SH7262/SH7264 Group

Using Deep Standby Mode in Power-down Mode

R01AN0548EJ0101
Rev. 1.01
Feb. 23, 2011

Summary

This application note describes an example of using the SH7262/SH7264 deep standby mode in power-down mode.

Target Device

SH7264 MCU (In this document, SH7262/SH7264 are described as "SH7264".)

Contents

1. Introduction

1.1 Specifications

This application note describes the application using the SH7264 deep standby mode. Operates the realtime clock during deep standby mode, and stores data to be retained in the data-retention internal RAM. An example of turning OFF the external device is also described.

1.2 Modules Used

- Power-down mode
- Realtime Clock (RTC)
- Interrupt controller (INTC)
- General-purpose I/O ports
- Serial Communication Interface with FIFO (SCIF)

1.3 Applicable Conditions

MCU	SH7262/SH7264
Operating Frequency	Internal clock: 144 MHz Bus clock: 72 MHz Peripheral clock: 36 MHz
Integrated Development Environment	Renesas Electronics Corporation High-performance Embedded Workshop Ver.4.07.00
C Compiler	Renesas Electronics SuperH RISC engine Family C/C++ compiler package Ver.9.03 Release 00
Compiler Options	Default setting in the High-performance Embedded Workshop (-cpu=sh2afpu -fpu=single -debug -gbr=auto -global_volatile=0 -opt_range=all -infinite_loop=0 -del_vacant_loop=0 -struct_alloc=1)

1.4 Related Application Note

Refer to the related application notes as follows:

- SH7262/SH7264 Group Example of Initialization

1.5 About Active-low Pins (Signals)

The symbol "#" suffixed to the pin (or signal) names indicates that the pins (or signals) are active-low.

2. Applications

This chapter describes an overview of deep standby mode including the pin connection with an external device, and the flow chart of the sample program of deep standby mode setting procedure in this application. The sample program does not assume to turn OFF the external device in deep standby mode, however, an example of turning OFF the external device is described for reference.

2.1 Deep Standby Mode Operation

2.1.1 Overview of Power-down Modes

The SH7264 power-down mode has four modes: sleep mode, software standby mode, deep standby mode, and module standby mode. Table 1 lists the status of each mode and how to wake up the MCU.

As deep standby mode used in this application stops supplying power to modules inside LSI, will reduce power consumption greatly compared to other power-down modes. Data can be stored in the data-retention internal RAM.

Table 1 Power-Down Modes

Power-Down Modes	States									How to Wake the MCU up
	CPG	CPU	CPU register	High-speed internal RAM Cache Memory	Large-capacity internal RAM (Data-retention RAM included)	Internal Peripherals	RTC	Power Supply	External Memory	
Sleep Mode	ON	OFF	States held	ON	ON	ON	ON (1)	ON	Auto-refresh	• Interrupt • Manual reset • Power-on reset • DMA address error
Software Standby Mode	OFF	OFF	States held	OFF (Data is retained) (2) (3)	OFF (Data is retained) (2) (4)	OFF	ON (1)	ON	Self-refresh	• NMI interrupt • IRQ interrupt • Power-on reset
Deep Standby Mode	OFF	OFF	OFF	OFF (Data is NOT retained)	OFF (Data in the data-retention RAM is retained (5)	OFF	ON (1)	OFF	Self-refresh	• NMI interrupt (6) • Power-on reset (6) • Realtime clock alarm interrupt (6) • Change pins to wake up the MCU (6)
Module Standby Mode	ON	ON	States held	ON	ON	Specified module is OFF	OFF	ON	Auto-refresh	• Clear the MSTP bit by 0 • Power-on reset (Only for the user debugging interface, and direct memory access controller)

(1) The realtime clock is ON when the START bit in the RCR2 register is 1. When waking the MCU up from deep standby mode, the status cannot be retained. Thus, configure the realtime clock again.

(2) When waking up the MCU from software standby mode by the power-on reset, the retained data is initialized.

(3) Disable the RAME bit in the SYSCR1 register or the RAMWE bit in the SYSCR2 register to keep retaining data in the high-speed internal RAM, when the MCU is woke up from software standby mode by the power-on reset.

(4) Disable the VRAME bit in the SYSCR3 register or the VRAMWE bit in the SYSCR4 register to keep retaining data in the large-capacity internal RAM (data-retention RAM included).

(5) Set bits RRAMKP3 to RRAMKP0 in the RRAMKP register to 1 to retain data in the target area in the data-retention internal RAM when transition to deep standby mode. Note that the data is initialized when waking up the MCU from deep standby mode by the power-on reset. Bits RRAMKP3 and RRMKP2 can be used only on MCU with 640-KB internal RAM.

(6) The MCU is woke up from deep standby mode by interrupts (NMI, realtime clock alarm interrupt), reset (power-on reset) and changing pins to wake up the MCU (PC8 to PC5, PG11, PG10, PJ3, and PJ1). When waking up the MCU not by reset, power-on reset exception handling is executed instead of interrupt exception handling. PG11 and PG10 operate as wake-up pins only on MCU with 640-KB internal RAM.

2.1.2 Transition Procedure to Deep Standby Mode

Figure 1 shows the transition procedure to deep standby mode. Configure the data-retention internal RAM and realtime clock. Specify the wake-up factors in deep standby mode and how to activate that mode.

In case of setting registers related to power-down mode, when executing the consecutive instructions after reflecting the setting in registers, the dummy read is required between the register writing instruction and the consecutive instructions. Note that when the NMI, IRQ interrupts and manual reset occurs in parallel with executing the SLEEP instruction, the interrupt may be accepted, and the MCU may be woke up from deep standby mode.

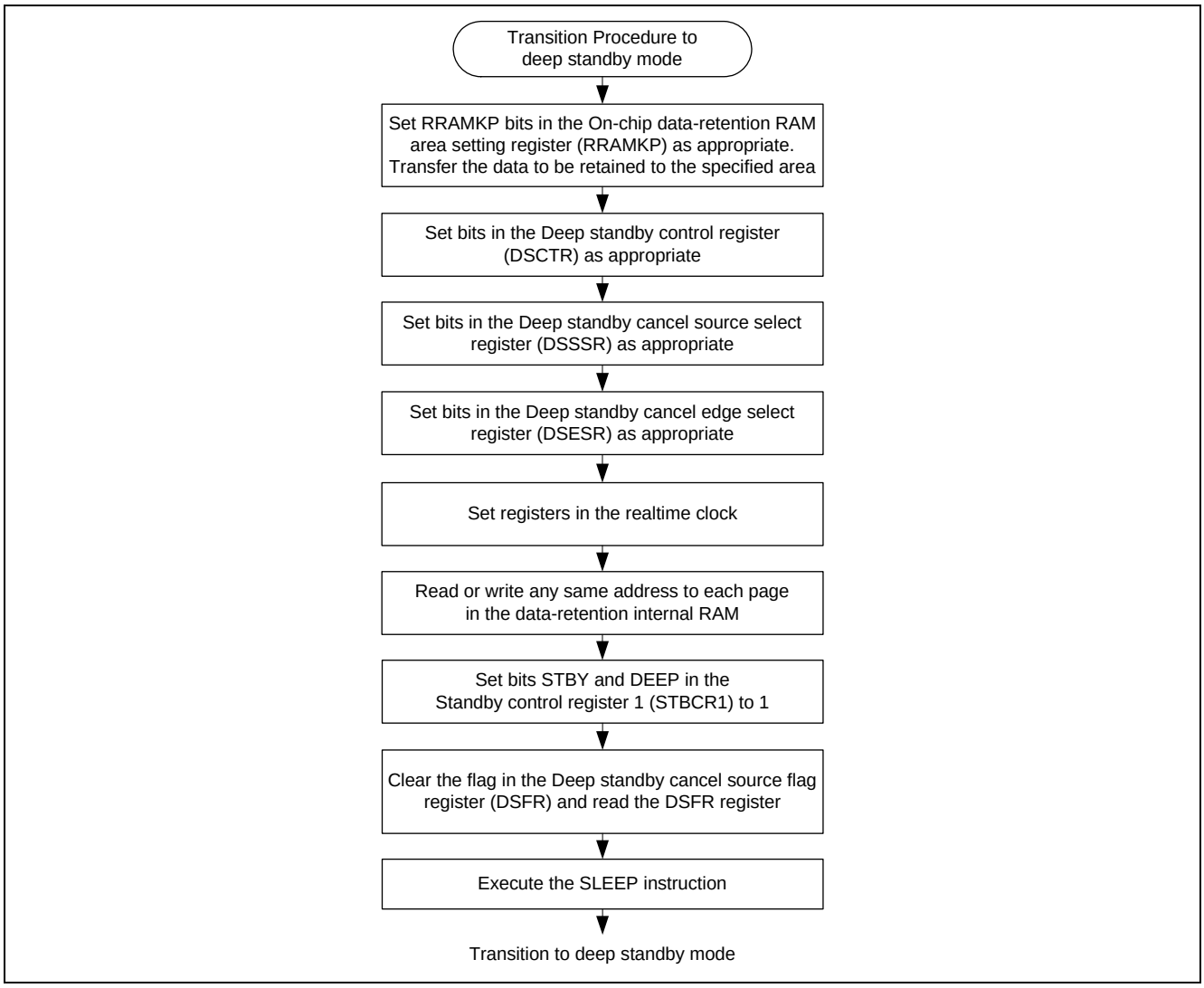


Figure 1 Transition Procedure to Deep Standby Mode

2.1.3 Pin States in Deep Standby Mode

State of output pins in deep standby mode can be specified as "states retained" or Hi-Z. However, the pin state cannot be specified individually. Table 2 lists pin states (excerpt). Table 3 lists the registers to specify pin states in deep standby mode. For details on pin state, refer to Appendix A, Pin States in the SH7262 Group, SH7264 Group, Hardware Manual.

After waking up the MCU from deep standby mode, pin states are retained (except some pins) until the IOKEEP bit in the DSFR register is cleared by 0. For pin states after waking up from deep standby mode, refer to 2.1.6 Activating from Data-Retention Internal RAM and Releasing the Pin State.

