RE01 1500KB Group

R_SYSTEM Driver Detailed Specification

R01AN4770EJ0130
Rev.1.30
May. 12, 2021

Introduction

This document describes the detailed specifications of the system driver R_SYSTEM provided in the RE01 1500KB CMSIS Driver Package.

Target Device

RE01 1500KB Group

Contents

1. Overview ... 3

2. Internal Structure of Software Components ... 4
2.1 File Structure ... 4

3. Internal Operation of Software Components ... 7

4. Detailed Information of Software Unit ... 8
4.1 Configurations ... 8
4.1.1 Parameter Checking ... 8
4.1.2 Critical Section ... 8
4.1.3 Register Protection ... 8
4.1.4 Value of API Timeout ... 9
4.1.5 Event Link Number Setting ... 9
4.1.6 Function Allocation to RAM ... 13
4.2 Macro and Type Definitions ... 14
4.3 Function Specifications ... 16
4.3.1 R_SYS_CodeCopy Function ... 16
4.3.2 R_SYS_Initialize Function ... 17
4.3.3 R_SYS_BoostSpeedModeSet Function ... 18
4.3.4 R_SYS_HighSpeedModeSet Function ... 20
4.3.5 R_SYS_LowSpeedModeSet Function ... 23
4.3.6 R_SYS_32kHzSpeedModeSet Function ... 25
4.3.7 R_SYS_SpeedModeGet Function ... 27
4.3.8 R_SYS_SystemClockHOCOSet Function ... 28
4.3.9 R_SYS_SystemClockMOCOSet Function ... 31
4.3.10 R_SYS_SystemClockLOCOSet Function ... 32
4.3.11 R_SYS_SystemClockMOSCSet Function ... 33
4.3.12 R_SYS_SystemClockSOSCSet Function ... 34
4.3.13 R_SYS_SystemClockPLLSet Function ... 35
4.3.14 R_SYS_SystemClockFreqGet Function ... 38
4.3.15 R_SYS_PeripheralClockFreqGet Function ... 40
4.3.16 R_SYS_SystemClockDividerSet Function ... 42
4.3.17 R_SYS_MainOscSpeedClockStart Function ... 45
4.3.18 R_SYS_MainOscSpeedClockStop Function ... 46
4.3.19 R_SYS_HighSpeedClockStart Function ... 47

4.3.20 R_SYS_HighSpeedClockStop Function 49
4.3.21 R_SYS_MediumSpeedClockStart Function 50
4.3.22 R_SYS_MediumSpeedClockStop Function 51
4.3.23 R_SYS_LowSpeedClockStart Function 52
4.3.24 R_SYS_LowSpeedClockStop Function 53
4.3.25 R_SYS_SubOscSpeedClockStart Function 54
4.3.26 R_SYS_SubOscSpeedClockStop Function 55
4.3.27 R_SYS_PLLSpeedClockStart Function 56
4.3.28 R_SYS_PLLSpeedClockStop Function 58
4.3.29 R_SYS_OscStabilizationFlagGet Function 59
4.3.30 R_SYS_IrqEventLinkSet Function 60
4.3.31 R_SYS_IrqStatusGet Function 61
4.3.32 R_SYS_IrqStatusClear Function 62
4.3.33 R_SYS_EnterCriticalSection Function 63
4.3.34 R_SYS_ExitCriticalSection Function 63
4.3.35 R_SYS_ResourceLock Function 64
4.3.36 R_SYS_ResourceUnlock Function 64
4.3.37 R_SYS_RegisterProtectEnable Function 65
4.3.38 R_SYS_RegisterProtectDisable Function 65
4.3.39 R_SYS_SoftwareDelay Function 66
4.3.40 R_SYS_GetVersion Function 68
4.3.41 r_sys_BoostFlagGet Function 69
4.3.42 r_sys_BoostFlagSet Function 69
4.3.43 r_sys_BoostFlagClr Function 70
4.3.44 r_system_wdt_refresh Function 70
4.3.45 IELn_IRQHandler Function (n = 0 to 31) 71
4.3.46 R_NVIC_EnableIRQ Function 72
4.3.47 R_NVIC_GetEnableIRQ Function 73
4.3.48 R_NVIC_DisableIRQ Function 74
4.3.49 R_NVIC_GetPendingIRQ Function 75
4.3.50 R_NVIC_SetPendingIRQ Function 76
4.3.51 R_NVIC_ClearPendingIRQ Function 77
4.3.52 R_NVIC_SetPriority Function 78
4.3.53 R_NVIC_GetPriority Function 79
4.3.54 R_NVIC_SetVector Function 80
4.3.55 R_NVIC_GetVector Function 80
4.3.56 R_NVIC_SystemReset Function 81

Website and Support 82

1. Overview

The following shows a list of abbreviations used in this document and a list of related documents.

Table 1-1 Abbreviation List

Name	Abbreviation
RENESAS-DRIVER R_SYSTEM	R_SYSTEM Driver
RENESAS CMSIS-Core	R_CORE
RE01 Group User's Manual: Hardware	UMH

Table 1-2 Related Document List

Document Name	Document Number
RE01 Group (with 1.5-Mbyte Flash Memory) User's Manual: Hardware	r01uh0796
RE01 1500KB,256KB Group Getting Started Guide to Development Using CMSIS Package	r01an4660

Table 1-3 ROM and RAM Size List

ROM/RAM Name	Cache Type	Size
Program ROM	ROM/Flash memory	1.5 Mbytes
ROM	ROM/Flash memory	256 bytes
Option-setting memory	ROM/Flash memory	32 bytes
Memory mirror	ROM/Flash memory	8 Mbytes
RAM	RAM	256 Kbytes

Table 1-4 Maximum Stack Size

Maximum stack size	0x400 (1 Kbyte)

2. Internal Structure of Software Components

2.1 File Structure

The R_SYSTEM driver is part of the Hardware Abstraction Layer (HAL) compatible with the CMSIS Driver specification, this consists of three files: r_system_api.c, r_system_api.h, and r_system_cfg.h in the vendor-specific file storage directory. The roles of the files are shown in Table 2-1. Figure 2.1 shows the file structure of the R_SYSTEM driver in the RE01 1500KB Group CMSIS Driver Package. The R_SYSTEM driver capabilities are implemented by the functions shown in Figure 2.2.

Table 2-1 Roles of the Files of R_SYSTEM Driver

File Name	Description
r_system_api.c	Driver source file. It provides the entities of driver functions. To use the R_SYSTEM driver, it is necessary to build this file.
r_system_api.h	Driver header file. It provides macro, type, and prototype declarations that can be referenced by the user. To use the R_SYSTEM driver, it is necessary to include this file.
r_system_cfg.h	Configuration definition file. It provides configuration definitions that can be modified by the user.

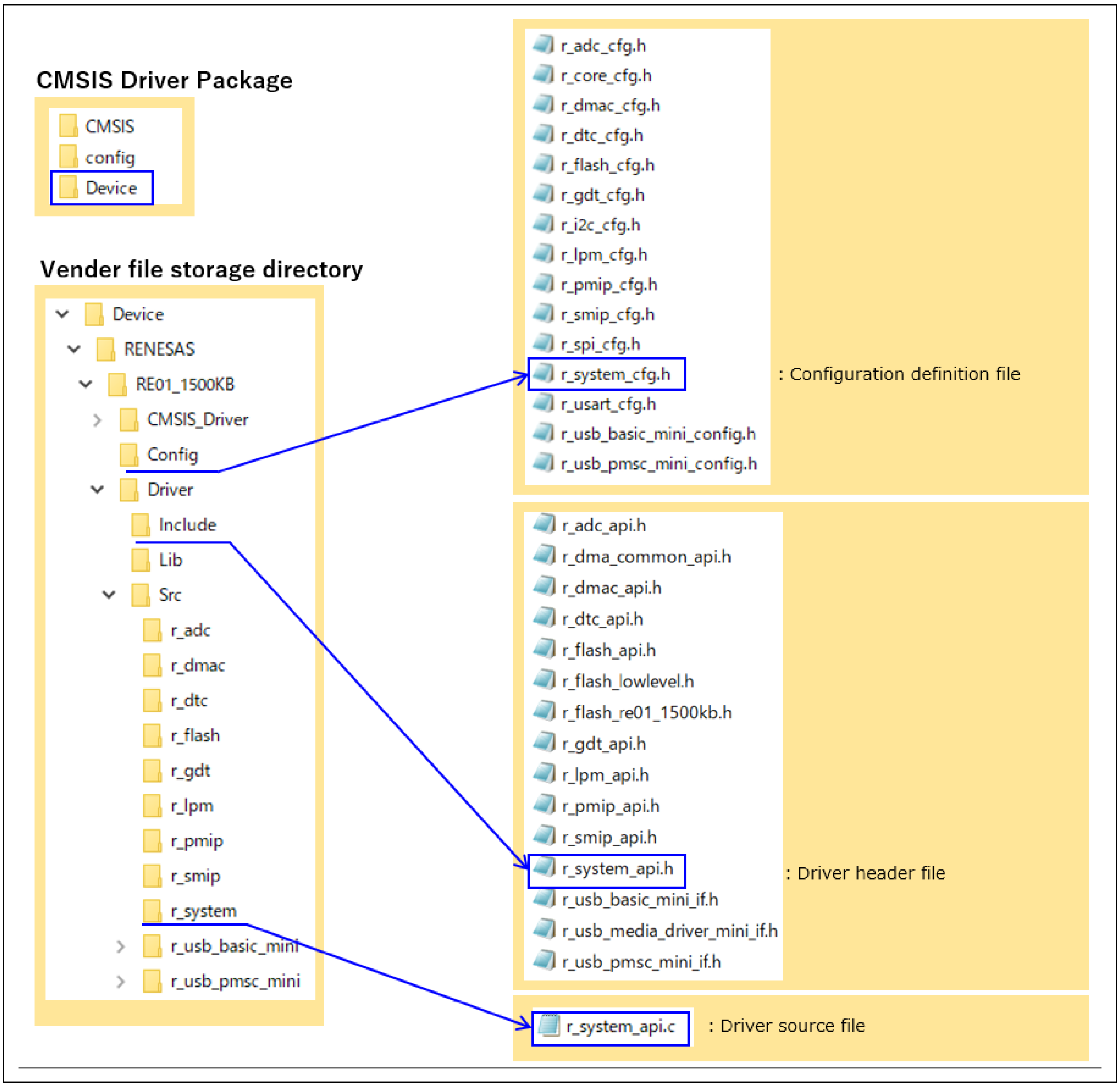


Figure 2.1 File Structure of R_SYSTEM Driver in CMSIS Driver Package

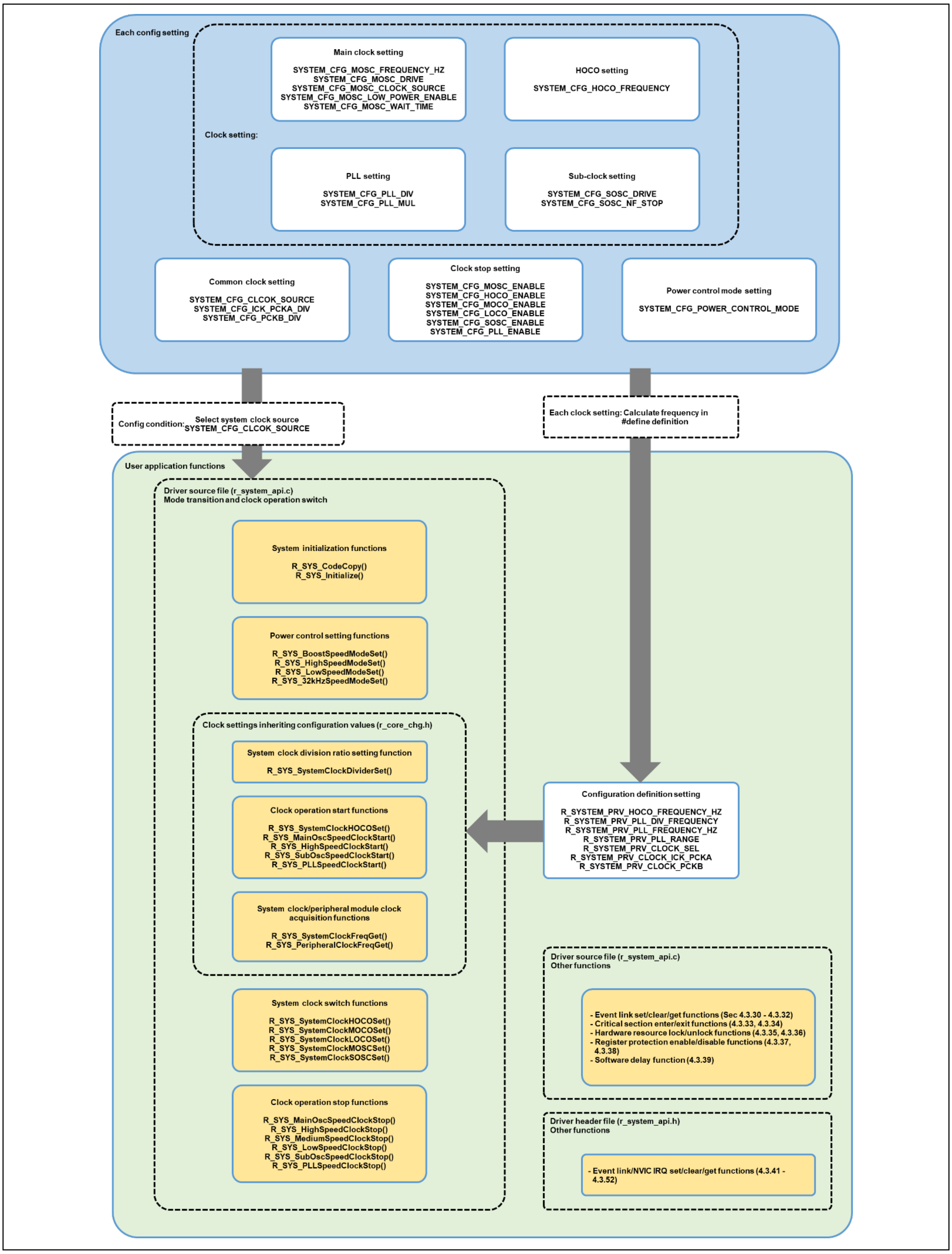
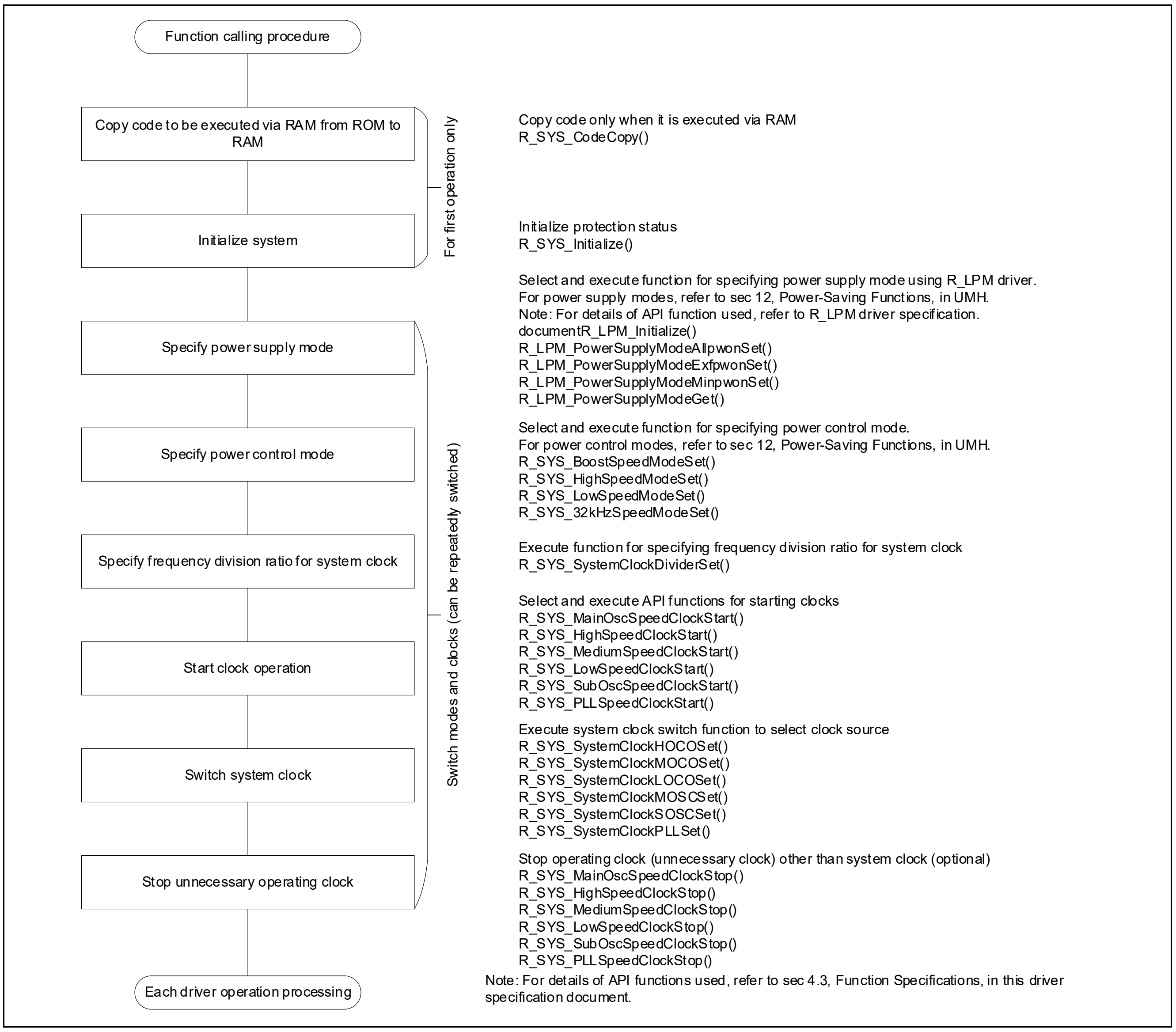

Each config setting
Clock setting:
Main clock setting
SYSTEM_CFG_MOSC_FREQUENCY_HZ
SYSTEM_CFG_MOSC_DRIVE
SYSTEM_CFG_MOSC_CLOCK_SOURCE
SYSTEM_CFG_MOSC_LOW_POWER_ENABLE
SYSTEM_CFG_MOSC_WAIT_TIME
HOCO setting
SYSTEM_CFG_HOCO_FREQUENCY
PLL setting
SYSTEM_CFG_PLL_DIV
SYSTEM_CFG_PLL_MUL
Sub-clock setting
SYSTEM_CFG_SOSC_DRIVE
SYSTEM_CFG_SOSC_NF_STOP
Common clock setting
SYSTEM_CFG_CLCOK_SOURCE
SYSTEM_CFG_ICK_PCKA_DIV
SYSTEM_CFG_PCKB_DIV
Clock stop setting
SYSTEM_CFG_MOSC_ENABLE
SYSTEM_CFG_HOCO_ENABLE
SYSTEM_CFG_MOCO_ENABLE
SYSTEM_CFG_LOCO_ENABLE
SYSTEM_CFG_SOSC_ENABLE
SYSTEM_CFG_PLL_ENABLE
Power control mode setting
SYSTEM_CFG_POWER_CONTROL_MODE
Config condition: Select system clock source SYSTEM_CFG_CLCOK_SOURCE
Each clock setting: Calculate frequency in #define definition
User application functions
Driver source file (r_system_api.c)
Mode transition and clock operation switch
System initialization functions
R_SYS_CodeCopy()
R_SYS_Initialize()
Power control setting functions
R_SYS_BoostSpeedModeSet()
R_SYS_HighSpeedModeSet()
R_SYS_LowSpeedModeSet()
R_SYS_32kHzSpeedModeSet()
Clock settings inheriting configuration values (r_core_chg.h)
System clock division ratio setting function
R_SYS_SystemClockDividerSet()
Clock operation start functions
R_SYS_SystemClockHOCOSet()
R_SYS_MainOscSpeedClockStart()
R_SYS_HighSpeedClockStart()
R_SYS_SubOscSpeedClockStart()
R_SYS_PLLSpeedClockStart()
System clock/peripheral module clock acquisition functions
R_SYS_SystemClockFreqGet()
R_SYS_PeripheralClockFreqGet()
System clock switch functions
R_SYS_SystemClockHOCOSet()
R_SYS_SystemClockMOCOSet()
R_SYS_SystemClockLOCOSet()
R_SYS_SystemClockMOSCSet()
R_SYS_SystemClockSOSCSet()
Clock operation stop functions
R_SYS_MainOscSpeedClockStop()
R_SYS_HighSpeedClockStop()
R_SYS_MediumSpeedClockStop()
R_SYS_LowSpeedClockStop()
R_SYS_SubOscSpeedClockStop()
R_SYS_PLLSpeedClockStop()
Configuration definition setting
R_SYSTEM_PRV_HOCO_FREQUENCY_HZ
R_SYSTEM_PRV_PLL_DIV_FREQUENCY
R_SYSTEM_PRV_PLL_FREQUENCY_HZ
R_SYSTEM_PRV_PLL_RANGE
R_SYSTEM_PRV_CLOCK_SEL
R_SYSTEM_PRV_CLOCK_ICK_PCKA
R_SYSTEM_PRV_CLOCK_PCKB
Driver source file (r_system_api.c)
Other functions
- Event link set/clear/get functions (Sec 4.3.30 - 4.3.32)
- Critical section enter/exit functions (4.3.33, 4.3.34)
- Hardware resource lock/unlock functions (4.3.35, 4.3.36)
- Register protection enable/disable functions (4.3.37, 4.3.38)
- Software delay function (4.3.39)
Driver header file (r_system_api.h)
Other functions
- Event link/NVIC IRQ set/clear/get functions (4.3.41 - 4.3.52)

Figure 2.2 Relationship between R_CORE Configuration Settings and R_SYSTEM Driver Functions

3. Internal Operation of Software Components

The R_SYSTEM driver implements mode transitions and clock operation switching. This section shows the procedure for calling the R_SYSTEM driver functions that make mode transitions and select clock operation. For procedures for causing transitions between power supply modes or entry to VBB mode, refer to the specification of the R_LPM driver.


Function calling procedure
Copy code to be executed via RAM from ROM to RAM
Initialize system
For first operation only
Specify power supply mode
Specify power control mode
Specify frequency division ratio for system clock
Start clock operation
Switch system clock
Stop unnecessary operating clock
Switch modes and clocks (can be repeatedly switched)
Each driver operation processing

Copy code only when it is executed via RAM
R_SYS_CodeCopy()

Initialize protection status
R_SYS_Initialize()

Select and execute function for specifying power supply mode using R_LPM driver.
For power supply modes, refer to sec 12, Power-Saving Functions, in UMH.
Note: For details of API function used, refer to R_LPM driver specification.
documentR_LPM_Initialize()
R_LPM_PowerSupplyModeAllpwonSet()
R_LPM_PowerSupplyModeExfpwonSet()
R_LPM_PowerSupplyModeMinpwonSet()
R_LPM_PowerSupplyModeGet()

Select and execute function for specifying power control mode.
For power control modes, refer to sec 12, Power-Saving Functions, in UMH.
R_SYS_BoostSpeedModeSet()
R_SYS_HighSpeedModeSet()
R_SYS_LowSpeedModeSet()
R_SYS_32kHzSpeedModeSet()

Execute function for specifying frequency division ratio for system clock
R_SYS_SystemClockDividerSet()

Select and execute API functions for starting clocks
R_SYS_MainOscSpeedClockStart()
R_SYS_HighSpeedClockStart()
R_SYS_MediumSpeedClockStart()
R_SYS_LowSpeedClockStart()
R_SYS_SubOscSpeedClockStart()
R_SYS_PLLSpeedClockStart()

Execute system clock switch function to select clock source
R_SYS_SystemClockHOCOSet()
R_SYS_SystemClockMOCOSet()
R_SYS_SystemClockLOCOSet()
R_SYS_SystemClockMOSCSet()
R_SYS_SystemClockSOSCSet()
R_SYS_SystemClockPLLSet()

Stop operating clock (unnecessary clock) other than system clock (optional)
R_SYS_MainOscSpeedClockStop()
R_SYS_HighSpeedClockStop()
R_SYS_MediumSpeedClockStop()
R_SYS_LowSpeedClockStop()
R_SYS_SubOscSpeedClockStop()
R_SYS_PLLSpeedClockStop()

Note: For details of API functions used, refer to sec 4.3, Function Specifications, in this driver specification document.

Figure 3.1 Procedure for Calling API Functions Using R_SYSTEM Driver

4. Detailed Information of Software Unit

4.1 Configurations

For the R_SYSTEM driver, configuration definitions that can be modified by the user are provided in the r_system_cfg.h file.

4.1.1 Parameter Checking

This enables or disables the parameter checking in the R_SYSTEM driver.

Name: SYSTEM_CFG_PARAM_CHECKING_ENABLE

Table 4-1 Settings of SYSTEM_CFG_PARAM_CHECKING_ENABLE

Setting	Description
0	Disables the parameter checking. The error conditions described in Function Specifications will not be detected.
1 (initial value)	Enables the parameter checking. The error conditions described in Function Specifications will be detected.

4.1.2 Critical Section

This enables or disables the critical section control in the R_SYSTEM driver.

In reading the value of a register, modifying the value of some bits, and then writing it back to the register, it is necessary to control the critical section so that no interrupt will occur during this process.

Name: SYSTEM_CFG_ENTER_CRITICAL_SECTION_ENABLE

Table 4-2 Settings of SYSTEM_CFG_ENTER_CRITICAL_SECTION_ENABLE

Setting	Description
0	Disables the control of critical sections.
1 (initial value)	Enables the control of critical sections.

4.1.3 Register Protection

This enables or disables the register write protection control in the R_SYSTEM driver.

In writing to a target register, it is necessary to control register write protection.

Name: SYSTEM_CFG_REGISTER_PROTECTION_ENABLE

Table 4-3 Settings of SYSTEM_CFG_REGISTER_PROTECTION_ENABLE

Setting	Description
0	Disables the control of register write protection.
1 (initial value)	Enables the control of register write protection.

4.1.4 Value of API Timeout

This specifies the timeout time when a CMSIS driver API waits for a value to be reflected.

Name: SYSTEM_CFG_API_TIMEOUT_COUNT

Table 4-4 Settings of SYSTEM_CFG_API_TIMEOUT_COUNT

Setting	Description
268,435,456 (0x10000000)	This specifies the timeout time when a CMSIS driver API waits for a value to be reflected.

4.1.5 Event Link Number Setting

The interrupt handler of each event link number specified here is called as a callback function.

For the event signal linked with this setting, refer to UMH.

Name: SYSTEM_CFG_EVENT_NUMBER_****_****

Table 4-5 Settings of SYSTEM_CFG_EVENT_NUMBER_****_****

Setting	Description
0x00 (initial value) SYSTEM_IRQ_EVENT_NUMBER_NOT_USED	Disables an event output to the specified peripheral module.
0x01-0xAB SYSTEM_IRQ_EVENT_NUMBERn (n=0-31)	Specifies the number of the event signal to be linked to.

Table 4-6 Event Number Settings of SYSTEM_CFG_EVENT_NUMBER_****_****

Event Number	Source of Interrupt Request	Name	Configuration Definition Name of Event (r_system_cfg.h)
01h	Port	PORT_IRQ0	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ0
02h		PORT_IRQ1	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ1
03h		PORT_IRQ2	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ2
04h		PORT_IRQ3	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ3
05h		PORT_IRQ4	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ4
06h		PORT_IRQ5	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ5
07h		PORT_IRQ6	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ6
08h		PORT_IRQ7	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ7
09h	DMAC0	DMAC0_INT	SYSTEM_CFG_EVENT_NUMBER_DMAC0_INT
0Ah	DMAC1	DMAC1_INT	SYSTEM_CFG_EVENT_NUMBER_DMAC1_INT
0Bh	DMAC2	DMAC2_INT	SYSTEM_CFG_EVENT_NUMBER_DMAC2_INT
0Ch	DMAC3	DMAC3_INT	SYSTEM_CFG_EVENT_NUMBER_DMAC3_INT
0Dh	DTC	DTC_COMPLETE	SYSTEM_CFG_EVENT_NUMBER_DTC_COMPLETE
0Fh	ICU	ICU_SNZCANCEL	SYSTEM_CFG_EVENT_NUMBER_ICU_SNZCANCEL
10h	FCU	FCU_FIFERR	SYSTEM_CFG_EVENT_NUMBER_FCU_FIFERR
11h		FCU_FRDYI	SYSTEM_CFG_EVENT_NUMBER_FCU_FRDYI
12h	LVD	LVD_LVD1	SYSTEM_CFG_EVENT_NUMBER_LVD_LVD1
13h		LVD_LVDBAT	SYSTEM_CFG_EVENT_NUMBER_LVD_LVDBAT
14h	MOSC	MOSC_STOP	SYSTEM_CFG_EVENT_NUMBER_MOSC_STOP
15h	Low power consumption mode	SYSTEM_SNZREQ	SYSTEM_CFG_EVENT_NUMBER_SYSTEM_SNZREQ
16h	EHC	SOL_DH	SYSTEM_CFG_EVENT_NUMBER_SOL_DH

17h		SOL_DL	SYSTEM_CFG_EVENT_NUMBER_SOL_DL
18h	AGT0	AGT0_AGTI	SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTI
1Ah		AGT0_AGTCMBI	SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTCMBI
1Bh	AGT1	AGT1_AGTI	SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTI
1Ch		AGT1_AGTCMAI	SYSTEM_CFG_EVENT_NUMBER_AGT1_AGTCMAI
1Dh	AGT0	AGT0_AGTCMAI	SYSTEM_CFG_EVENT_NUMBER_AGT0_AGTCMAI
1Eh	IWDT	IWDT_NMIUNDF	SYSTEM_CFG_EVENT_NUMBER_IWDT_NMIUNDF
1Fh	WDT	WDT_NMIUNDF	SYSTEM_CFG_EVENT_NUMBER_WDT_NMIUNDF
20h	RTC	RTC_ALM	SYSTEM_CFG_EVENT_NUMBER_RTC_ALM
21h		RTC_PRD	SYSTEM_CFG_EVENT_NUMBER_RTC_PRD
22h		RTC_CUP	SYSTEM_CFG_EVENT_NUMBER_RTC_CUP
23h	S14AD	ADC140_ADI	SYSTEM_CFG_EVENT_NUMBER_ADC140_ADI
24h		ADC140_GBADI	SYSTEM_CFG_EVENT_NUMBER_ADC140_GBADI
25h		ADC140_CMPAI	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPAI
26h		ADC140_CMPBI	SYSTEM_CFG_EVENT_NUMBER_ADC140_CMPBI
27h		ADC140_WCMPM	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPM
28h		ADC140_WCMPUM	SYSTEM_CFG_EVENT_NUMBER_ADC140_WCMPUM
29h		ADC140_GCADI	SYSTEM_CFG_EVENT_NUMBER_ADC140_GCADI
2Ah	ACMP	ACMP_CMPI	SYSTEM_CFG_EVENT_NUMBER_ACMP_CMPI
2Bh	USB	USBFS_D0FIFO	SYSTEM_CFG_EVENT_NUMBER_USBFS_D0FIFO
2Ch		USBFS_D1FIFO	SYSTEM_CFG_EVENT_NUMBER_USBFS_D1FIFO
2Dh		USBFS_USBI	SYSTEM_CFG_EVENT_NUMBER_USBFS_USBI
2Eh		USBFS_USBR	SYSTEM_CFG_EVENT_NUMBER_USBFS_USBR
2Fh	RIIC0	IIC0_RXI	SYSTEM_CFG_EVENT_NUMBER_IIC0_RXI
30h		IIC0_TXI	SYSTEM_CFG_EVENT_NUMBER_IIC0_TXI
31h		IIC0_TEI	SYSTEM_CFG_EVENT_NUMBER_IIC0_TEI
32h		IIC0_EEI	SYSTEM_CFG_EVENT_NUMBER_IIC0_EEI
33h	RIIC1	IIC1_RXI	SYSTEM_CFG_EVENT_NUMBER_IIC1_RXI
34h		IIC1_TXI	SYSTEM_CFG_EVENT_NUMBER_IIC1_TXI
35h		IIC1_TEI	SYSTEM_CFG_EVENT_NUMBER_IIC1_TEI
36h		IIC1_EEI	SYSTEM_CFG_EVENT_NUMBER_IIC1_EEI
37h	KINT	KEY_INTKR	SYSTEM_CFG_EVENT_NUMBER_KEY_INTKR
38h	DOC	DOC_DOPCI	SYSTEM_CFG_EVENT_NUMBER_DOC_DOPCI
39h	CAC	CAC_FEERI	SYSTEM_CFG_EVENT_NUMBER_CAC_FEERI
3Ah		CAC_MENDI	SYSTEM_CFG_EVENT_NUMBER_CAC_MENDI
3Bh		CAC_OVFI	SYSTEM_CFG_EVENT_NUMBER_CAC_OVFI
3Ch	I/O port	IOPORT_GROUP3	SYSTEM_CFG_EVENT_NUMBER_IOPORT_GROUP3
3Dh		IOPORT_GROUP2	SYSTEM_CFG_EVENT_NUMBER_IOPORT_GROUP2
3Eh	ELC	ELC_SWEVT0	SYSTEM_CFG_EVENT_NUMBER_ELC_SWEVT0
3Fh		ELC_SWEVT1	SYSTEM_CFG_EVENT_NUMBER_ELC_SWEVT1
40h	POE	POEG_GROUPA	SYSTEM_CFG_EVENT_NUMBER_POEG_GROUPA
41h		POEG_GROUPB	SYSTEM_CFG_EVENT_NUMBER_POEG_GROUPB
42h	TMR	TMR_CMIA0	SYSTEM_CFG_EVENT_NUMBER_TMR_CMIA0
43h		TMR_CMIB0	SYSTEM_CFG_EVENT_NUMBER_TMR_CMIB0
44h		TMR_OVF0	SYSTEM_CFG_EVENT_NUMBER_TMR_OVF0
45h		TMR_CMIA1	SYSTEM_CFG_EVENT_NUMBER_TMR_CMIA1
46h		TMR_CMIB1	SYSTEM_CFG_EVENT_NUMBER_TMR_CMIB1
47h		TMR_OVF1	SYSTEM_CFG_EVENT_NUMBER_TMR_OVF1
48h	CCC	CCC_PRD	SYSTEM_CFG_EVENT_NUMBER_CCC_PRD
49h		CCC_CUP	SYSTEM_CFG_EVENT_NUMBER_CCC_CUP
4Ah	LPG	CCC_ERR	SYSTEM_CFG_EVENT_NUMBER_CCC_ERR
4Bh	MTDV	MTDV_PM1INT	SYSTEM_CFG_EVENT_NUMBER_MTDV_PM1INT

4Ch		MTDV_PM25INT	SYSTEM_CFG_EVENT_NUMBER_MTDV_PM25INT
4Dh		MTDV_PM36INT	SYSTEM_CFG_EVENT_NUMBER_MTDV_PM36INT
4Eh	ELC	ELC_INT0	SYSTEM_CFG_EVENT_NUMBER_ELC_INT0
4Fh		ELC_INT1	SYSTEM_CFG_EVENT_NUMBER_ELC_INT1
50h	GPT320	GPT0_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT0_CCMPA
51h		GPT0_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT0_CCMPB
52h		GPT0_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT0_CMPC
53h		GPT0_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT0_CMPD
54h		GPT0_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT0_OVF
55h		GPT0_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT0_UDF
56h	GPT321	GPT1_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT1_CCMPA
57h		GPT1_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT1_CCMPB
58h		GPT1_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT1_CMPC
59h		GPT1_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT1_CMPD
5Ah		GPT1_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT1_OVF
5Bh		GPT1_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT1_UDF
5Ch	GPT162	GPT2_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT2_CCMPA
5Dh		GPT2_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT2_CCMPB
5Eh		GPT2_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT2_CMPC
5Fh		GPT2_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT2_CMPD
60h		GPT2_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT2_OVF
61h		GPT2_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT2_UDF
62h	GPT163	GPT3_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT3_CCMPA
63h		GPT3_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT3_CCMPB
64h		GPT3_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT3_CMPC
65h		GPT3_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT3_CMPD
66h		GPT3_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT3_OVF
67h		GPT3_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT3_UDF
68h	GPT164	GPT4_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT4_CCMPA
69h		GPT4_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT4_CCMPB
6Ah		GPT4_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT4_CMPC
6Bh		GPT4_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT4_CMPD
6Ch		GPT4_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT4_OVF
6Dh		GPT4_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT4_UDF
6Eh	GPT165	GPT5_CCMPA	SYSTEM_CFG_EVENT_NUMBER_GPT5_CCMPA
6Fh		GPT5_CCMPB	SYSTEM_CFG_EVENT_NUMBER_GPT5_CCMPB
70h		GPT5_CMPC	SYSTEM_CFG_EVENT_NUMBER_GPT5_CMPC
71h		GPT5_CMPD	SYSTEM_CFG_EVENT_NUMBER_GPT5_CMPD
72h		GPT5_OVF	SYSTEM_CFG_EVENT_NUMBER_GPT5_OVF
73h		GPT5_UDF	SYSTEM_CFG_EVENT_NUMBER_GPT5_UDF
74h	GPT	GPT_UVWEDGE	SYSTEM_CFG_EVENT_NUMBER_GPT_UVWEDGE
75h	SCI0	SCI0_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI0_RXI
76h		SCI0_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI0_TXI
77h		SCI0_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI0_TEI
78h		SCI0_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI0_ERI
79h		SCI0_AM	SYSTEM_CFG_EVENT_NUMBER_SCI0_AM
7Ah		SCI0_RXI_OR_ERI	Unused
7Bh	SCI1	SCI1_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI1_RXI
7Ch		SCI1_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI1_TXI
7Dh		SCI1_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI1_TEI
7Eh		SCI1_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI1_ERI
7Fh		SCI1_AM	SYSTEM_CFG_EVENT_NUMBER_SCI1_AM

80h	SCI2	SCI2_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI2_RXI
81h		SCI2_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI2_TXI
82h		SCI2_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI2_TEI
83h		SCI2_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI2_ERI
84h		SCI2_AM	SYSTEM_CFG_EVENT_NUMBER_SCI2_AM
85h	SCI3	SCI3_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI3_RXI
86h		SCI3_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI3_TXI
87h		SCI3_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI3_TEI
88h		SCI3_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI3_ERI
89h		SCI3_AM	SYSTEM_CFG_EVENT_NUMBER_SCI3_AM
8Ah	SCI4	SCI4_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI4_RXI
8Bh		SCI4_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI4_TXI
8Ch		SCI4_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI4_TEI
8Dh		SCI4_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI4_ERI
8Eh		SCI4_AM	SYSTEM_CFG_EVENT_NUMBER_SCI4_AM
8Fh	SCI5	SCI5_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI5_RXI
90h		SCI5_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI5_TXI
91h		SCI5_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI5_TEI
92h		SCI5_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI5_ERI
93h		SCI5_AM	SYSTEM_CFG_EVENT_NUMBER_SCI5_AM
94h	SCI9	SCI9_RXI	SYSTEM_CFG_EVENT_NUMBER_SCI9_RXI
95h		SCI9_TXI	SYSTEM_CFG_EVENT_NUMBER_SCI9_TXI
96h		SCI9_TEI	SYSTEM_CFG_EVENT_NUMBER_SCI9_TEI
97h		SCI9_ERI	SYSTEM_CFG_EVENT_NUMBER_SCI9_ERI
98h		SCI9_AM	SYSTEM_CFG_EVENT_NUMBER_SCI9_AM
99h	SPI0	SPI0_SPRI	SYSTEM_CFG_EVENT_NUMBER_SPI0_SPRI
9Ah		SPI0_SPTI	SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTI
9Bh		SPI0_SPII	SYSTEM_CFG_EVENT_NUMBER_SPI0_SPII
9Ch		SPI0_SPEI	SYSTEM_CFG_EVENT_NUMBER_SPI0_SPEI
9Dh		SPI0_SPTEND	SYSTEM_CFG_EVENT_NUMBER_SPI0_SPTEND
9Eh	SPI1	SPI1_SPRI	SYSTEM_CFG_EVENT_NUMBER_SPI1_SPRI
9Fh		SPI1_SPTI	SYSTEM_CFG_EVENT_NUMBER_SPI1_SPTI
A0h		SPI1_SPII	SYSTEM_CFG_EVENT_NUMBER_SPI1_SPII
A1h		SPI1_SPEI	SYSTEM_CFG_EVENT_NUMBER_SPI1_SPEI
A2h		SPI1_SPTEND	SYSTEM_CFG_EVENT_NUMBER_SPI1_SPTEND
A3h	QSPI	QSPI_INTR	SYSTEM_CFG_EVENT_NUMBER_QSPI_INTR
A4h	DIV	DIV_CALCCOMP	SYSTEM_CFG_EVENT_NUMBER_DIV_CALCCOMP
A6h	MLCD	MLCD_TEI	SYSTEM_CFG_EVENT_NUMBER_MLCD_TEI
A7h		MLCD_TEMI	SYSTEM_CFG_EVENT_NUMBER_MLCD_TEMI
A8h	GDT	GDT_DATII	SYSTEM_CFG_EVENT_NUMBER_GDT_DATOI
A9h		GDT_DATOI	SYSTEM_CFG_EVENT_NUMBER_GDT_FDCENDI
AAh		GDT_FDCENDI	SYSTEM_CFG_EVENT_NUMBER_GDT_DATII
B4h	Port	PORT_IRQ8	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ8
B5h		PORT_IRQ9	SYSTEM_CFG_EVENT_NUMBER_PORT_IRQ9

4.1.6 Function Allocation to RAM

This makes the settings for executing specific functions of the R_SYSTEM driver from the RAM.

