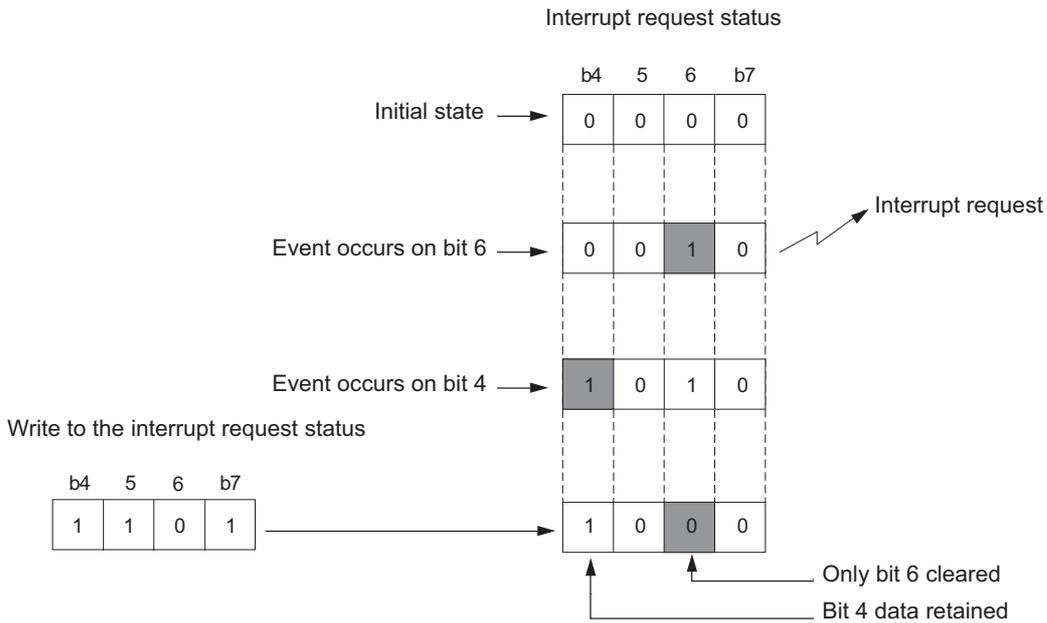


Figure 4.4.4 Example configuration of Interrupt request status register and Mask register (MJT)

- Internal peripheral I/O interrupt request status bit

This status bit is used to determine whether any interrupt is requested. This bit is set in hardware when an interrupt request occurs, but cannot be set in software. The status bit is cleared by writing as "0" but does not change its state by writing as "1". Because this bit behaves independently of interrupt mask bits, it can also be used to verify the operation of peripheral functions. During interrupt processing, make sure that of the grouped interrupt request status flags, only the flag for which interrupt has been serviced is cleared. Clearing any flag for which interrupt has not been serviced yet results in the pending interrupt requests being also cleared. When all interrupt request status flags for grouped interrupt sources are cleared, the level-type interrupt request for that interrupt group is cleared.

● Example for clearing interrupt request status



● Program example

- To clear the Interrupt Request Status Register 0 (ISTREG) interrupt request status 1, ISTAT1 (0x02 bit)



```
ISTREG = 0xfd; /* Clear ISTAT1 (0x02 bit) only */
```

To clear an interrupt request status, always be sure to write "1" to all other interrupt request status bits. At this time, avoid using a logic operation like the one shown below. Because it requires three step-ISTREG read, logic operation and write, if another interrupt request occurs between the read and write, status may be inadvertently cleared.



```
ISTREG &= 0xfd; /* Clear ISTAT1 (0x02 bit) only */
```

