RE01 1500KB, 256KB Group

CMSIS Driver R_LPM Specifications

Introduction

This specification document describes R_LPM low consumption driver specifications of RE01 1500KB Group, 256KB Group CMSIS Driver Package.

Target Device

RE01 1500KB Group

RE01 256KB Group

Contents

1. Overview

The CMSIS Driver Package is provided for each of the RE01 1500KB group and the RE01 256KB group. Unless noted otherwise the specification of the R_LPM driver for both the RE01 1500KB group and the RE01 256KB group is same.

A list of abbreviations in this specifications document is shown in Table 1-1, and a list of related documents is shown in Table 1-2.

Table 1-1 List of Abbreviations

Name	Abbreviations
RENESAS-DRIVER R_LPM	R_LPM driver
RENESAS-DRIVER R_SYSTEM	R_SYSTEM driver
RE01 1500KB Group User's Manual: Hardware	1500KB Group UMH
RE01 256KB Group User's Manual: Hardware	256KB Group UMH
Snooze	Snooze
Software standby	SSTBY
Deep software standby	DSTBY
Low-leakage current mode	BackBias mode, VBB
Operation mode (Not in standby)	OPE

Table 1-2 List of Related Document

Document Name	Document Number
RE01 Group (with 1.5-Mbyte Flash Memory)User's Manual: Hardware	R01UH0796
RE01 Group (with 256KB Flash Memory)User's Manual: Hardware	R01UH0894
RE01 1500KB,256KB Group Getting Started Guide to Development Using CMSIS Package	R01AN4660

2. Internal Configuration of Software Component

The R_LPM driver consists of the functions shown in Figure 2-1 for each power consumption reduction function.

The R_LPM driver calls functions of the R_SYSTEM driver according to the user-configurable configuration. The call conditions and function names for the R_SYSTEM driver functions are shown in Figure 2-1.

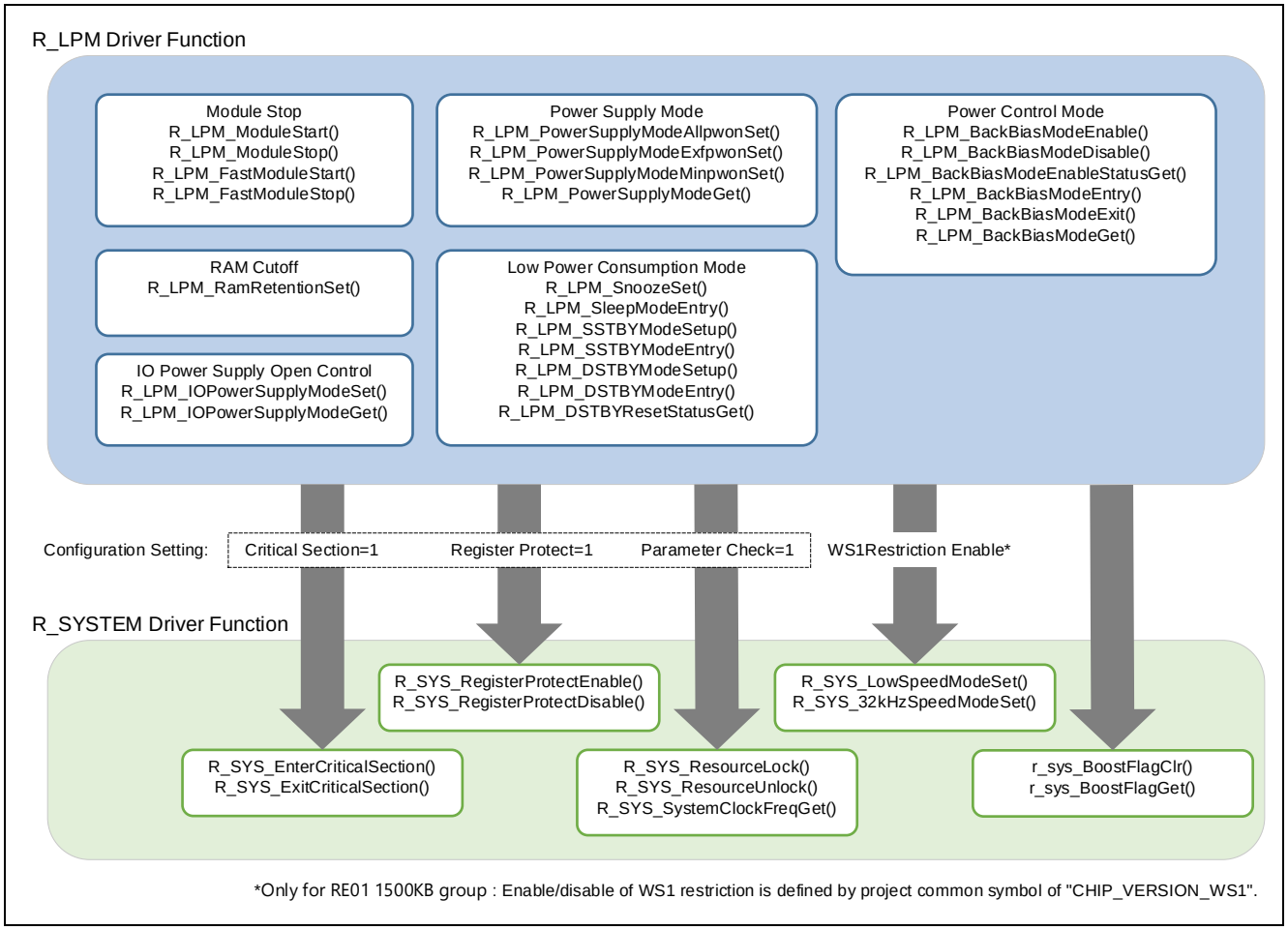

Figure 2-1 Relationship Between R_LPM Driver Function and R_SYSTEM Driver Function

2.1 File Configuration

The R_LPM drive corresponds to DeviceHAL in the CMSIS Driver Package and consists of three files which are r_lpm_api.c, r_lpm_api.h, r_lpm_cfg.h in vendor-specific file storage directory. The function of each file are shown in Table 2-1. The file configuration of the R_LPM driver in the RE01 1500KB Group CMSIS Driver Package is shown by the blue line in Figure 2-2. The components of the CMSIS Driver package for the RE01 256KB group is different from RE01 1500KB group because of hardware difference, however file composition of the R_LPM driver indicated by blue frame on Figure 2-2 is same with the RE01 1500KB group.

Table 2-1 R_LPM Driver Function of Each File

File Name	Description
r_lpm_api.c	Driver source file Provides actual condition of driver function Build this file for using R_LPM driver
r_lpm_api.h	Driver header file Provides user-referenceable macro / type/ prototype declaration Include this file for using R_LPM driver
r_lpm_cfg.h	Configuration definition file Provides user-configurable configuration definition

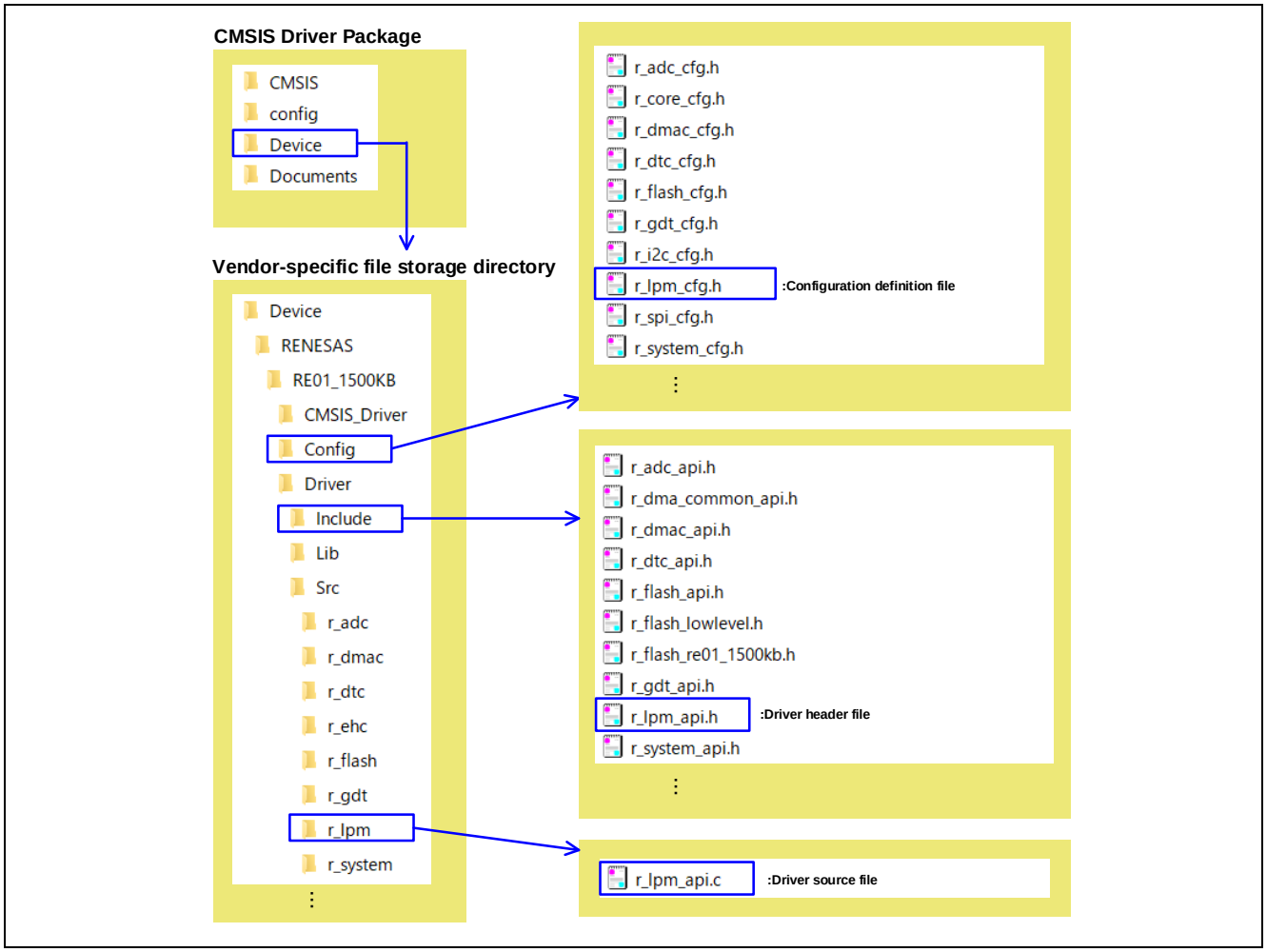

Figure 2-2 R_LPM Driver File Configuration in CMSIS Driver Package

3. Internal Operation of Software Components

R_LPM driver actualize function for reducing power consumption. This chapter describes the procedure for calling the R_LPM driver function for reducing power consumption by mode transition. Processing other than R_LPM driver is not described in the procedure flow, however conditions before execution of R_LPM driver function, execution frequency, end conditions of the function are described in comments.

3.1 Power Control Mode

The procedure for transitioning to BackBias mode is shown in Figure 3-1.

When you use R_LPM_BackBiasModeEntry() function to transition to BackBias mode, start setup of BackBias control using R_LPM_BackBiasModeEnable() function and confirm BackBias control setup completion by using R_LPM_BackBiasModeEnableStatusGet() function. This is only procedure which needs to be followed regarding R_LPM driver.

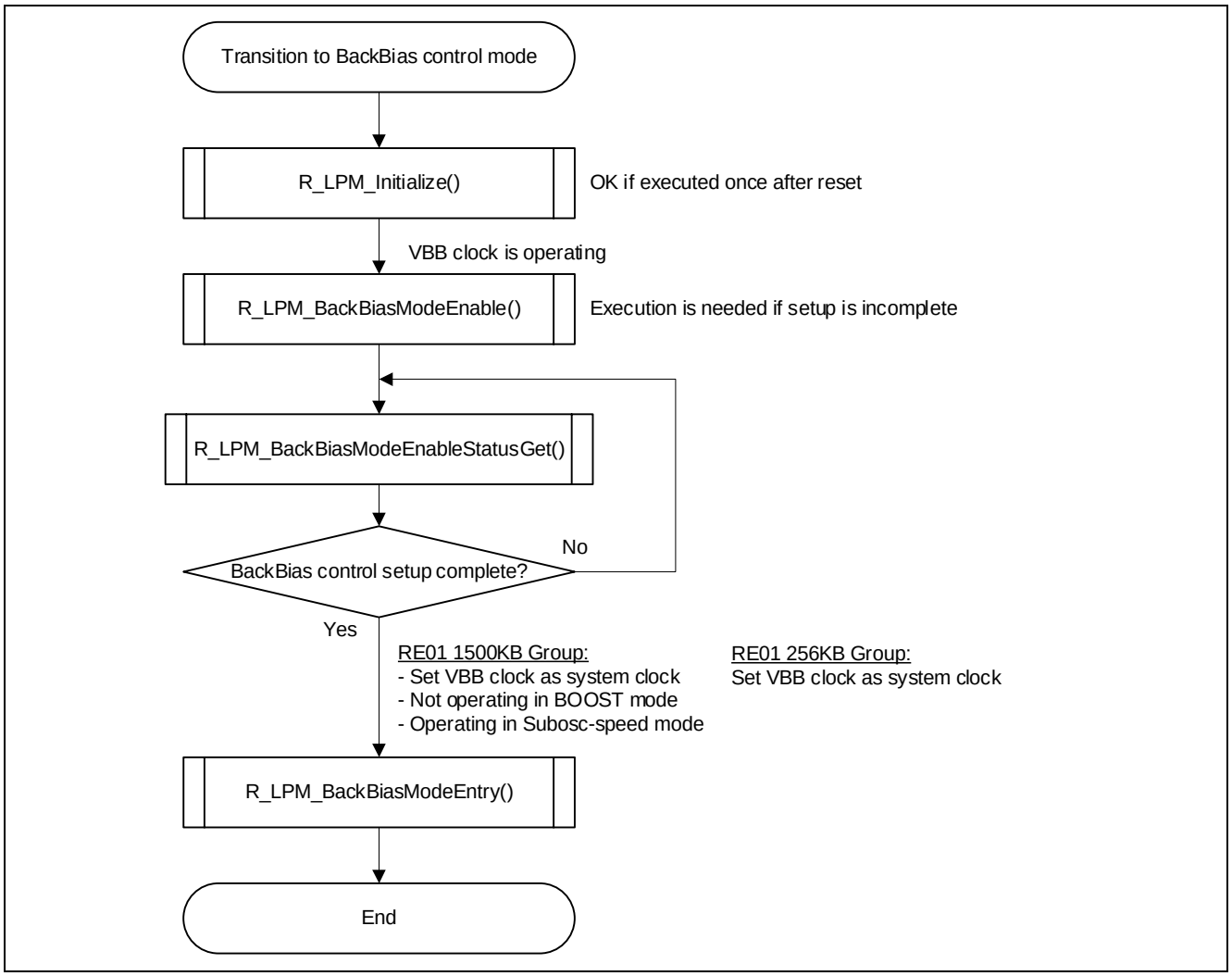

Figure 3-1 Transition Procedure to BackBias Mode

3.2 Power Supply Mode

The transition procedure to each power supply mode is shown in Figure 3-2.

The RE01 1500KB hardware does not support the direct transition between ALLPWON mode and MINPWON mode, however, R_LPM driver supports these mode transitions by performing two-step transitions in the function. User can specify the transition destination mode without being aware of the operating mode.

The RE01 256KB hardware supports the direct transition between ALLPWON mode and MINPWON mode. User can specify the transition destination mode as with the RE01 1500KB R_LPM driver.