Programs to be executed while the power supply to the flash memory is switched off need to be allocated to RAM and executed from the RAM.

This configuration definition for setting function allocation to RAM has function-specific definitions.

Name: SYSTEM_CFG_SECTION_R_SYS_xxxxx

SYSTEM_CFG_SECTION_IELn_IRQHANDLER (n = 0 to 31)

An API name xxxxx should be written in all capital letters.

Example: R_SYS_Initialize function → SYSTEM_CFG_SECTION_R_SYS_INITIALIZE

Table 4-7 Settings of SYSTEM_CFG_SECTION_xxxxx

Setting	Description
SYSTEM_SECTION_CODE	Does not allocate the function to RAM.
SYSTEM_SECTION_RAM_FUNC	Allocates the function to RAM.

Table 4-8 Initial State of Function Allocation to RAM

No.	Function Name	Allocation to RAM
1	R_SYS_Initialize	
2	R_SYS_BoostSpeedModeSet	
3	R_SYS_HighSpeedModeSet	✓
4	R_SYS_LowSpeedModeSet	✓
5	R_SYS_32kHzSpeedModeSet	✓
6	R_SYS_SpeedModeGet	✓
7	R_SYS_SystemClockHOCOSet	✓
8	R_SYS_SystemClockMOCOSet	✓
9	R_SYS_SystemClockLOCOSet	✓
10	R_SYS_SystemClockMOSCSet	✓
11	R_SYS_SystemClockSOSCSet	✓
12	R_SYS_SystemClockPLLSet	
13	R_SYS_SystemClockFreqGet	✓
14	R_SYS_PeripheralClockFreqGet	✓
15	R_SYS_SystemClockDividerSet	✓
16	R_SYS_MainOscSpeedClockStart	✓
17	R_SYS_MainOscSpeedClockStop	✓
18	R_SYS_HighSpeedClockStart	✓
19	R_SYS_HighSpeedClockStop	✓
20	R_SYS_MediumSpeedClockStart	✓
21	R_SYS_MediumSpeedClockStop	✓
22	R_SYS_LowSpeedClockStart	✓
23	R_SYS_LowSpeedClockStop	✓
24	R_SYS_SubOscSpeedClockStart	✓
25	R_SYS_SubOscSpeedClockStop	✓
26	R_SYS_PLLSpeedClockStart	
27	R_SYS_PLLSpeedClockStop	
28	R_SYS_OscStabilizationFlagGet	✓

29	R_SYS_IrqEventLinkSet	✓
30	R_SYS_IrqStatusGet	✓
31	R_SYS_IrqStatusClear	✓
32	R_SYS_EnterCriticalSection	✓
33	R_SYS_ExitCriticalSection	✓
34	R_SYS_ResourceLock	✓
35	R_SYS_ResourceUnlock	✓
36	R_SYS_RegisterProtectEnable	✓
37	R_SYS_RegisterProtectDisable	✓
38	R_SYS_SoftwareDelay	✓
39 to 70	IELn_IRQHandler (n = 0 to 31)	✓
71	R_SYS_GetVersion	

4.2 Macro and Type Definitions

For the R_SYSTEM driver, macro and type definitions that can be referenced by the user are provided in the r_system_api.h file.

Table 4-9 Macro Definition List

Macro Definition	Setting	Remarks
R_SYSTEM_PRV_PRCR_KEY	(0xA500U)	Releases PRCR register protection.
R_SYSTEM_PRV_IRQ_EVENT_NUMBER_TOTAL	(32)	Total number of interrupts of IRQ event links: 32 interrupts
R_SYSTEM_PRV_LOCK_LOCKED	(0x01)	Lock value of Valid st_system_lock_t: 1
R_SYSTEM_PRV_LOCK_UNLOCKED	(0x00)	Unlock value of Valid st_system_lock_t: 0
R_SYSTEM_PRV_IELSR_IR_MSK	(0x00010000)	Mask value for IR interrupt status flag in ICU->IELSR register
R_SYSTEM_PRV_IELSR_IELS_MSK	(0x0000001F)	Mask value for IELS in ICU->IELSR register
R_SYSTEM_PRV_OSCSF_HOCOSF_MSK	(0x01)	Mask value for HOCO clock oscillation stabilization flag
R_SYSTEM_PRV_OSCSF_MOSCSF_MSK	(0x08)	Mask value for main clock oscillation stabilization flag
R_SYSTEM_PRV_OSCSF_PLLSF_MSK	(0x20)	Mask value for PLL clock oscillation stabilization flag
R_SYSTEM_PRV_SCKSCR_CKSEL_MSK	(0x07)	Mask value for clock source selection
R_SYSTEM_PRV_SCKSCR_CKSEL_HOCO	(0x00)	Selects HOCO for the clock source.
R_SYSTEM_PRV_SCKSCR_CKSEL_MOCO	(0x01)	Selects MOCO for the clock source.
R_SYSTEM_PRV_SCKSCR_CKSEL_LOCO	(0x02)	Selects LOCO for the clock source.
R_SYSTEM_PRV_SCKSCR_CKSEL_MOSC	(0x03)	Selects the main clock for the clock source.
R_SYSTEM_PRV_SCKSCR_CKSEL_SOSC	(0x04)	Selects the sub-clock for the clock source.
R_SYSTEM_PRV_SCKSCR_CKSEL_PLL	(0x05)	Selects the PLL for the clock source.

R_SYSTEM_PRV_HOCO_FREQUENCY_HZ	(24000000U)	Set to 24 MHz when SYSTEM_CFG_HOCO_FREQUENCY = 0.
	(32000000U)	Set to 32 MHz when SYSTEM_CFG_HOCO_FREQUENCY = 1.
	(48000000U)	Set to 48 MHz when SYSTEM_CFG_HOCO_FREQUENCY = 2.
	(64000000U)	Set to 64 MHz when SYSTEM_CFG_HOCO_FREQUENCY = 3.
R_SYSTEM_PRV_MOCO_FREQUENCY_HZ	(2000000U)	Set to 2 MHz when MOCO is selected.
R_SYSTEM_PRV_LOCO_FREQUENCY_HZ	(32768U)	Set to 32.768 kHz when LOCO is selected.
R_SYSTEM_PRV_SUBCLOCK_FREQUENCY_HZ	(32768U)	Set to 32.768 kHz when the sub-clock is selected.
R_SYSTEM_PRV_PLL_DIV_FREQUENCY	(SYSTEM_CFG_MOSC_FREQUENCY_HZ / (SYSTEM_CFG_PLL_DIV+1))	Specifies the frequency to be input to the PLL circuit in accordance with the setting of the main clock frequency and the division ratio of the PLL frequency.
R_SYSTEM_PRV_PLL_FREQUENCY_HZ	(R_SYSTEM_PRV_PLL_DIV_FREQUENCY*(SYSTEM_CFG_PLL_MUL+1))	Specifies the frequency to be generated by PLL operation in accordance with the frequency input to the PLL circuit and the multiplication ratio of the PLL frequency.
R_SYSTEM_PRV_PLL_RANGE	(1)	Specify this value when 48000000 < output frequency of PLL circuit ≤ 64000000.
	(0)	Specify this value when 32000000 ≤ output frequency of PLL circuit ≤ 48000000.
R_SYSTEM_PRV_CLOCK_SEL	(R_SYSTEM_PRV_HOCO_FREQUENCY_HZ)	Specifies the frequency selected by R_SYSTEM_PRV_HOCO_FREQUENCY_HZ when HOCO is selected.
	(R_SYSTEM_PRV_MOCO_FREQUENCY_HZ)	Specifies the frequency selected by R_SYSTEM_PRV_MOCO_FREQUENCY_HZ when MOCO is selected.
	(R_SYSTEM_PRV_LOCO_FREQUENCY_HZ)	Specifies the frequency selected by R_SYSTEM_PRV_LOCO_FREQUENCY_HZ when a LOCO is selected.
	(SYSTEM_CFG_MOSC_FREQUENCY_HZ)	Specifies the frequency selected by SYSTEM_CFG_MOSC_FREQUENCY_HZ when the main clock is selected.
	(R_SYSTEM_PRV_SUBCLOCK_FREQUENCY_HZ)	Specifies the frequency selected by R_SYSTEM_PRV_SUBCLOCK_FREQUENCY_HZ when the sub-clock is selected.
	(R_SYSTEM_PRV_PLL_FREQUENCY_HZ)	Specifies the frequency selected by R_SYSTEM_PRV_PLL_FREQUENCY_HZ when the PLL is selected.
R_SYSTEM_PRV_CLOCK_ICK_PCKA	(R_SYSTEM_PRV_CLOCK_SEL / (1 << SYSTEM_CFG_ICK_PCKA_DIV))	Specifies the frequency obtained by dividing each determined clock by the frequency division ratio SYSTEM_CFG_ICK_PCKA_DIV.
R_SYSTEM_PRV_CLOCK_PCKB	(R_SYSTEM_PRV_CLOCK_SEL / (1 << SYSTEM_CFG_PCKB_DIV))	Specifies the frequency obtained by dividing each determined clock by the frequency division ratio SYSTEM_CFG_PCKB_DIV.
R_SYSTEM_PRV_DELAY_LOOP_CYCLES	(4)	Specifies the number of delay cycles: 4 cycles

4.3 Function Specifications

The specifications and processing flow of each function of the R_SYSTEM driver are described in this section.

The function specification tables in this section are equivalent to the descriptions in Doxygen.

For error checking in the processing flow, only the error conditions are listed and the specific checking method is omitted.

Conditional branch descriptions in the processing flow include the register names and variable names to clarify what are used for judgment on conditions. However, the judgment is not always made as that described in the processing flow.

4.3.1 R_SYS_CodeCopy Function

Table 4-10 R_SYS_CodeCopy Function Specifications

Format	void R_SYS_CodeCopy(void)
Description	Expands the data and programs stored in the specified addresses in ROM to the specified addresses in RAM.
Argument	None
Return value	None
Remarks	–

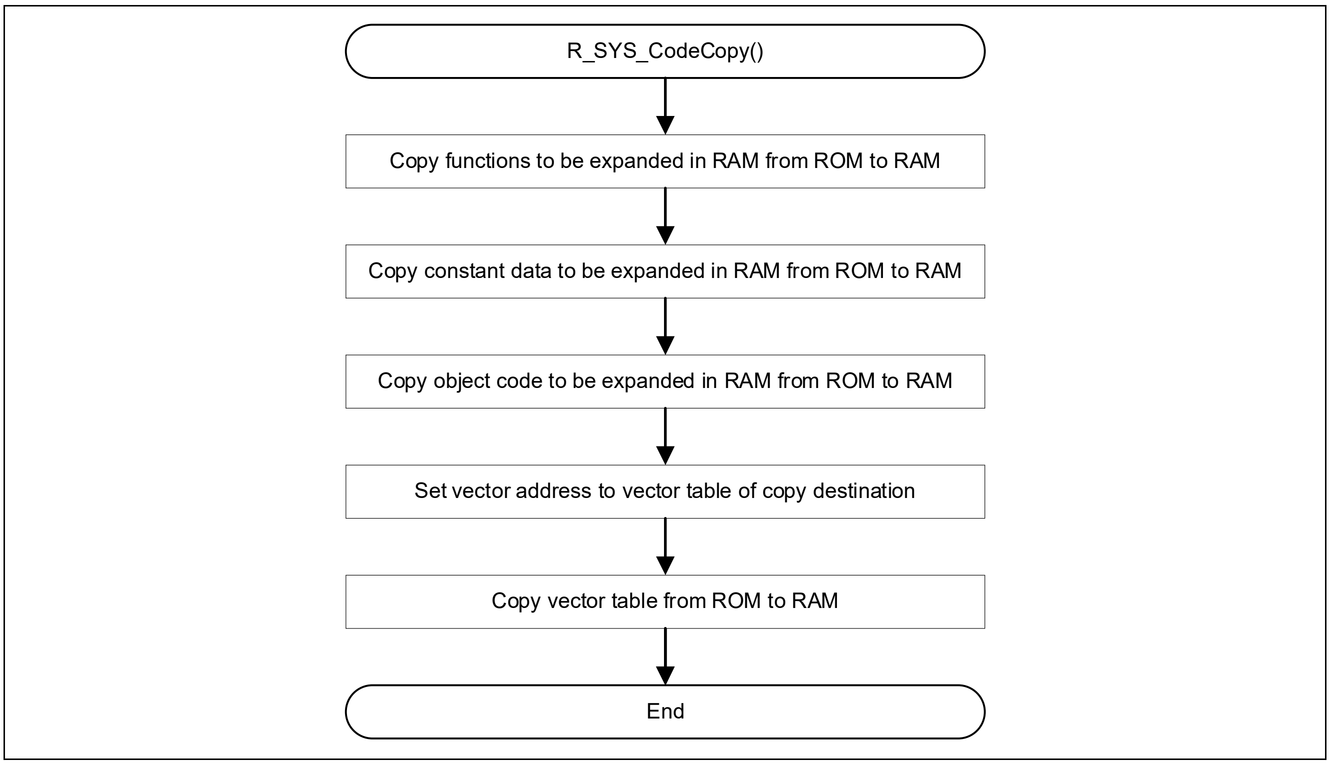

R_SYS_CodeCopy()
Copy functions to be expanded in RAM from ROM to RAM
Copy constant data to be expanded in RAM from ROM to RAM
Copy object code to be expanded in RAM from ROM to RAM
Set vector address to vector table of copy destination
Copy vector table from ROM to RAM
End

Figure 4.1 R_SYS_CodeCopy Function Processing Flow

4.3.2 R_SYS_Initialize Function

Table 4-11 R_SYS_Initialize Function Specifications

Format	void R_SYS_Initialize(void)
Description	Initializes the RAM (callback functions, resource lock status, and register protection status).
Argument	None
Return value	None
Remarks	–

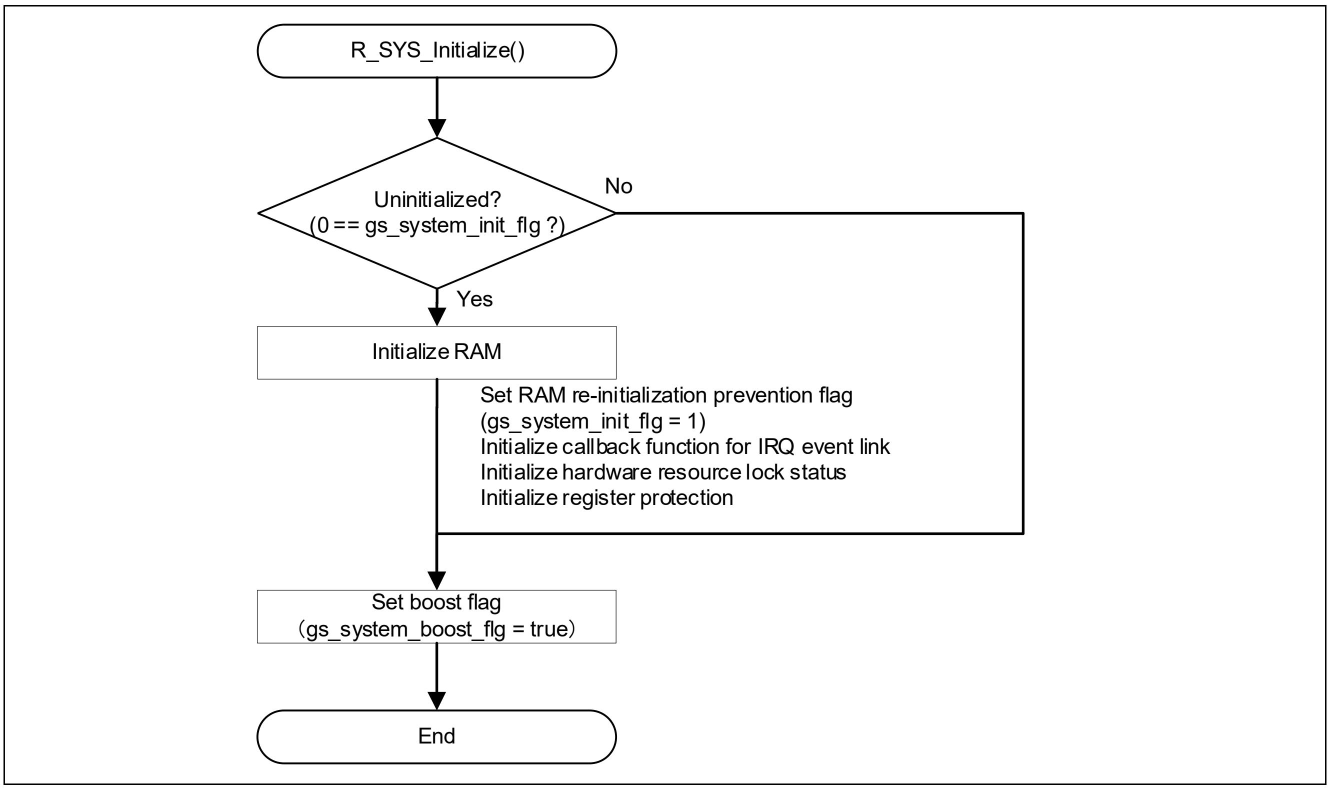

R_SYS_Initialize()
Uninitialized?
(0 == gs_system_init_flg ?)
No
Yes
Initialize RAM
Set RAM re-initialization prevention flag
(gs_system_init_flg = 1)
Initialize callback function for IRQ event link
Initialize hardware resource lock status
Initialize register protection
Set boost flag
(gs_system_boost_flg = true)
End

Figure 4.2 R_SYS_Initialize Function Processing Flow

4.3.3 R_SYS_BoostSpeedModeSet Function

Table 4-12 R_SYS_BoostSpeedModeSet Function Specifications

Format	int32_t R_SYS_BoostSpeedModeSet(void)
Description	Sets the power control mode to boost mode.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

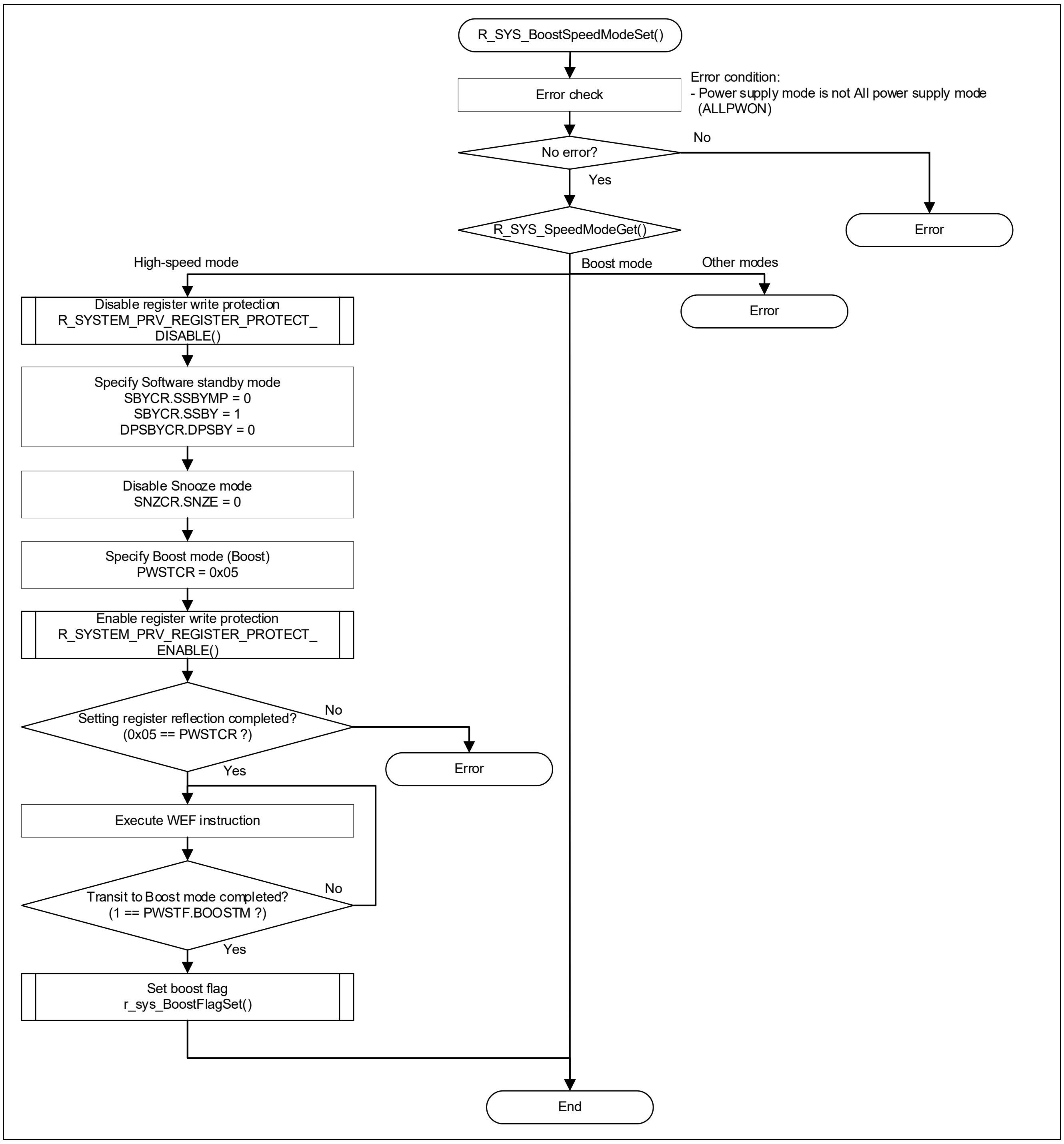

R_SYS_BoostSpeedModeSet()
Error check
Error condition:
- Power supply mode is not All power supply mode (ALLPWON)
No error?
No
Yes
Error
R_SYS_SpeedModeGet()
High-speed mode
Boost mode
Other modes
Error
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Specify Software standby mode SBYCR.SSBYMP = 0 SBYCR.SSBY = 1 DPSBYCR.DPSBY = 0
Disable Snooze mode SNZCR.SNZE = 0
Specify Boost mode (Boost) PWSTCR = 0x05
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x05 == PWSTCR ?)
No
Error
Yes
Execute WEF instruction
Transit to Boost mode completed? (1 == PWSTF.BOOSTM ?)
No
Yes
Set boost flag r_sys_BoostFlagSet()
End

Figure 4.3 R_SYS_BoostSpeedModeSet Function Processing Flow

4.3.4 R_SYS_HighSpeedModeSet Function

Table 4-13 R_SYS_HighSpeedModeSet Function Specifications

Format	int32_t R_SYS_HighSpeedModeSet(void)
Description	Sets the power control mode to high-speed mode.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

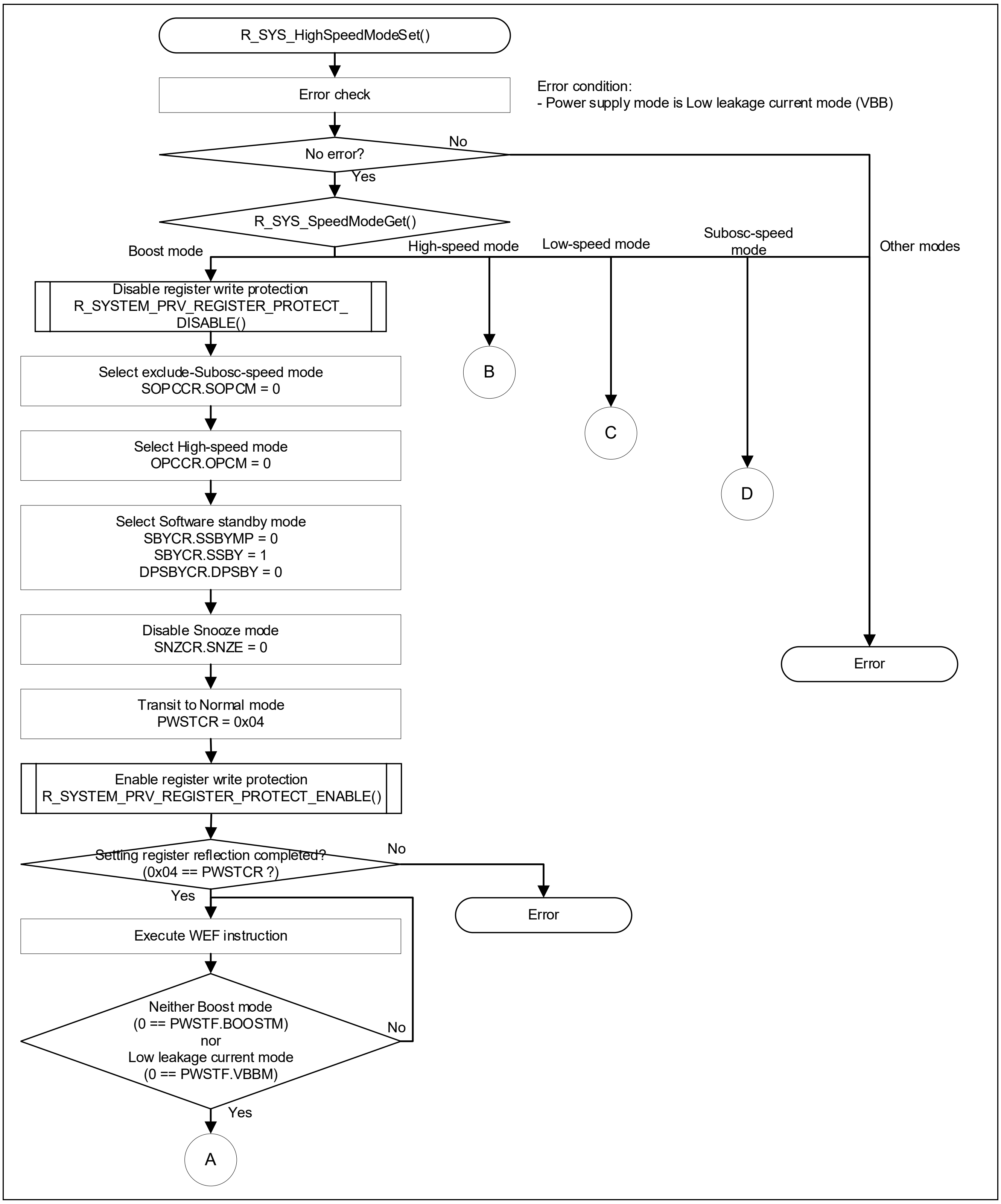

R_SYS_HighSpeedModeSet()
Error check
Error condition:
- Power supply mode is Low leakage current mode (VBB)
No error?
No
Yes
R_SYS_SpeedModeGet()
Boost mode
High-speed mode
Low-speed mode
Subosc-speed mode
Other modes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Select exclude-Subosc-speed mode SOPCCR.SOPCM = 0
Select High-speed mode OPCCR.OPCM = 0
Select Software standby mode SBYCR.SSBYMP = 0 SBYCR.SSBY = 1 DPSBYCR.DPSBY = 0
Disable Snooze mode SNZCR.SNZE = 0
Transit to Normal mode PWSTCR = 0x04
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x04 == PWSTCR ?)
No
Yes
Error
Execute WEF instruction
Neither Boost mode (0 == PWSTF.BOOSTM) nor Low leakage current mode (0 == PWSTF.VBBM)
No
Yes
A
B
C
D
Error

Figure 4.4 R_SYS_HighSpeedModeSet Function Processing Flow (1/2)

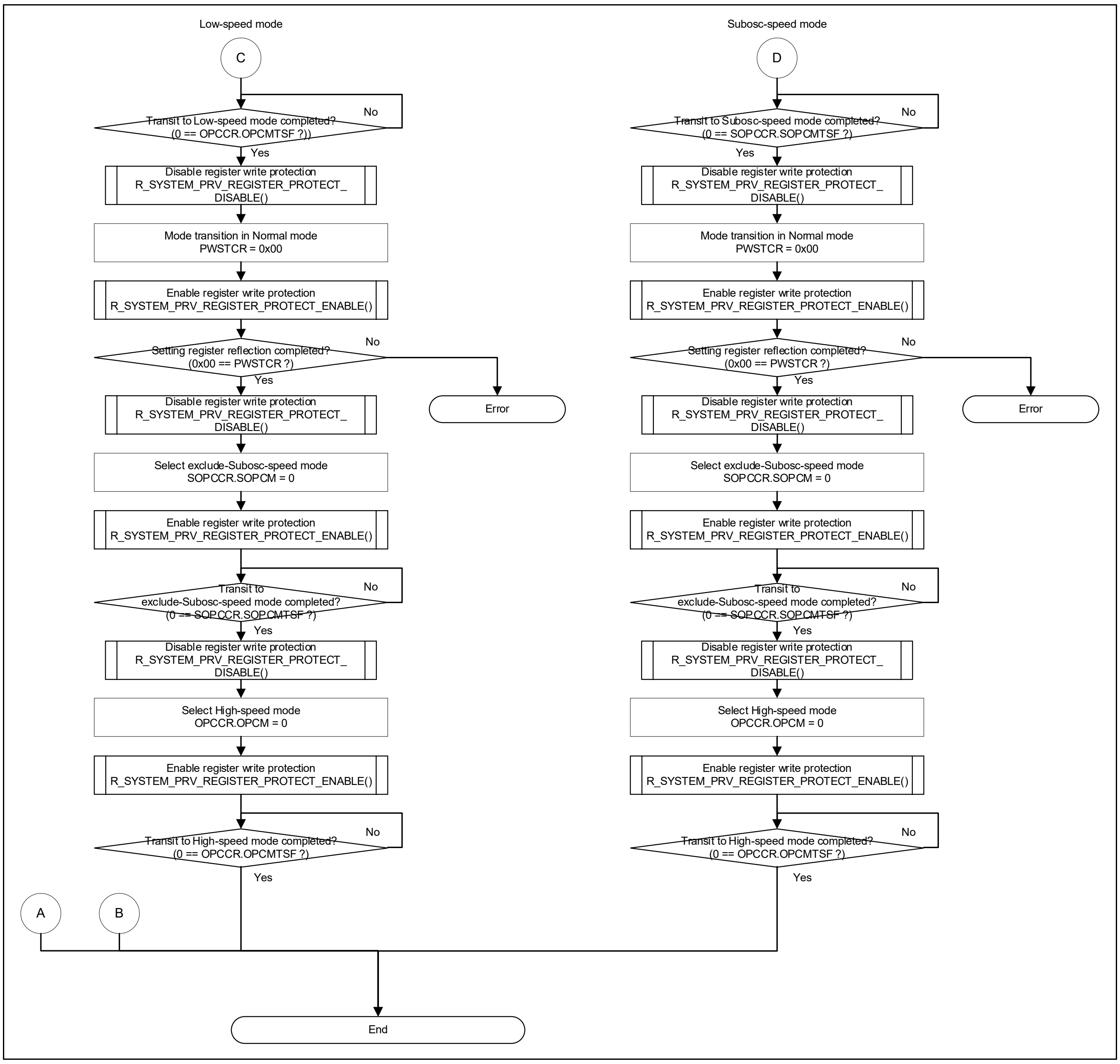

Low-speed mode
C
Transit to Low-speed mode completed? (0 == OPCCR.OPCMTSF ?))
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Mode transition in Normal mode PWSTCR = 0x00
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x00 == PWSTCR ?)
Error
Select exclude-Subosc-speed mode SOPCCR.SOPCM = 0
Transit to exclude-Subosc-speed mode completed? (0 == SOPCCR.SOPCMTSF ?)
Select High-speed mode OPCCR.OPCM = 0
Transit to High-speed mode completed? (0 == OPCCR.OPCMTSF ?)
Subosc-speed mode
D
Transit to Subosc-speed mode completed? (0 == SOPCCR.SOPCMTSF ?)
A
B
End

Figure 4.5 R_SYS_HighSpeedModeSet Function Processing Flow (2/2)

4.3.5 R_SYS_LowSpeedModeSet Function

Table 4-14 R_SYS_LowSpeedModeSet Function Specifications

Format	int32_t R_SYS_LowSpeedModeSet(void)
Description	Sets the power control mode to low-speed mode.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

