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.

H8SX Family

Using the 32K Timer to Drive a 24-Hour Clock (Updated at 1-Minute Intervals)

Introduction

A 24-hour clock (which is updated at 1-minute intervals) is created by using the 32K timer.

Target Device

H8SX/1668R

Preface

Although the writing of this application note is in accord with the hardware manual for the H8SX/1668 Group, the program covered in this application note can be run on the target device indicated above. However, since some functional modules may be changed for the addition of functionality etc., be sure to perform a thorough evaluation by confirming the details with the hardware manual for the target device.

Contents

1. Specification 2
2. Applicable Conditions 3
3. Description of Modules Used 4
4. Principles of Operation 10
5. Description of Software 11
6. Documents and Programs for Reference 28

1. Specification

A 24-hour clock is created by using the 32K timer.

- The configuration of this clock is shown in figure 1.
- An LCD module is used to display the time.
- The clock starts operating after the current time has been set.
- The overflow period of the 32K timer is set to 60 s. The clock counts up at the end of each overflow period.
- The time data to be displayed on the LCD is stored in the on-chip RAM.
- If data in the on-chip RAM are erroneous, the LCD displays "error."
- The deep software standby mode is entered except when the clock counts up.
- If the IRQ0, IRQ1 or the NMI is generated while the clock is running, the clock continues to operate normally.

The operations of this clock are as follows.

(1) The LCD displays "00:00" after power has been applied.
(2) Set the "hours" by using the IRQ0 switch. Each time you press this switch, the "hours" data is incremented. "Hours" data from "00:00" to "23:00" is displayed.
(3) After setting the "hours" to the desired value, press the IRQ1 switch. This establishes the "hours" setting.
(4) Set the "minutes" by using the IRQ0 switch. Each time you press this switch, the "minutes" data is incremented. "Minutes" data from "00:00" to "00:59" is displayed.
(5) After setting the "minutes" to the desired value, press the IRQ1 switch. This completes initialization of the time and starts operation of the clock.
(6) The clock display is updated at 1-minute intervals. Note that the display on the LCD display ranges from "00:00" to "23:59."

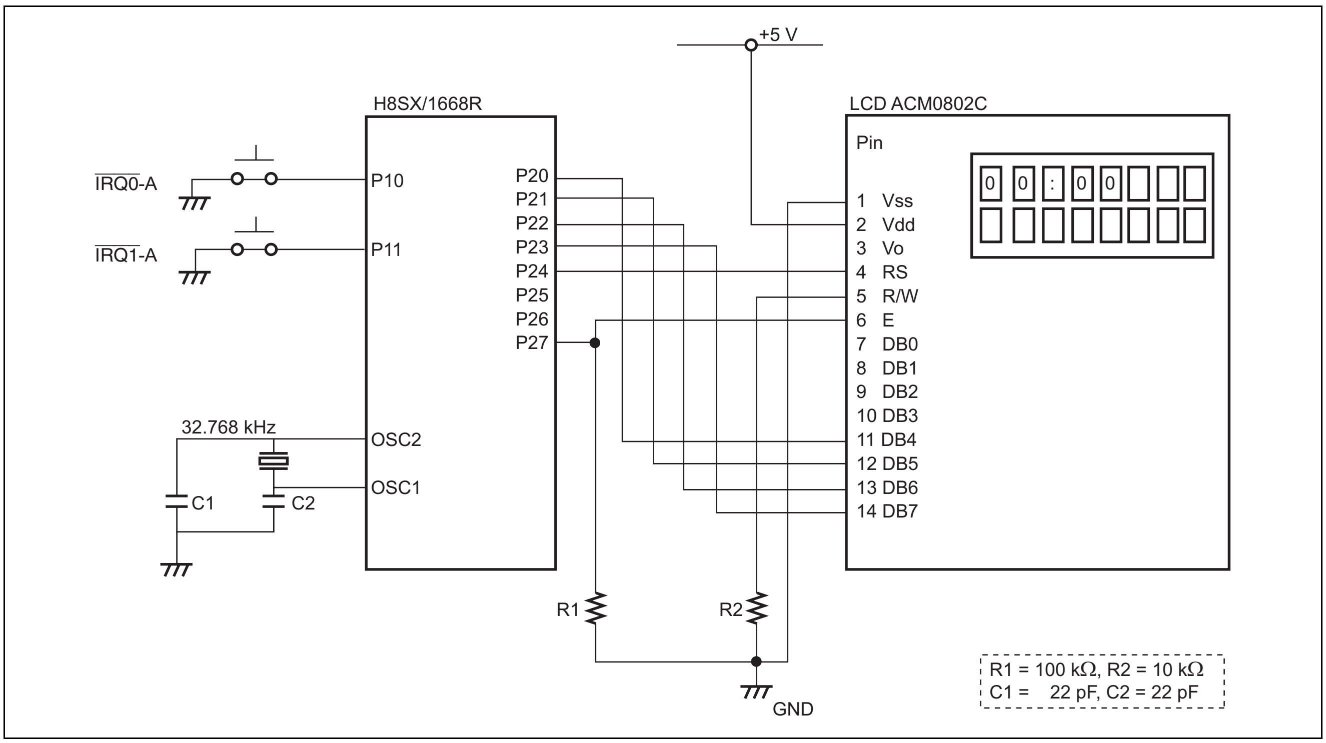


Figure 1 Configuration of the 24-Hour Clock

2. Applicable Conditions

Table 1 Applicable Conditions

Item	Setting	
Operating frequency	Input clock	12.5 MHz
	System clock (Iϕ)	50 MHz (input clock frequency × 4)
	Peripheral module clock (Pϕ)	25 MHz (input clock frequency × 2)
	External bus clock (Bϕ)	50 MHz (input clock frequency × 4)
Operating mode	Mode 7 (MD3 = 0, MD2 = 1, MD1 = 1, MD0 = 1, MD_CLK = 0)	

Table 2 Section Settings

Address	Section Name	Description
H'00000400	P, C	Program area, constant area
H'00FF4000	BCLOCK, D	RAM area

Table 3 Vector Table

Exception Handling Source	Vector No.	Address in Vector Table	Function to interrupt destination
Reset	0	H'000000	main

3. Description of Modules Used

3.1 32K Timer

3.1.1 Overview

Operation with 8- or 24-bit can be selected for the timer counting. Setting of the TME bit in TCR32K to 1 initiates counting up by the counter (TCNT32K). A 32K-timer interrupt is generated at the end of each interrupt cycle. Timer counting operations according to settings of the CKS0 bit and interrupt cycles are shown in table 4.

Table 4 Counter Settings and OVI Cycles

Setting			Dividing Ratio	Counter in Use			Interrupt Cycles	
EXCKSN	CKS1	CKS2	Internally Divided CLK	TCNT32K1	TCNT32K2	TCNT32K3	Counter Value	32KOVI Cycle
1	0	0	SUBCK/32	✓	–	–	TCNT32K1 = H'FF	250 ms
1	0	1	SUBCK/64	✓	–	–		500 ms
1	1	0	SUBCK/128	✓	–	–		1 s
1	1	1	SUBCK/256	✓	–	–		2 s
0	0	0	SUBCK/16384	✓	✓	✓	TCNT32K3 = H'3B	30 s
0	0	1	SUBCK/32768	✓	✓	✓		60 s
0	1	0	SUBCK/16384	✓	✓	✓	TCNT32K1 to3 = H'FFFF3B	Approx. 22.7 days
0	1	1	SUBCK/32768	✓	✓	✓		Approx. 45.5 days

✓: In use –: Not in use

3.1.2 Description of Operations

The counter (TCNT32K) starts counting up when the TME bit in the timer control register (TCR32K) is set to 1. A 32-K timer interrupt (32KOVI) is generated each time the counter overflows. The interrupt cycle can be changed by using the extended clock select bit and clock select bits 0 and 1 of the timer control register (TCR32K). If, for example, the extended clock select bit is 1, the 32K timer operates as shown in figure 2.