Table 2 Pin States (Excerpt)

Pin Functions			Pin States (O: Output, I: Input, Z: High impedance, H: High, L: Low)					
Type	Pin Name		Default	Power-on reset	Wake up from deep standby mode to clear IOKEEP bit			Deep standby mode
					EBUSKEEPE bit		Reset state (1)	
					0	1		
Clock	EXTAL	Clock operating modes 0 and 2	I	I	I			I/Z (2)
		Clock operating modes 1 and 3	Z	Z	Z			Z
	XTAL		O	O	O			O/L (2)
	CKIO		O	O	O	O/Z (3)		O/Z (3)
Address bus	A25 to A21, A0		O	— (4)	O/Z (5)			O/Z (5)
	A20 to A1	Boot mode 0	O	Z	O	O/Z (5)		O/Z (5)
		Boot modes 1 to 3	O	— (4)	O/Z (5)			O/Z (5)
Data bus	D15 to D0	Boot mode 0	I/Z	Z	I/Z		Z	Z
			O/Z	Z	O/Z	Z		Z
		Boot mode 0	I/Z	— (4)	—			Z
			O/Z	— (4)	Z			Z
Bus control	CS0#	Boot mode 0	O	Z	O	H/Z (5)		H/Z (5)
		Boot modes 1 to 3	O	— (4)	H/Z (5)			H/Z (5)
	CS6# to CS1#, CE1A#, CE1B#, CE2A#, CE2B#		O	— (4)	H/Z (5)			H/Z (5)
	RD#	Boot mode 0	O	Z	O	H/Z (5)		H/Z (5)
		Boot modes 1 to 3	O	— (4)	H/Z (5)			H/Z (5)
	RD/WR#, BS#, ICIOWR#/AH#, ICIORD#, WE1#/DQMLU/WE#, WE0#/DQMLL		O	— (4)	H/Z (5)			H/Z (5)
	RAS#, CAS#, CKE		O	— (4)	O/Z (6)			O/Z (6)
	WAIT#, IOIS16#		I	— (4)	—			Z
Realtime clock	RTC_X1		I/Z	I	I/Z (7)			I/Z (7)
	RTC_X2		O/H	O	O/H (7)			O/H (7)
General-purpose I/O ports	PB21 (A21)		I	Z	Z			Z
			O/Z	Z	O/Z (8)			O/Z (8)

(1) The MCU enters power-on reset state for a certain period after waking up from deep standby mode.

(2) The pin state is same as the RCKSEL bit in the Control register (RCR5).

(3) The pin state is same as the CKOEN bit in the Frequency control register (FRQCR).

(4) The pin state is as default (general-purpose I/O port).

(5) The pin state is same as the HIZMEM bit in the Common control register (CMNCR).

(6) The pin state is same as the HIZCNT bit in the Common control register (CMNCR).

(7) The pin state is same as the RTCEN bit in the Control register (RCR2).

(8) The pin state is same as the HIZ bit in the Standby control register 3 (STBCR3).

Table 3 Registers to Specify Pin States in Deep Standby Mode

Register Name	Address	Bit Name	Setting	Description
Frequency control register (FRQCR)	H'FFFE 0010	CKOEN [1:0]	B'00 (or B'11)	Specifies the CKIO pin to output low level or high level (or Hi-Z) in deep standby mode
Standby control register (STBCR3)	H'FFFE 0408	HIZ	B'0 (or B'1)	Specifies the output pin to retain the pin state (or Hi-Z) in deep standby mode (note)
Common control register (CMNCR)	H'FFFC 0000	HIZMEM	B'1 (or B'0)	Specifies states of pins A25 to A0, BS#, CSn#, CE2x#, RD/WR#, WEn#/DQMx/AH#, and RD# as drive (or Hi-Z) in deep standby mode
		HIZCNT	B'1 (or B'0)	Specifies states of pins CKE, RAS#, and CAS# as drive (or Hi-Z)
Control register 5 (RCR5)	H'FFFE 6026	RCKSEL	B'0 (or B'1)	Specifies the RTC operating clock as the RTC_X1 (or EXTAL)
Control register 2 (RCR2)	H'FFFE 601E	RTCEN	B'1 (or B'0)	Specifies the RTC_X1 pin internal crystal units to operate/enable to input an external clock (or stop the crystal unit/disable to output an external clock)

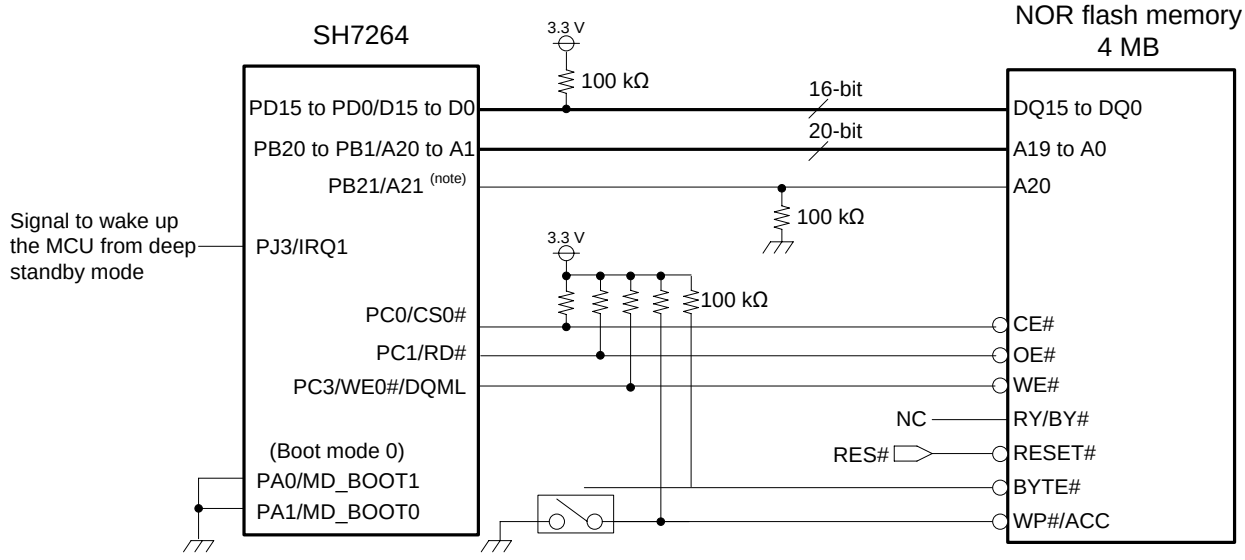
Note: Set this register when the TME bit in the WTSCR register (watchdog timer) is 0.

2.1.4 Pin Connection Example When Not Turning OFF the External Device

Figure 2 shows the pin connection example when not turning OFF the external device in deep standby mode. This example assumes to connect an NOR flash memory and activate the MCU in boot mode 0.

When not turning OFF the external device, the pin state is set to be retained. Make sure not to flow unnecessary electric current.

As the pull-down resistor is connected to PB21/A21 pin in Figure 2, the sample program specifies the PB21/A21 pin function as general-purpose I/O port before transition to deep standby mode. Pull-down resistors are connected to pins CS0#, RD#, WE0#, and D15 to D0, however no processing is required, since these pins retain pin states to high level. Fore details on pin states, refer to Table 2. 100 kΩ pull-up/pull-down resistor is recommended to reduce power consumption.



Note: As the PB21/A21 pin is configured as PB21 by default, pull it down when using it as an address pin on booting the MCU.

Figure 2 Pin Connection Example (External Device is Not Turned OFF)

2.1.5 Waking Up the MCU from Deep Standby Mode

Table 4 lists wake-up factors from deep standby mode.

Set pin levels of the NMI, and all other pins to select as wake-up factors when transition to deep standby mode as follows:

- Trigger to wake up the MCU is the rising edge: low level
- Trigger to wake up the MCU is the falling edge: high level

Set levels of pins to use as factors to wake up the MCU when the oscillation stabilization time elapsed as following. When waking up the MCU by reset, retain the RES# pin level as low until the oscillator is stabilized.

- Trigger to wake up the MCU is the rising edge: high level
- Trigger to wake up the MCU is the falling edge: low level

Table 4 Factors to Wake Up the MCU from Deep Standby Mode

Wake-up Factor	Trigger	Related Register	Remarks
RES#	low level	–	
NMI interrupt	Rising edge or falling edge	Deep standby cancel edge select register (DSESR)	
Realtime Clock Alarm Interrupt	Specified time elapsed	Deep standby cancel source select register (DSSSR)	Interrupt is accepted regardless of the interrupt priority.
Pins to wake up the MCU	Rising edge or falling edge	Deep standby cancel edge select register (DSESR) Deep standby cancel source select register (DSSSR)	

Figure 3 shows the flow chart of waking up the MCU from deep standby mode.

When waking up the MCU by reset, the power-on reset exception handling is executed according to the boot mode on reset, as same as usual power-on reset. Otherwise, the power-on reset execution handling is executed after the oscillation stabilization time counted LSI inside is elapsed. The area to read the program counter (PC) and stack pointer (SP) varies according to the RAMBOOT bit value in the DSCTR register. For details on activation, refer to 2.1.6 Activating from Data-Retention Internal RAM and Releasing the Pin State.

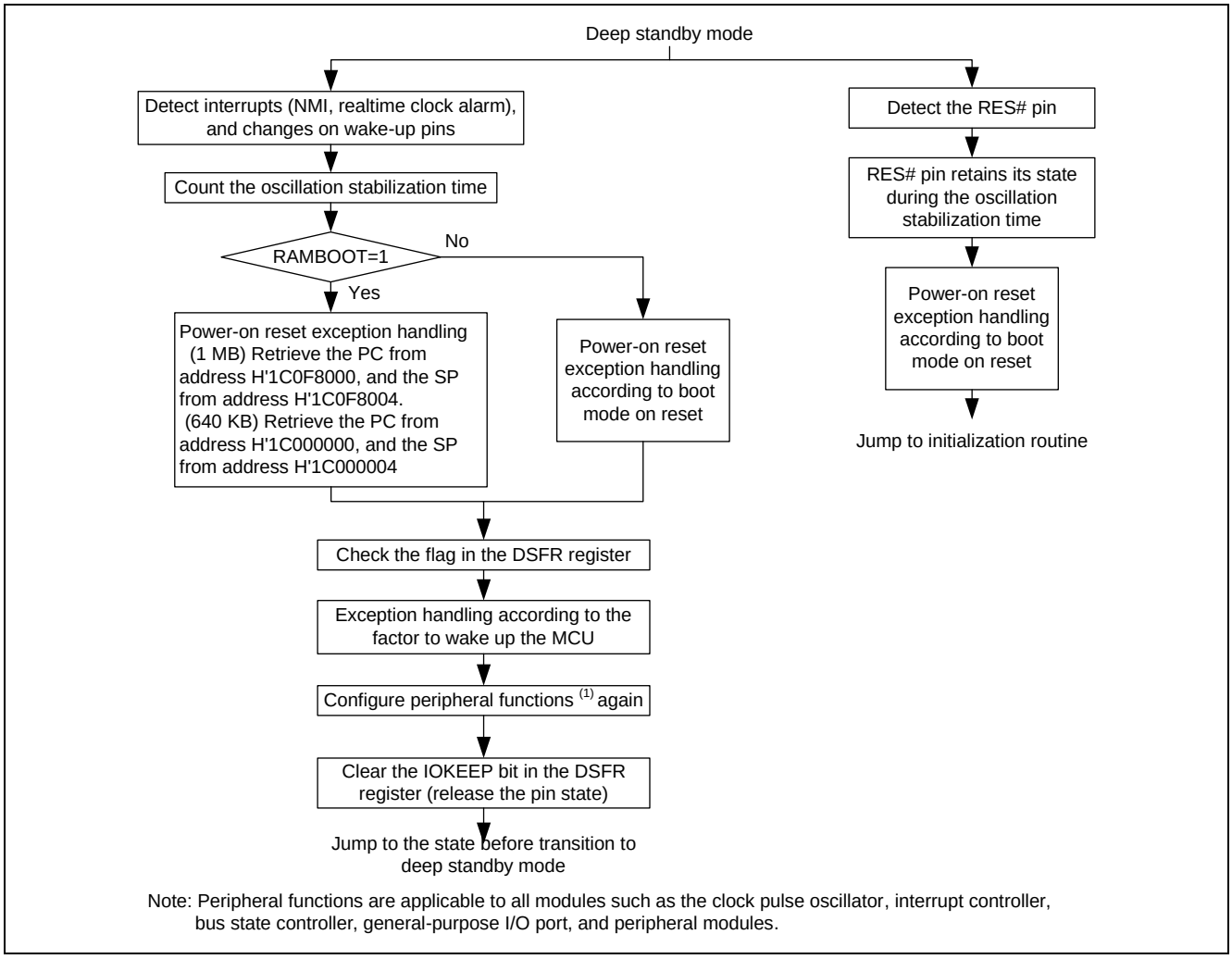


Figure 3 Flow Chart of Wake-up from Deep Standby Mode

2.1.6 Activating from Data-Retention Internal RAM and Releasing the Pin State

After the wake-up from deep standby mode, set the RAMBOOT bit in the DSCTR register whether to activate the MCU from an external memory as usual boot mode, or activate the MCU from the data-retention internal RAM.