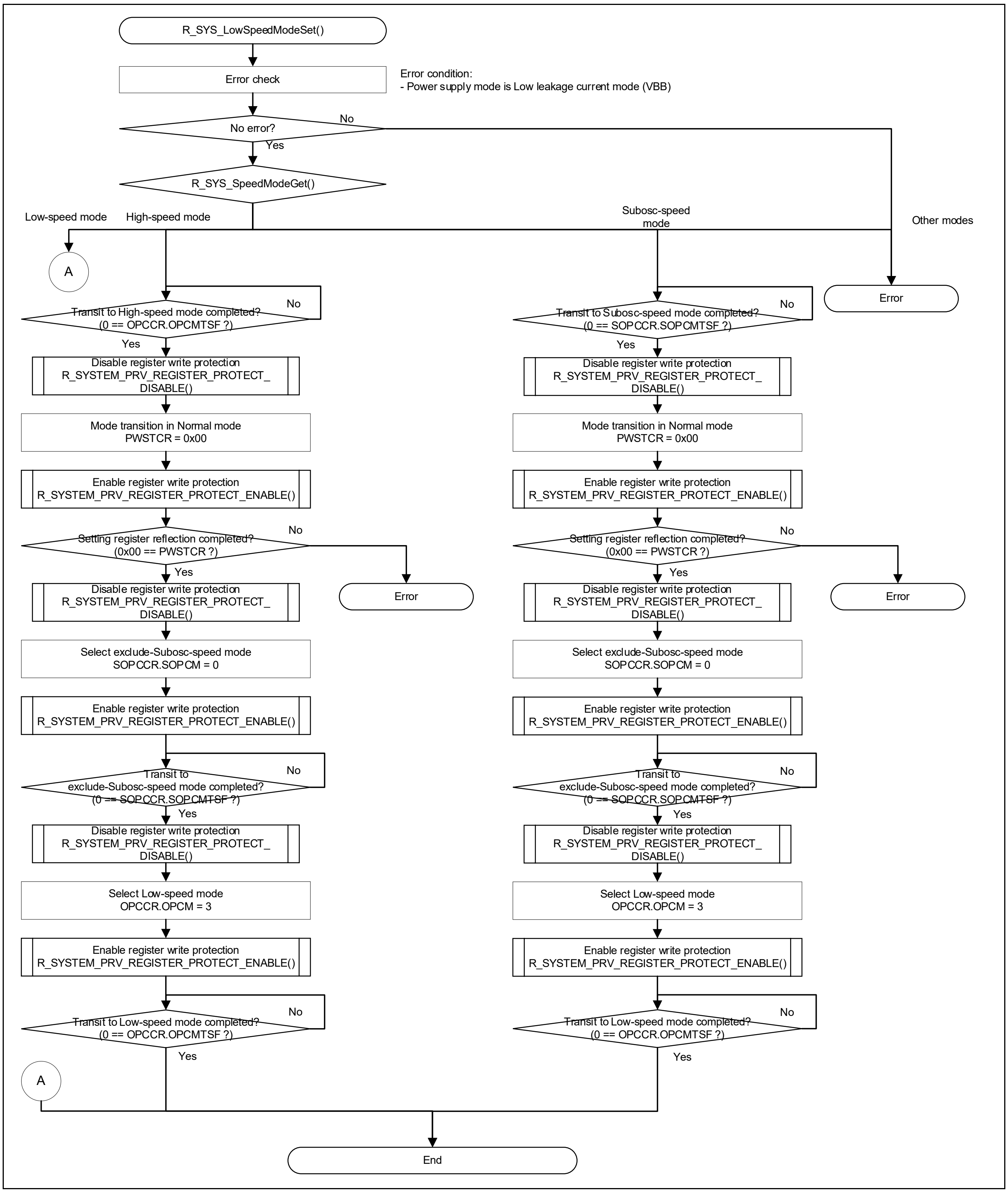

R_SYS_LowSpeedModeSet()
Error check
Error condition:
- Power supply mode is Low leakage current mode (VBB)
No error?
No
Yes
R_SYS_SpeedModeGet()
Low-speed mode
High-speed mode
Subosc-speed mode
Other modes
A
Error
Transit to High-speed mode completed? (0 == OPCCR.OPCMTSF ?)
Transit to Subosc-speed mode completed? (0 == SOPCCR.SOPCMTSF ?)
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Mode transition in Normal mode PWSTCR = 0x00
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x00 == PWSTCR ?)
Select exclude-Subosc-speed mode SOPCCR.SOPCM = 0
Transit to exclude-Subosc-speed mode completed? (0 == SOPCCR.SOPCMTSF ?)
Select Low-speed mode OPCCR.OPCM = 3
Transit to Low-speed mode completed? (0 == OPCCR.OPCMTSF ?)
End

Figure 4.6 R_SYS_LowSpeedModeSet Function Processing Flow

4.3.6 R_SYS_32kHzSpeedModeSet Function

Table 4-15 R_SYS_32kHzSpeedModeSet Function Specifications

Format	int32_t R_SYS_32kHzSpeedModeSet(void)
Description	Sets the power control mode to subosc-speed mode.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

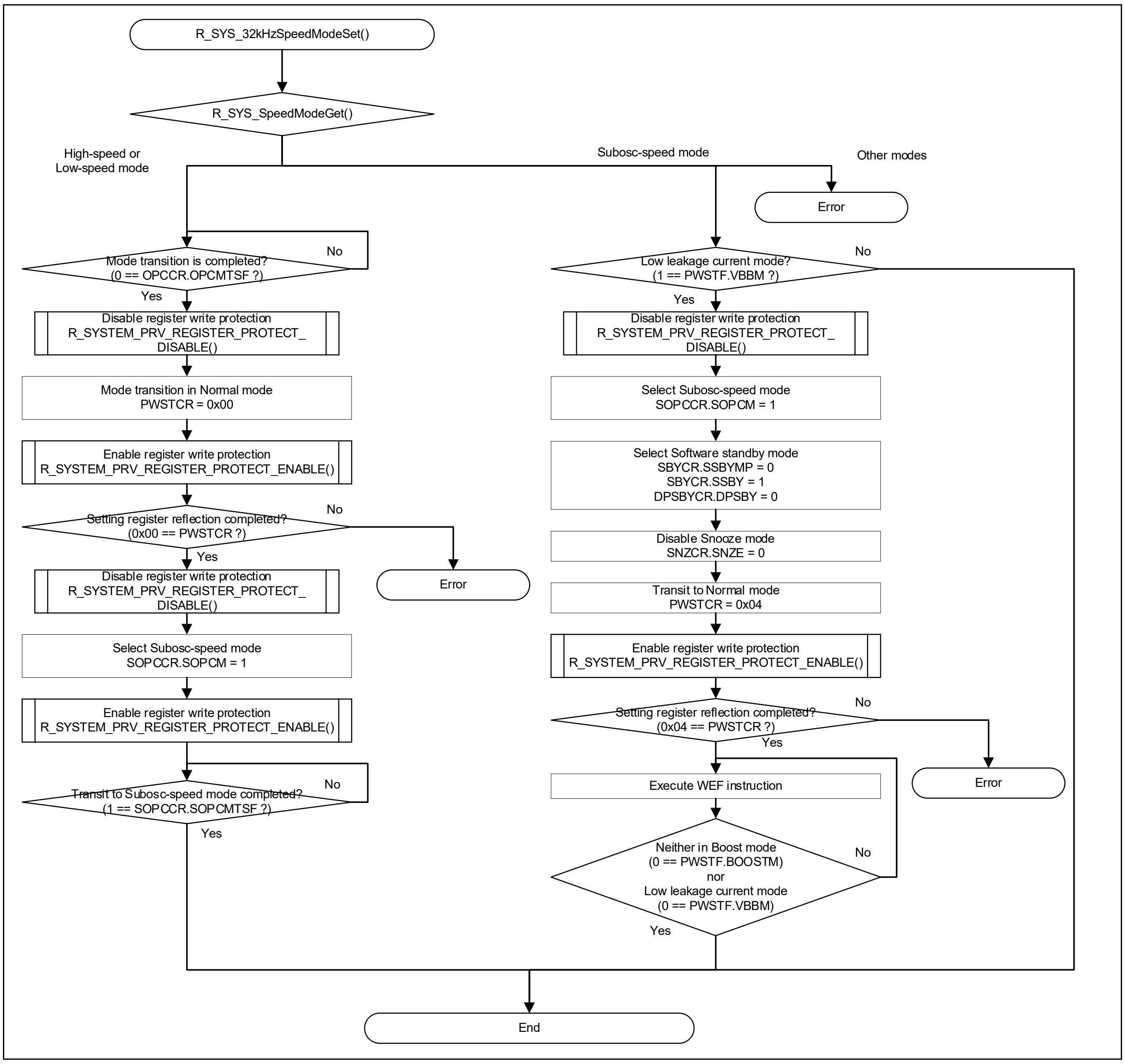

R_SYS_32kHzSpeedModeSet()
R_SYS_SpeedModeGet()
High-speed or Low-speed mode
Subosc-speed mode
Other modes
Error
Mode transition is completed? (0 == OPCCR.OPCMTSF ?)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Mode transition in Normal mode PWSTCR = 0x00
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x00 == PWSTCR ?)
No
Error
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Select Subosc-speed mode SOPCCR.SOPCM = 1
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Transit to Subosc-speed mode completed? (1 == SOPCCR.SOPCMTSF ?)
No
Yes
Low leakage current mode? (1 == PWSTF.VBBM ?)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Select Subosc-speed mode SOPCCR.SOPCM = 1
Select Software standby mode SBYCR.SSBYMP = 0 SBYCR.SSBY = 1 DPSBYCR.DPSBY = 0
Disable Snooze mode SNZCR.SNZE = 0
Transit to Normal mode PWSTCR = 0x04
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Setting register reflection completed? (0x04 == PWSTCR ?)
No
Yes
Error
Execute WEF instruction
Neither in Boost mode (0 == PWSTF.BOOSTM) nor Low leakage current mode (0 == PWSTF.VBBM)
No
Yes
End

Figure 4-7 R_SYS_32kHzSpeedModeSet Function Processing Flow

4.3.7 R_SYS_SpeedModeGet Function

Table 4-16 R_SYS_SpeedModeGet Function Specifications

Format	e_system_speed_mode_t R_SYS_SpeedModeGet(void)
Description	Obtains the current power control mode.
Argument	None
Return value	Boost (0)
	High-speed (1)
	Low-speed (2)
	32kHz-speed (3)
Remarks	–

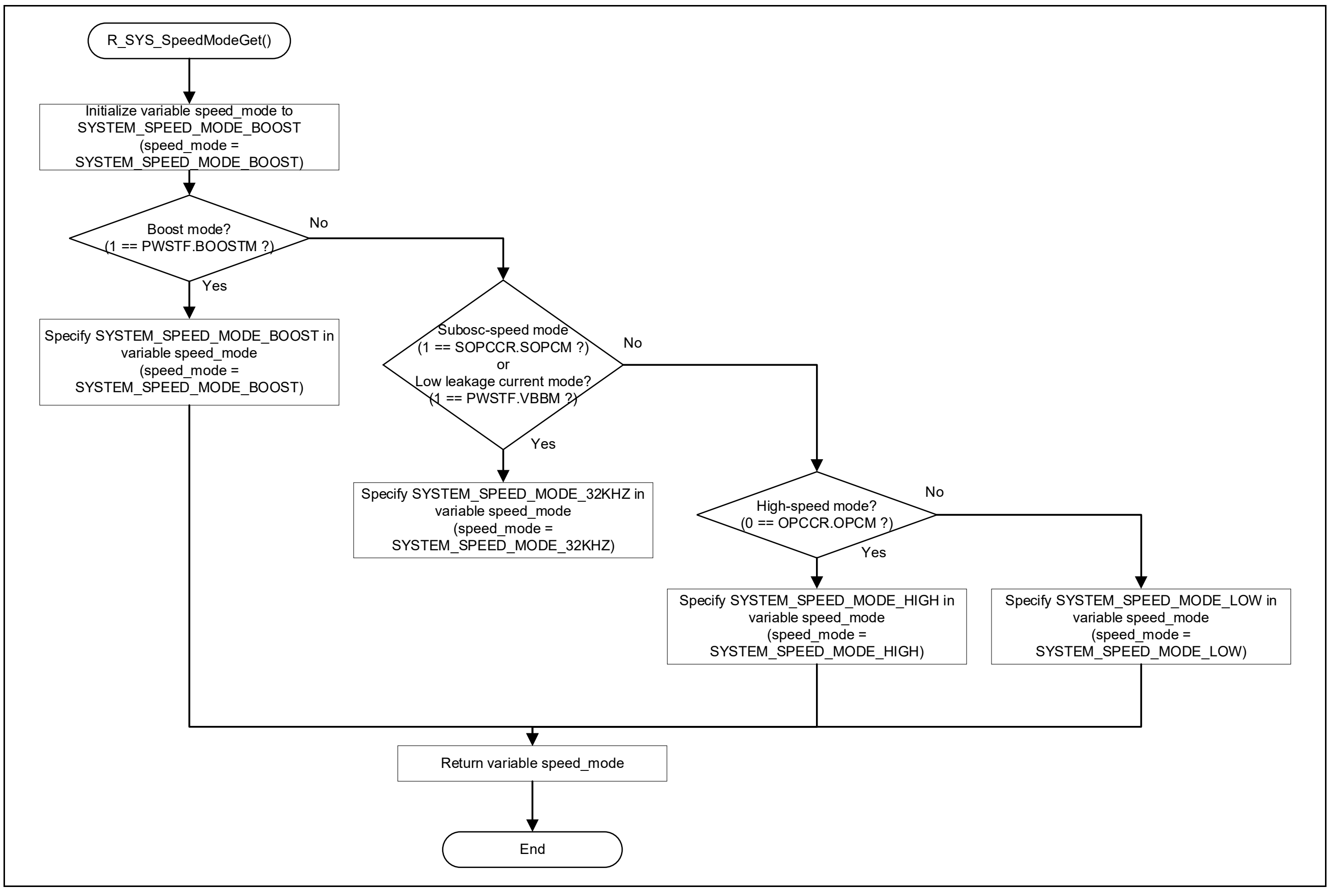

R_SYS_SpeedModeGet()
Initialize variable speed_mode to SYSTEM_SPEED_MODE_BOOST (speed_mode = SYSTEM_SPEED_MODE_BOOST)
Boost mode? (1 == PWSTF.BOOSTM ?)
No
Yes
Specify SYSTEM_SPEED_MODE_BOOST in variable speed_mode (speed_mode = SYSTEM_SPEED_MODE_BOOST)
Subosc-speed mode (1 == SOPCCR.SOPCM ?) or Low leakage current mode? (1 == PWSTF.VBBM ?)
No
Yes
Specify SYSTEM_SPEED_MODE_32KHZ in variable speed_mode (speed_mode = SYSTEM_SPEED_MODE_32KHZ)
High-speed mode? (0 == OPCCR.OPCM ?)
No
Yes
Specify SYSTEM_SPEED_MODE_HIGH in variable speed_mode (speed_mode = SYSTEM_SPEED_MODE_HIGH)
Specify SYSTEM_SPEED_MODE_LOW in variable speed_mode (speed_mode = SYSTEM_SPEED_MODE_LOW)
Return variable speed_mode
End

Figure 4.8 R_SYS_SpeedModeGet Function Processing Flow

4.3.8 R_SYS_SystemClockHOCOSet Function

Table 4-17 R_SYS_SystemClockHOCOSet Function Specifications

Format	int32_t R_SYS_SystemClockHOCOSet(void)
Description	Specifies the high-speed on-chip oscillator for the system clock source. When the operating frequency is higher than 32 MHz, the wait cycle in flash memory access is set to one cycle. When the operating frequency is 32 MHz or lower, the wait cycle count for the flash memory is set to zero cycles.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

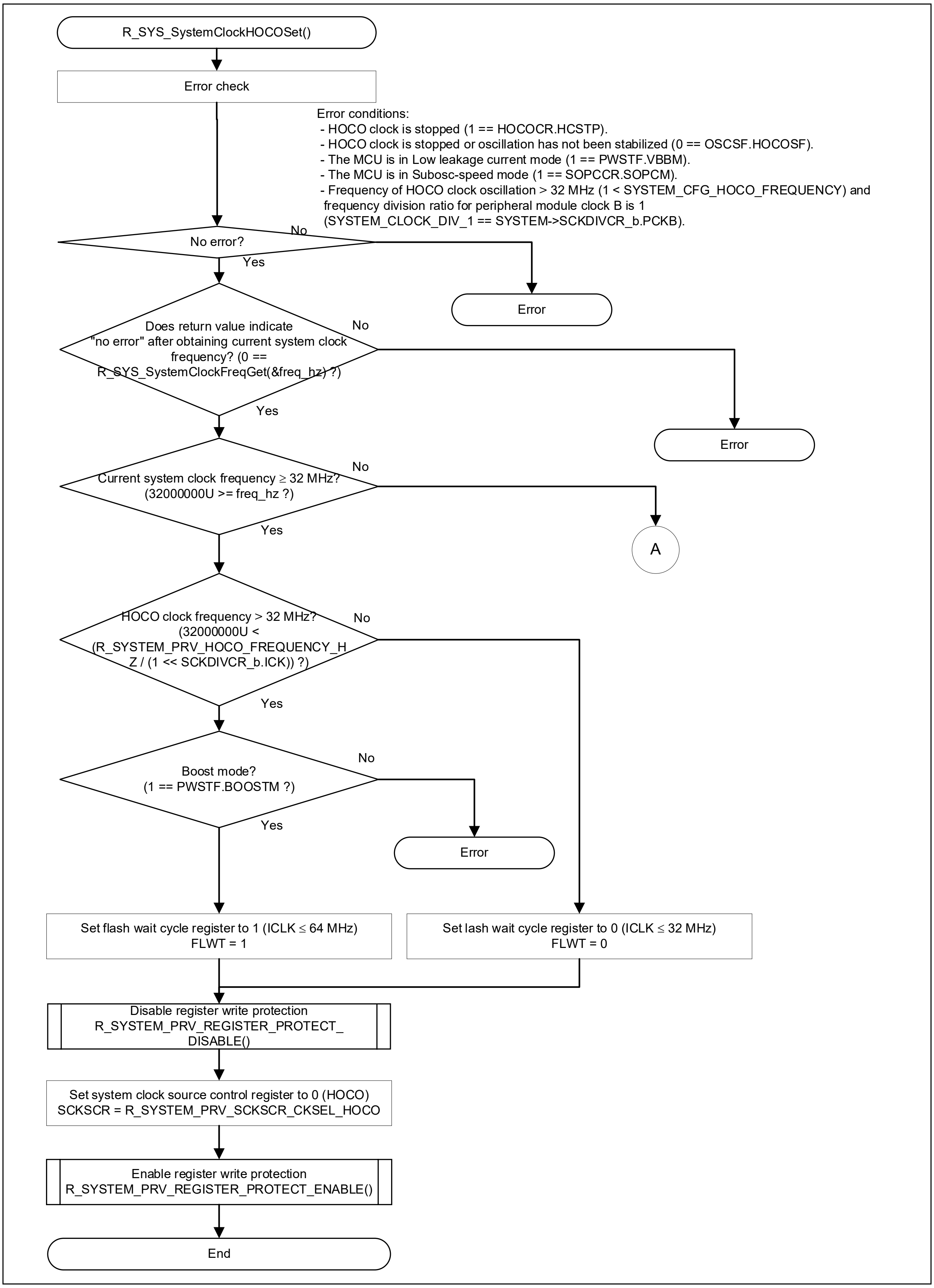

R_SYS_SystemClockHOCOSet()
Error check
Error conditions:
- HOCO clock is stopped (1 == HOCOCR.HCSTP).
- HOCO clock is stopped or oscillation has not been stabilized (0 == OSCSF.HOCOSF).
- The MCU is in Low leakage current mode (1 == PWSTF.VBBM).
- The MCU is in Subosc-speed mode (1 == SOPCCR.SOPCM).
- Frequency of HOCO clock oscillation > 32 MHz (1 < SYSTEM_CFG_HOCO_FREQUENCY) and frequency division ratio for peripheral module clock B is 1 (SYSTEM_CLOCK_DIV_1 == SYSTEM->SCKDIVCR_b.PCKB).
No error?
No
Yes
Error
Does return value indicate "no error" after obtaining current system clock frequency? (0 == R_SYS_SystemClockFreqGet(&freq_hz) ?)
No
Yes
Error
Current system clock frequency ≥ 32 MHz? (32000000U >= freq_hz ?)
No
Yes
A
HOCO clock frequency > 32 MHz? (32000000U < (R_SYSTEM_PRV_HOCO_FREQUENCY_HZ / (1 << SCKDIVCR_b.ICK)) ?)
No
Yes
Boost mode? (1 == PWSTF.BOOSTM ?)
No
Yes
Error
Set flash wait cycle register to 1 (ICLK ≤ 64 MHz) FLWT = 1
Set lash wait cycle register to 0 (ICLK ≤ 32 MHz) FLWT = 0
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 0 (HOCO) SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_HOCO
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
End

Figure 4.9 R_SYS_SystemClockHOCOSet Function Processing Flow (1/2)

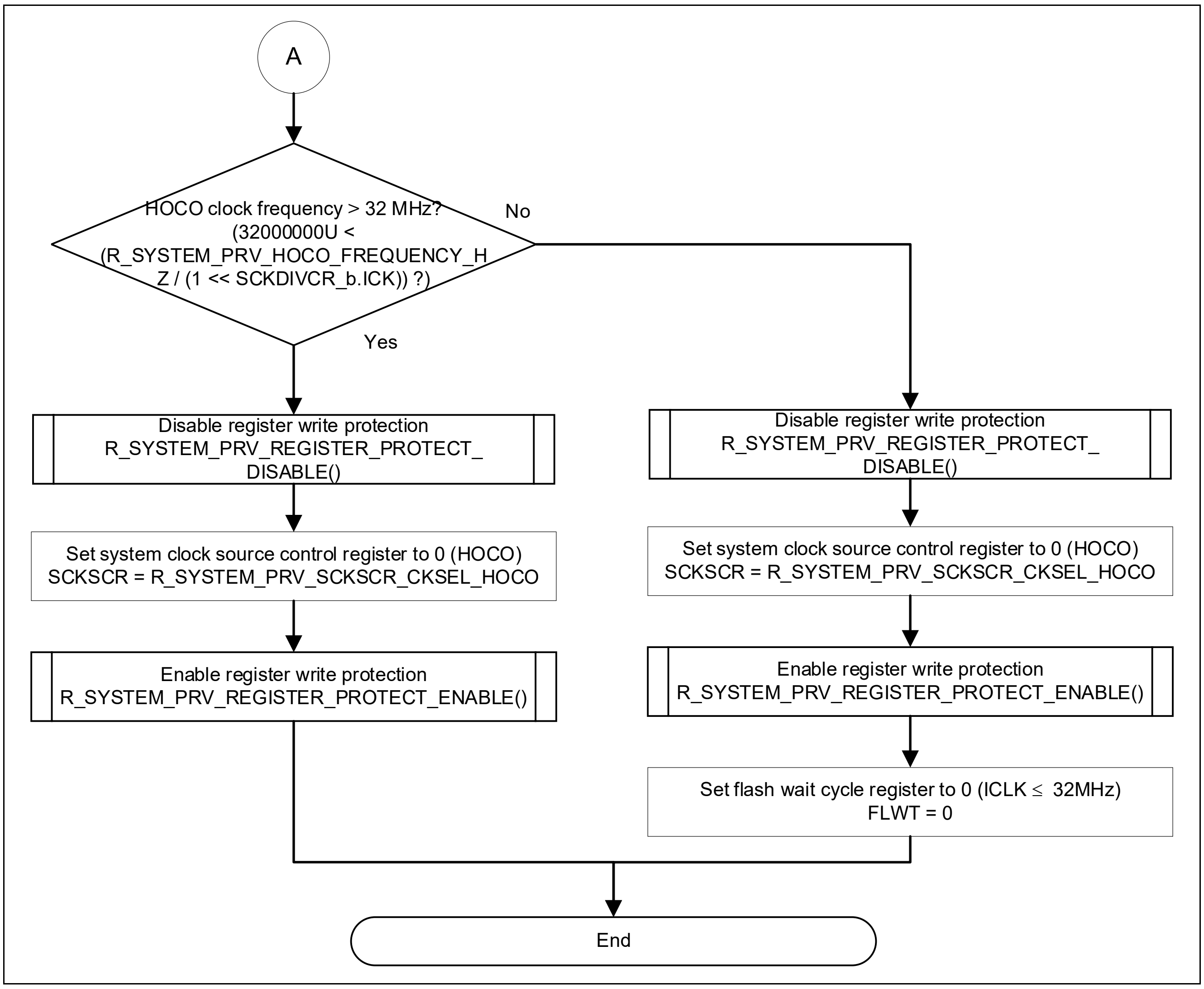

A
HOCO clock frequency > 32 MHz?
(32000000U <
(R_SYSTEM_PRV_HOCO_FREQUENCY_HZ / (1 << SCKDIVCR_b.ICK)) ?)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 0 (HOCO)
SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_HOCO
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 0 (HOCO)
SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_HOCO
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz)
FLWT = 0
End

Figure 4.10 R_SYS_SystemClockHOCOSet Function Processing Flow (2/2)

4.3.9 R_SYS_SystemClockMOCOSet Function

Table 4-18 R_SYS_SystemClockMOCOSet Function Specifications

Format	int32_t R_SYS_SystemClockMOCOSet(void)
Description	Specifies the middle-speed on-chip oscillator for the system clock source. The flash memory wait state count is set to zero cycles.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

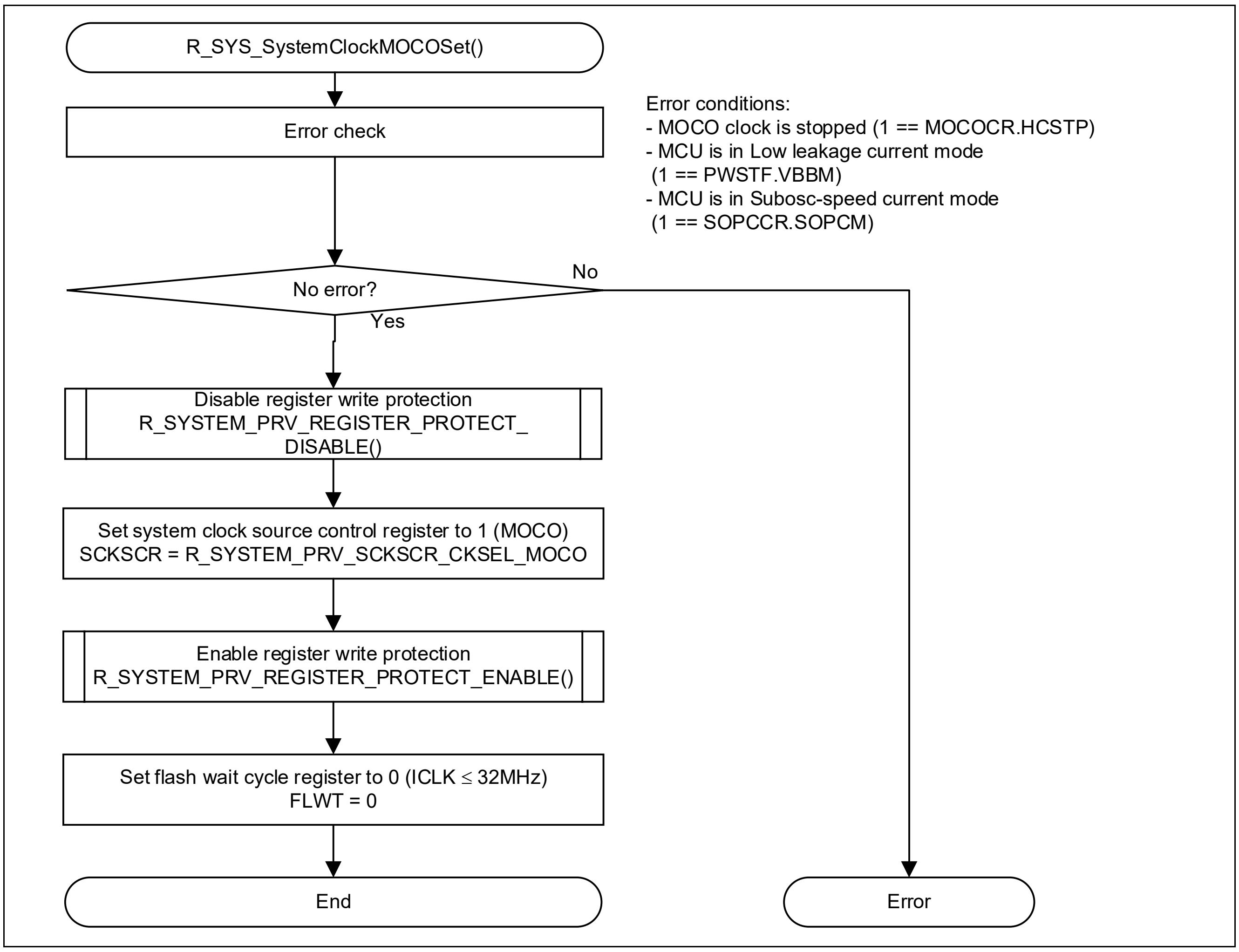
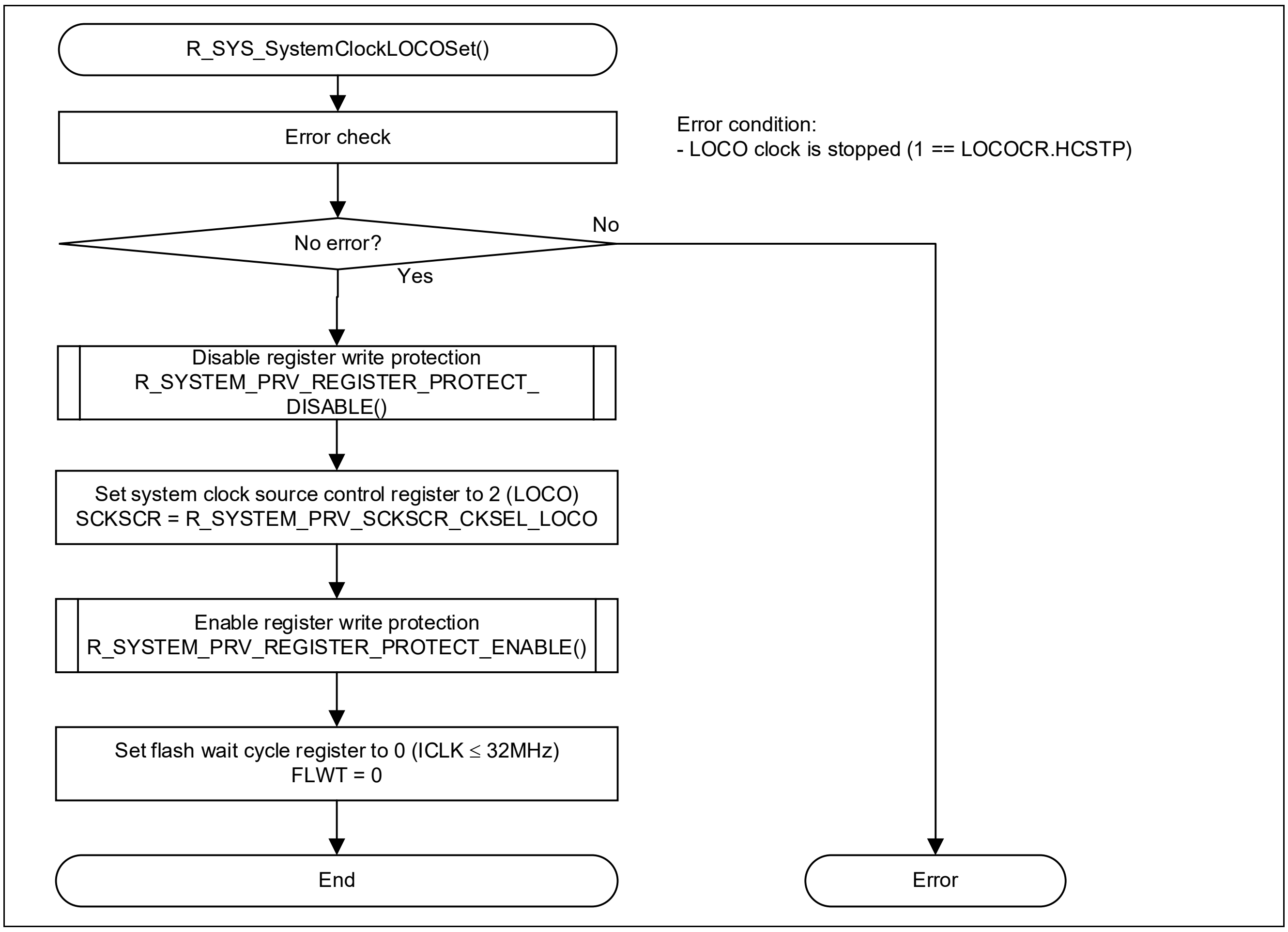

R_SYS_SystemClockMOCOSet()
Error check
Error conditions:
- MOCO clock is stopped (1 == MOCOCR.HCSTP)
- MCU is in Low leakage current mode
(1 == PWSTF.VBBM)
- MCU is in Subosc-speed current mode
(1 == SOPCCR.SOPCM)
No error?
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 1 (MOCO)
SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_MOCO
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz)
FLWT = 0
End
Error

Figure 4.11 R_SYS_SystemClockMOCOSet Function Processing Flow

4.3.10 R_SYS_SystemClockLOCOSet Function

Table 4-19 R_SYS_SystemClockLOCOSet Function Specifications

Format	int32_t R_SYS_SystemClockLOCOSet(void)
Description	Specifies the low-speed on-chip oscillator for the system clock source. The flash memory wait state count is set to zero cycles.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–



R_SYS_SystemClockLOCOSet()
Error check
Error condition:
- LOCO clock is stopped (1 == LOCOCR.HCSTP)
No error?
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 2 (LOCO) SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_LOCO
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz) FLWT = 0
End
Error

Figure 4.12 R_SYS_SystemClockLOCOSet Function Processing Flow

4.3.11 R_SYS_SystemClockMOSCSet Function

Table 4-20 R_SYS_SystemClockMOSCSet Function Specifications

Format	int32_t R_SYS_SystemClockMOSCSet(void)
Description	Specifies the main clock oscillator for the system clock source. The flash memory wait cycle count is set to zero cycles.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