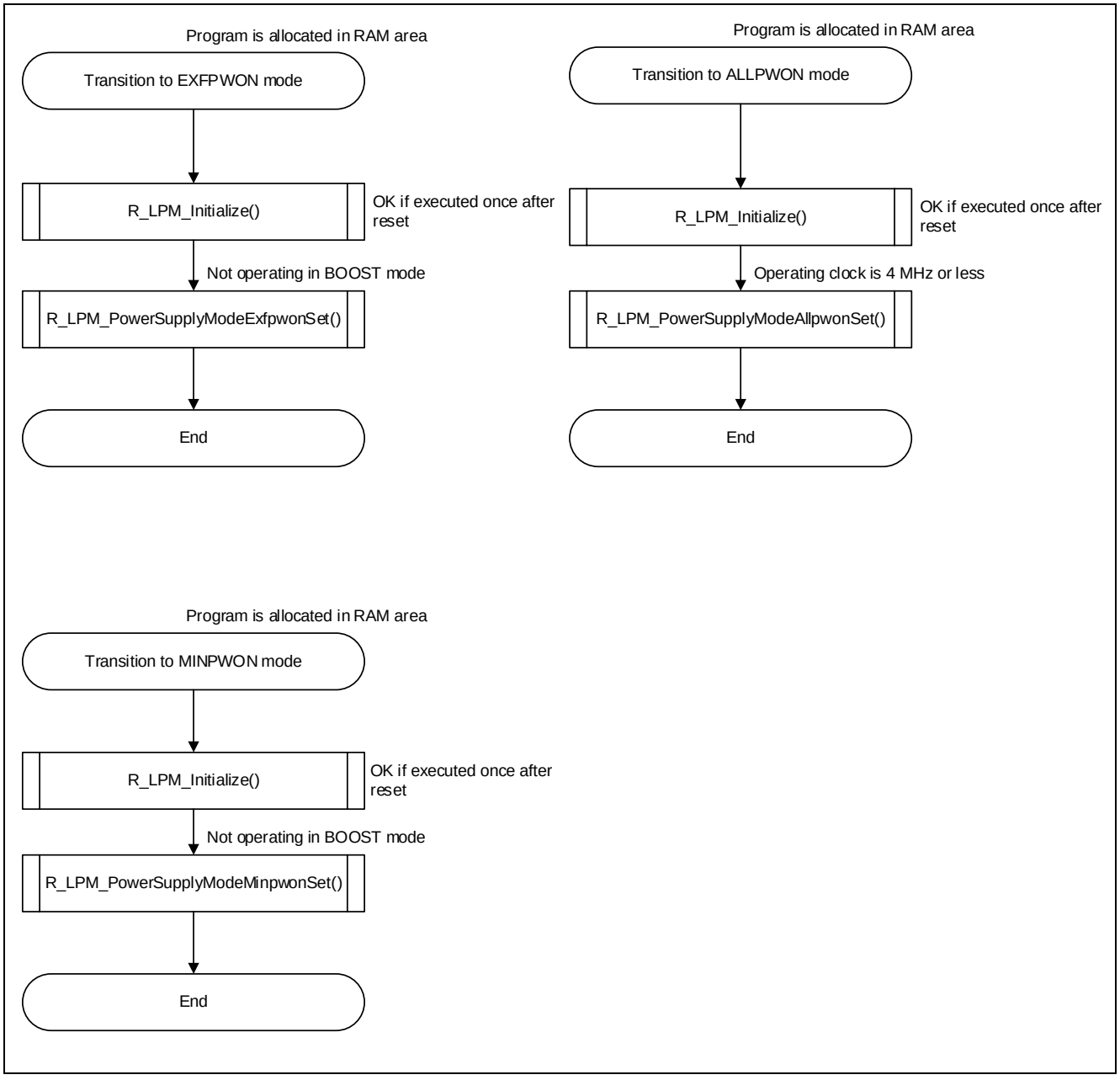

Figure 3-2 Transition Procedure to Each Power Supply Mode

3.3 Low Power Consumption Mode

The transition procedure to each low power consumption mode is shown in Figure 3-3.

Before transitioning to SSTBY mode and DSTBY mode, use R_LPM_SSTBYModeSetup() and R_LPM_DSTBYModeSetup () functions to set the operating conditions such as the return factor from low power consumption mode. If the operating conditions are not set, mode transition is performed under conditions equivalent to the initial values of the hardware.

Each Entry function transition to low power consumption mode within the function and that leads to CPU to stop temporarily. The CPU restarts and each entry function exits when it detects a return factor from low power consumption mode. However, the R_LPM_DSTBYModeEntry () function jumps to the reset vector without function exits due to release of reset when it detects a return factor.

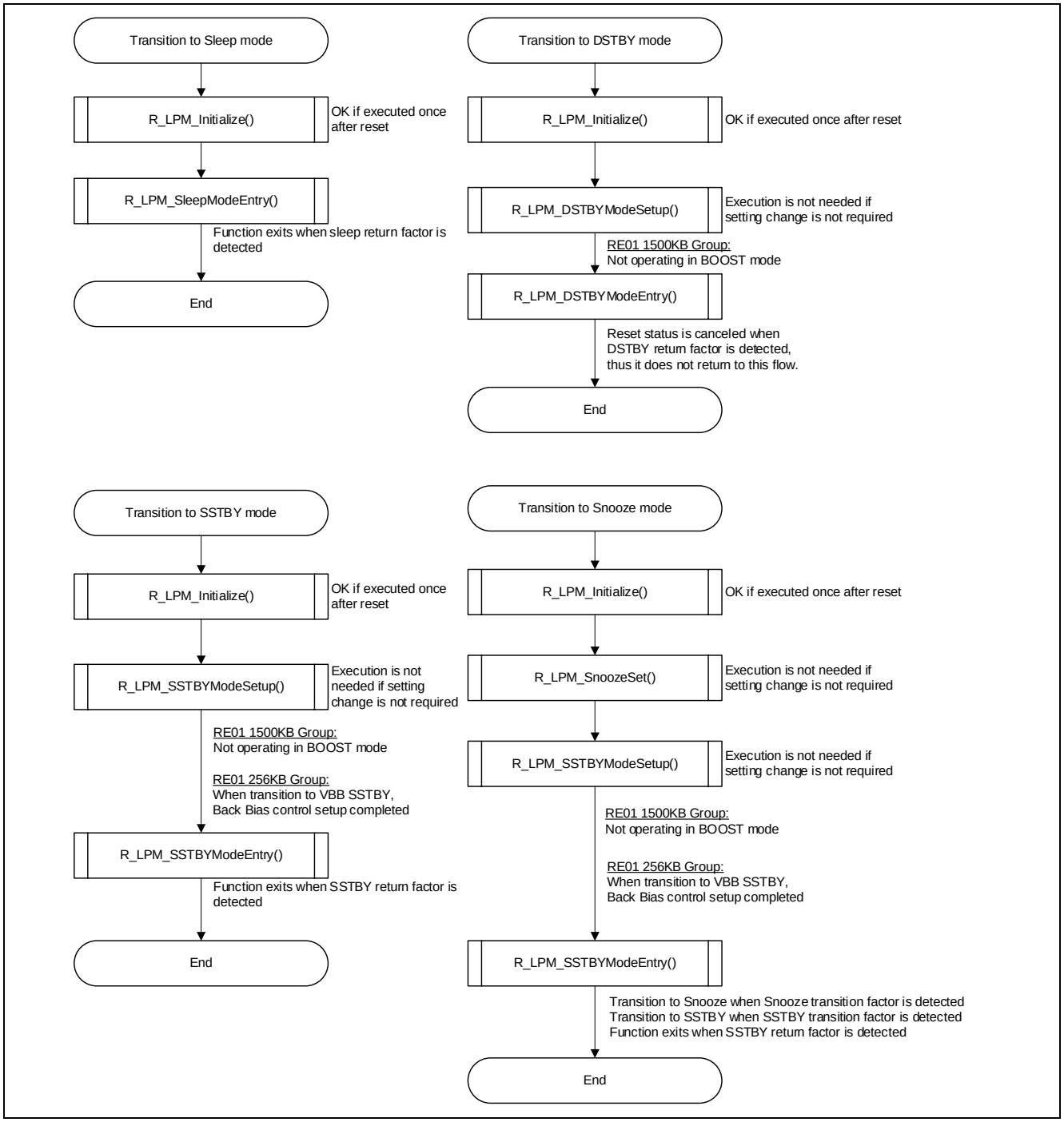

Figure 3-3 Transition Procedure to Each Low Power Consumption Mode

4. Software Unit Detail Information

4.1 Configuration

R_LPM driver prepares user-configurable configurations in the r_lpm_cfg.h file.

4.1.1 Parameter Check

Set enable / disable of parameter check in R_LPM driver.

Name：LPM_CFG_PARAM_CHECKING_ENABLE

Table 4-1 LPM_CFG_PARAM_CHECKING_ENBLE Settings

Setting value	Description
0	Disable parameter check. Does not detect the error condition described in the function specification. However, errors corresponding to “Unable to set xxx in hardware” will always be detected regardless of the settings in this configuration.
1 (initial value)	Enable parameter check. Detect the error condition described in the function specification.

4.1.2 Critical Section

Set enable / disable critical section control in R_LPM driver.

In order to change some bits of the register, control the critical section so that there is no interruption when reading out the register value and changing it partially and resetting it.

Name：LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE

Table 4-2 LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE Settings

Setting value	Description
0	Disable control of critical section.
1 (initial value)	Enable control of critical section using the following R_SYSTEM driver functions. R_SYS_EnterCriticalSection() function R_SYS_ExitCriticalSection() function

4.1.3 Register Write Protection

Set enable / disable register write protection control in R_LPM driver.

Reducing power consumption function has register write protection register. When writing to corresponding register, it is necessary to control register write protection.

Name：LPM_CFG_REGISTER_PROTECTION_ENABLE

Table 4-3 LPM_CFG_REGISTER_PROTECTION_ENABLE Settings

Setting value	Description
0	Disable control of register write protection.
1 (initial value)	Enable control of register write protection using the following R_SYSTEM driver functions. R_SYS_RegisterProtectEnable() function R_SYS_RegisterProtectDisable() function

4.1.4 RAM Allocation of Function

Settings to execute a specific function of the R_LPM driver in RAM.

Programs that run while the flash is shut off must be allocated in RAM and executed in RAM.

The configuration to set the RAM allocation of functions have different definitions for each function.

Name：LPM_CFG_SECTION_xxx

The function name is written in all capital letters for “xxx”.

Example: R_LPM_Initialize function → LPM_CFG_SECTION_R_LPM_INITIALIZE

Table 4-4 LPM_CFG_SECTION_xxx Settings

Definition name	Description
SYSTEM_SECTION_CODE	Does not allocate function in RAM.
SYSTEM_SECTION_RAM_FUNC	Allocate function in RAM.

Table 4-5 RAM Allocation Initial State of Each Function

No.	Function	RAM Allocation
1	R_LPM_GetVersion	
2	R_LPM_Initialize	
3	R_LPM_ModuleStart	✔
4	R_LPM_ModuleStop	✔
5	R_LPM_FastModuleStart	✔
6	R_LPM_FastModuleStop	✔
7	R_LPM_BackBiasModeEnable	
8	R_LPM_BackBiasModeDisable	
9	R_LPM_BackBiasModeEnableStatusGet	
10	R_LPM_BackBiasModeEntry	✔
11	R_LPM_BackBiasModeExit	✔
12	R_LPM_BackBiasModeGet	✔
13	R_LPM_PowerSupplyModeAllpwonSet	✔
14	R_LPM_PowerSupplyModeExfpwonSet	✔
15	R_LPM_PowerSupplyModeMinpwonSet	✔
16	R_LPM_PowerSupplyModeGet	✔
17	R_LPM_SnoozeSet	✔
18	R_LPM_SleepModeEntry	✔
19	R_LPM_SSTBYModeSetup	✔
20	R_LPM_SSTBYModeEntry	✔
21	R_LPM_DSTBYModeSetup	✔
22	R_LPM_DSTBYModeEntry	✔
23	R_LPM_DSTBYResetStatusGet	
24	R_LPM_RamRetentionSet	
25	R_LPM_IOPowerSupplyModeSet	✔
26	R_LPM_IOPowerSupplyModeGet	✔

4.2 Macro / Type Definition

The R_LPM driver prepares user-referenceable macros and type definitions in r_lpm_api.h file.

The defined value is a value that can be set directly to the hardware register in consideration of acceleration.

Use #define definition for applications where multiple definitions need to be selected, such as return factor setting from low power consumption mode. When you set value of defined #define in a register, by setting the logical AND of value set by user and LPM_xxx_ALL that defines the settable bit of the setting corresponding register to the register, it prevents writing in reserved bits when user sets an undefined value.

4.2.1 Acquisition of Version

The definition used in version acquisition is shown in Figure 4-1.

Definition of R_LPM_GetVersion function return value(In the case of Version 1.23)

```
#define LPM_VERSION_MAJOR               (1)
#define LPM_VERSION_MINOR               (23)
```

Figure 4-1 Definition Used in Version Acquisition

4.2.2 Module Stop

(1) RE01 1500KB Group

The definition used in module stop is shown in Figure 4-2.

Type of R_LPM_ModuleStart function input parameter "module"
Type of R_LPM_ModuleStop function input parameter "module"

```
typedef enum e_lpm_mstp
{
    LPM_MSTP_DTC_DMAC,
    LPM_MSTP_IRDA,
    LPM_MSTP_QSPI,
    LPM_MSTP_RIIC1,
    LPM_MSTP_RIIC0,
    LPM_MSTP_USB,
    LPM_MSTP_SPI1,
    LPM_MSTP_SPI0,
    LPM_MSTP_SCI9,
    LPM_MSTP_SCI5,
    LPM_MSTP_SCI4,
    LPM_MSTP_SCI3,
    LPM_MSTP_SCI2,
    LPM_MSTP_SCI1,
    LPM_MSTP_SCI0,
    LPM_MSTP_CAC,
    LPM_MSTP_CRC,
    LPM_MSTP_LED,
    LPM_MSTP_DOC,
    LPM_MSTP_ELC,
    LPM_MSTP_DIV,
    LPM_MSTP_DIL,
    LPM_MSTP_MLCD,
    LPM_MSTP_GDT,
    LPM_MSTP_TSIP_LITE,
    LPM_MSTP_LST,
    LPM_MSTP_TMR,
    LPM_MSTP_AGT1,
    LPM_MSTP_AGT0,
    LPM_MSTP_LPG,
    LPM_MSTP_GPT320_321,
    LPM_MSTP_GPT162_165,
    LPM_MSTP_CCC,
    LPM_MSTP_MTDV,
    LPM_MSTP_POEG,
    LPM_MSTP_S14AD,
    LPM_MSTP_VREF,
    LPM_MSTP_R12DA,
    LPM_MSTP_TEMPS,
    LPM_MSTP_ACMP,
    LPM_MSTP_NUM
} e_lpm_mstp_t;
```

Definition of R_LPM_FastModuleStart function input parameter "module_a"
Definition of R_LPM_FastModuleStop function input parameter "module_a"

```
#define LPM_FAST_MSTP_A_DTC_DMAC        (0x00400000UL)
#define LPM_FAST_MSTP_A_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_A_ALL             (0x00400000UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_b"
Definition of R_LPM_FastModuleStop function input parameter "module_b"

```
#define LPM_FAST_MSTP_B_IRDA            (0x00000020UL)
#define LPM_FAST_MSTP_B_QSPI            (0x00000040UL)
#define LPM_FAST_MSTP_B_RIIC1           (0x00000100UL)
#define LPM_FAST_MSTP_B_RIIC0           (0x00000200UL)
#define LPM_FAST_MSTP_B_USB             (0x00000800UL)
#define LPM_FAST_MSTP_B_SPI1            (0x00040000UL)
#define LPM_FAST_MSTP_B_SPI0            (0x00080000UL)
#define LPM_FAST_MSTP_B_SCI9            (0x00400000UL)
#define LPM_FAST_MSTP_B_SCI5            (0x04000000UL)
#define LPM_FAST_MSTP_B_SCI4            (0x08000000UL)
#define LPM_FAST_MSTP_B_SCI3            (0x10000000UL)
#define LPM_FAST_MSTP_B_SCI2            (0x20000000UL)
#define LPM_FAST_MSTP_B_SCI1            (0x40000000UL)
#define LPM_FAST_MSTP_B_SCI0            (0x80000000UL)
#define LPM_FAST_MSTP_B_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_B_ALL             (0xFC4C0B60UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_c"
Definition of R_LPM_FastModuleStop function input parameter "module_c"

```
#define LPM_FAST_MSTP_C_CAC             (0x00000001UL)
#define LPM_FAST_MSTP_C_CRC             (0x00000002UL)
#define LPM_FAST_MSTP_C_LED             (0x00000400UL)
#define LPM_FAST_MSTP_C_DOC             (0x00002000UL)
#define LPM_FAST_MSTP_C_ELC             (0x00004000UL)
#define LPM_FAST_MSTP_C_DIV             (0x00008000UL)
#define LPM_FAST_MSTP_C_DIL             (0x00400000UL)
#define LPM_FAST_MSTP_C_MLCD            (0x02000000UL)
#define LPM_FAST_MSTP_C_GDT             (0x04000000UL)
#define LPM_FAST_MSTP_C_TSIPLITE        (0x80000000UL)
#define LPM_FAST_MSTP_C_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_C_ALL             (0x8640E403UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_d"
Definition of R_LPM_FastModuleStop function input parameter "module_d"