Specify the activation from the data-retention RAM when the external memory cannot be accessed just after the wake-up from deep standby mode, or when distinguishing the boot program after the wake-up from deep standby mode from the boot program by power-on reset. Table 5 lists the activation procedure after the wake-up from deep standby mode and retained pin state.

Table 5 Activation Procedure after the Wake-up from Deep Standby Mode and Retained Pin State

EBUSKEEPE bit in the DSCTR register	RAMBOOT bit in the DSCTR register	Activate by	Pin State after the Wake-up from Deep Standby Mode
0	0	External memory	External memory control pin does not retain its state. Other pins release states when the IOKEEP bit is cleared.
0	1	Data-retention internal RAM	External memory control pin does not retain its state. External memory control pin release its state after the wake-up from deep standby mode. Other pins release states when the IOKEEP bit is cleared.
1	0	–	Setting prohibited
1	1	Data-retention internal RAM	External memory control pin retains its state. All pins (including external memory control pin) release states when the IOKEEP bit is cleared

The EBUSKEEPE bit in the DSCTR register and the IOKEEP bit in the DSFR register release the retained states of pins after waking up the MCU from deep standby mode.

Clearing the IOKEEP bit to 0 releases the all pin states, however, the pin states are released just after waking up from deep standby mode, when the EBUSKEEPE bit is 0. Set the EBUSKEEPE bit as 0 when activating the MCU from an external memory. Table 6 lists the external memory control pins which enable the EBUSKEEPE bit setting. Figure 4 shows the EBUSKEEPE bit and retaining pin state timing.

Table 6 External Memory Control Pins to Enable EBUSKEEPE Bit Setting

Boot mode 0 (CS0 space)	Boot mode 2 (NAND flash memory)	Boot modes 1 and 3 (Serial flash memory)
A [20:1] D [15:0] CS0#, RD#, CKIO	NAF [7:0] FRE#, FCLE, FALE, FWE#, FCE#, FRB	RSPCK0, SSL00, MOSI0, MISO0

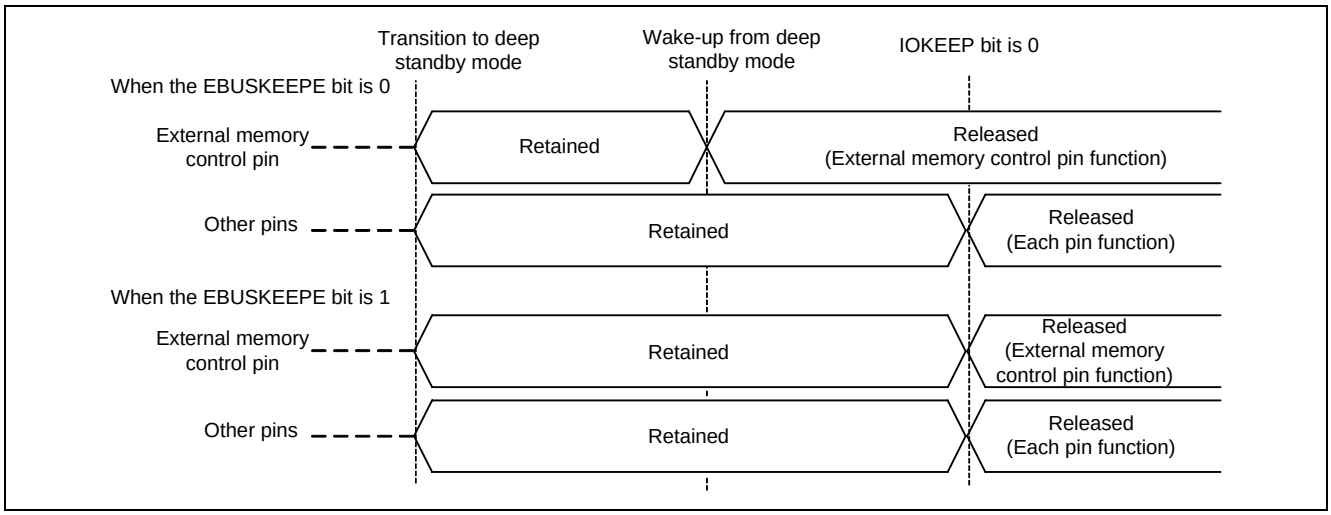


Figure 4 EBUSKEEPE Bit and Retaining Pin State Timing

Figure 5 shows the operating image to activate the MCU from the data-retention internal RAM. Retrieve the program counter and stack pointer values from the vector area, and jump to the boot program prepared for activating the MCU from the data-retention internal RAM (RAM boot program).Set values in the vector area and the RAM boot program before the MCU transitions to deep standby mode.

Note that the vector area address and the capacity of the data-retention RAM depend on the MCU type, with 1-MB internal RAM or 640-KB internal RAM.

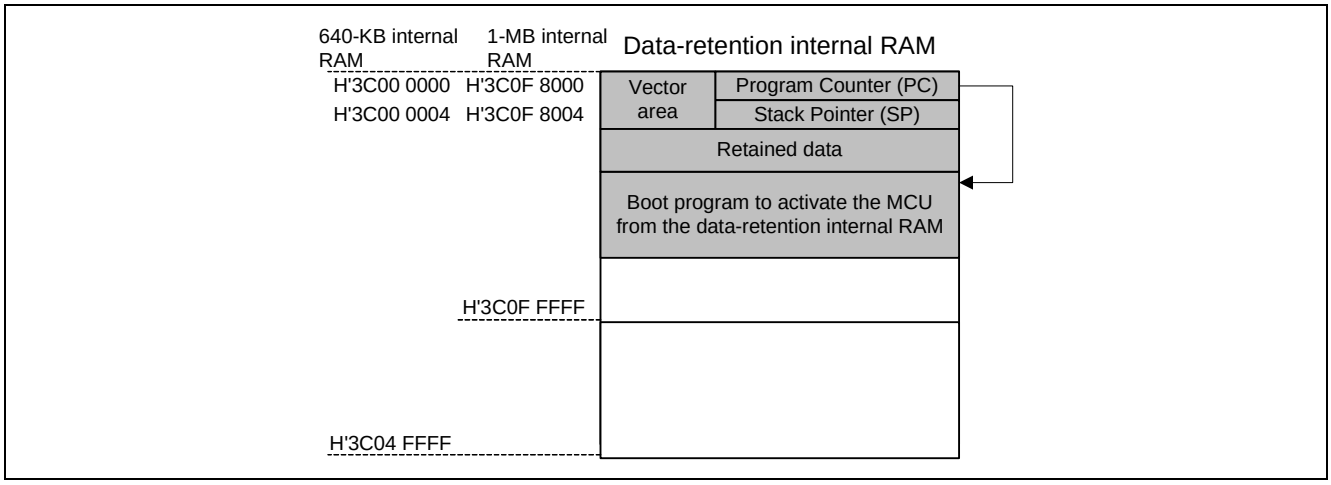


Figure 5 Operating Image to Activate the MCU from the Data-retention Internal RAM

2.1.7 Application Timing Chart

Figure 6 shows the timing chart of this application. This application uses the NMI interrupt to set the state transition and transition to deep standby mode, and then wake up the MCU from deep standby mode at the rising edge of PJ3 pin.

As this application activate the data-retention internal RAM, set the RAMBOOT bit to 1. Set CKOEN [1:0] bits to B'11 and specify the CKIO pin not to output signals during deep standby mode. CKIO bit outputs signal after waking up from deep standby mode and clearing the IOKEEP bit to 0.

Power-on reset exception handling initializes registers, however, the power-on reset exception handling when the MCU is woke up from deep standby mode retains the DSFR register value, except the wake-up factor is the reset. When the wake-up factor is reset, DSFR register is also initialized. Refer to the DSFR register to know the wake up factor from deep standby mode.

For initializing registers, refer to 36.3 Register States in Each Operating Mode in the SH7262 Group, SH7264 Group Hardware Manual.