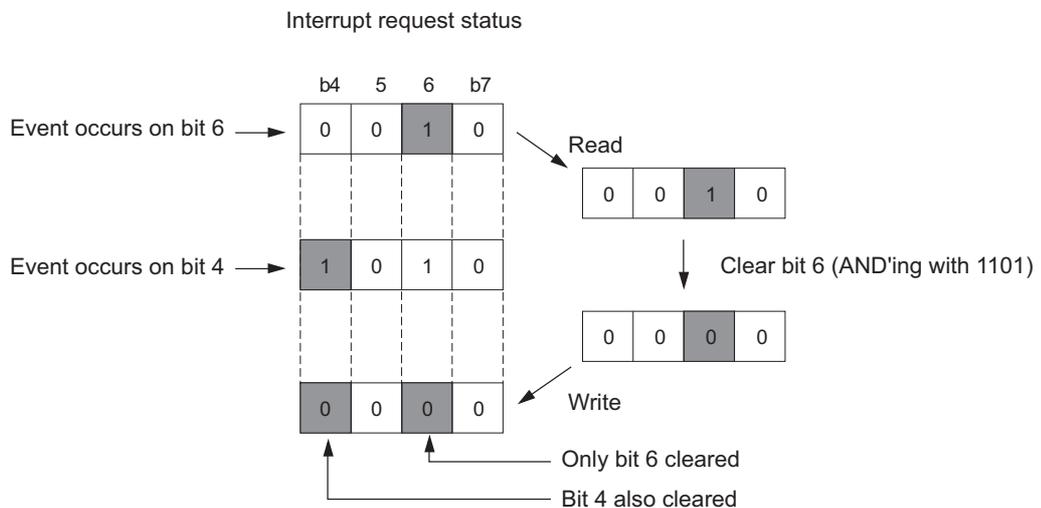


Figure 4.4.5 Example for Clearing Interrupt Status

5. Sample Programming Code

The sample program of a startup program (startup.ms) is shown below.

Depending on the contents of an application program, there may be some processing that should be added or deleted besides the following processing,

```

1  ;"FILE COMMENT" *****
2  ;*      M32R C Programming                      Rev. 1.01
3  ;*      < Sample startup Program for 32170/32171/32174/32176 >
4  ;*
5  ;*      Copyright (c) 2003 Renesas Technology Corporation
6  ;*      And Renesas Solutions Corporation
7  ;*      All Rights Reserved
8  ;*****
9  ;
10 ;*****
11 ; EIT Vector Entry
12 ;*****
13 ;
14     .SECTION          EITVECT, CODE, ALIGN=4
15 ;
16     .EXPORT reset, EIT_reset, EIT_loop
17 ;
18 reset:
19     BRA                EIT_reset:24          ; H'0000 0000 Reset Interrupt (RI)
20     NOP
21     NOP
22     NOP
23     NOP
24     NOP
25     NOP
26     BRA                EIT_loop:24          ; H'0000 0010 System Break Interrupt (SBI)
27     NOP
28     NOP
29     NOP
30     NOP
31     NOP
32     NOP
33     BRA                EIT_loop:24          ; H'0000 0020 Reserved Instruction Exception (RIE)
34     NOP
35     NOP
36     NOP
37     NOP
38     NOP
39     NOP
40     BRA                EIT_loop:24          ; H'0000 0030 Address Exception (AE)
41     NOP
42     NOP
43     NOP
44     NOP
45     NOP
46     NOP
47     BRA                EIT_loop:24          ; H'0000 0040 Trap 0
48     BRA                EIT_loop:24          ; H'0000 0044 Trap 1
49     BRA                EIT_loop:24          ; H'0000 0048 Trap 2
50     BRA                EIT_loop:24          ; H'0000 004C Trap 3
51     BRA                EIT_loop:24          ; H'0000 0050 Trap 4
52     BRA                EIT_loop:24          ; H'0000 0054 Trap 5
53     BRA                EIT_loop:24          ; H'0000 0058 Trap 6
54     BRA                EIT_loop:24          ; H'0000 005C Trap 7
55     BRA                EIT_loop:24          ; H'0000 0060 Trap 8
56     BRA                EIT_loop:24          ; H'0000 0064 Trap 9
57     BRA                EIT_loop:24          ; H'0000 0068 Trap 10
58     BRA                EIT_loop:24          ; H'0000 006C Trap 11
59     BRA                EIT_loop:24          ; H'0000 0070 Trap 12
60     BRA                EIT_loop:24          ; H'0000 0074 Trap 13
61     BRA                EIT_loop:24          ; H'0000 0078 Trap 14
62     BRA                EIT_loop:24          ; H'0000 007C Trap 15
63     BRA                EIT_ei:24           ; H'0000 0080 External Interrupt (EI)
64 ;
65     .SECTION          PROTECTID, DATA, ALIGN=1
66     .DATA.B           H'FF,H'FF,H'FF,H'FF,H'FF,H'FF,H'FF,H'FF ; H'0000 0084 Protect ID
67     .DATA.B           H'FF,H'FF,H'FF,H'FF,H'FF,H'FF,H'FF,H'FF ;
68 ;
69 ;
70 ;*****
71 ; ICU Vector Table
72 ;*****
73 ;
74     .SECTION          ICUVECT, DATA, ALIGN=4
75 ;
76 ;
77 vectbl:
78     .DATA.W           EIT_reset            ; H'0000 0094 MJT Input Interrupt 4:TIN3-TIN6
79     .DATA.W           EIT_reset            ; H'0000 0098 MJT Input Interrupt 3:TIN20-TIN23