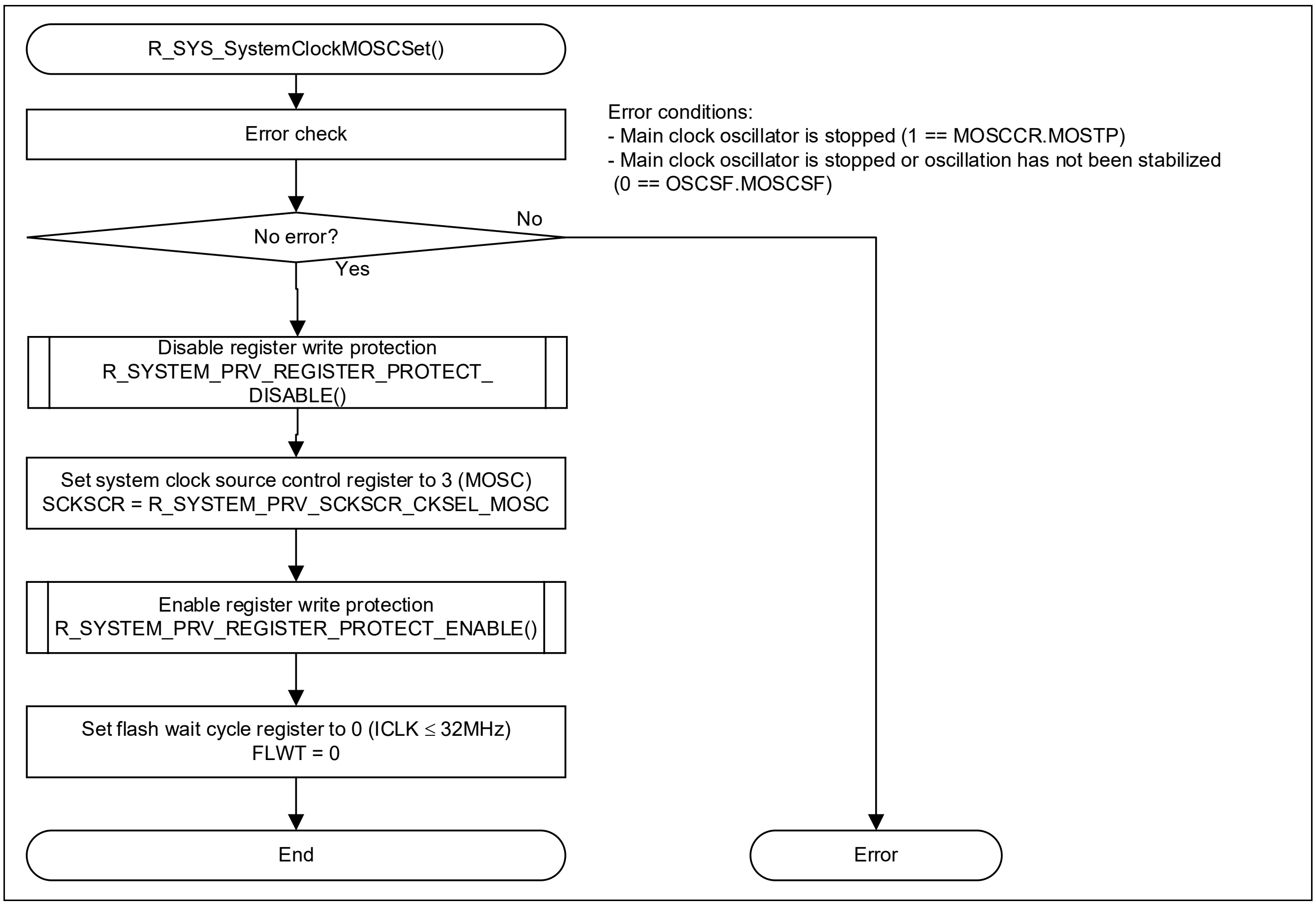
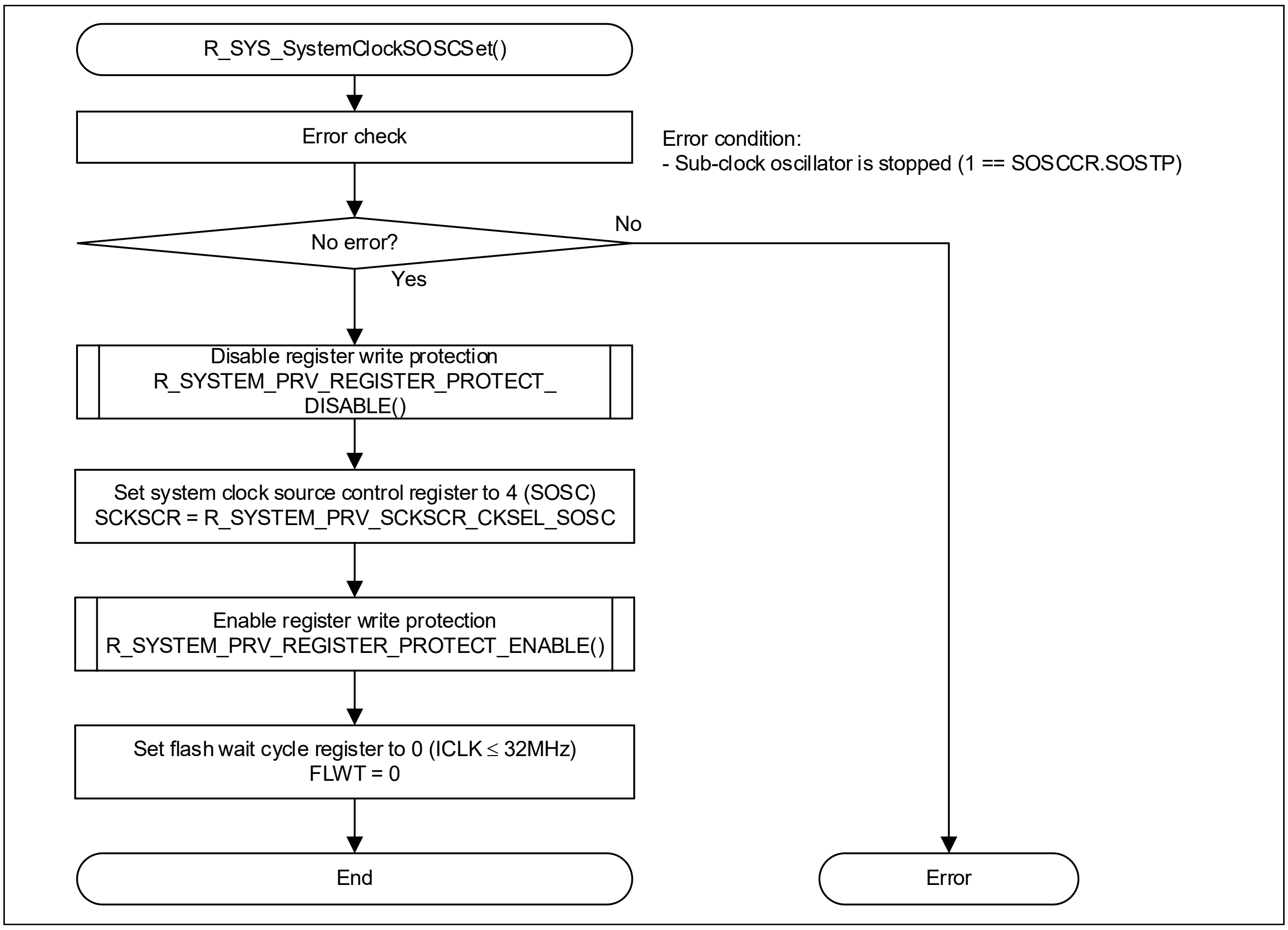

R_SYS_SystemClockMOSCSet()
Error check
Error conditions:
- Main clock oscillator is stopped (1 == MOSCCR.MOSTP)
- Main clock oscillator is stopped or oscillation has not been stabilized (0 == OSCSF.MOSCSF)
No error?
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 3 (MOSC) SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_MOSC
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz) FLWT = 0
End
Error

Figure 4.13 R_SYS_SystemClockMOSCSet Function Processing Flow

4.3.12 R_SYS_SystemClockSOSCSet Function

Table 4-21 R_SYS_SystemClockSOSCSet Function Specifications

Format	int32_t R_SYS_SystemClockSOSCSet(void)
Description	Specifies the sub-clock oscillator for the system clock source. The flash memory wait cycle count is set to zero cycles.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–



R_SYS_SystemClockSOSCSet()
Error check
Error condition:
- Sub-clock oscillator is stopped (1 == SOSCCR.SOSTP)
No error?
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 4 (SOSC) SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_SOSC
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz) FLWT = 0
End
Error

Figure 4.14 R_SYS_SystemClockSOSCSet Function Processing Flow

4.3.13 R_SYS_SystemClockPLLSet Function

Table 4-22 R_SYS_SystemClockPLLSet Function Specifications

Format	int32_t R_SYS_SystemClockPLLSet(void)	
Description	Specifies the PLL circuit for the system clock source. When the operating frequency is higher than 32 MHz, the wait cycle in flash memory access is set to one cycle. When the operating frequency is 32 MHz or lower, the flash memory wait cycle count is set to zero cycles.	
Argument	None	
Return value	Normal (0)	
	Abnormal (-1)	
Remarks	–	

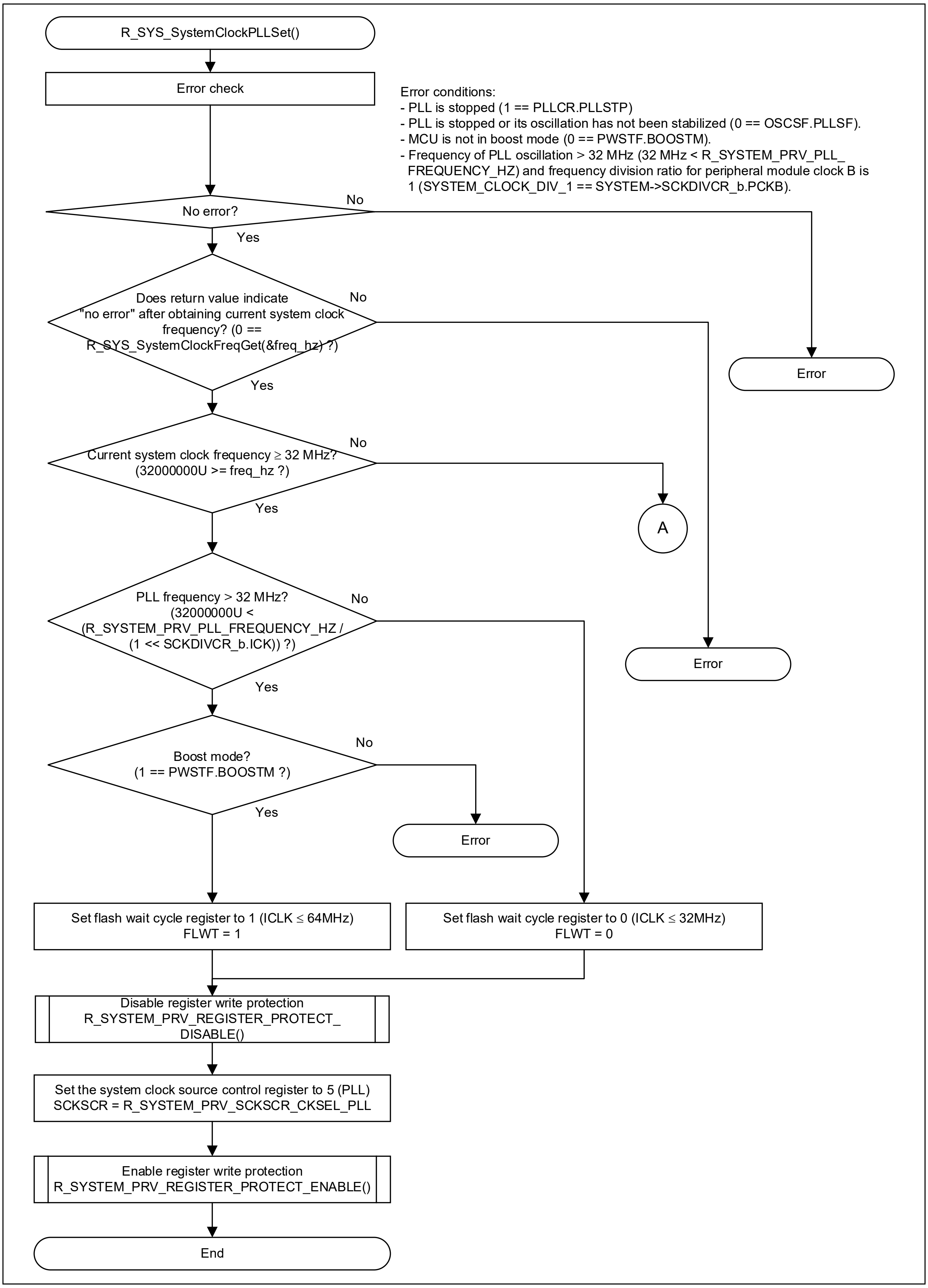

R_SYS_SystemClockPLLSet()
Error check
Error conditions:
- PLL is stopped (1 == PLLCR.PLLSTP)
- PLL is stopped or its oscillation has not been stabilized (0 == OSCSF.PLLSF).
- MCU is not in boost mode (0 == PWSTF.BOOSTM).
- Frequency of PLL oscillation > 32 MHz (32 MHz < R_SYSTEM_PRV_PLL_FREQUENCY_HZ) and frequency division ratio for peripheral module clock B is 1 (SYSTEM_CLOCK_DIV_1 == SYSTEM->SCKDIVCR_b.PCKB).
No error?
No
Yes
Error
Does return value indicate "no error" after obtaining current system clock frequency? (0 == R_SYS_SystemClockFreqGet(&freq_hz) ?)
No
Yes
Current system clock frequency ≥ 32 MHz? (32000000U >= freq_hz ?)
No
Yes
A
PLL frequency > 32 MHz? (32000000U < (R_SYSTEM_PRV_PLL_FREQUENCY_HZ / (1 << SCKDIVCR_b.ICK)) ?)
No
Yes
Error
Boost mode? (1 == PWSTF.BOOSTM ?)
No
Yes
Error
Set flash wait cycle register to 1 (ICLK ≤ 64MHz) FLWT = 1
Set flash wait cycle register to 0 (ICLK ≤ 32MHz) FLWT = 0
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set the system clock source control register to 5 (PLL) SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_PLL
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
End

Figure 4.15 R_SYS_SystemClockPLLSet Function Processing Flow (1/2)

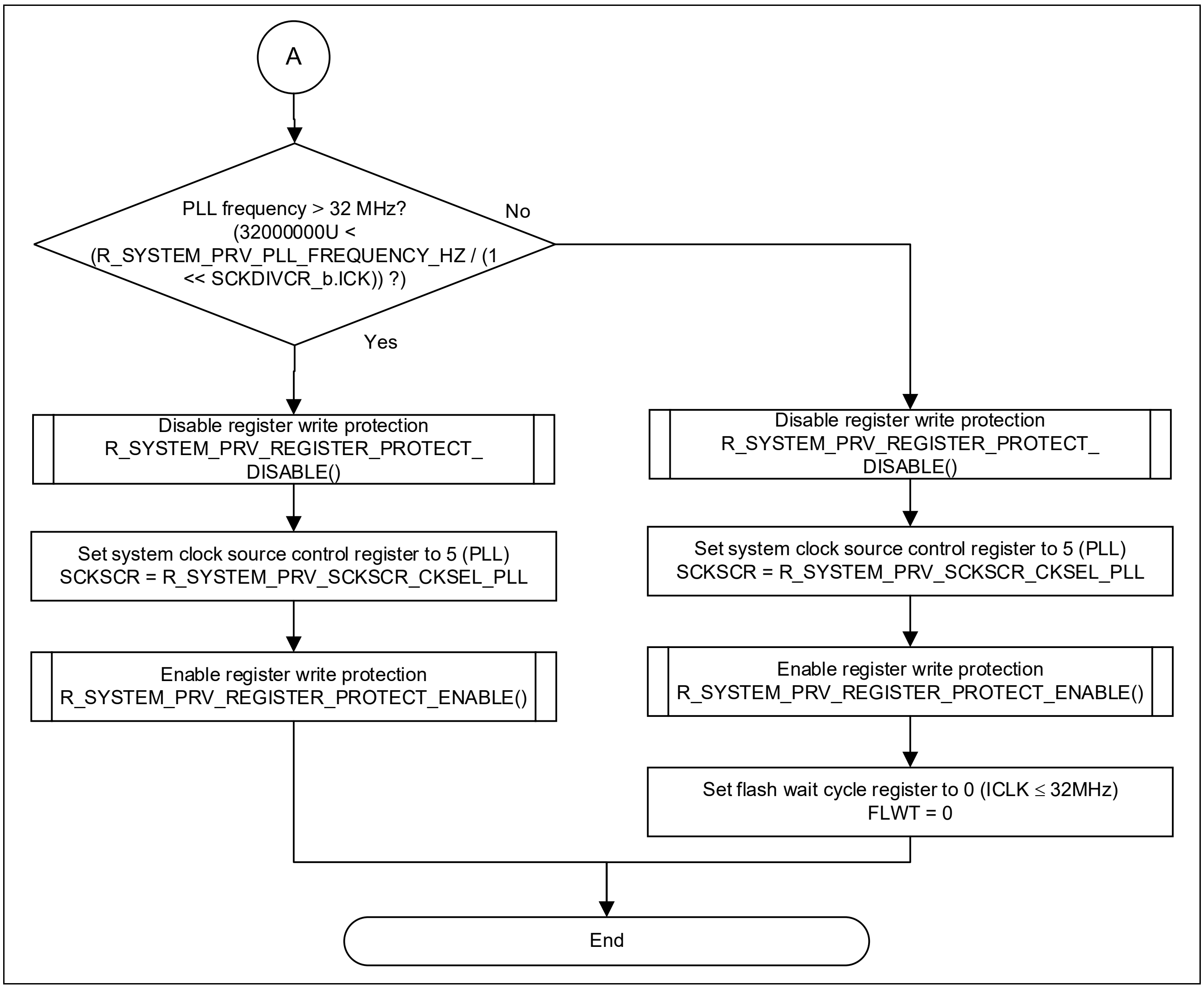

A
PLL frequency > 32 MHz?
(32000000U <
(R_SYSTEM_PRV_PLL_FREQUENCY_HZ / (1
<< SCKDIVCR_b.ICK)) ?)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 5 (PLL)
SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_PLL
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set system clock source control register to 5 (PLL)
SCKSCR = R_SYSTEM_PRV_SCKSCR_CKSEL_PLL
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Set flash wait cycle register to 0 (ICLK ≤ 32MHz)
FLWT = 0
End

Figure 4.16 R_SYS_SystemClockPLLSet Function Processing Flow (2/2)

4.3.14 R_SYS_SystemClockFreqGet Function

Table 4-23 R_SYS_SystemClockFreqGet Function Specifications

Format	int32_t R_SYS_SystemClockFreqGet(uint32_t * p_freq_hz)
Description	Obtains the frequency of the system clock (ICLK) and peripheral module clock (PCLKA).
Argument	uint32_t * p_freq_hz [Input]: Specifies the location for storing the obtained frequency.
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

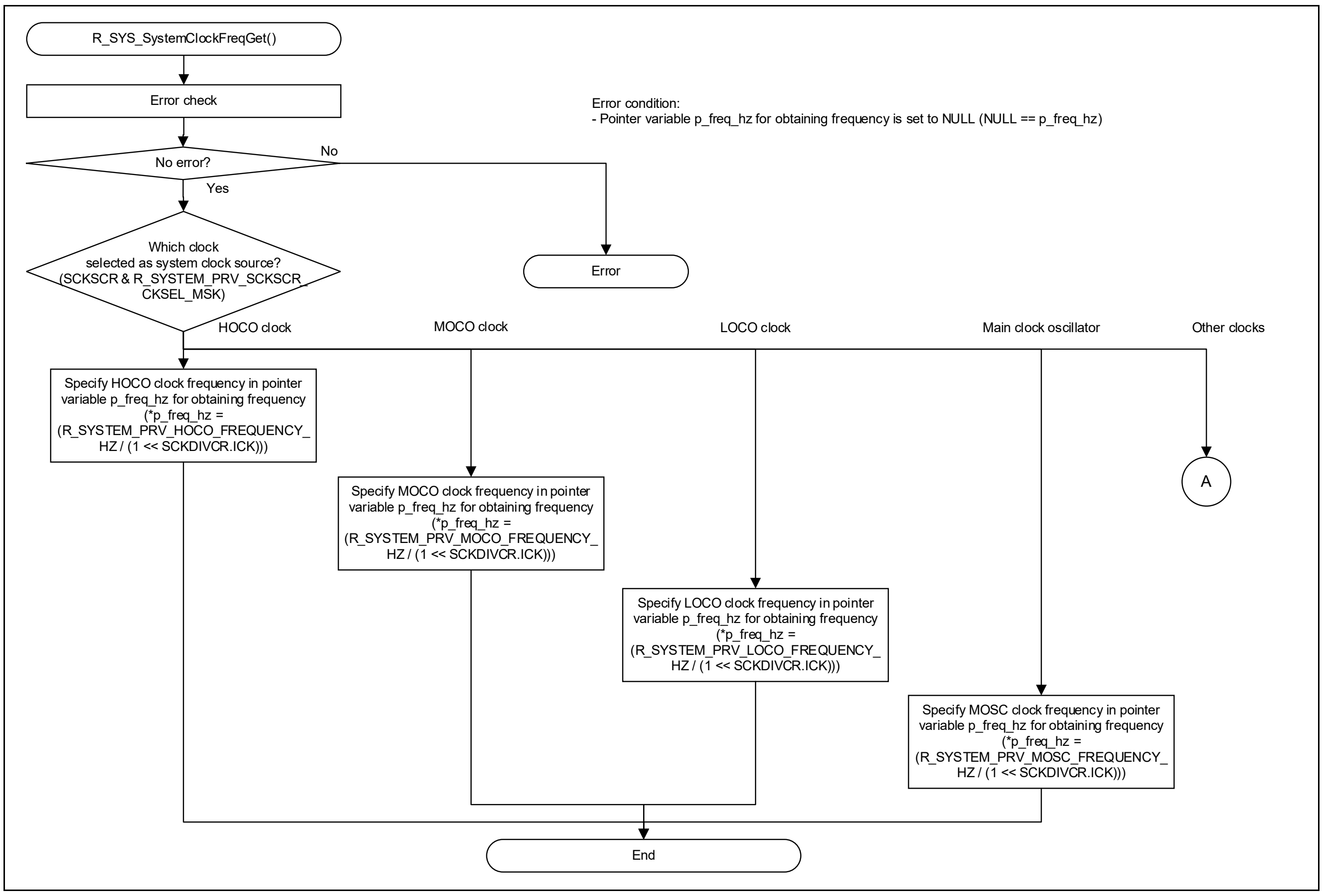

R_SYS_SystemClockFreqGet()
Error check
Error condition:
- Pointer variable p_freq_hz for obtaining frequency is set to NULL (NULL == p_freq_hz)
No error?
No
Yes
Error
Which clock selected as system clock source? (SCKSCR & R_SYSTEM_PRV_SCKSCR_CKSEL_MSK)
HOCO clock
MOCO clock
LOCO clock
Main clock oscillator
Other clocks
Specify HOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_HOCO_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
Specify MOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_MOCO_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
Specify LOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_LOCO_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
Specify MOSC clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_MOSC_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
A
End

Figure 4.17 R_SYS_SystemClockFreqGet Function Processing Flow (1/2)

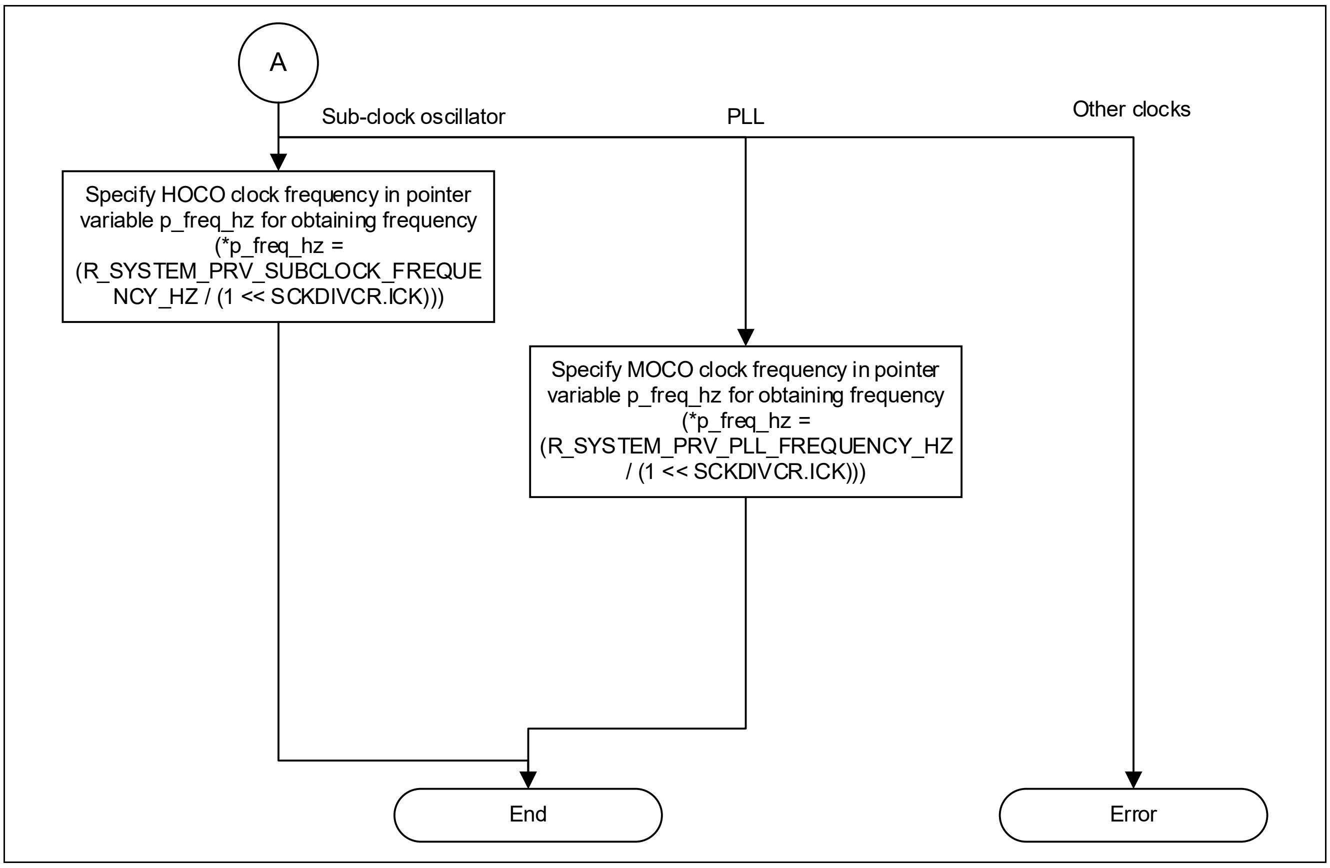
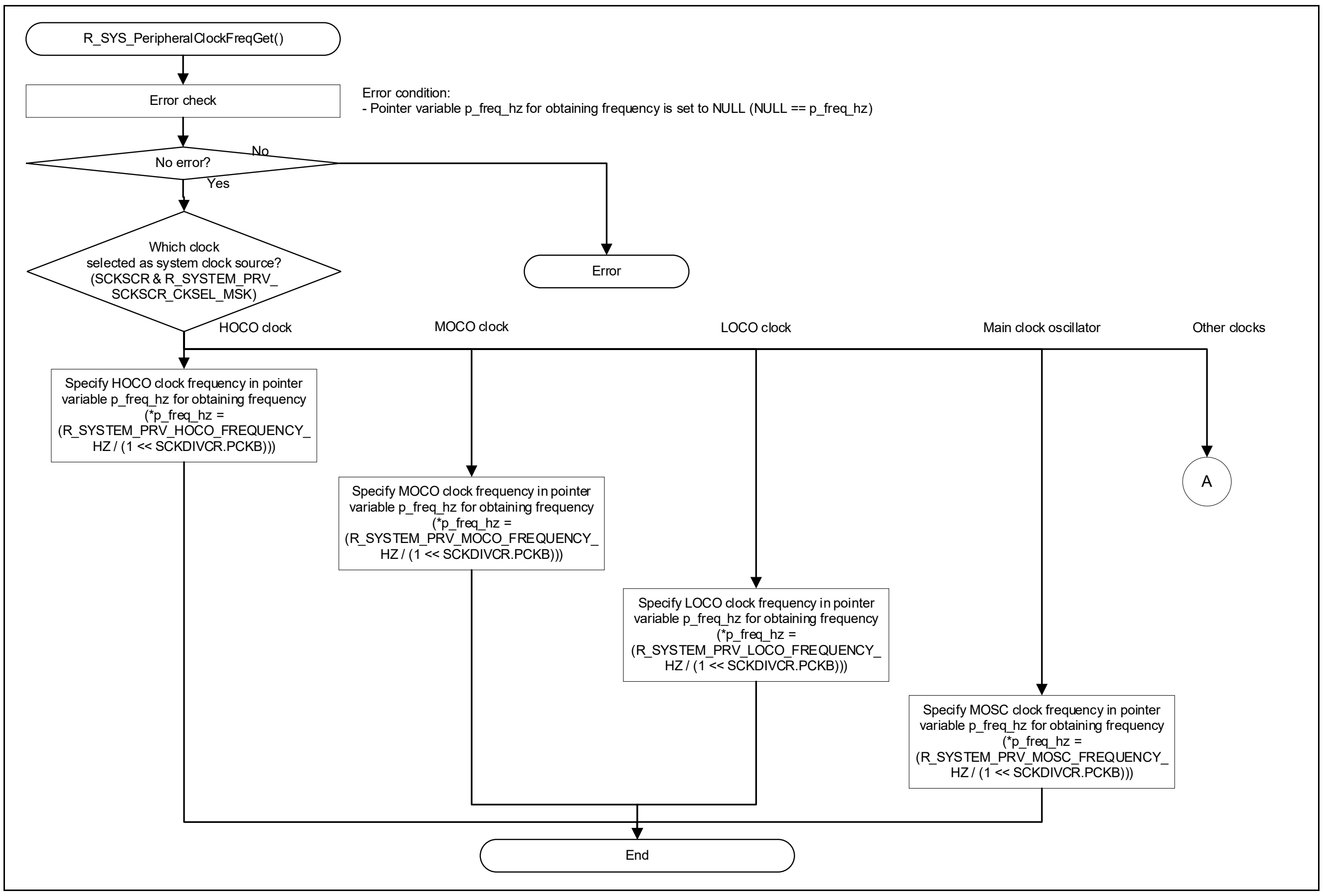

A
Sub-clock oscillator
PLL
Other clocks
Specify HOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_SUBCLOCK_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
Specify MOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_PLL_FREQUENCY_HZ / (1 << SCKDIVCR.ICK)))
End
Error

Figure 4.18 R_SYS_SystemClockFreqGet Function Processing Flow (2/2)

4.3.15 R_SYS_PeripheralClockFreqGet Function

Table 4-24 R_SYS_PeripheralClockFreqGet Function Specifications

Format	int32_t R_SYS_PeripheralClockFreqGet(uint32_t * p_freq_hz)
Description	Obtains the frequency of the peripheral module clock B (PCLKB).
Argument	uint32_t * p_freq_hz [Input]: Specifies the location for storing the obtained frequency.
Return value	Normal (0)
	Abnormal (-1)
Remarks	–



R_SYS_PeripheralClockFreqGet()
Error check
Error condition:
- Pointer variable p_freq_hz for obtaining frequency is set to NULL (NULL == p_freq_hz)
No error?
No
Yes
Error
Which clock selected as system clock source? (SCKSCR & R_SYSTEM_PRV_SCKSCR_CKSEL_MSK)
HOCO clock
MOCO clock
LOCO clock
Main clock oscillator
Other clocks
Specify HOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_HOCO_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
Specify MOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_MOCO_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
Specify LOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_LOCO_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
Specify MOSC clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_MOSC_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
A
End

Figure 4.19 R_SYS_PeripheralClockFreqGet Function Processing Flow (1/2)

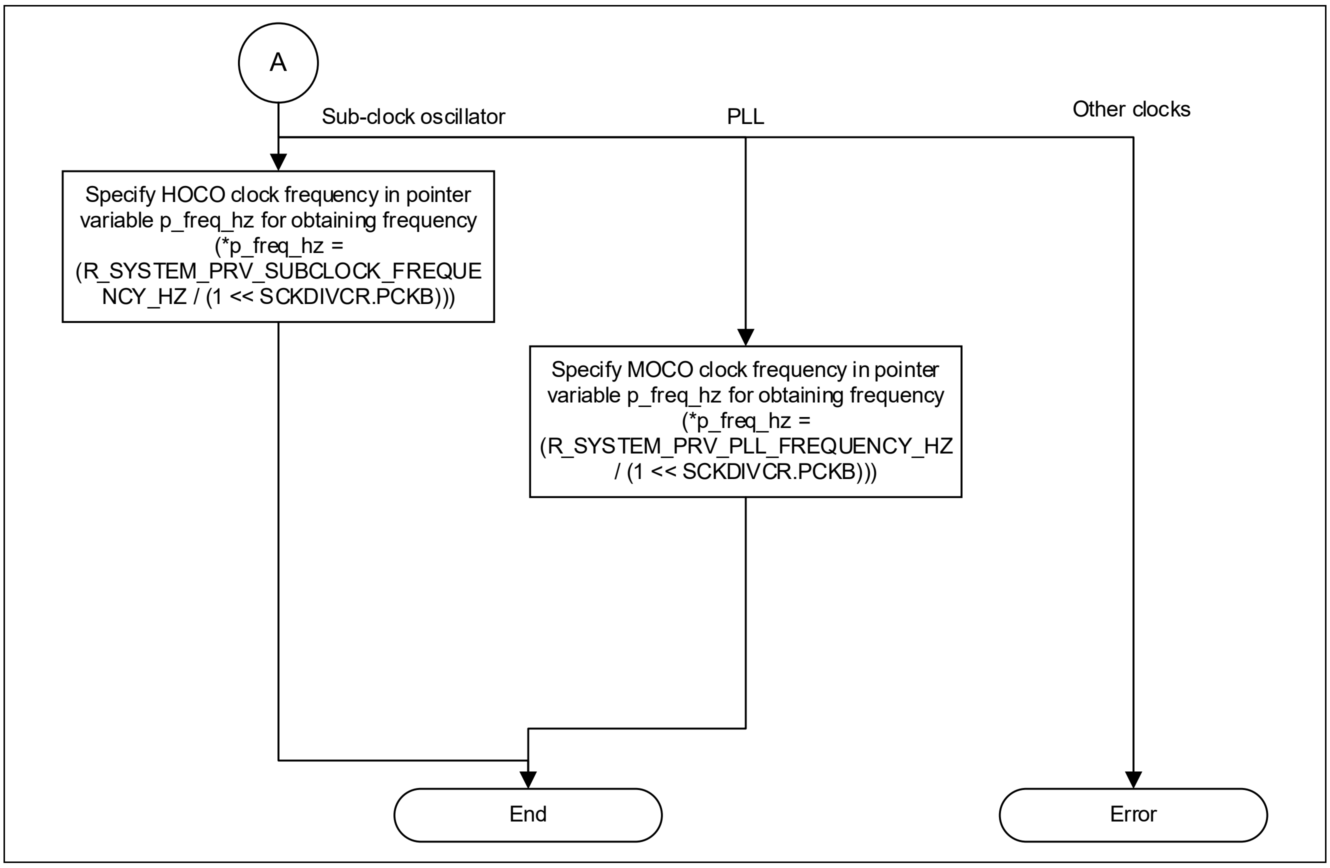

A
Sub-clock oscillator
PLL
Other clocks
Specify HOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_SUBCLOCK_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
Specify MOCO clock frequency in pointer variable p_freq_hz for obtaining frequency (*p_freq_hz = (R_SYSTEM_PRV_PLL_FREQUENCY_HZ / (1 << SCKDIVCR.PCKB)))
End
Error

Figure 4.20 R_SYS_PeripheralClockFreqGet Function Processing Flow (2/2)

4.3.16 R_SYS_SystemClockDividerSet Function

Table 4-25 R_SYS_SystemClockDividerSet Function Specifications

Format	int32_t R_SYS_SystemClockDividerSet(e_system_sys_clock_div_t iclk_div, e_system_sys_clock_div_t pclkb_div)
Description	Specifies the frequency division values for the system clock (ICLK)/peripheral module clock A (PCLKA) and the peripheral module clock B (PCLKB).
Argument	e_system_sys_clock_div_t iclk_div [Input]: Specifies the frequency division value for the system clock (ICLK) and peripheral module clock A (PCLKA).
	e_system_sys_clock_div_t pclkb_div [Input]: Specifies the frequency division value for the peripheral module clock B (PCLKB).
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