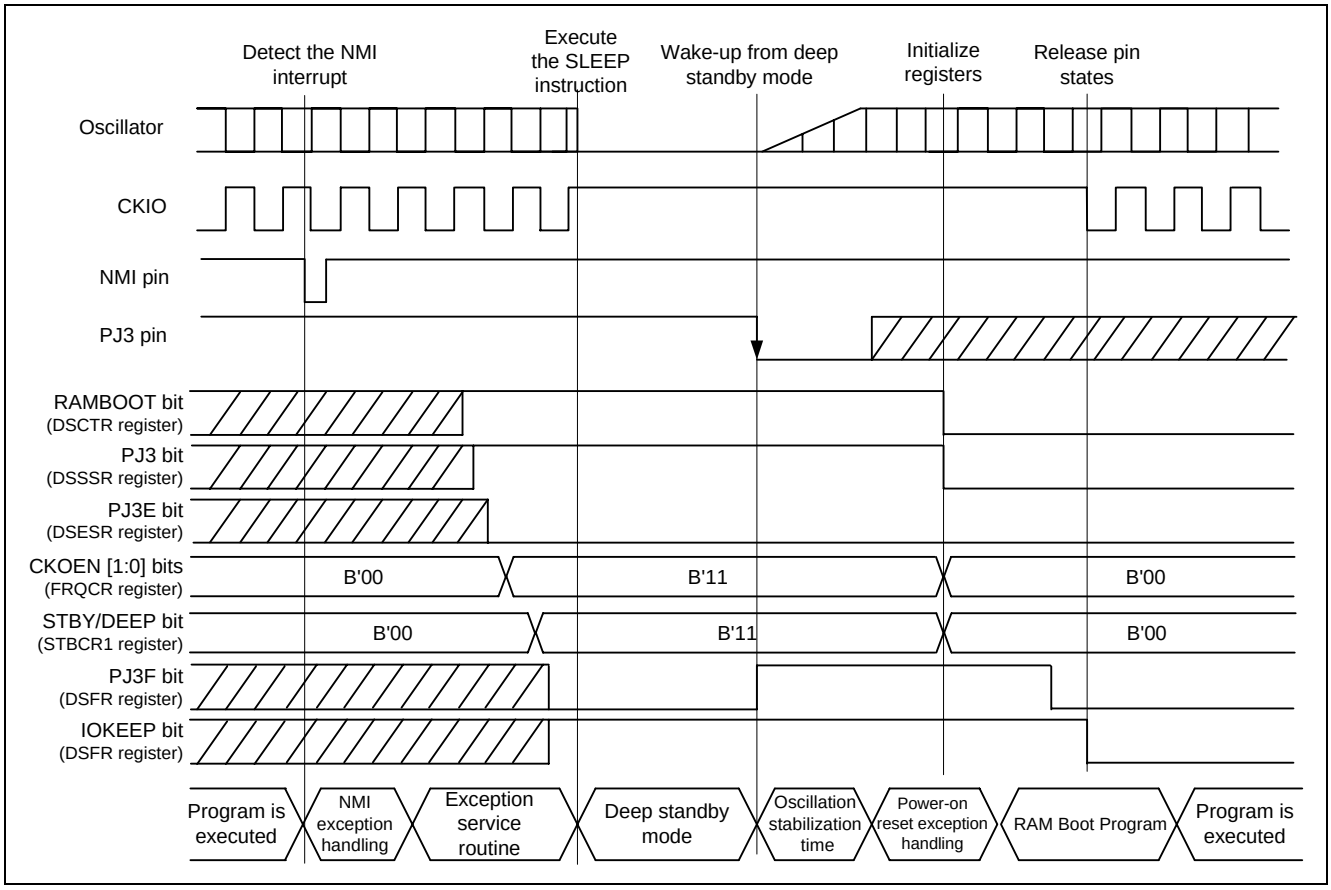


Figure 6 Application Timing Chart

2.2 Sample Program Operation

This section describes the sample program operation.

Main function outputs the RTC clock time to the terminal, uses the NMI interrupt routine to set state transition to deep standby mode. After waking up from deep standby mode, the RAM boot program is executed and it jumps to the main function.

The sample program does not assume to turn OFF the external device in deep standby mode. SDRAM self-refresh function is not set in the sample program.

2.2.1 Main Function Flow Chart

Figure 7 shows the main function flow chart in the sample program. It polls the RTC time, and outputs the time data to SCIF when the time is updated.

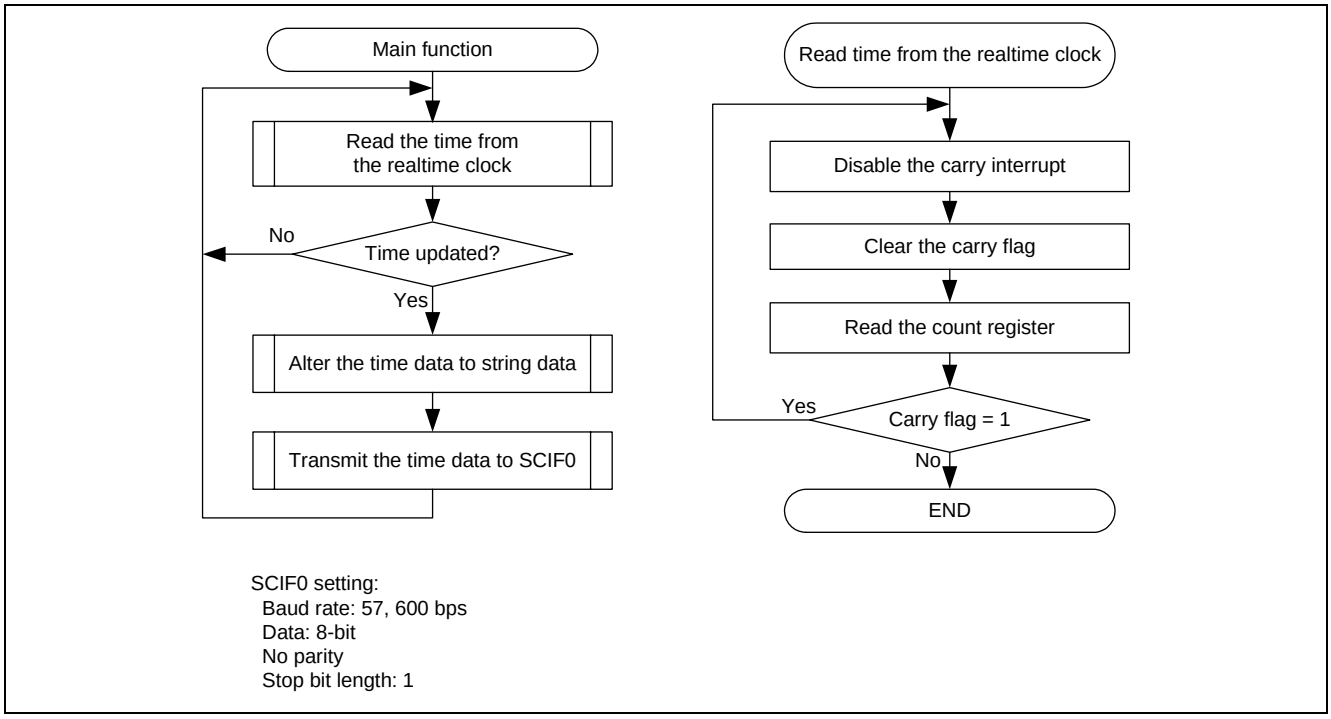


Figure 7 Main Function Flow Chart

2.2.2 NMI Interrupt Flow Chart

Figure 8 and Figure 9 show flow charts of the NMI interrupt in the sample program. It stores the standby start time in the data-retention internal RAM, handles pins not to flow unnecessary electric current, and transitions to deep standby mode.

The NMIE bit in the Interrupt controller register 0 (ICR0) disables the NMI interrupt on the SH7264 with 640-KB internal RAM. To solve the problem, clear the NMIE bit to 0 after the power-on reset or waking up from deep standby mode.

Following points must be observed when the MCU transitions to deep standby mode, not to retain pin states while accessing the external bus unintentionally.

- Align sections such as aligning the program and variable area on internal RAM, to avoid accessing the external bus by CPU
- Disable bus masters other than CPU such as the DMAC and VDC3 before transition to deep standby mode, to avoid accessing the external bus

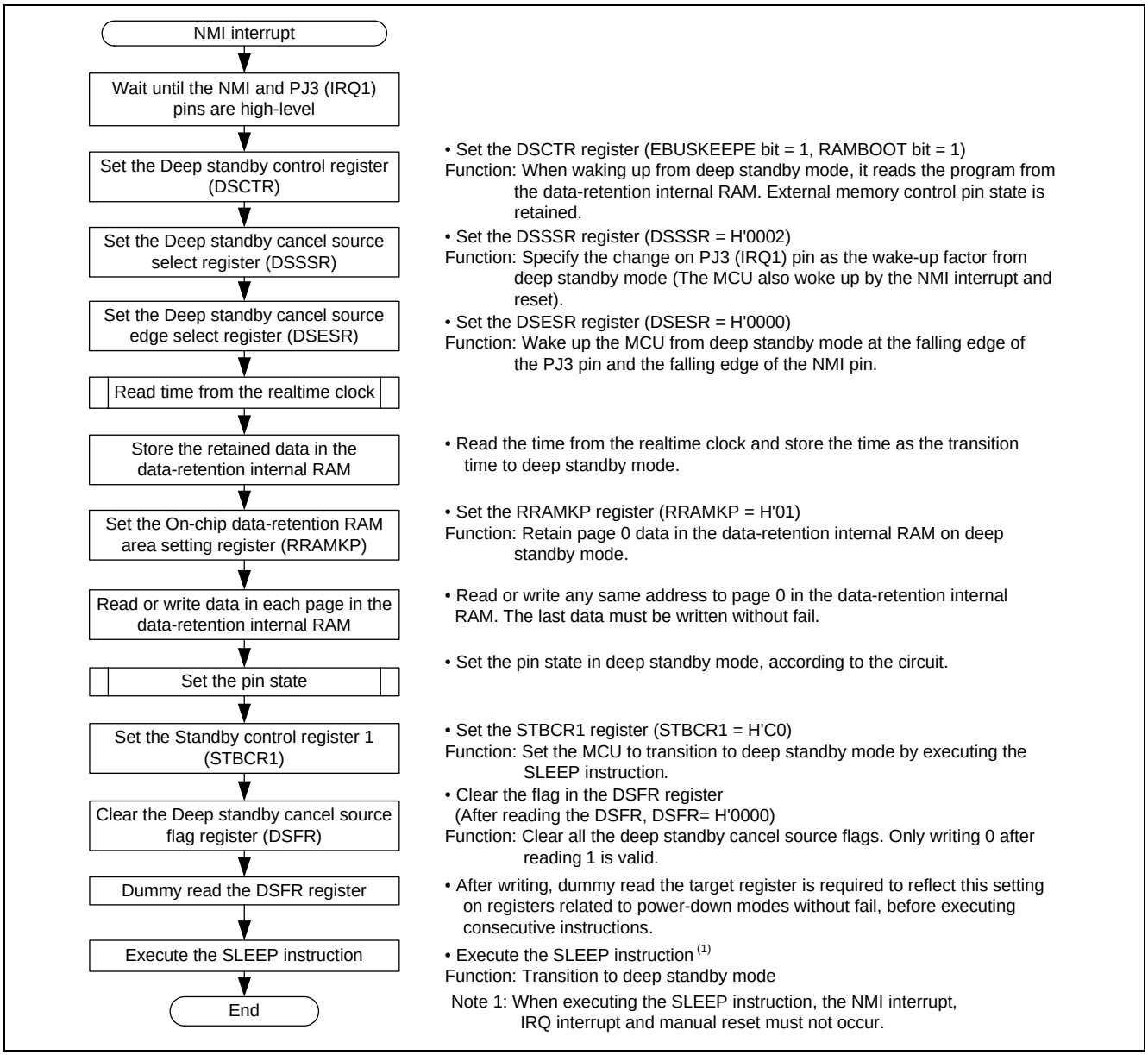


Figure 8 NMI Interrupt Flow Chart (1/2)