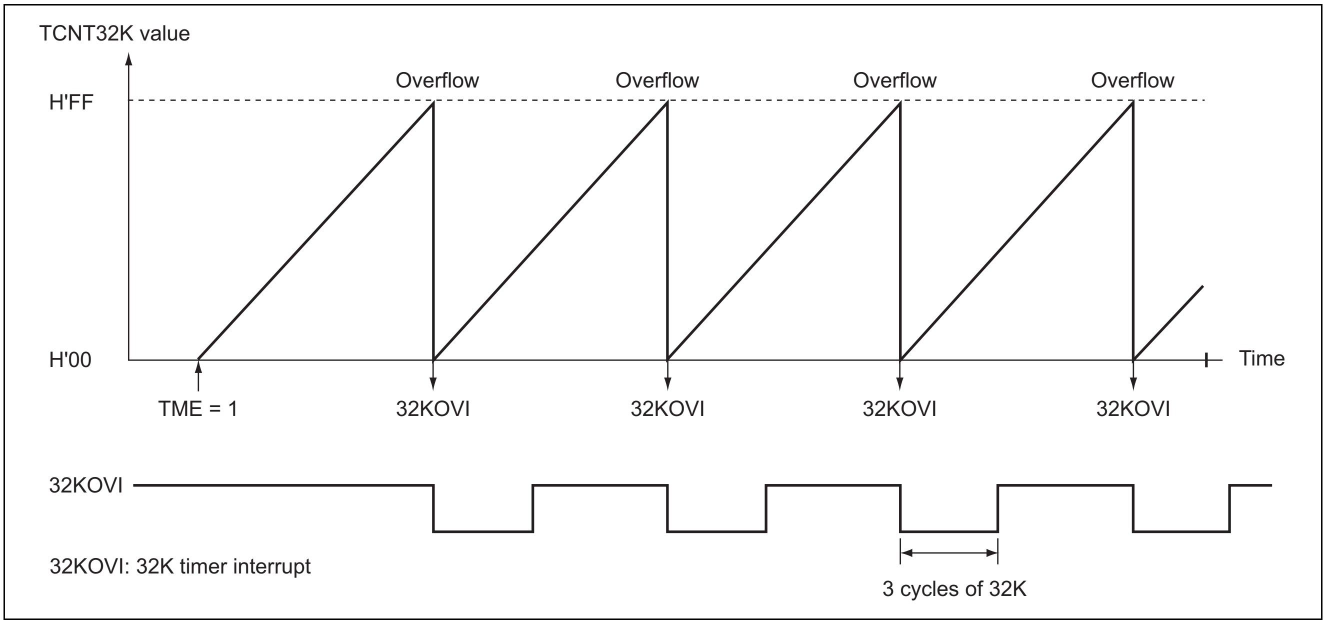


Figure 2 Operation of the 32K Timer

3.2 Deep Software Standby Mode

3.2.1 Overview

In deep software standby mode, the CPU, on-chip peripheral functions (except for the USB and 32K timer), on-chip RAM areas 6 to 4, and oscillator functionality are all halted. In addition, the internal power supply to these modules stops, resulting in a significant reduction in power consumption. The states of operation are described in table 5.

Table 5 States of Operation

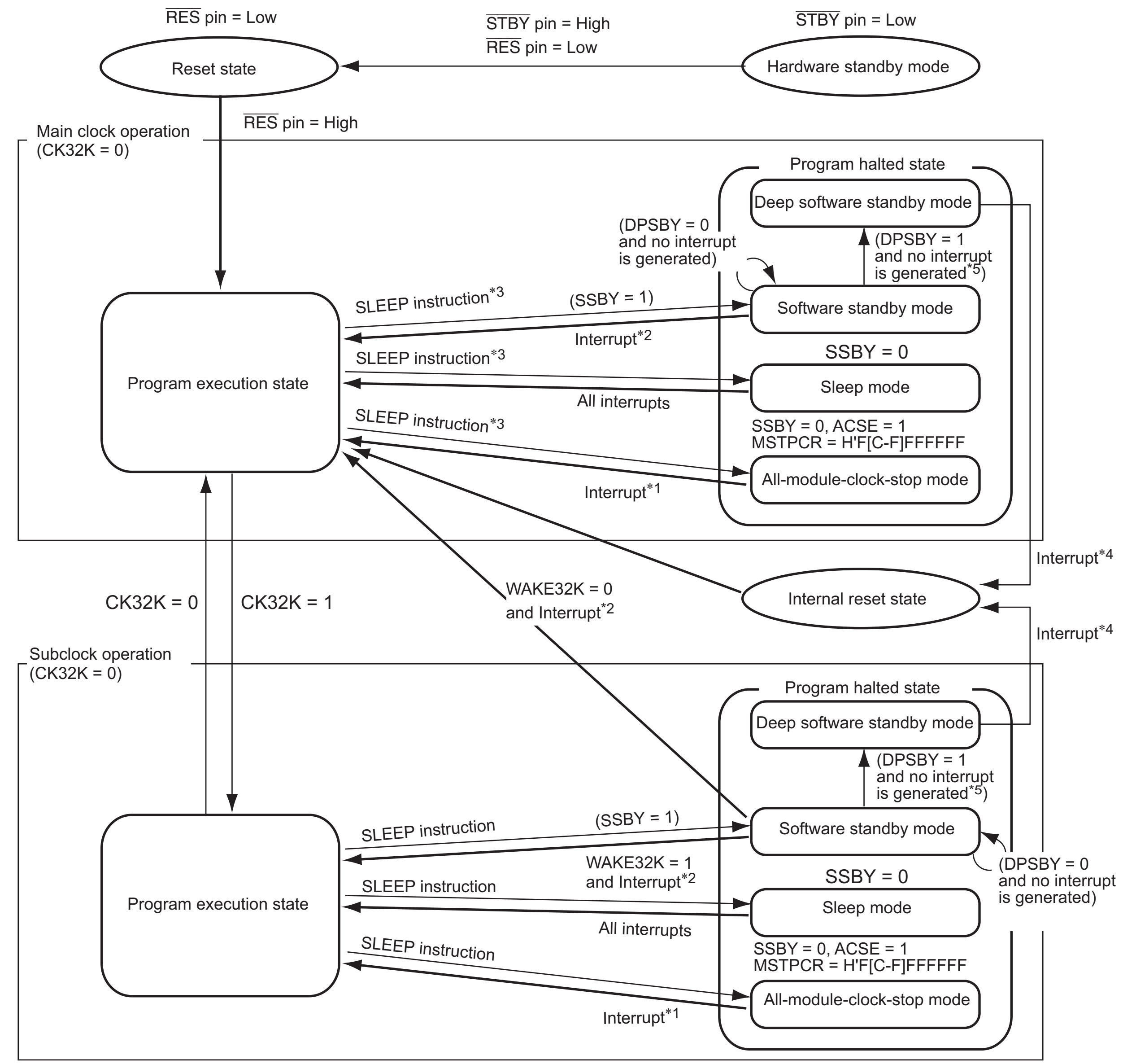
State of Operation	Sleep Mode	All-Module-Clock-Stop Mode	Software Standby Mode	Deep Software Standby Mode	Hardware Standby Mode
Transition condition	Control register + instruction	Control register + instruction	Control register + instruction	Control register + instruction	
Cancellation method	Interrupt	Interrupt*2	Interrupt*8	Interrupt*8	
Oscillator	Operating	Operating	Halted	Halted	Halted
Subclock oscillator	Operating*9	Operating*9	Operating*9	Operating*9	Halted
CPU	Halted (retained)	Halted (retained)	Halted (retained)	Halted (undefined)	Halted (undefined)
On-chip RAM areas 6 to 4 (H'FEE000 to H'FF3FFF)	Operating (retained)	Halted (retained)	Halted (retained)	Halted (undefined)	Halted (undefined)
On-chip RAM areas 3 to 0 (H'FF4000 to H'FFBFFF)	Operating (retained)	Halted (retained)	Halted (retained)	Halted (retained/ undefined)*5	Halted (undefined)
Universal Serial Bus interface	Operating	Halted (retained)	Halted (retained)	Halted (retained/ undefined)*5	Halted (undefined)
Watchdog timer	Operating	Operating	Halted (retained)	Halted (undefined)	Halted (undefined)
8-bit timer (unit 0/1)	Operating	Operating*4	Halted (retained)	Halted (undefined)	Halted (undefined)
32K timer	Operating	Operating	Operating	Operating	Halted
Other peripheral modules	Operating	Halted*1	Halted*1	Halted*7 (undefined)	Halted*3 (undefined)
I/O ports	Operating	Retained	Retained*6	Halted*6 (undefined)	Hi-Z

Notes: "Halted (retained)" in the table means that the internal values are retained and internal operations are suspended.
"Halted (undefined)" in the table means that the internal values are undefined and the power supply for internal operations is turned off.