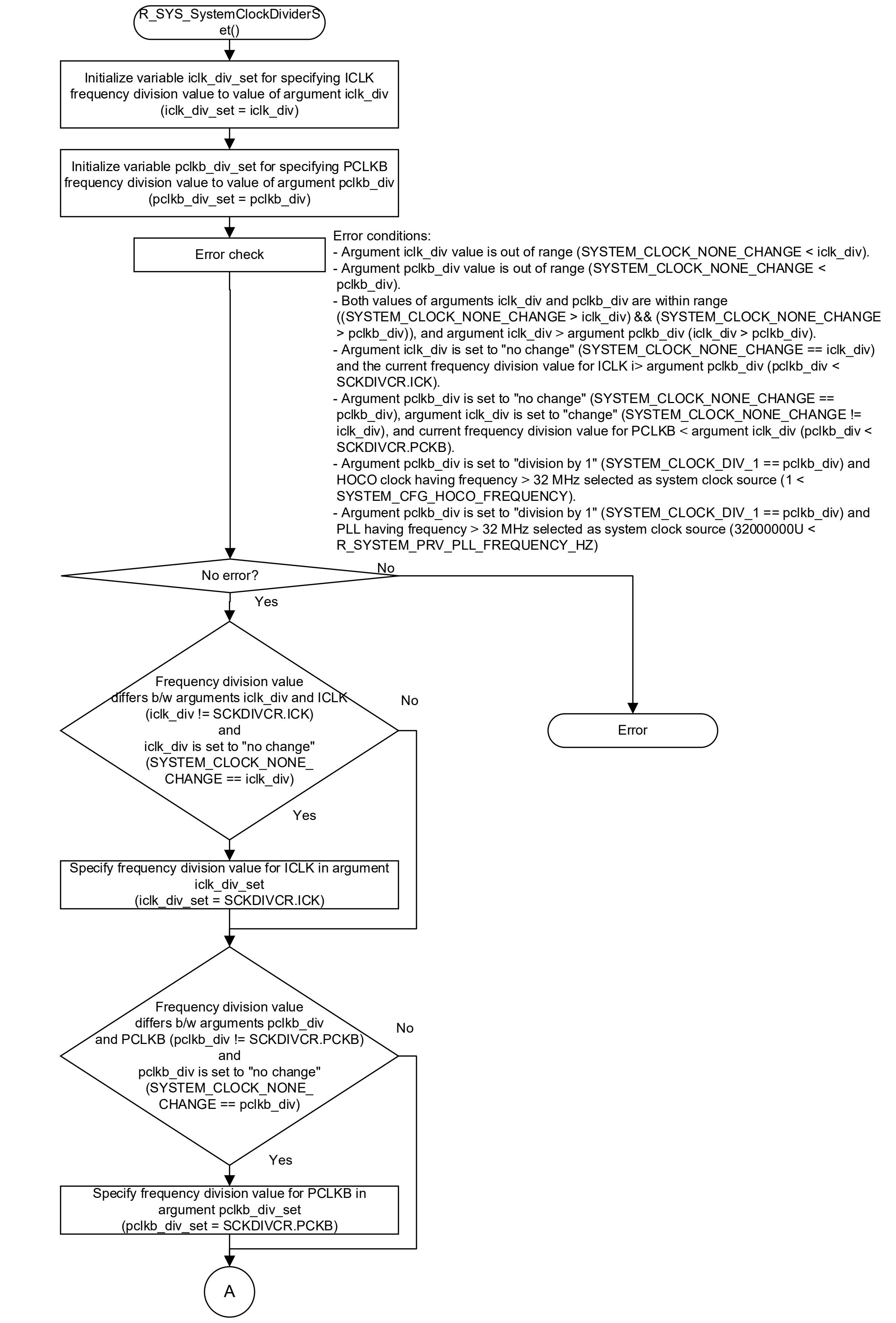

R_SYS_SystemClockDividerSet()
Initialize variable iclk_div_set for specifying ICLK frequency division value to value of argument iclk_div (iclk_div_set = iclk_div)
Initialize variable pclkb_div_set for specifying PCLKB frequency division value to value of argument pclkb_div (pclkb_div_set = pclkb_div)
Error check
Error conditions:
- Argument iclk_div value is out of range (SYSTEM_CLOCK_NONE_CHANGE < iclk_div).
- Argument pclkb_div value is out of range (SYSTEM_CLOCK_NONE_CHANGE < pclkb_div).
- Both values of arguments iclk_div and pclkb_div are within range ((SYSTEM_CLOCK_NONE_CHANGE > iclk_div) && (SYSTEM_CLOCK_NONE_CHANGE > pclkb_div)), and argument iclk_div > argument pclkb_div (iclk_div > pclkb_div).
- Argument iclk_div is set to "no change" (SYSTEM_CLOCK_NONE_CHANGE == iclk_div) and the current frequency division value for ICLK i> argument pclkb_div (pclkb_div < SCKDIVCR.ICK).
- Argument pclkb_div is set to "no change" (SYSTEM_CLOCK_NONE_CHANGE == pclkb_div), argument iclk_div is set to "change" (SYSTEM_CLOCK_NONE_CHANGE != iclk_div), and current frequency division value for PCLKB < argument iclk_div (pclkb_div < SCKDIVCR.PCKB).
- Argument pclkb_div is set to "division by 1" (SYSTEM_CLOCK_DIV_1 == pclkb_div) and HOCO clock having frequency > 32 MHz selected as system clock source (1 < SYSTEM_CFG_HOCO_FREQUENCY).
- Argument pclkb_div is set to "division by 1" (SYSTEM_CLOCK_DIV_1 == pclkb_div) and PLL having frequency > 32 MHz selected as system clock source (32000000U < R_SYSTEM_PRV_PLL_FREQUENCY_HZ)
No error?
No
Yes
Error
Frequency division value differs b/w arguments iclk_div and ICLK (iclk_div != SCKDIVCR.ICK) and iclk_div is set to "no change" (SYSTEM_CLOCK_NONE_CHANGE == iclk_div)
No
Yes
Specify frequency division value for ICLK in argument iclk_div_set (iclk_div_set = SCKDIVCR.ICK)
Frequency division value differs b/w arguments pclkb_div and PCLKB (pclkb_div != SCKDIVCR.PCKB) and pclkb_div is set to "no change" (SYSTEM_CLOCK_NONE_CHANGE == pclkb_div)
No
Yes
Specify frequency division value for PCLKB in argument pclkb_div_set (pclkb_div_set = SCKDIVCR.PCKB)
A

Figure 4.21 R_SYS_SystemClockDividerSet Function Processing Flow (1/2)

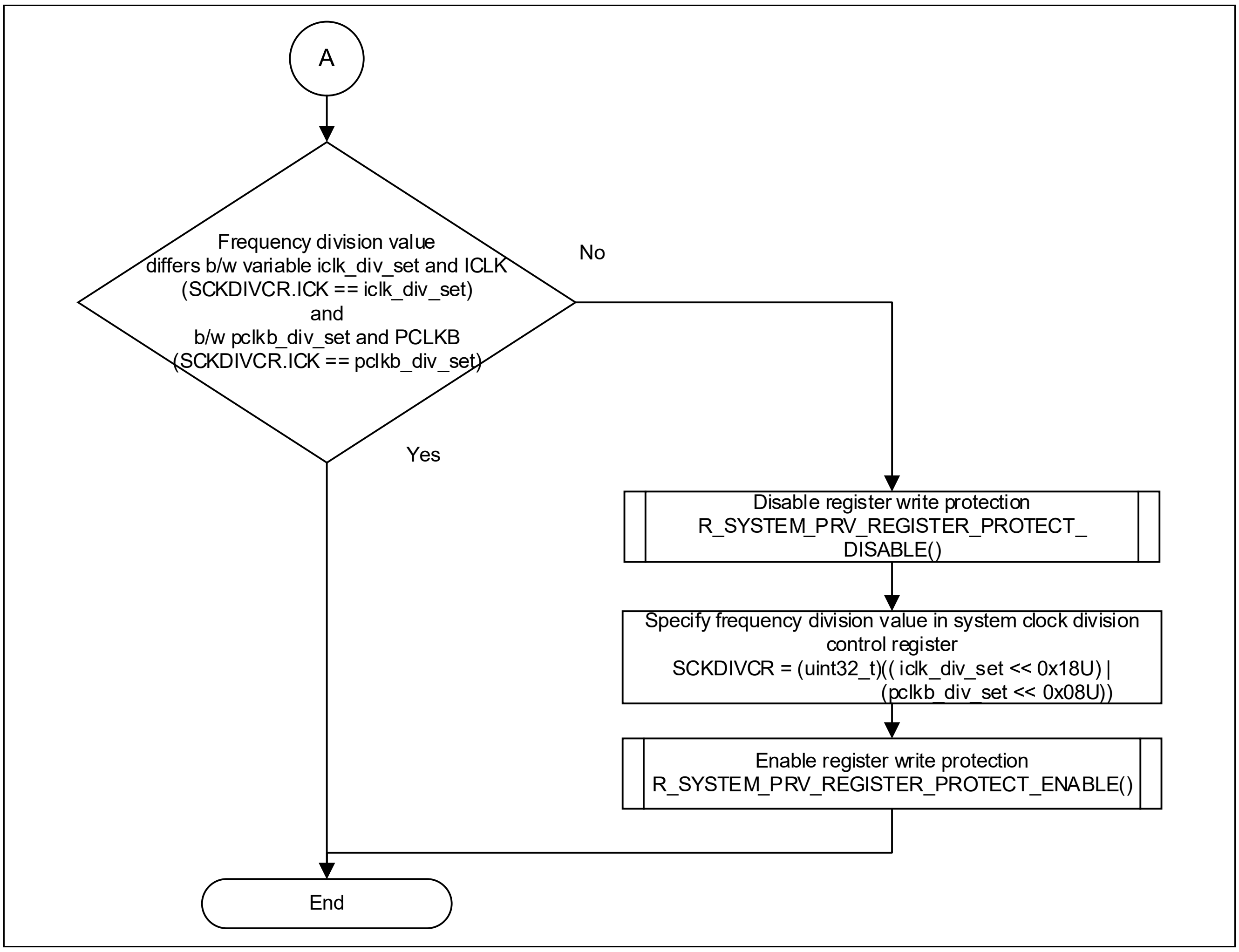

A
Frequency division value differs b/w variable iclk_div_set and ICLK (SCKDIVCR.ICK == iclk_div_set) and b/w pclkb_div_set and PCLKB (SCKDIVCR.ICK == pclkb_div_set)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Specify frequency division value in system clock division control register SCKDIVCR = (uint32_t)((iclk_div_set << 0x18U) | (pclkb_div_set << 0x08U))
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
End

Figure 4.22 R_SYS_SystemClockDividerSet Function Processing Flow (2/2)

4.3.17 R_SYS_MainOscSpeedClockStart Function

Table 4-26 R_SYS_MainOscSpeedClockStart Function Specifications

Format	void R_SYS_MainOscSpeedClockStart(void)
Description	Starts the operation of the main clock oscillator.
Argument	None
Return value	None
Remarks	–

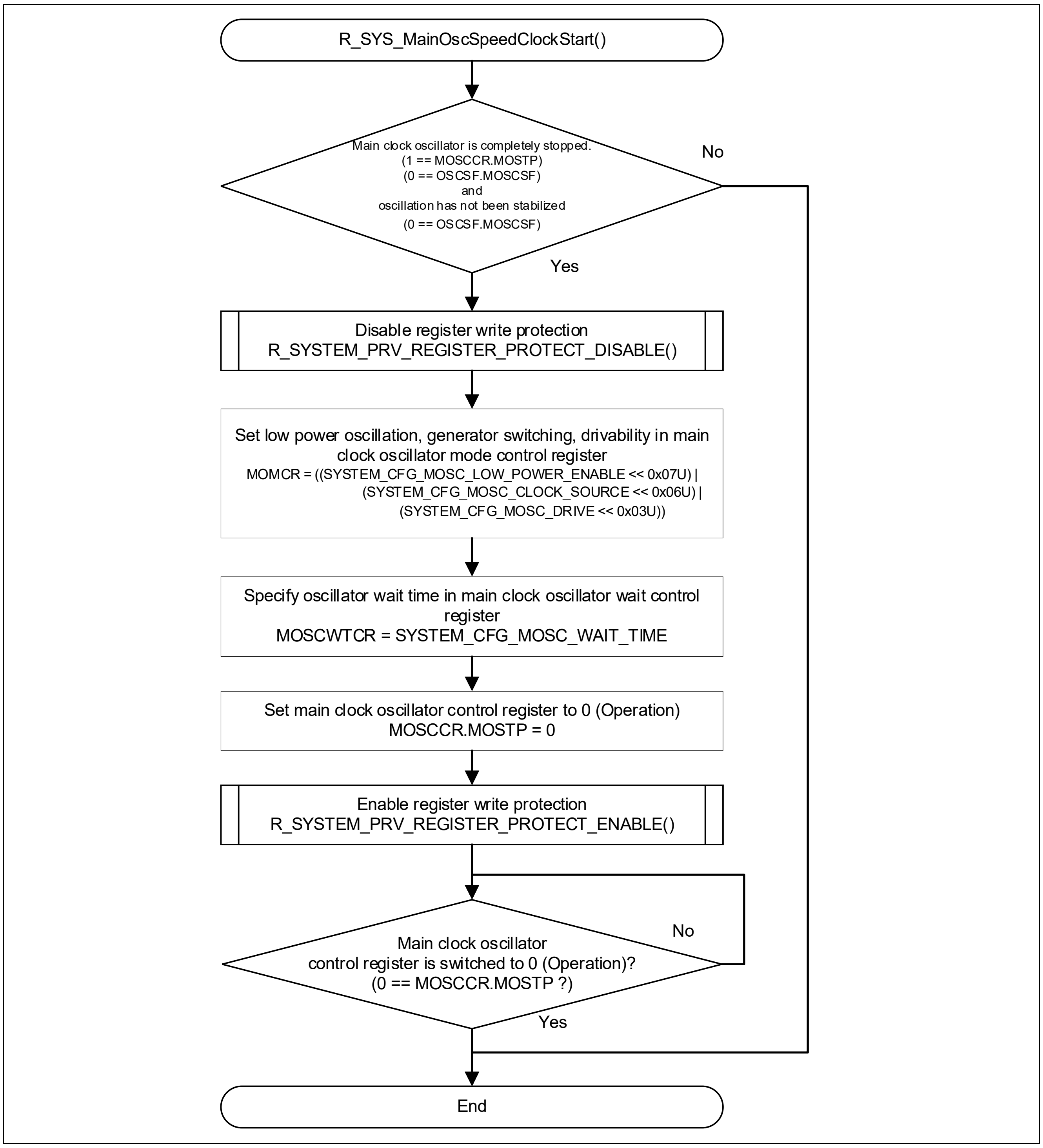
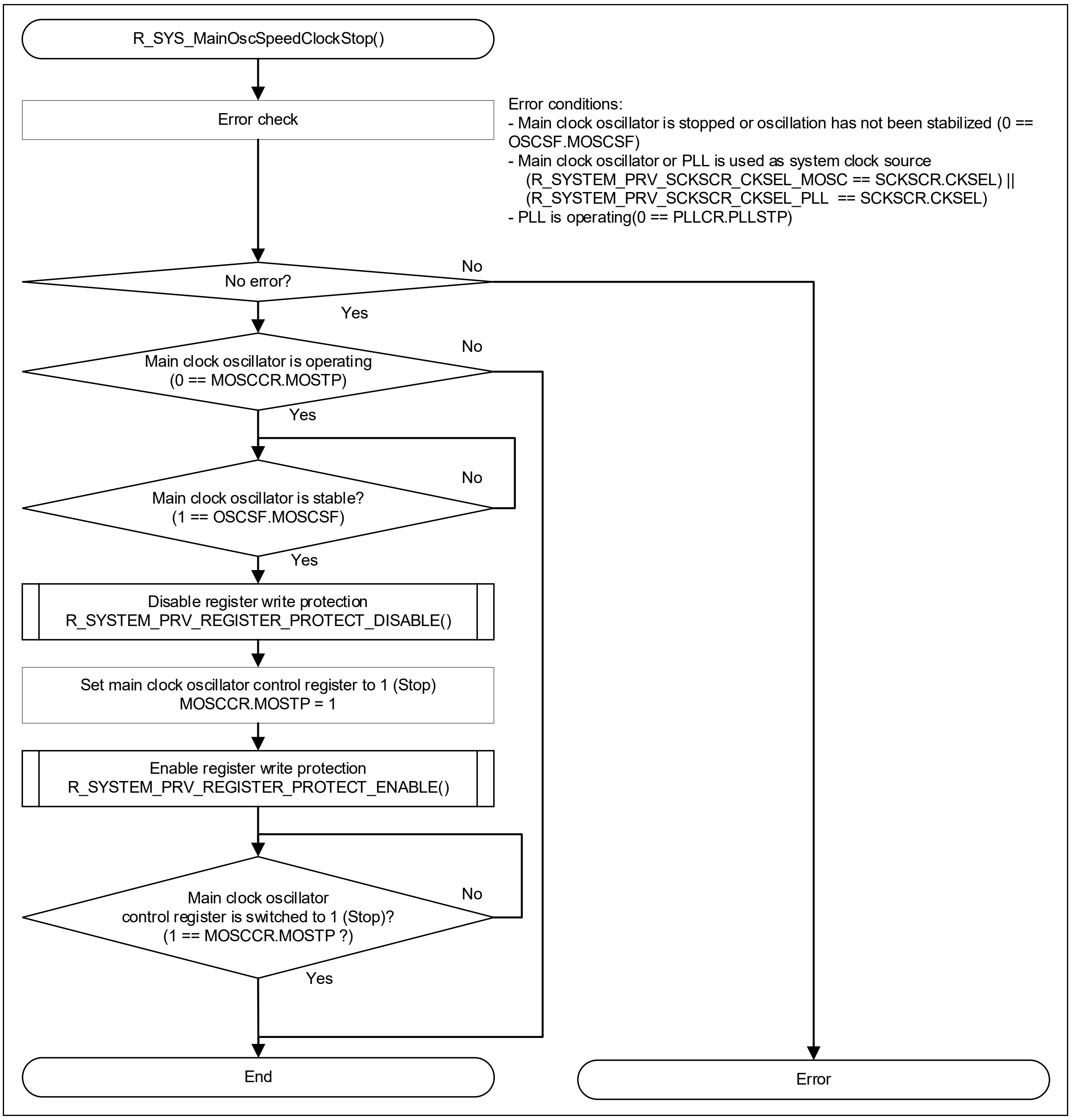

R_SYS_MainOscSpeedClockStart()
Main clock oscillator is completely stopped. (1 == MOSCCR.MOSTP) (0 == OSCSF.MOSCSF) and oscillation has not been stabilized (0 == OSCSF.MOSCSF)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set low power oscillation, generator switching, drivability in main clock oscillator mode control register
MOMCR = ((SYSTEM_CFG_MOSC_LOW_POWER_ENABLE << 0x07U) | (SYSTEM_CFG_MOSC_CLOCK_SOURCE << 0x06U) | (SYSTEM_CFG_MOSC_DRIVE << 0x03U))
Specify oscillator wait time in main clock oscillator wait control register
MOSCWTCR = SYSTEM_CFG_MOSC_WAIT_TIME
Set main clock oscillator control register to 0 (Operation)
MOSCCR.MOSTP = 0
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Main clock oscillator control register is switched to 0 (Operation)? (0 == MOSCCR.MOSTP ?)
No
Yes
End

Figure 4.23 R_SYS_MainOscSpeedClockStart Function Processing Flow

4.3.18 R_SYS_MainOscSpeedClockStop Function

Table 4-27 R_SYS_MainOscSpeedClockStop Function Specifications

Format	int32_t R_SYS_MainOscSpeedClockStop(void)
Description	Stops the operation of the main clock oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–



R_SYS_MainOscSpeedClockStop()
Error check
Error conditions:
- Main clock oscillator is stopped or oscillation has not been stabilized (0 == OSCSF.MOSCSF)
- Main clock oscillator or PLL is used as system clock source
(R_SYSTEM_PRV_SCKSCR_CKSEL_MOSC == SCKSCR.CKSEL) ||
(R_SYSTEM_PRV_SCKSCR_CKSEL_PLL == SCKSCR.CKSEL)
- PLL is operating(0 == PLLCR.PLLSTP)
No error?
No
Yes
Main clock oscillator is operating (0 == MOSCCR.MOSTP)
No
Yes
Main clock oscillator is stable? (1 == OSCSF.MOSCSF)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set main clock oscillator control register to 1 (Stop) MOSCCR.MOSTP = 1
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Main clock oscillator control register is switched to 1 (Stop)? (1 == MOSCCR.MOSTP ?)
No
Yes
End
Error

Figure 4.24 R_SYS_MainOscSpeedClockStop Function Processing Flow

4.3.19 R_SYS_HighSpeedClockStart Function

Table 4-28 R_SYS_HighSpeedClockStart Function Specifications

Format	int32_t R_SYS_HighSpeedClockStart(void)
Description	Starts the operation of the high-speed on-chip oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

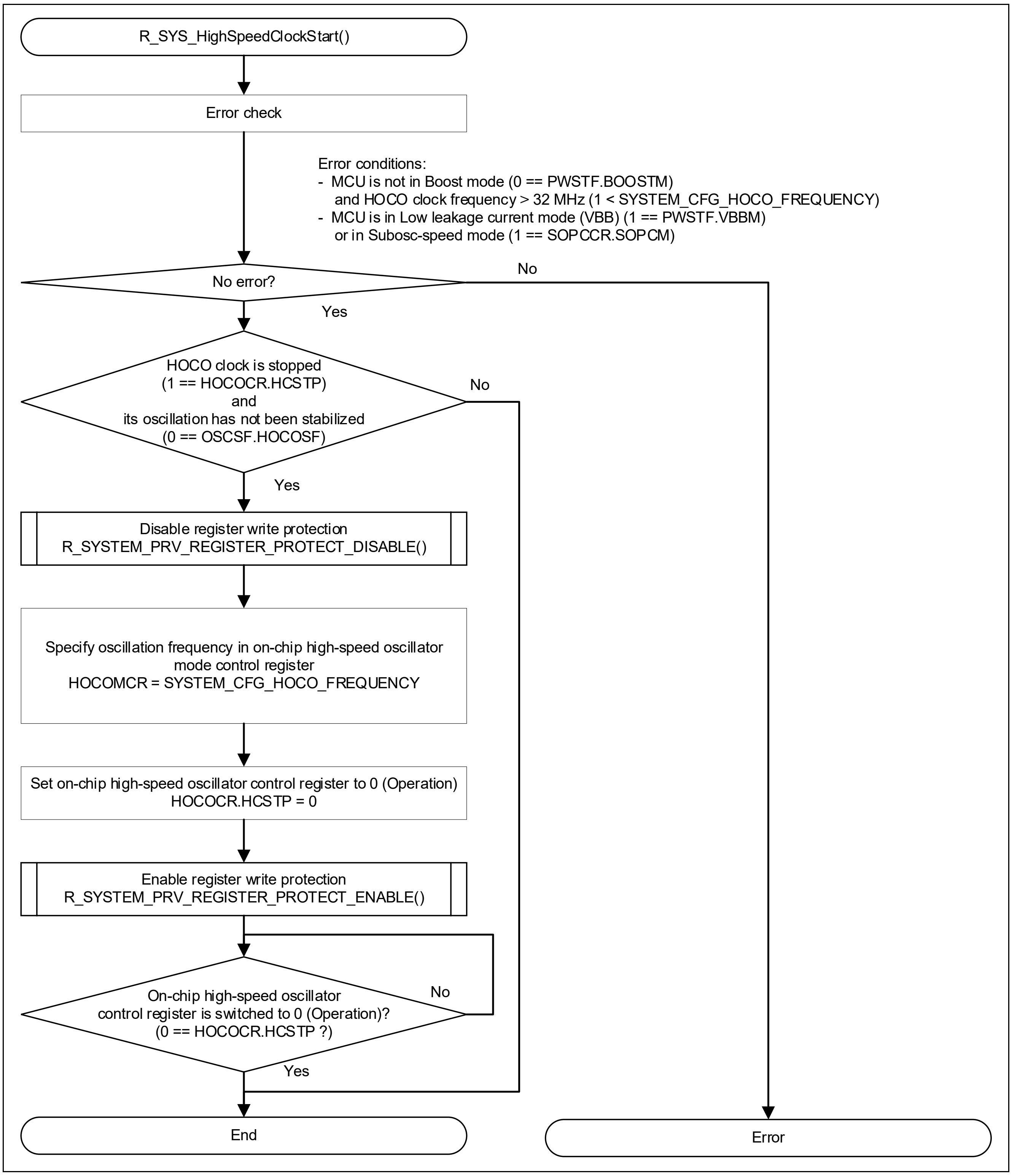

R_SYS_HighSpeedClockStart()
Error check
Error conditions:
- MCU is not in Boost mode (0 == PWSTF.BOOSTM) and HOCO clock frequency > 32 MHz (1 < SYSTEM_CFG_HOCO_FREQUENCY)
- MCU is in Low leakage current mode (VBB) (1 == PWSTF.VBBM) or in Subosc-speed mode (1 == SOPCCR.SOPCM)
No error?
No
Yes
HOCO clock is stopped (1 == HOCOCR.HCSTP) and its oscillation has not been stabilized (0 == OSCSF.HOCOSF)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Specify oscillation frequency in on-chip high-speed oscillator mode control register HOCOMCR = SYSTEM_CFG_HOCO_FREQUENCY
Set on-chip high-speed oscillator control register to 0 (Operation) HOCOCR.HCSTP = 0
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip high-speed oscillator control register is switched to 0 (Operation)? (0 == HOCOCR.HCSTP ?)
No
Yes
End
Error

Figure 4.25 R_SYS_HighSpeedClockStart Function Processing Flow

4.3.20 R_SYS_HighSpeedClockStop Function

Table 4-29 R_SYS_HighSpeedClockStop Function Specifications

Format	int32_t R_SYS_HighSpeedClockStop(void)
Description	Stops the operation of the high-speed on-chip oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

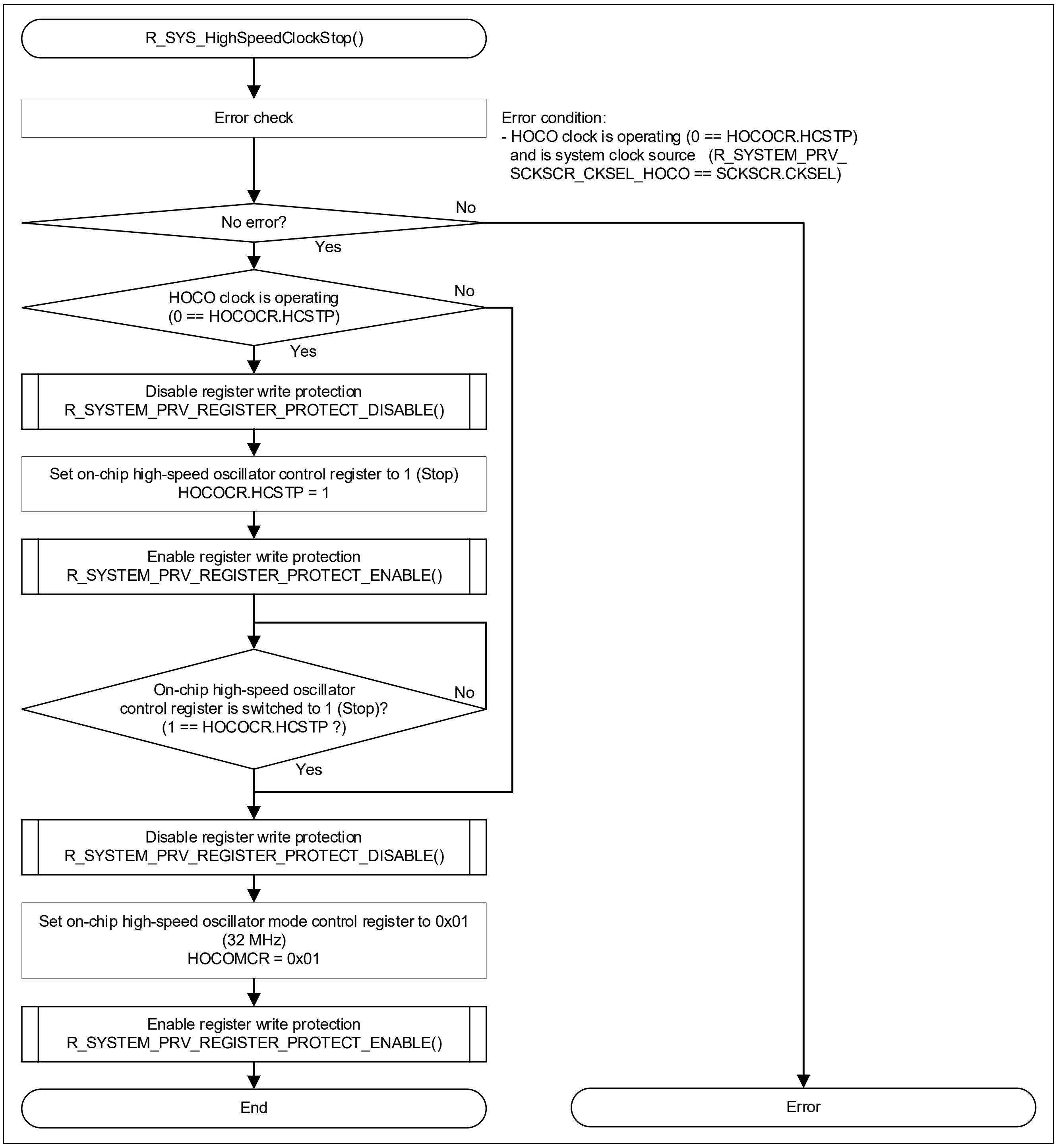

R_SYS_HighSpeedClockStop()
Error check
Error condition:
- HOCO clock is operating (0 == HOCOCR.HCSTP) and is system clock source (R_SYSTEM_PRV_SCKSCR_CKSEL_HOCO == SCKSCR.CKSEL)
No error?
No
Yes
HOCO clock is operating (0 == HOCOCR.HCSTP)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip high-speed oscillator control register to 1 (Stop) HOCOCR.HCSTP = 1
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip high-speed oscillator control register is switched to 1 (Stop)? (1 == HOCOCR.HCSTP ?)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip high-speed oscillator mode control register to 0x01 (32 MHz) HOCOMCR = 0x01
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
End
Error

Figure 4.26 R_SYS_HighSpeedClockStop Function Processing Flow

4.3.21 R_SYS_MediumSpeedClockStart Function

Table 4-30 R_SYS_MediumSpeedClockStart Function Specifications

Format	int32_t R_SYS_MediumSpeedClockStart(void)
Description	Starts the operation of the middle-speed on-chip oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

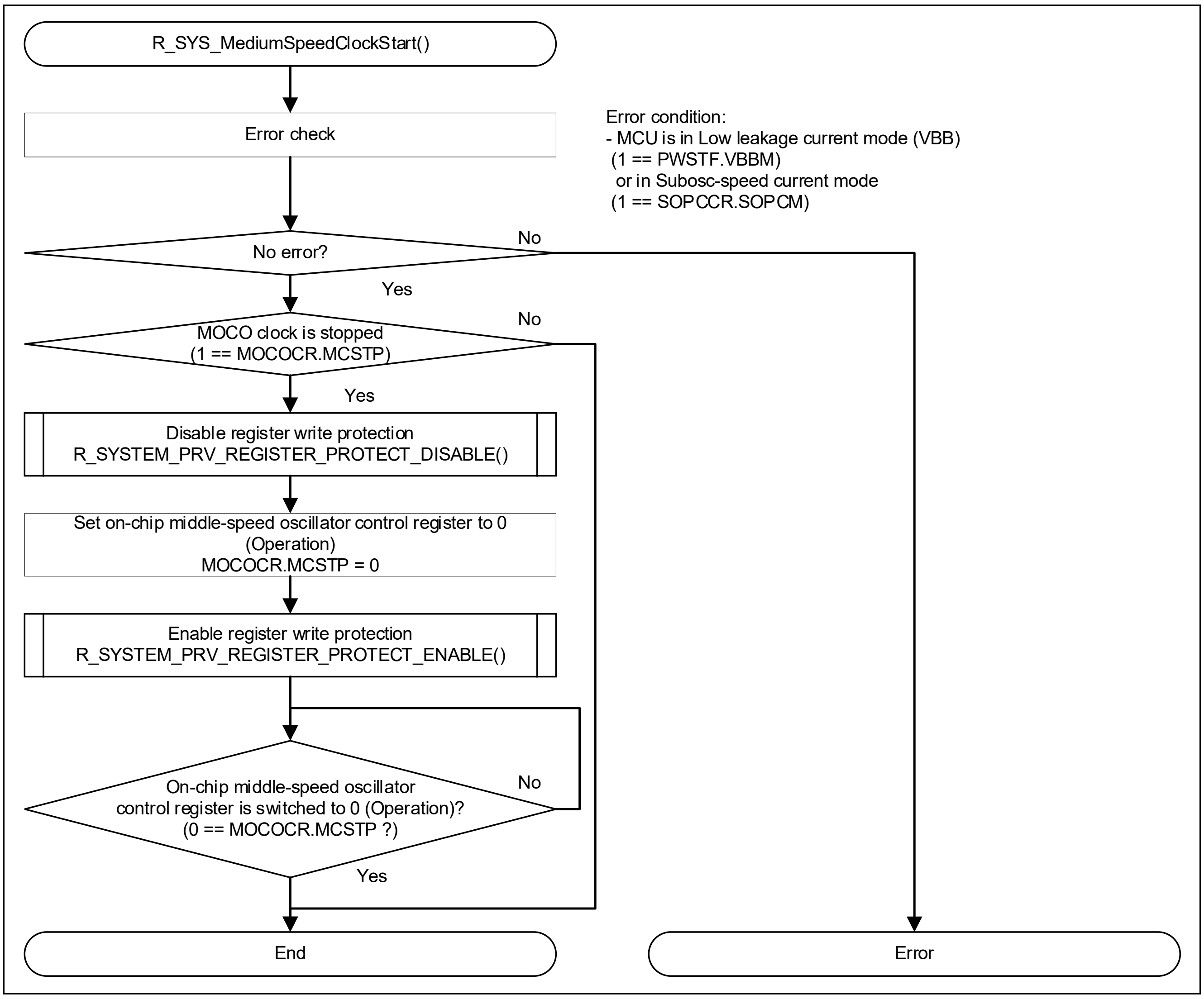

R_SYS_MediumSpeedClockStart()
Error check
Error condition:
- MCU is in Low leakage current mode (VBB)
(1 == PWSTF.VBBM)
or in Subosc-speed current mode
(1 == SOPCCR.SOPCM)
No error?
No
Yes
MOCO clock is stopped
(1 == MOCOCR.MCSTP)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip middle-speed oscillator control register to 0
(Operation)
MOCOCR.MCSTP = 0
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip middle-speed oscillator
control register is switched to 0 (Operation)?
(0 == MOCOCR.MCSTP ?)
No
Yes
End
Error

Figure 4.27 R_SYS_MediumSpeedClockStart Function Processing Flow

4.3.22 R_SYS_MediumSpeedClockStop Function

Table 4-31 R_SYS_MediumSpeedClockStop Function Specifications

Format	int32_t R_SYS_MediumSpeedClockStop(void)
Description	Stops the operation of the middle-speed on-chip oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

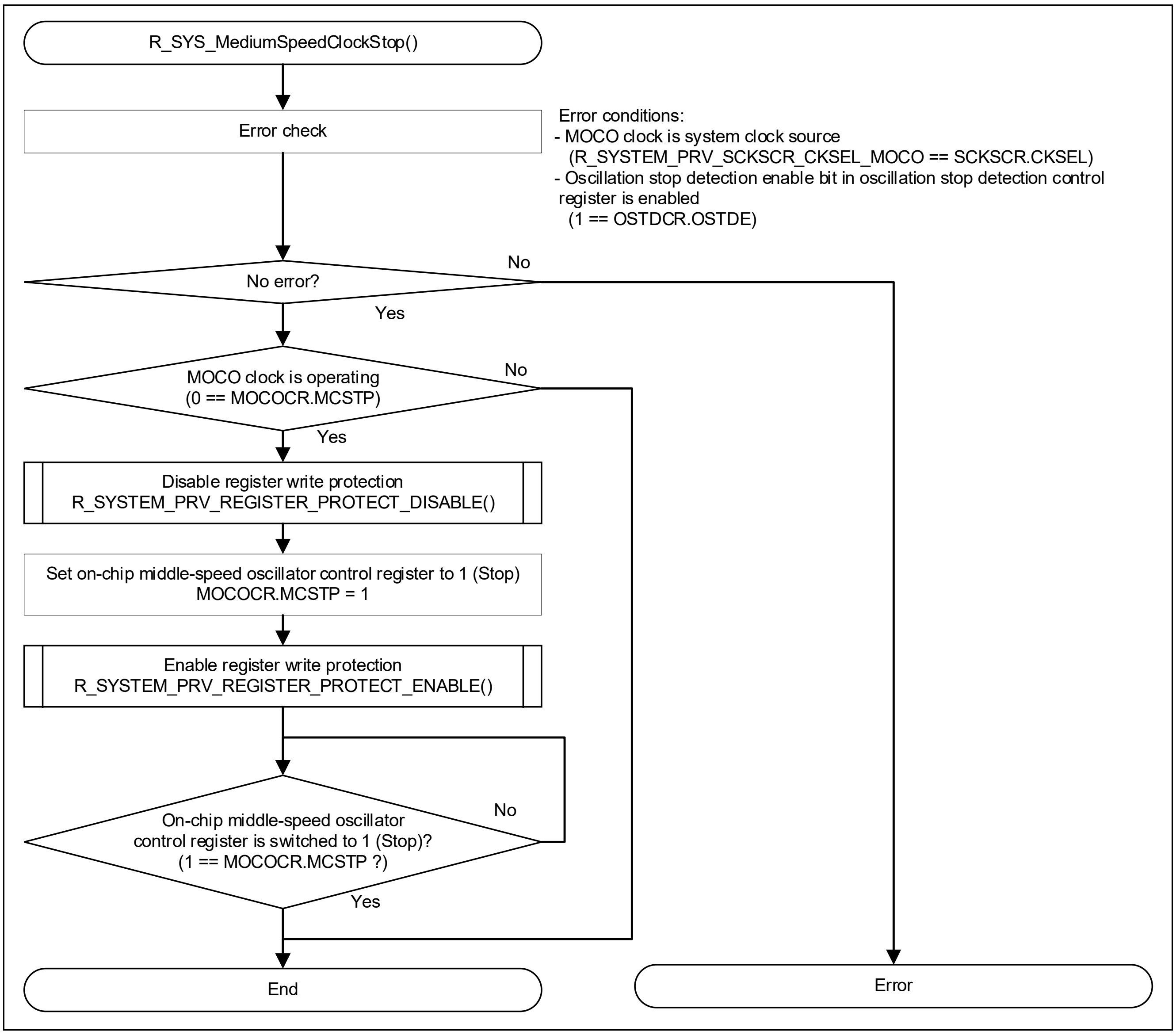

R_SYS_MediumSpeedClockStop()
Error check
Error conditions:
- MOCO clock is system clock source
(R_SYSTEM_PRV_SCKSCR_CKSEL_MOCO == SCKSCR.CKSEL)
- Oscillation stop detection enable bit in oscillation stop detection control register is enabled
(1 == OSTDCR.OSTDE)
No error?
No
Yes
MOCO clock is operating
(0 == MOCOCR.MCSTP)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip middle-speed oscillator control register to 1 (Stop)
MOCOCR.MCSTP = 1
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip middle-speed oscillator control register is switched to 1 (Stop)?
(1 == MOCOCR.MCSTP ?)
No
Yes
End
Error

