RZ/A1H Group

Example of Using Real Time Clock (Preliminary Version)

R01AN2189EJ0081
Rev.0.81
Sep. 05, 2014

Abstract

This document describes an example of the real time clock (hereinafter called "RTC") for the RZ/A1H.

Products

RZ/A1H

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

The RTC time (Second, minute, hour, day of the week, day, month, and year) should be set to start the time count operation using RTC. After starting the time count operation, the time is read from RTC and displayed on the terminal on the host PC. Then the RZ/A1H transits to the power-down mode (deep standby mode) and returns from the mode when the RTC alarm interrupt is generated. These basic operations with RTC can be executed by inputting commands from the terminal. Note that the time does not need to be reset after returning from deep standby mode because RTC continues the time count operation during deep standby mode.

In this application note, the serial communication interface with FIFO and the power-down mode are abbreviated as SCIF and STB respectively.

Table 1.1 lists the Peripheral Functions and Their Applications and Figure 1.1 shows the Operation Overview.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
RTC	Used to set and display time using clock/calendar functions. Uses alarm interrupt as deep standby mode cancel source.
SCIF	For communication between SCIF channel 2 and the host PC.
STB	Used for clock supply to RTC and for transition to and release from deep standby mode.

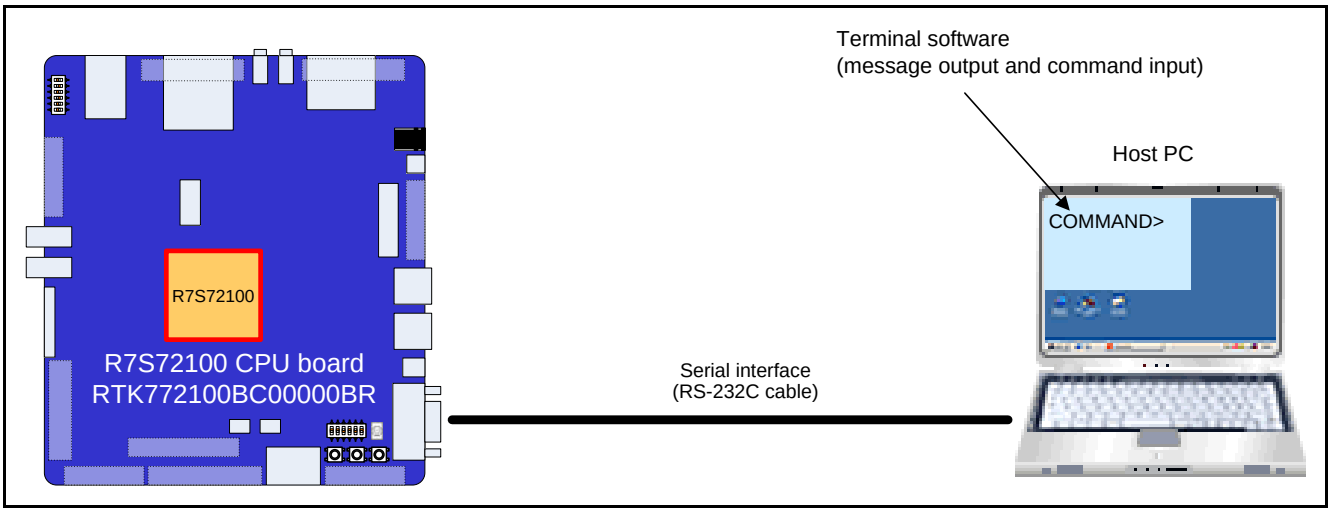


Figure 1.1 Operation Overview

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	RZ/A1H
Operating frequency*	CPU clock (Iϕ): 400MHz Image processing clock (Gϕ): 266.67MHz Internal bus clock (Bϕ): 133.33MHz Peripheral clock 1 (P1ϕ): 66.67MHz Peripheral clock 0 (P0ϕ): 33.33MHz
Operating voltage	Power supply voltage (I/O): 3.3V Power supply voltage (Internal): 1.18V
Integrated development environment	ARM® integrated development environment ARM Development Studio 5 (DS-5™) Version 5.16
C compiler	ARM C/C++ Compiler/Linker/Assembler Ver.5.03 [Build 102]
	Compiler options (excluding addition of directory path) -O3 -Ospace --cpu=Cortex-A9 --littleend --arm --apcs=/interwork --no_unaligned_access --fpu=vfpv3_fp16 -g --asm
Operating mode	Boot mode 0 (CS0-space 16-bit booting)
Terminal software communication setting	• Communication speed: 115200bps • Data length: 8 bits • Parity: None • Stop bit length: 1 bit • Flow control: None
Board used	GENMAI board • RTK772100BC00000BR (R7S72100 CPU board)
Device used	• NOR flash memory (Connected to CS0 and CS1 spaces) Manufacturer: Spansion Inc. Part No.: S29GL512S10TFI01 • Serial inter face (D-sub 9-pin connector) • LED1

Note: * The operating frequency used in clock mode 0 (Clock input 13.33MHz from EXTAL pin).

3. Reference Application Notes

For additional information associated with this document, refer to the following application notes.

- RZ/A1H Group I/O definition header file <iodefine.h> (R01AN1860EJ)
- RZ/A1H Group Example of Initialization (R01AN1864EJ)

4. Peripheral Functions

This chapter provides supplementary information on RTC. Refer to the "RZ/A1H Group User's Manual: Hardware" for basic information.

RTC has registers(hereinafter collectively called "time counter") which perform BCD coding for second, minute, hour, day of week, day, month, and year to count time. After the time counter has been set and the time count operation starts, the time counter can be used as the information of current time (second, minute, hour, day of week, day, month, and year). Note that the time count operation continues after transition to deep standby mode because the on-chip crystal oscillator circuit does not stop during power-down mode (deep standby mode).

RTC supports the alarm function and generates alarm interrupt with any of or combination of second, minute, hour, day of week, day, month, and year. The interrupt using alarm function can be used as the deep standby mode cancel source. After the alarm time has been set, the RZ/A1H transits to deep standby mode and then returns from it at the alarm time. Note that if the alarm interrupt is used as the deep standby mode cancel source, it runs as the cancel source regardless of the setting value of the running priority register for the interrupt controller. Therefore it is not necessary to enable the alarm interrupt using the alarm interrupt enable flag.

5. Hardware

5.1 Pins Used

Table 5.1 lists the Pins Used and Their Functions.

Table 5.1 Pins Used and Their Functions

Pin Name	I/O	Function
A25 to A1	Output	Address signal output to NOR flash memory
D15 to D0	Input/output	Data signal input/output of NOR flash memory
CS0#	Output	Device select signal output to NOR flash memory connected to CS0 space
RD#	Output	Read control signal output to NOR flash memory
WE0#	Output	Write enable control signal output to NOR flash memory
MD_BOOT1	Input	Selects boot mode MD_BOOT1: "L", MD_BOOT0: "L" (Set to boot mode 0)
MD_BOOT0	Input	
P4_10	Output	LED on/off
RxD2	Input	Serial receive data signal
TxD2	Output	Serial transmit data signal
RTC_X1	Input	Used to connect a 32.768kHz crystal resonator for RTC
RTC_X2	Output	

Note: The symbol # indicates negative logic (or active low).

6. Software

6.1 Operation Overview

This sample code executes six types of sample operations using RTC. The sample operations can be performed by inputting commands from the terminal. Table 6.1 lists the Sample Operations.

The basic command input procedure to operate sample code is listed as follows.

(1) Turn ON the GENMAI board under the operation environment shown in Figure 1.1.
(2) Input "1"+"Enter" from the terminal and initialize RTC.
(3) Input "2"+"Enter" from the terminal and set the RTC time counter using the time input according to the message displayed on the terminal.
(4) Input "4"+"Enter" from the terminal and start the RTC time count operation.
(5) Input "3"+"Enter" from the terminal and display the current time on the terminal.
(6) Input "6"+"Enter" from the terminal and set the RTC alarm time using the time input according to the message displayed on the terminal. Then transit to deep standby mode.
(7) Return from deep standby mode when the value of time counter indicates the alarm time set in step (6) by using the time count operation in deep standby mode.

Table 6.1 Sample Operations

Sample operation	Outline	Command
RTC initial setting	Initializes RTC. The RTC time count operation is stopped.	1
RTC time setting	Sets current time to time counter.	2
RTC time display	Displays current time using time counter.	3
Starting RTC time count operation	Starts the RTC time count operation.	4
Stopping RTC time count operation	Stops the RTC time count operation.	5
RTC alarm time setting and transition to deep standby mode	Sets RTC alarm time and transits to deep standby mode. Returns from deep standby mode at the alarm time.	6

Note: Command 1 should be executed before executing Commands 2 to 6.

6.1.1 RTC Initial Setting (Command 1)

Set STB to supply a clock to RTC, and initialize RTC after the time count operation is stopped.

When this command has been executed, the RTC time count operation is in stopped state. Note that this command should be executed before executing other commands.

Figure 6.1 shows the Terminal Display for Initial Setting.

```
RTC SAMPLE> 1

RTC initialize complete.
```

Figure 6.1 Terminal Display for Initial Setting

6.1.2 RTC Time Setting (Command 2)

Sets the RTC time counter (Second, minute, hour, day of week, day, month, and year). Input the time by using decimal digit in the order of day of week, month, day, year, hour, minute, and second. Note that no value is set to the RTC time counter if "-1" is input. The time count operation starts with the initial value listed in "RZ/A1H group User's Manual: Hardware" when starting the time count operation without setting time using this command after the GENMAI board turned ON.

Table 6.2 Table of Time Input

Time	Value can be Input (Decimal digit)
Day of week	0 to 6 0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday -1: No change
Month	1 to 12 -1: No change
Day	1 to 31 -1: No change
Year	0 to 9999 -1: No change
Hour	0 to 23 -1: No change
Minute	0 to 59 -1: No change
Second	0 to 59 -1: No change

Figure 6.2 shows the example to set "Tuesday April 1st, 2014 at 10:15" ("Seconds" is not used.)

```
RTC SAMPLE> 2

Enter time (decimal).
Enter "-1" to the item where you do not want to change the time.

Day of week (Sun=0/Mon=1/Tue=2/Wed=3/Thu=4/Fri=5/Sat=6) : 2
Month (1 - 12) : 4
Day (1 - 31) : 1
Year (0 - 9999) : 2014
Hour (0 - 23) : 10
Minute (0 - 59) : 15
Second (0 - 59) : -1

RTC time counter setting complete.
```

Figure 6.2 Example of Time Setting

6.1.3 RTC Time Display (Command 3)

Obtain BCD-coded time information from the RTC time counter and display it on the terminal in the form of "Day of week month day, year at hour: minute: second".

Figure 6.3 shows the example of "Tuesday April 1st, 2014 at 10:15:45".

```
RTC SAMPLE> 3

------------------------------
Tue. Apr.  1, 2014 at 10:15:45
------------------------------
```

Figure 6.3 Example of Time Display

6.1.4 Starting RTC Time Count Operation (Command 4)

Start the RTC time count operation. Display the time to start it.

Figure 6.4 shows the Example of Starting of Time Count Operation on "Tuesday April 1st, 2014 at 10:15:10".

```
RTC SAMPLE> 4

------------------------------
Tue. Apr.  1, 2014 at 10:15:10
------------------------------

RTC time count start.
```

Figure 6.4 Example of Starting of Time Count Operation

6.1.5 Stopping RTC Time Count Operation (Command 5)

Stop the RTC time count operation. Display the time to stop it.

Figure 6.5 shows the Example of Stopping of Time Count Operation on "Tuesday April 1st, 2014 at 10:15:45".

```
RTC SAMPLE> 5

RTC time count stop.

------------------------------
Tue. Apr.  1, 2014 at 10:15:45
------------------------------
```

Figure 6.5 Example of Stopping of Time Count Operation

6.1.6 RTC Alarm Time Setting and Transition to Deep Standby Mode (Command 6)

The following is performed when executing this command.

(1) Input time to be set to the RTC alarm time from the terminal.
(2) Set the time input in step (1) to the alarm time.
(3) Obtain current time from the RTC time counter and display it on the terminal as transition time to deep standby mode.
(4) Set the deep standby mode cancel source to the alarm interrupt and transit to deep standby mode.
(5) Return from deep standby mode when the value of time counter indicates the alarm time set in step (2) by using time count operation in deep standby mode.
(6) Check the deep standby mode cancel source. When the cancel source is the RTC alarm interrupt, obtain time from the time counter and display it on the terminal as the deep standby mode cancel time.

Figure 6.6 shows the example when the deep standby mode cancel time is "10:20" and transition to deep standby mode on "Tuesday April 1st, 2014 at 10:15:45".

```
RTC SAMPLE> 6

Transit to deep standby mode.
Enter time to cancel deep standby mode (decimal).
Enter "-1" to the item where you set current time.

Hour (0 - 23) : 10
Minute (0 - 59) : 20

Transit to deep standby mode at..
------------------------------
Tue. Apr.  1, 2014 at 10:15:45
------------------------------

Deep standby mode will be canceled at 10:20:00.


RZ/A1H CPU Board Sample Program. Ver.X.XX
Copyright (C) 2014 Renesas Electronics Corporation. All rights reserved.

Deep standby mode canceled at..
------------------------------
Tue. Apr.  1, 2014 at 10:20:00
------------------------------

select sample program.

SAMPLE>
```

Figure 6.6 Example of Alarm Time Setting and Transition to Deep Standby Mode

The deep standby mode cancel time is stored in the sample function STB_CancelDeepStandby called by the initial setting function for peripheral function ($Sub$$main). The stored time is displayed on the terminal using the sample function RTC_DispTimeCanDeepStb called by the main function.

6.2 Peripheral Functions and Memory Allocation in Sample Code

6.2.1 Setting for Peripheral Functions

Table 6.3 Peripheral Function Settings

Module	Setting
RTC	Operating clock: Select 32.768kHz from RTC_X1 Periodic interrupt generation: Disabled Carry interrupt generation: Disabled Alarm interrupt generation: Disabled
SCIF	Sets the channel 2 in asynchronous communication mode. • Data length: 8 bits • Stop bit length: 1 bit • Parity: None Sets the clock source without frequency dividing and the bit rate value at 17. Sets the bit rate to be 115200pbs when P1ϕ is 66.67MHz. Difference is 0.46%.
STB	Clock supply to RTC. Transition to deep standby mode. Deep standby mode is cancelled when the RTC alarm interrupt is generated. Startup from the external memory after deep standby mode is cancelled.

6.2.2 Section Assignment in Sample Code

Table 6.4 and Table 6.5 list the Sections Used in this sample code. Figure 6.7 shows the Section Assignment for the initial condition of the sample code and the condition after using the scatter loading function.

Refer to "Image structure and generation" in "ARM Compiler toolchain Using the Linker" provided by the ARM for more information about the section and the scatter-loading function.

Table 6.4 Sections to be Used (1/2)

Area Name	Description	Type	Loading Area	Execution Area
VECTOR_TABLE	Exception processing vector table	Code	FLASH	FLASH
RESET_HANDLER	Program code area of reset handler processing This area consists of the following sections. • INITCA9CACHE (L1 cache setting) • INIT_TTB (MMU setting) • RESET_HANDLER (Reset handler)	Code	FLASH	FLASH
CODE_BASIC_SETUP	Program code area to optimize operating frequency and flash memory	Code	FLASH	FLASH
InRoot	This area consists of the sections located in the root area such as C standard library.	Code and RO Data	FLASH	FLASH
CODE_FPU_INIT	Program code area for NEON and VFP initializations This area consists of the following sections. • CODE_FPU_INIT • FPU_INIT	Code	FLASH	FLASH
CODE_RESET	Program code area for hardware initializations This area consists of the following sections. • CODE_RESET (Startup processing) • INIT_VBAR (Vector base setting)	Code	FLASH	FLASH
CODE_IO_REGRW	Program code area for read/write function of I/O register	Code	FLASH	FLASH
CODE	Program code area for defaults All the Code type sections which do not define section names with C source are assigned in this area.	Code	FLASH	FLASH
CONST	Constant data area for defaults All the RO Data type sections which do not define section names with C source are assigned in this area.	RO Data	FLASH	FLASH

Table 6.5 Sections to be Used (2/2)

Area Name	Description	Type	Load Area	Execution Area
VECTOR_MIRROR_TABLE	Exception processing vector table (Section to transfer data to large-capacity on-chip RAM)	Code	FLASH	LRAM
CODE_HANDLER_JMPTBL	Program code area for user-defined functions of IRQ interrupt handler	Code	FLASH	LRAM
CODE_HANDLER	Program code area of IRQ interrupt handler This area consists of the following sections. • CODE_HANDLER • IRQ_FIQ_HANDLER	Code	FLASH	LRAM
DATA_HANDLER_JMPTBL	Registration table data area for user-defined functions of IRQ interrupt handler	RW Data	FLASH	LRAM
DATA_RESET	Data area with initial value for hardware initializations	RW Data	FLASH	LRAM
BSS_RESET	Data area without initial value for hardware initializations	ZI Data	-	LRAM
ARM_LIB_STACK	Application stack area	ZI Data	-	LRAM
IRQ_STACK	IRQ mode stack area	ZI Data	-	LRAM
FIQ_STACK	FIQ mode stack area	ZI Data	-	LRAM
SVC_STACK	Supervisor (SVC) mode stack area	ZI Data	-	LRAM
ABT_STACK	Abort (ABT) mode stack area	ZI Data	-	LRAM
TTB	MMU translation table area	ZI Data	-	LRAM
ARM_LIB_HEAP	Application heap area	ZI Data	-	LRAM
DATA	Data area with initial value for defaults All the RW Data type sections which do not define section names with C source are assigned in this area.	RW Data	FLASH	LRAM
BSS	Data area without initial value for defaults All the ZI Data type sections which do not define section names with C source area assigned in this area.	ZI Data	-	LRAM

Notes: 1. "FLASH" and "LRAM" shown in Loading Area and Execution Area indicate the NOR flash memory area and the large-capacity on-chip RAM area respectively.

2. Basically the section name is set to be the same as the region's, however it consists of some sections in the areas of RESET_HANDLER, InRoot, CODE_FPU_INIT, CODE_RESET, CODE, CONST, CODE_HANDLER, DATA, and BSS. Refer to the ARM compiler toolchain manual about the region and the section.