1. SCI enters the reset state, and other peripheral modules retain their states.
2. External interrupt and some internal interrupts (8-bit timer, watchdog timer, and 32K timer).
3. All peripheral modules enter the reset state.
4. "Functioning" or "Halted" is selectable through the setting of bits MSTPA9 and MSTPA8 in MSTPCRA.
5. "Retained" or "undefined" for the contents of RAM is selected by the setting of the bits RAMCUT2 to RAMCUT0 in DPSBYCR.
6. Retention or high-impedance for the address bus and bus-control signals ($\overline{\text{CS0}}$ to $\overline{\text{CS7}}$, $\overline{\text{AS}}$, $\overline{\text{RD}}$, $\overline{\text{HWR}}$, and $\overline{\text{LWR}}$) is selected by the setting of the OPE bit in SBYCR.
7. Some peripheral modules enter a state where the register values are retained.
8. An external interrupt, 32-K timer interrupt, or USB suspend/resume interrupt.
9. Start/stop can be selected by setting the OSC32STP bit in TCR32K.

3.2.2 Transition to the Deep Software Standby Mode

When both of the SSBY bit in the standby control register (SBYCR) and the DPSBY bit in the deep standby control register (DPSBYCR) are set to 1 and the SLEEP instruction is executed, a transition to deep software standby mode will be made. The mode transitions are described in figure 3.



[Legend] ——▶ Transition after exception handling

Notes: 1. NMI, IRQ0 to IRQ11, 8-bit timer interrupts, watchdog timer interrupts, and 32K timer interrupts. Note that the 8-bit timer interrupt is valid when the MSTPCRA9 or MSTPCRA8 bit is cleared to 0.

2. NMI, IRQ0 to IRQ11, 32K timer, and USB suspend/resume interrupts. Note that IRQ, 32K timer, or USB interrupt is valid only when the corresponding bit in DPSIER is set to 1.

3. The SLPIE bit is cleared to 0.

4. NMI, IRQ0-A to IRQ3-A, 32K timer, and USB suspend/resume interrupts. Note that IRQ, 32K timer, and USB suspend/resume interrupts are valid only when the corresponding bit in DPSIER is set to 1.

5. If a conflict between a transition to deep software standby mode and generation of software standby mode clearing source occurs, a transition to deep software standby mode is not made but the software standby mode clearing sequence is executed. In this case, an interrupt exception handling for the input interrupt starts after the oscillation settling time for software standby mode (set by the STS4 to STS0 bits in SBYCR) has elapsed.

From any state, a transition to hardware standby mode occurs when STBY is driven low.

From any state except hardware standby mode, a transition to the reset state oc curs when RES is driven low.

Figure 3 Mode Transitions

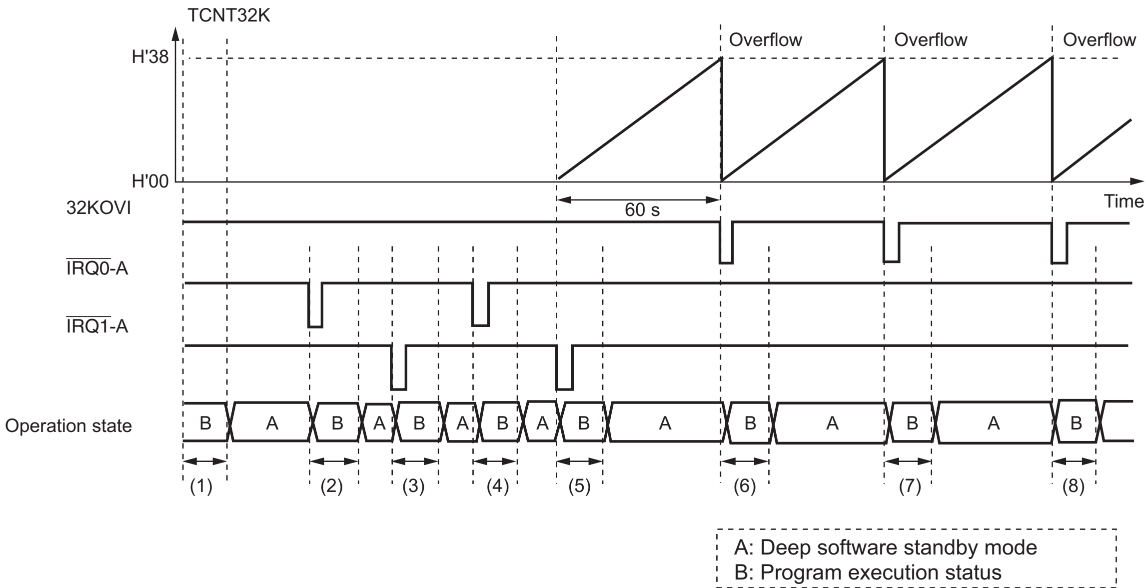
3.2.3 Cancellation of the Deep Software Standby Mode

Exit from deep software standby mode is initiated by signals on the external interrupt pins (the NMI and $\overline{\text{IRQ0}}$-A to $\overline{\text{IRQ3}}$-A), internal interrupt signals (32K timer and USB suspend/resume), $\overline{\text{RES}}$ pin, or $\overline{\text{STBY}}$ pin.

When deep software standby mode clearing source is generated, internal power supply starts simultaneously with the start of clock oscillation, and internal reset signal is generated for the entire LSI. Once the time specified by the WTSTS5 to WTSTS0 bits in a deep standby control register (DPSWCR) has elapsed, a stable clock signal is being supplied throughout the LSI and the internal reset is cleared.
Deep software standby mode is canceled on clearing of the internal reset, and then the reset exception handling starts.
When deep software standby mode is canceled by an external interrupt pin or internal interrupt signal, the DPSRSTF bit in a reset status register (RSTSR) is set to 1.

4. Principles of Operation

The timing of operations of this sample task is shown in figure 4. In this figure, note that the clock operation starts after the initial time has been set to 0101 hours.



Flow No.	Hardware Processing	Software Processing	LCD Display	
			First Row	Second Row
(1)	Power-on	1. Initializes the register. 2. Updates the LCD display. 3. Enter deep software standby mode.	00:00	set hour
(2)	Returns to the program execution status with the IRQ0 interrupt request flag.	1. Increments the "hours" data. 2. Updates the LCD display. 3. Enter deep software standby mode.	01:00	set hour
(3)	Returns to the program execution status with the IRQ1 interrupt request flag.	1. Sets the mode variable. 2. Updates the LCD display. 3. Enter deep software standby mode.	01:00	set min
(4)	Returns to the program execution status with the IRQ0 interrupt request flag.	1. Increments the "minutes" data. 2. Updates the LCD display. 3. Enter deep software standby mode.	01:01	set min
(5)	Returns to the program execution status with the IRA1 interrupt request flag.	1. Sets the mode variable. 2. Updates the 32K timer. 3. Enter deep software standby mode.	01:01	( )*
(6)	Returns to the program execution status with the 32K-timer overflow cycle interrupt request.	1. Increments the clock. 2. Updates“ the LCD display. 3. Enter deep software standby mode.	01:02	( )*
(7)			01:03	( )*
(8)			01:04	( )*

Note: * Eight half-width spaces are displayed.