```
#define LPM_FAST_MSTP_D_LST             (0x00000001UL)
#define LPM_FAST_MSTP_D_TMR             (0x00000002UL)
#define LPM_FAST_MSTP_D_AGT1            (0x00000004UL)
#define LPM_FAST_MSTP_D_AGT0            (0x00000008UL)
#define LPM_FAST_MSTP_D_LPG             (0x00000010UL)
#define LPM_FAST_MSTP_D_GPT320_321      (0x00000020UL)
#define LPM_FAST_MSTP_D_GPT162_165      (0x00000040UL)
#define LPM_FAST_MSTP_D_CCC             (0x00000080UL)
#define LPM_FAST_MSTP_D_MTDV            (0x00000100UL)
#define LPM_FAST_MSTP_D_POEG            (0x00004000UL)
#define LPM_FAST_MSTP_D_S14AD           (0x00010000UL)
#define LPM_FAST_MSTP_D_VREF            (0x00020000UL)
#define LPM_FAST_MSTP_D_R12DA           (0x00100000UL)
#define LPM_FAST_MSTP_D_TEMPS           (0x00400000UL)
#define LPM_FAST_MSTP_D_ACMP            (0x10000000UL)
#define LPM_FAST_MSTP_D_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_D_ALL             (0x105341FFUL)
```

Figure 4-2 Definition Used in Module Stop

(2) RE01 256KB Group

The definition used in module stop is shown in Figure 4-3.

Type of R_LPM_ModuleStart function input parameter "module"
Type of R_LPM_ModuleStop function input parameter "module"

```
typedef enum e_lpm_mstp
{
    LPM_MSTP_DTC_DMAC,
    LPM_MSTP_IRDA,
    LPM_MSTP_QSPI,
    LPM_MSTP_RIIC1,
    LPM_MSTP_RIIC0,
    LPM_MSTP_SPI1,
    LPM_MSTP_SPI0,
    LPM_MSTP_SCI9,
    LPM_MSTP_SCI5,
    LPM_MSTP_SCI4,
    LPM_MSTP_SCI3,
    LPM_MSTP_SCI2,
    LPM_MSTP_SCI1,
    LPM_MSTP_SCI0,
    LPM_MSTP_CAC,
    LPM_MSTP_CRC,
    LPM_MSTP_DOC,
    LPM_MSTP_ELC,
    LPM_MSTP_DIV,
    LPM_MSTP_DIL,
    LPM_MSTP_MLCD,
    LPM_MSTP_GDT,
    LPM_MSTP_TSIP_LITE,
    LPM_MSTP_LST,
    LPM_MSTP_TMR,
    LPM_MSTP_AGT1,
    LPM_MSTP_AGT0,
    LPM_MSTP_GPT320_321,
    LPM_MSTP_GPT162_165,
    LPM_MSTP_CCC,
    LPM_MSTP_RTC,
    LPM_MSTP_IWDT,
    LPM_MSTP_WDT,
    LPM_MSTP_AGTW0,
    LPM_MSTP_AGTW1,
    LPM_MSTP_POEG,
    LPM_MSTP_S14AD,
    LPM_MSTP_VREF,
    LPM_MSTP_WUPT,
    LPM_MSTP_TEMPS,
    LPM_MSTP_NUM
} e_lpm_mstp_t;
```

Definition of R_LPM_FastModuleStart function input parameter "module_a"
Definition of R_LPM_FastModuleStop function input parameter "module_a"

```
#define LPM_FAST_MSTP_A_DTC_DMAC        (0x00400000UL)
#define LPM_FAST_MSTP_A_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_A_ALL             (0x00400000UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_b"
Definition of R_LPM_FastModuleStop function input parameter "module_b"

```
#define LPM_FAST_MSTP_B_IRDA            (0x00000020UL)
#define LPM_FAST_MSTP_B_QSPI            (0x00000040UL)
#define LPM_FAST_MSTP_B_RIIC1           (0x00000100UL)
#define LPM_FAST_MSTP_B_RIIC0           (0x00000200UL)
#define LPM_FAST_MSTP_B_SPI1            (0x00040000UL)
#define LPM_FAST_MSTP_B_SPI0            (0x00080000UL)
#define LPM_FAST_MSTP_B_SCI9            (0x00400000UL)
#define LPM_FAST_MSTP_B_SCI5            (0x04000000UL)
#define LPM_FAST_MSTP_B_SCI4            (0x08000000UL)
#define LPM_FAST_MSTP_B_SCI3            (0x10000000UL)
#define LPM_FAST_MSTP_B_SCI2            (0x20000000UL)
#define LPM_FAST_MSTP_B_SCI1            (0x40000000UL)
#define LPM_FAST_MSTP_B_SCI0            (0x80000000UL)
#define LPM_FAST_MSTP_B_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_B_ALL             (0xFC4C0360UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_c"
Definition of R_LPM_FastModuleStop function input parameter "module_c"

```
#define LPM_FAST_MSTP_C_CAC             (0x00000001UL)
#define LPM_FAST_MSTP_C_CRC             (0x00000002UL)
#define LPM_FAST_MSTP_C_DOC             (0x00002000UL)
#define LPM_FAST_MSTP_C_ELC             (0x00004000UL)
#define LPM_FAST_MSTP_C_DIV             (0x00008000UL)
#define LPM_FAST_MSTP_C_DIL             (0x00400000UL)
#define LPM_FAST_MSTP_C_MLCD            (0x02000000UL)
#define LPM_FAST_MSTP_C_GDT             (0x04000000UL)
#define LPM_FAST_MSTP_C_TSIPLITE        (0x80000000UL)
#define LPM_FAST_MSTP_C_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_C_ALL             (0x8640E003UL)
```

Definition of R_LPM_FastModuleStart function input parameter "module_d"
Definition of R_LPM_FastModuleStop function input parameter "module_d"

```
#define LPM_FAST_MSTP_D_LST             (0x00000001UL)
#define LPM_FAST_MSTP_D_TMR             (0x00000002UL)
#define LPM_FAST_MSTP_D_AGT1            (0x00000004UL)
#define LPM_FAST_MSTP_D_AGT0            (0x00000008UL)
#define LPM_FAST_MSTP_D_GPT320_321      (0x00000020UL)
#define LPM_FAST_MSTP_D_GPT162_165      (0x00000040UL)
#define LPM_FAST_MSTP_D_CCC             (0x00000080UL)
#define LPM_FAST_MSTP_D_RTC             (0x00000200UL)
#define LPM_FAST_MSTP_D_IWDT            (0x00000400UL)
#define LPM_FAST_MSTP_D_WDT             (0x00000800UL)
#define LPM_FAST_MSTP_D_AGTW0           (0x00001000UL)
#define LPM_FAST_MSTP_D_AGTW1           (0x00002000UL)
#define LPM_FAST_MSTP_D_POEG            (0x00004000UL)
#define LPM_FAST_MSTP_D_S14AD           (0x00010000UL)
#define LPM_FAST_MSTP_D_VREF            (0x00020000UL)
#define LPM_FAST_MSTP_D_WUPT            (0x00040000UL)
#define LPM_FAST_MSTP_D_TEMPS           (0x00400000UL)
#define LPM_FAST_MSTP_D_NONE            (0x00000000UL)
#define LPM_FAST_MSTP_D_ALL             (0x00477EEFUL)
```

Figure 4-3 Definition Used in Module Stop

4.2.3 RAM Cutoff

(1) RE01 1500KB Group

The definition used in RAM cutoff is shown in Figure 4-4.

Definition of R_LPM_RamRetentionSet function input parameter "area"

```
#define LPM_RAM_RETENTION_AREA0          (0x01UL)
#define LPM_RAM_RETENTION_AREA1          (0x02UL)
#define LPM_RAM_RETENTION_AREA2          (0x04UL)
#define LPM_RAM_RETENTION_AREA3          (0x08UL)
#define LPM_RAM_RETENTION_AREA4          (0x10UL)
#define LPM_RAM_RETENTION_AREA5          (0x20UL)
#define LPM_RAM_RETENTION_AREA6          (0x40UL)
#define LPM_RAM_RETENTION_AREA7          (0x80UL)
#define LPM_RAM_RETENTION_NONE           (0x00UL)
#define LPM_RAM_RETENTION_ALL            (0xFFUL)
```

Figure 4-4 Definition Used in RAM Cutoff

(2) RE01 256KB Group

The definition used in RAM cutoff is shown in Figure 4-5.

Definition of R_LPM_RamRetentionSet function input parameter "area"

```
#define LPM_RAM_RETENTION_AREA0          (0x01UL)
#define LPM_RAM_RETENTION_AREA1          (0x02UL)
#define LPM_RAM_RETENTION_AREA2          (0x04UL)
#define LPM_RAM_RETENTION_AREA3          (0x08UL)
#define LPM_RAM_RETENTION_NONE           (0x00UL)
#define LPM_RAM_RETENTION_ALL            (0x0FUL)
```

Figure 4-5 Definition Used in RAM Cutoff

4.2.4 IO Power Supply Open Control

(1) RE01 1500KB Group

The definition used in IO power supply open control is shown in Figure 4-6.

Definition of R_LPM_IOPowerSupplyModeSet function input parameter "vocr"
Definition of R_LPM_IOPowerSupplyModeSet function return value

```
#define LPM_IOPOWER_SUPPLY_AVCC0        (0x01)
#define LPM_IOPOWER_SUPPLY_AVCC1        (0x02)
#define LPM_IOPOWER_SUPPLY_IOVCC0       (0x04)
#define LPM_IOPOWER_SUPPLY_IOVCC1       (0x08)
#define LPM_IOPOWER_SUPPLY_IOVCC2       (0x10)
#define LPM_IOPOWER_SUPPLY_IOVCC3       (0x20)
#define LPM_IOPOWER_SUPPLY_USBVCC       (0x40)
#define LPM_IOPOWER_SUPPLY_MTDV         (0x80)
#define LPM_IOPOWER_SUPPLY_NONE         (0x00)
#define LPM_IOPOWER_SUPPLY_ALL          (0xFF)
```

Figure 4-6 Definition Used in IO Power Supply Open Control

(2) RE01 256KB Group

The definition used in IO power supply open control is shown in Figure 4-7.

Definition of R_LPM_IOPowerSupplyModeSet function input parameter "vocr"
Definition of R_LPM_IOPowerSupplyModeSet function return value

```
#define LPM_IOPOWER_SUPPLY_AVCC0        (0x01)
#define LPM_IOPOWER_SUPPLY_IOVCC0       (0x04)
#define LPM_IOPOWER_SUPPLY_IOVCC1       (0x08)
#define LPM_IOPOWER_SUPPLY_NONE         (0x00)
#define LPM_IOPOWER_SUPPLY_ALL          (0x0D)
```

Figure 4-7 Definition Used in IO Power Supply Open Control

4.2.5 Power Supply Mode

The definition used in power supply mode is shown in Figure 4-8.

Type of R_LPM_PowerSupplyModeGet function return value

```
typedef enum e_lpm_power_supply_mode
{
    LPM_POWER_SUPPLY_MODE_ALLPWON = 1,
    LPM_POWER_SUPPLY_MODE_EXFPWON = 2,
    LPM_POWER_SUPPLY_MODE_MINPWON = 4
} e_lpm_power_supply_mode_t;
```

Figure 4-8 Definition Used in Power Supply Mode

4.2.6 Low Power Consumption Mode

(1) RE01 1500KB Group

The definition used in snooze is shown in Figure 4-9.

Type of R_LPM_SnoozeSet function input parameter "p_cfg"

```
typedef struct st_lpm_snooze_cfg
{
    e_lpm_snz_enable_t              enable;
    uint32_t                        req;
    uint32_t                        snz_sstby;
    e_lpm_snz_wup_t                 wup;
    e_lpm_snz_dtc_enable_t          dtc_enable;
    e_lpm_snz_rx0_falling_edge_t    req_rx0;
} st_lpm_snooze_cfg_t;
```

```
typedef enum e_lpm_snz_enable
{
    LPM_SNZ_DISABLE     = 0,
    LPM_SNZ_ENABLE      = 1
} e_lpm_snz_enable_t;
```

```
#define LPM_SNZ_REQ_IRQ0                (0x00000001UL)
#define LPM_SNZ_REQ_IRQ1                (0x00000002UL)
#define LPM_SNZ_REQ_IRQ2                (0x00000004UL)
#define LPM_SNZ_REQ_IRQ3                (0x00000008UL)
#define LPM_SNZ_REQ_IRQ4                (0x00000010UL)
#define LPM_SNZ_REQ_IRQ5                (0x00000020UL)
#define LPM_SNZ_REQ_IRQ6                (0x00000040UL)
#define LPM_SNZ_REQ_IRQ7                (0x00000080UL)
#define LPM_SNZ_REQ_KINT                (0x00020000UL)
#define LPM_SNZ_REQ_ACMP0               (0x00800000UL)
#define LPM_SNZ_REQ_RTCALM              (0x01000000UL)
#define LPM_SNZ_REQ_RTCPRD              (0x02000000UL)
#define LPM_SNZ_REQ_AGT1UD              (0x10000000UL)
#define LPM_SNZ_REQ_AGT1CA              (0x20000000UL)
#define LPM_SNZ_REQ_AGT0CA              (0x40000000UL)
#define LPM_SNZ_REQ_NONE                (0x00000000UL)
#define LPM_SNZ_REQ_ALL                 (0x738200FFUL)
```

```
#define LPM_SNZ_SSTBY_AGT1UD            (0x01UL)
#define LPM_SNZ_SSTBY_DTCZ              (0x02UL)
#define LPM_SNZ_SSTBY_DTCN              (0x04UL)
#define LPM_SNZ_SSTBY_AD0MAT            (0x08UL)
#define LPM_SNZ_SSTBY_AD0UMT            (0x10UL)
#define LPM_SNZ_SSTBY_SCI0              (0x80UL)
#define LPM_SNZ_SSTBY_NONE              (0x00UL)
#define LPM_SNZ_SSTBY_ALL               (0x9FUL)
```

```
typedef enum e_lpm_snz_rx0_falling_edge
{
    LPM_RX0_FALLING_EDGE_DISABLE    = 0,
    LPM_RX0_FALLING_EDGE_ENABLE     = 1
} e_lpm_snz_rx0_falling_edge_t;
```

```
typedef enum e_lpm_snz_dtc_enable
{
    LPM_SNZ_DTC_DISABLE = 0,
    LPM_SNZ_DTC_ENABLE  = 2
} e_lpm_snz_dtc_enable_t;
```

```
typedef enum e_lpm_snz_wup
{
    LPM_SNZ_WUP_DTC             = 0x0D,
    LPM_SNZ_WUP_ADC140_WCMPM    = 0x27,
    LPM_SNZ_WUP_ADC140_WCMPUM   = 0x28,
    LPM_SNZ_WUP_DOC_DOPCI       = 0x38,
    LPM_SNZ_WUP_SCI0_AM         = 0x79,
    LPM_SNZ_WUP_SCI0_RXI_OR_ERI = 0x7A,
    LPM_SNZ_WUP_NONE            = 0x00
} e_lpm_snz_wup_t;
```

Figure 4-9 Definition Used in Snooze

The definition used in SSTBY mode is shown in Figure 4-10.

Type of R_LPM_SSTBYModeSetup function input parameter "p_cfg"

```
typedef struct st_lpm_sstby_cfg
{
    e_lpm_ope_to_sstby_power_supply_mode_t  ope_sstby;
    e_lpm_sstby_to_ope_power_supply_mode_t  sstby_ope;
    uint32_t                                wup;
} st_lpm_sstby_cfg_t;
```

```
typedef enum e_lpm_ope_to_sstby_power_supply_mode
{
    LPM_OPE_TO_SSTBY_ALLPWON_TO_EXFPWON,
    LPM_OPE_TO_SSTBY_ALLPWON_TO_MINPWON,
    LPM_OPE_TO_SSTBY_EXFPWON_TO_EXFPWON,
    LPM_OPE_TO_SSTBY_EXFPWON_TO_MINPWON,
    LPM_OPE_TO_SSTBY_MINPWON_TO_MINPWON
} e_lpm_ope_to_sstby_power_supply_mode_t;
```