Figure 4.28 R_SYS_MediumSpeedClockStop Function Processing Flow

4.3.23 R_SYS_LowSpeedClockStart Function

Table 4-32 R_SYS_LowSpeedClockStart Function Specifications

Format	void R_SYS_LowSpeedClockStart(void)
Description	Starts the operation of the low-speed on-chip oscillator.
Argument	None
Return value	None
Remarks	–

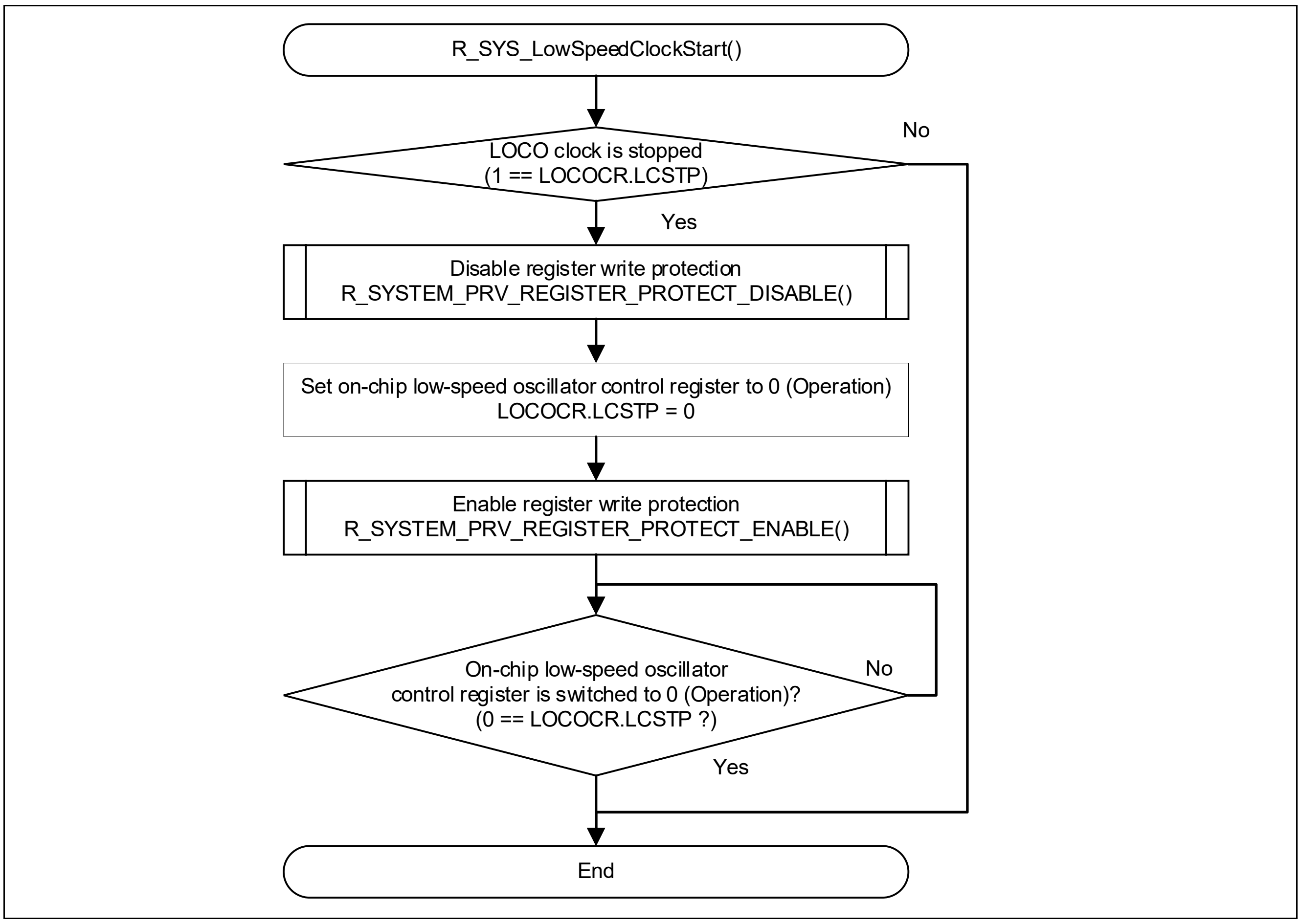

R_SYS_LowSpeedClockStart()
LOCO clock is stopped
(1 == LOCOCR.LCSTP)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip low-speed oscillator control register to 0 (Operation)
LOCOCR.LCSTP = 0
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip low-speed oscillator
control register is switched to 0 (Operation)?
(0 == LOCOCR.LCSTP ?)
No
Yes
End

Figure 4.29 R_SYS_LowSpeedClockStart Function Processing Flow

4.3.24 R_SYS_LowSpeedClockStop Function

Table 4-33 R_SYS_LowSpeedClockStop Function Specifications

Format	int32_t R_SYS_LowSpeedClockStop(void)
Description	Stops the operation of the low-speed on-chip oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

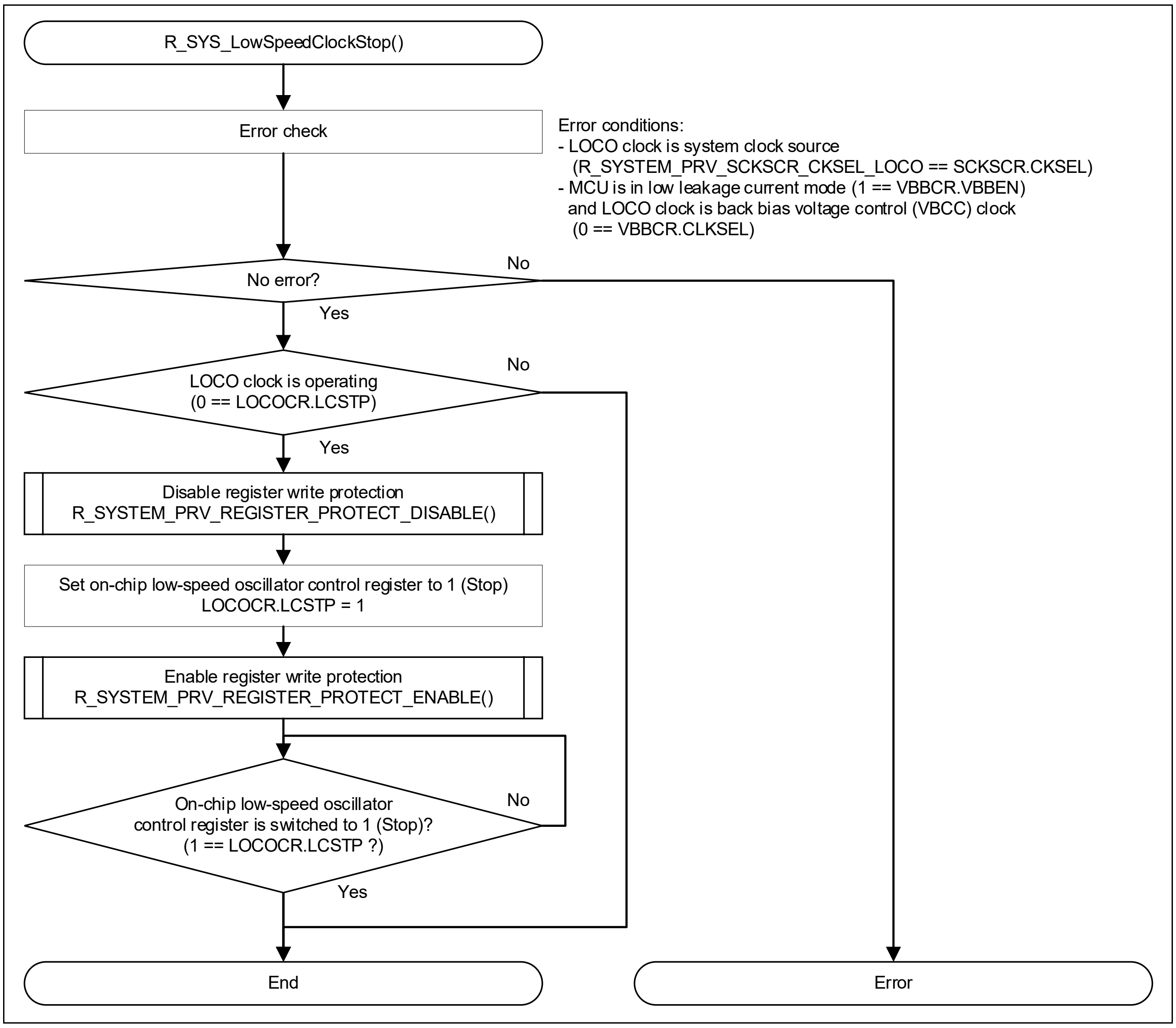
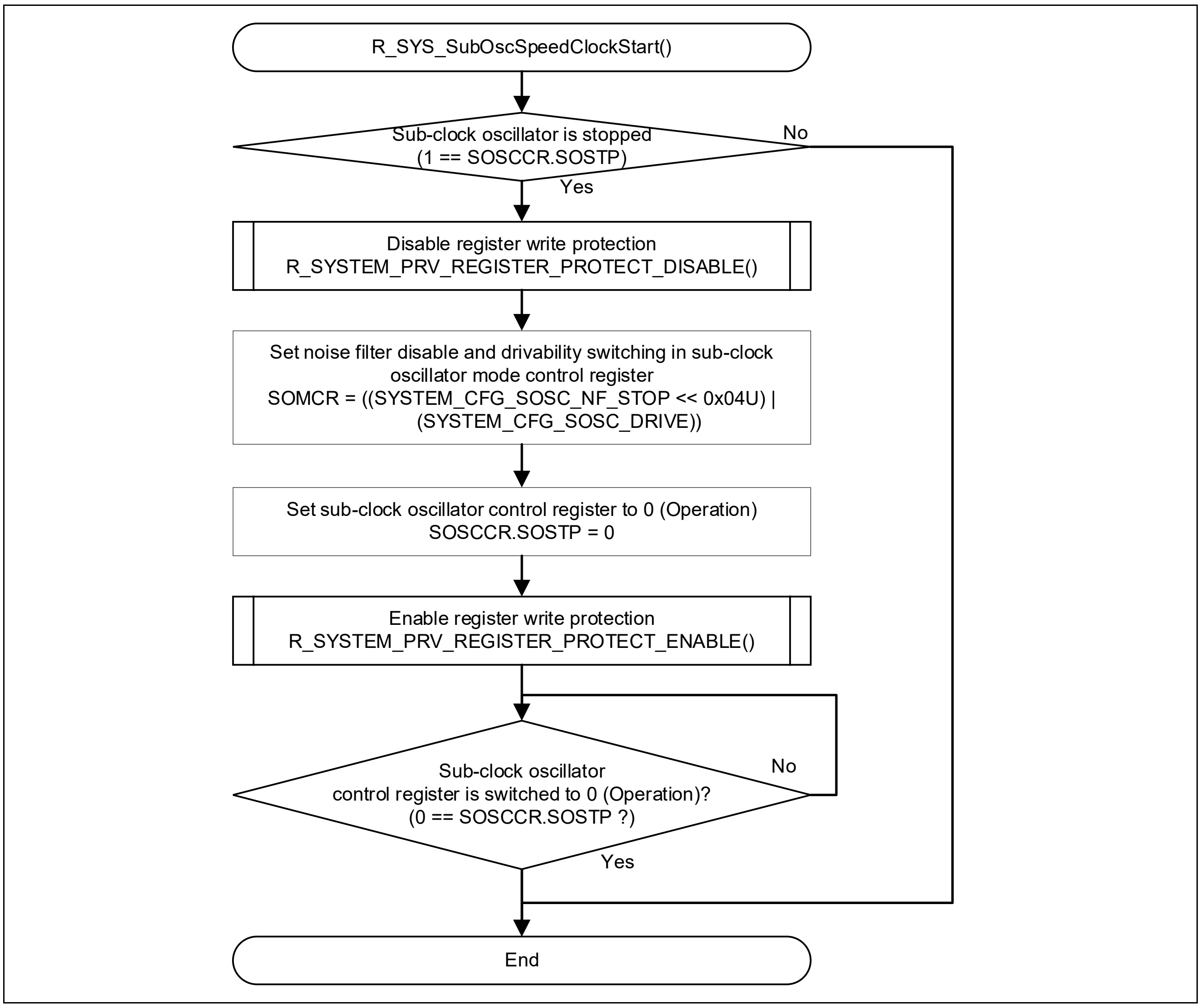

R_SYS_LowSpeedClockStop()
Error check
Error conditions:
- LOCO clock is system clock source
(R_SYSTEM_PRV_SCKSCR_CKSEL_LOCO == SCKSCR.CKSEL)
- MCU is in low leakage current mode (1 == VBBCR.VBBEN)
and LOCO clock is back bias voltage control (VBCC) clock
(0 == VBBCR.CLKSEL)
No error?
No
Yes
LOCO clock is operating
(0 == LOCOCR.LCSTP)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set on-chip low-speed oscillator control register to 1 (Stop)
LOCOCR.LCSTP = 1
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
On-chip low-speed oscillator
control register is switched to 1 (Stop)?
(1 == LOCOCR.LCSTP ?)
No
Yes
End
Error

Figure 4.30 R_SYS_LowSpeedClockStop Function Processing Flow

4.3.25 R_SYS_SubOscSpeedClockStart Function

Table 4-34 R_SYS_SubOscSpeedClockStart Function Specifications

Format	void R_SYS_SubOscSpeedClockStart(void)
Description	Starts the operation of the sub-clock oscillator.
Argument	None
Return value	None
Remarks	–



R_SYS_SubOscSpeedClockStart()
Sub-clock oscillator is stopped (1 == SOSCCR.SOSTP)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set noise filter disable and drivability switching in sub-clock oscillator mode control register
SOMCR = ((SYSTEM_CFG_SOSC_NF_STOP << 0x04U) | (SYSTEM_CFG_SOSC_DRIVE))
Set sub-clock oscillator control register to 0 (Operation) SOSCCR.SOSTP = 0
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Sub-clock oscillator control register is switched to 0 (Operation)? (0 == SOSCCR.SOSTP ?)
No
Yes
End

Figure 4.31 R_SYS_SubOscSpeedClockStart Function Processing Flow

4.3.26 R_SYS_SubOscSpeedClockStop Function

Table 4-35 R_SYS_SubOscSpeedClockStop Function Specifications

Format	int32_t R_SYS_SubOscSpeedClockStop(void)
Description	Stops the operation of the sub-clock oscillator.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

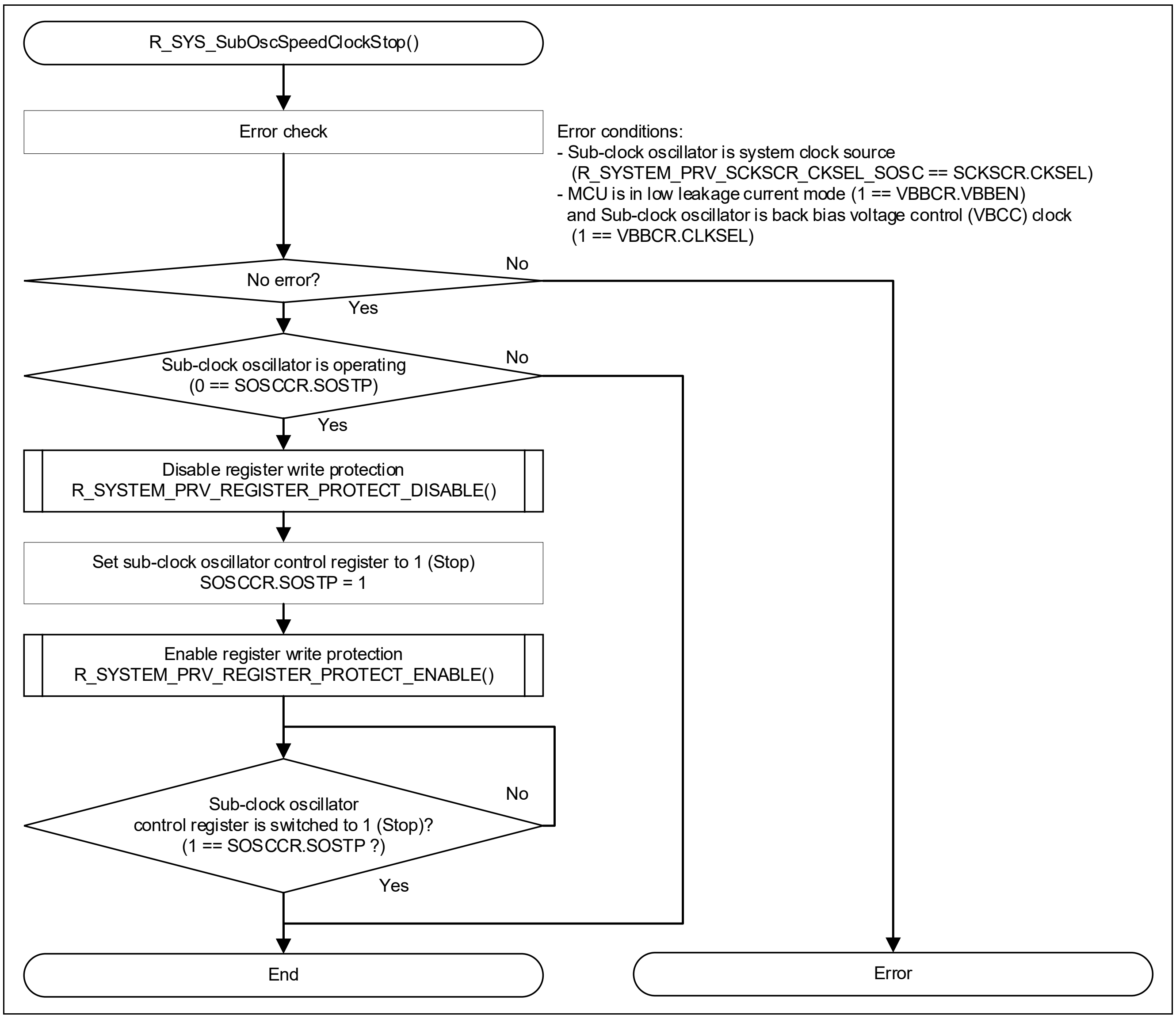

R_SYS_SubOscSpeedClockStop()
Error check
Error conditions:
- Sub-clock oscillator is system clock source (R_SYSTEM_PRV_SCKSCR_CKSEL_SOSC == SCKSCR.CKSEL)
- MCU is in low leakage current mode (1 == VBBCR.VBBEN) and Sub-clock oscillator is back bias voltage control (VBCC) clock (1 == VBBCR.CLKSEL)
No error?
No
Yes
Sub-clock oscillator is operating (0 == SOSCCR.SOSTP)
No
Yes
Disable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set sub-clock oscillator control register to 1 (Stop) SOSCCR.SOSTP = 1
Enable register write protection R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Sub-clock oscillator control register is switched to 1 (Stop)? (1 == SOSCCR.SOSTP ?)
No
Yes
End
Error

Figure 4.32 R_SYS_SubOscSpeedClockStop Function Processing Flow

4.3.27 R_SYS_PLLSpeedClockStart Function

Table 4-36 R_SYS_PLLSpeedClockStart Function Specifications

Format	int32_t R_SYS_PLLSpeedClockStart(void)
Description	Starts the operation of the PLL circuit.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

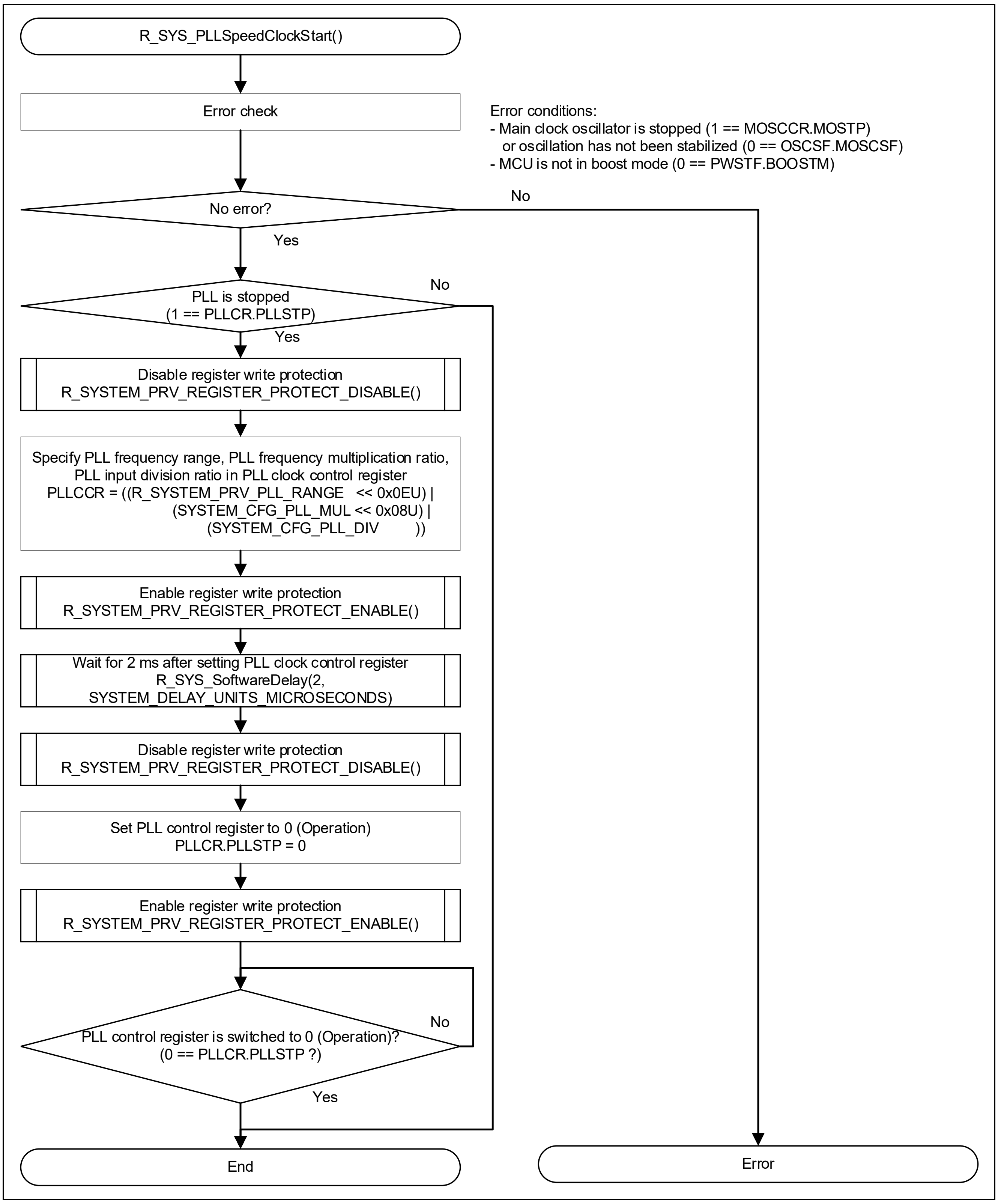

R_SYS_PLLSpeedClockStart()
Error check
Error conditions:
- Main clock oscillator is stopped (1 == MOSCCR.MOSTP)
or oscillation has not been stabilized (0 == OSCSF.MOSCSF)
- MCU is not in boost mode (0 == PWSTF.BOOSTM)
No error?
No
Yes
PLL is stopped
(1 == PLLCR.PLLSTP)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Specify PLL frequency range, PLL frequency multiplication ratio,
PLL input division ratio in PLL clock control register
PLLCCR = ((R_SYSTEM_PRV_PLL_RANGE << 0x0EU) |
(SYSTEM_CFG_PLL_MUL << 0x08U) |
(SYSTEM_CFG_PLL_DIV))
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
Wait for 2 ms after setting PLL clock control register
R_SYS_SoftwareDelay(2,
SYSTEM_DELAY_UNITS_MICROSECONDS)
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set PLL control register to 0 (Operation)
PLLCR.PLLSTP = 0
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
PLL control register is switched to 0 (Operation)?
(0 == PLLCR.PLLSTP ?)
No
Yes
End
Error

Figure 4.33 R_SYS_PLLSpeedClockStart Function Processing Flow

4.3.28 R_SYS_PLLSpeedClockStop Function

Table 4-37 R_SYS_PLLSpeedClockStop Function

Format	int32_t R_SYS_PLLSpeedClockStop(void)
Description	Stops the operation of the PLL circuit.
Argument	None
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

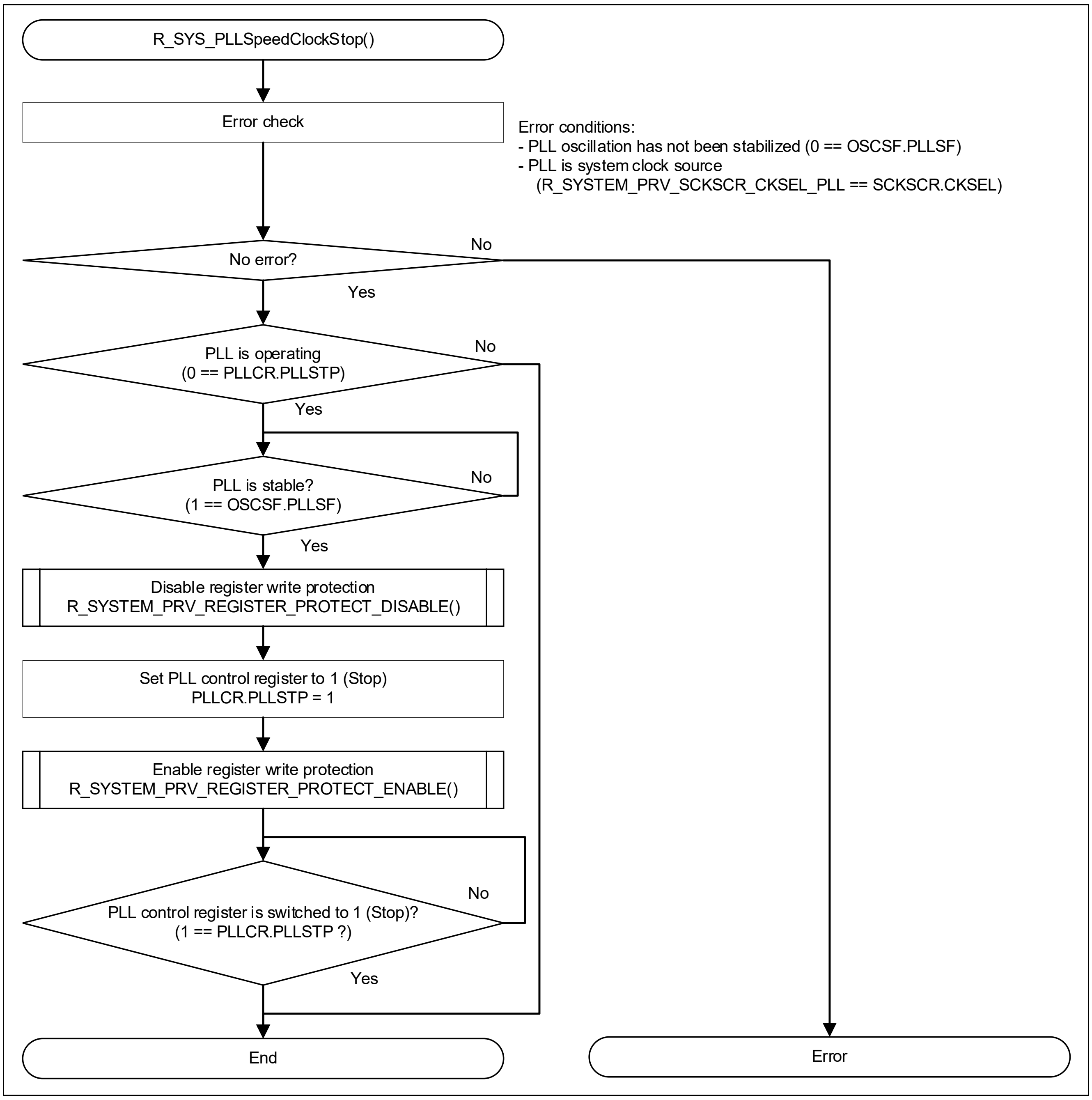

R_SYS_PLLSpeedClockStop()
Error check
Error conditions:
- PLL oscillation has not been stabilized (0 == OSCSF.PLLSF)
- PLL is system clock source
(R_SYSTEM_PRV_SCKSCR_CKSEL_PLL == SCKSCR.CKSEL)
No error?
No
Yes
PLL is operating
(0 == PLLCR.PLLSTP)
No
Yes
PLL is stable?
(1 == OSCSF.PLLSF)
No
Yes
Disable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_DISABLE()
Set PLL control register to 1 (Stop)
PLLCR.PLLSTP = 1
Enable register write protection
R_SYSTEM_PRV_REGISTER_PROTECT_ENABLE()
PLL control register is switched to 1 (Stop)?
(1 == PLLCR.PLLSTP ?)
No
Yes
End
Error

Figure 4.34 R_SYS_PLLSpeedClockStop Function Processing Flow

4.3.29 R_SYS_OscStabilizationFlagGet Function

Table 4-38 R_SYS_OscStabilizationFlagGet Function

Format	uint8_t R_SYS_OscStabilizationFlagGet(void)
Description	Obtains the value of the OSCSF register.
Argument	None
Return value	uint8_t: Returns the value of the OSCSF register.
Remarks	–

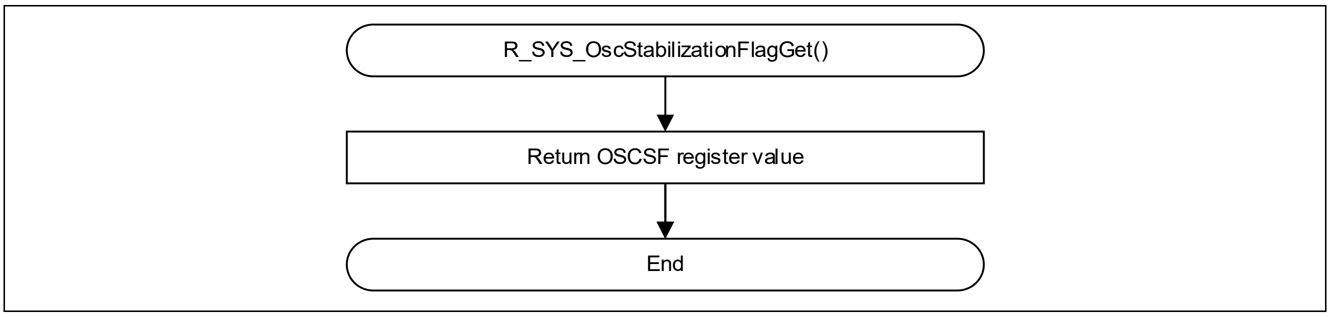


Figure 4.35 R_SYS_OscStabilizationFlagGet Function Processing Flow

4.3.30 R_SYS_IrqEventLinkSet Function

Table 4-39 R_SYS_IrqEventLinkSet Function Specifications

Format	int32_t R_SYS_IrqEventLinkSet(IRQn_Type irq, uint32_t iels_value, system_int_cb_t callback)
Description	Registers an interrupt handler as a callback function. This callback function is called from the interrupt handler of a specified IELx_IRQn number.
Argument	IRQn_Type irq [Input]: Specifies an event link number (0 to 31).
	uint32_t iels_value [Input]: Specifies a value of the event link signal to set in the IELSRn.IELS register.
	system_int_cb_t callback [Input]: Specifies a callback function.
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

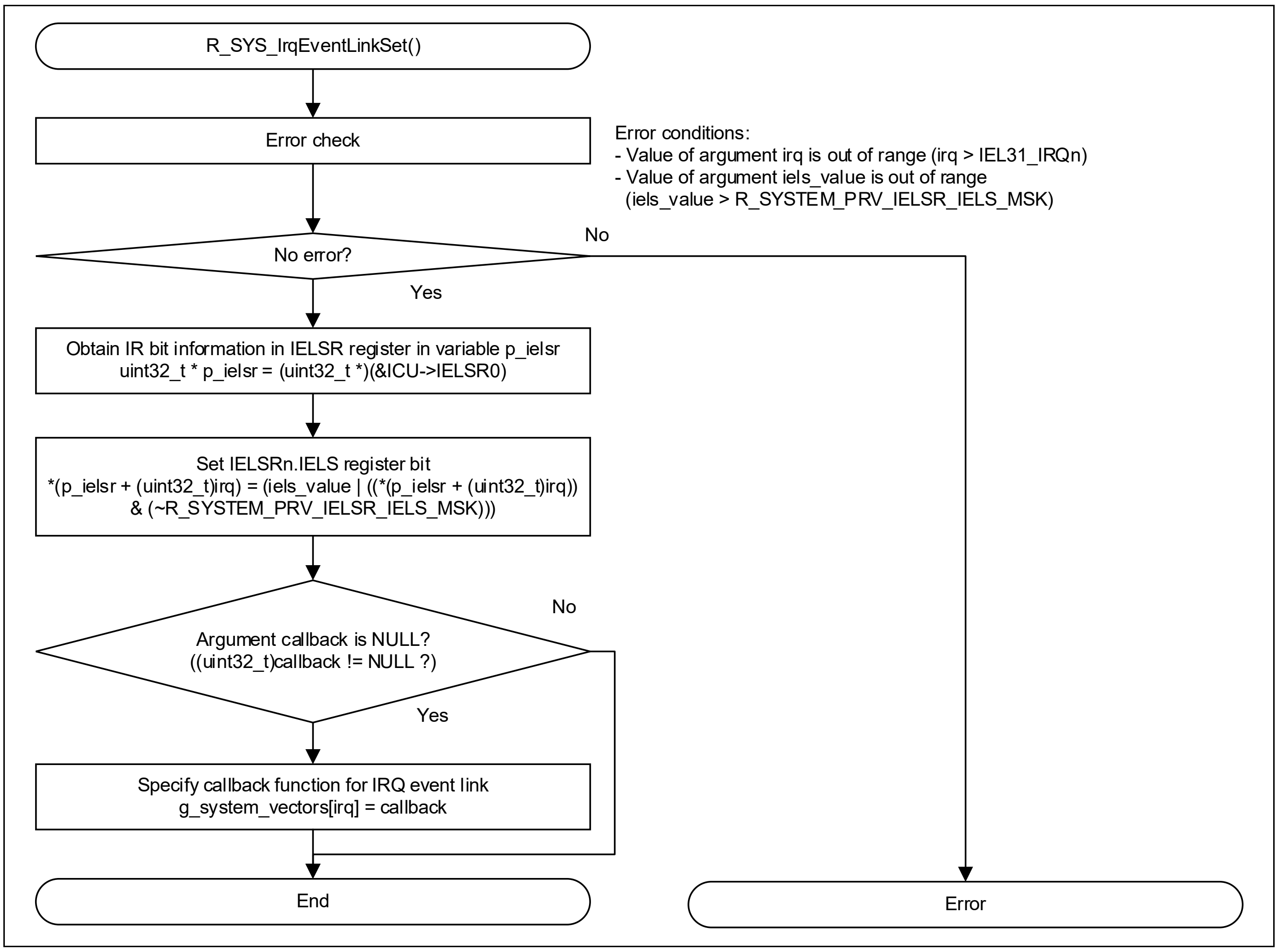

R_SYS_IrqEventLinkSet()
Error check
Error conditions:
- Value of argument irq is out of range (irq > IEL31_IRQn)
- Value of argument iels_value is out of range (iels_value > R_SYSTEM_PRV_IELSR_IELS_MSK)
No error?
No
Yes
Obtain IR bit information in IELSR register in variable p_ielsr
uint32_t * p_ielsr = (uint32_t *)(&ICU->IELSR0)
Set IELSRn.IELS register bit
(p_ielsr + (uint32_t)irq) = (iels_value | (((p_ielsr + (uint32_t)irq)) & (~R_SYSTEM_PRV_IELSR_IELS_MSK)))
Argument callback is NULL?
((uint32_t)callback != NULL ?)
No
Yes
Specify callback function for IRQ event link
g_system_vectors[irq] = callback
End
Error

Figure 4.36 R_SYS_IrqEventLinkSet Function Processing Flow

4.3.31 R_SYS_IrqStatusGet Function

Table 4-40 R_SYS_IrqStatusGet Function Specifications

Format	int32_t R_SYS_IrqStatusGet(IRQn_Type irq, uint8_t * p_ir)
Description	Obtains the status of the IR flag of a specified IELx_IRQn number.
Argument	IRQn_Type irq [Input]: Specifies an event link number (0 to 31).
	uint8_t * p_ir [Input]: Specifies the location for storing the obtained IR flag.
Return value	Normal (0)
	Abnormal (-1)
Remarks	–