Figure 4 Timing of Operations

5. Description of Software

5.1 Operating Environment

Table 6 Operating Environment

Item	Details
Development tool	High-performance Embedded Workshop Ver.4.02.00.022
C/C++ compiler	H8S, H8/300 Series C/C++ Compiler Ver.6.01.03 (manufactured by Renesas Technology)
Compiler options	-cpu = h8sxa:24:md, -optimize = 1
Linker options	-start = P, C/400, BCLOCK, D/OFF4000

5.2 List of Functions

The functions of this sample task are listed in table 7.

Table 7 List of Functions

Function Name	Description
main	• Main function Calls the init, refresh, timeupdate and lcd functions, judges the mode and sets the mode variables.
init	• Initialization function Makes initial settings for various registers and in RAM.
set	• Time set function Makes initial settings for the time, judges the mode and sets the mode variables.
mode	• Mode set function Judges the mode and sets the mode variables.
refresh	• Reset function Resets the registers.
timeupdate	• Time update function Increments the time, judges the mode and sets the mode variables.
lcd	• LCD display function Calls the InitializeDisplay and DisplayString functions and judges the mode.
InitialiseDisplay*	• LCD initialization function Initializes the LCD.
DisplayString*	• Display function Updates the LCD display.

Note: * See section 6, Documents and Programs for Reference.

5.3 Symbolic Constants

Table 8 Symbolic Constants

Constant Name	Setting	Description
RAM_mode	–	Tag for mode variables
MODE_SET_HOUR	0	• Enumeration constant for enumeration RAM_mode • "Hours" setting mode
MODE_SET_MIN	1	• Enumeration constant for enumeration RAM_mode • "Minutes" setting mode
MODE_SET_END	2	• Enumeration constant for enumeration RAM_mode • Clock operation mode
MODE_DEFAULT	3	• Enumeration constant for enumeration RAM_mode • Abnormal-operation mode
LCD_SPACE	"( )"*	Character strings to be set in the second row of the LCD
LCD_HOUR	"set hour"	
LCD_MIN	"set min "	
LCD_ERROR	"error"	

Note: * Eight half-width spaces are displayed.

5.4 RAM Usage

Table 9 RAM Usage

Type	Variable Name	Description	Used in
unsigned char	RAMmin1	First digit of minutes data (--:-X)	init, set, timeupdate, lcd
unsigned char	RAMmin2	Second digit of minutes data (--:X-)	
unsigned char	RAMhr1	First digit of hours data (-X:--)	
unsigned char	RAMhr2	Second digit of hours data (X-:--)	
unsigned char	RAMflg	Flag for causing a branch from the setting mode.	init, set, mode, timeupdate, lcd
char	RAMlcd_data[9]	Data to be passed to the LCD function for display on the LCD	lcd, DisplayString
enum	RAM_mode_variable	Variable for storing the mode	init, set, mode, timeupdate

5.5 Description of Functions

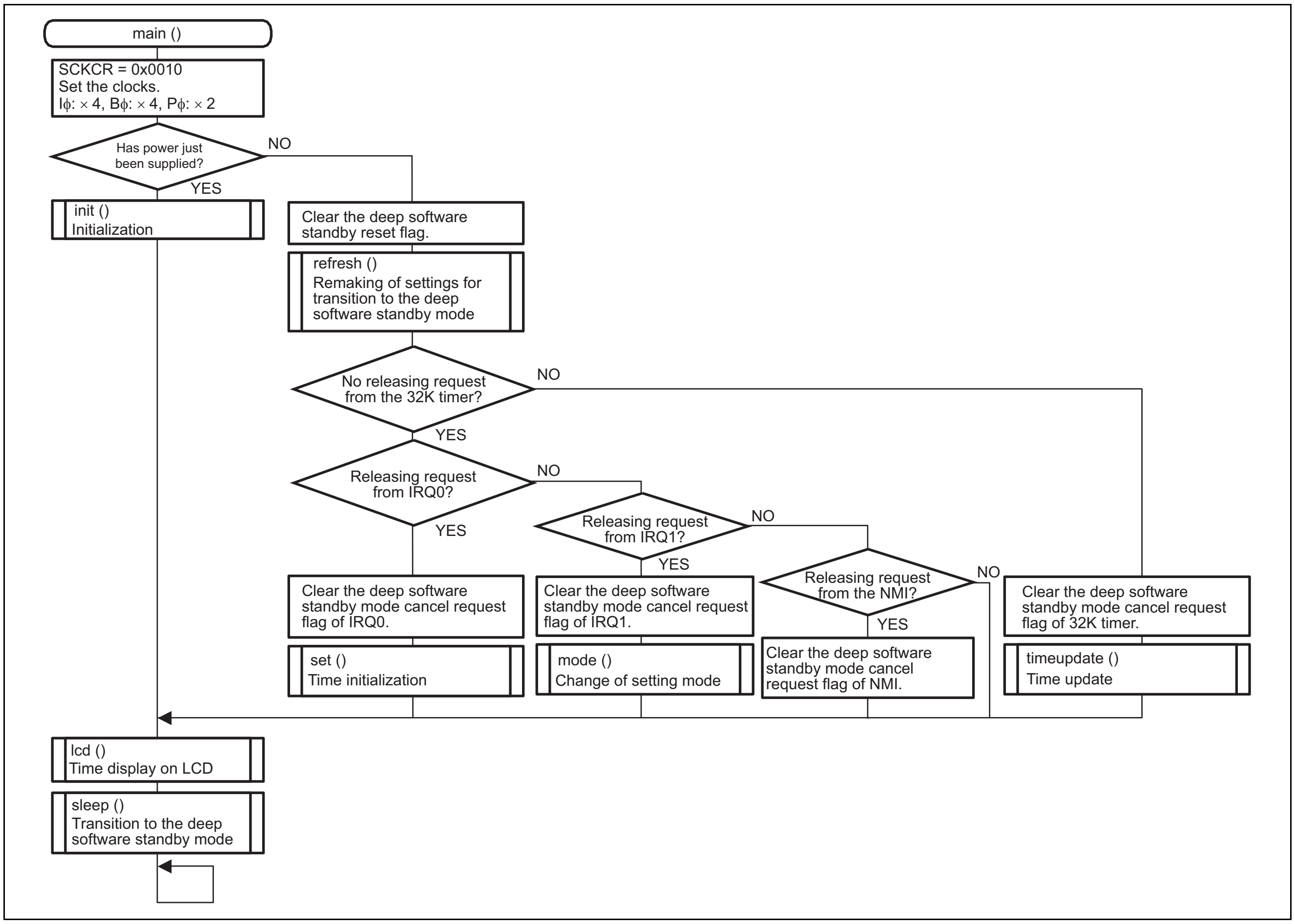
5.5.1 main Function

1. Functional overview
   Calls the init, refresh, timeupdate and lcd functions, judges the mode and sets the mode variable.
2. Arguments
   None
3. Return value
   None
4. Description of internal registers used
   The internal registers used in this sample task are described below. The settings shown in these tables are the values used in this sample task and differ from the initial values.

- System clock control register (SCKCR) Number of bits: 16 Address: H'FFFDC4

Bit	Bit Name	Setting	R/W	Description
10 9 8	ICK2 ICK1 ICK0	0 0 0	R/W	System Clock (Iϕ) Select Select the frequency of the system clock provided to the CPU, EXDMAC, DMAC, and DTC. The ratio to the input clock is as follows: ICK (2:0) 000: Input clock × 4 The frequencies of the peripheral module clock and external bus clock change to the same frequency as the system clock if the frequency of the system clock is lower than that of the two clocks.
6 5 4	PCK2 PCK1 PCK0	0 0 1	R/W	Peripheral Module Clock (Pϕ) Select Select the frequency of the peripheral module clock. The ratio to the input clock is as follows: PCK (2:0) 001: Input clock × 2 The frequency of the peripheral module clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the peripheral module clock higher than that of the system clock, the clocks will have the same frequency in reality.
2 1 0	BCK2 BCK1 BCK0	0 0 0	R/W	External-Bus Clock (Bϕ) Select Select the frequency of the external bus clock. The ratio to the input clock is as follows: BCK (2:0) 000: Input clock × 4 The frequency of the external bus clock should be lower than that of the system clock. Though these bits can be set so as to make the frequency of the external bus clock higher than that of the system clock, the clocks will have the same frequency in reality.