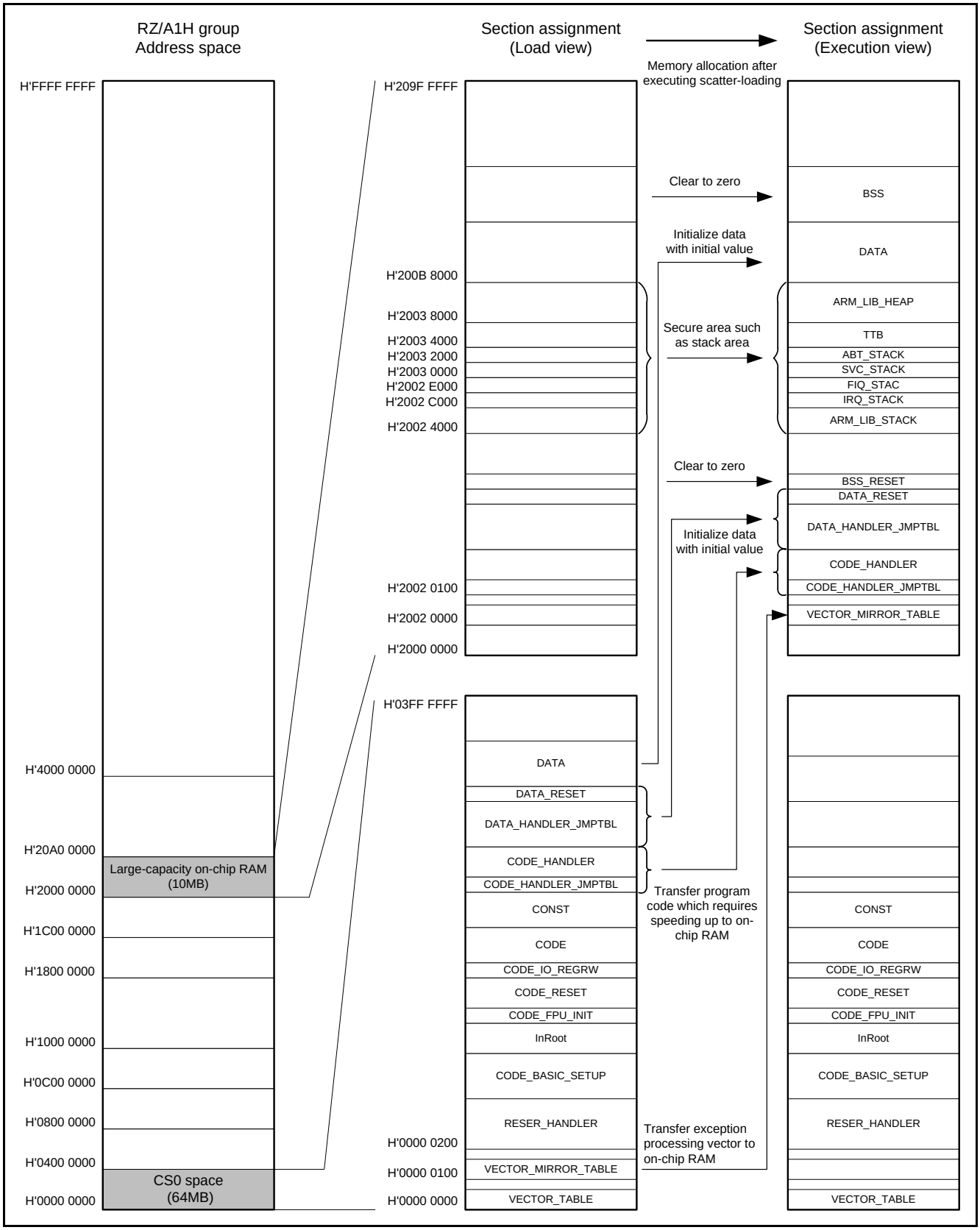


Figure 6.7 Section Assignment

6.3 Fixed-Width Integers

Table 6.6 lists the Fixed-Width Integers Used in Sample Code.

Table 6.6 Fixed-Width Integers Used in Sample Code

Symbol	Contents
char_t	8-bit character
bool_t	Boolean type, value: true (1) or false (0)
int_t	High-speed integer, signed 32-bit integer in this sample code
int8_t	8-bit integer, singed (Defined by standard library)
int16_t	16-bit integer, singed (Defined by standard library)
int32_t	32-bit integer, singed (Defined by standard library)
int64_t	64-bit integer, singed (Defined by standard library)
uint8_t	8-bit integer, unsigned (Defined by standard library)
uint16_t	16-bit integer, unsigned (Defined by standard library)
uint32_t	32-bit integer, unsigned (Defined by standard library)
uint64_t	64-bit integer, unsigned (Defined by standard library)
float32_t	32-bit floating point (Defined by standard library when specifying "__ARM_NEON__")
float64_t	64-bit floating point (Defined by standard library) (Defined by standard library when specifying "__ARM_NEON__")
float128_t	128-bit floating point

6.4 Constants

Table 6.7 lists the Constants Used in Sample Code.

Table 6.7 Constants Used in Sample Code

Constant Name	Setting Value	Contents
RTC_ENABLE	(1)	Enable setting and reading time counter (Second, minute, hour, day of week, day, month, and year)
RTC_DISABLE	(0)	Disable setting and reading time counter (Second, minute, hour, day of week, day, month, and year)
		Day of week definitions
RTC_WK_SUNDAY	(0)	Sunday
RTC_WK_MONDAY	(1)	Monday
RTC_WK_TUESDAY	(2)	Tuesday
RTC_WK_WEDNESDAY	(3)	Wednesday
RTC_WK_THURSDAY	(4)	Thursday
RTC_WK_FRIDAY	(5)	Friday
RTC_WK_SATURDAY	(6)	Saturday
RTC_WK_TOTAL	(7)	Total days of week
STB_GENERATE_ALARM_INT	(1)	The definition in the state where deep standby mode was canceled by RTC alarm interrupt
STB_NO_GENERATE_ALARM_INT	(0)	The definition in the state where deep standby mode was not canceled by RTC alarm interrupt (Startup from power-on reset also uses this definition)

6.5 Structure List

Figure 6.8 shows the Structure Used in Sample Code.

```
/* ==== Type declaration for time setting ==== */
/* ---- Type declaration for registers with 8-bit width ---- */
typedef struct rtc_8
{
   uint8_t value;    /* Setting values for time counter or alarm register   */
   uint8_t enable;   /* Specification for setting or acquisition            */
                     /* (RTC_DISABLE : Disable setting or acquisition,      */
                     /*  RTC_ENABLE  : Enable setting or acquisition)       */
} rtc_8_t;

/* ---- Type declaration for registers with 16-bit width
          (For year counter and year alarm registers) ---- */
typedef struct rtc_16
{
   uint16_t value;   /* Setting values for year counter or year alarm register */
   uint8_t  enable;  /* Specification for setting or acquisition             */
                     /* (RTC_DISABLE : Disable setting or acquisition,       */
                     /*  RTC_ENABLE  : Enable setting or acquisition)        */
} rtc_16_t;

/* ==== Structure declaration for time setting ==== */
typedef struct rtc_time
{                    /*             value       enable                      */
   rtc_8_t  second;  /* Second      (0 to 59)   (RTC_DISABLE or RTC_ENABLE) */
   rtc_8_t  minute;  /* Minute      (0 to 59)   (RTC_DISABLE or RTC_ENABLE) */
   rtc_8_t  hour;    /* Hour        (0 to 23)   (RTC_DISABLE or RTC_ENABLE) */
   rtc_8_t  week;    /* Day of week (0 to 6)    (RTC_DISABLE or RTC_ENABLE) */
   rtc_8_t  day;     /* Day         (1 to 31)   (RTC_DISABLE or RTC_ENABLE) */
   rtc_8_t  month;   /* Month       (1 to 12)   (RTC_DISABLE or RTC_ENABLE) */
   rtc_16_t year;    /* Year        (0 to 9999) (RTC_DISABLE or RTC_ENABLE) */
} rtc_time_t;

/* ==== Structure declaration for alarm-specified information (ENB bit) ==== */
typedef struct rtc_alarm_enb
{
   uint8_t second;   /* Second      (0 or 1) */
   uint8_t minute;   /* Minute      (0 or 1) */
   uint8_t hour;     /* Hour        (0 or 1) */
   uint8_t week;     /* Day of week (0 or 1) */
   uint8_t day;      /* Day         (0 or 1) */
   uint8_t month;    /* Month       (0 or 1) */
   uint8_t year;     /* Year        (0 or 1) */
} rtc_alarm_enb_t;
```

Figure 6.8 Structure Used in Sample Code

6.6 Functions

The sample code consists of the interface functions to control RTC (API functions), the user-defined functions which need to be prepared by the user for the system applications (functions called from the API functions), and the sample function required to operate the sample code.

Table 6.8, Table 6.9, and Table 6.10 list the Sample Functions, API Functions, and User-Defined Function.

Table 6.8 Sample Functions

Function Name	Outline
$Sub$$main	Initialization of on-chip peripheral functions (Call $Super$$main function, and branch to main function)
STB_CancelDeepStandby	Processing corresponding to deep standby mode cancel source
RTC_GetTimeCanDeepStb	Acquisition processing for deep standby mode cancel time
main	Main processing
RTC_DispTimeCanDeepStb	Processing for deep standby mode cancel time display
Sample_Main	Sample code main processing
Sample_RTC_Main	RTC sample code main processing
Sample_RTC_Init	RTC initial setting
Sample_RTC_SetTime	RTC time setting
Sample_RTC_GetTime	RTC time display
Sample_RTC_Start	Starting RTC time count operation
Sample_RTC_Stop	Stopping RTC time count operation
Sample_RTC_DeepStandby	RTC alarm time setting and transition to deep standby mode

Table 6.9 API Functions

Function Name	Outline
R_RTC_Init	RTC initial setting
R_RTC_Open	Starting RTC time count operation
R_RTC_Close	Stopping RTC time count operation
R_RTC_SetCnt	Setting values to RTC time counter
R_RTC_GetCnt	Obtaining values from RTC time counter
R_RTC_SetAlarm	Setting values to RTC alarm registers
R_RTC_GetAlarm	Obtaining values from RTC alarm registers

Table 6.10 User-Defined Functions

Function Name	Outline
Userdef_RTC_Init	RTC initial setting

6.7 Function Specifications

The following tables list the sample code function specifications.

$Sub$$main	
Outline	Initialization of on-chip peripheral functions
Declaration	void $Sub$$main(void)
Description	Executes initial setting for the peripheral functions, and jumps to the main function by calling the $Super$$main of the library function. In the sample code, the processing corresponding to the deep standby mode cancel source is executed by calling the sample function STB_CancelDeepStandby, and the initial settings for STB, PORT, INTC, and L1 cache are performed. The vector address is also set in the on-chip RAM area. The IRQ and the FIQ interrupts are enabled by calling the __enable_irq and __enable_fiq of the library function.
Arguments	None
Return Value	None

STB_CancelDeepStandby	
Outline	Processing corresponding to deep standby mode cancel source
Declaration	void STB_CancelDeepStandby(void)
Description	Determine if this function has been called after returning from deep standby mode with reference to the cancel source confirmation flag for the deep standby cancel source flag register (DSFR). If this function has been called after returning from deep standby mode, the processing corresponding to the deep standby mode cancel source should be executed to release the retention of pin state. In the sample code, if the deep standby mode cancel source is the alarm interrupt of the RTC, the RTCARF bit in the deep standby cancel source flag register (DSFR) is cleared to "0", and each counter value of the RTC is stored in the area retained in the large-capacity on-chip RAM as the time when the deep standby mode is cancelled. The stored time can be obtained from the sample function RTC_GetTimeCanDeepStb. The flag should be set to notify that this function has been called after returning from deep standby mode. Also, when the IOKEEP bit in the deep standby cancel source flag register (DSFR) is "1", the retention of the pin state can be released by setting the IOKEEP bit to "0".
Arguments	None
Return Value	None

RTC_GetTimeCanDeepStb

Outline	Acquisition processing for deep standby mode cancel time	
Declaration	uint32_t RTC_GetTimeCanDeepStb(rtc_time_t * time)	
Description	Stores the time of deep standby mode cancellation in the area specified by the argument time.	
Arguments	rtc_time_t * time	: Time of deep standby mode cancellation
	time->second.value	Second (0 to 59)
	time->minute.value	: Minute (0 to 59)
	time->hour.value	: Hour (0 to 23)
	time->week.value	: Day of week (0 to 6) 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
	time->day.value	: Day (1 to 31)
	time->month.value	: Month (1 to 12)
	time->year.value	: Year (0 to 9999)
Return Value	STB_GENERATE_ALARM_INT:	The state where deep standby mode was canceled by RTC alarm interrupt
	STB_NO_GENERATE_ALARM_INT:	The state where deep standby mode was not canceled by RTC alarm interrupt or the state started from power-on reset
Note	The sample function STB_CancelDeepStandby is used to store the time when the deep standby mode is cancelled. Therefore, this function should be called after the function STB_CancelDeepStandby is executed. If this function return STB_NO_GENERATE_ALARM_INT, time of deep standby mode cancellation is not stored to the argument time.	

main	
Outline	Main processing
Declaration	int_t main(void)
Description	Displays the sample code information on the terminal of the host PC which is connected to the GENMAI board by the serial interface. If this function is called after the deep standby mode is cancelled, displays the time when the deep standby mode is cancelled. Also, executes initial setting for the PORT connected with the LEDs on the board. Executes initial setting for the OSTM channel 0, and sets the timer counter so that the OSTM0 interrupt occurs every 500ms.
Arguments	None
Return Value	0

RTC_DispTimeCanDeepStb	
Outline	Processing for deep standby mode cancel time display
Declaration	void RTC_DispTimeCanDeepStb(void)
Description	This is a sample function to obtain the time when the deep standby mode is cancelled. In the sample code, the value of the RTC time counter is obtained by using the sample function RTC_GetTimeCanDeepStb. If the time when the deep standby mode is cancelled is stored, the value of the BCD-coded time counter is converted into integer value and display to the terminal.
Arguments	None
Return Value	None
Note	The RTC time counter values (BCD) are stored as the time when deep standby mode is cancelled by using the sample function STB_CancelDeepStandby.

Sample_Main	
Outline	Sample code main processing
Declaration	void Sample_Main(void)
Description	Waits for character input from the terminal running on the host PC connected to the GENMAI board via the serial interface. Activates the RTC sample code when "RTC" + "Enter" key is input.
Arguments	None
Return Value	None

Sample_RTC_Main	
Outline	RTC sample code main processing
Declaration	int32_t Sample_RTC_Main(int32_t argc, char_t ** argv)
Description	Waits for character input from the terminal on the host PC connected to GENMAI board by the serial interface. Runs each sample according to the input characters input. When "1"+"Enter" keys are input, execute RTC initial settings. When "2"+"Enter" keys are input, execute RTC time setting. When "3"+"Enter" keys are input, execute RTC time display. When "4"+"Enter" keys are input, execute starting RTC time count operation. When "5"+"Enter" keys are input, execute stopping RTC time count operation. When "6"+"Enter" keys are input, execute RTC alarm time setting and transition to deep standby mode.
Arguments	int32_t argc : The number of command arguments input from the terminal char_t **argv : Pointer to the command input from the terminal
Return Value	COMMAND_EXIT : Termination of RTC sample code processing

Sample_RTC_Init	
Outline	RTC initial setting
Declaration	int32_t Sample_RTC_Init(int32_t argc, char_t ** argv)
Description	This is a sample function to initialize RTC. Called when "1"+"Enter" keys are input during the wait processing for character input using the sample function Sample_RTC_Main. In the sample code, RTC is initialized by using the API function R_RTC_Init.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing

Sample_RTC_SetTime	
Outline	RTC time setting
Declaration	int32_t Sample_RTC_SetTime(int32_t argc, char_t ** argv)
Description	This is a sample function to set the time to the RTC time counter. Called when "2"+"Enter" keys are input during the wait processing for character input using the sample function Sample_RTC_Main. In the sample code, the time (decimal digit) input from the terminal is set to the RTC time counter using the API function R_RTC_SetCnt. The items for the time should be input in order day of week, month, day, year, hour, minute, and second. Inputs the values of the specified range (Refer to Table 6.2 for effective range for input values). If a value outside the specified range is input, waits until appropriate value is input. When "-1" is input, the item should not be set to the RTC time counter.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing
Note	This sample function runs on the condition that initial settings have been made by using the sample function Sample_RTC_Init.

Sample_RTC_GetTime	
Outline	RTC time display
Declaration	int32_t Sample_RTC_GetTime(int32_t argc, char_t ** argv)
Description	This is a sample function to obtain time from the RTC time counter. Called when "3"+"Enter" keys are input during wait processing for character input using the sample function Sample_RTC_Main. In the sample code, this function displays the time obtained from the RTC time counter to the terminal using the API function R_RTC_GetCnt.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing
Note	This sample function runs on the condition that initial settings have been made by using the sample function Sample_RTC_Init.

Sample_RTC_Start	
Outline	Starting RTC time count operation
Declaration	int32_t Sample_RTC_Start(int32_t argc, char_t ** argv)
Description	This is a sample function to start the RTC time count operation. Called when "4"+"Enter" keys are input during wait processing for character input using the sample function Sample_RTC_Main. In the sample code, this function displays the time obtained from the RTC time counter to the terminal and starts the RTC time count operation using the API function R_RTC_Open.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing
Note	This sample function runs on the condition that initial settings have been made by using the sample function Sample_RTC_Init.

Sample_RTC_Stop	
Outline	Stopping RTC time count operation
Declaration	int32_t Sample_RTC_Stop(int32_t argc, char_t ** argv)
Description	This is a sample function to stop the RTC time count operation. Called when "5"+"Enter" keys are input during wait processing for character input using the sample function Sample_RTC_Main. In the sample code, this function stops the RTC time count operation using the API function R_RTC_Close, and displays the time obtained from the RTC time counter to the terminal.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing
Note	This sample function runs on the condition that initial settings have been made by using the sample function Sample_RTC_Init.

Sample_RTC_DeepStandby	
Outline	RTC alarm time setting and transition to deep standby mode
Declaration	int32_t Sample_RTC_DeepStandby(int32_t argc, char_t ** argv)
Description	This is a sample processing to transit to deep standby mode after selecting alarm interrupt as deep standby mode cancel source. Called when "6"+"Enter" keys are input during wait processing for character input using the sample function Sample_RTC_Main. In the sample code, the time values input from the terminal are specified for the hour alarm register (RHRAR) and the minute alarm register (RMINAR), and "0" for the second alarm register (RSECAR) to generate alarm interrupt at the time (hour and minute) input from the terminal. Then the ENB bits of each register are set. Reads the time from the RTC time counter and displays it to the terminal as the time to transit to deep standby mode. Sets the STB so that the cancel deep standby mode is cancelled when the RTC alarm interrupt is generated, and transits to deep standby mode. The STB settings are made to activation through the external memory (NOR flash memory connected to the CS0 space) after the deep standby mode is cancelled. When both the time (Hour, Minute) input from the terminal is "-1", or when reading time from the RTC time counter and an error occurs, COMMAND_ERROR is returned by this function.
Arguments	int32_t argc : The number of command arguments input from the terminal. Not used in this function. char_t **argv : Pointer to the command input from the terminal. Not used in this function.
Return Value	COMMAND_SUCCESS : Success of RTC sample code processing COMMAND_ERROR : Failure of RTC sample code processing
Note	This sample function performs initial setting by using the sample function Sample_RTC_Init and runs on the condition that RTC is in the time count operation.

R_RTC_Init	
Outline	RTC initial setting
Declaration	void R_RTC_Init(void)
Description	Initializes the RTC. Calls the user-defined function Userdef_RTC_Init and initializes the RTC by Userdef_RTC_Init.
Arguments	None
Return Value	None

R_RTC_Open	
Outline	Starting RTC time count operation
Declaration	void R_RTC_Open(void)
Description	Starts the RTC time count operation. RTC starts the time count according to the current time of time counter.
Arguments	None
Return Value	None
Note	Set available time to RTC using the API function R_RTC_SetCnt before calling this API function.

R_RTC_Close	
Outline	Stopping RTC time count operation
Declaration	void R_RTC_Close(void)
Description	Stops the RTC time count operation.
Arguments	None
Return Value	None

R_RTC_SetCnt

Outline	Setting values to RTC time counter
Declaration	int32_t R_RTC_SetCnt(rtc_time_t * time)
Description	Sets the time specified by the argument time to RTC time counter. Sets the RESET bit in the control register 2 (RCR2). If RTC_ENABLE is specified for the argument time->second.enable, performs BCD-coding for the time of the argument time->second.value and writes it to the RTC second counter (RSECCNT). If RTC_DISABLE is specified for the argument time->second.enable, writing is not performed. Otherwise, any other member of the time should also be written to the respective RTC time counters. Because writing to the time counter is disabled while the RTC is in time count operation, suspends the time count operation at the beginning of this function and restarts at the end of this function.
Arguments	rtc_time_t * time : Time time->second.value : Second (0 to 59) time->minute.value : Minute (0 to 59) time->hour.value : Hour (0 to 23) time->week.value : Day of week (0 to 6) 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday time->day.value : Day (1 to 31) time->month.value : Month (1 to 12) time->year.value : Year (0 to 9999) Specification for setting object (RTC_ENABLE: Do set, RTC_DISABLE: Do NOT set) time->second.enable : Second counter setting time->minute.enable : Minute counter setting time->hour.enable : Hour counter setting time->week.enable : Day of week counter setting time->day.enable : Day counter setting time->month.enable : Month counter setting time->year.enable : Year counter setting
Return Value	DEVDRV_SUCCESS : Success in setting value to RTC time counter DEVDRV_ERROR : Failure in setting value to RTC time counter
Note	Call the API function R_RTC_Close before calling this API function to stop the time count operation.