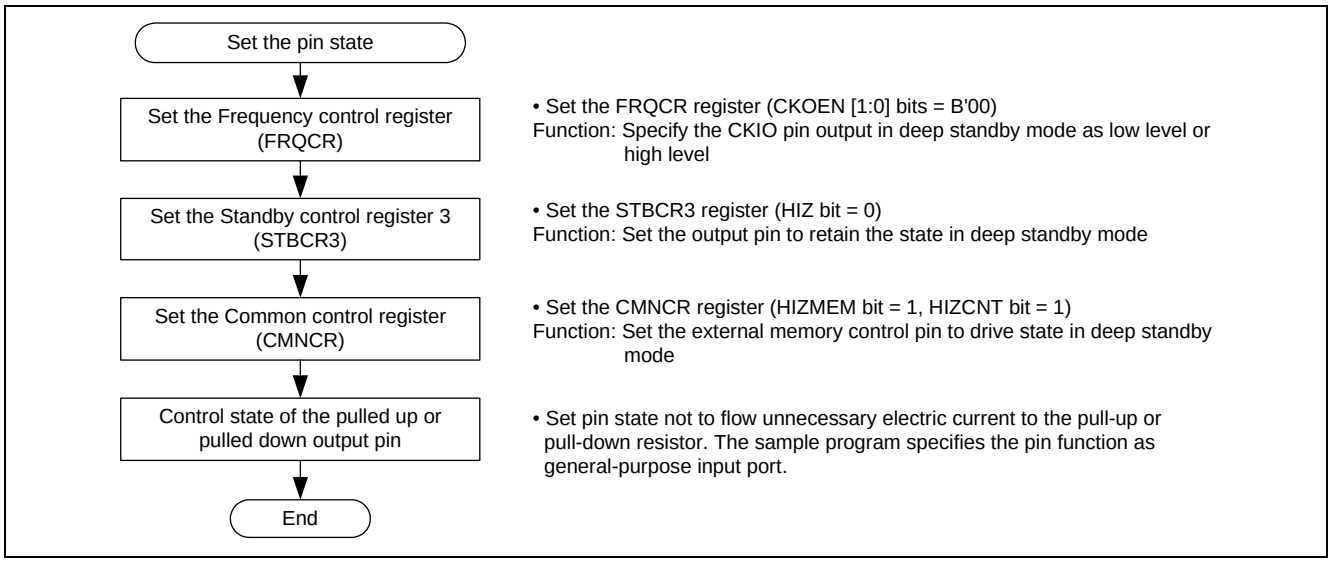
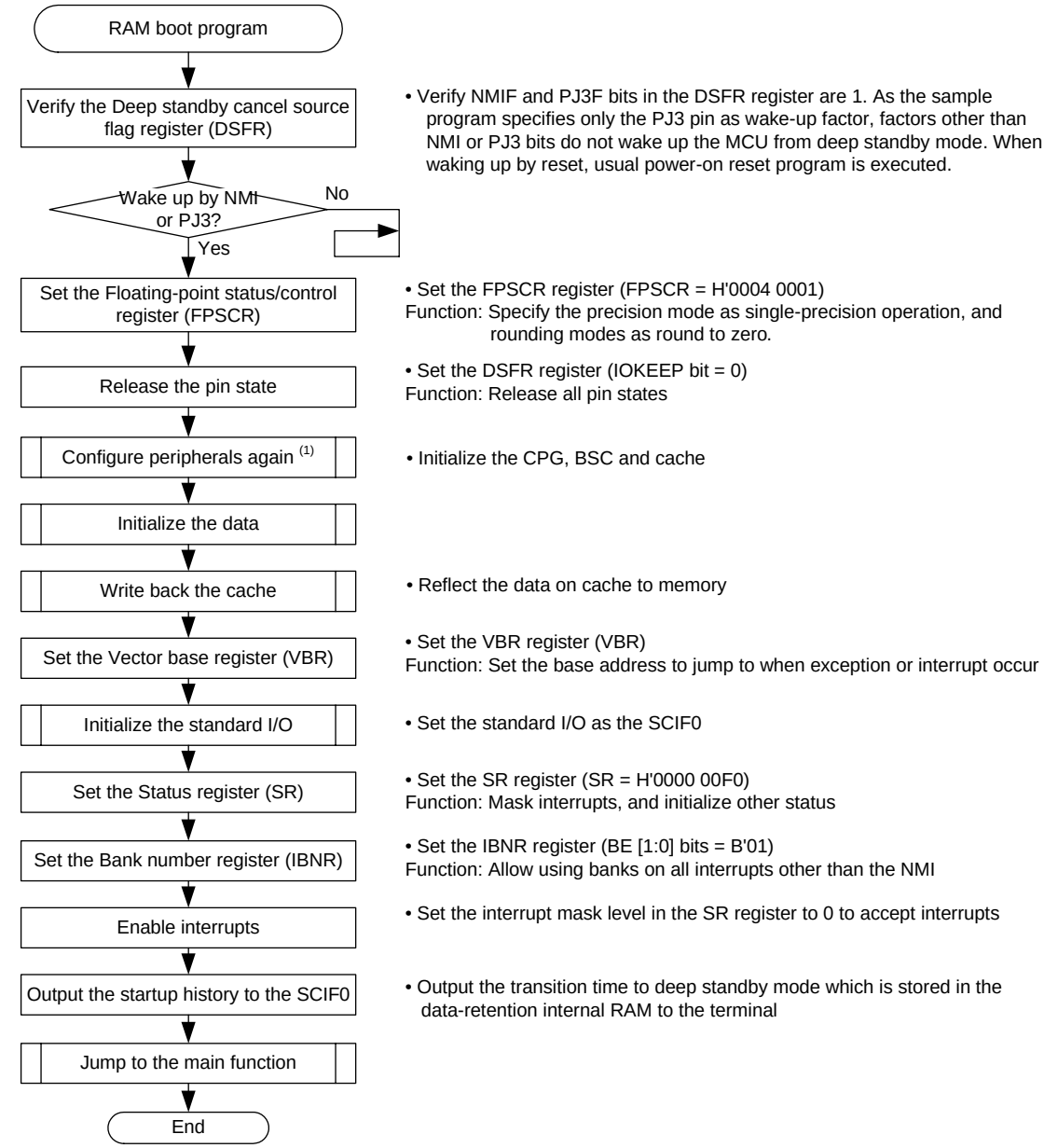


Figure 9 NMI Interrupt Flow Chart (2/2)

2.2.3 RAM Boot Program Flow Chart

Figure 10 shows the flow chart of the RAM boot program in the sample program. It releases the pin state, and initializes the peripherals and data. Then, it outputs the startup history information to the SCIF and jumps to the main function.



Note 1: As the sample program allocates the program to configure peripherals on NOR flash memory, it releases the pin state before accessing the external device. Allocate the program to configure peripherals on the data-retention internal RAM to configure peripherals again before releasing pin states.

Figure 10 RAM Boot Program Flow Chart

2.2.4 Realtime Clock Initialization Flow Chart

Figure 11 shows the flow chart of initializing the realtime clock in the sample program. The realtime clock is initialized by the power-on reset program.

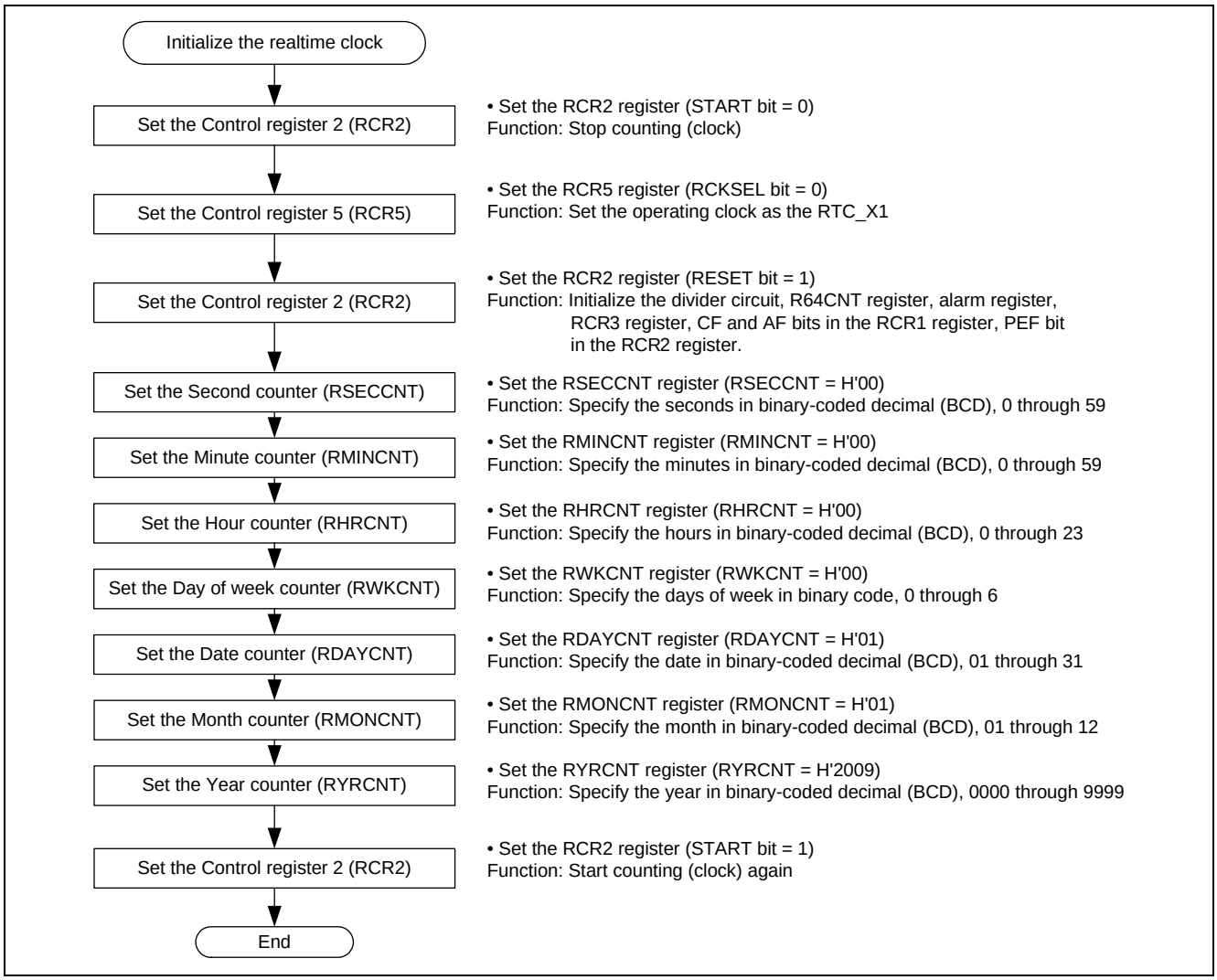


Figure 11 Realtime Clock Initialization Flow Chart

2.3 (Supplement) When Turning OFF the External Device

This application turns ON the external device in deep standby mode, however, the external device can be turned OFF to reduce power consumption additionally. This section describes an overview of turning OFF the external device.