5. Flowchart


main ()
SCKCR = 0x0010
Set the clocks.
Iϕ: × 4, Bϕ: × 4, Pϕ: × 2
Has power just been supplied?
NO
YES
init ()
Initialization
Clear the deep software standby reset flag.
refresh ()
Remaking of settings for transition to the deep software standby mode
No releasing request from the 32K timer?
NO
YES
Releasing request from IRQ0?
NO
YES
Releasing request from IRQ1?
NO
YES
Releasing request from the NMI?
NO
YES
Clear the deep software standby mode cancel request flag of IRQ0.
Clear the deep software standby mode cancel request flag of IRQ1.
Clear the deep software standby mode cancel request flag of NMI.
Clear the deep software standby mode cancel request flag of 32K timer.
set ()
Time initialization
mode ()
Change of setting mode
timeupdate ()
Time update
lcd ()
Time display on LCD
sleep ()
Transition to the deep software standby mode

5.5.2 init Function

1. Functional overview
   Makes initial settings for registers and in RAM.
2. Arguments
   None
3. Return value
   None
4. Description of internal registers used
   The internal registers used in this sample task are described below. The settings shown in these tables are the values used in this sample task and differ from the initial values.

- Deep standby wait control register (DPSWCR) Number of bits: 8 Address: H'FFFE71

Bit	Bit Name	Setting	R/W	Description
5	WTSTS5	0	R/W	Deep Software Standby Wait Time Setting Select the time for which the MCU waits until the clock settles when deep software standby mode is canceled by an interrupt. 001101: Wait time = 131072 states (CPU cycles)
4	WTSTS4	0	R/W	
3	WTSTS3	1	R/W	
2	WTSTS2	1	R/W	
1	WTSTS1	0	R/W	
0	WTSTS0	1	R/W	

- Standby control register (SBYCR) Number of bits: 16 Address: H'FFFDC6

Bit	Bit Name	Setting	R/W	Description
15	SSBY	1	R/W	Software Standby Specifies the transition mode after executing the SLEEP instruction. 1: Enter software standby mode after execution of the SLEEP instruction is executed. This bit does not change when clearing software standby mode by using interrupts and shifting to normal operation. For clearing, write 0 to this bit.
12	STS4	0	R/W	Standby Timer Select 4 to 0 Select the time the MCU waits for the clock to settle when software standby mode is cleared by an interrupt or when a transition is made from subclock operation to main clock operation. 01110: Wait time = 262144 states (CPU cycles)
11	STS3	1	R/W	
10	STS2	1	R/W	
9	STS1	1	R/W	
8	STS0	0	R/W	

- Deep standby control register (DPSBYCR) Number of bits: 8 Address: H'FFFE70

Bit	Bit Name	Setting	R/W	Description
7	DPSBY	1	R/W	Deep Software Standby When the SSBY bit in SBYCR has been set to 1, executing the SLEEP instruction causes a transition to software standby mode. At this time, if there is no source to clear software standby mode and this bit is set to 1, a transition to deep software standby mode is made. SSBY / DPSBY / Entry to 1 / 1 / Enters deep software standby mode after execution of a SLEEP instruction. When deep software standby mode is canceled due to an interrupt, this bit remains at 1. Write a 0 here to clear it.

- Deep standby interrupt edge register (DPSIEGR) Number of bits: 8 Address: H'FFFE74

Bit	Bit Name	Setting	R/W	Description
7	DNMIEG	0	R/W	NMI Edge Select Selects the active edge for NMI pin input. 0: The interrupt request is generated by a falling edge.
3	DIRQ3EG	0	R/W	IRQ3 Interrupt Edge Select Selects the active edge for IRQ3 pin input. 0: The interrupt request is generated by a falling edge.
2	DIRQ2EG	0	R/W	IRQ2 Interrupt Edge Select Selects the active edge for IRQ2 pin input. 0: The interrupt request is generated by a falling edge.
1	DIRQ1EG	0	R/W	IRQ1 Interrupt Edge Select Selects the active edge for IRQ1 pin input. 0: The interrupt request is generated by a falling edge.
0	DIRQ0EG	0	R/W	IRQ0 Interrupt Edge Select Selects the active edge for IRQ0 pin input. 0: The interrupt request is generated by a falling edge.

- Deep standby interrupt enable register (DPSIER) Number of bits: 8 Address: H'FFFE72

Bit	Bit Name	Setting	R/W	Description
5	DT32KIE	1	R/W	32-K Timer Interrupt Enable Enables/disables exit from deep software standby mode by the 32-K timer interrupt signal. 1: Enables exit from deep software standby mode by the 32-K timer interrupt signal.

- Deep standby interrupt flag register (DPSIFR) Number of bits: 8 Address: H'FFFE73

Bit	Bit Name	Setting	R/W	Description
5	DT32KIF	0	R/W	32-K Timer Interrupt Flag [Setting condition] Generation of a 32-K timer interrupt [Clearing condition] Writing of 0 to this bit after reading it as 1
1	DIRQ1F	0	R/W	IRQ1 Interrupt Flag [Setting condition] Generation of IRQ1 input specified in DPSIEGR [Clearing conditio] Writing of 0 to this bit after reading it as 1
0	DIRQ0F	0	R/W	IRQ0 Interrupt Flag [Setting condition] Generation of IRQ0 input specified in DPSIEGR [Clearing condition] Writing of 0 to this bit after reading it as 1

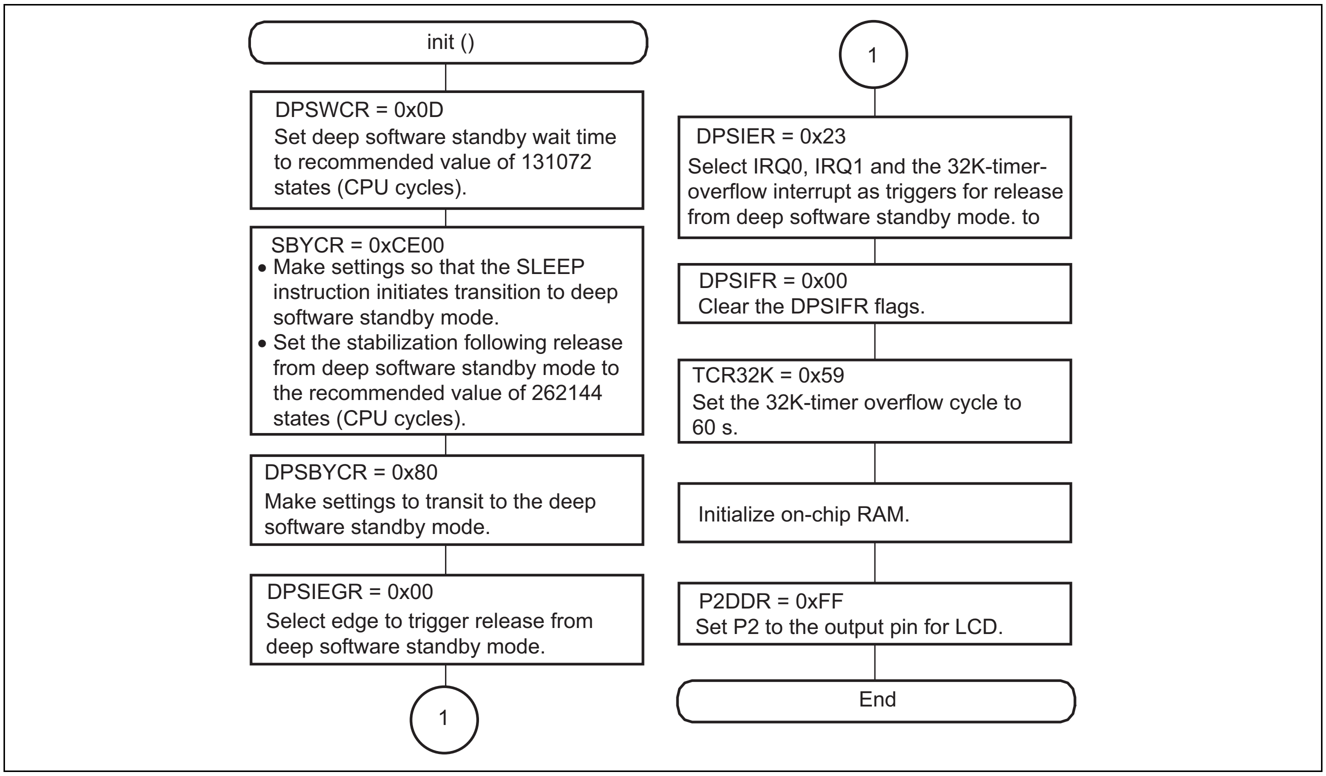
- Timer control register (TCR32K) Number of bits: 8 Address: H'FFFABC