```

```

80      .DATA.W      EIT_reset      ; H'0000 009C  MJT Input Interrupt 2:TIN12-TIN19
81      .DATA.W      EIT_reset      ; H'0000 00A0  MJT Input Interrupt 1:TIN0-TIN2
82      .DATA.W      EIT_reset      ; H'0000 00A4  MJT Input Interrupt 0:TIN7-TIN11
83      .DATA.W      EIT_reset      ; H'0000 00A8  MJT Output Interrupt 7:TMS0,TMS1
84      .DATA.W      EIT_reset      ; H'0000 00AC  MJT Output Interrupt 6:TOP8,TOP9
85      .DATA.W      EIT_reset      ; H'0000 00B0  MJT Output Interrupt 5:TOP10
86      .DATA.W      EIT_reset      ; H'0000 00B4  MJT Output Interrupt 4:TIO4-TIO7
87      .DATA.W      EIT_reset      ; H'0000 00B8  MJT Output Interrupt 3:TIO8,TIO9
88      .DATA.W      EIT_reset      ; H'0000 00BC  MJT Output Interrupt 2:TOP0-TOP5
89      .DATA.W      EIT_reset      ; H'0000 00C0  MJT Output Interrupt 1:TOP6,TOP7
90      .DATA.W      EIT_reset      ; H'0000 00C4  MJT Output Interrupt 0:TIO0-TIO3
91      .DATA.W      EIT_reset      ; H'0000 00C8  DMAC0-4 Interrupt:DMA0-DMA4
92      .DATA.W      EIT_reset      ; H'0000 00CC  SIO1 Receive Interrupt
93      .DATA.W      EIT_reset      ; H'0000 00D0  SIO1 Transmit Interrupt
94      .DATA.W      EIT_reset      ; H'0000 00D4  SIO0 Receive Interrupt
95      .DATA.W      EIT_reset      ; H'0000 00D8  SIO0 Transmit Interrupt
96      .DATA.W      EIT_reset      ; H'0000 00DC  A-D0 Conversion Interrupt
97      .DATA.W      EIT_reset      ; H'0000 00E0  TID0 Output Interrupt
98      .DATA.W      EIT_reset      ; H'0000 00E4  TOD0 Output Interrupt
99      .DATA.W      EIT_reset      ; H'0000 00E8  DMAC5-9 Interrupt:DMA5-DMA9
100     .DATA.W      EIT_reset      ; H'0000 00EC  SIO2,3 Transmit/Receive Interrupt
101     .DATA.W      EIT_reset      ; H'0000 00F0  RTD Interrupt
102     .DATA.W      EIT_reset      ; H'0000 00F4  TID1 Output Interrupt
103     .DATA.W      EIT_reset      ; H'0000 00F8  TOD1,TOM0 Output Interrupt
104     .DATA.W      EIT_reset      ; H'0000 00FC  SIO4,5 Transmit/Receive Interrupt
105     .DATA.W      EIT_reset      ; H'0000 0100  A-D1 Conversion Interrupt
106     .DATA.W      EIT_reset      ; H'0000 0104  TID2 Output Interrupt
107     .DATA.W      EIT_reset      ; H'0000 0108  TML1 Input Interrupt
108     .DATA.W      EIT_reset      ; H'0000 010C  CAN0 Transmit/Receive & Error Interrupt
109     .DATA.W      EIT_reset      ; H'0000 0110  CAN1 Transmit/Receive & Error Interrupt
110     ;
111     ;*****
112     ; Set SFR Address
113     ;*****
114     ;
115     .SECTION      SFR, DATA, ALIGN=1
116     .RES.B        H'4000
117     ;
118     IMASK          .EQU    H'00800004      ; IMASK Address
119     ;
120     ;*****
121     ; External Interrupt Handler
122     ;*****
123     ;
124     .SECTION      EIT_P, CODE, ALIGN=4
125     ;
126     .EXPORT EIT_ei
127     ;
128     EIT_ei:
129     ST            R0,@-R15          ; PUSH R0                (R0)
130     ;
131     ST            R1,@-R15          ; PUSH R1                (R1)
132     ST            R2,@-R15          ; PUSH R2                (R2)
133     LD24          R0,#IMASK         ;
134     ST            R3,@-R15          ; PUSH R3                (R3)
135     LDI           R3,#H'40          ;
136     ;
137     MVFACHI       R1                ; PUSH Accumulator
138     MVFACLO       R2                ;
139     ST            R1,@-R15          ;                        (AccH)
140     ST            R2,@-R15          ;                        (AccL)
141     ;
142     LDB           R2,@R0             ; Read IMASK(H'0080 0004) Register
143     ADDI          R0,#-4            ;
144     ST            R4,@-R15          ; PUSH R4                (R4)
145     MVFC          R4,PSW            ;
146     ST            R5,@-R15          ; PUSH R5                (R5)
147     MVFC          R5,BPC            ;
148     ST            R6,@-R15          ; PUSH R6                (R6)
149     ;
150     LDH           R1,@R0             ; Read IVECT(H'0080 0000) Register
151     ;
152     LD24          R0,#IMASK         ; Overwrite IMASK Register
153     LDB           R6,@R0             ;
154     STB          R6,@R0             ;
155     ;
156     ST            R7,@-R15          ; PUSH R7                (R7)
157     ST            R14,@-R15         ; PUSH Link Register     (R14)
158     ;
159     LD            R1,@R1             ; Read ICU Vector Table
160     ;
161     ST            R4,@-R15          ; PUSH PSW                (PSW)
162     ST            R5,@-R15          ; PUSH BPC                (BPC)
163     ST            R2,@-R15          ; PUSH IMASK Register    (IMASK)
164     ;
165     MVTC          R3,PSW            ; Enable Interrupt
166     ;