Figure 12 shows the pin connection example when turning OFF the external device in deep standby mode. This example assumes to connect NOR flash memory and SDRAM to the MCU, and boot the MCU from boot mode 0.

When turning OFF the external device, set the output pin level to Hi-Z in deep standby mode, as the external device pin cannot be applied voltage. Connect the pull-up resistor not to the SH7264 power supply, but to external device power supply (EXVcc). Generate the reset signal (EXRES#) for external device which is at low level in deep standby mode.

Control the power supply to satisfy the timing chart example shown in Figure 13. As the example uses the NOR flash memory to boot, turn ON the EXVcc just after the power-on reset to assert the EXRES# signal for a certain period of time. As the EXVcc is turned OFF in deep standby mode, set the EXRES# signal as low level. After waking up from deep standby mode, turn ON the EXVcc before releasing the pin state. Note that the pin state must be released before initializing external devices such as SDRAM, which require initialization before use.

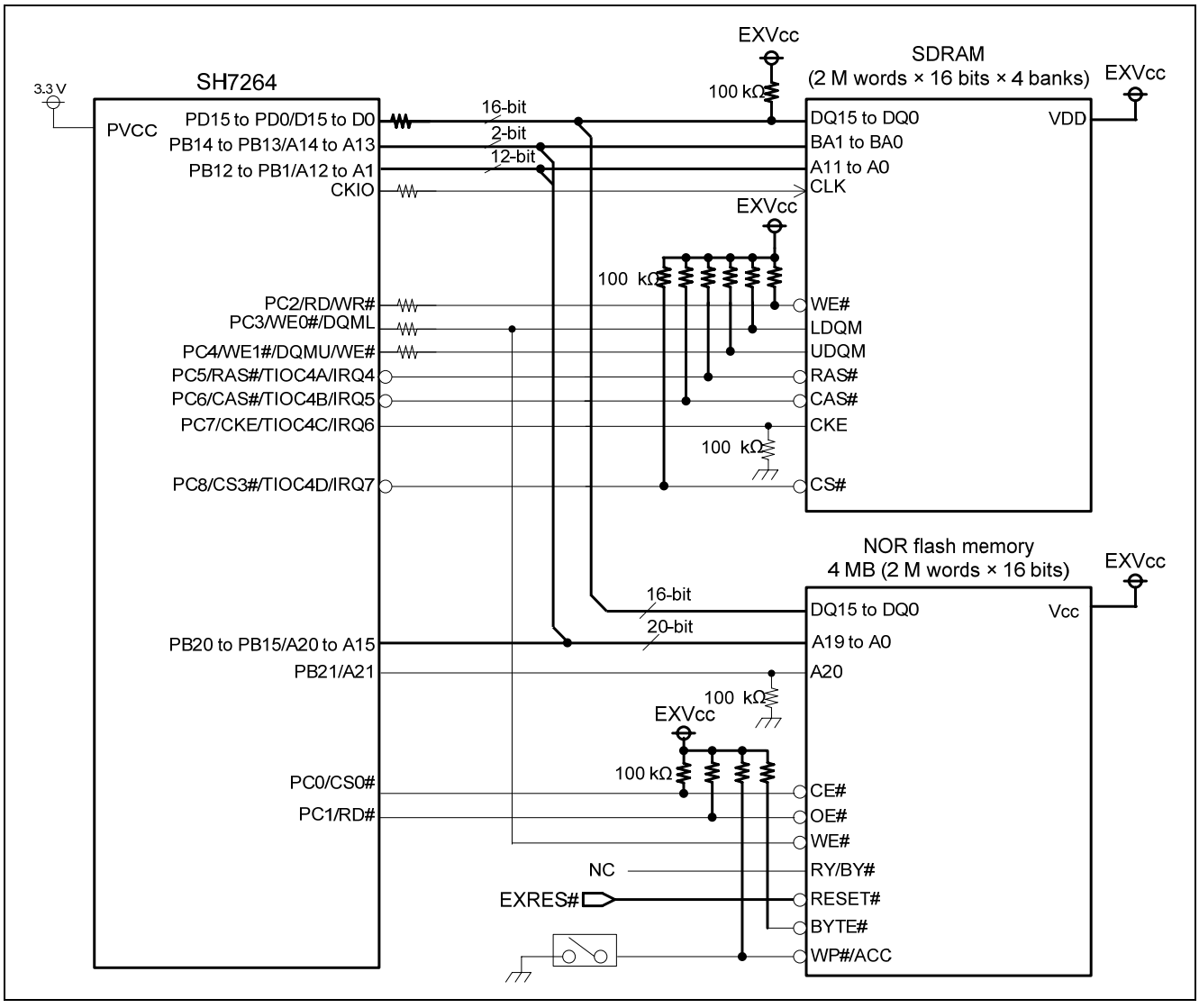
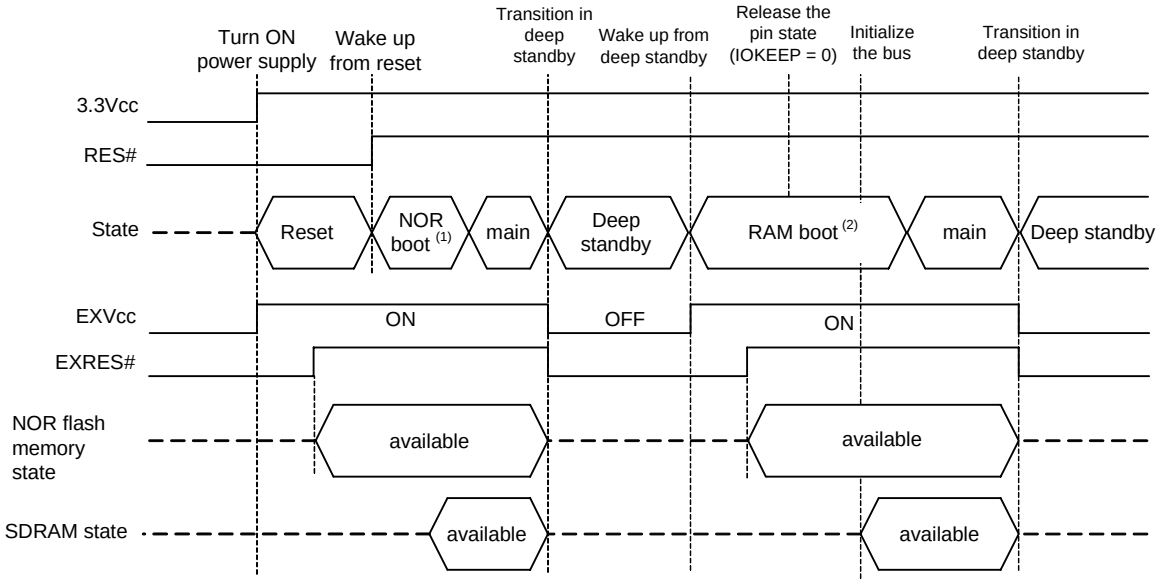


Figure 12 Pin Connection Example (When Turning OFF the External Device)



Notes:
1. This is the power-on reset exception handling according to the boot mode on reset. This example assumes to activate the MCU from NOR flash memory.
2. This is to activate the MCU from the data-retention internal RAM.

Figure 13 Timing Chart Example to Control Power Supply

3. Sample Program Listing

3.1 Sample Program Listing "main.c" (1/9)

```
/***********************************************************************
*   DISCLAIMER
*
*   This software is supplied by Renesas Electronics Corporation and is only
*   intended for use with Renesas products. No other uses are authorized.
*
*   This software is owned by Renesas Electronics Corporation and is protected under
*   all applicable laws, including copyright laws.
*
*   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
*   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
*   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
*   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
*   DISCLAIMED.
*
*   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
*   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
*   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
*   FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
*   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
*
*   Renesas reserves the right, without notice, to make changes to this
*   software and to discontinue the availability of this software.
*   By using this software, you agree to the additional terms and
*   conditions found by accessing the following link:
*   http://www.renesas.com/disclaimer
***********************************************************************
*   Copyright (C) 2009. Renesas Electronics Corporation. All Rights Reserved.
*""FILE COMMENT""************ Technical reference data *************************
*   System Name : SH7264 Sample Program
*   File Name   : main.c
*   Abstract    : Deep Standby Mode
*   Version     : 1.00.00
*   Device      : SH7262/SH7264
*   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
*               : C/C++ compiler package for the SuperH RISC engine family
*               :                               (Ver.9.02 Release00).
*   OS          : None
*   H/W Platform: M3A-HS64G50 (CPU board)
*   Description :
***********************************************************************
*   History     : Jul.28,2009 Ver.1.00.00
*""FILE COMMENT END""*********************************************************/
#include <machine.h>
#include <_h_c_lib.h>
#include <stdio.h>
#include "iodefine.h"
#include "rtc.h"
```

3.2 Sample Program Listing "main.c" (2/9)

```
49
50    /* ==== Macro definition ==== */
51    #define RETENTION_RAM_PAGE_0  0x3C0F8000 /* Page 0 on the MCU with 1-MB
52                                          internal RAM */
53    #define RETENTION_RAM_PAGE_1  0x3C0FC000 /* Page 1 on the MCU with 1-MB
54                                          internal RAM */
55    //#define RETENTION_RAM_PAGE_0 0x3C000000 /* Page 0 on the MCU with 640-KB
56    //                                      internal RAM */
57    //#define RETENTION_RAM_PAGE_1 0x3C004000 /* Page 1 on the MCU with 640-KB
58    //                                      internal RAM */
59    //#define RETENTION_RAM_PAGE_2 0x3C008000 /* Page 2 on the MCU with 640-KB
60    //                                      internal RAM */
61    //#define RETENTION_RAM_PAGE_3 0x3C028000 /* Page 3 on the MCU with 640-KB
62    //                                      internal RAM */
63    #define INT_OFFSET 0x10           /* Offset from the top of the exception handling
64                                vector table to INT_Vectors */
65    #define FPSCR_Init 0x00040001 /* FPSCR register default setting */
66    #define SR_Init    0x000000F0 /* SR register default setting */
67
68    /* ==== Function prototype declaration ==== */
69    void main(void);
70    void io_int_nmi( void );
71    void DeepStandby_RamBoot( void );
72    /* ---- External reference ---- */
73    extern void HardwareSetup(void);
74    extern void io_cache_writeback(void);
75    extern void _INIT_IOLIB(void);
76    /* ---- Local function ---- */
77    static void io_set_pin_status( void );
78    static void print_log( void );
79    static void time_to_str( char *str, const RTC_TIME *tm );
80
81    /* ==== Variable definition ==== */
82    extern unsigned int INT_Vectors;  /* Vector table (Vector number 5 or later) */
83
84    /* ---- Data in the data-retention internal RAM ---- */
85    #pragma section RETENTION_AREA    /* MCU with 1-MB internal RAM: 0x3C0F8000,
86                                  MCU with 640-KB: 0x3C000000 */
87    unsigned long vecttbl_dpstby[2] = {   /* Vector area to use when waking up from
88                                  deep standby mode */
89      (unsigned long)DeepStandby_RamBoot,
90      (unsigned long)(__secend("S")),
91    };
92    static RTC_TIME stbyin_time;      /* Time data to transition to deep standby mode */
93    #pragma section
94
```