Bit	Bit Name	Setting	R/W	Description
7	EXCKSN	0	R/W	Extended Clock Select and Clock Select 1, 0
1	CKS1	0	R/W	Select the clock source to be input to TCNT32K. The overflow
0	CKS0	1	R/W	cycle for SUBCK = 32.768 kHz is indicated in parentheses. When EXCKSN = 0: 01: Clock SUBCK/32768 (cycle: 60 s)

- Data direction register (P2DDR) Number of bits: 8 Address: H'FFFB81

Bit	Bit Name	Setting	R/W	Description
7	P27DDR	1	W	DDR is an 8-bit write-only register that specifies the port input or
6	P26DDR	1	W	output for each bit. A read from the DDR is invalid and DDR is
5	P25DDR	1	W	always read as an undefined value.
4	P24DDR	1	W	When the general I/O port function is selected, the corresponding
3	P23DDR	1	W	pin functions as an output port by setting the corresponding DDR
2	P22DDR	1	W	bit to 1; the corresponding pin functions as an input port by
1	P21DDR	1	W	clearing the corresponding DDR bit to 0.
0	P20DDR	1	W	

5. Flowchart


init ()
DPSWCR = 0x0D
Set deep software standby wait time to recommended value of 131072 states (CPU cycles).
SBYCR = 0xCE00
• Make settings so that the SLEEP instruction initiates transition to deep software standby mode.
• Set the stabilization following release from deep software standby mode to the recommended value of 262144 states (CPU cycles).
DPSBYCR = 0x80
Make settings to transit to the deep software standby mode.
DPSIEGR = 0x00
Select edge to trigger release from deep software standby mode.
1
1
DPSIER = 0x23
Select IRQ0, IRQ1 and the 32K-timer-overflow interrupt as triggers for release from deep software standby mode. to
DPSIFR = 0x00
Clear the DPSIFR flags.
TCR32K = 0x59
Set the 32K-timer overflow cycle to 60 s.
Initialize on-chip RAM.
P2DDR = 0xFF
Set P2 to the output pin for LCD.
End

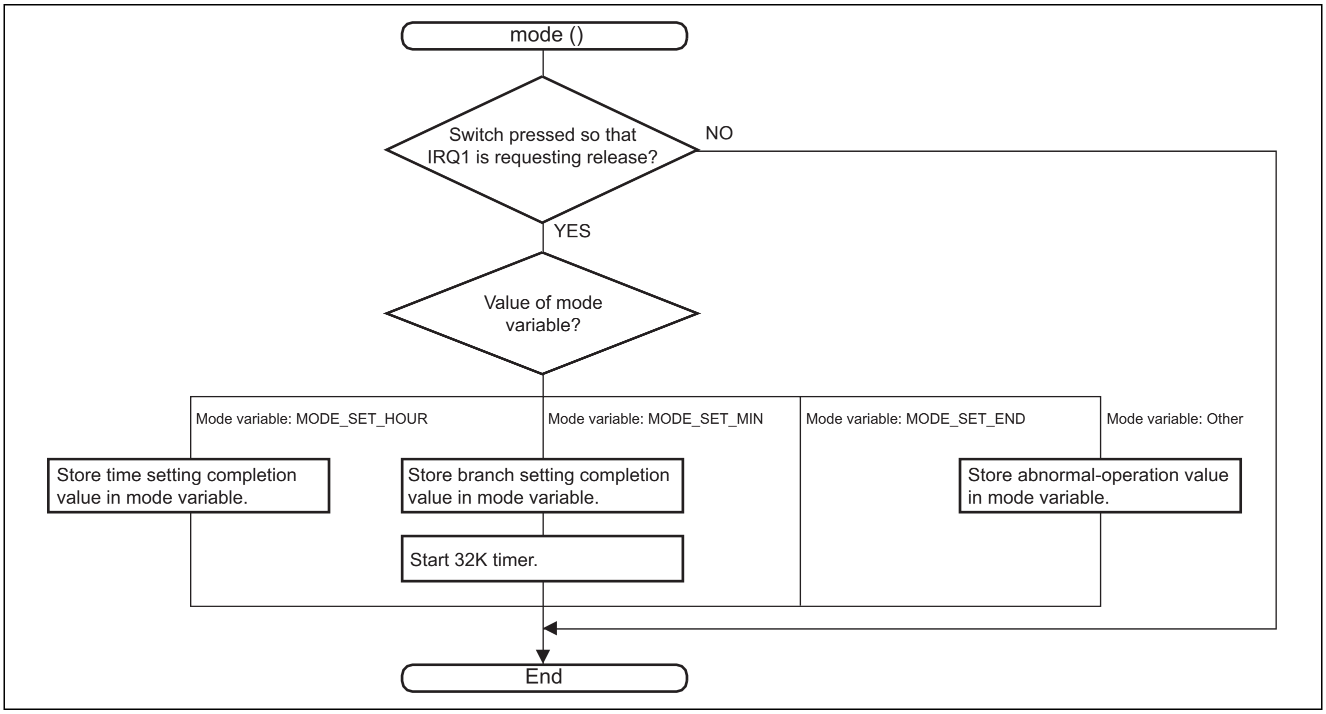
5.5.3 mode Function

1. Functional overview
   Judges the mode and sets the mode variables.
2. Arguments
   None
3. Return value
   None
4. Description of internal registers used
   The internal register used in this sample task is described below. The setting shown in the table is the value used in this sample task and differs from the initial values.

- Timer control register (TCR32K) Number of bits: 8 Address: H'FFFABC

Bit	Bit Name	Setting	R/W	Description
5	TME	1	R/W	Timer Enable When this bit is set to 1, TCNT32K starts counting. When it is cleared to 0, TCNT32K stops counting and is initialized to H'00_00_00.