R_RTC_GetCnt

Outline	Obtaining values from RTC time counter	
Declaration	int32_t R_RTC_GetCnt(rtc_time_t * time)	
Description	Obtains the time from the RTC time counter and stores it in the area specified by the argument time. If RTC_ENABLE is specified for the argument time->second.enable, converts the BCD-coded time read from the RTC second counter (RSECCNT) into integer value and stores it in the argument time->second.value. If RTC_DISABLE is specified for the argument time->second.enable, readout is not performed. Otherwise, any other member of the time should also be readout from the respective RTC time counters. During the readout processing from the RTC time counter, the carry flag (CF) of the control register 1 (RCR1) is cleared to "0", the count value from the time counter is read, and the carry flag is verified. If the carry flag is set when reading out from the time counter, the readout is determined to be invalid and the readout processing from the time counter is re-executed. If the carry flag is not set even after the readout processing has been executed twice, this function returns DEVDRV_ERROR.	
Arguments	rtc_time_t * time	: Storage area for obtained time
		time->second.value : Second (0 to 59)
		time->minute.value : Minute (0 to 59)
		time->hour.value : Hour (0 to 23)
		time->week.value : Day of week (0 to 6) 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
		time->day.value : Day (1 to 31)
		time->month.value : Month (1 to 12)
		time->year.value : Year (0 to 9999)
		Specification for obtaining object (RTC_ENABLE: Do obtain, RTC_DISABLE: Do NOT obtain)
		time->second.enable : Second counter obtaining
		time->minute.enable : Minute counter obtaining
		time->hour.enable : Hour counter obtaining
		time->week.enable : Day of week counter obtaining
		time->day.enable : Day counter obtaining
		time->month.enable : Month counter obtaining
		time->year.enable : Year counter obtaining
Return Value	DEVDRV_SUCCESS	: Success in obtaining value from RTC time counter
	DEVDRV_ERROR	: Failure in obtaining value from RTC time counter

R_RTC_SetAlarm

Outline	Setting values to RTC alarm registers
Declaration	int32_t R_RTC_SetAlarm(rtc_time_t * time, rtc_alarm_enb_t * alarm_enb)
Description	Sets the alarm time specified by the argument time to the RTC alarm register. Also, specifies the time (second, minute, hour, day of week, day, month, and year) for alarm to become active using member of the argument alarm_enb. If this function is called in the state of the alarm interrupt enable, disables the alarm interrupt. If RTC_ENABLE is specified for the argument time->second.enable, performs BCD-coding for the alarm time of the argument time->second.value and writes it to the RTC second alarm register (RSECAR). If RTC_DISABLE is specified for the argument time->second.enable, writing is not performed. Otherwise, any other member of the alarm timer should also be written to the respective RTC alarm registers. Writes the value of the argument alarm_enb->second to the ENB bit in the RTC second alarm register (RSECAR). Other member of the setting information of activating of the alarm time is also written to the ENB bits in the RTC alarm registers. Clears the alarm flag (AF bit in the Control Register 1 (RCR1)) to "0". If this function is called in the state of the alarm interrupt enable, enables the alarm interrupt. When the time of the alarm register in which "1" has been set to the ENB bit matches the time counter, "1" is set to the alarm flag. That means the alarm flag informs that the current time matches the alarm time.
Arguments	rtc_time_t * time : Alarm time time->second.value : Second (0 to 59) time->minute.value : Minute (0 to 59) time->hour.value : Hour (0 to 23) time->week.value : Day of week (0 to 6) 0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday time->day.value : Day (1 to 31) time->month.value : Month (1 to 12) time->year.value : Year (0 to 9999) Specification for setting object (RTC_ENABLE: Do set, RTC_DISABLE: Do NOT set) time->second.enable : Second alarm setting time->minute.enable : Minute alarm setting time->hour.enable : Hour alarm setting time->week.enable : Day of week alarm setting time->day.enable : Day alarm setting time->month.enable : Month alarm setting time->year.enable : Year alarm setting rtc_alarm_enb_t * alarm_enb : Setting information of activating of the alarm time (0: Activate alarm time, 1: Deactivate alarm time) alarm_enb->second : Second (0 or 1) alarm_enb->minute : Minute (0 or 1) alarm_enb->hour : Hour (0 or 1) alarm_enb->week : Day of week (0 or 1) alarm_enb->day : Day (0 or 1) alarm_enb->month : Month (0 or 1) alarm_enb->year : Year (0 or 1)
Return Value	DEVDRV_SUCCESS : Success in setting value to RTC alarm register DEVDRV_ERROR : Failure in setting value to RTC alarm register

R_RTC_GetAlarm

Outline	Obtaining values from RTC alarm registers	
Declaration	int32_t R_RTC_GetAlarm(rtc_time_t * time, rtc_alarm_enb_t * alarm_enb)	
Description	Obtains the alarm time from the RTC alarm register and stores it in the area specified by the argument time. If RTC_ENABLE is specified for the argument time->second.enable, converts the BCD-coded alarm time read from the RTC second alarm register (RSECAR) into integer value and stores it in the argument time->second.value. If RTC_DISABLE is specified for the argument time->second.enable, any readout is not performed. Any other member of the alarm time is read out from the respective RTC alarm registers. Reads the setting information of activating of the alarm time from the ENB bit in the second alarm register (RSECAR) and stores it in the argument alarm_enb->second. Other member of the setting information of activating of the alarm time is also read from the ENB bits in the RTC alarm registers.	
Arguments	rtc_time_t * time	: Storage area for obtained alarm time
		time->second.value : Second (0 to 59)
		time->minute.value : Minute (0 to 59)
		time->hour.value : Hour (0 to 23)
		time->week.value : Day of week (0 to 6) 0: Sunday 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday
		time->day.value : Day (1 to 31)
		time->month.value : Month (1 to 12)
		time->year.value : Year (0 to 9999)
		Specification for obtaining object (RTC_ENABLE: Do obtain, RTC_DISABLE: Do NOT obtain)
		time->second.enable : Second alarm obtaining
		time->minute.enable : Minute alarm obtaining
		time->hour.enable : Hour alarm obtaining
		time->week.enable : Day of week alarm obtaining
		time->day.enable : Day alarm obtaining
		time->month.enable : Month alarm obtaining
		time->year.enable : Year alarm obtaining
	rtc_alarm_enb_t * alarm_enb	: Storage area for obtained setting information of activating of the alarm time (0: Activate alarm time, 1: Deactivate alarm time)
		alarm_enb->second : Second (0 or 1)
		alarm_enb->minute : Minute (0 or 1)
		alarm_enb->hour : Hour (0 or 1)
		alarm_enb->week : Day of week (0 or 1)
		alarm_enb->day : Day (0 or 1)
		alarm_enb->month : Month (0 or 1)
		alarm_enb->year : Year (0 or 1)
Return Value	DEVDRV_SUCCESS	: Success in obtaining value from RTC alarm register
	DEVDRV_ERROR	: Failure in obtaining value from RTC alarm register

Userdef_RTC_Init	
Outline	RTC initial setting
Declaration	void Userdef_RTC_Init(void)
Description	This is a user-defined function. RTC should be initialized. In the sample code, this function stops the time count operation and disables the carry interrupt, alarm interrupt, and periodic interrupt after the RTC module standby has been cancelled. Selects 32.768kHz from RTC_X1 as an operation clock and sets RTC to operate the on-chip crystal oscillator. The RESET bit in the RTC control register 2 (RCR2) is also set.
Arguments	None
Return Value	None

6.8 Flowcharts

6.8.1 Initialization of Peripheral Functions ($Sub$$main Function)

Figure 6.9 and Figure 6.10 show flowchart of Initialization of Peripheral Functions ($Sub$$main Function).

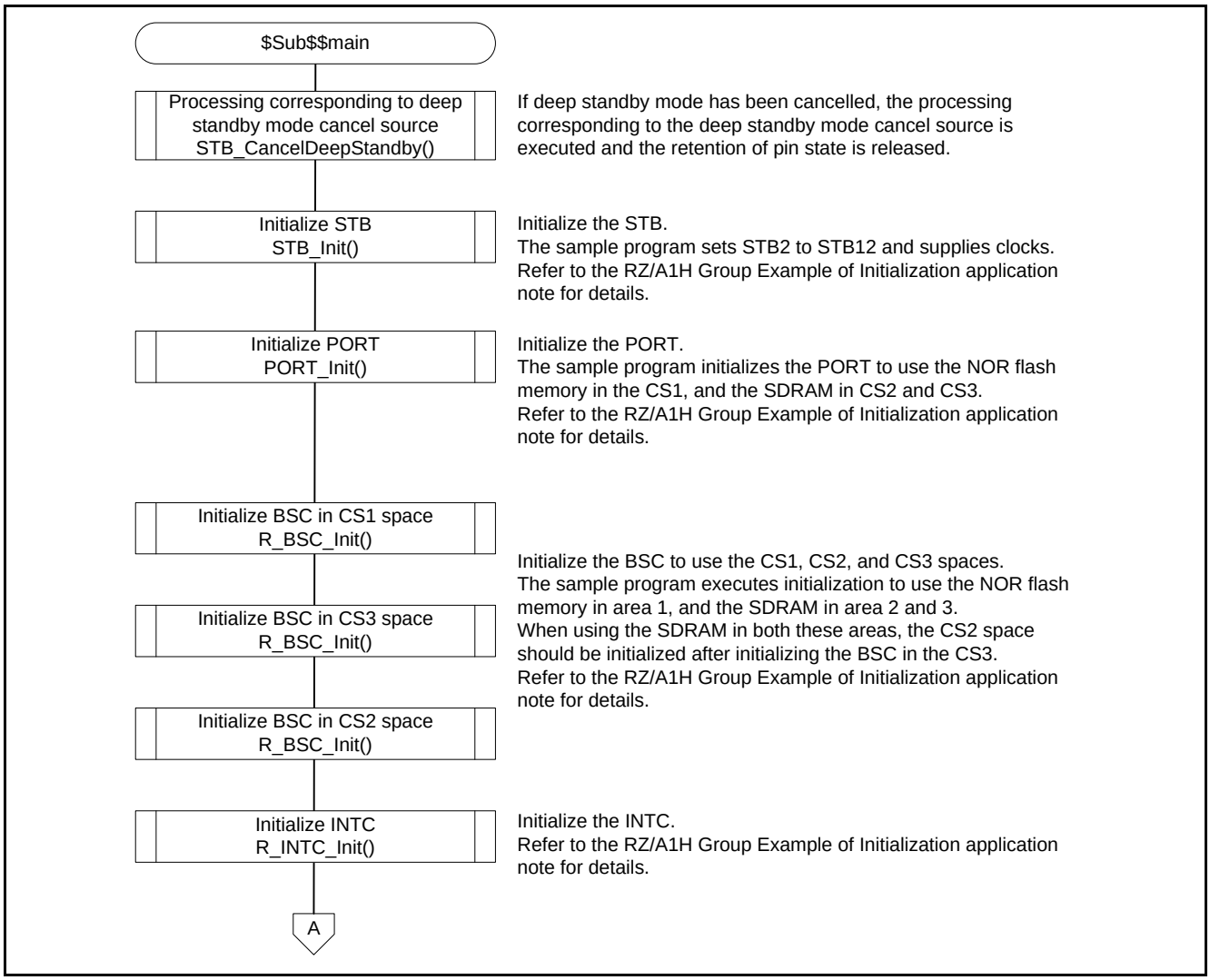


Figure 6.9 Initialization of Peripheral Functions ($Sub$$main Function) (1/2)

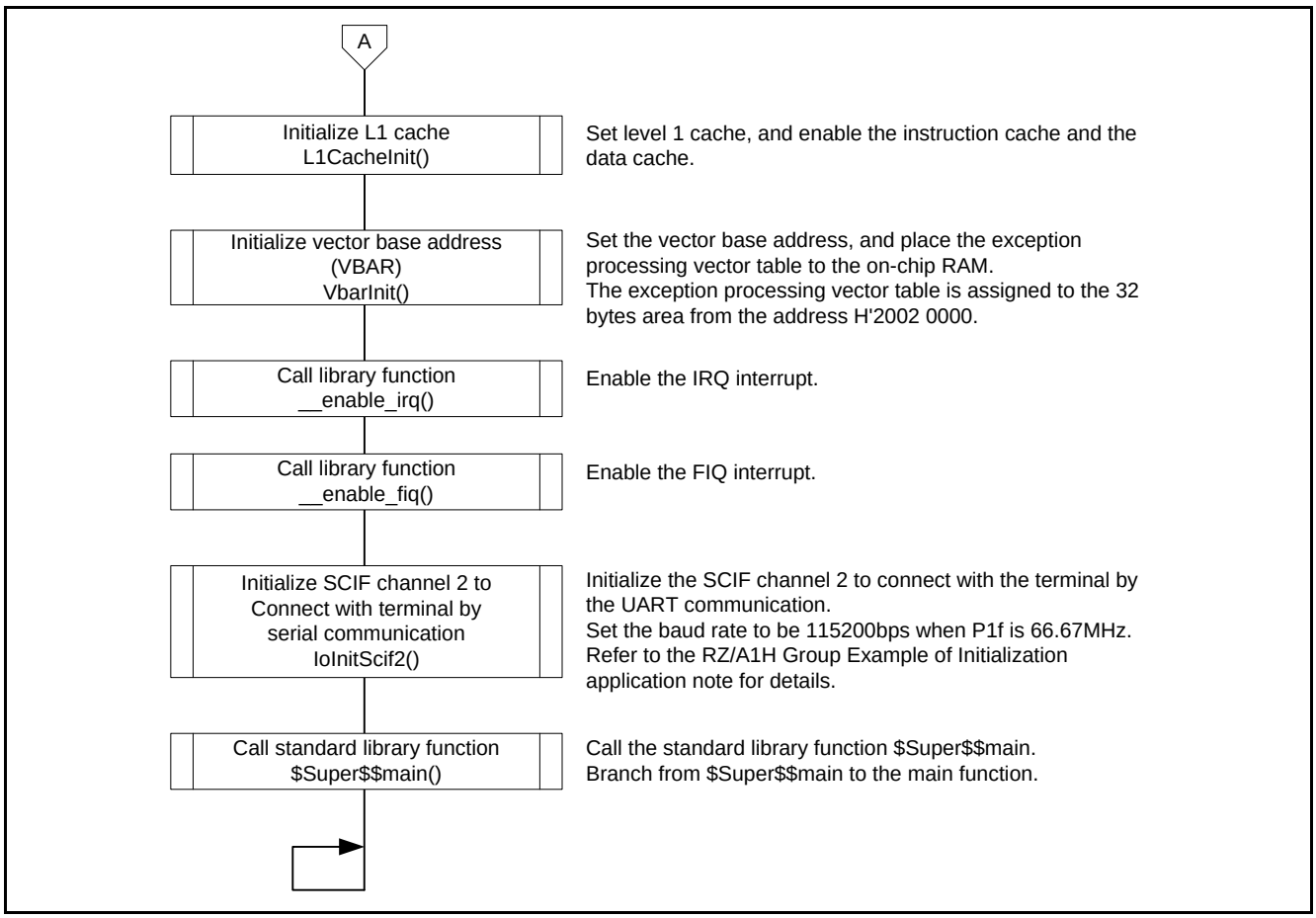


Figure 6.10 Initialization of Peripheral Functions ($Sub$$main function) (2/2)

6.8.2 Processing Corresponding to Deep Standby Mode Cancel Source

Figure 6.11 shows the flowchart of Processing Corresponding to Deep Standby Mode Cancel Source.

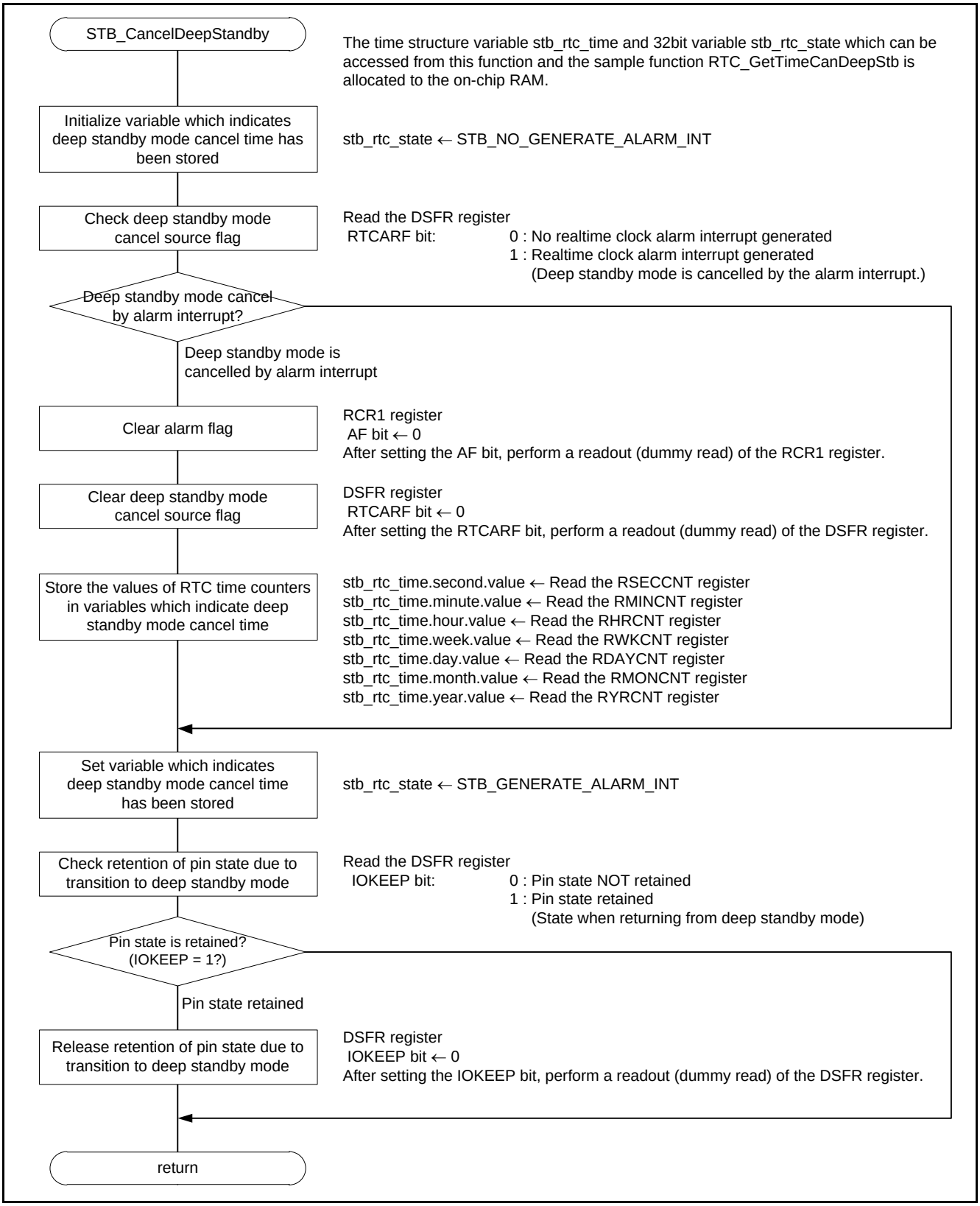


Figure 6.11 Processing Corresponding to Deep Standby Mode Cancel Source

6.8.3 Acquisition Processing for Deep Standby Mode Cancel Time

Figure 6.12 shows the flowchart of Acquisition Processing for Deep Standby Mode Cancel Time.