```
typedef enum e_lpm_sstby_to_ope_power_supply_mode
{
    LPM_SSTBY_TO_OPE_EXFPWON_TO_ALLPWON,
    LPM_SSTBY_TO_OPE_EXFPWON_TO_EXFPWON,
    LPM_SSTBY_TO_OPE_MINPWON_TO_MINPWON
} e_lpm_sstby_to_ope_power_supply_mode_t;
```

```
#define LPM_SSTBY_WUP_IRQ0              (0x00000001UL)
#define LPM_SSTBY_WUP_IRQ1              (0x00000002UL)
#define LPM_SSTBY_WUP_IRQ2              (0x00000004UL)
#define LPM_SSTBY_WUP_IRQ3              (0x00000008UL)
#define LPM_SSTBY_WUP_IRQ4              (0x00000010UL)
#define LPM_SSTBY_WUP_IRQ5              (0x00000020UL)
#define LPM_SSTBY_WUP_IRQ6              (0x00000040UL)
#define LPM_SSTBY_WUP_IRQ7              (0x00000080UL)
#define LPM_SSTBY_WUP_IRQ8              (0x00000100UL)
#define LPM_SSTBY_WUP_IRQ9              (0x00000200UL)
#define LPM_SSTBY_WUP_IWDT              (0x00010000UL)
#define LPM_SSTBY_WUP_KEY               (0x00020000UL)
#define LPM_SSTBY_WUP_LVD1              (0x00040000UL)
#define LPM_SSTBY_WUP_LVDBAT            (0x00080000UL)
#define LPM_SSTBY_WUP_SOLDH             (0x00400000UL)
#define LPM_SSTBY_WUP_ACMP0             (0x00800000UL)
#define LPM_SSTBY_WUP_RTCALM            (0x01000000UL)
#define LPM_SSTBY_WUP_RTCPRD            (0x02000000UL)
#define LPM_SSTBY_WUP_CCCPRD            (0x04000000UL)
#define LPM_SSTBY_WUP_USBFS             (0x08000000UL)
#define LPM_SSTBY_WUP_AGT1UD            (0x10000000UL)
#define LPM_SSTBY_WUP_AGT1CA            (0x20000000UL)
#define LPM_SSTBY_WUP_AGT0CA            (0x40000000UL)
#define LPM_SSTBY_WUP_SOLDL             (0x80000000UL)
#define LPM_SSTBY_WUP_NONE              (0x00000000UL)
#define LPM_SSTBY_WUP_ALL               (0xFFCF03FFUL)
```

Figure 4-10 Definition Used in SSTBY Mode

The definition used in DSTBY mode is shown in Figure 4-11.

Type of R_LPM_DSTBYModeSetup function input parameter "p_cfg"

```
typedef struct st_lpm_dstby_cfg
{
    uint32_t     wup;
    uint32_t     wup_edge;
} st_lpm_dstby_cfg_t;
```

```
#define LPM_DSTBY_WUP_IRQ0DS          (0x0001UL)
#define LPM_DSTBY_WUP_IRQ1DS          (0x0002UL)
#define LPM_DSTBY_WUP_IRQ2DS          (0x0004UL)
#define LPM_DSTBY_WUP_IRQ3DS          (0x0008UL)
#define LPM_DSTBY_WUP_LVD1            (0x0100UL)
#define LPM_DSTBY_WUP_LVDBAT          (0x0200UL)
#define LPM_DSTBY_WUP_NMI             (0x1000UL)
#define LPM_DSTBY_WUP_CCC             (0x2000UL)
#define LPM_DSTBY_WUP_NONE            (0x0000UL)
#define LPM_DSTBY_WUP_ALL             (0x330FUL)
```

```
#define LPM_DSTBY_WUP_EDGE_IRQ0DS     (0x0001UL)
#define LPM_DSTBY_WUP_EDGE_IRQ1DS     (0x0002UL)
#define LPM_DSTBY_WUP_EDGE_IRQ2DS     (0x0004UL)
#define LPM_DSTBY_WUP_EDGE_IRQ3DS     (0x0008UL)
#define LPM_DSTBY_WUP_EDGE_LVD1       (0x0100UL)
#define LPM_DSTBY_WUP_EDGE_LVDBAT     (0x0200UL)
#define LPM_DSTBY_WUP_EDGE_NMI        (0x1000UL)
#define LPM_DSTBY_WUP_EDGE_NONE       (0x0000UL)
#define LPM_DSTBY_WUP_EDGE_ALL        (0x130FUL)
```

Type of R_LPM_DSTBYModeEntry function input parameter "io"

```
typedef enum e_lpm_dstby_io
{
    LPM_DSTBY_IO_CLEAR  = 0x00,
    LPM_DSTBY_IO_KEEP   = 0x40
} e_lpm_dstby_io_t;
```

Definition of R_LPM_DSTBYResetStatusGet function return value

```
#define LPM_DSTBY_RESET_IRQ0DS        (0x0001UL)
#define LPM_DSTBY_RESET_IRQ1DS        (0x0002UL)
#define LPM_DSTBY_RESET_IRQ2DS        (0x0004UL)
#define LPM_DSTBY_RESET_IRQ3DS        (0x0008UL)
#define LPM_DSTBY_RESET_LVD1          (0x0100UL)
#define LPM_DSTBY_RESET_LVDBAT        (0x0200UL)
#define LPM_DSTBY_RESET_NMI           (0x1000UL)
#define LPM_DSTBY_RESET_CCC           (0x2000UL)
#define LPM_DSTBY_RESET_NONE          (0x0000UL)
```

Figure 4-11 Definition Used in DSTBY Mode

(2) RE01 256KB Group

The definition used in snooze is shown in Figure 4-12.

Type of R_LPM_SnoozeSet function input parameter "p_cfg"

```
typedef struct st_lpm_snooze_cfg
{
    e_lpm_snz_enable_t              enable;
    uint32_t                        req;
    uint32_t                        snz_sstby;
    e_lpm_snz_wup_t                 wup;
    e_lpm_snz_dtc_enable_t          dtc_enable;
    e_lpm_snz_rx0_falling_edge_t    req_rx0;
} st_lpm_snooze_cfg_t;
```

```
typedef enum e_lpm_snz_enable
{
    LPM_SNZ_DISABLE     = 0,
    LPM_SNZ_ENABLE      = 1
} e_lpm_snz_enable_t;
```

```
#define LPM_SNZ_REQ_IRQ0                (0x00000001UL)
#define LPM_SNZ_REQ_IRQ1                (0x00000002UL)
#define LPM_SNZ_REQ_IRQ2                (0x00000004UL)
#define LPM_SNZ_REQ_IRQ3                (0x00000008UL)
#define LPM_SNZ_REQ_IRQ4                (0x00000010UL)
#define LPM_SNZ_REQ_IRQ5                (0x00000020UL)
#define LPM_SNZ_REQ_IRQ6                (0x00000040UL)
#define LPM_SNZ_REQ_IRQ7                (0x00000080UL)
#define LPM_SNZ_REQ_KINT                (0x00020000UL)
#define LPM_SNZ_REQ_AGTW0CA             (0x00800000UL)
#define LPM_SNZ_REQ_RTCALM              (0x01000000UL)
#define LPM_SNZ_REQ_RTCPRD              (0x02000000UL)
#define LPM_SNZ_REQ_AGT1UD              (0x10000000UL)
#define LPM_SNZ_REQ_AGT1CA              (0x20000000UL)
#define LPM_SNZ_REQ_AGT0CA              (0x40000000UL)
#define LPM_SNZ_REQ_NONE                (0x00000000UL)
#define LPM_SNZ_REQ_ALL                 (0x738200FFUL)
```

```
#define LPM_SNZ_SSTBY_AGT1UD            (0x01UL)
#define LPM_SNZ_SSTBY_DTCZ              (0x02UL)
#define LPM_SNZ_SSTBY_DTCN              (0x04UL)
#define LPM_SNZ_SSTBY_AD0MAT            (0x08UL)
#define LPM_SNZ_SSTBY_AD0UMT            (0x10UL)
#define LPM_SNZ_SSTBY_SCI0              (0x80UL)
#define LPM_SNZ_SSTBY_NONE              (0x00UL)
#define LPM_SNZ_SSTBY_ALL               (0x9FUL)
```

```
typedef enum e_lpm_snz_rx0_falling_edge
{
    LPM_RX0_FALLING_EDGE_DISABLE    = 0,
    LPM_RX0_FALLING_EDGE_ENABLE     = 1
} e_lpm_snz_rx0_falling_edge_t;
```

```
typedef enum e_lpm_snz_dtc_enable
{
    LPM_SNZ_DTC_DISABLE = 0,
    LPM_SNZ_DTC_ENABLE  = 2
} e_lpm_snz_dtc_enable_t;
```

```
typedef enum e_lpm_snz_wup
{
    LPM_SNZ_WUP_DTC             = 0x0D,
    LPM_SNZ_WUP_ADC140_WCMPM    = 0x27,
    LPM_SNZ_WUP_ADC140_WCMPUM   = 0x28,
    LPM_SNZ_WUP_DOC_DOPCI       = 0x38,
    LPM_SNZ_WUP_SCI0_AM         = 0x79,
    LPM_SNZ_WUP_SCI0_RXI_OR_ERI = 0x7A,
    LPM_SNZ_WUP_NONE            = 0x00
} e_lpm_snz_wup_t;
```

Figure 4-12 Definition Used in Snooze

The definition used in SSTBY mode is shown in Figure 4-13.

Type of R_LPM_SSTBYModeSetup function input parameter "p_cfg"

```
typedef struct st_lpm_sstby_cfg
{
    e_lpm_sstby_power_supply_mode_t power_supply;
    e_lpm_sstby_speed_mode_t        speed;
    uint32_t                        wup;
} st_lpm_sstby_cfg_t;
```

```
typedef enum e_lpm_sstby_power_supply_mode
{
    LPM_OPE_TO_SSTBY_EXFPWON     = 0,
    LPM_OPE_TO_SSTBY_MINPWON     = 1,
    LPM_OPE_TO_SSTBY_EXFPWON_VBB = 2,
    LPM_OPE_TO_SSTBY_MINPWON_VBB = 3
} e_lpm_sstby_power_supply_mode_t;
```

```
typedef enum e_lpm_sstby_speed_mode
{
    LPM_SSTBY_SPEED_MODE_OFF = 0,
    LPM_SSTBY_SPEED_MODE_ON  = 1
} e_lpm_sstby_speed_mode_t;
```

```
#define LPM_SSTBY_WUP_IRQ0              (0x00000001UL)
#define LPM_SSTBY_WUP_IRQ1              (0x00000002UL)
#define LPM_SSTBY_WUP_IRQ2              (0x00000004UL)
#define LPM_SSTBY_WUP_IRQ3              (0x00000008UL)
#define LPM_SSTBY_WUP_IRQ4              (0x00000010UL)
#define LPM_SSTBY_WUP_IRQ5              (0x00000020UL)
#define LPM_SSTBY_WUP_IRQ6              (0x00000040UL)
#define LPM_SSTBY_WUP_IRQ7              (0x00000080UL)
#define LPM_SSTBY_WUP_IRQ8              (0x00000100UL)
#define LPM_SSTBY_WUP_IRQ9              (0x00000200UL)
#define LPM_SSTBY_WUP_IWDT              (0x00010000UL)
#define LPM_SSTBY_WUP_KEY               (0x00020000UL)
#define LPM_SSTBY_WUP_LVD1              (0x00040000UL)
#define LPM_SSTBY_WUP_LVDBAT            (0x00080000UL)
#define LPM_SSTBY_WUP_SOLDH             (0x00400000UL)
#define LPM_SSTBY_WUP_AGTW0CA           (0x00800000UL)
#define LPM_SSTBY_WUP_RTCALM            (0x01000000UL)
#define LPM_SSTBY_WUP_RTCPRD            (0x02000000UL)
#define LPM_SSTBY_WUP_CCCPRD            (0x04000000UL)
#define LPM_SSTBY_WUP_WUPTOVI           (0x04000000UL)
#define LPM_SSTBY_WUP_AGTW1CA           (0x08000000UL)
#define LPM_SSTBY_WUP_AGT1UD            (0x10000000UL)
#define LPM_SSTBY_WUP_AGT1CA            (0x20000000UL)
#define LPM_SSTBY_WUP_AGT0CA            (0x40000000UL)
#define LPM_SSTBY_WUP_SOLDL             (0x80000000UL)
#define LPM_SSTBY_WUP_NONE              (0x00000000UL)
#define LPM_SSTBY_WUP_ALL               (0xFFCF03FFUL)
```

Figure 4-13 Definition Used in SSTBY Mode

The definition used in DSTBY mode is shown in Figure 4-14.

Type of R_LPM_DSTBYModeSetup function input parameter "p_cfg"

```
typedef struct st_lpm_dstby_cfg
{
    uint32_t     wup;
    uint32_t     wup_edge;
} st_lpm_dstby_cfg_t;
```

```
#define LPM_DSTBY_WUP_IRQ0DS          (0x0001UL)
#define LPM_DSTBY_WUP_IRQ1DS          (0x0002UL)
#define LPM_DSTBY_WUP_IRQ2DS          (0x0004UL)
#define LPM_DSTBY_WUP_IRQ3DS          (0x0008UL)
#define LPM_DSTBY_WUP_LVD1            (0x0100UL)
#define LPM_DSTBY_WUP_LVDBAT          (0x0200UL)
#define LPM_DSTBY_WUP_RTCI            (0x0400UL)
#define LPM_DSTBY_WUP_RTCA            (0x0800UL)
#define LPM_DSTBY_WUP_NMI             (0x1000UL)
#define LPM_DSTBY_WUP_CCC             (0x2000UL)
#define LPM_DSTBY_WUP_WUPT            (0x2000UL)
#define LPM_DSTBY_WUP_NONE            (0x0000UL)
#define LPM_DSTBY_WUP_ALL             (0x3F0FUL)
```

```
#define LPM_DSTBY_WUP_EDGE_IRQ0DS     (0x0001UL)
#define LPM_DSTBY_WUP_EDGE_IRQ1DS     (0x0002UL)
#define LPM_DSTBY_WUP_EDGE_IRQ2DS     (0x0004UL)
#define LPM_DSTBY_WUP_EDGE_IRQ3DS     (0x0008UL)
#define LPM_DSTBY_WUP_EDGE_LVD1       (0x0100UL)
#define LPM_DSTBY_WUP_EDGE_LVDBAT     (0x0200UL)
#define LPM_DSTBY_WUP_EDGE_NMI        (0x1000UL)
#define LPM_DSTBY_WUP_EDGE_NONE       (0x0000UL)
#define LPM_DSTBY_WUP_EDGE_ALL        (0x130FUL)
```

Type of R_LPM_DSTBYModeEntry function input parameter "io"

```
typedef enum e_lpm_dstby_io
{
    LPM_DSTBY_IO_CLEAR  = 0x00,
    LPM_DSTBY_IO_KEEP   = 0x40
} e_lpm_dstby_io_t;
```

Definition of R_LPM_DSTBYResetStatusGet function return value

```
#define LPM_DSTBY_RESET_IRQ0DS        (0x0001UL)
#define LPM_DSTBY_RESET_IRQ1DS        (0x0002UL)
#define LPM_DSTBY_RESET_IRQ2DS        (0x0004UL)
#define LPM_DSTBY_RESET_IRQ3DS        (0x0008UL)
#define LPM_DSTBY_RESET_LVD1          (0x0100UL)
#define LPM_DSTBY_RESET_LVDBAT        (0x0200UL)
#define LPM_DSTBY_RESET_RTCI          (0x0400UL)
#define LPM_DSTBY_RESET_RTCA          (0x0800UL)
#define LPM_DSTBY_RESET_NMI           (0x1000UL)
#define LPM_DSTBY_RESET_CCC           (0x2000UL)
#define LPM_DSTBY_RESET_WUPT          (0x2000UL)
#define LPM_DSTBY_RESET_NONE          (0x0000UL)
```

Figure 4-14 Definition Used in DSTBY Mode

4.2.7 Power Control Mode

The definition used in BackBias mode is shown in Figure 4-15.

Type of R_LPM_BackBiasModeEnable function input parameter "clock"