```

```

167         JL             R1                ; Call Interrupt Handler
168 ;
169         LDI            R5, #H'00         ;
170         MVTC           R5, PSW          ; Disable Interrupt
171 ;
172         LD24           R3, #IMASK        ;
173         LD              R2, @R15+        ; POP  IMASK          (IMASK)
174         LD              R1, @R15+        ; POP  BPC            (BPC)
175         LD              R0, @R15+        ; POP  PSW            (PSW)
176 ;
177         LD              R14, @R15+       ; POP  Link Register  (R14)
178         STB            R2, @R3          ;
179         LD              R7, @R15+       ; POP  R7              (R7)
180         MVTC           R1, BPC          ;
181         LD              R6, @R15+       ; POP  R6              (R6)
182         LD              R5, @R15+       ; POP  R5              (R5)
183         LD              R4, @R15+       ; POP  R4              (R4)
184 ;
185         LD              R2, @R15+       ; POP  Accumulator    (AccL)
186         LD              R1, @R15+       ;                      (AccH)
187         MVTC           R0, PSW          ;
188         MVTACLO        R2              ;
189         MVTACHI        R1              ;
190 ;
191         LD              R3, @R15+       ; POP  R3              (R3)
192         LD              R2, @R15+       ; POP  R2              (R2)
193         LD              R1, @R15+       ; POP  R1              (R1)
194 ;
195         LD              R0, @R15+       ; POP  R0              (R0)
196 ;
197         RTE
198 ;
199 ;*****
200 ; EIT_loop
201 ;*****
202 ;
203 EIT_loop:
204         BRA            EIT_loop
205 ;
206 ;*****
207 ; Start Up Program
208 ;*****
209 ;
210         .IMPORT $main
211 ;
212         .SECTION       C, DATA, ALIGN=4
213         .SECTION       D, DATA, ALIGN=4
214         .SECTION       B, DATA, ALIGN=4
215         .SECTION       ROM_D, DATA, ALIGN=4
216 ;
217 ;*****
218 ; Set Interrupt Stack
219 ;
220         .SECTION       SPINT, DATA, ALIGN=4
221         .RES.B         2048             ; Interrupt Stack Area
222 ;
223 ;*****
224 ; Set User Stack
225 ;
226         .SECTION       SPUSR, DATA, ALIGN=4
227         .RES.B         2048             ; User Stack Area
228 ;
229 ;*****
230 ; Startup & Exit
231 ;
232         .SECTION       P, CODE, ALIGN=4
233 EIT_reset:
234         LDI            R0, #H'00         ;
235         MVTC           R0, PSW          ; Disable Interrupt
236         LDI            R0, #7           ;
237         LD24           R1, #IMASK        ;
238         STB            R0, @R1          ;
239 ;
240 ;         LD24           R1, #(SPUSR+sizeof(SPUSR)) ;
241 ;         LD24           R2, #(SPINT+sizeof(SPINT)) ;
242 ;         MVTC           R1, SPU          ; Set User Stack Pointer
243 ;         MVTC           R2, SPI          ; Set Interrupt Stack Pointer
244 ;
245 ;*****
246 ; Clear B Section
247 ;
248         LD24           R5, #sizeof(B)
249         BLEZ           R5, loop_cnt0
250         LD24           R4, #B
251         LDI            R0, #0
252 loop0:
253         STB            R0, @R4