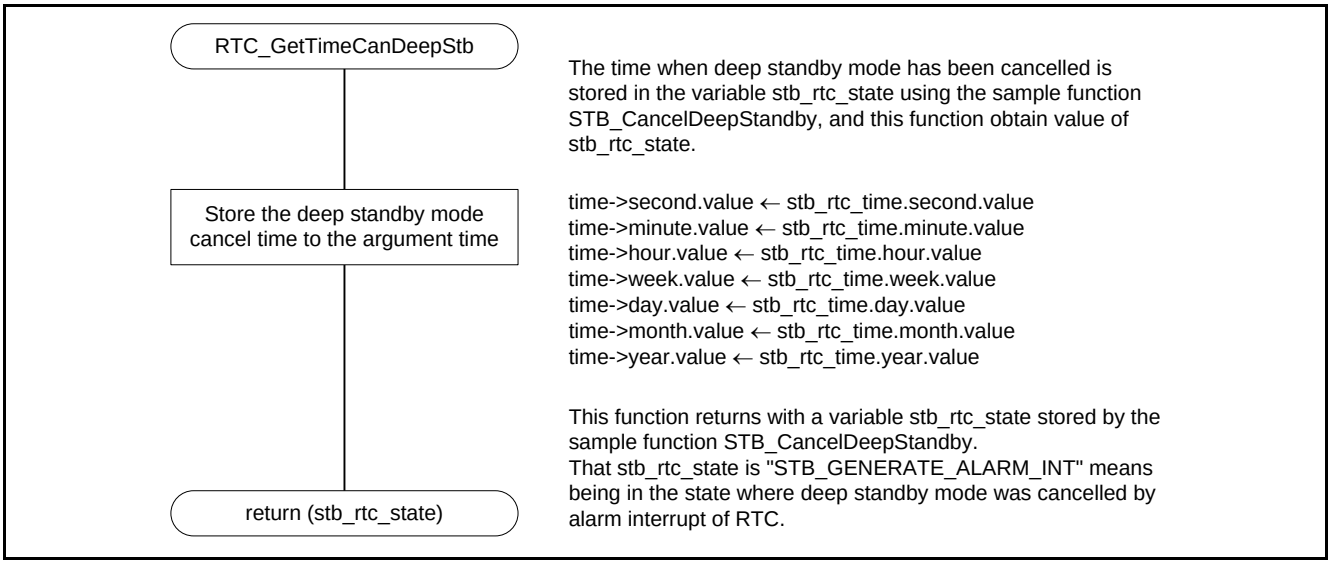


Figure 6.12 Acquisition Processing for Deep Standby Mode Cancel Time

6.8.4 Main Processing

Figure 6.13 shows the flowchart of Main Processing.

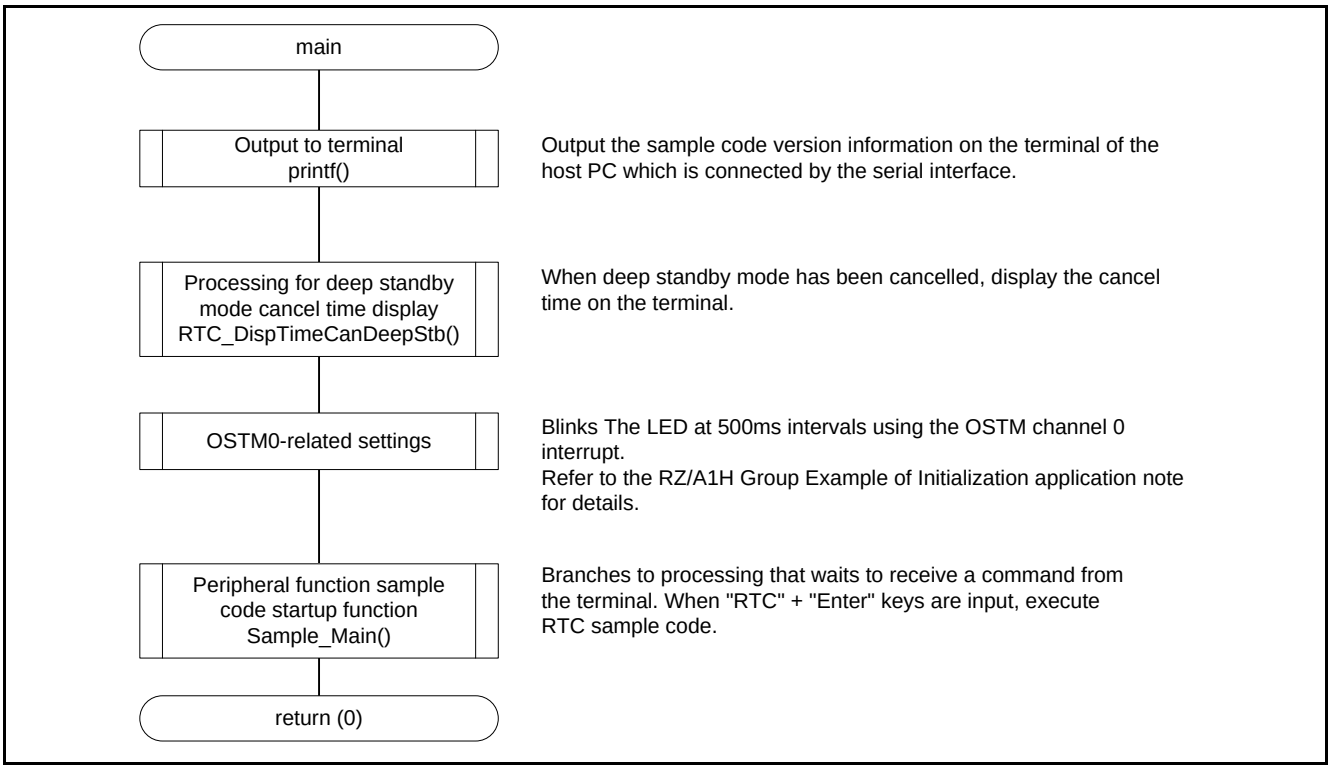


Figure 6.13 Main Processing

6.8.5 Processing for Deep Standby Mode Cancel Time Display

Figure 6.14 shows the flowchart of Processing for Deep Standby Mode Cancel Time Display.

The time when deep standby mode has been cancelled is obtained using the sample function RTC_GetTimeCanDeepStb, the BCD-coded time counter values are converted into integer values and then displayed on the terminal.

By using the sample function STB_CancelDeepStandby, the value of the RTC time counter (BCD) are stored as the time when deep standby mode has been cancelled.

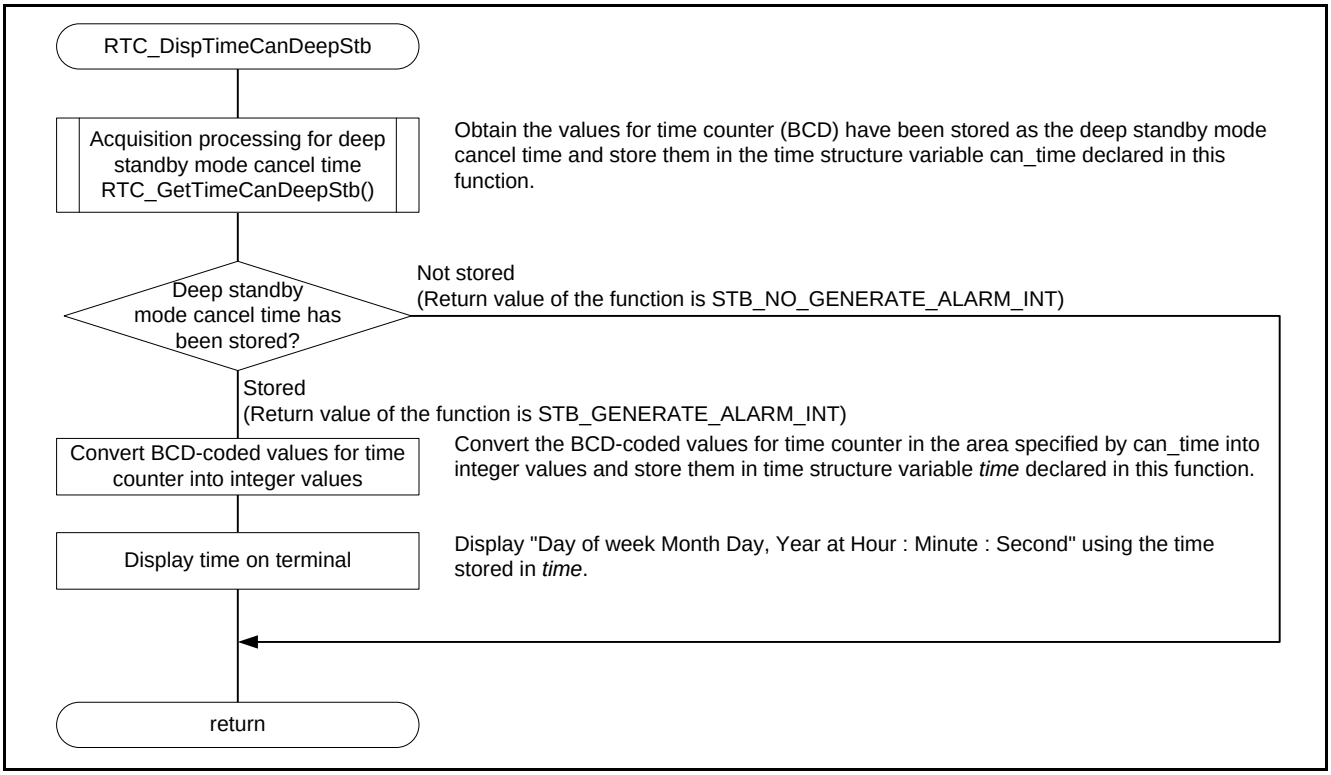


Figure 6.14 Processing for Deep Standby Mode Cancel Time Display

6.8.6 Sample Code Main Processing

Figure 6.15 shows the flowchart of Sample Code Main Processing. This function waits for the character input from the terminal running on the host PC.

The RTC sample code is executed when "RTC" + "Enter" keys is input.

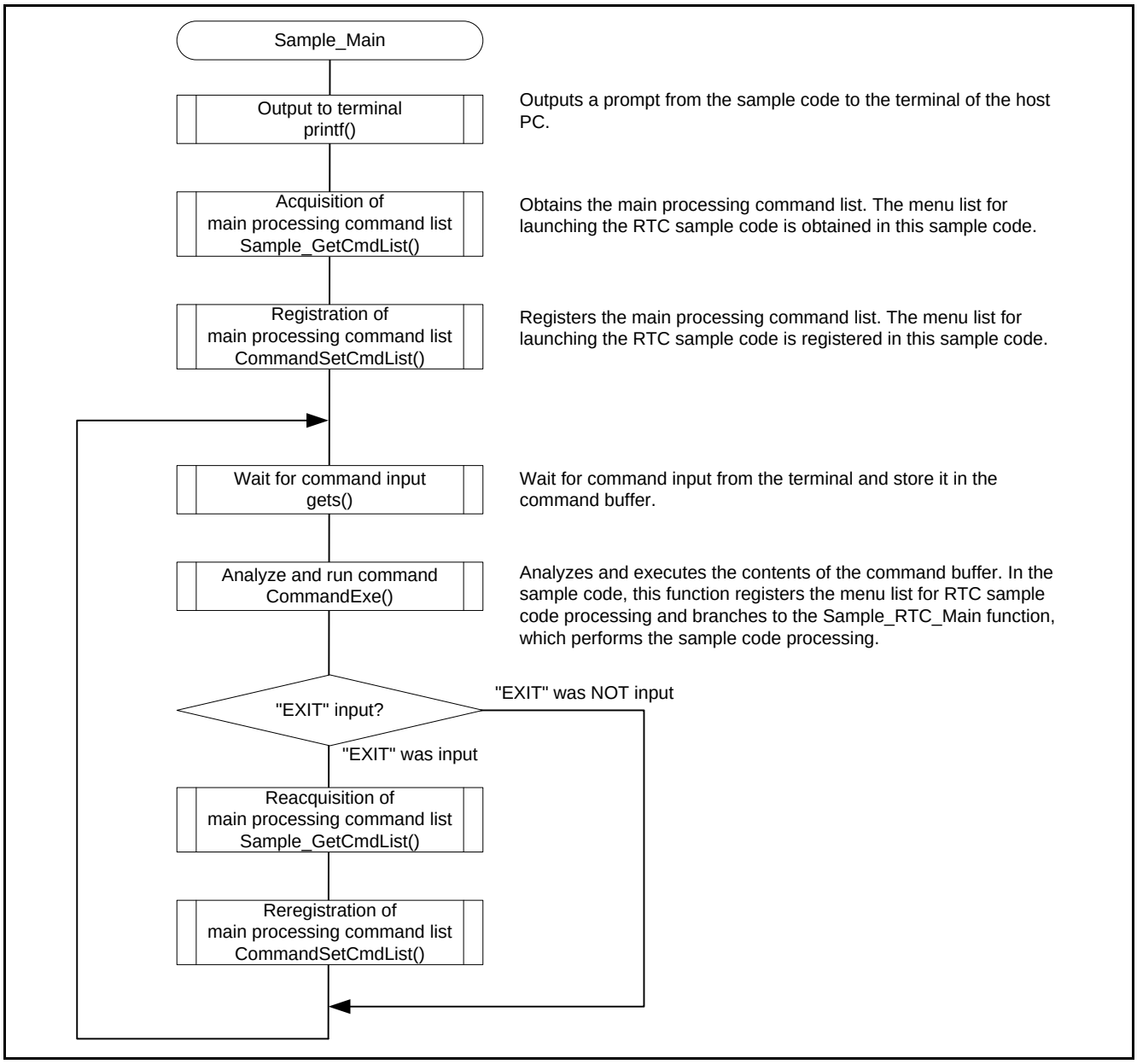


Figure 6.15 Sample Code Main Processing

6.8.7 RTC Sample Code Main Processing

Figure 6.16 shows the flowchart of RTC Sample Code Main Processing. This function waits for the character input from the terminal running on the host PC and branches to the RTC sample code processing according to the input command.

When "1" + "Enter" key is input, execute RTC initial setting.

When "2" + "Enter" key is input, execute RTC time setting.

When "3" + "Enter" key is input, execute RTC time display.

When "4" + "Enter" key is input, execute starting RTC time count operation.

When "5" + "Enter" key is input, execute stopping RTC time count operation.

When "6" + "Enter" key is input, execute RTC alarm time setting and transition to deep standby mode.

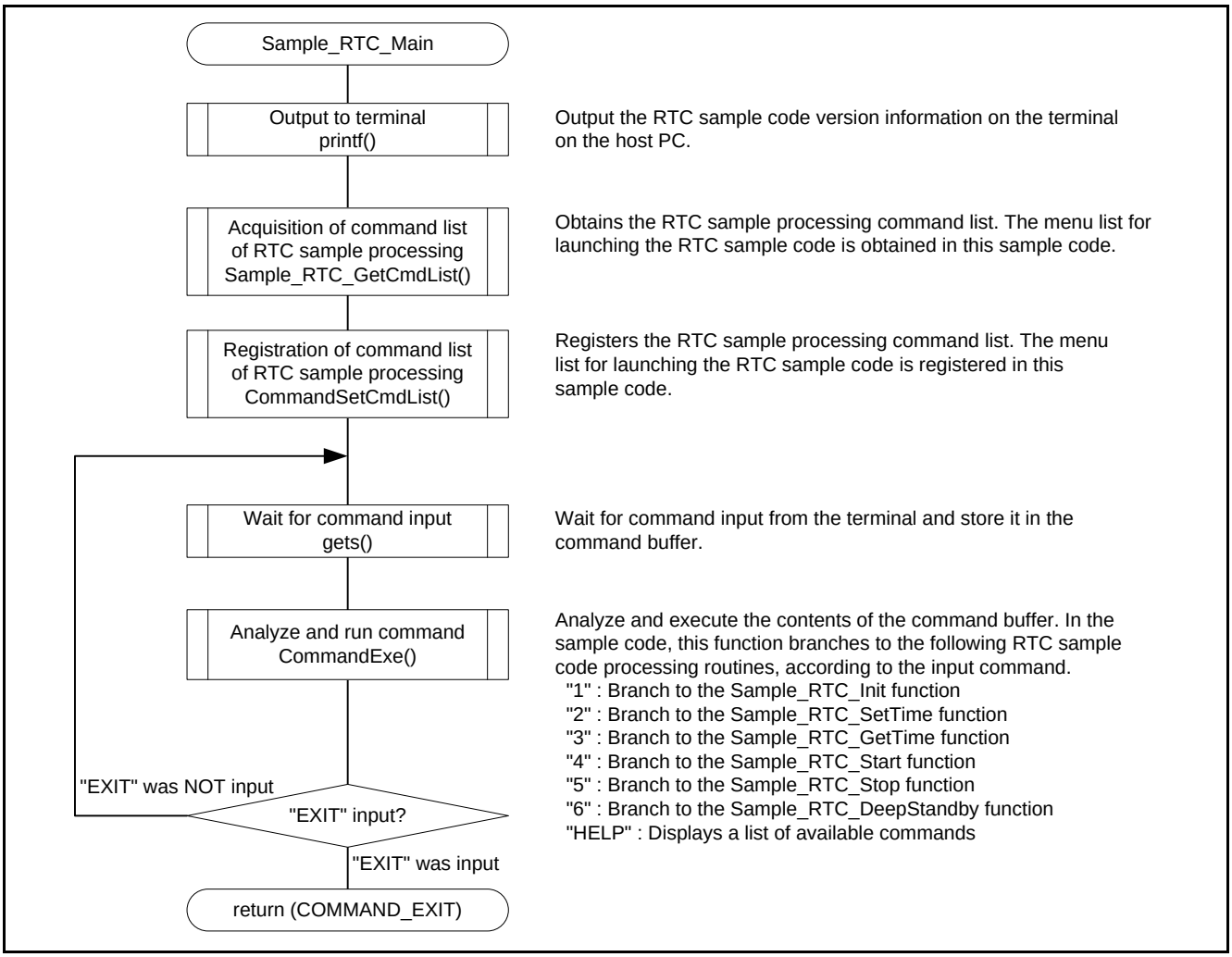


Figure 6.16 RTC Sample Code Main Processing

6.8.8 RTC Initial Setting

Figure 6.17 shows the flowchart of RTC Initial Setting.

This function runs when Command 1 is input during the RTC command wait processing of the sample function Sample_RTC_Main.