5. Flowchart



5.5.4 set Function

1. Functional overview
   Makes initial settings for the time, judges the mode and sets the mode variables.

2. Arguments
   None

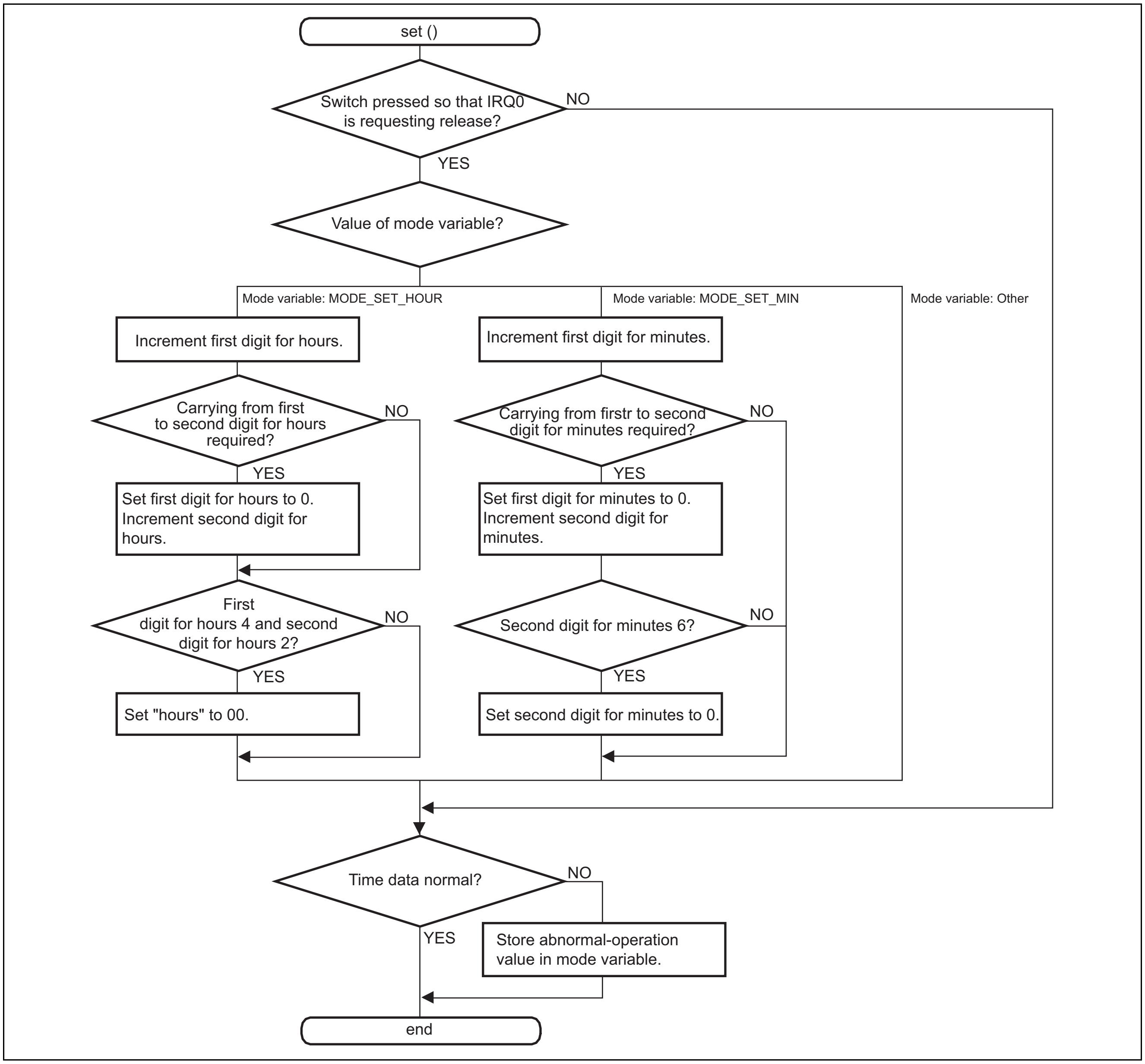
3. Return value
   None

4. Description of internal registers used
   The internal register used in this sample task is described below. The setting below is the value used in this sample task and differs from the initial values.

- Port register (PORT1) Number of bits: 8 Address: H'FFFF40

Function: PORT is an 8-bit read-only register that reflects the port pin state. A write to PORT is invalid. When PORT is read, the DR bits that correspond to the respective DDR bits set to 1 are read and the status of each pin whose corresponding DDR bit is cleared to 0 is also read regardless of the ICR value. The initial value of PORT is undefined and is determined based on the port pin state.

5. Flowchart


set ()
Switch pressed so that IRQ0 is requesting release?
NO
YES
Value of mode variable?
Mode variable: MODE_SET_HOUR
Mode variable: MODE_SET_MIN
Mode variable: Other
Increment first digit for hours.
Increment first digit for minutes.
Carrying from first to second digit for hours required?
NO
YES
Carrying from firstr to second digit for minutes required?
NO
YES
Set first digit for hours to 0. Increment second digit for hours.
Set first digit for minutes to 0. Increment second digit for minutes.
First digit for hours 4 and second digit for hours 2?
NO
YES
Second digit for minutes 6?
NO
YES
Set "hours" to 00.
Set second digit for minutes to 0.
Time data normal?
NO
YES
Store abnormal-operation value in mode variable.
end

5.5.5 refresh Function

1. Functional overview
   Resets the registers.
2. Arguments
   None
3. Return value
   None
4. Description of internal registers used
   The internal registers used in this sample task are described below. The settings shown in these tables are the values used in this sample task and differ from the initial values.

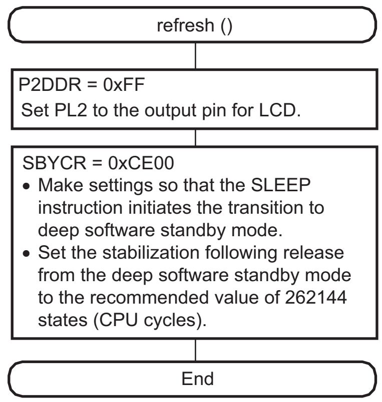
- Data direction register (P2DDR) Number of bits: 8 Address: H'FFFB81

Bit	Bit Name	Setting	R/W	Description
7	P27DDR	1	W	DDR is an 8-bit write-only register that specifies the port input or output for each bit. A read from the DDR is invalid and DDR is always read as an undefined value. When the general I/O port function is selected, the corresponding pin functions as an output port by setting the corresponding DDR bit to 1; the corresponding pin functions as an input port by clearing the corresponding DDR bit to 0.
6	P26DDR	1	W	
5	P25DDR	1	W	
4	P24DDR	1	W	
3	P23DDR	1	W	
2	P22DDR	1	W	
1	P21DDR	1	W	
0	P20DDR	1	W	