```
typedef enum e_lpm_vbb_clock
{
    LPM_VBB_CLOCK_LOCO  = 0,
    LPM_VBB_CLOCK_SOSC  = 1
} e_lpm_vbb_clock_t;
```

Type of R_LPM_BackBiasModeEnableStatusGet function return value

```
typedef enum e_lpm_vbb_enable_status
{
    LPM_VBB_DISABLED = 0,
    LPM_VBB_ENABLED  = 1
} e_lpm_vbb_enable_status_t;
```

Type of R_LPM_BackBiasModeGet function return value

```
typedef enum e_lpm_backbias_mode
{
    LPM_BACKBIAS_MODE_NORMAL    = 0,
    LPM_BACKBIAS_MODE_VBB       = 1
} e_lpm_backbias_mode_t;
```

Figure 4-15 Definition Used in BackBias Mode

4.3 Specification of the Function

The specifications and processing flow of each function of the R_LPM driver is shown.

The table of specification of the function in this section corresponds to the contents described in Doxygen. In the specification of the function table, low leakage current mode is described as BackBias to contrast with Doxygen.

For processing flow error check, only error condition is listed and detailed check method is omitted.

In the conditional branch of the processing flow, register names and variable names are described in order to clarify the targets to be used for the condition judgment. However, judgment method does not necessarily match the description in the processing flow.

The notes to each specification of the function are described below.

∗1. This function uses the R_SYS_RegisterProtectEnable() function and R_SYS_RegisterProtectDisable() function of the R_SYSTEM driver when “1” is set in LPM_CFG_REGISTER_PROTECTION_ENABLE.
Before this function is executed, it is necessary to execute the R_SYS_Initialize() function.

∗2. This function uses the R_SYS_EnterCriticalSection() function and R_SYS_ExitCriticalSection() function of the R_SYSTEM driver when "1" is set in LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE.
Before this function is executed, it is necessary to execute the R_SYS_Initialize() function.

4.3.1 R_LPM_GetVersion Function

Table 4-6 Specification of the R_LPM_GetVersion Function

Outline	uint32_t R_LPM_GetVersion(void)
Description	This function gets the version number of R_LPM.
Input	None
Return value	Version number
Remark	-

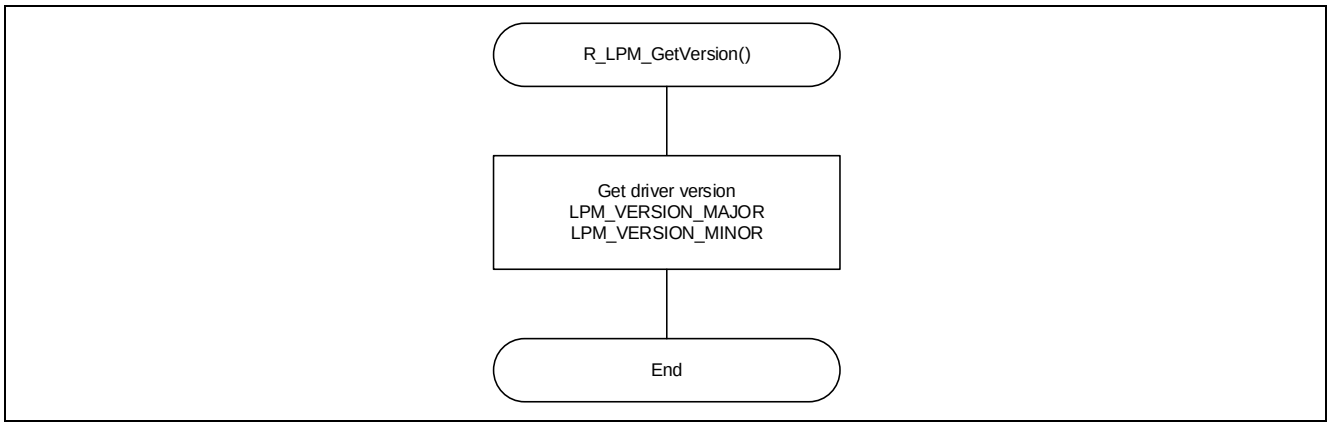

Figure 4-16 R_LPM_GetVersion Function Processing Flow

4.3.2 R_LPM_Initialize Function

Table 4-7 Specification of the R_LPM_Initialize Function

Outline	void R_LPM_Initialize(void)
Description	This function initializes R_LPM function. Execute variable initialization only once after reset. If executed multiple times, the function exits without any performance.
Input	None
Return value	None
Remark	-

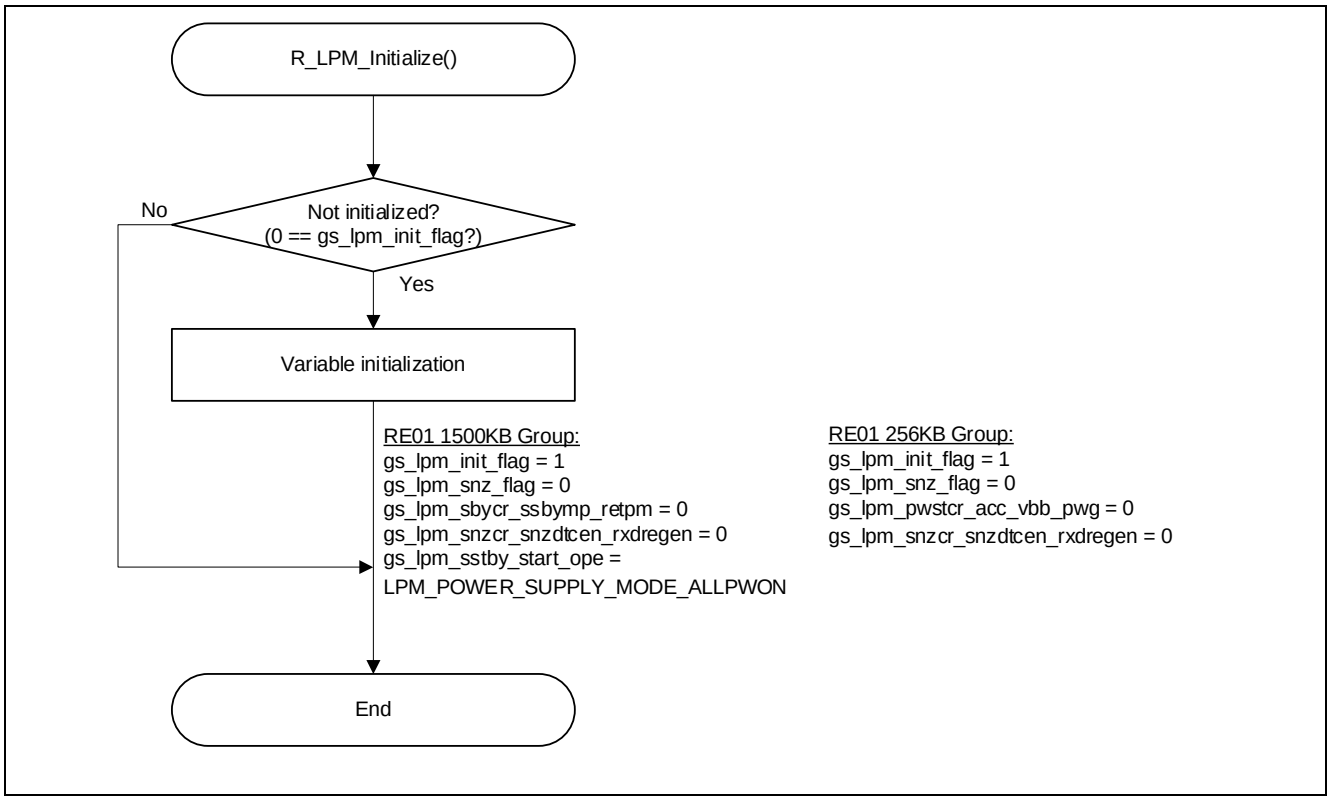

Figure 4-17 R_LPM_Initialize Function Processing Flow

4.3.3 R_LPM_ModuleStart Function

Table 4-8 Specification of the R_LPM_ModuleStart Function

Outline	int32_t R_LPM_ModuleStart(e_lpm_mstp_t module)
Description	This function starts module (cancel module stop). Starting a module enables clocks to the peripheral and allows registers to set. *2 Only one module can be specified.
Input Note	e_lpm_mstp_t module [in] Set the module which releases the module stop status
Return value	Success (0)
	Error (-1) : The specified module is not started. An error occurs if the following conditions are detected. <RE01 1500KB Group> — The input parameter "module" exceeds the definition range of e_lpm_mstp_t. — LPM_MSTP_USB is specified for input parameter "module" and it is not operating in BOOST mode. <RE01 256KB Group> — The input parameter "module" exceeds the definition range of e_lpm_mstp_t.
Remark	-

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.2 Module Stop.

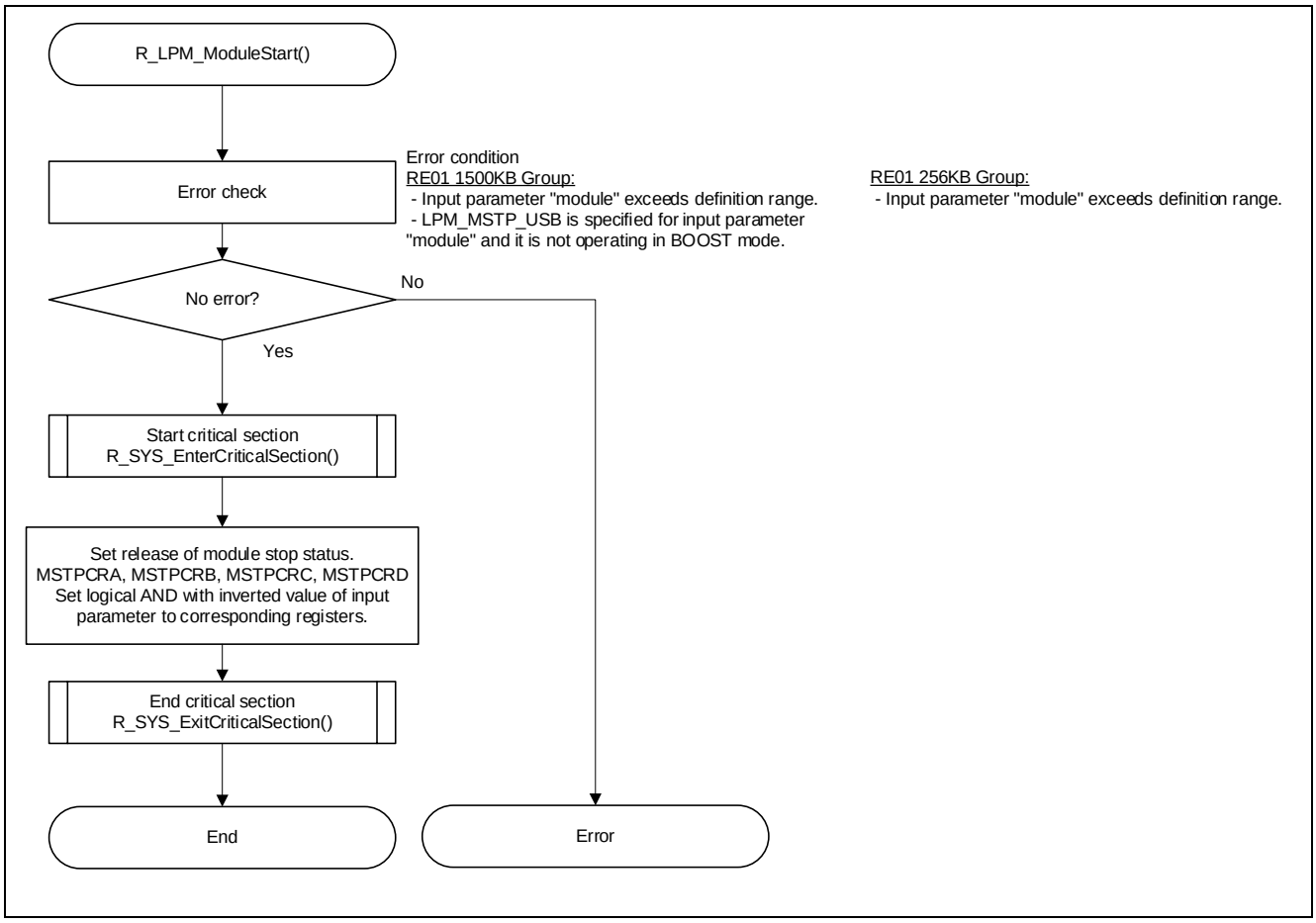

Figure 4-18 R_LPM_ModuleStart Function Processing Flow

4.3.4 R_LPM_ModuleStop Function

Table 4-9 Specification of the R_LPM_ModuleStop Function

Outline	int32_t R_LPM_ModuleStop(e_lpm_mstp_t module)
Description	This function stops module (enter module stop). Stopping a module disables clocks to the peripheral to save power. *2 R_LPM_FastModuleStop function can specify multiple modules in one execution of function, however this function can specify only one module. This function uses the R_SYS_ResourceLock() function and R_SYS_ResourceUnLock() function of the R_SYSTEM driver. Before this function is executed, it is necessary to execute the R_SYS_Initialize() function.