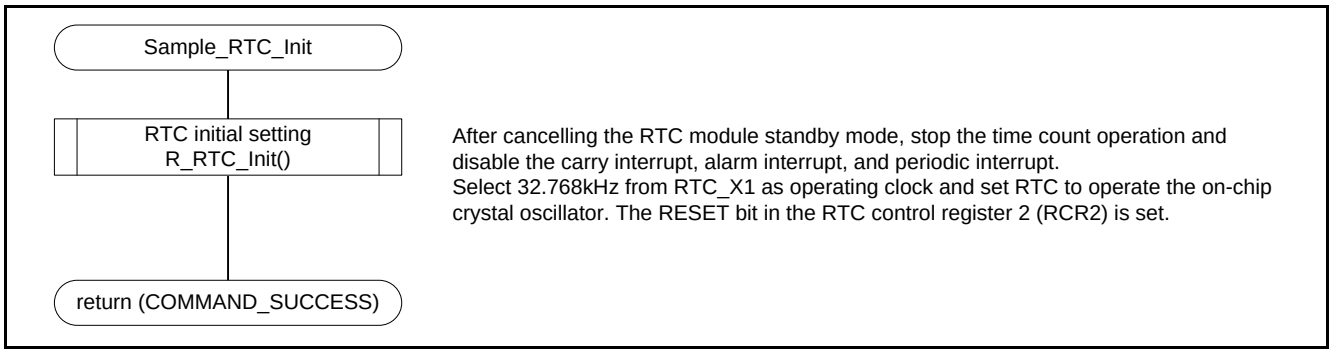
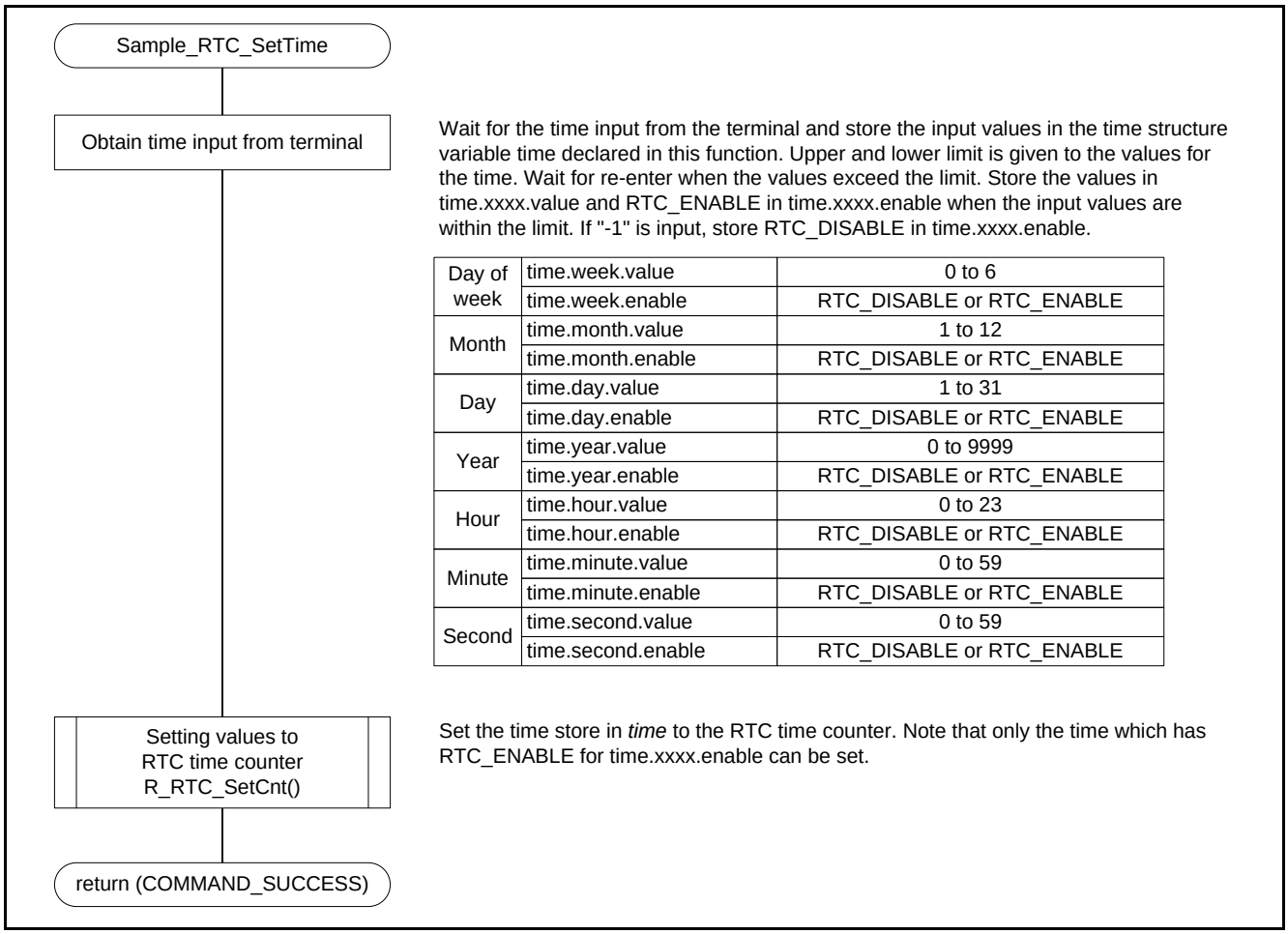


Figure 6.17 RTC Initial Setting

6.8.9 RTC Time Setting

Figure 6.18 shows the flowchart of RTC Time Setting.

This function runs when Command 2 is input during the RTC command wait processing of the sample function Sample_RTC_Main.



Day of week	time.week.value	0 to 6
	time.week.enable	RTC_DISABLE or RTC_ENABLE
Month	time.month.value	1 to 12
	time.month.enable	RTC_DISABLE or RTC_ENABLE
Day	time.day.value	1 to 31
	time.day.enable	RTC_DISABLE or RTC_ENABLE
Year	time.year.value	0 to 9999
	time.year.enable	RTC_DISABLE or RTC_ENABLE
Hour	time.hour.value	0 to 23
	time.hour.enable	RTC_DISABLE or RTC_ENABLE
Minute	time.minute.value	0 to 59
	time.minute.enable	RTC_DISABLE or RTC_ENABLE
Second	time.second.value	0 to 59
	time.second.enable	RTC_DISABLE or RTC_ENABLE

Figure 6.18 RTC Time Setting

6.8.10 RTC Time Display

Figure 6.19 shows the flowchart of RTC Time Display.

This function runs when Command 3 is input during the RTC command wait processing of the sample function Sample_RTC_Main.

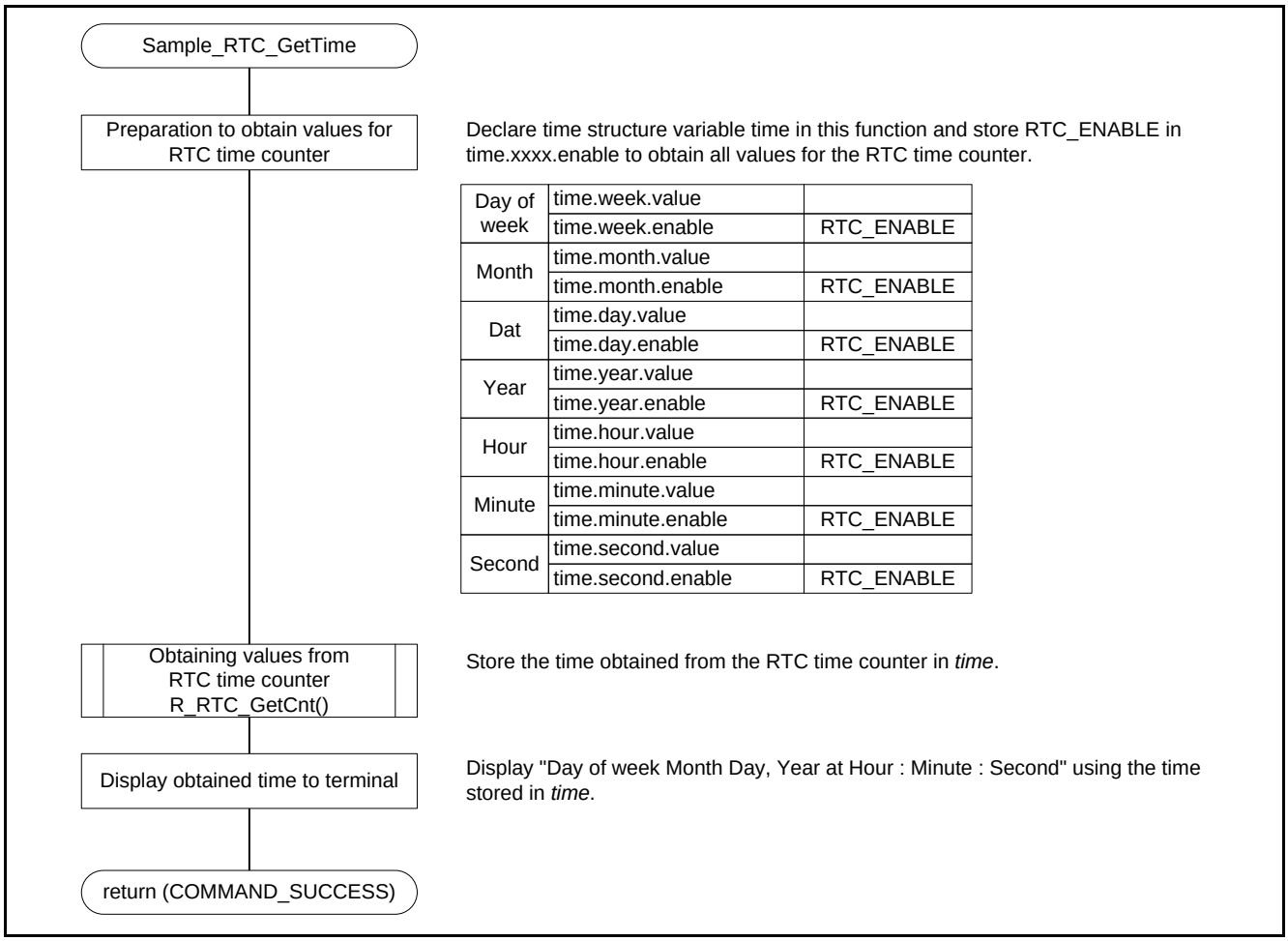


Figure 6.19 RTC Time Display

6.8.11 Starting RTC Time Count Operation

Figure 6.20 shows the flowchart of Starting RTC Time Count Operation.

This function runs when Command 4 is input during the RTC command wait processing of the sample function Sample_RTC_Main.

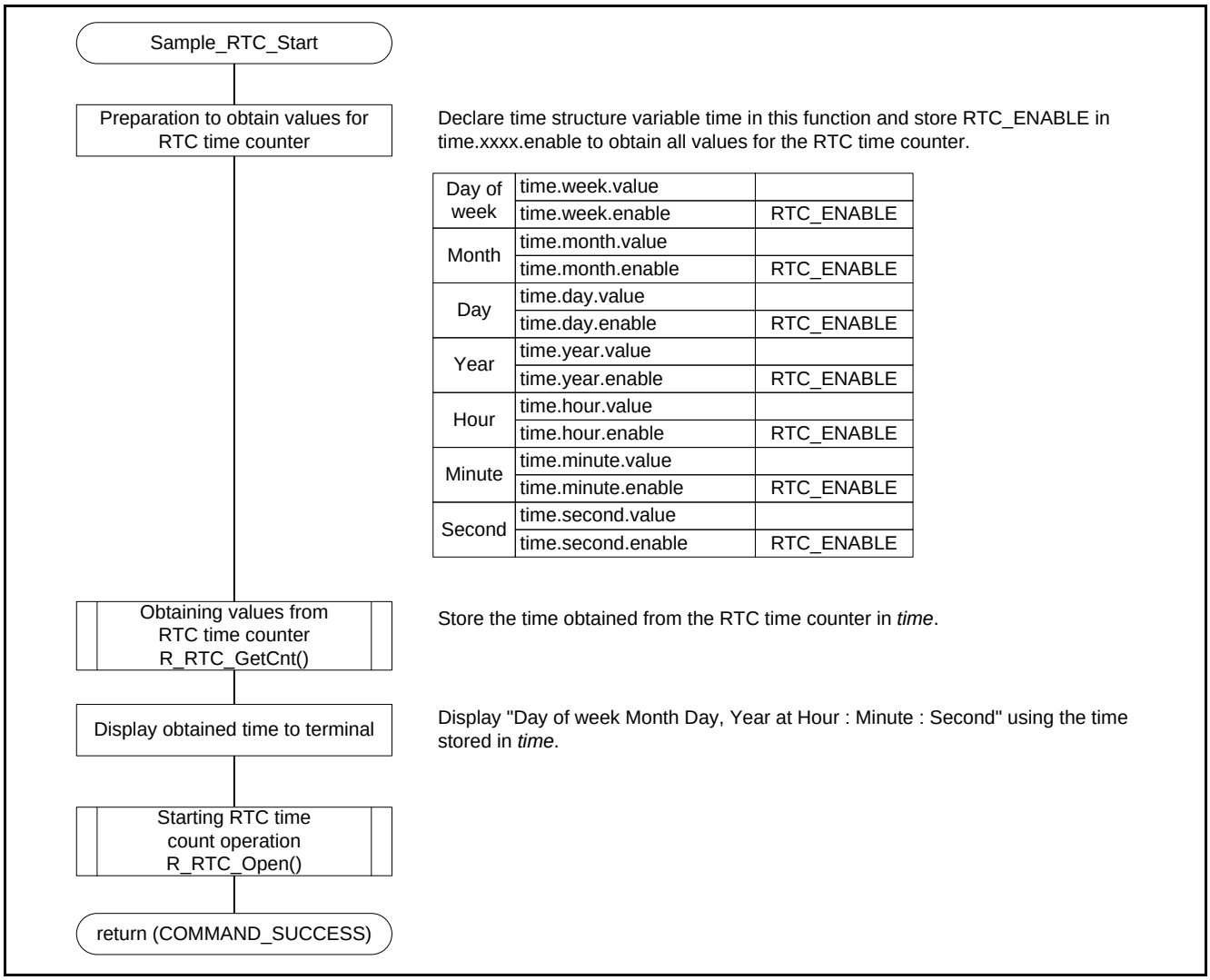


Figure 6.20 Starting RTC Time Count Operation

6.8.12 Stopping RTC Time Count Operation

Figure 6.21 shows the flowchart of Stopping RTC Time Count Operation.

This function runs when Command 5 is input during the RTC command wait processing of the sample function Sample_RTC_Main.

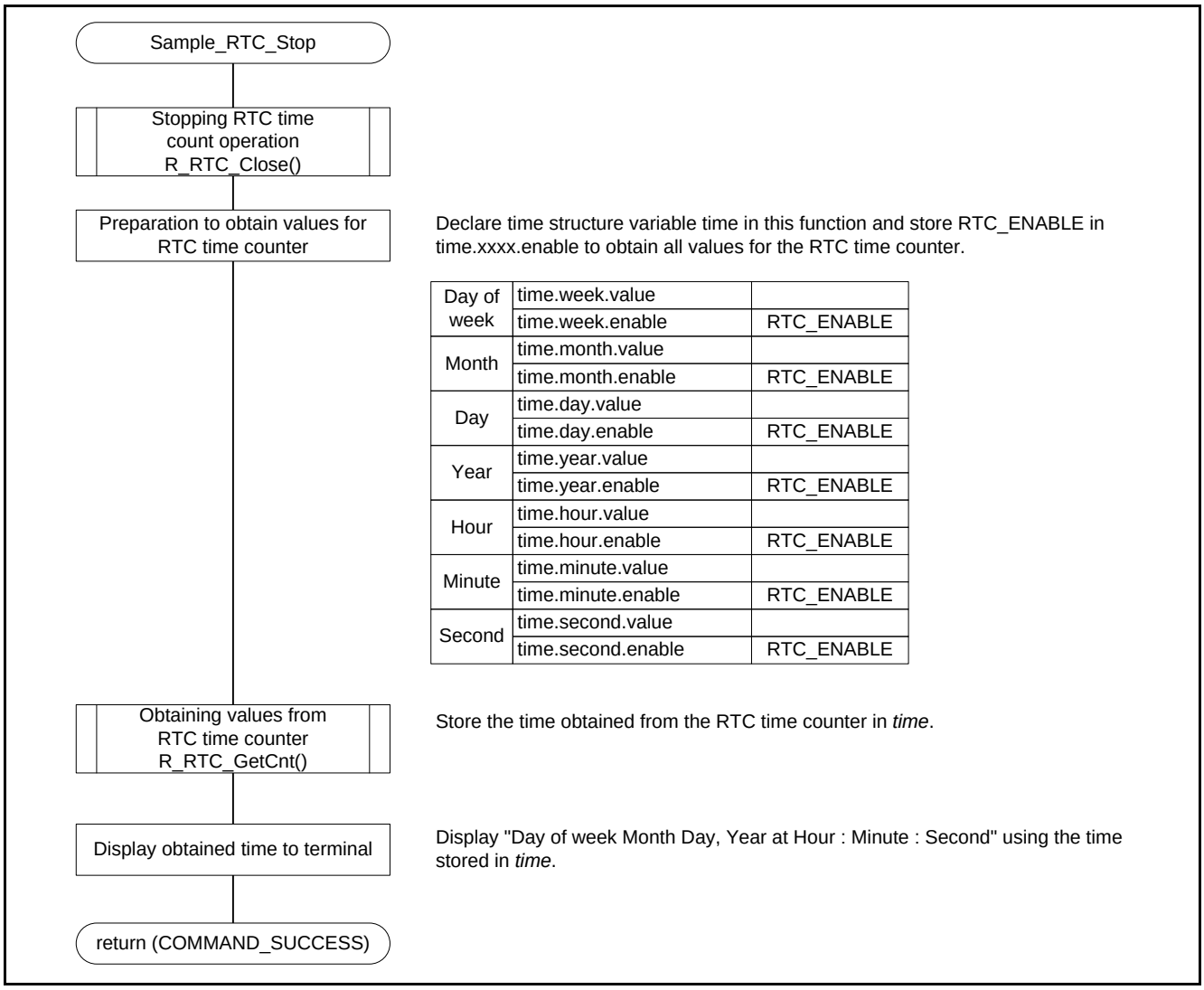


Figure 6.21 Stopping RTC Time Count Operation

6.8.13 RTC Alarm Time Setting and Transition to Deep Standby Mode

Figure 6.22 to Figure 6.24 show the flowcharts of RTC Alarm Time Setting and Transition to Deep Standby Mode.

This function runs when Command 6 is input during the RTC command wait processing of the sample function Sample_RTC_Main.

RTC has been set to generate an alarm interrupt at the time input from the terminal. It sets STB so that deep standby mode may be cancelled by the alarm interrupt and transits to deep standby mode.

This function is executed on the condition that the RTC time counter is in operation.