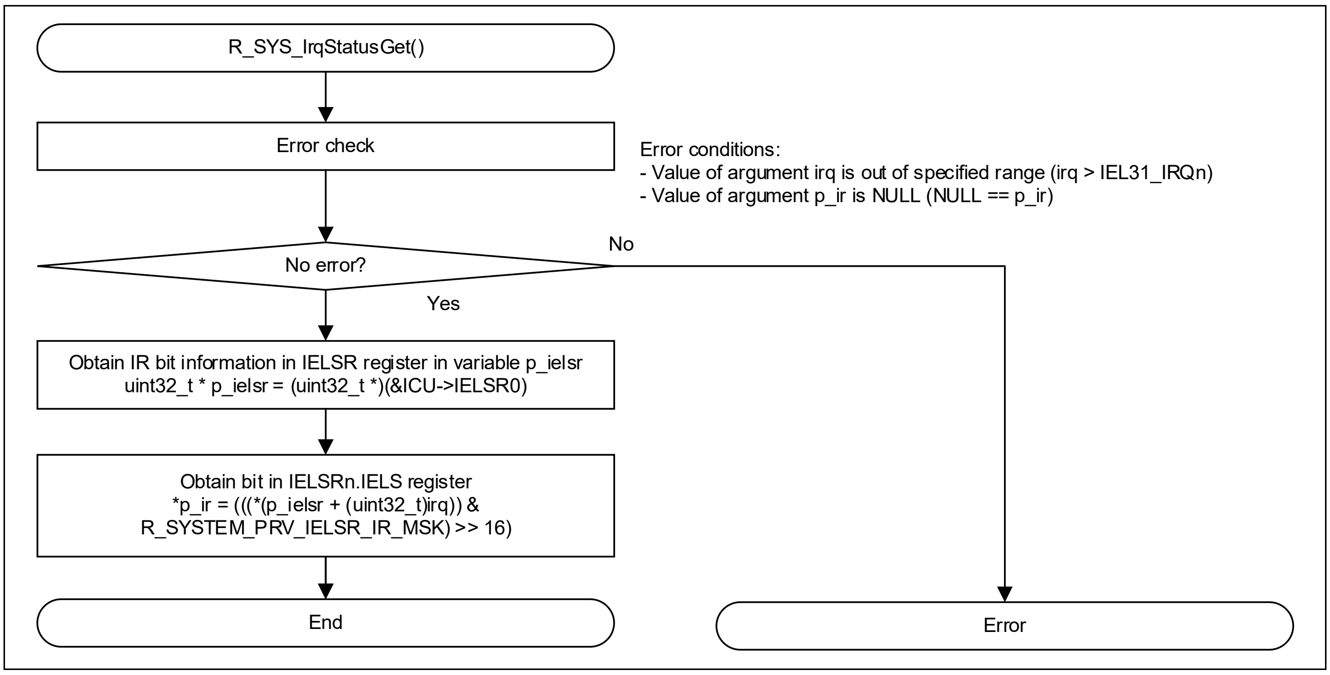
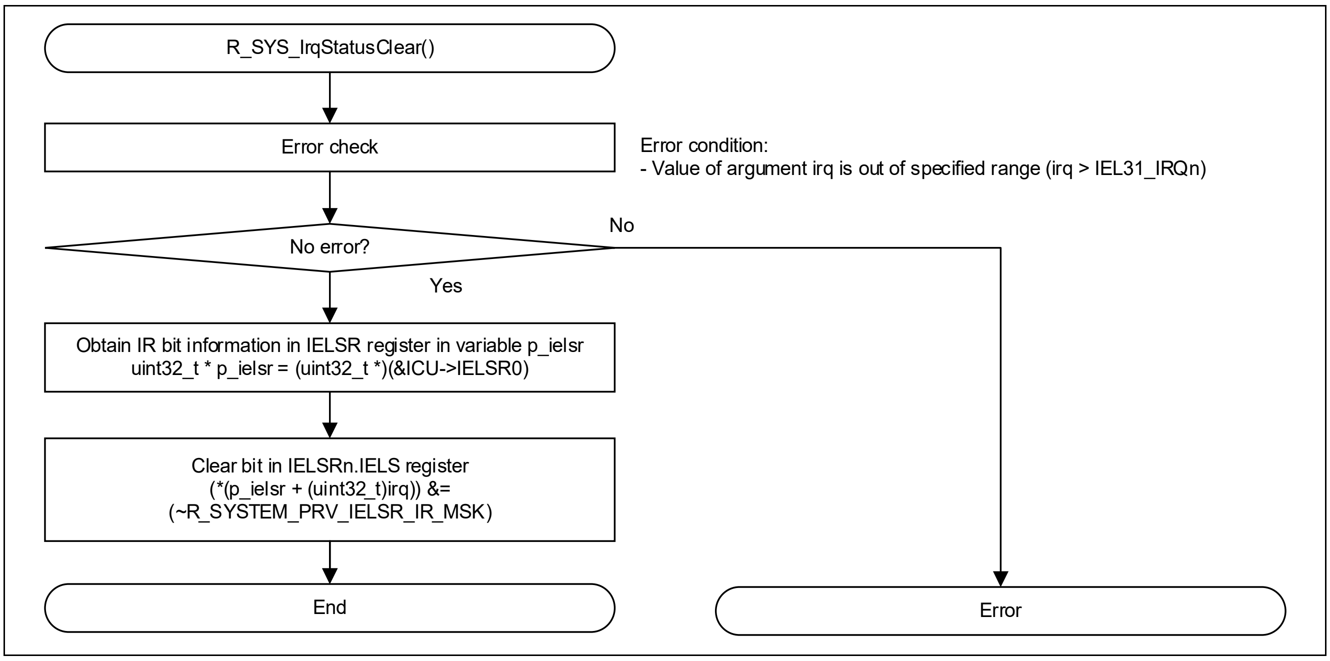

R_SYS_IrqStatusGet()
Error check
Error conditions:
- Value of argument irq is out of specified range (irq > IEL31_IRQn)
- Value of argument p_ir is NULL (NULL == p_ir)
No error?
No
Yes
Obtain IR bit information in IELSR register in variable p_ielsr
uint32_t * p_ielsr = (uint32_t *)(&ICU->IELSR0)
Obtain bit in IELSRn.IELS register
p_ir = ((((p_ielsr + (uint32_t)irq)) &
R_SYSTEM_PRV_IELSR_IR_MSK) >> 16)
End
Error

Figure 4.37 R_SYS_IrqStatusGet Function Processing Flow

4.3.32 R_SYS_IrqStatusClear Function

Table 4-41 R_SYS_IrqStatusClear Function Specifications

Format	int32_t R_SYS_IrqStatusClear(IRQn_Type irq)
Description	Clears the status of the IR flag of a specified IELx_IRQn number.
Argument	IRQn_Type irq [Input]: Specifies an event link number (0 to 31).
Return value	Normal (0)
	Abnormal (-1)
Remarks	–


R_SYS_IrqStatusClear()
Error check
Error condition:
- Value of argument irq is out of specified range (irq > IEL31_IRQn)
No error?
No
Yes
Obtain IR bit information in IELSR register in variable p_ielsr
uint32_t * p_ielsr = (uint32_t *)(&ICU->IELSR0)
Clear bit in IELSRn.IELS register
(*(p_ielsr + (uint32_t)irq)) &=
(~R_SYSTEM_PRV_IELSR_IR_MSK)
End
Error
Figure 4.38 R_SYS_IrqStatusClear Function Processing Flow

4.3.33 R_SYS_EnterCriticalSection Function

Table 4-42 R_SYS_EnterCriticalSection Function Specifications

Format	void R_SYS_EnterCriticalSection(void)
Description	Strts prohibiting interrupts
Argument	None
Return value	None
Remarks	–

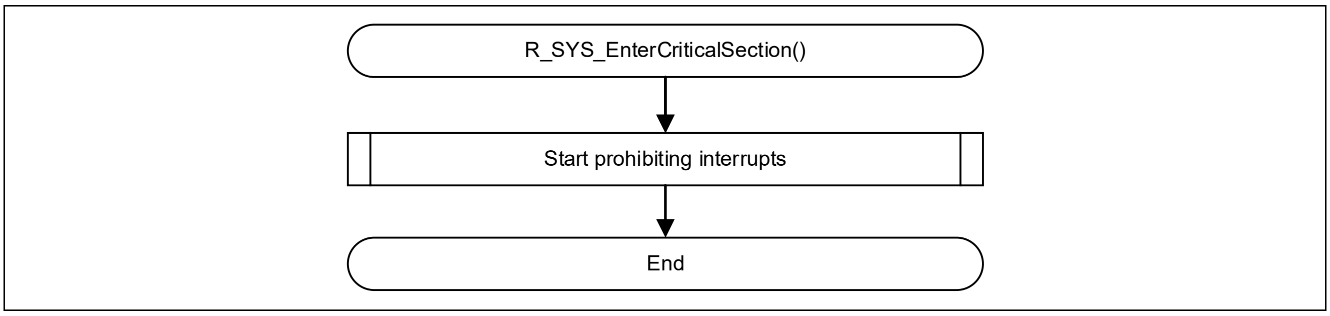


Figure 4.39 R_SYS_EnterCriticalSection Function Processing Flow

4.3.34 R_SYS_ExitCriticalSection Function

Table 4-43 R_SYS_ExitCriticalSection Function Specifications

Format	void R_SYS_ExitCriticalSection(void)
Description	Stops prohibiting interrupts
Argument	None
Return value	None
Remarks	–

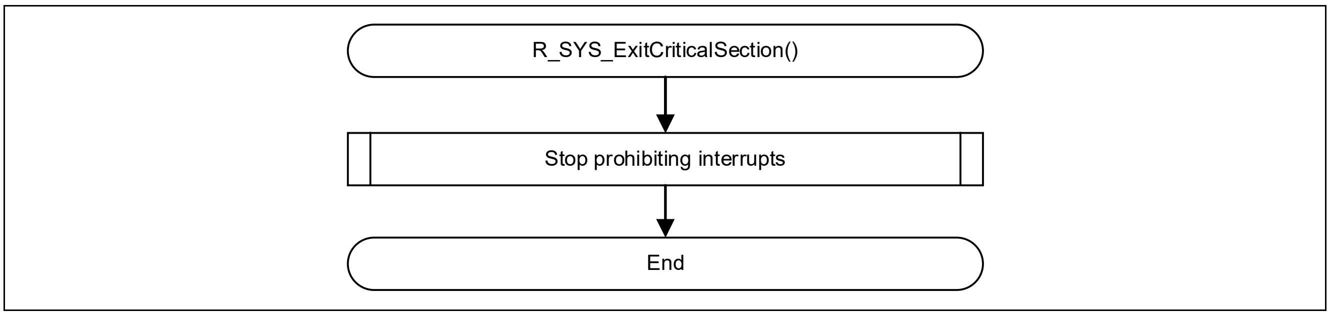


Figure 4.40 R_SYS_ExitCriticalSection Function Processing Flow

4.3.35 R_SYS_ResourceLock Function

Table 4-44 R_SYS_ResourceLock Function Specifications

Format	int32_t R_SYS_ResourceLock(e_system_mcu_lock_t hw_index)
Description	Locks a hardware resource.
Argument	e_system_mcu_lock_t hw_index [Input]: Specifies a hardware resource number.
Return value	Lock succeeded (0)
	Lock failed (-1)
Remarks	–

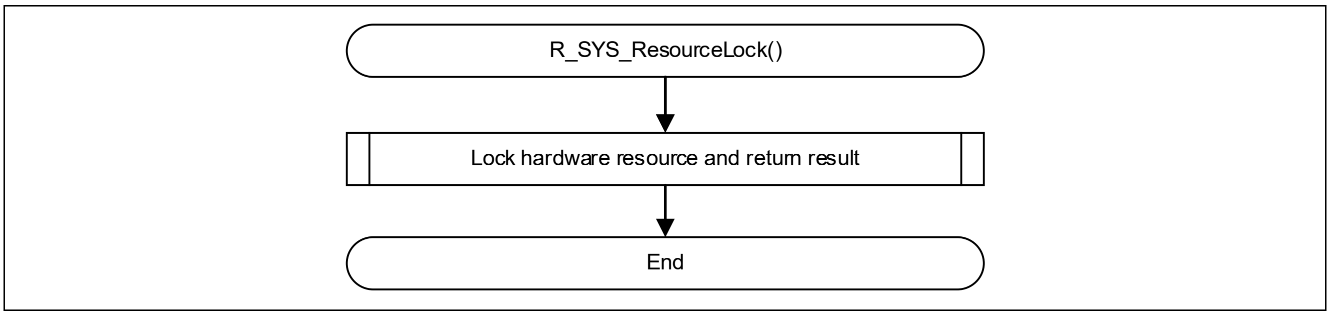


Figure 4.41 R_SYS_ResourceLock Function Processing Flow

4.3.36 R_SYS_ResourceUnlock Function

Table 4-45 R_SYS_ResourceUnlock Function Specifications

Format	void R_SYS_ResourceUnlock(e_system_mcu_lock_t hw_index)
Description	Unlocks a hardware resource.
Argument	e_system_mcu_lock_t hw_index [Input]: Specifies a hardware resource number.
Return value	None
Remarks	–

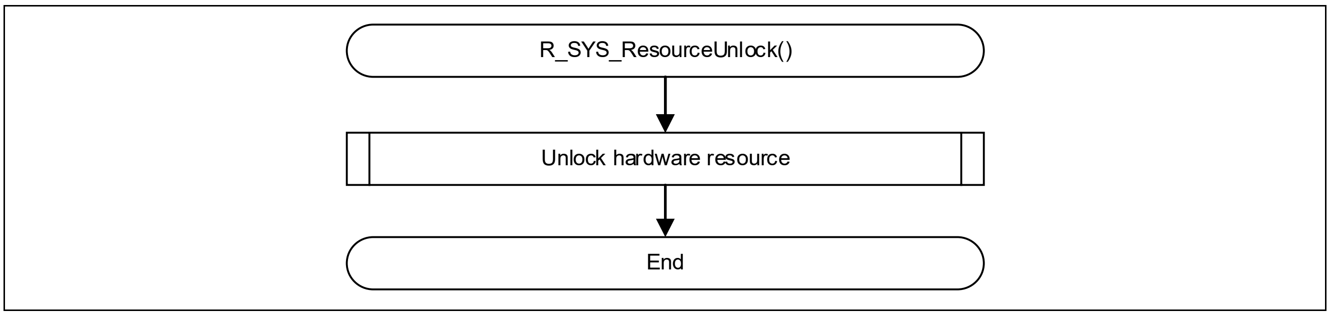


Figure 4.42 R_SYS_ResourceUnlock Function Processing Flow

4.3.37 R_SYS_RegisterProtectEnable Function

Table 4-46 R_SYS_RegisterProtectEnable Function Specifications

Format	void R_SYS_RegisterProtectEnable(e_system_reg_protect_t regs_to_protect)
Description	Enables register protection.
Argument	e_system_reg_protect_t regs_to_protect [Input]: Specifies a register protection number.
Return value	None
Remarks	–

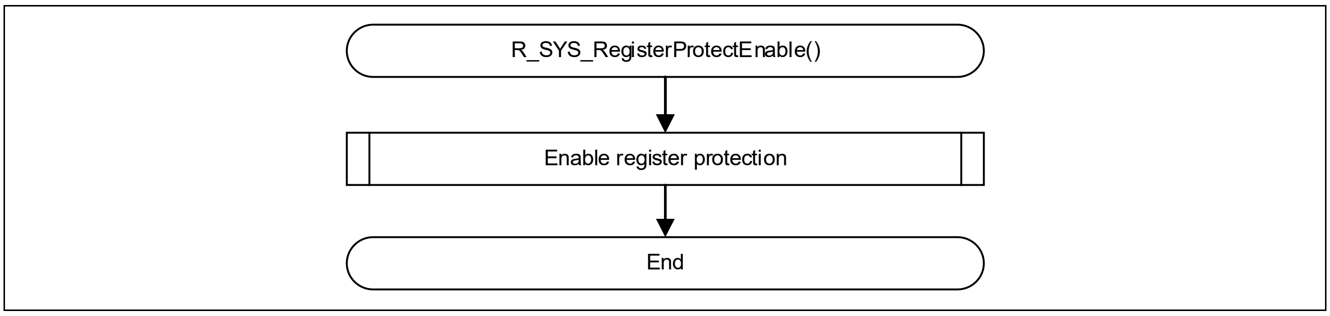


Figure 4.43 R_SYS_RegisterProtectEnable Function Processing Flow

4.3.38 R_SYS_RegisterProtectDisable Function

Table 4-47 R_SYS_RegisterProtectDisable Function Specifications

Format	void R_SYS_RegisterProtectDisable(e_system_reg_protect_t regs_to_unprotect)
Description	Disables register protection.
Argument	e_system_reg_protect_t regs_to_unprotect [Input]: Specifies a register protection number.
Return value	None
Remarks	–

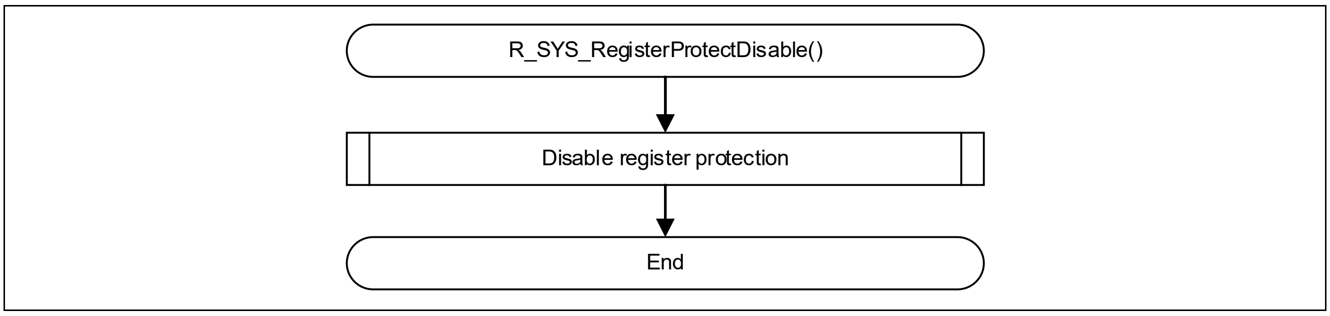


Figure 4.44 R_SYS_RegisterProtectDisable Function Processing Flow

4.3.39 R_SYS_SoftwareDelay Function

Table 4-48 R_SYS_SoftwareDelay Function Specifications

Format	void R_SYS_SoftwareDelay(uint32_t delay, e_system_delay_units_t units)
Description	Generates a software delay of the specified number of milliseconds or microseconds.
Argument	uint32_t delay [Input]: Specifies a delay time.
	e_system_delay_units_t units [Input]: Specifies the unit (milliseconds or microseconds) of the delay time.
Return value	None
Remarks	–

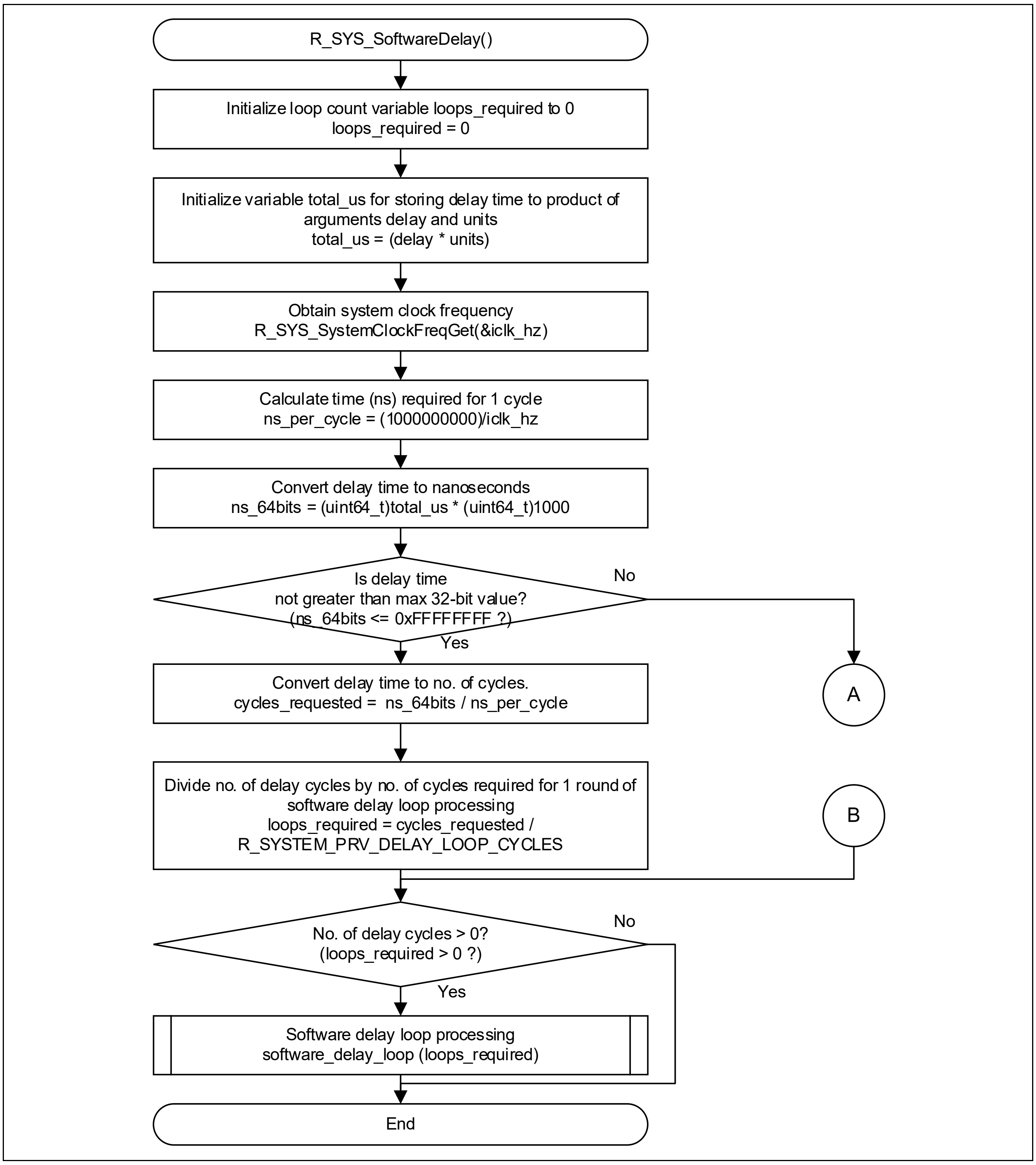

R_SYS_SoftwareDelay()
Initialize loop count variable loops_required to 0
loops_required = 0
Initialize variable total_us for storing delay time to product of arguments delay and units
total_us = (delay * units)
Obtain system clock frequency
R_SYS_SystemClockFreqGet(&iclk_hz)
Calculate time (ns) required for 1 cycle
ns_per_cycle = (1000000000)/iclk_hz
Convert delay time to nanoseconds
ns_64bits = (uint64_t)total_us * (uint64_t)1000
Is delay time not greater than max 32-bit value?
(ns_64bits <= 0xFFFFFFFF ?)
No
Yes
A
Convert delay time to no. of cycles.
cycles_requested = ns_64bits / ns_per_cycle
Divide no. of delay cycles by no. of cycles required for 1 round of software delay loop processing
loops_required = cycles_requested / R_SYSTEM_PRV_DELAY_LOOP_CYCLES
B
No. of delay cycles > 0?
(loops_required > 0 ?)
No
Yes
Software delay loop processing
software_delay_loop (loops_required)
End

Figure 4.45 R_SYS_SoftwareDelay Function Processing Flow (1/2)

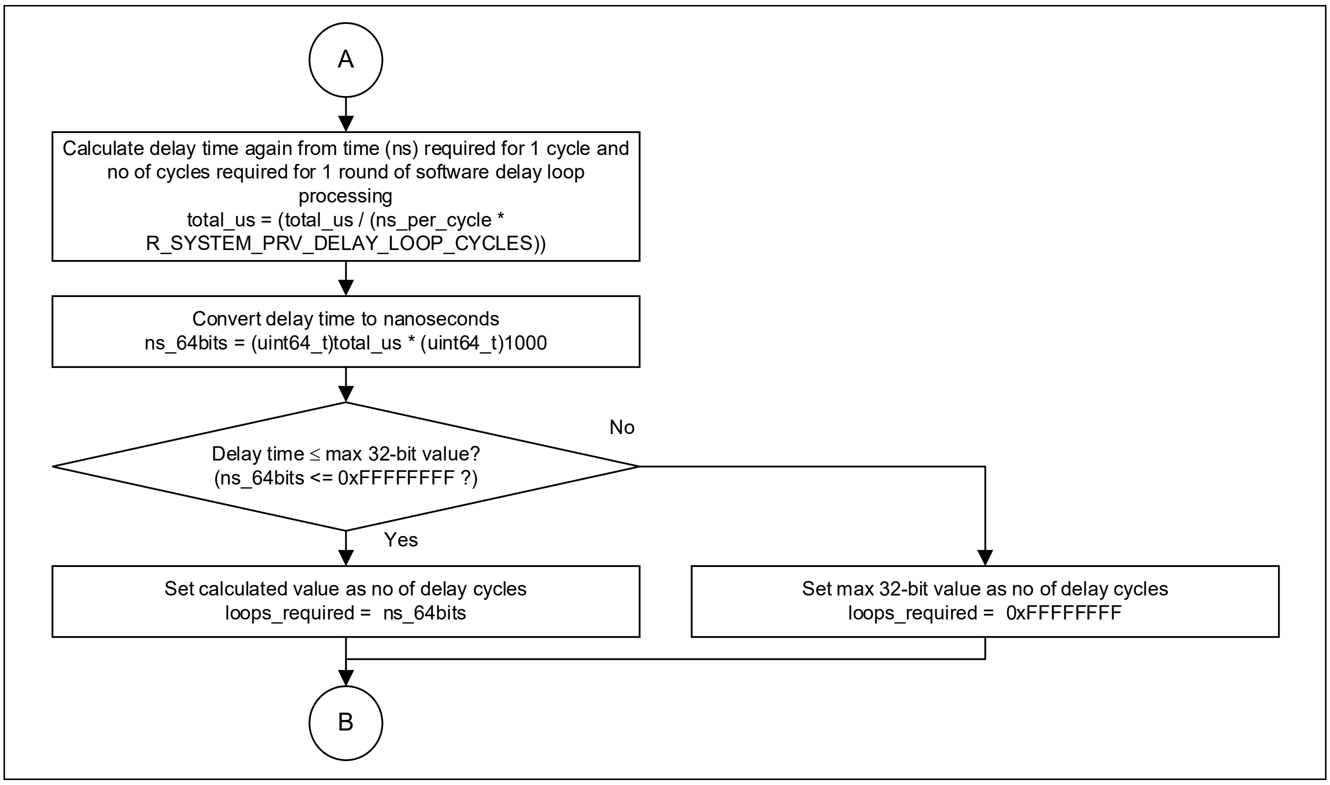

A
Calculate delay time again from time (ns) required for 1 cycle and no of cycles required for 1 round of software delay loop processing
total_us = (total_us / (ns_per_cycle * R_SYSTEM_PRV_DELAY_LOOP_CYCLES))
Convert delay time to nanoseconds
ns_64bits = (uint64_t)total_us * (uint64_t)1000
Delay time ≤ max 32-bit value?
(ns_64bits <= 0xFFFFFFFF ?)
No
Yes
Set calculated value as no of delay cycles
loops_required = ns_64bits
Set max 32-bit value as no of delay cycles
loops_required = 0xFFFFFFFF
B

Figure 4.46 R_SYS_SoftwareDelay Function Processing Flow (2/2)

4.3.40 R_SYS_GetVersion Function

Table 4-49 R_SYS_GetVersion Function Specifications

Format	uint32_t R_SYS_GetVersion(void)
Description	Obtains the version of the R_SYSTEM driver.
Argument	None
Return value	Obtained version of the R_SYSTEM driver
Remarks	–

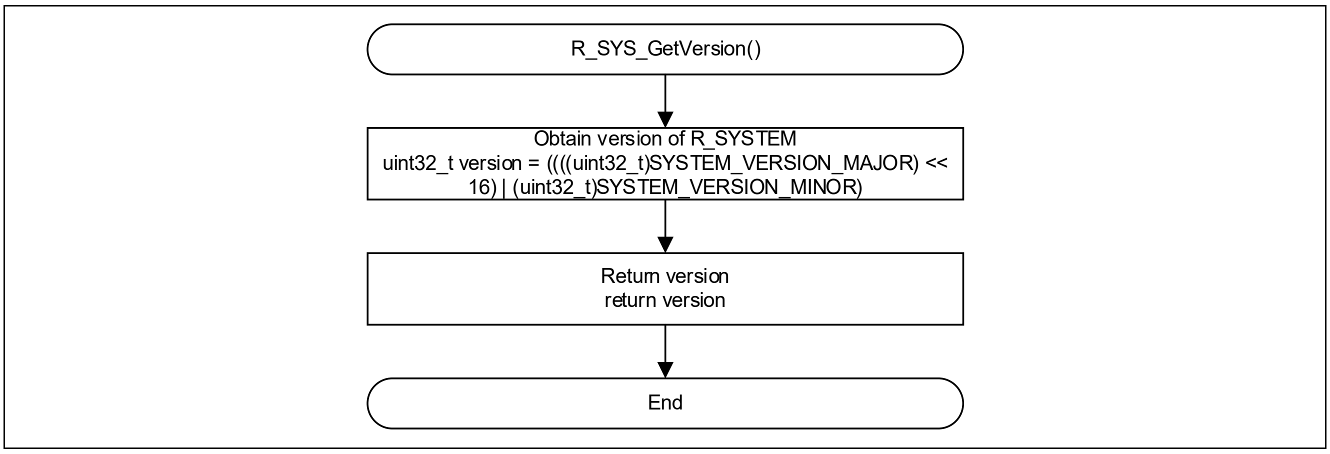

R_SYS_GetVersion()
Obtain version of R_SYSTEM
uint32_t version = ((((uint32_t)SYSTEM_VERSION_MAJOR) << 16) | (uint32_t)SYSTEM_VERSION_MINOR)
Return version
return version
End

Figure 4.47 R_SYS_GetVersion Function Processing Flow

4.3.41 r_sys_BoostFlagGet Function

Table 4-50 r_sys_BoostFlagGet Function Specifications

Format	int32_t r_sys_BoostFlagGet(bool * boost_flg)
Description	Obtains the flag indicating the occurrence of a transition to boost mode.
Argument	bool * boost_flg [Input]: Specifies the location for storing the obtained flag indicating a transition to boost mode.
Return value	Normal end (0)
Remarks	boost_flg == true: Transition to boost mode occurred. boost_flg == false: Transition to boost mode has not occurred.

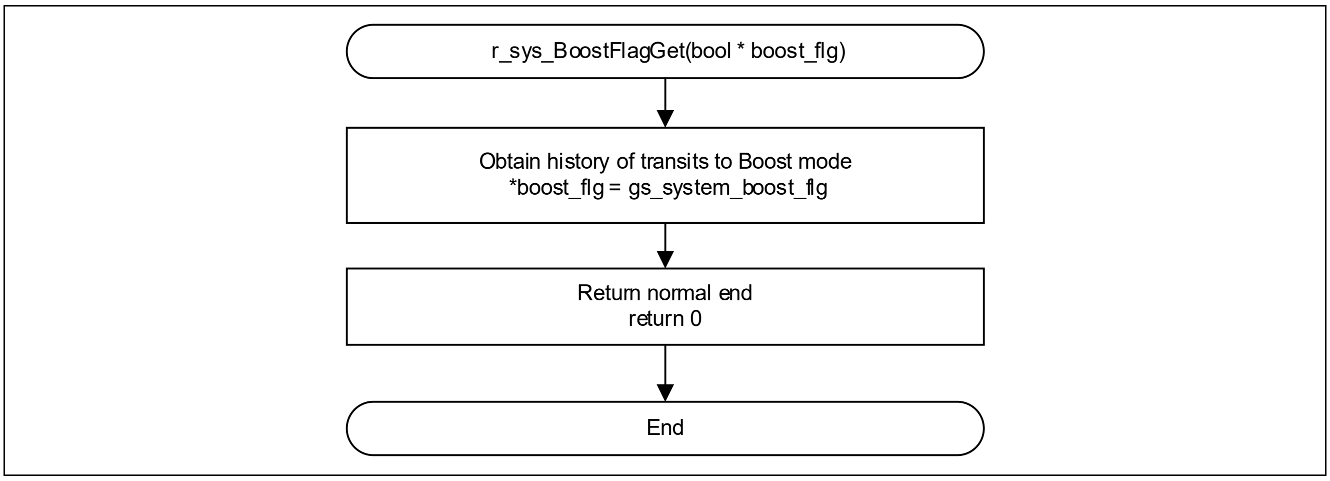

r_sys_BoostFlagGet(bool * boost_flg)
Obtain history of transits to Boost mode
*boost_flg = gs_system_boost_flg
Return normal end
return 0
End

Figure 4.48 r_sys_BoostFlagGet Function Processing Flow

4.3.42 r_sys_BoostFlagSet Function

Table 4-51 r_sys_BoostFlagSet Function Specifications

Format	int32_t r_sys_BoostFlagSet(void)
Description	Sets the flag indicating the occurrence of a transition to boost mode.
Argument	None
Return value	Normal end (0)
Remarks	–

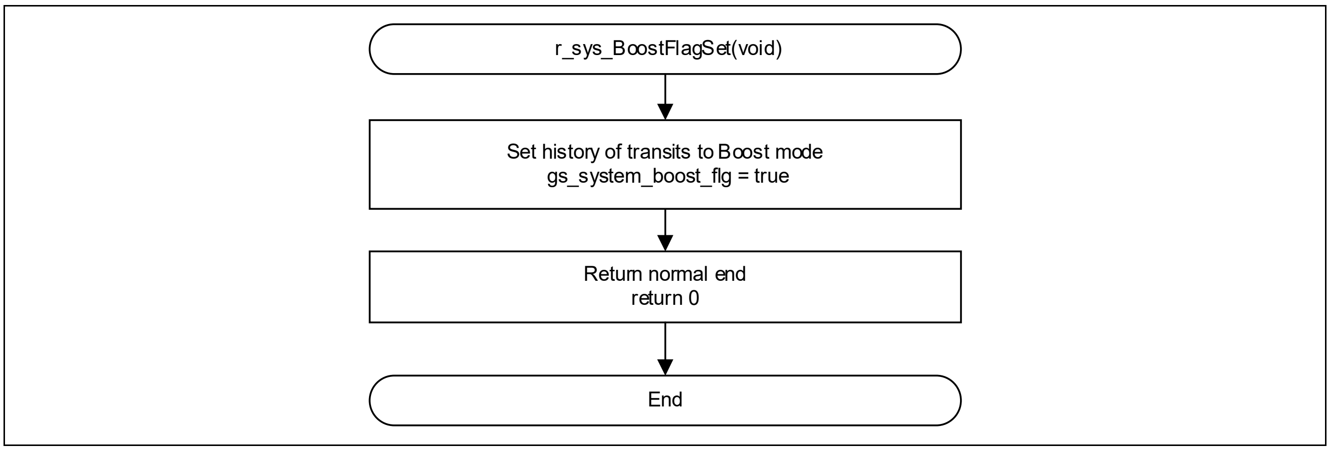

r_sys_BoostFlagSet(void)
Set history of transits to Boost mode
gs_system_boost_flg = true
Return normal end
return 0
End

Figure 4.49 r_sys_BoostFlagSet Function Processing Flow

4.3.43 r_sys_BoostFlagClr Function

Table 4-52 r_sys_BoostFlagClr Function Specifications

Format	int32_t r_sys_BoostFlagClr(void)
Description	Clears the flag indicating the occurrence of a transition to boost mode.
Argument	None
Return value	Normal end (0)
Remarks	–

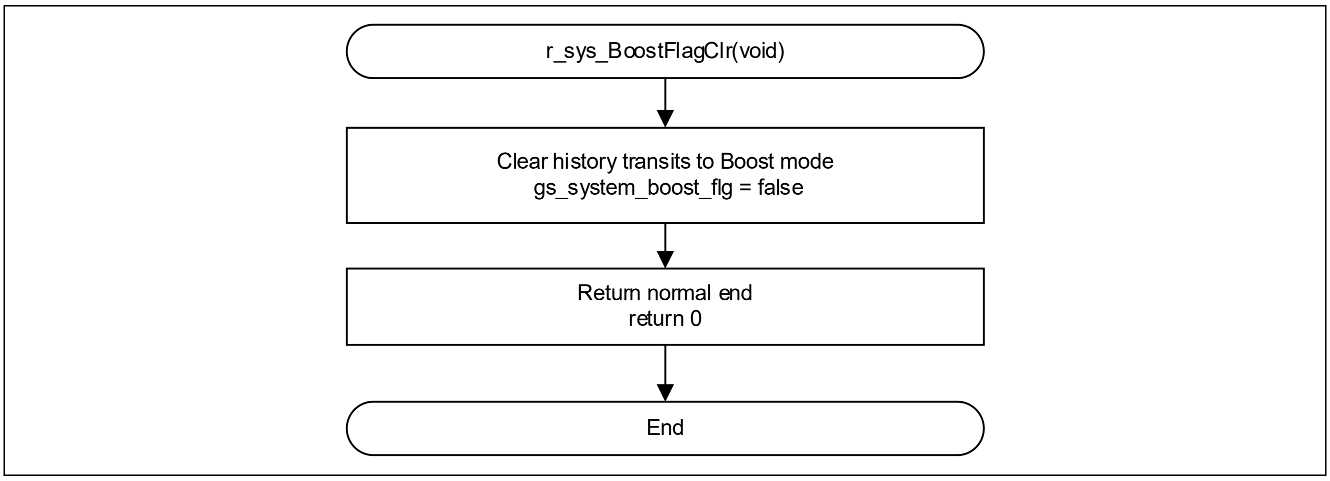

r_sys_BoostFlagClr(void)
Clear history transits to Boost mode
gs_system_boost_flg = false
Return normal end
return 0
End

Figure 4.50 r_sys_BoostFlagClr Function Processing Flow

4.3.44 r_system_wdt_refresh Function

Table 4-53 r_system_wdt_refresh Function Specifications

書式	void r_system_wdt_refresh (void)
仕様説明	Refresh the down-counter of WDT.
引数	None
戻り値	None
備考	This function is implemented as a WEAK function in the R_SYSTEM Driver. Implementing a non-weak function with the same name will disable the corresponding function in R_SYSTEM driver.