Input Note	e_lpm_mstp_t module [in] Set the module which transitions to the module stop state
Return value	Success (0) : The specified module is stopped.
	Error (-1) : The specified module is not stopped. An error occurs if the following conditions are detected. — The input parameter "module" exceeds the definition range of e_lpm_mstp_t. — Other peripheral modules sharing the module stop bit with the module specified by the input parameter "module" are in operation. Whether a peripheral module is in operation or not is determined by the execution state of the R_SYS_ResourceLock function.
Remark	-

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.2 Module Stop.

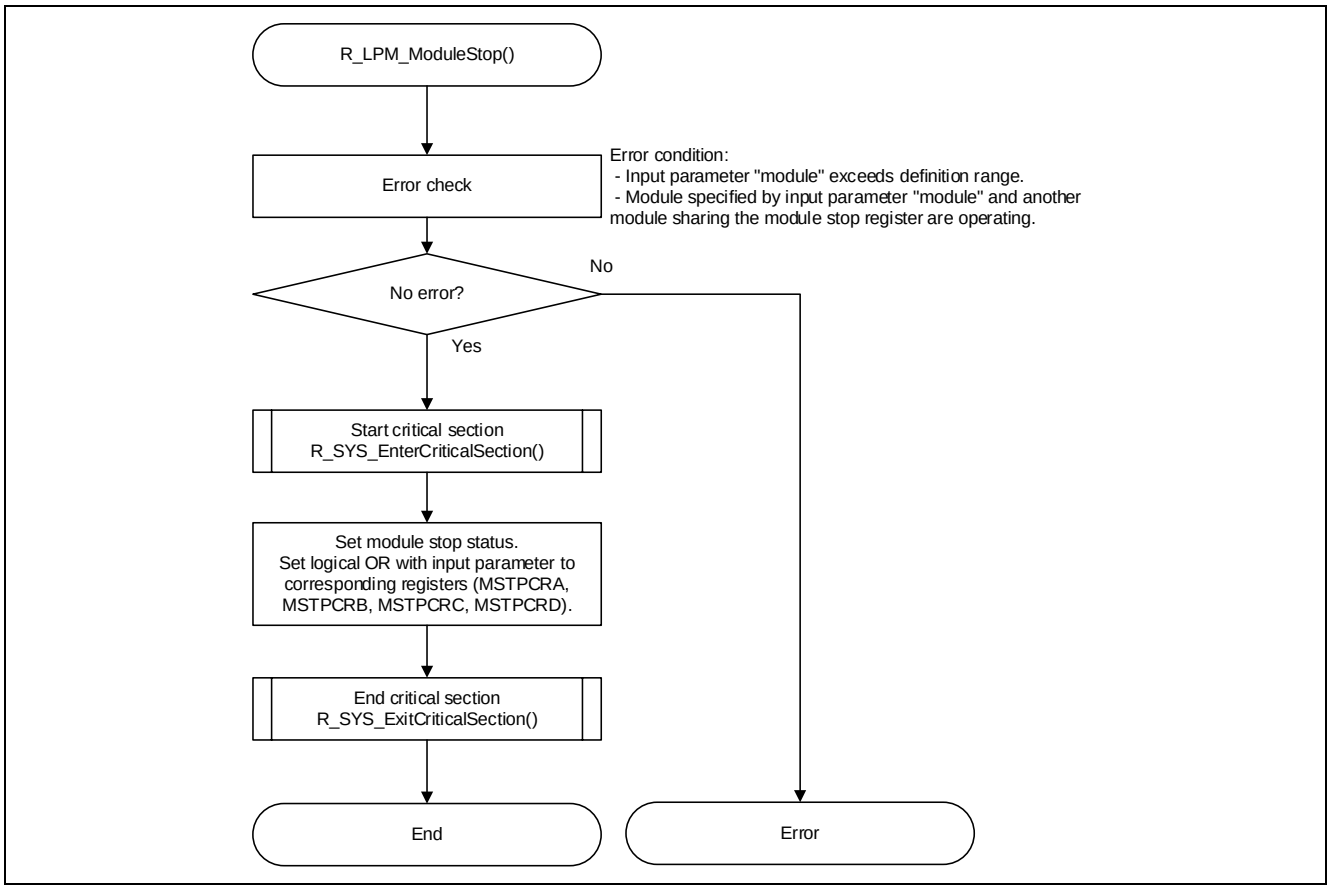

Figure 4-19 R_LPM_ModuleStop Function Processing Flow

4.3.5 R_LPM_FastModuleStart Function

Table 4-10 Specification of the R_LPM_FastModuleStart Function

Outline	int32_t R_LPM_FastModuleStart(uint32_t module_a, uint32_t module_b, uint32_t module_c, uint32_t module_d)
Description	This function starts modules (cancel module stop). Starting modules enable clocks to the peripheral and allows registers to be set. *2 Start the module set in the input parameters "module_a", "module_b", "module_c" and "module_d". Multiple modules can be set for each input parameter by logical OR.
Input Note	uint32_t module_a[in] : Defined by LPM_FAST_MSTPT_A. Set the module which releases the module stop status
	uint32_t module_b[in] : Defined by LPM_FAST_MSTPT_B. Set the module which releases the module stop status
	uint32_t module_c[in] : Defined by LPM_FAST_MSTPT_C. Set the module which releases the module stop status
	uint32_t module_d[in] : Defined by LPM_FAST_MSTPT_D. Set the module which releases the module stop status
Return value	Success (0) : The specified modules are started.
	Error (-1) : The specified modules are not started. An error occurs if the following conditions are detected. <RE01 1500KB Group> — LPM_MSTP_USB is specified for input parameter "module_b" and it is not operating in BOOST mode. <RE01 256KB Group> — No error. The Return value of this function is always "Success(0)".
Remark	-

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.2 Module Stop.

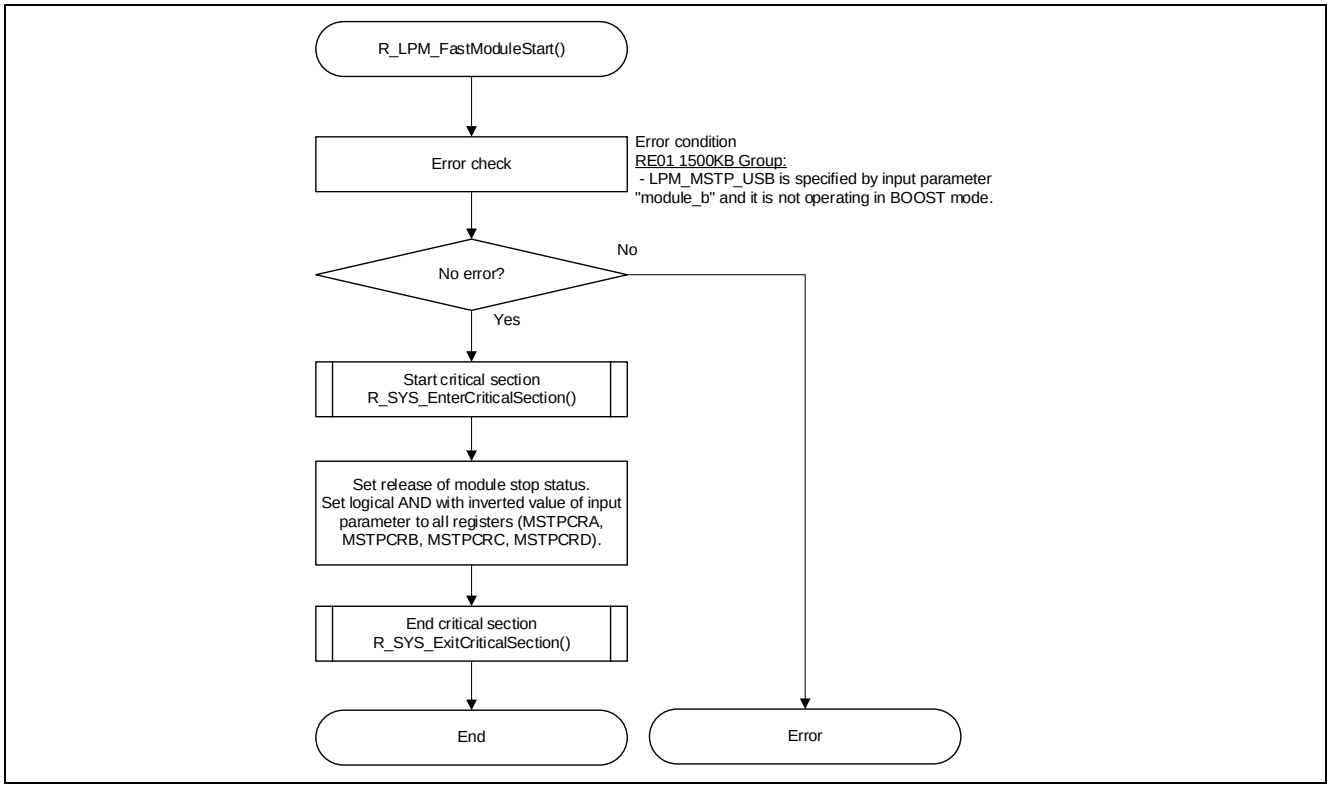

Figure 4-20 R_LPM_FastModuleStart Function Processing Flow

4.3.6 R_LPM_FastModuleStop Function

Table 4-11 Specification of the R_LPM_FastModuleStop Function

Outline	int32_t R_LPM_FastModuleStop(uint32_t module_a, uint32_t module_b, uint32_t module_c, uint32_t module_d)
Description	This function stops modules (enter module stop). Stopping modules disable clocks to the peripheral to save power. *2 Stop the module set in the input parameters "module_a", "module_b", "module_c" and "module_d". Multiple factors can be set for each input parameter by logical OR. This function uses the R_SYS_ResourceLock() function and R_SYS_ResourceUnLock() function of the R_SYSTEM driver. Before this function is executed, it is necessary to execute the R_SYS_Initialize() function.
Input Note	uint32_t module_a[in] : Defined by LPM_FAST_MSTPT_A. Set the module which transitions to the module stop state
	uint32_t module_b[in] : Defined by LPM_FAST_MSTPT_B. Set the module which transitions to the module stop state
	uint32_t module_c[in] : Defined by LPM_FAST_MSTPT_C. Set the module which transitions to the module stop state
	uint32_t module_d[in] : Defined by LPM_FAST_MSTPT_D. Set the module which transitions to the module stop state
Return value	Success (0) : The specified modules are stopped.
	Error (-1) : The specified modules are not stopped. An error occurs if the following conditions are detected. — Other peripheral modules sharing the module stop bit with the module set for the input parameter "module_a", "module_b", "module_c" and "module_d" are in operation. Whether a peripheral module is in operation or not is determined by the execution state of the R_SYS_ResourceLock function.
Remark	-

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.2 Module Stop.

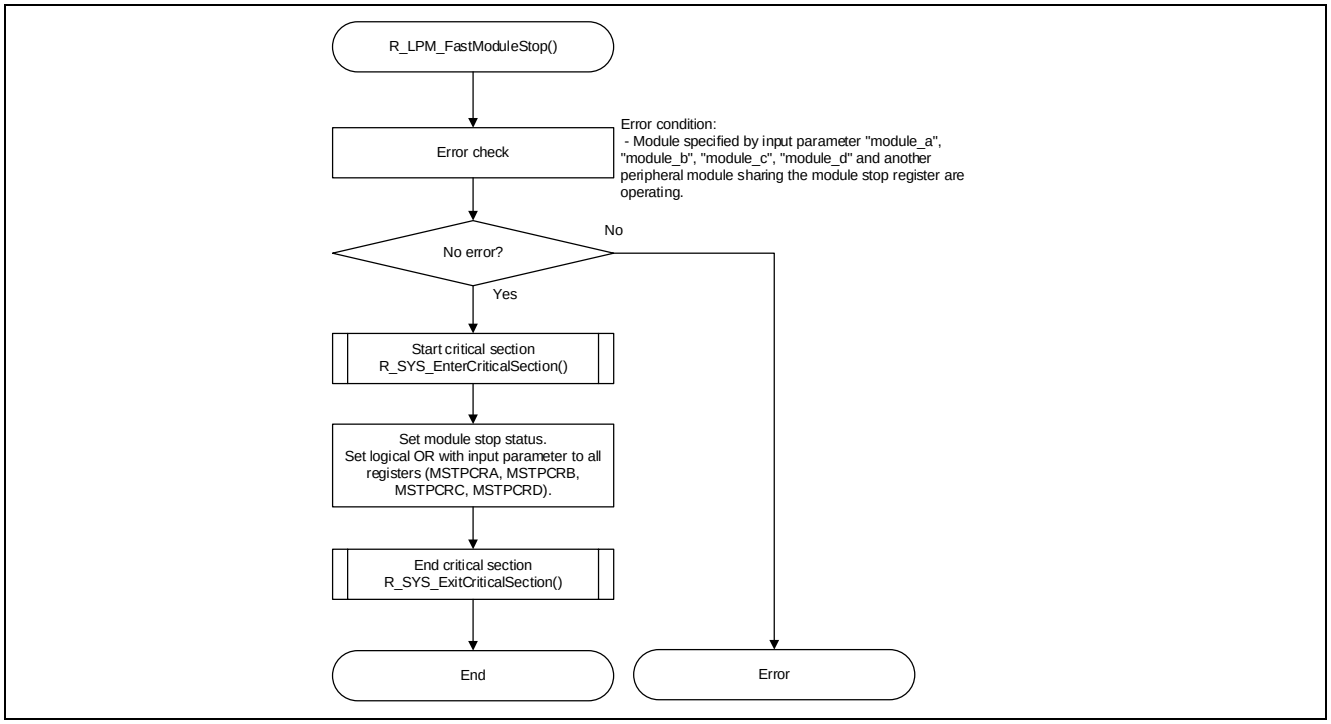

Figure 4-21 R_LPM_FastModuleStop Function Processing Flow