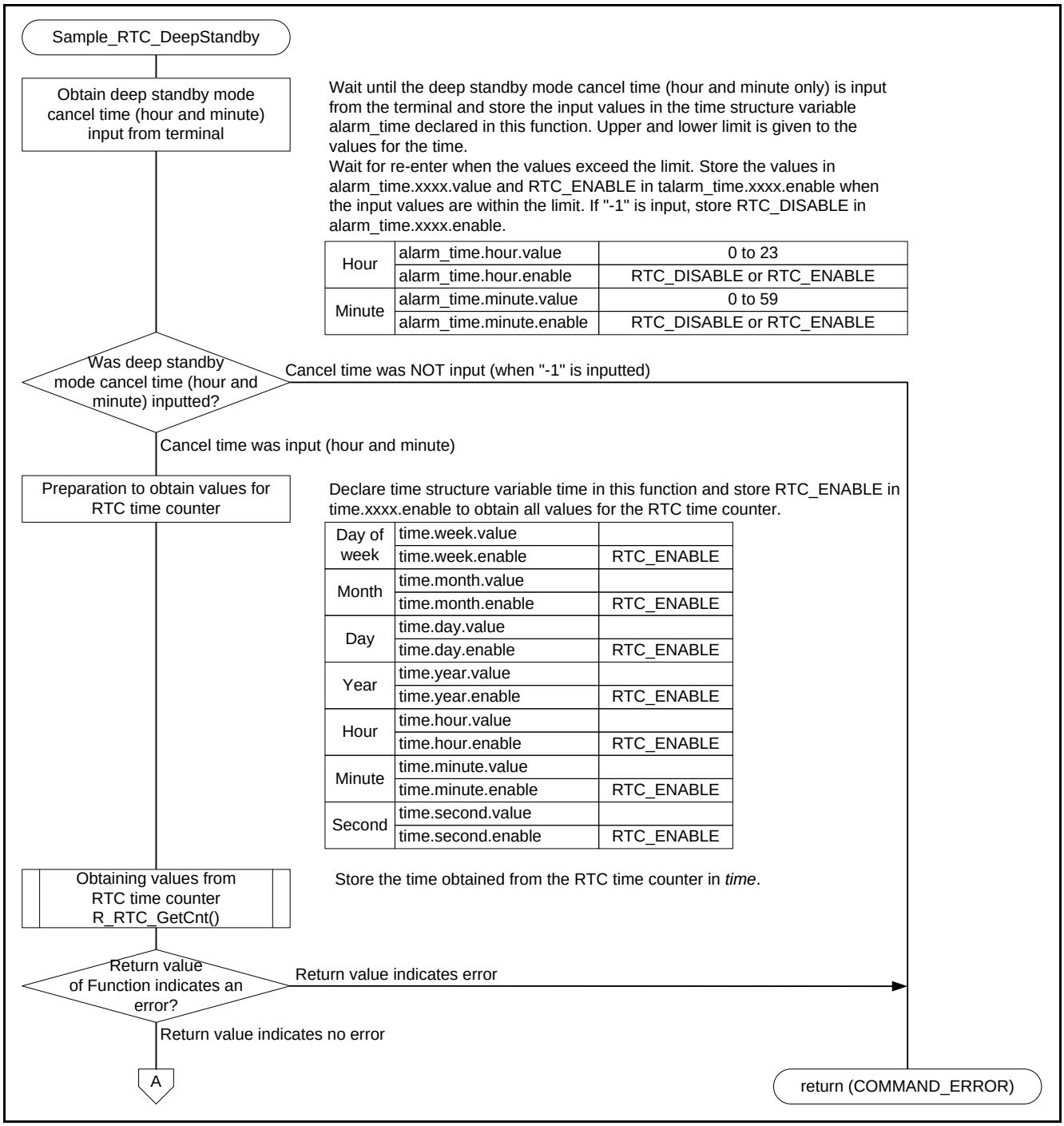


Figure 6.22 RTC Alarm Time Setting and Transition to Deep Standby Mode (1/3)

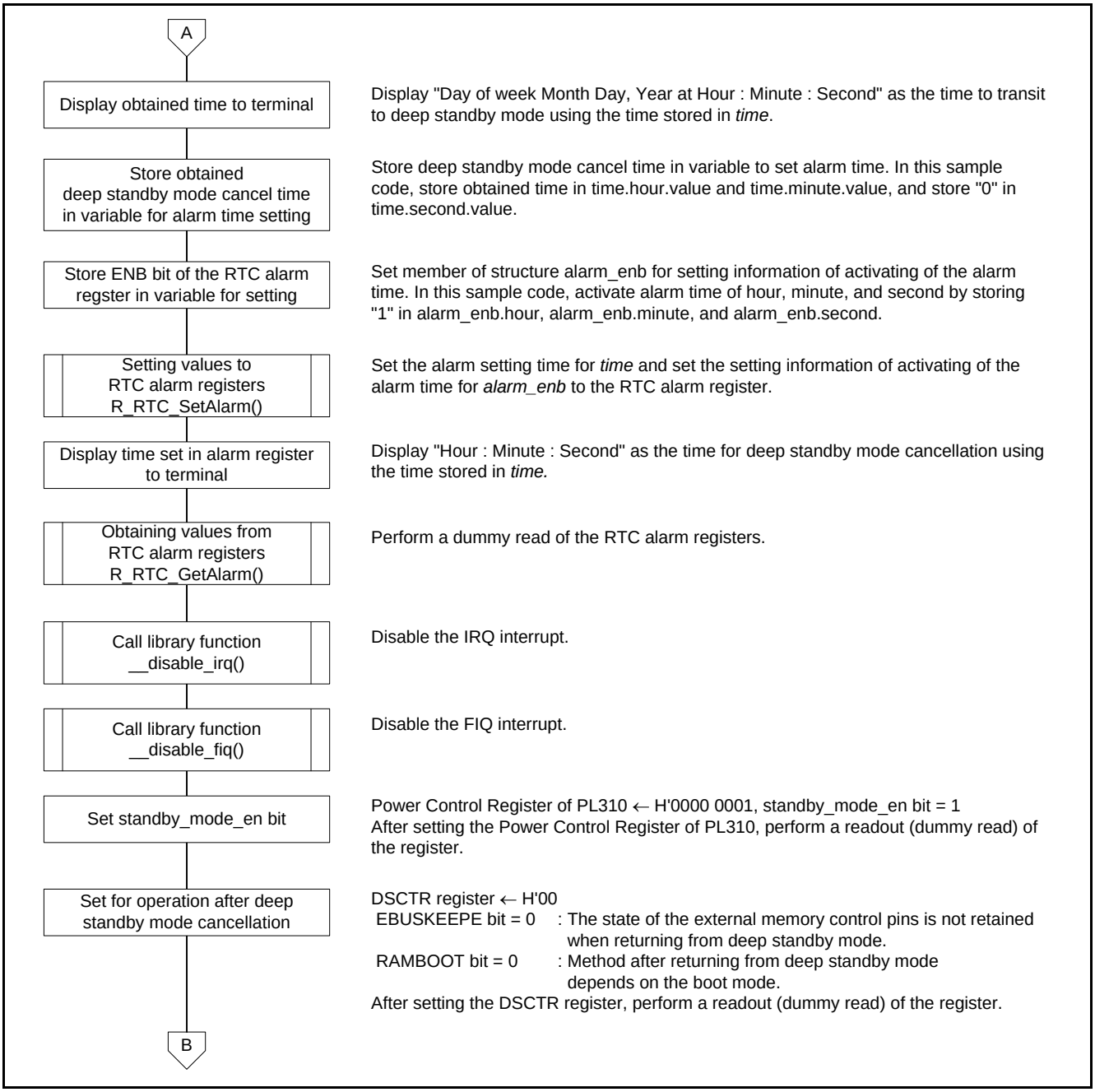


Figure 6.23 RTC Alarm Time Setting and Transition to Deep Standby Mode (2/3)

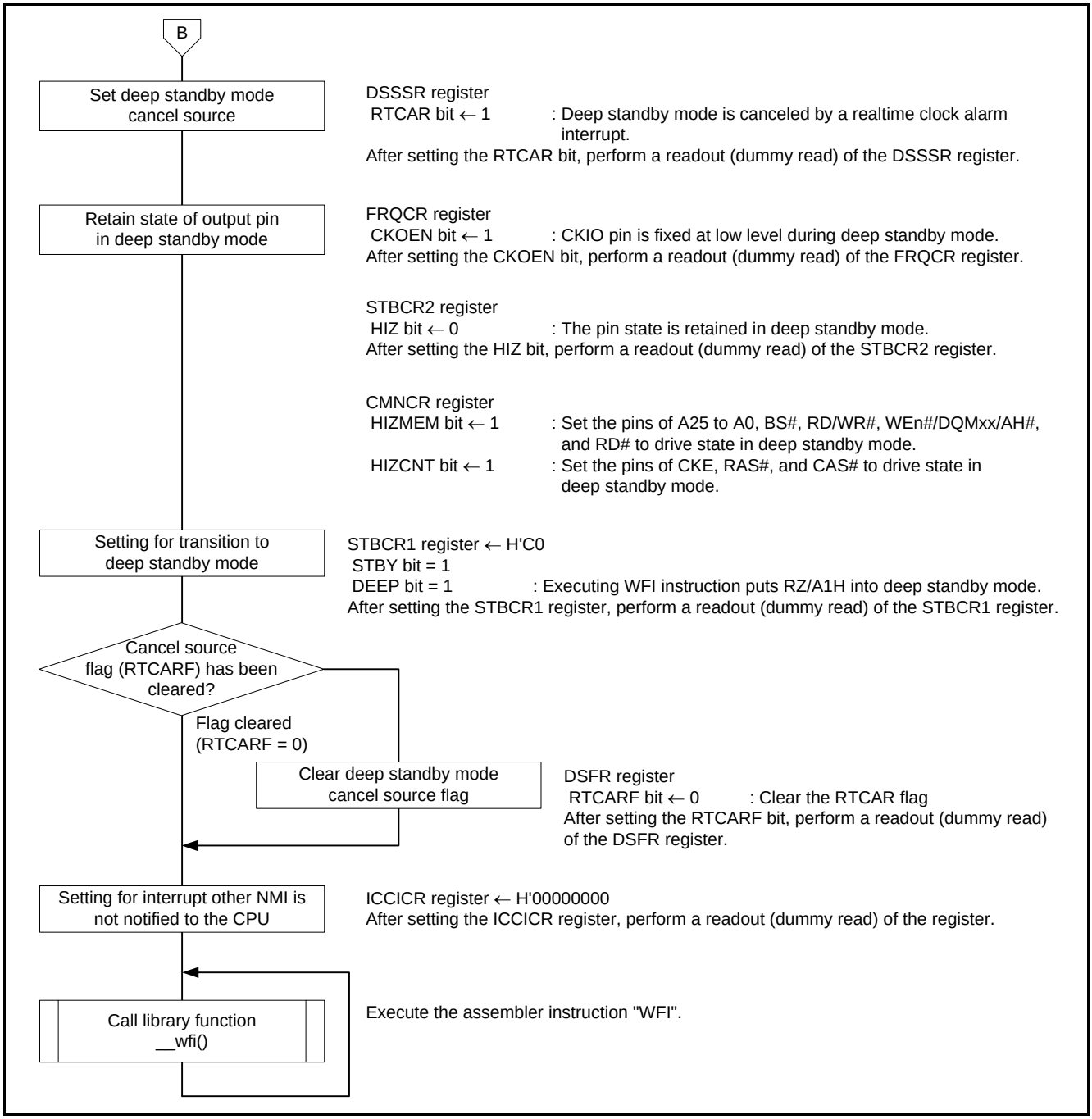


Figure 6.24 RTC Alarm Time Setting and Transition to Deep Standby Mode (3/3)

6.8.14 RTC Initial Setting

Figure 6.25 shows the flowchart of RTC Initial Setting.

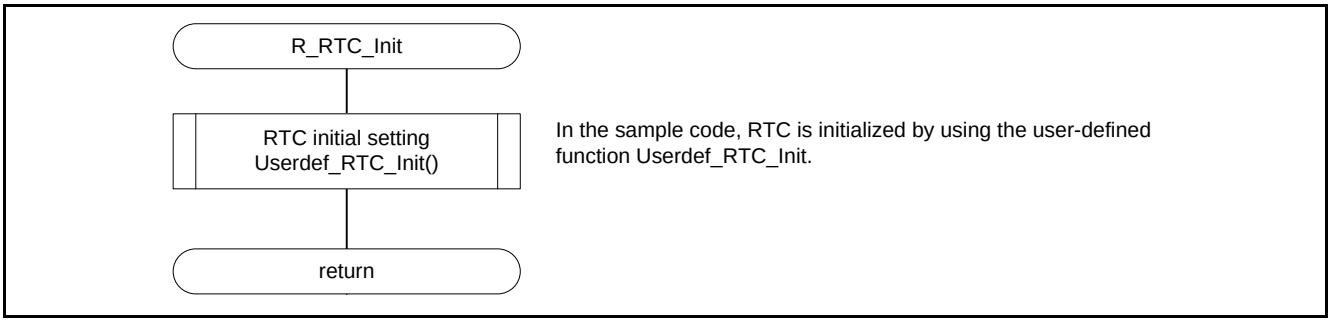


Figure 6.25 RTC Initial Setting

6.8.15 Starting RTC Time Count Operation

Figure 6.26 shows the flowchart of Starting RTC Time Count Operation.

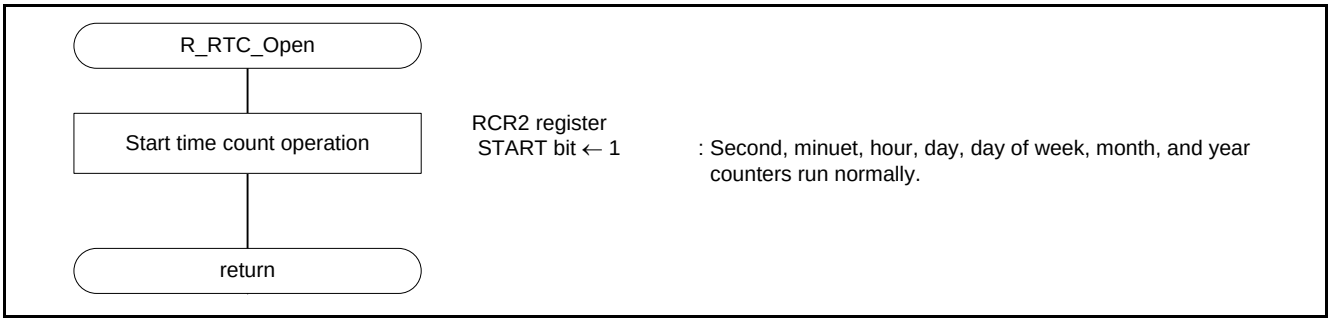


Figure 6.26 Starting RTC Time Count Operation

6.8.16 Stopping RTC Time Count Operation

Figure 6.27 shows the flowchart of Stopping RTC Time Count Operation.

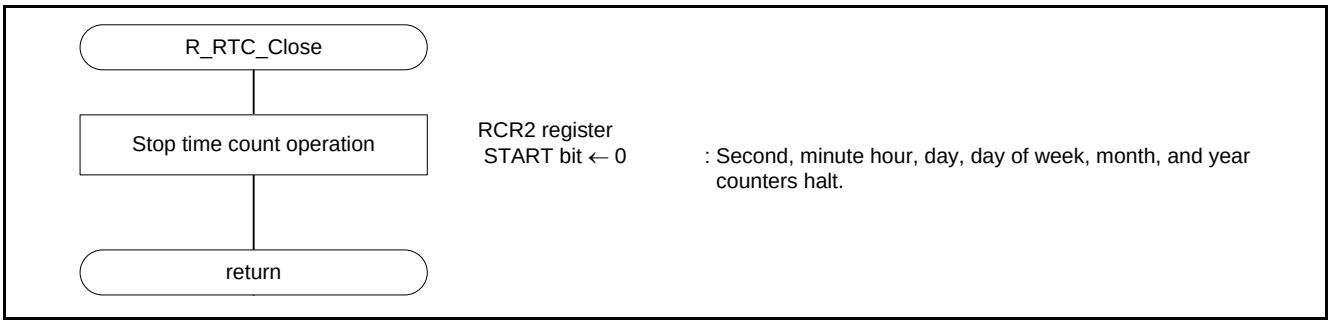


Figure 6.27 Stopping RTC Time Count Operation

6.8.17 Setting Values to RTC Time Counter

Figure 6.28 and Figure 6.29 show the flowcharts of Setting Values to RTC Time Counter.

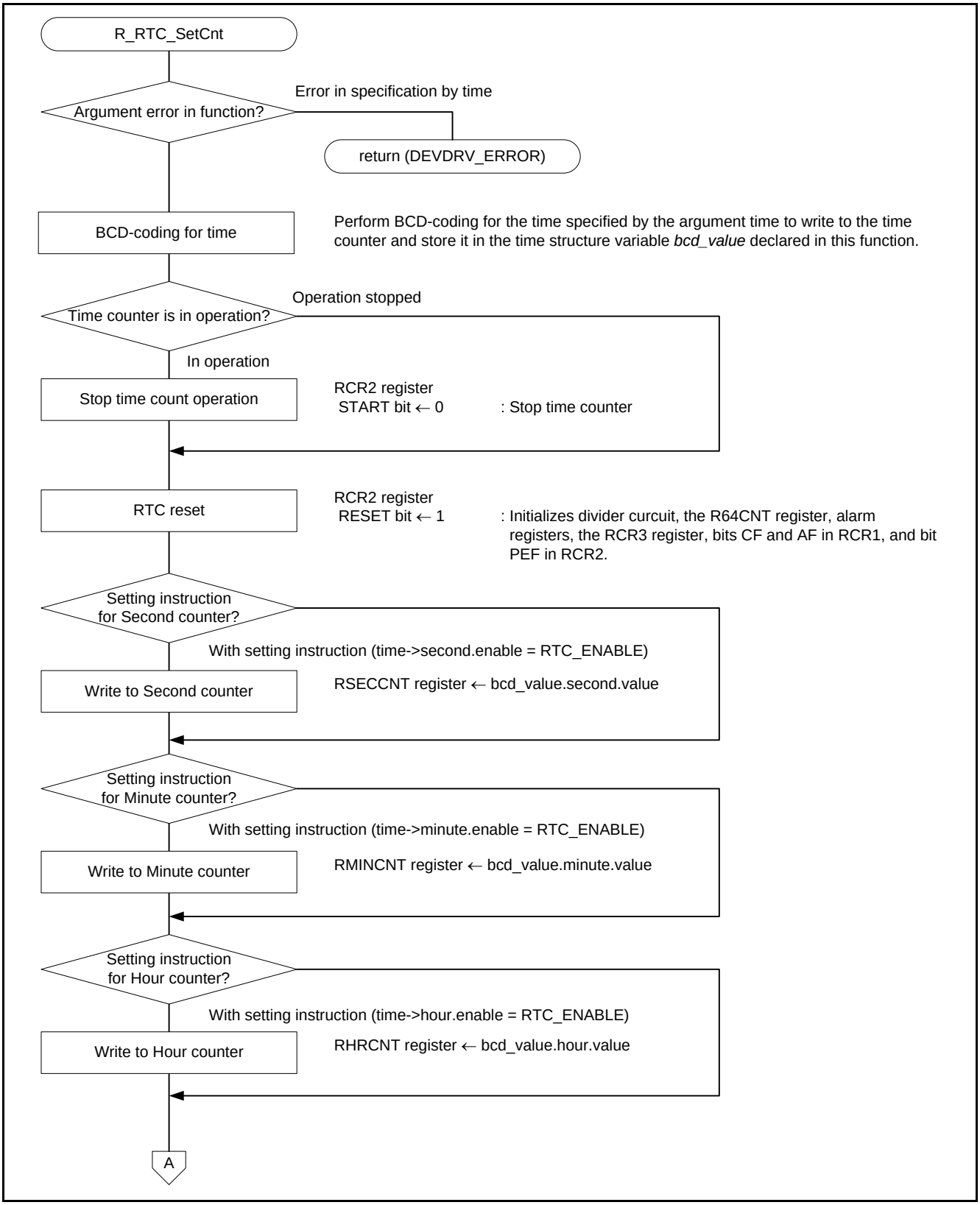


Figure 6.28 Setting Values to RTC Time Counter (1/2)

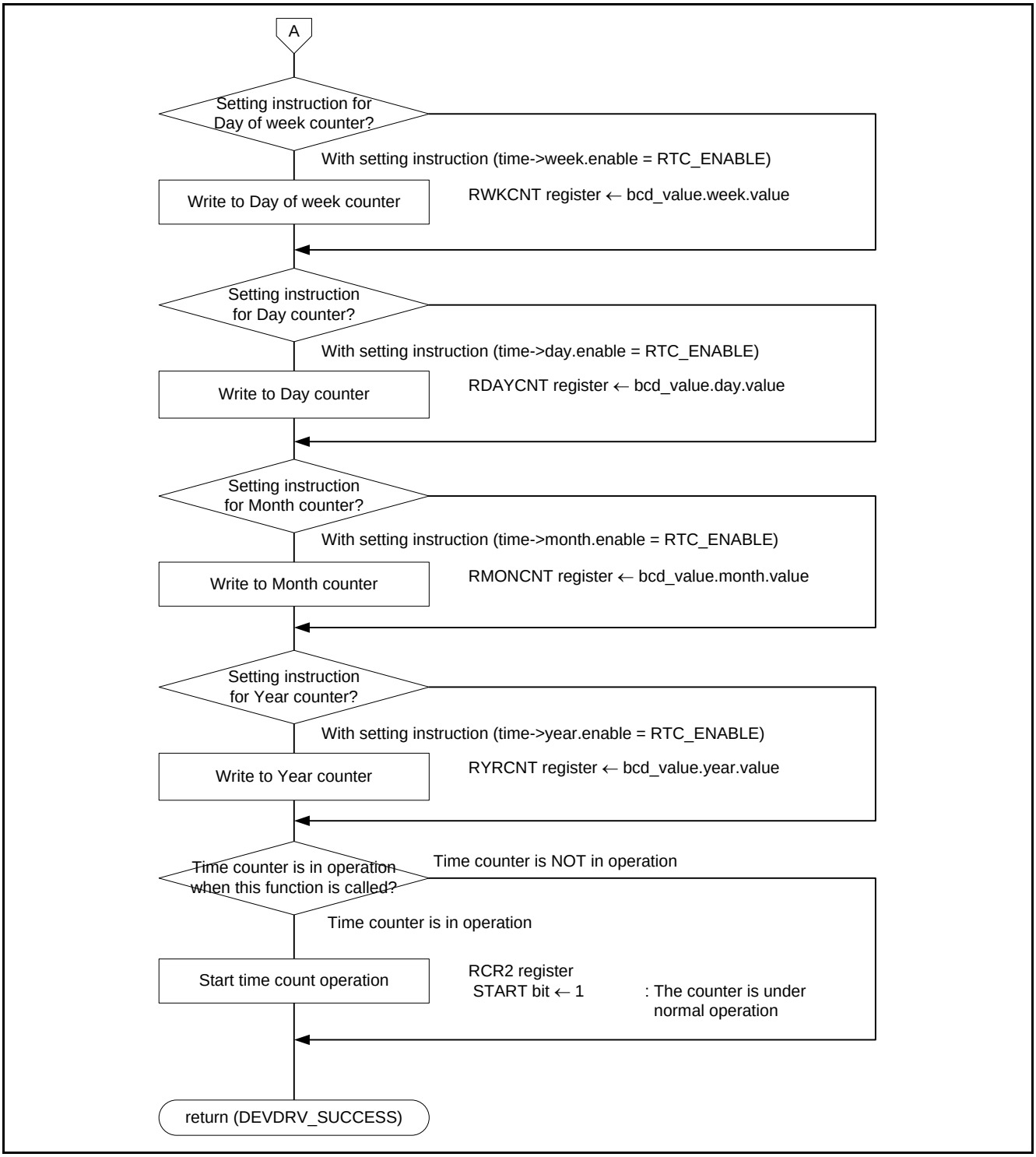


Figure 6.29 Setting Values to RTC Time Counter (2/2)

6.8.18 Obtaining Values from RTC Time Counter

Figure 6.30 and Figure 6.31 show the flowcharts of Obtaining Values from RTC Time Counter.