3.3 Sample Program Listing "main.c" (3/9)

```
95    /*""FUNC COMMENT""*****************************************************************
96     * ID           :
97     * Outline      : Main function
98     *-----------------------------------------------------------------------------
99     * Include      : stdio.h, rtc.h
100    *-----------------------------------------------------------------------------
101    * Declaration  : void main(void);
102    *-----------------------------------------------------------------------------
103    * Description  : Outputs the realtime clock time data to the terminal.
104    *-----------------------------------------------------------------------------
105    * Argument     : void
106    *-----------------------------------------------------------------------------
107    * Return Value : void
108    *-----------------------------------------------------------------------------
109    * Note         : None
110    *""FUNC COMMENT END""*************************************************************/
111   void main(void)
112   {
113     static RTC_TIME tm;
114     unsigned char old_sec = 0xFF;
115     char str[9];
116
117     while(1){
118       /* ---- Reads the time from the realtime clock ---- */
119       io_read_rtc( &tm );
120       /* ---- Time is updated ---- */
121       if( tm.sec != old_sec ){
122           old_sec= tm.sec;
123           /* ---- Alters the time data to string data ---- */
124           time_to_str( str, &tm );
125           /* ---- Outputs the time to the terminal ---- */
126           printf( "RTC time is[%s]\n", str );
127           fflush( stdout );
128       }
129     }
130   }
131
```

3.4 Sample Program Listing "main.c" (4/9)

```
132    /*""FUNC COMMENT""*****************************************************************
133     * ID           :
134     * Outline      : NMI interrupt
135     *-------------------------------------------------------------------------------
136     * Include      : iodefine.h
137     *-------------------------------------------------------------------------------
138     * Declaration  : void io_int_nmi( void );
139     *-------------------------------------------------------------------------------
140     * Description  : Stores the standby start time in the data-retention internal RAM,
141     *              : handles pins not to flow unnecessary electric current, and
142     *              : transitions to deep standby mode.
143     *-------------------------------------------------------------------------------
144     * Argument     : void
145     *-------------------------------------------------------------------------------
146     * Return Value : void
147     *-------------------------------------------------------------------------------
148     * Note         : None
149     *""FUNC COMMENT END""*************************************************************/
150    void io_int_nmi( void )
151    {
152      static RTC_TIME tm;
153      volatile int dummy;
154      int *p;
155
156      /* ---- Waits until pins NMI and PJ3 (IRQ1) are high-level ---- */
157      while( INTC.ICR0.BIT.NMIL == 0){
158        /* Wait */
159      }
160      while( PORT.PJPR0.BIT.PJ3PR == 0 ){
161        /* Wait */
162      }
163      /* ---- Sets the Deep standby control register (DSCTR) ---- */
164      CPG.DSCTR.BYTE = 0xC0u;         /* EBUSKEEPE bit =1: Retains the external
165                                                  memory control pin state */
166                                  /* RAMBOOT bit = 1: Activates the MCU from the
167                                                  data-retention internal RAM */
168      /* ---- Set the Deep standby cancel source select register (DSSSR) ---- */
169      CPG.DSSSR.WORD = 0x0002u;       /* Specifies the PJ3 (IRQ1) as the wake-up factor */
170
171      /* ---- Sets the Deep standby cancel source edge select register (DSESR) ---- */
172      CPG.DSESR.WORD = 0x0000u;       /* Specifies the falling edges of pins PJ3
173                                    (IRQ1) and NMI */
174
175      /* ---- Reads the time from the realtime clock and store the time in the
176           data-retention internal RAM ---- */
177      io_read_rtc( &tm );
178      stbyin_time = tm;
179
```

3.5 Sample Program Listing "main.c" (5/9)

```
180     /* ---- Sets the On-chip data retention RAM area setting register (RRAMKP) ---- */
181     CPG.RRAMKP.BYTE = 0x01u;    /* Retaines page 0 only */
182
183     /* ---- Dummy read/write to guarantee the writing in the data-retention internal RAM ----
184   */
185     p = (int *)RETENTION_RAM_PAGE_0;
186     dummy = *p;
187     *p = dummy;
188
189     /* ---- Sets the pin state ---- */
190     io_set_pin_status();
191
192     /* ---- Sets the Standby controller register 1 (STBCR1) ---- */
193     CPG.STBCR1.BYTE = 0xC0u;    /* Transition to deep standby mode by the SLEEP instruction
194   */
195
196     /* ---- Clears the Deep standby cancel source flag register (DSFR) ---- */
197     CPG.DSFR.WORD &= 0x0000u;       /* Clears all the deep standby cancel source flags
198                                 (Reads 1 and clears flags) */
199     dummy = CPG.DSFR.WORD;          /* Dummy read to enable this setting before
200                                 executing consecutive instructions */
201
202     /* ---- Transitions to deep standby mode ---- */
203     sleep();
204
205     nop();
206     nop();
207     nop();
208     nop();
209   }
210   /*""FUNC COMMENT""*****************************************************************
211    * ID           :
212    * Outline      : RAM boot program
213    *-------------------------------------------------------------------------------
214    * Include      : stdio.h, machine.h, _h_c_lib.h
215    *-------------------------------------------------------------------------------
216    * Declaration  : void DeepStandby_RamBoot( void );
217    *-------------------------------------------------------------------------------
218    * Description  : This module is executed when waking up the MCU from deep
219    *              : standby mode. It releases the pin state, initializes the
220    *              : hardware, and then outputs the startup history information to
221    *              : the SCIF (standard output). Allocate this module in the target
222    *              : page to retain on the data-retention internal RAM.
223    *-------------------------------------------------------------------------------
224    * Argument     : void
225    *-------------------------------------------------------------------------------
226    * Return Value : void
227    *-------------------------------------------------------------------------------
       * Note         : None
       *""FUNC COMMENT END""*************************************************************/
```

3.6 Sample Program Listing "main.c" (6/9)

```
228   #pragma section RETENTION_AREA
229   void DeepStandby_RamBoot( void )
230   {
231     unsigned short reg;
232
233     /* ---- Verifies the Deep standby cancel source flag register (DSFR) ---- */
234     reg = CPG.DSFR.WORD;
235     if( (reg & 0x0102u) == 0x0000u ){   /* Wake-up other than the PJ3 or
236                                         NMI interrupt is error */
237       while(1){
238           /* error */
239       }
240     }
241     /* ---- Sets the Floating-point status/control register (FPSCR) ---- */
242     set_fpscr( FPSCR_Init );    /* Single-precision, round to zero */
243
244     /* ---- Releases the pin state ---- */
245     CPG.DSFR.BIT.IOKEEP = 0;    /* 1 must be read before writing 0 */
246
247     /* ---- Initializes the hardware ---- */
248     HardwareSetup();            /*
249
250     /* ---- Initializes the data ---- */
251     _INITSCT();
252     io_cache_writeback();
253
254     /* ---- Sets the Vector base register (VBR) ====*/
255     set_vbr((void *)((char *)&INT_Vectors - INT_OFFSET));
256
257     /* ---- Initializes the standard I/O ---- */
258     _INIT_IOLIB();
259
260     /* ---- Sets the Status register (SR) ---- */
261     set_cr(SR_Init);
262     nop();
263
264     /* ----  Sets the Bank number register (IBNR) ---- */
265     INTC.IBNR.BIT.BE = 0x01;    /* Allow using banks on all interrupts other
266                                  than the NMI */
267
268     /* ---- Enables interrupts ---- */
269     set_imask(0);
270
271     /* ---- Outputs the stardup history to the terminal ---- */
272     print_log();
273
```

3.7 Sample Program Listing "main.c" (7/9)

```
274     /* ---- Jumps to the main function ---- */
275     main();
276     sleep();
277   }
278   #pragma section
279   /*""FUNC COMMENT""*****************************************************************
280    * ID           :
281    * Outline      : Set the pin state
282    *-------------------------------------------------------------------------------
283    * Include      : iodefine.h
284    *-------------------------------------------------------------------------------
285    * Declaration  : static void io_set_pin_status( void );
286    *-------------------------------------------------------------------------------
287    * Description  : Sets the pin state in deep standby mode.
288    *-------------------------------------------------------------------------------
289    * Argument     : void
290    *-------------------------------------------------------------------------------
291    * Return Value : void
292    *-------------------------------------------------------------------------------
293    * Note         : None
294    *""FUNC COMMENT END""*************************************************************/
295   static void io_set_pin_status( void )
296   {
297     /* ---- Sets the Frequency control register (FRQCR) ---- */
298     CPG.FRQCR.BIT.CKOEN = 0;    /* Specifies the CKIO pin output in deep
299                                  * standby mode as low level or high level
300                                  */
301     /* ---- Sets the Standby control register 3 (STBCR3) ---- */
302     CPG.STBCR3.BIT.HIZ = 0;         /* Sets the output pin in deep standby mode
303                                  * not as Hi-Z, but to retain the state
304                                  */
305     /* ---- Sets the Common control register (CMNCR)  ---- */
306     BSC.CMNCR.BIT.HIZMEM = 1;       /* Sets the external memory control pin to
307                                  drive state */
308     BSC.CMNCR.BIT.HIZCNT = 1;       /* Sets the external memory control pin for
309                                   SDRAM to drive state */
310
311     /* ---- Handles pins not to flow unnecessary electric current to
312      pull-up/pull-down resistor ---- */
313     /* A21 */
314     PORT.PBIOR1.BIT.PB21IOR = 0;/* Sets the direction of PB21 pin to input */
315     PORT.PBCR5.BIT.PB21MD = 0;      /* Sets the PB21/A21 pin function as
316                                  general-purpose I/O port */
317   }
318
```

3.8 Sample Program Listing "main.c" (8/9)

```
319  /*""FUNC COMMENT""*****************************************************************
320   * ID           :
321   * Outline      : Output the startup history
322   *-------------------------------------------------------------------------------
323   * Include      : stdio.h, rtc.h
324   *-------------------------------------------------------------------------------
325   * Declaration  : static void print_log( void );
326   *-------------------------------------------------------------------------------
327   * Description  : Time data to transition to deep standby mode and wake up from
328   *              : deep standby mode to the terminal.
329   *-------------------------------------------------------------------------------
330   * Argument     : void
331   *-------------------------------------------------------------------------------
332   * Return Value : void
333   *-------------------------------------------------------------------------------
334   * Note         : None
335   *""FUNC COMMENT END""*************************************************************/
336  static void print_log( void )
337  {
338    RTC_TIME tm;
339    char str1[9], str2[9];
340
341    /* ---- Time to transition to deep standby mode ---- */
342    time_to_str( str1, &stbyin_time );
343
344    /* ---- Time to wake up from deep standby mode ---- */
345    io_read_rtc( &tm );
346    time_to_str( str2, &tm );
347
348    /* ---- Outputs the time to the terminal ---- */
349    printf( "Deep standby time\n");
350    printf( "    => From %s\n", str1 );
351    printf( "    => To   %s\n", str2 );
352    fflush( stdout );
353  }
354
```