4.3.7 R_LPM_BackBiasModeEnable Function

Table 4-12 Specification of the R_LPM_BackBiasModeEnable Function

Outline	int32_t R_LPM_BackBiasModeEnable(e_lpm_vbb_clock_t clock)
Description	This function sets up Back Bias control. *1 Before this function is executed, the clock used in Back Bias mode must be oscillated. This function sets enable of Back Bias control. But this function does not wait for setup completion. Therefore, after executing this function, before executing Entry to Back Bias mode, it is necessary to execute R_LPM_BackBiasModeEnableStatusGet() function to confirm that setup is completed.
Input	e_lpm_vbb_clock_t clock[in] Set the clock for Back Bias control.
Return value	Success (0) : Setup of Back Bias control succeeded.
	Error (-1) : Setup of Back Bias control failed. An error occurs if the following conditions are detected. — Specified clock is not oscillating. — The input parameter "clock" exceeds the definition range of e_lpm_vbb_clock_t.
Remark	-

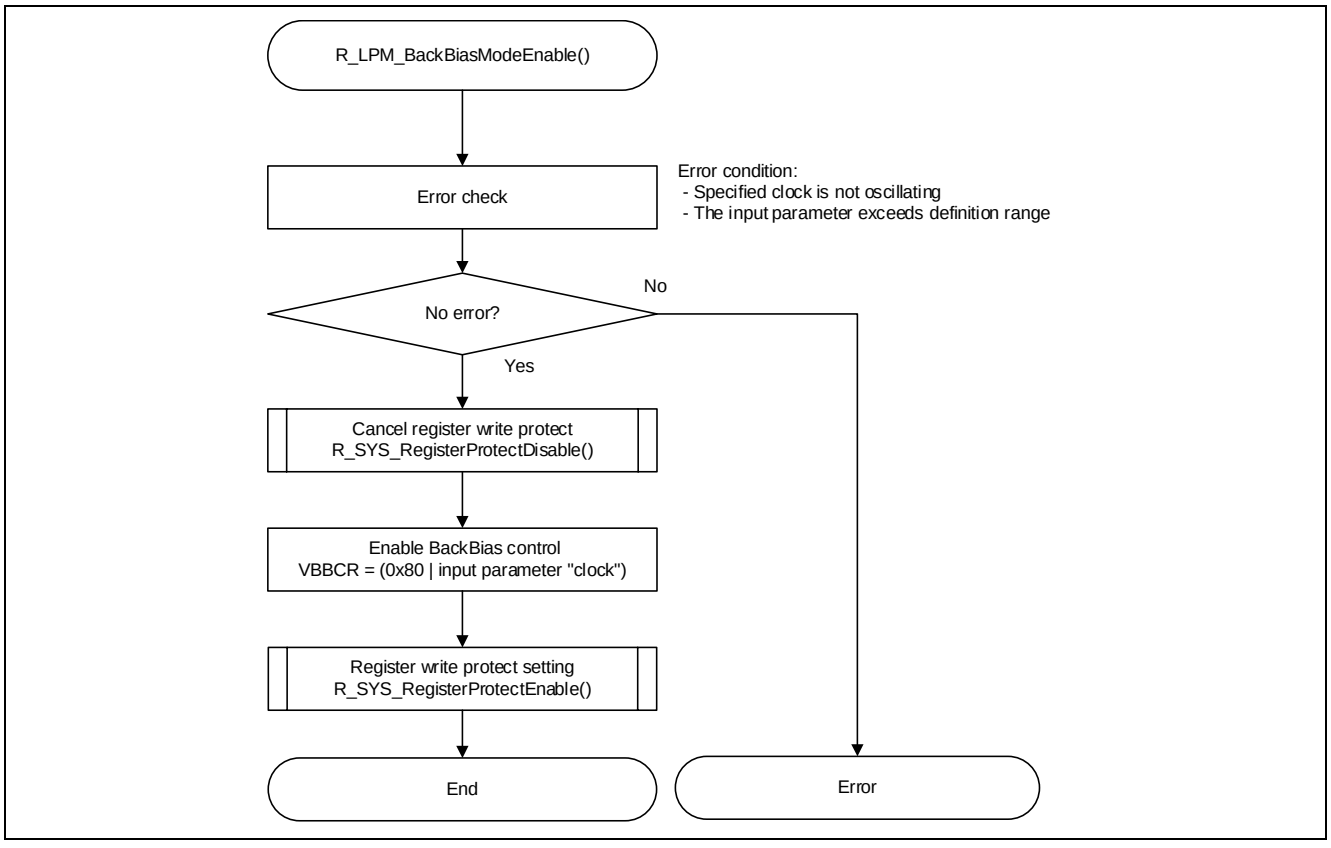

Figure 4-22 R_LPM_BackBiasModeEnable Function Processing Flow

4.3.8 R_LPM_BackBiasModeDisable Function

Table 4-13 Specification of the R_LPM_BackBiasModeDisable Function

Outline	int32_t R_LPM_BackBiasModeDisable(void)
Description	This function cancels the setup state of Back Bias mode. *1 Before this function is executed, operation in Back Bias mode must be stopped.
Input	None
Return value	Success (0) : Back Bias mode cancel succeeded.
	Error (-1) : Back Bias mode cancel failed. An error occurs if the following conditions are detected. — Operating in Back Bias mode.
Remark	-

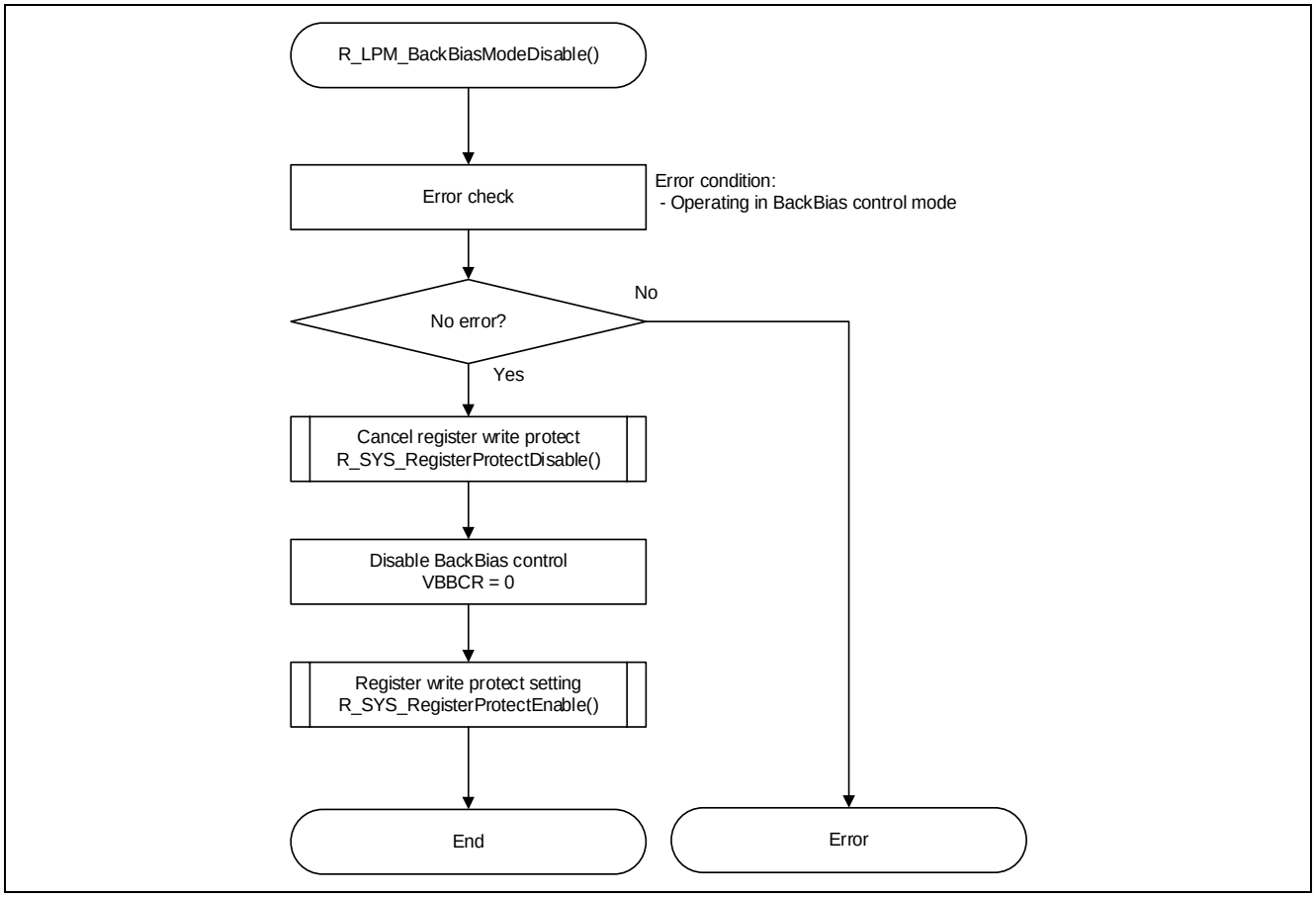

Figure 4-23 R_LPM_BackBiasModeDisable Function Processing Flow

4.3.9 R_LPM_BackBiasModeEnableStatusGet Function

Table 4-14 Specification of the R_LPM_BackBiasModeEnableStatusGet Function

Outline	e_lpm_vbb_enable_status_t R_LPM_BackBiasModeEnableStatusGet(void)
Description	This function gets the setup status of Back Bias control. After executing the R_LPM_BackBiasModeEnable() function, please confirm that setup is completed with this function before entering Back Bias mode.
Input	None
Return value	LPM_VBB_DISABLED : Back Bias control setup is not complete.
	LPM_VBB_ENABLED : Back Bias control setup is completed.
Remark	-

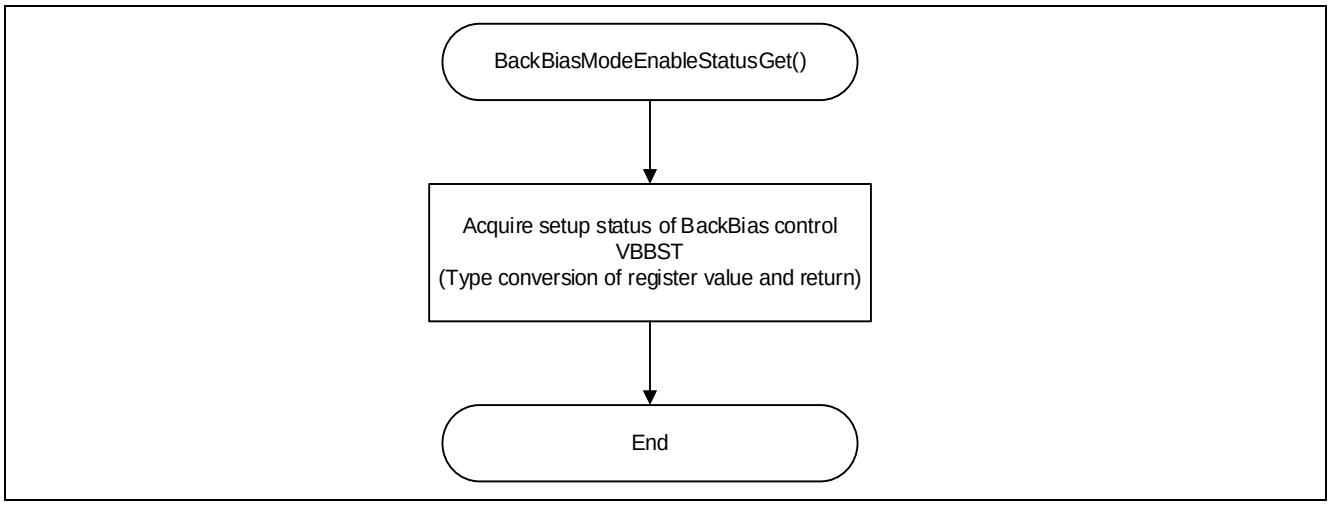

Figure 4-24 R_LPM_BackBiasModeEnableStatusGet Function Processing Flow

4.3.10 R_LPM_BackBiasModeEntry Function

Table 4-15 Specification of the R_LPM_BackBiasModeEntry Function

Outline	int32_t R_LPM_BackBiasModeEntry(void)
Description	This function transits to Back Bias mode. *1 Before this function is executed, it is necessary to execute the R_LPM_BackBiasModeEnable() function to complete the setup. Before this function is executed, ICLK must be switched to the clock specified by the R_LPM_BackBiasModeEnable() function. Before this function is executed, it is necessary to execute the R_SYS_32kHzSpeedModeSet() function and set Subosc-Speed mode. Note This function uses internal functions of the R_SYSTEM driver.Before this function is executed, it is necessary to execute the R_SYS_Initialize() function.
Input	None
Return value	Success (0) : Back Bias mode transition succeeded.
	Error (-1) : Back Bias mode transition failed. An error occurs if the following conditions are detected. <RE01 1500KB Group> — BackBias control setup is incomplete. — Operating in BOOST mode. — It is not operating in Subosc-Speed mode. — The specified clock in setup is not set to ICLK. — Back Bias mode transition can not be set to hardware. <RE01 256KB Group> — BackBias control setup is incomplete. — The specified clock in setup is not set to ICLK. — Back Bias mode transition can not be set to hardware.
Remark	<RE01 1500KB Group> When using the device of WS1, there are the following restrictions. (Restriction No.60) After executing the transition to Software Standby mode in Normal operation mode with SOSC-Speed, if you change to Back Bias mode without changing the speed mode, you can not recover from Software Standby mode. Before switching to the Back Bias mode, processing to set SOSC-Speed mode again after switching to Low-Speed is performed. This function uses the R_SYS_LowSpeedModeSet() function and R_SYS_32kHzSpeedModeSet() function of the R_SYSTEM driver. *2