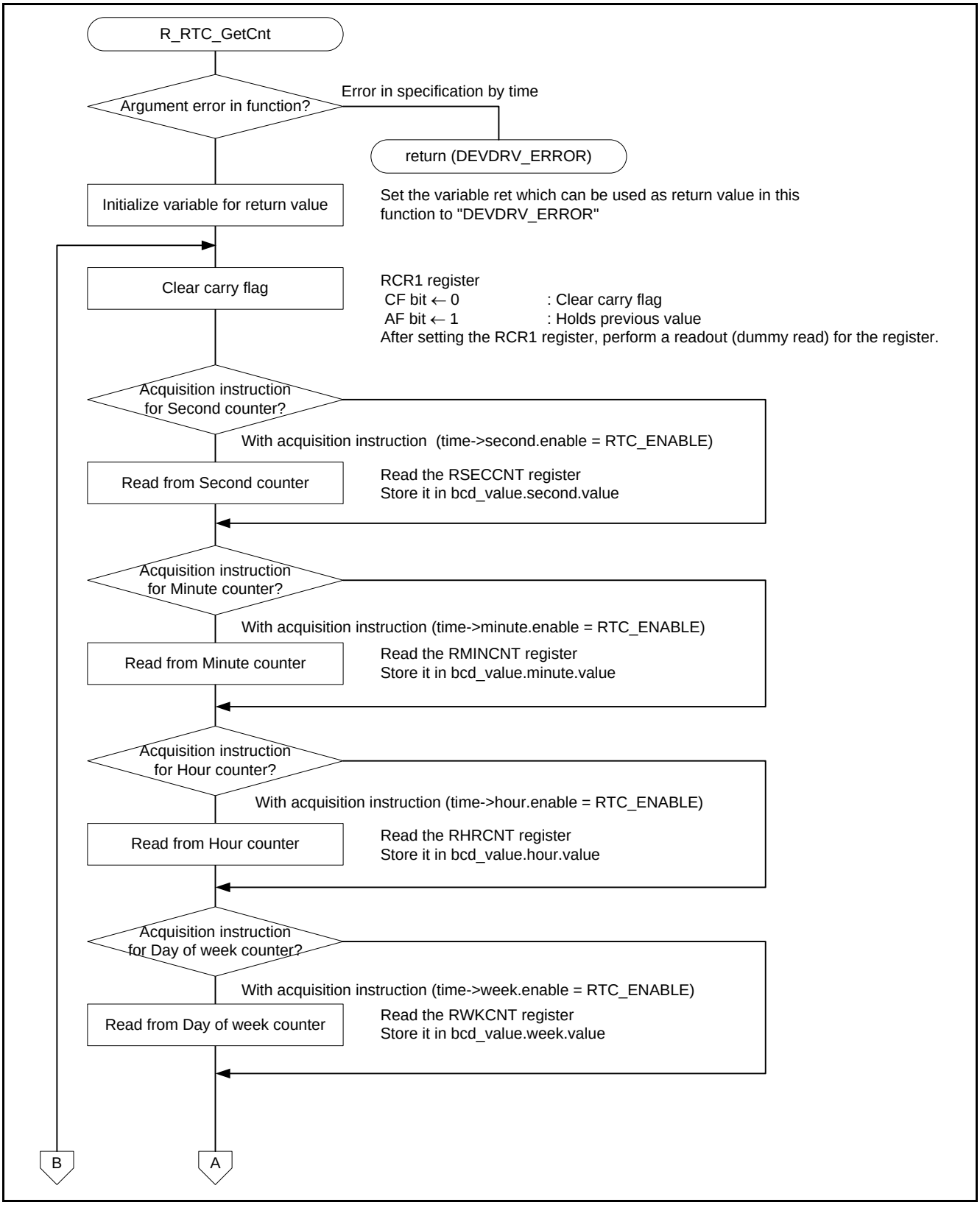


Figure 6.30 Obtaining Values from RTC Time Counter (1/2)

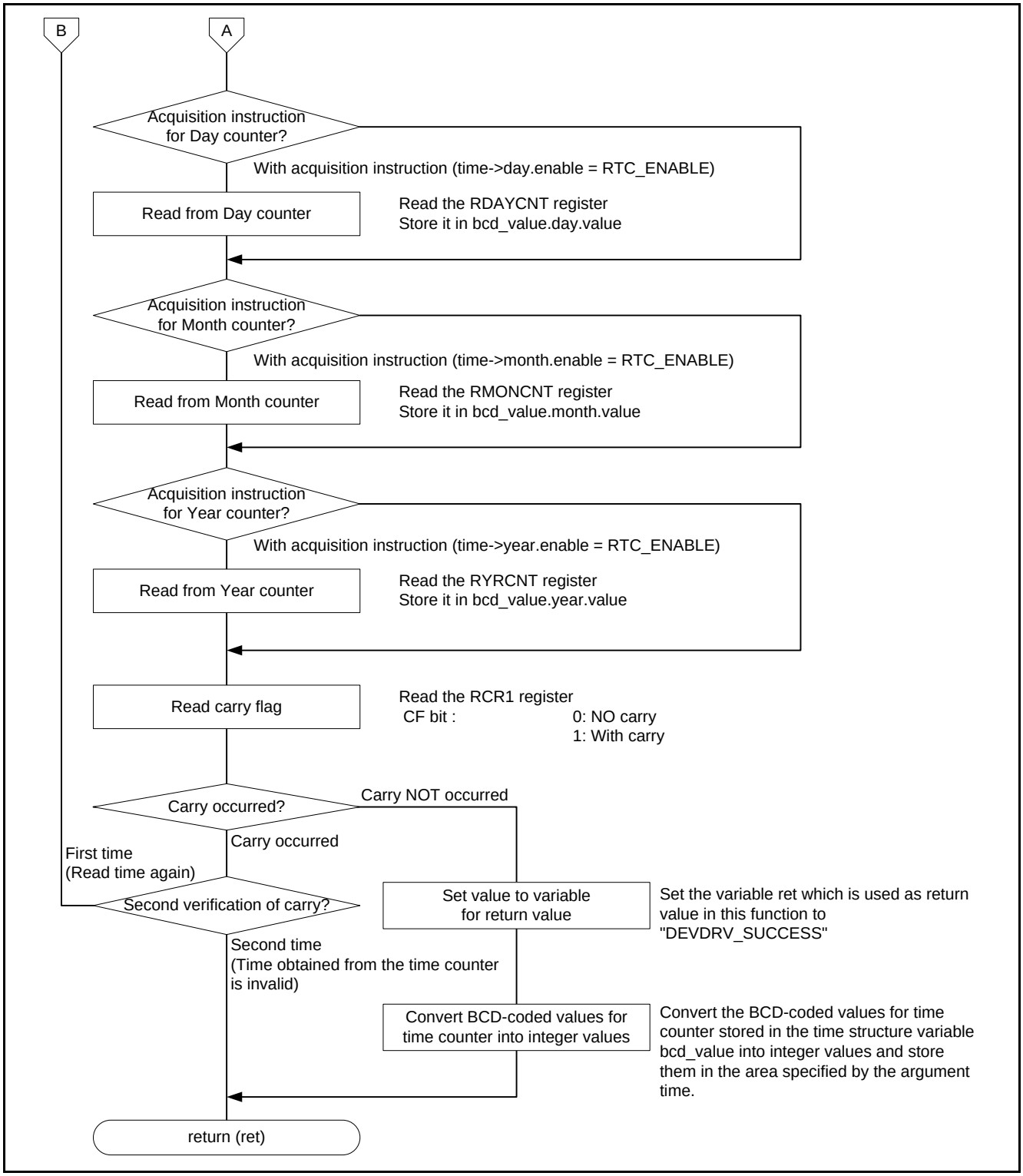


Figure 6.31 Obtaining Values from RTC Time Counter (2/2)

6.8.19 Setting Values to RTC Alarm Registers

Figure 6.32 to Figure 6.34show the flowchart of Setting Values to RTC Alarm Registers.

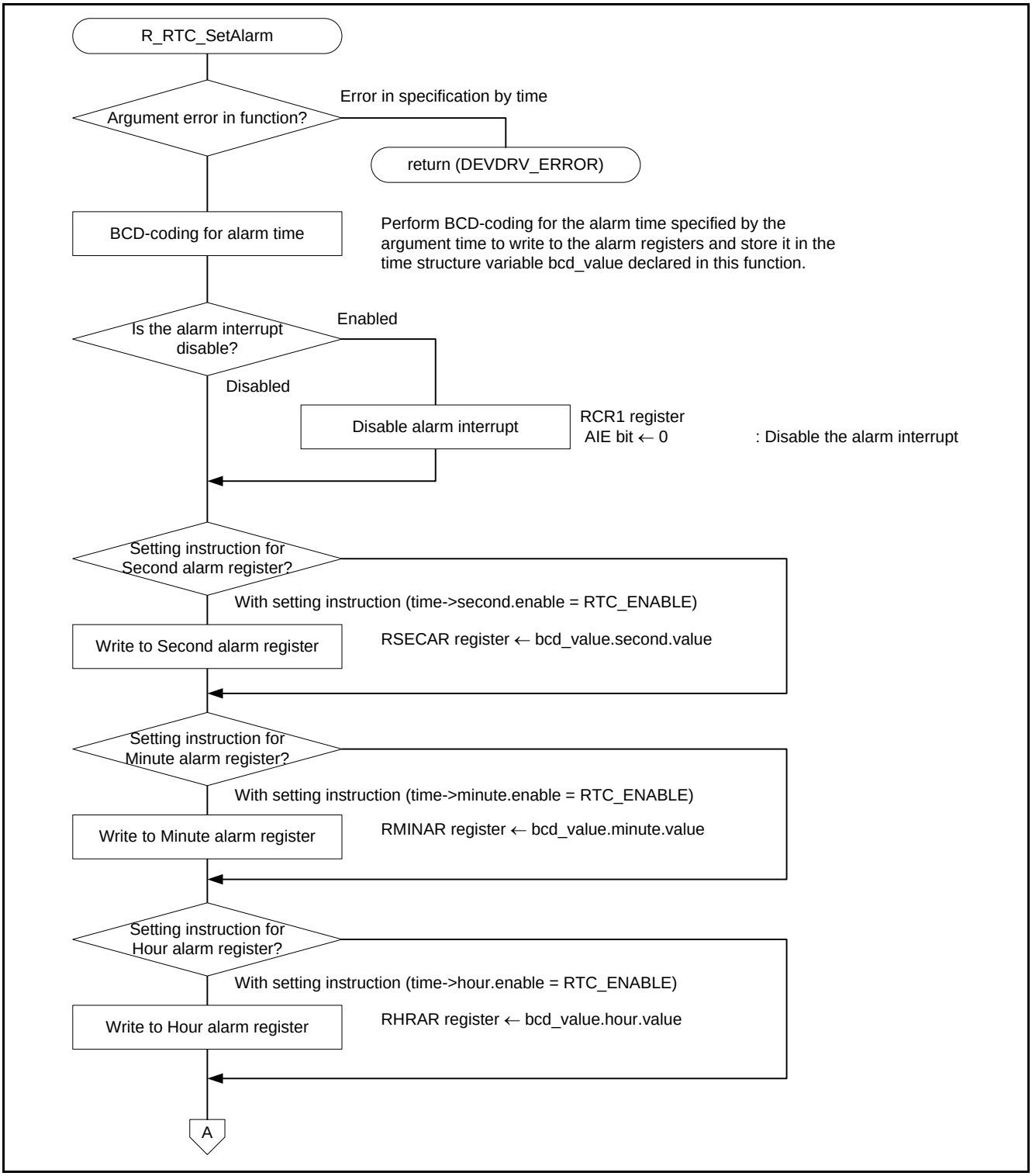


Figure 6.32 Setting Values to RTC Alarm Registers (1/3)

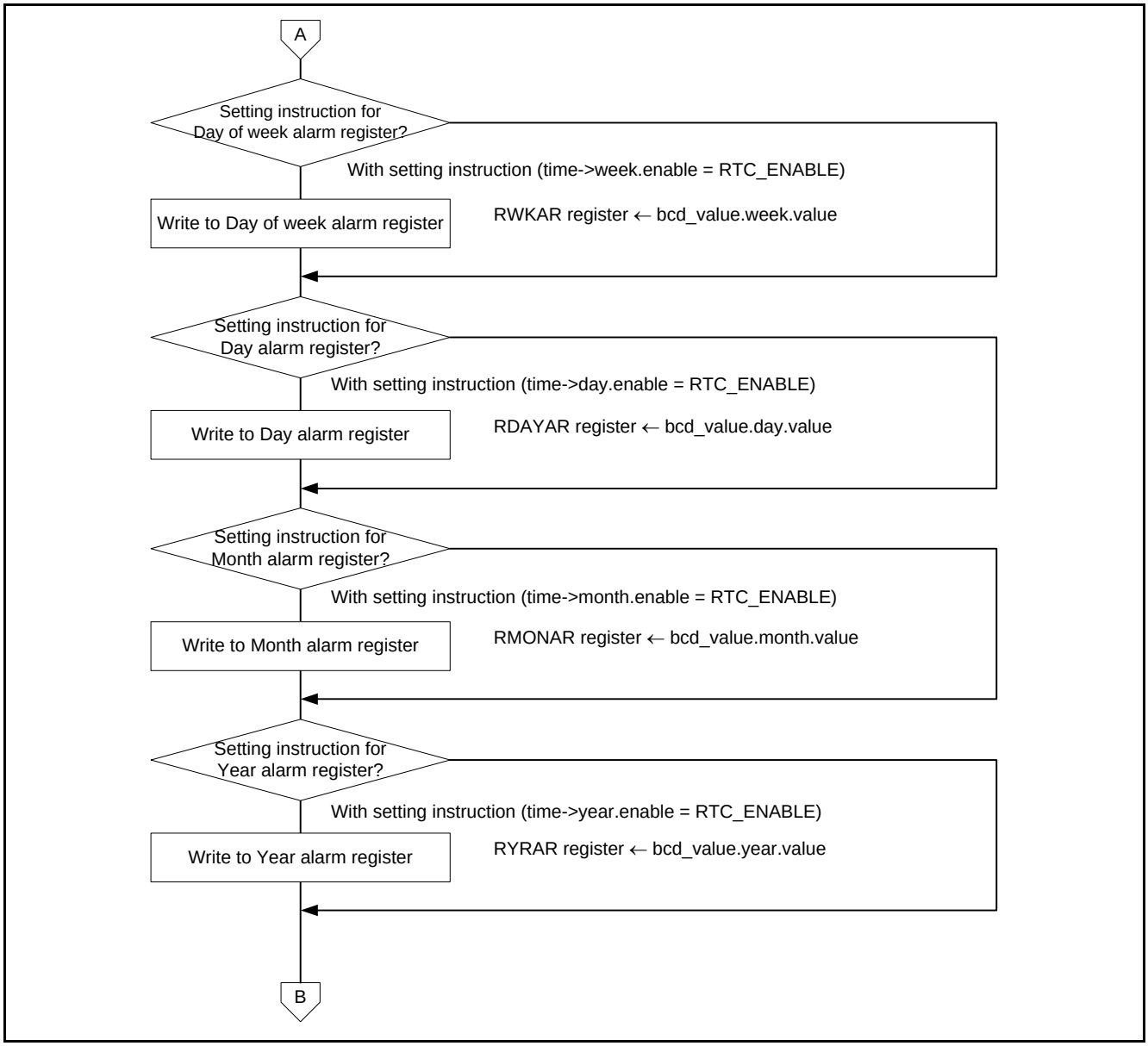


Figure 6.33 Setting Values to RTC Alarm Registers (2/3)

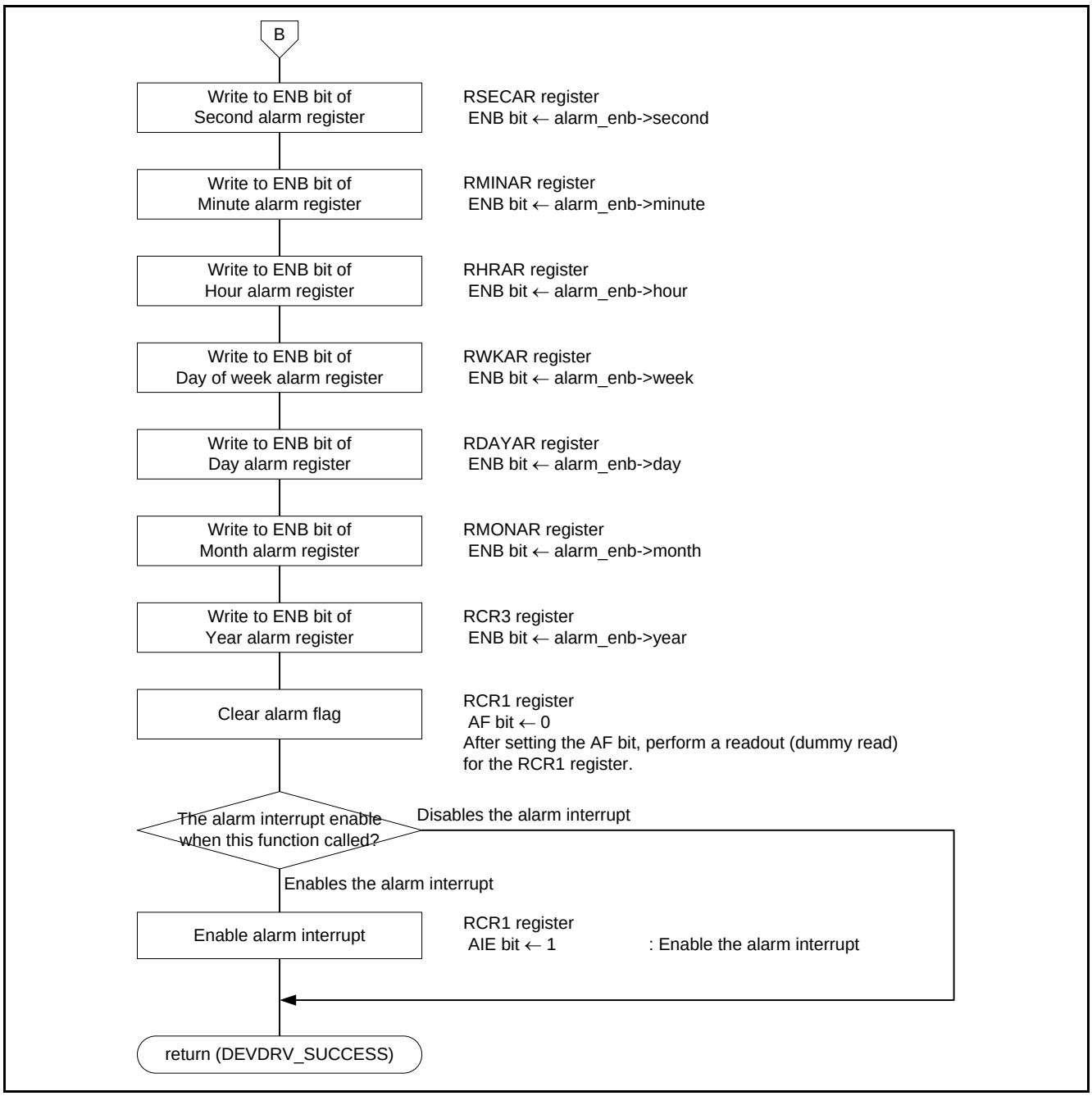


Figure 6.34 Setting Values to RTC Alarm Registers (3/3)

6.8.20 Obtaining Values from RTC Alarm Registers

Figure 6.35 to Figure 6.37show the flowcharts of Obtaining Values from RTC Alarm Registers.