```

```

254         ADDI          R4, #1
255         ADDI          R5, #-1
256         BNEZ         R5, loop0
257 loop_cnt0:
258         ;
259         ;+++++
260         ; Data Set (ROM_D Section => D Section)
261         ;
262         LD24          R6, #sizeof(ROM_D)
263         BLEZ         R6, loop_cnt1
264         LD24          R4, #D
265         LD24          R5, #ROM_D
266 loop1:
267         LDB          R0, @R5
268         STB          R0, @R4
269         ADDI          R4, #1
270         ADDI          R5, #1
271         ADDI          R6, #-1
272         BNEZ         R6, loop1
273 loop_cnt1:
274         ;
275         ;+++++
276         ; Set Base Register
277         ;
278         ; .EXPORT      __REL_BASE13
279         ; .EXPORT      __REL_BASE12
280         ; .EXPORT      __REL_BASE11
281         ; __REL_BASE13 .EQU 0x00808000
282         ; __REL_BASE12 .EQU 0x00808000
283         ; __REL_BASE11 .EQU 0x00808000
284         ;
285         ; SETH         R13, #HIGH(__REL_BASE13)
286         ; OR3         R13, R13, #LOW(__REL_BASE13)
287         ;
288         SETH         R12, #HIGH(__REL_BASE12)
289         OR3         R12, R12, #LOW(__REL_BASE12)
290         ;
291         SETH         R11, #HIGH(__REL_BASE11)
292         OR3         R11, R11, #LOW(__REL_BASE11)
293         ;
294         ;+++++
295         ; Call main()
296         ;
297         ; LDI          R0, #H'C0           ; Enable Interrupt, Use User Stack
298         ; LDI          R0, #H'40           ; Enable Interrupt, Use Interrupt Stack
299         MVTC         R0, PSW
300         ;
301         BL           $main                 ; Call C(main) Routine
302         ;
303         ;+++++
304         ;
305 endless:
306         BRA          endless                 ; Dummy
307         ;
308         ;+++++
309         ; Set/Clear IE Flag Routine
310         ;
311         .EXPORT      $EnInt
312         .EXPORT      $DisInt
313 $EnInt:
314         MVFC         R0, PSW
315         OR3         R0, R0, #H'0040
316         MVTC         R0, PSW
317         JMP          R14
318         ;
319 $DisInt:
320         MVFC         R0, PSW
321         AND3         R0, R0, #H'FFBF
322         MVTC         R0, PSW
323         JMP          R14
324         ;
325         .END

```

6. Reference of Document

- 32170/32174 Group User's Manual Rev.2.10
- 32171 Group User's Manual Rev.2.00
- 32176 Group User's Manual Rev.1.01
- M32R Family Software Manual Rev.1.20
- M3T-CC32R V.4.30 User's Manual (Compiler)
- M3T-AS32R V.4.30 User's Manual (Assembler)

(Please get the latest one from Renesas Technology Corp. website.)

7. Website and Support Center

- Renesas Technology Corp. website
<http://www.renesas.com/>
- Customer Support Center for all Products and Technical Support Center for M32R Family
Customer Support Center: csc@renesas.com

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan 13, 2006	-	First Edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of non-flammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein. The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors. Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (<http://www.renesas.com>).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination. Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.