- Standby control register (SBYCR) Number of bits: 16 Address: H'FFFDC6

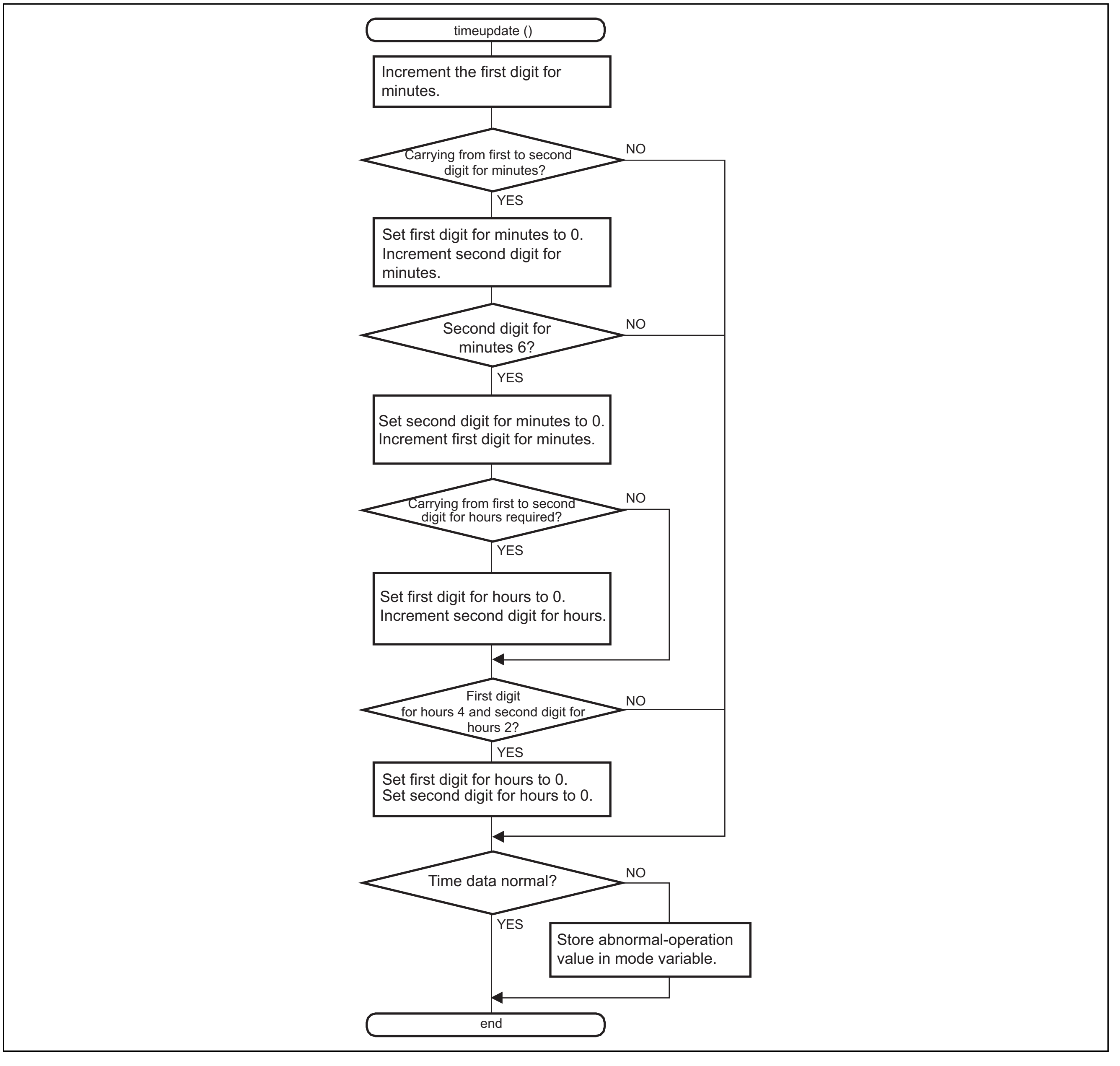
Bit	Bit Name	Setting	R/W	Description
15	SSBY	1	R/W	Software Standby Specifies the transition mode after executing the SLEEP instruction. 1: Enters software standby mode after the SLEEP instruction is executed
12	STS4	0	R/W	Standby Timer Select 4 to 0 Select the time the MCU waits for the clock to settle when software standby mode is cleared by an interrupt or when a transition is made from subclock operation to main clock operation. 01110: Wait time = 262144 states (CPU cycles)
11	STS3	1	R/W	
10	STS2	1	R/W	
9	STS1	1	R/W	
8	STS0	0	R/W	

5. Flowchart


refresh ()
P2DDR = 0xFF
Set PL2 to the output pin for LCD.
SBYCR = 0xCE00
• Make settings so that the SLEEP instruction initiates the transition to deep software standby mode.
• Set the stabilization following release from the deep software standby mode to the recommended value of 262144 states (CPU cycles).
End

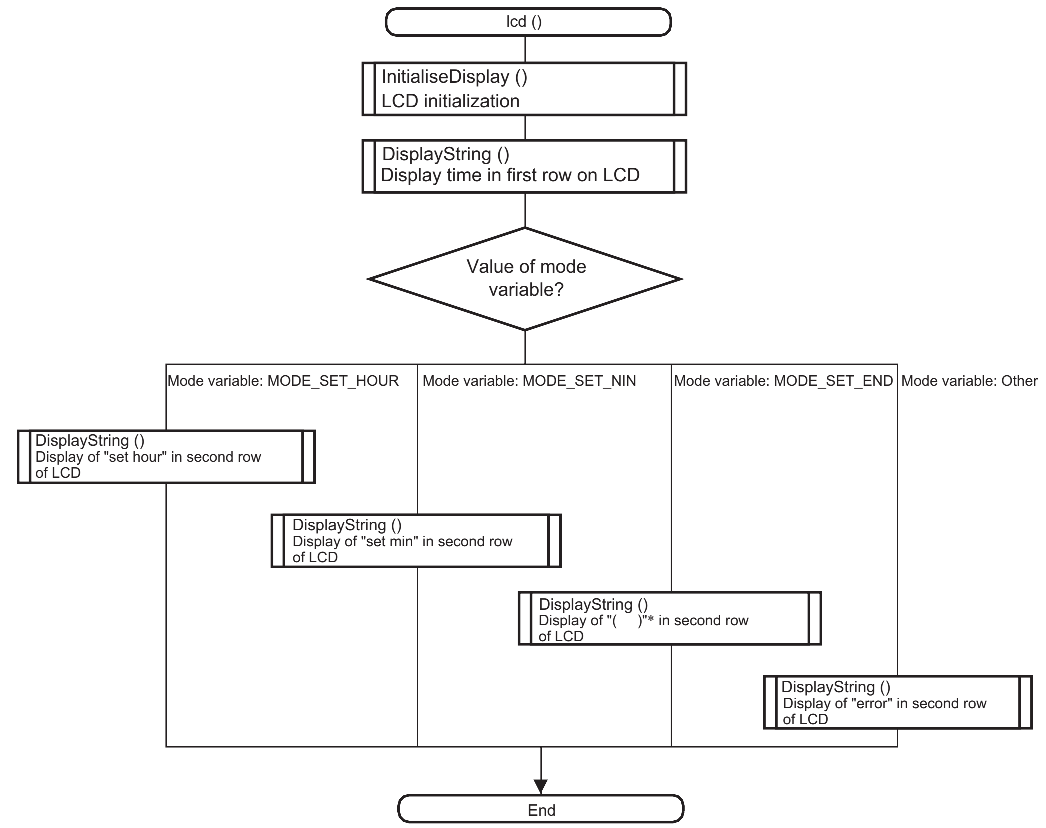
5.5.6 timeupdate Function

1. Functional overview
   Increments the time, judges the mode and sets the mode variables.
2. Arguments
   None
3. Return value
   None
4. Flowchart



5.5.7 lcd Function

1. Functional overview
   Calls the InitialiseDisplay and DisplayString functions and judges the mode.
2. Arguments
   None
3. Return value
   None
4. Flowchart



Note: * Eight half-width spaces are displayed.

5.5.8 InitialiseDisplay Function*

1. Functional overview
   Initializes the LCD.
2. Arguments
   None
3. Return value
   None

Note: * See section 6, Documents and Programs for Reference.

5.5.9 DisplayString Function*

1. Functional overview
   Displays character strings on the LCD.
2. Arguments
   LCD display contents and display position (which is either the first or second row)
3. Return value
   None

Note: * See section 6, Documents and Programs for Reference.

5.6 Notes

- If an IRQ interrupt request causes the system to return from deep software standby mode to the program execution state using the input buffer control register (ICR) to enable the input buffer is unnecessary.
- If an IRQ interrupt flag is to cause the system to return from deep software standby mode to the program execution state, use pins $\overline{\text{IRQ0}}$-A to $\overline{\text{IRQ3}}$-A. If you use pins $\overline{\text{IRQ0}}$-B to $\overline{\text{IRQ3}}$-B, the system will not be capable of returning to the program execution state.
- Some registers are initialized while others are not when the system is returned to the program execution state from deep software standby mode.
- When the system is returned to the program execution state from deep software standby mode, the amount of current consumption is increased during the period of waiting for stable oscillation by the MCU.

6. Documents and Programs for Reference

6.1 Documents for Reference

- Hardware Manual
  H8SX/1668R Group Hardware Manual
  The most up-to-date version of this document is available on the Renesas Technology Website.
- Technical News/Technical Update
  The most up-to-date information is available on the Renesas Technology Website.

6.2 Programs for Reference

- InitialiseDisplay Function
  This LCD display function is included in a sample program for Renesas Starter Kit (RSK).
  The most up-to-date version of information is available on the Renesas Technology Website.
- DisplayString Function
  This LCD display function is included in a sample program for Renesas Starter Kit (RSK).
  The most up-to-date version of information is available on the Renesas Technology Website.

Website and Support

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

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

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.18.08	—	First edition issued