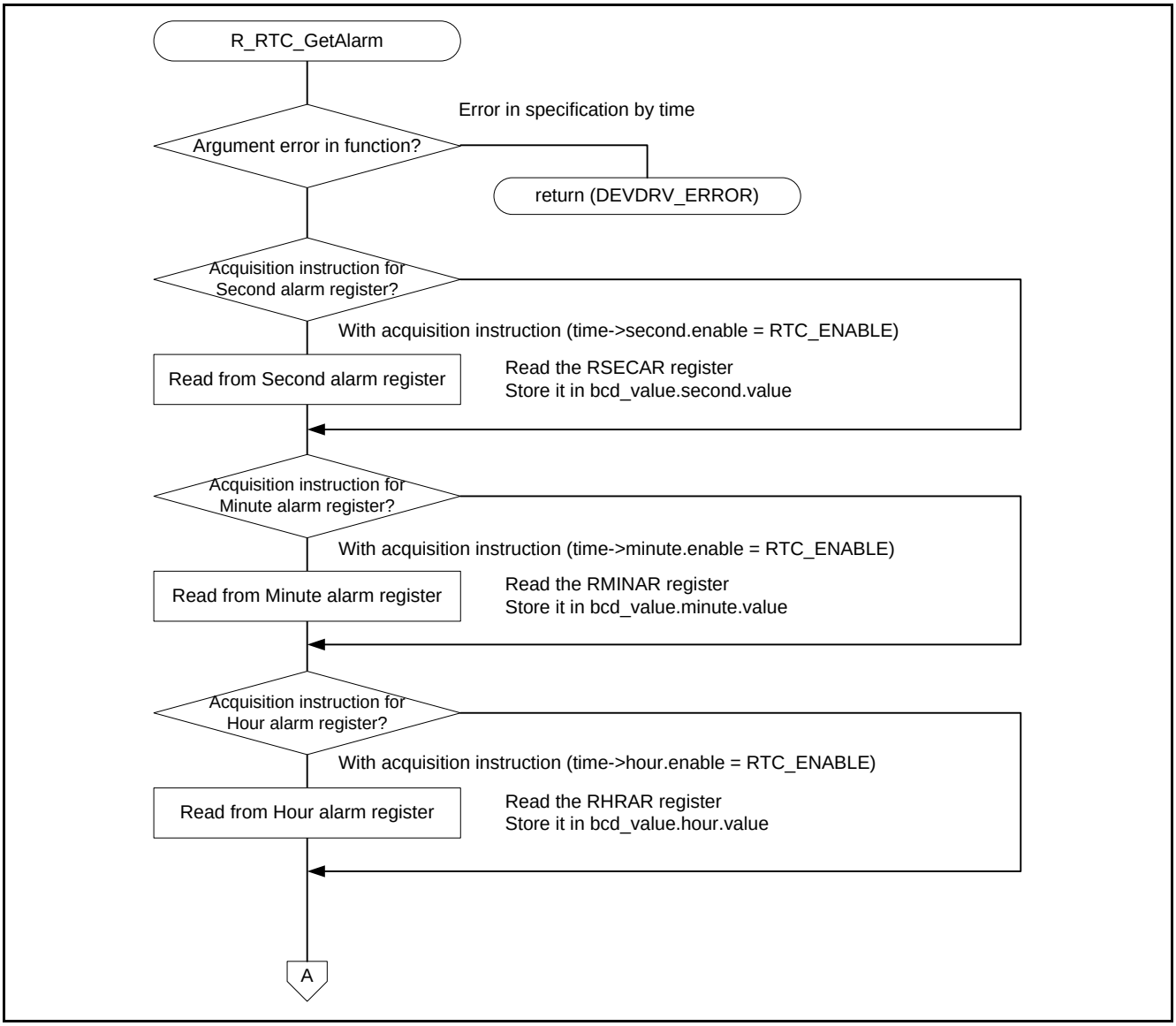


Figure 6.35 Obtaining Values from RTC Alarm Registers (1/3)

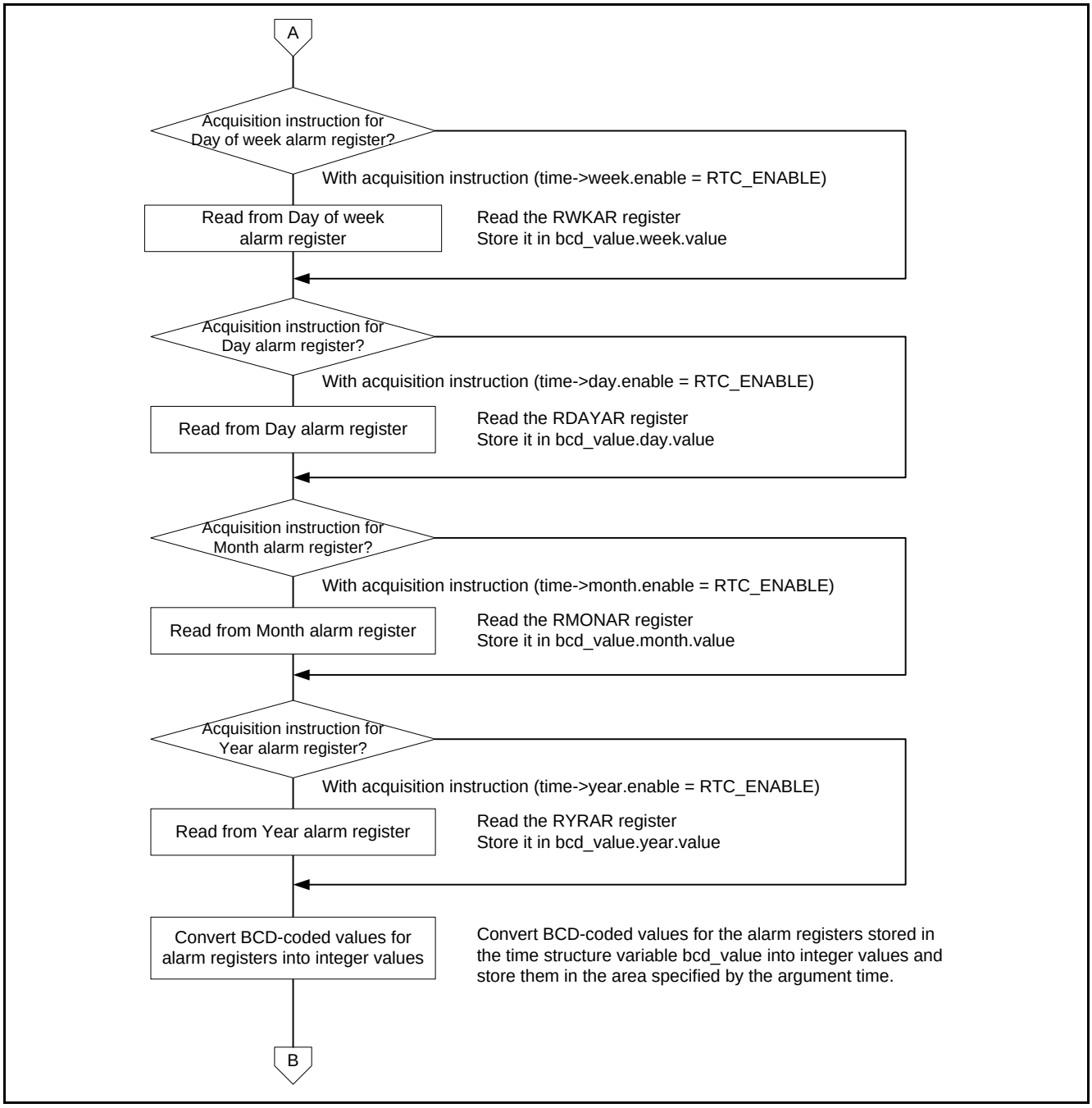


Figure 6.36 Obtaining Values from RTC Alarm Registers (2/3)

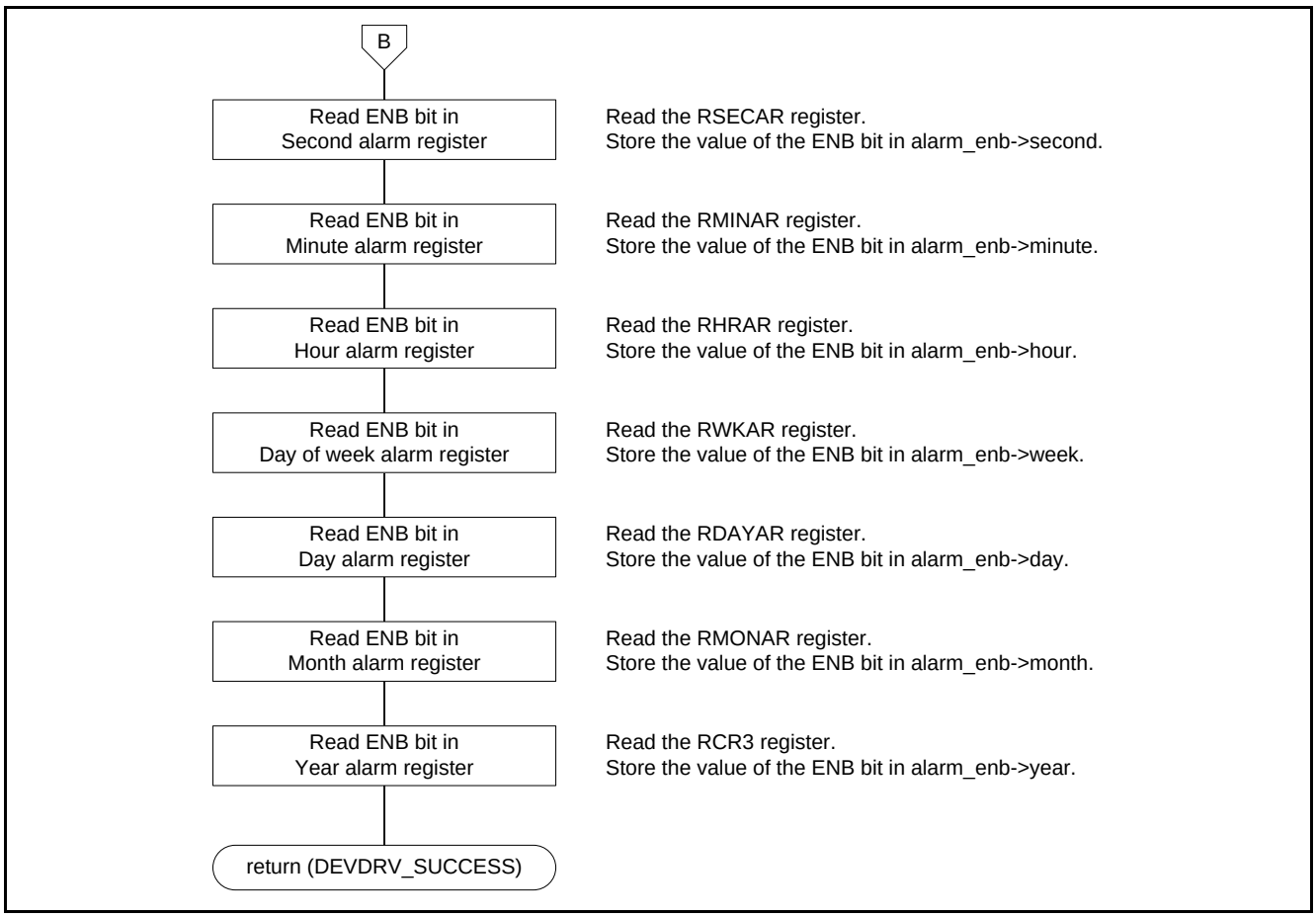


Figure 6.37 Obtaining Values from RTC Alarm Registers (3/3)

6.8.21 RTC Initial Setting

Figure 6.38 shows the flowchart of RTC Initial Setting.

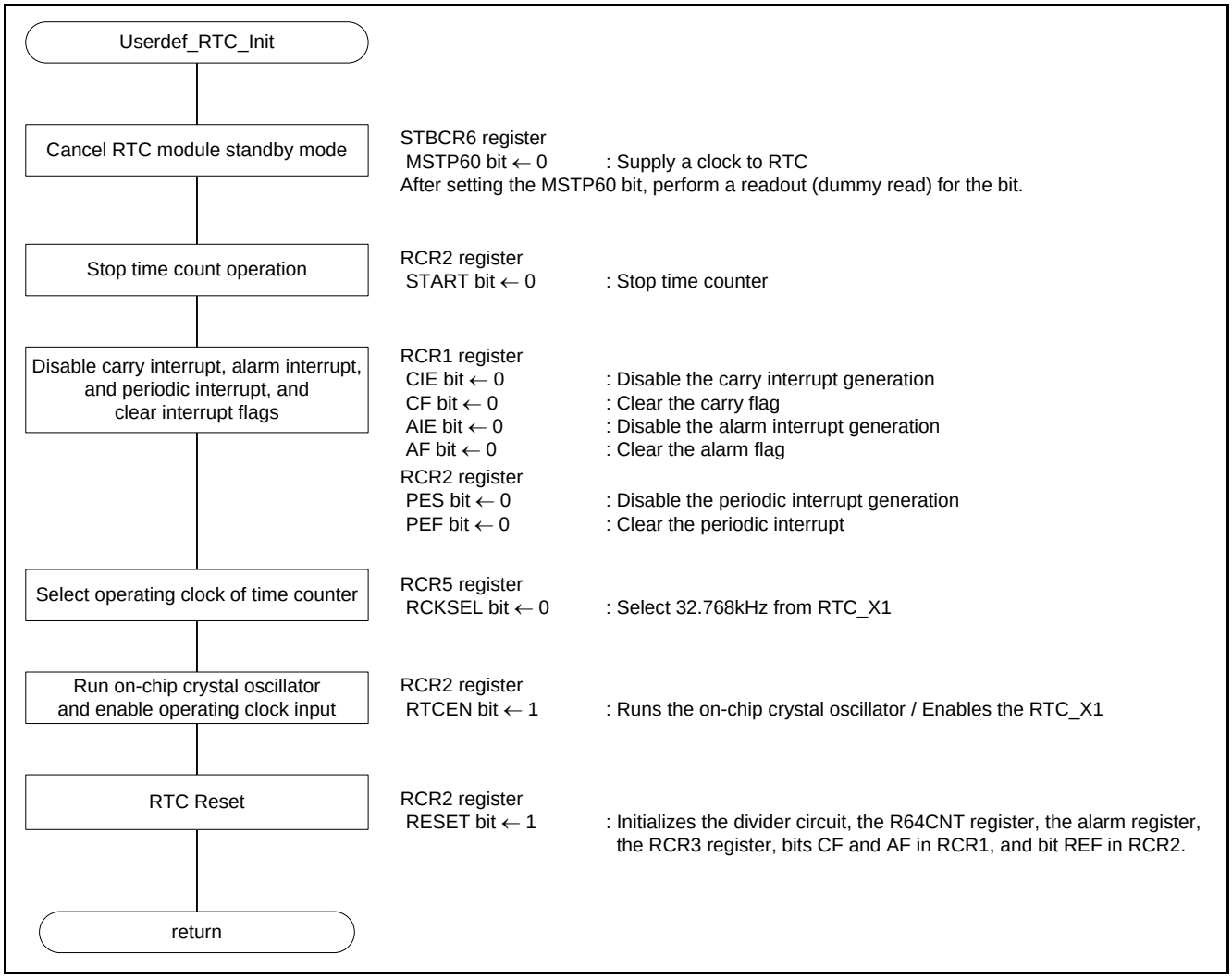


Figure 6.38 RTC Initial Setting

6.9 Running Sample Code

The sample code is operated by entering commands in the terminal program running on the host PC connected to the GENMAI board via the serial interface.

After supplying power to the GENMAI board, the message (1) in Figure 6.39 is output. To run the RTC sample code, input "RTC" + "Enter" key subsequent to the "SAMPLE>" prompt. When the message (2) in Figure 6.39 is output. Input "1" to "6" + "Enter" key subsequent to the "RTC SAMPLE>" prompt to run the RTC sample code.

By inputting "HELP" + "Enter" key, the sample code information (3) is displayed. "EXIT" + "Enter" key terminates the RTC sample code operation.

Ver.X.XX and Ver.Y.YY shows in Figure 6.39 indicates the main processing version of the sample code and the RTC sample code version respectively.

```
Display messages
```

```
RZ/A1H CPU Board Sample Program. Ver.X.XX                                (1)
Copyright (C) 2014 Renesas Electronics Corporation. All rights
reserved.

select sample program.

SAMPLE>
```

```
RZ/A1H Realtime Clock(RTC) Sample Program. Ver.Y.YY                      (2)
Copyright (C) 2014 Renesas Electronics Corporation. All rights
reserved.

select sample program.

RTC SAMPLE>
```

```
RTC SAMPLE> help                                                         (3)

  1    : Initialize RTC
  2    : Set time
  3    : Get time
  4    : Start RTC
  5    : Stop RTC
  6    : Transition to Deep Standby Mode
       :  -> Canceled by RTC alarm interrupt
  EXIT : Exit RTC sample

RTC SAMPLE>
```

Figure 6.39 Terminal Display at RTC Sample Code Startup

7. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

8. Reference Documents

User's Manual: Hardware

RZ/A1H Group User's Manual: Hardware
The latest version can be downloaded from the Renesas Electronics website.

R7S72100 RTK772100BC00000BR (GENMAI) User's Manual
The latest version can be downloaded from the Renesas Electronics website.

ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition Issue C
The latest version can be downloaded from the ARM website.

ARM Generic Interrupt Controller Architecture Specification Architecture version 1.0
The latest version can be downloaded from the ARM website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

User's Manual: Development Tools

ARM software development tools (ARM Compiler toolchain, ARM DS-5 etc.) are available on the ARM website.
The latest version can be downloaded from the ARM website.

Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
Rev.0.81	Sep. 05, 2014	-	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, 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.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

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: +852-2265-6688, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

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.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2014 Renesas Electronics Corporation. All rights reserved.

Colophon 4.0