3.9 Sample Program Listing "main.c" (9/9)

```
355   /*""FUNC COMMENT""*****************************************************************
356    * ID           :
357    * Outline      : Convert the time data to string data
358    *-------------------------------------------------------------------------------
359    * Include      : rtc.h
360    *-------------------------------------------------------------------------------
361    * Declaration  : static void time_to_str( char *str, const RTC_TIME tm );
362    *-------------------------------------------------------------------------------
363    * Description  : Converts the time data specified by the argument to string data
364    *              : ("hh:mm:ss"). The str must have more than 9 bytes of empty area.
365    *-------------------------------------------------------------------------------
366    * Argument     : char *str         ; O : Buffer address to store strings
367    *              : const RTC_TIME tm ; I : Time data
368    *-------------------------------------------------------------------------------
369    * Return Value : void
370    *-------------------------------------------------------------------------------
371    * Note         : None
372    *""FUNC COMMENT END""*************************************************************/
373   static void time_to_str( char *str, const RTC_TIME *tm )
374   {
375     *str++ = (( tm->hour >> 4 ) & 0xF) + '0';
376     *str++ = (( tm->hour      ) & 0xF) + '0';
377     *str++ = ':';
378     *str++ = (( tm->min  >> 4 ) & 0xF) + '0';
379     *str++ = (( tm->min       ) & 0xF) + '0';
380     *str++ = ':';
381     *str++ = (( tm->sec  >> 4 ) & 0xF) + '0';
382     *str++ = (( tm->sec       ) & 0xF) + '0';
383     *str   = '\0';
384   }
385   /* End of File */
```

3.10 Sample Program Listing "rtc.c" (1/3)

```
1   /***********************************************************************
2   *   DISCLAIMER
3   *
4   *   This software is supplied by Renesas Electronics Corporation and is only
5   *   intended for use with Renesas products. No other uses are authorized.
6   *
7   *   This software is owned by Renesas Electronics Corporation and is protected under
8   *   all applicable laws, including copyright laws.
9   *
10  *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11  *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12  *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13  *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14  *   DISCLAIMED.
15  *
16  *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17  *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18  *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19  *   FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20  *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21  *
22  *   Renesas reserves the right, without notice, to make changes to this
23  *   software and to discontinue the availability of this software.
24  *   By using this software, you agree to the additional terms and
25  *   conditions found by accessing the following link:
26  *   http://www.renesas.com/disclaimer
27  ***********************************************************************
28  *   Copyright (C) 2009. Renesas Electronics Corporation. All Rights Reserved.
29  *""FILE COMMENT""************ Technical reference data *************************
30  *   System Name : SH7264 Sample Program
31  *   File Name   : rtc.c
32  *   Abstract    : Realtime clock control
33  *   Version     : 1.00.00
34  *   Device      : SH7262/SH7264
35  *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
36  *               : C/C++ compiler package for the SuperH RISC engine family
37  *               :                               (Ver.9.02 Release00).
38  *   OS          : None
39  *   H/W Platform: M3A-HS64G50 (CPU board)
40  *   Description :
41  ***********************************************************************
42  *   History     : Jul.27,2009 Ver.1.00.00
43  *""FILE COMMENT END""*****************************************************/
44  #include "iodefine.h"
45  #include "rtc.h"
46
```

3.11 Sample Program Listing "rtc.c" (2/3)

```
47    /* ==== Macro definition ==== */
48    #define RYRCNT_INI     0x2009  /* 2009 */
49    #define RMONCNT_INI        0x01    /* January */
50    #define RDAYCNT_INI        0x01    /* 1st */
51    #define RWKCNT_INI     0x1     /* Monday */
52    #define RHRCNT_INI     0x00    /* hh: 0 */
53    #define RMINCNT_INI        0x00    /* mm: 0 */
54    #define RSECCNT_INI        0x00    /* ss: 0 */
55
56    /* ==== Variable definition ==== */
57    RTC_TIME tm_init = {
58      RYRCNT_INI, RMONCNT_INI, RDAYCNT_INI, RWKCNT_INI, RHRCNT_INI, RMINCNT_INI, RSECCNT_INI,
59    };
60
61    /*""FUNC COMMENT""*****************************************************************
62     * ID           :
63     * Outline      : Initialize the realtime clock
64     *-----------------------------------------------------------------------------
65     * Include      : iodefine.h, rtc.h
66     *-----------------------------------------------------------------------------
67     * Declaration  : void io_init_rtc( void );
68     *-----------------------------------------------------------------------------
69     * Description  : Initializes the realtime clock.
70     *-----------------------------------------------------------------------------
71     * Argument     : void
72     *-----------------------------------------------------------------------------
73     * Return Value : void
74     *-----------------------------------------------------------------------------
75     * Note         : None
76     *""FUNC COMMENT END""*************************************************************/
77    void io_init_rtc( void )
78    {
79      /* ---- Stop counting ---- */
80      RTC.RCR2.BIT.START = 0;
81
82      /* ---- Sets the Control register 5 (RCR5) ---- */
83      RTC.RCR5.BIT.RCKSEL = 0;    /* Sets the operating clock as the RTC_X1 */
84
85      /* ---- Sets the Control register 2 (RCR2) ---- */
86      RTC.RCR2.BIT.RESET = 1;         /* Initializes the divider circuit, R64CNT
87                                    * register, alarm register, RCR3 register,
88                                    * CF and AF bits in the RCR1 register, and
89                                    * PEF bit in the RCR2 register. Stops counting */
```

3.12 Sample Program Listing "rtc.c" (3/3)

```
90      /* ---- Sets the counter ---- */
91      RTC.RSECCNT = tm_init.sec;
92      RTC.RMINCNT = tm_init.min;
93      RTC.RHRCNT  = tm_init.hour;
94      RTC.RWKCNT  = tm_init.week;
95      RTC.RDAYCNT = tm_init.day;
96      RTC.RMONCNT = tm_init.mon;
97      RTC.RYRCNT  = tm_init.year;
98
99      /* ---- Starts counting again ---- */
100     RTC.RCR2.BIT.START = 1;
101   }
102   /*""FUNC COMMENT""*****************************************************************
103    * ID           :
104    * Outline      : Read time from the realtime clock
105    *-------------------------------------------------------------------------------
106    * Include      : iodefine.h, rtc.h
107    *-------------------------------------------------------------------------------
108    * Declaration  : void io_read_rtc( RTC_TIME *time );
109    *-------------------------------------------------------------------------------
110    * Description  : Reads the count register value of the realtime clock and
111    *              : stores the value in the area specified by the argument.
112    *-------------------------------------------------------------------------------
113    * Argument     : RTC_TIME *time : O : Area to store the count register value
114    *-------------------------------------------------------------------------------
115    * Return Value : void
116    *-------------------------------------------------------------------------------
117    * Note         : None
118    *""FUNC COMMENT END""*************************************************************/
119   void io_read_rtc( RTC_TIME *time )
120   {
121     /* ---- Disables the carry interrupt ---- */
122     RTC.RCR1.BIT.CIE = 0;
123
124     do{
125       /* ---- Clears the carry flag ---- */
126       RTC.RCR1.BIT.CF = 0;
127
128       /* ---- Reads the count register ---- */
129       time->sec = RTC.RSECCNT;
130       time->min = RTC.RMINCNT;
131       time->hour= RTC.RHRCNT;
132       time->week= RTC.RWKCNT;
133       time->day = RTC.RDAYCNT;
134       time->mon = RTC.RMONCNT;
135       time->year= RTC.RYRCNT;
136
137     }while( RTC.RCR1.BIT.CF == 1 ); /* Reads the value again when the carry
138                                        flag is 1 */
139   }
140   /* End of File */
```

3.13 Sample Program Listing "rtc.h" (1/2)

```
1     /***********************************************************************
2     *   DISCLAIMER
3     *
4     *   This software is supplied by Renesas Electronics Corporation and is only
5     *   intended for use with Renesas products. No other uses are authorized.
6     *
7     *   This software is owned by Renesas Electronics Corporation and is protected under
8     *   all applicable laws, including copyright laws.
9     *
10    *   THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
11    *   REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
12    *   INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
13    *   PARTICULAR PURPOSE AND NON-INFRINGEMENT.  ALL SUCH WARRANTIES ARE EXPRESSLY
14    *   DISCLAIMED.
15    *
16    *   TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
17    *   ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
18    *   FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
19    *   FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
20    *   AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
21    *
22    *   Renesas reserves the right, without notice, to make changes to this
23    *   software and to discontinue the availability of this software.
24    *   By using this software, you agree to the additional terms and
25    *   conditions found by accessing the following link:
26    *   http://www.renesas.com/disclaimer
27    ***********************************************************************
28    *   Copyright (C) 2009. Renesas Electronics Corporation. All Rights Reserved.
29    *""FILE COMMENT""************ Technical reference data *************************
30    *   System Name : SH7264 Sample Program
31    *   File Name   : rtc.h
32    *   Abstract    : Realtime clock control
33    *   Version     : 1.00.00
34    *   Device      : SH7262/SH7264
35    *   Tool-Chain  : High-performance Embedded Workshop (Ver.4.04.01).
36    *               : C/C++ compiler package for the SuperH RISC engine family
37    *               :                               (Ver.9.02 Release00).
38    *   OS          : None
39    *   H/W Platform: M3A-HS64G50 (CPU board)
40    *   Description :
41    ***********************************************************************
42    *   History     : Jul.27,2009 Ver.1.00.00
43    *""FILE COMMENT END""*****************************************************/
44    #ifndef _RTC_H_
45    #define _RTC_H_
```

3.14 Sample Program Listing "rtc.h" (2/2)

```
46
47    /* ==== Macro definition ==== */
48    typedef struct tag_trc{
49      unsigned short year;
50      unsigned char  mon;
51      unsigned char  day;
52      unsigned char  week;
53      unsigned char  hour;
54      unsigned char  min;
55      unsigned char  sec;
56    }RTC_TIME;
57
58    /* ==== Function prototype declaration ==== */
59    void io_init_rtc( void );
60    void io_read_rtc( RTC_TIME *time );
61
62    #endif /* _RTC_H_ */
63    /* End of File */
```

4. References

- Software Manual
  SH-2A/SH2A-FPU Software Manual Rev.3.00
  The latest version of the software manual can be downloaded from the Renesas Electronics website.

- Hardware Manual
  SH7262 Group, SH7264 Group Hardware manual Rev.2.00
  The latest version of the hardware manual can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Oct.09.09	—	First edition issued
1.01	Feb.23.11	5	Fixed errors in the hardware manual

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
   - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
   - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
   - "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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

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

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

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

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

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

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

Colophon 1.0