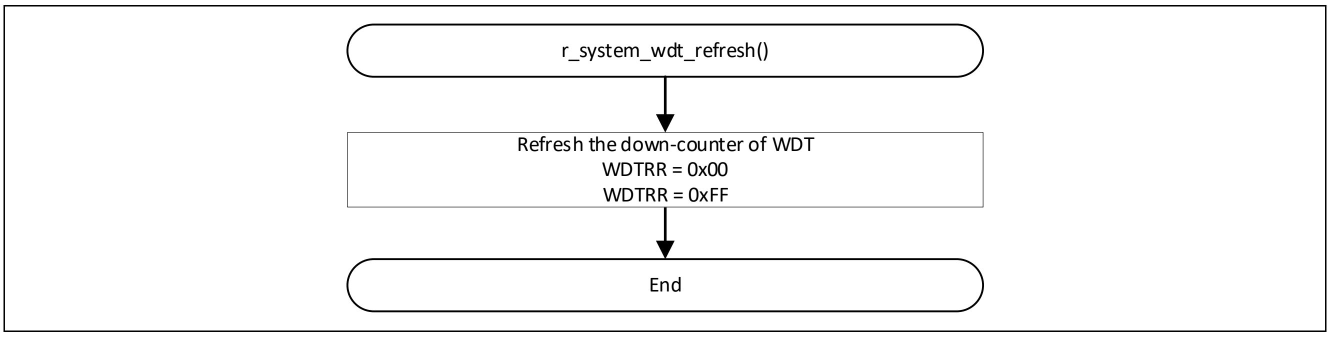


図 4.51 r_system_wdt_refresh Function Processing Flow

4.3.45 IELn_IRQHandler Function (n = 0 to 31)

Table 4-54 IELn_IRQHandler Function Specifications

Format	void IELn_IRQHandler(void)
Description	Executes the IRQ interrupt handler defined by the event link.
Argument	None
Return value	None
Remarks	–

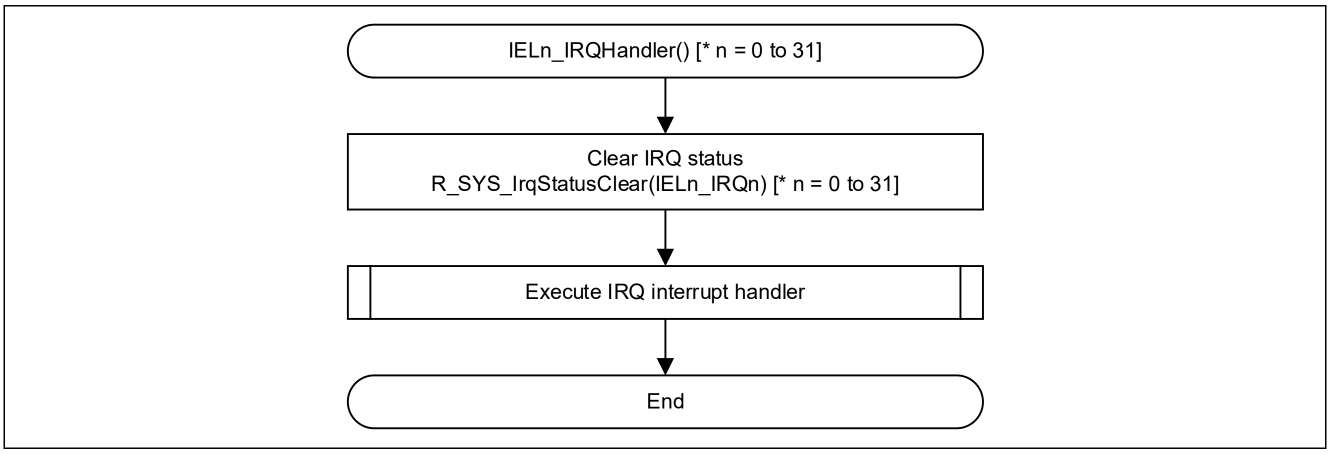

IELn_IRQHandler() [* n = 0 to 31]
Clear IRQ status
R_SYS_IrqStatusClear(IELn_IRQn) [* n = 0 to 31]
Execute IRQ interrupt handler
End

Figure 4.52 IELn_IRQHandler Function Processing Flow

4.3.46 R_NVIC_EnableIRQ Function

Table 4-55 R_NVIC_EnableIRQ Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_EnableIRQ(IRQn_Type IRQn)
Description	Enables the interrupt corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	None
Remarks	The interrupt is enabled by this function executed via RAM. (The code is to be expanded inline.)

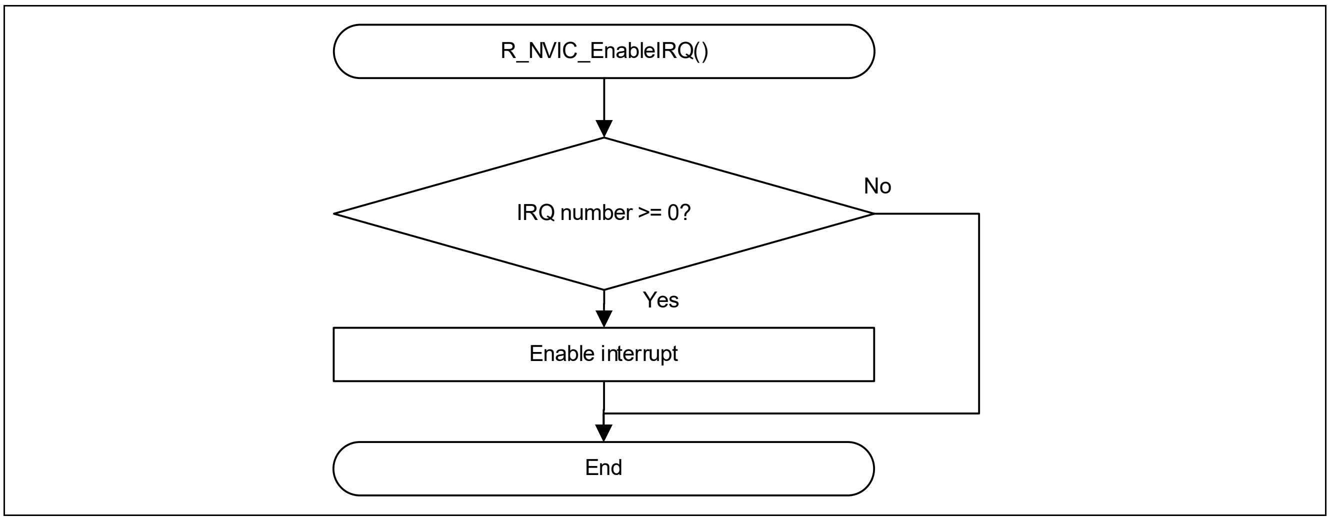


Figure 4.53 R_NVIC_EnableIRQ Function Processing Flow

4.3.47 R_NVIC_GetEnableIRQ Function

Table 4-56 R_NVIC_GetEnableIRQ Function Specifications

Format	__STATIC_FORCEINLINE uint32_t R_NVIC_GetEnableIRQ(IRQn_Type IRQn)
Description	Obtains the interrupt setting corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	Disabled (0)
	Enabled (1)
Remarks	The interrupt setting is obtained by this function executed via RAM. (The code is to be expanded inline.)

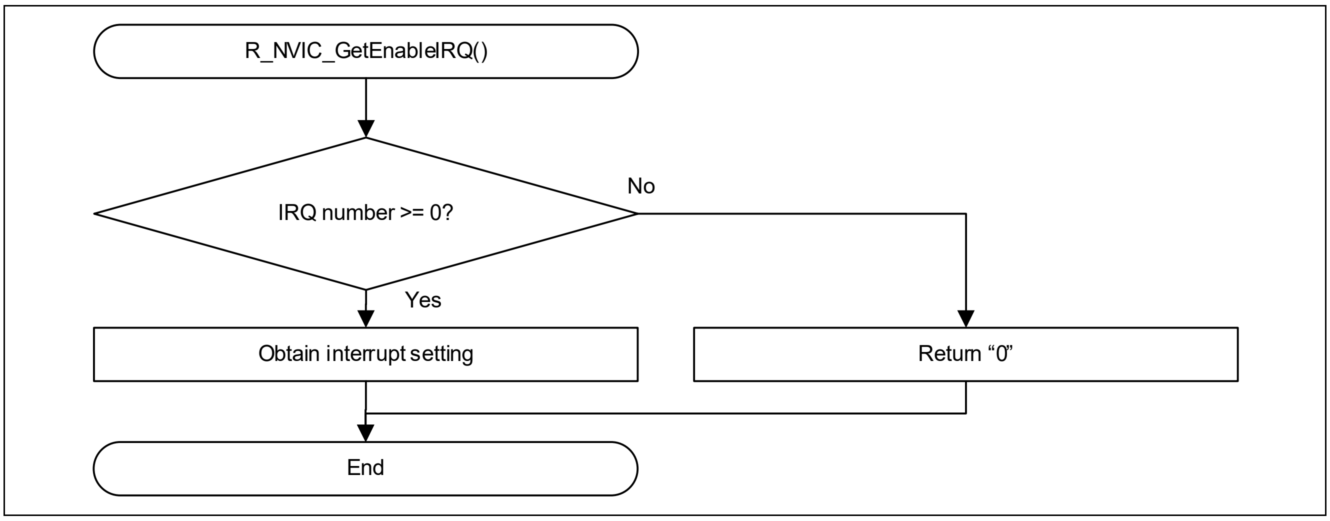


Figure 4.54 R_NVIC_GetEnableIRQ Function Processing Flow

4.3.48 R_NVIC_DisableIRQ Function

Table 4-57 R_NVIC_DisableIRQ Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_DisableIRQ(IRQn_Type IRQn)
Description	Disables the interrupt corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	None
Remarks	The interrupt is disabled by this function executed via RAM. (The code is to be expanded inline.)

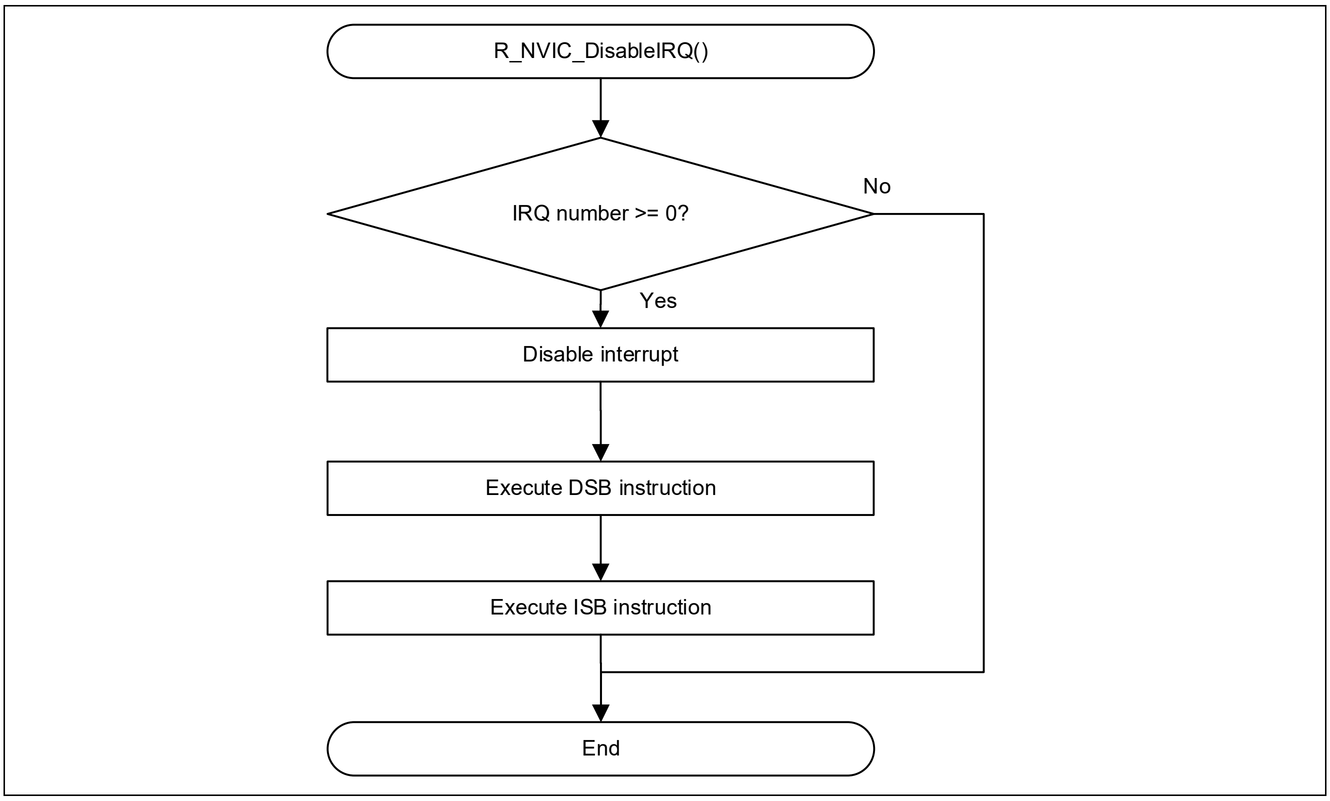


Figure 4.55 R_NVIC_DisableIRQ Function Processing Flow

4.3.49 R_NVIC_GetPendingIRQ Function

Table 4-58 R_NVIC_GetPendingIRQ Function Specifications

Format	__STATIC_FORCEINLINE uint32_t R_NVIC_GetPendingIRQ(IRQn_Type IRQn)
Description	Obtains the pending state of the interrupt corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	No interrupt pending (0)
	Interrupt pending (1)
Remarks	The interrupt pending state is obtained by this function executed via RAM. (The code is to be expanded inline.)

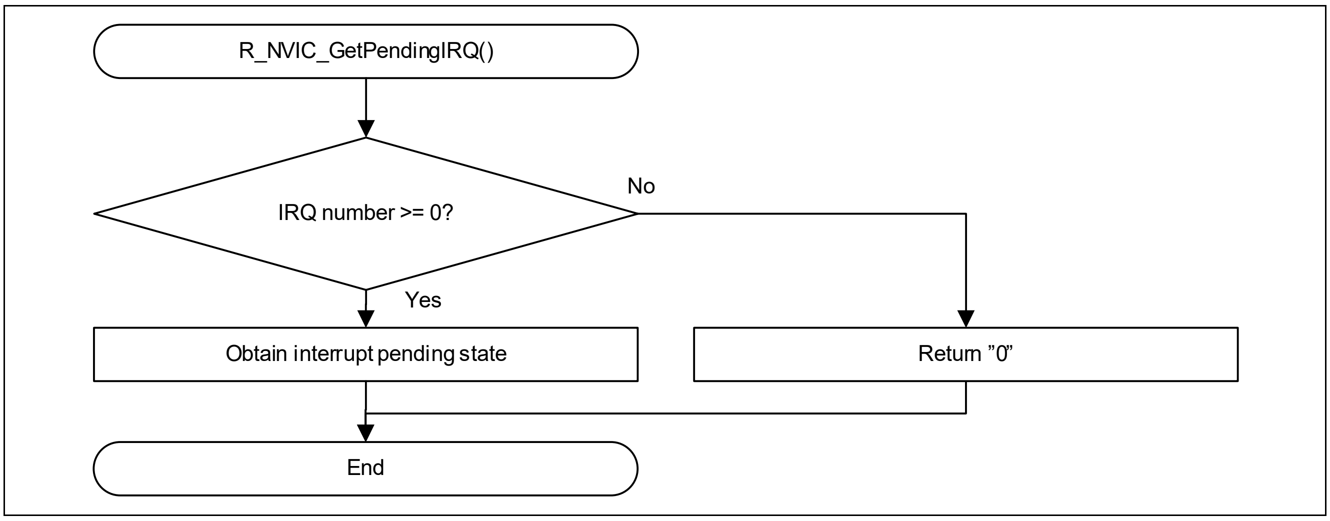


Figure 4.56 R_NVIC_GetPendingIRQ Function Processing Flow

4.3.50 R_NVIC_SetPendingIRQ Function

Table 4-59 R_NVIC_SetPendingIRQ Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_SetPendingIRQ(IRQn_Type IRQn)
Description	Places the interrupt corresponding to an IRQ number of the NVIC defined in Cortex-M0+ to the pending state.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	None
Remarks	The interrupt is placed in the pending state by this function executed via RAM. (The code is to be expanded inline.)

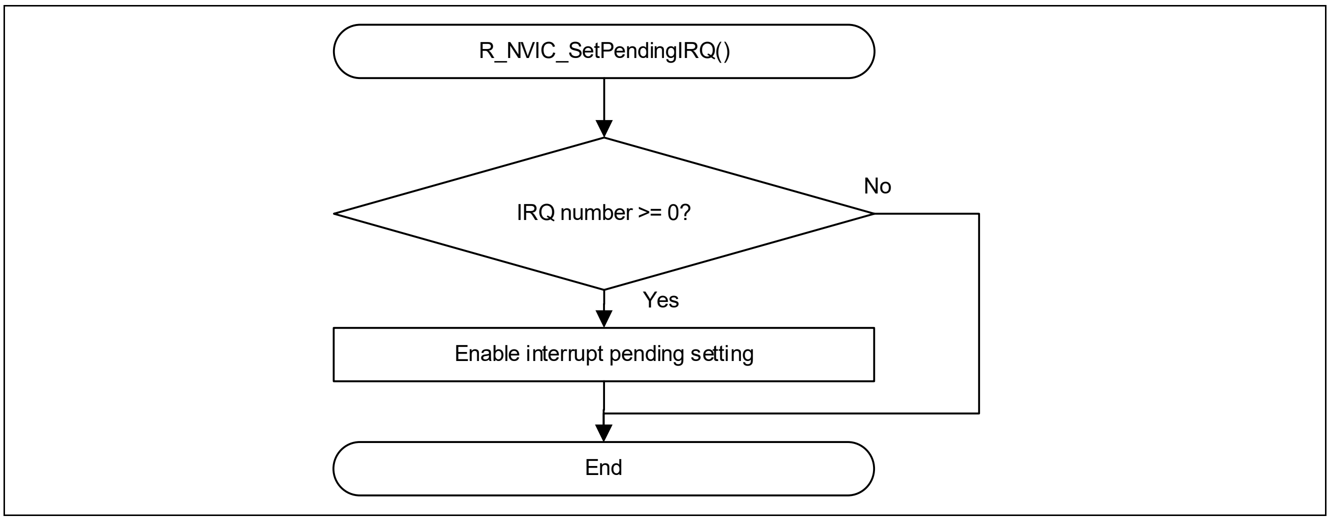


Figure 4.57 R_NVIC_SetPendingIRQ Function Processing Flow

4.3.51 R_NVIC_ClearPendingIRQ Function

Table 4-60 R_NVIC_ClearPendingIRQ Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_ClearPendingIRQ(IRQn_Type IRQn)
Description	Releases the interrupt corresponding to an IRQ number of the NVIC defined in Cortex-M0+ from the pending state.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	None
Remarks	The interrupt is released from the pending state by this function executed via RAM. (The code is to be expanded inline.)

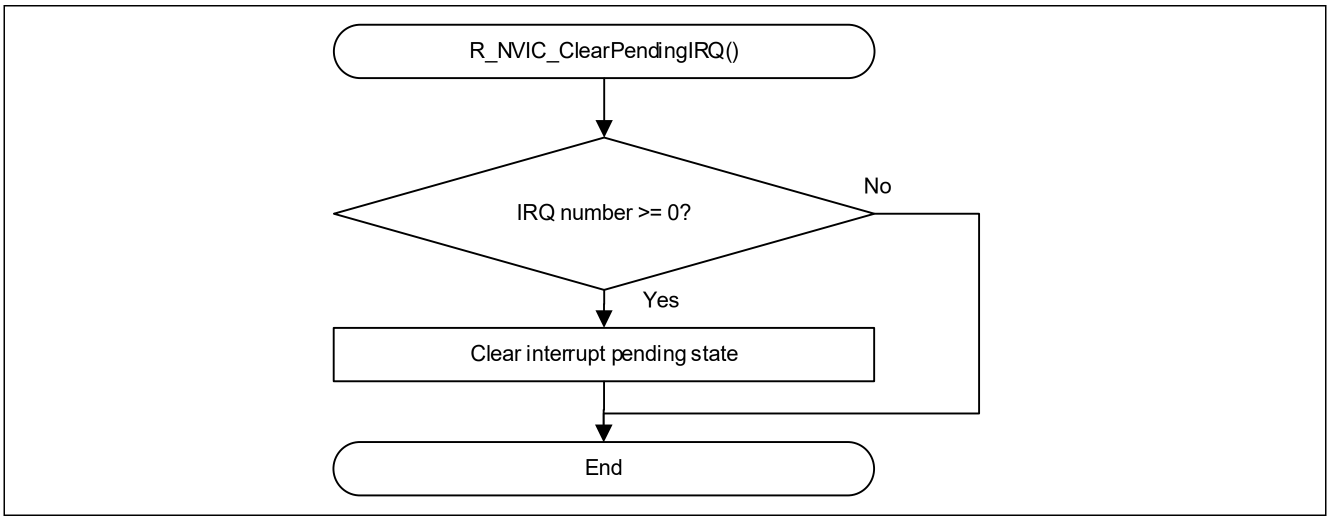


Figure 4.58 R_NVIC_ClearPendingIRQ Function Processing Flow

4.3.52 R_NVIC_SetPriority Function

Table 4-61 R_NVIC_SetPriority Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
Description	Specifies the priority of the interrupt or priority of System Handler corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number. uint32_t priority [Input]: Specifies the priority of the interrupt.
Return value	None
Remarks	The priority of the interrupt is specified by this function executed via RAM. (The code is to be expanded inline.) The smaller the value, the higher the priority of the interrupt.

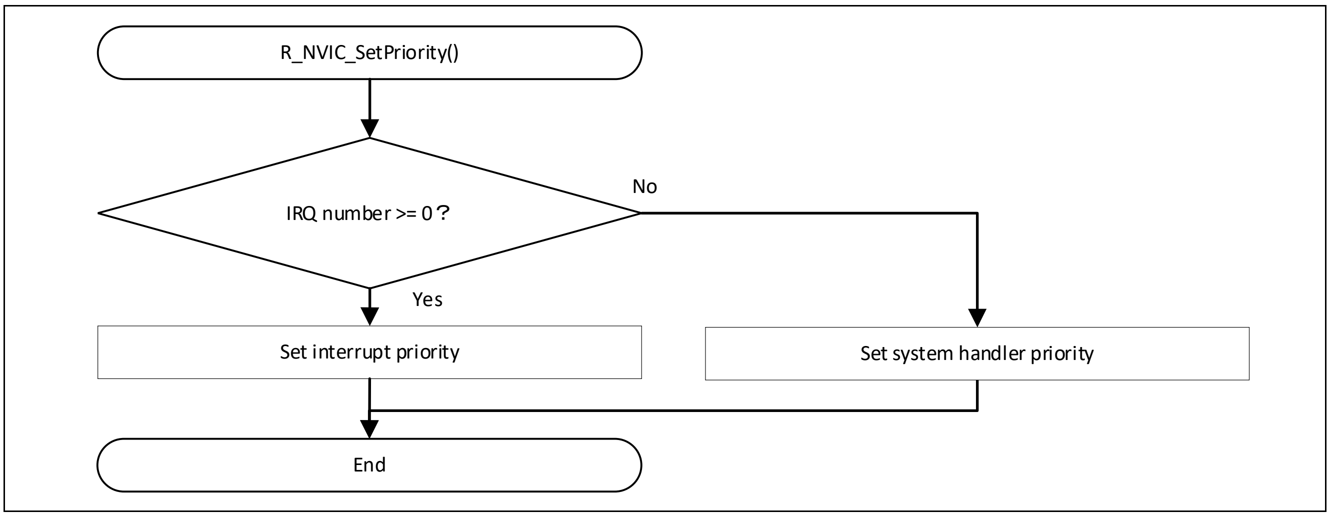


Figure 4.59 R_NVIC_SetPriority Function Processing Flow

4.3.53 R_NVIC_GetPriority Function

Table 4-62 R_NVIC_GetPriority Function Specifications

Format	__STATIC_FORCEINLINE uint32_t R_NVIC_GetPriority(IRQn_Type IRQn)
Description	Obtains the priority of the interrupt or priority of System Handler corresponding to an IRQ number of the NVIC defined in Cortex-M0+.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number.
Return value	Priority of interrupt
Remarks	The priority of the interrupt is obtained by this function executed via RAM. (The code is to be expanded inline) The smaller the value, the higher the priority of the interrupt.

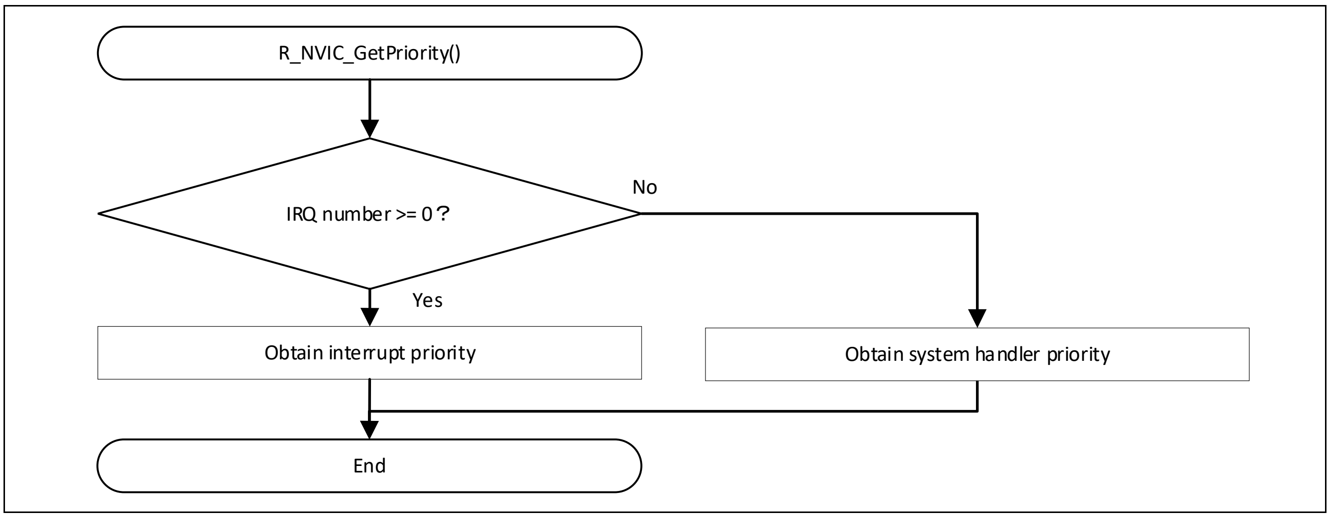


Figure 4.60 R_NVIC_GetPriority Function Processing Flow

4.3.54 R_NVIC_SetVector Function

Table 4-63 R_NVIC_SetVector Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_SetVector(IRQn_Type IRQn, uint32_t vector)
Description	Specifies the offset address of the vector table from the base address.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31). uint32_t vector [Input]: Specifies an offset address.
Return value	None
Remarks	The offset address is specified by this function executed via RAM. (The code is to be expanded inline.)

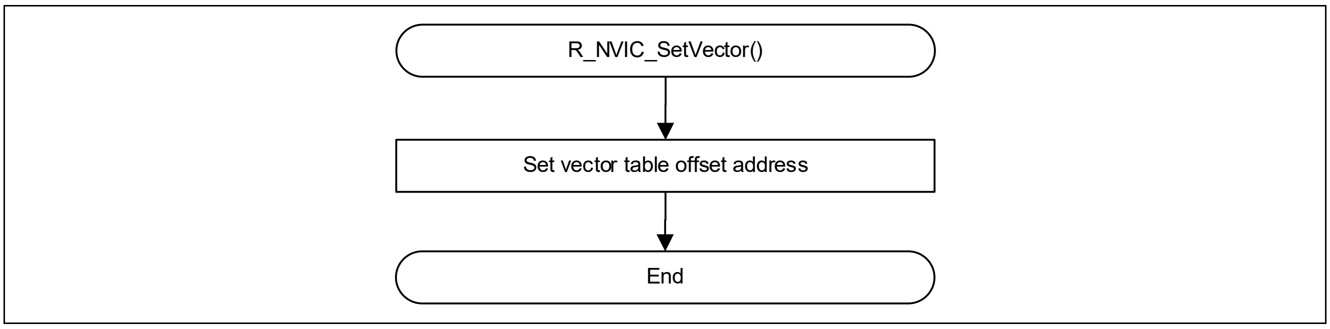


Figure 4.61 R_NVIC_SetVector Function Processing Flow

4.3.55 R_NVIC_GetVector Function

Table 4-64 R_NVIC_GetVector Function Specifications

Format	__STATIC_FORCEINLINE uint32_t R_NVIC_GetVector(IRQn_Type IRQn)
Description	Obtains the offset address of the vector table from the base address.
Argument	IRQn_Type IRQn [Input]: Specifies an IRQ number (0 to 31).
Return value	Offset address
Remarks	The offset address is obtained by this function executed via RAM. (The code is to be expanded inline)

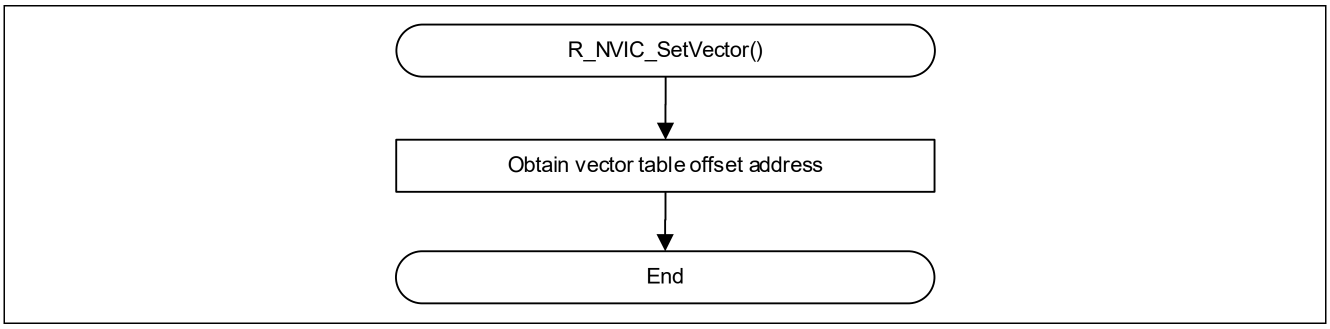


Figure 4.62 R_NVIC_GetVector Function Processing Flow

4.3.56 R_NVIC_SystemReset Function

Table 4-65 R_NVIC_SystemReset Function Specifications

Format	__STATIC_FORCEINLINE void R_NVIC_SystemReset(void)
Description	Requests a system-level reset.
Argument	None
Return value	None
Remarks	A reset is requested by this function executed via RAM.

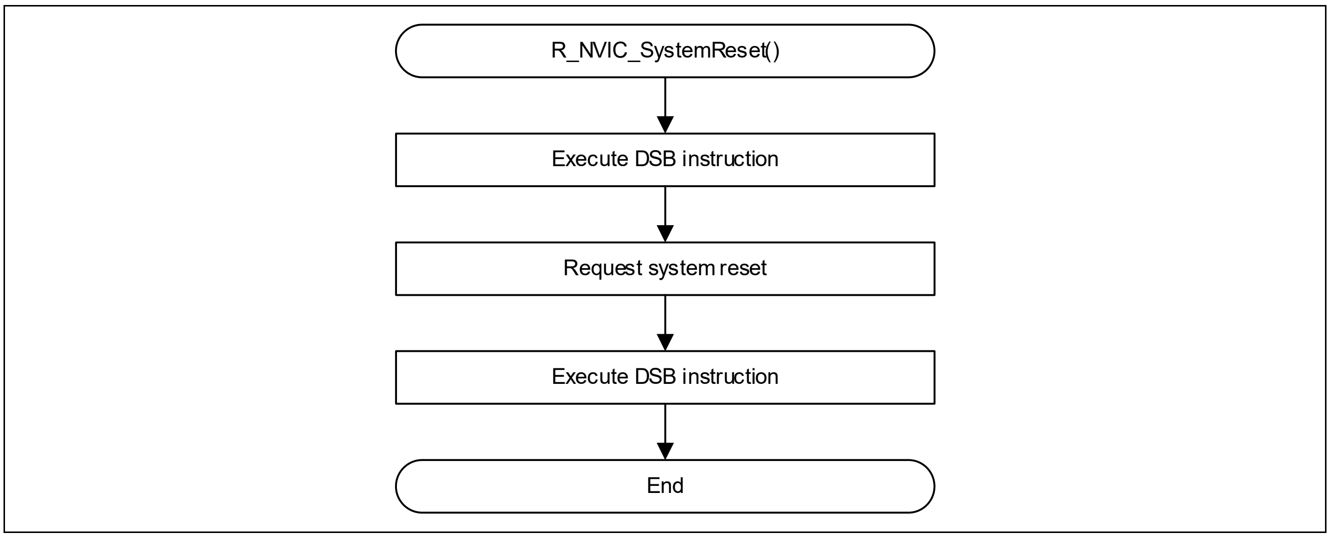


Figure 4.63 R_NVIC_SystemReset Function Processing Flow

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
0.72	July 1, 2019	—	First edition issued
1.00	August. 9, 2019	—	Update to support R_SYSTEM version 1.00
		—	Renamed document title and file to correspond to the official series name. Change series and group names in the document.
		3	Added Startup Guide to Development Using CMSIS Package to list of related document.
		4	Modefied the File Structure of R_SYSTEM Driver in RE 1.5 MB CMSIS Driver Package.
		20	Modified the R_SYS_HighSpeedModeSet Function Processing Flow.
		25	Modified the R_SYS_32kHzSpeedModeSet Function Processing Flow.
		53	Modified the R_SYS_SubOscSpeedClockStart Function Processing Flow.
		54	Modified the R_SYS_SubOscSpeedClockStop Function Processing Flow.
		68	Added the r_system_wdt_refresh Function.
1.10	Mar. 12. 2020	19	Modified the R_SYS_BoostSpeedModeSet Function Processing Flow.
		21 - 22	Modified the R_SYS_ HighSpeedModeSet Function Processing Flow.
		24	Modified the R_SYS_ LowSpeedModeSet Function Processing Flow.
		26	Modified the R_SYS_ 32kHzSpeedModeSet Function Processing Flow.
		45	Modified the R_SYS_MainOscSpeedClockStart Function Processing Flow
		46	Modified the R_SYS_MainOscSpeedClockStop Function Processing Flow.
		48	Modified the R_SYS_HighSpeedClockStart Function Processing Flow
		49	Modified the R_SYS_HighSpeedClockStop Function Processing Flow.
		50	Modified the R_SYS_MediumSpeedClockStart Function Processing Flow
		51	Modified the R_SYS_MediumSpeedClockStop Function Processing Flow.
		52	Modified the R_SYS_LowSpeedClockStart Function Processing Flow
		53	Modified the R_SYS_LowSpeedClockStop Function Processing Flow.
		54	Modified the R_SYS_SubOscSpeedClockStart Function Processing Flow
		55	Modified the R_SYS_SubOscSpeedClockStop Function Processing Flow.
		57	Modified the R_SYS_PLLSpeedClockStart Function Processing Flow
		58	Modified the R_SYS_PLLSpeedClockStop Function Processing Flow.
1.20	Aug. 17, 2020	54	Modified the R_SYS_SubOscSpeedClockStart Function Processing Flow

1.30	May. 12, 2021	78	Modified the R_NVIC_SetPriority Function Specifications and Processing Flow
		79	Modified the R_NVIC_GetPriority Function Specifications and Processing Flow

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit 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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.