Note: Only for RE01 1500KB group.

(1) RE01 1500KB Group

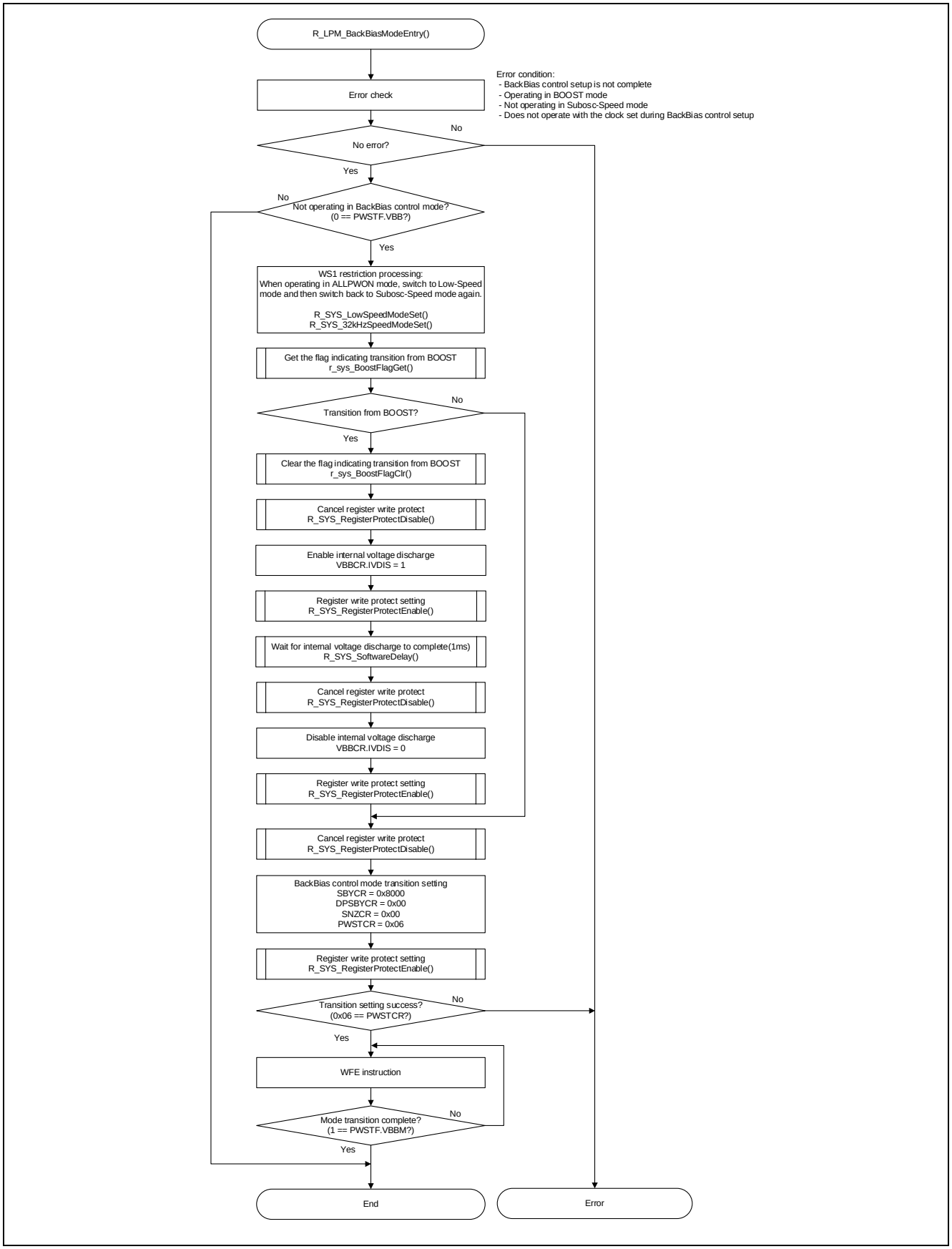

Figure 4-25 R_LPM_BackBiasModeEntry Function Processing Flow

(2) RE01 256KB Group

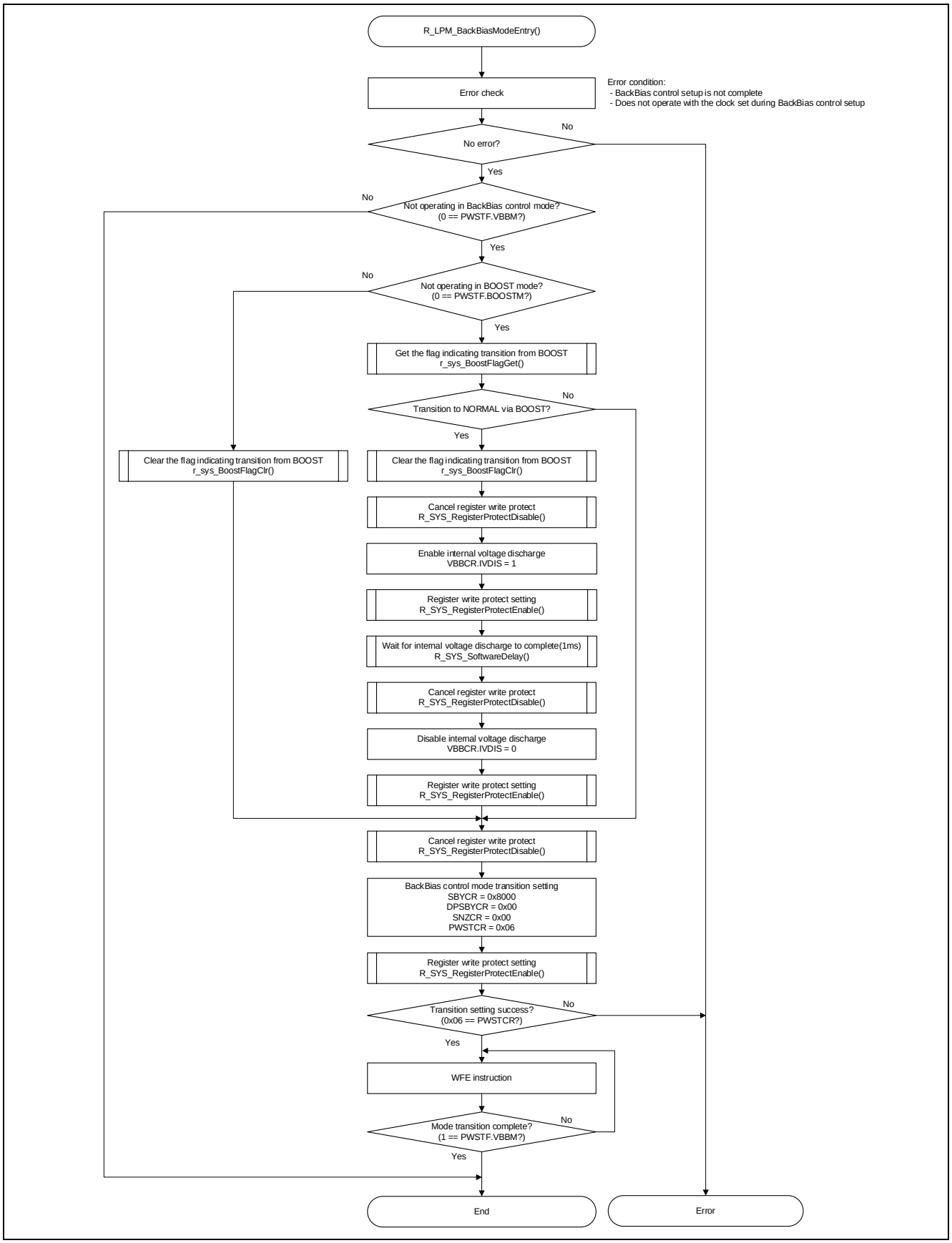

Figure 4-26 R_LPM_BackBiasModeEntry Function Processing Flow

4.3.11 R_LPM_BackBiasModeExit Function

Table 4-16 Specification of the R_LPM_BackBiasModeExit Function

Outline	int32_t R_LPM_BackBiasModeExit(void)
Description	This function transits from Back Bias mode to Normal mode. *1
Input	None
Return value	Success (0) : Transition from Back Bias mode to Normal mode succeeded.
	Error (-1) : Transition from Back Bias mode to Normal mode failed. An error occurs if the following conditions are detected. — Normal mode transition can not be set to hardware.
Remark	<RE01 256KB Group> When transitioning from Back Bias mode to BOOST mode, use R_SYS_BoostSpeedModeSet() function of the R_SYSTEM driver.

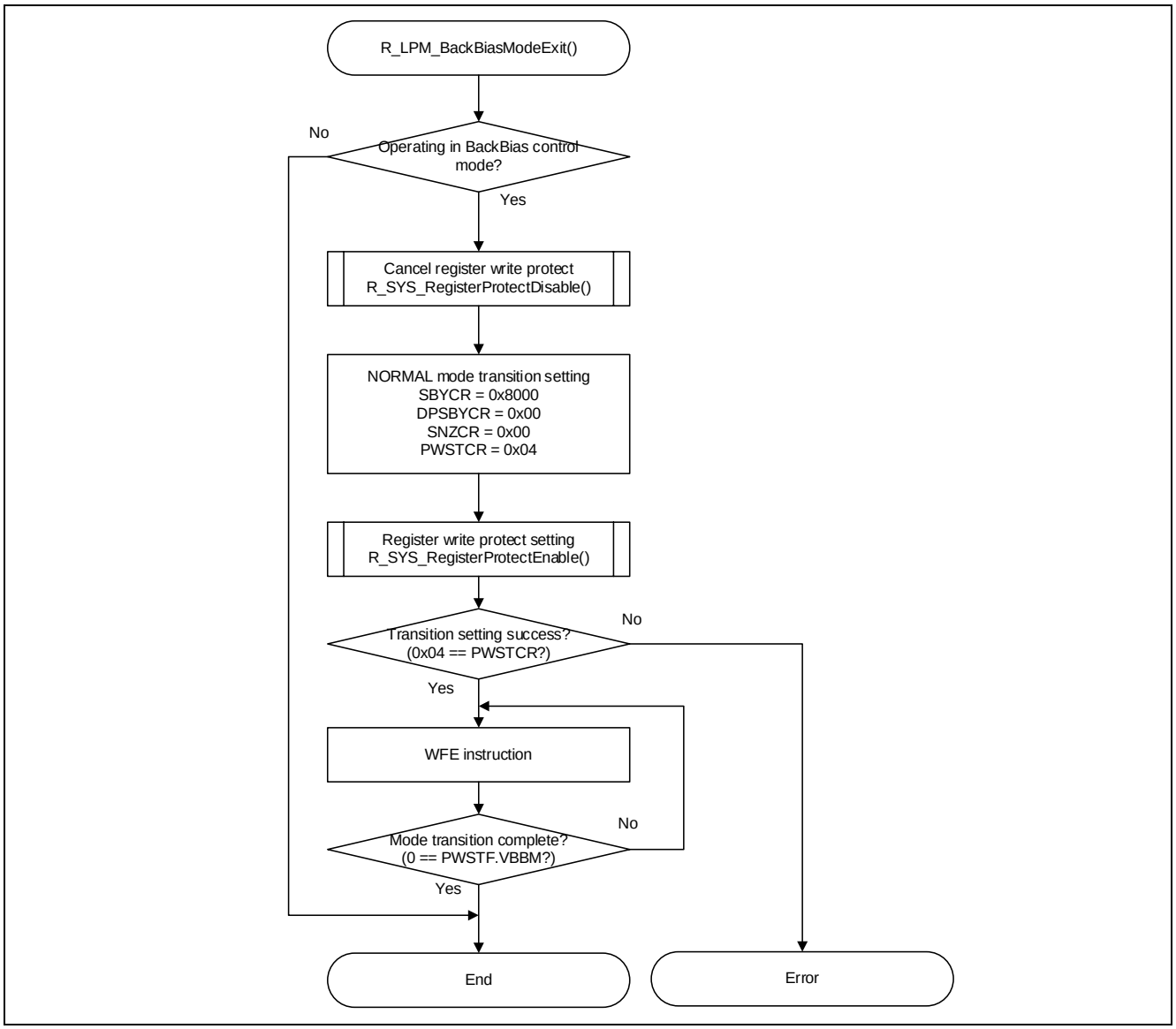

Figure 4-27 R_LPM_BackBiasModeExit Function Processing Flow

4.3.12 R_LPM_BackBiasModeGet Function

Table 4-17 Specification of the R_LPM_BackBiasModeGet Function

Outline	e_lpm_backbias_mode_t R_LPM_BackBiasModeGet(void)
Description	This function gets the operating mode, Back Bias mode or Normal mode.
Input	None
Return value	LPM_BACKBIAS_MODE_NORMAL : Operating in Normal mode or BOOST mode.
	LPM_BACKBIAS_MODE_VBB : Operating in Back Bias mode.
Remark	-

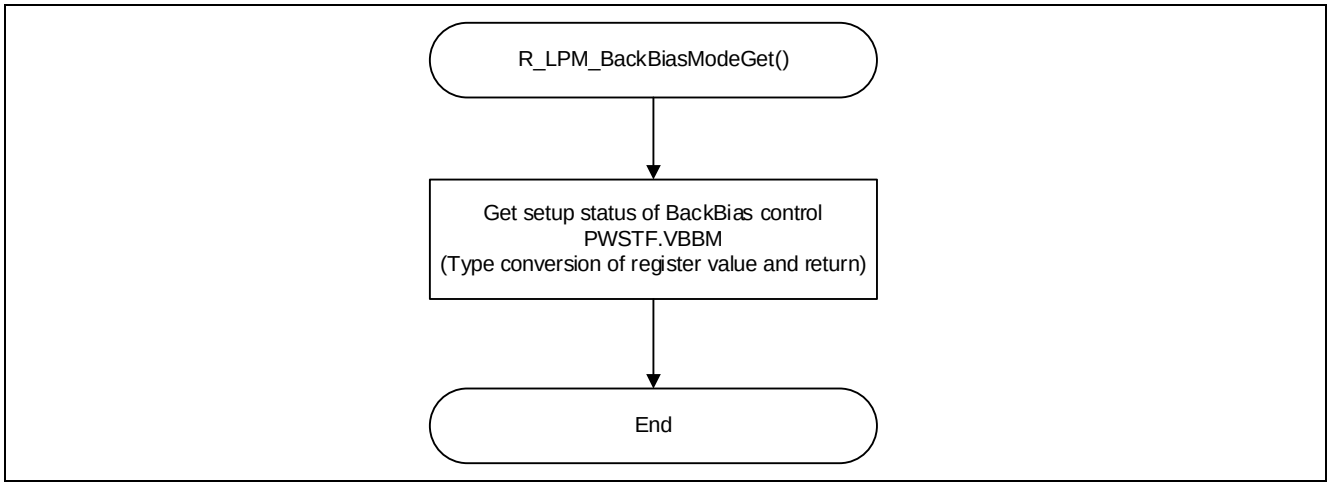

Figure 4-28 R_LPM_BackBiasModeGet Function Processing Flow

4.3.13 R_LPM_PowerSupplyModeAllpwonSet Function

Table 4-18 Specification of the R_LPM_PowerSupplyModeAllpwonSet Function

Outline	int32_t R_LPM_PowerSupplyModeAllpwonSet(void)
Description	This function supports power mode transition to ALLPWON. *1 <RE01 1500KB Group> When this function is executed in the MINPWON mode state, this function makes a transition from the MINPWON mode to the EXFPWON mode and then makes a transition from the EXFPWON mode to the ALLPWON mode.
	This function executes the r_lpm_wait_release_reset_iso2 () function after transition from the MINPWON mode to the ALLPWON mode in order to wait for reset release of the ISO2 area stopped in the MINPWON mode. This process is required when the PCLKB division setting is 4 to 64 division. The r_lpm_wait_release_reset_iso2 () function is implemented as a WEAK function in the R_LPM driver. It can disable the corresponding function in R_LPM driver by implementing the non-weak function with the same name.
Input	None
Return value	Success (0) : Transition to the specified power supply mode.
	Error (-1) : Not transition to the specified power supply mode. An error occurs if the following conditions are detected. — Power supply mode transition can not be set to hardware. — It is operating with a clock exceeding 4 MHz.
Remark	-

(1) RE01 1500KB Group

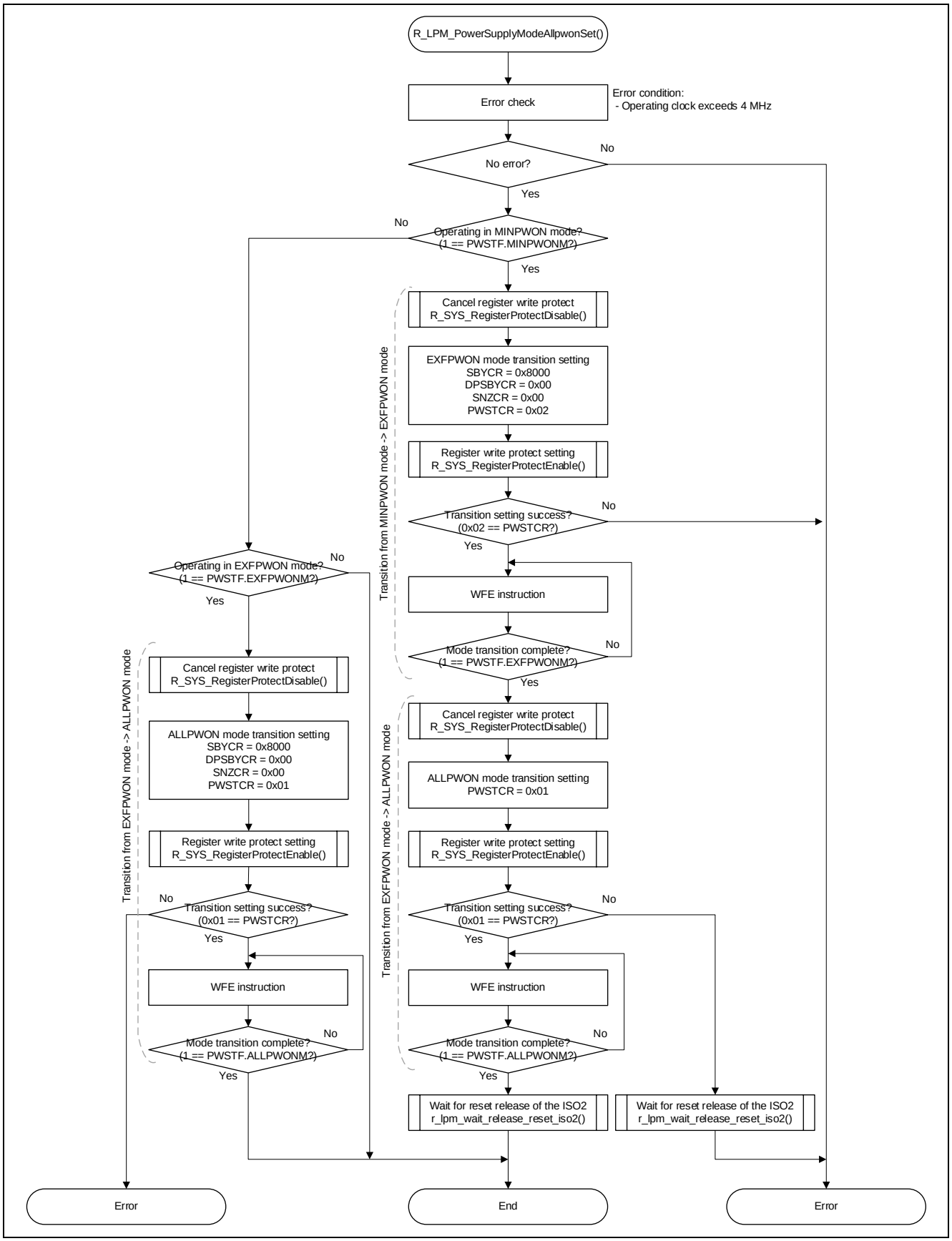

Figure 4-29 R_LPM_PowerSupplyModeAllpwonSet Function Processing Flow

(2) RE01 256KB Group

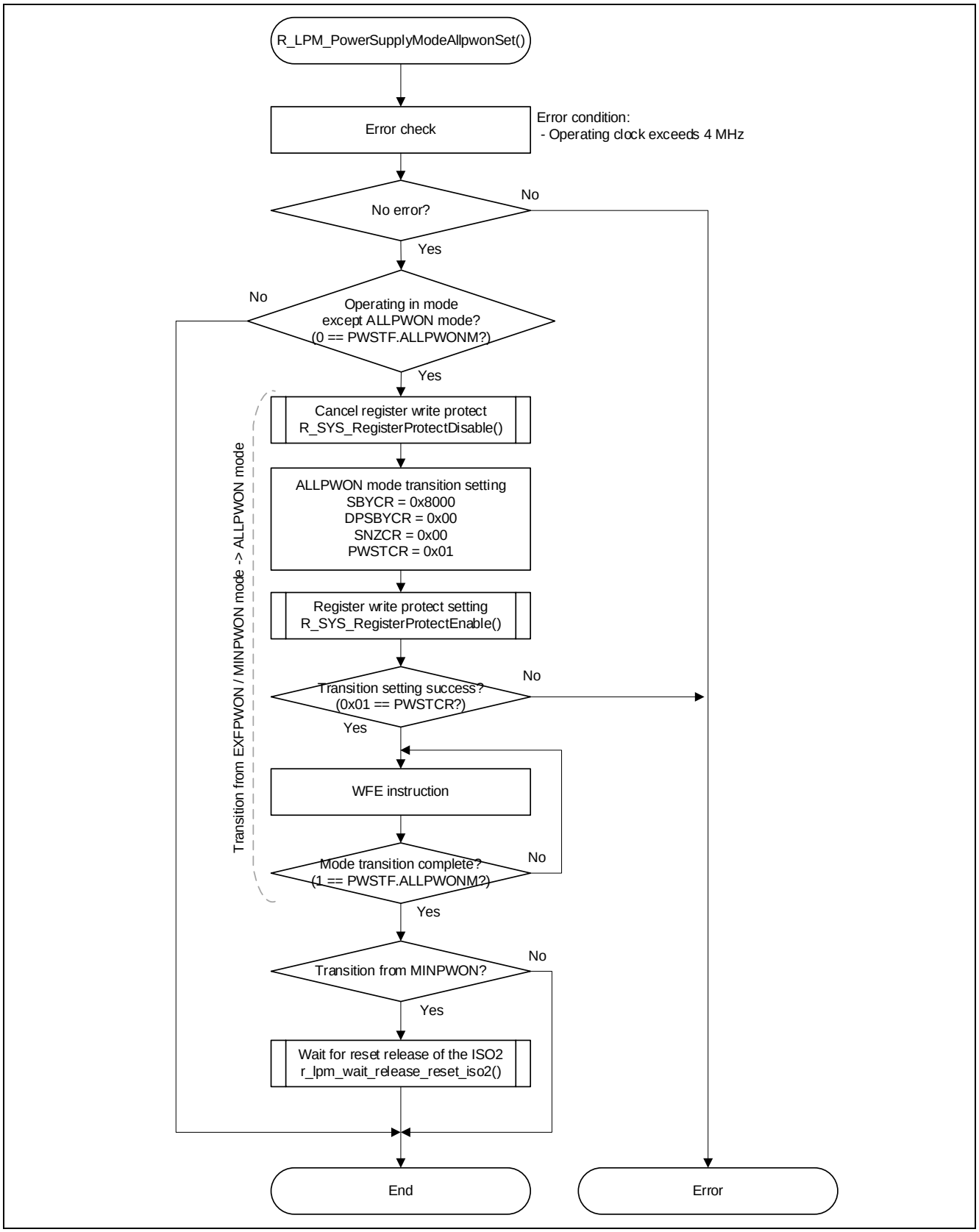

Figure 4-30 R_LPM_PowerSupplyModeAllpwonSet Function Processing Flow

4.3.14 R_LPM_PowerSupplyModeExfpwonSet Function

Table 4-19 Specification of the R_LPM_PowerSupplyModeExfpwonSet Function

Outline	int32_t R_LPM_PowerSupplyModeExfpwonSet(void)
Description	This function supports power mode transition to EXFPWON. *1 This function executes the r_lpm_wait_release_reset_iso2 () function after transition from the MINPWON mode to the EXFPWON mode in order to wait for reset release of the ISO2 area stopped in the MINPWON mode. This process is required when the PCLKB division setting is 4 to 64 division. The r_lpm_wait_release_reset_iso2 () function is implemented as a WEAK function in the R_LPM driver. It can disable the corresponding function in R_LPM driver by implementing the non-weak function with the same name.
Input	None
Return value	Success (0) : Transition to the specified power supply mode.
	Error (-1) : Not transition to the specified power supply mode. An error occurs if the following conditions are detected. — Power supply mode transition can not be set to hardware. — It is operating in BOOST mode.
Remark	-

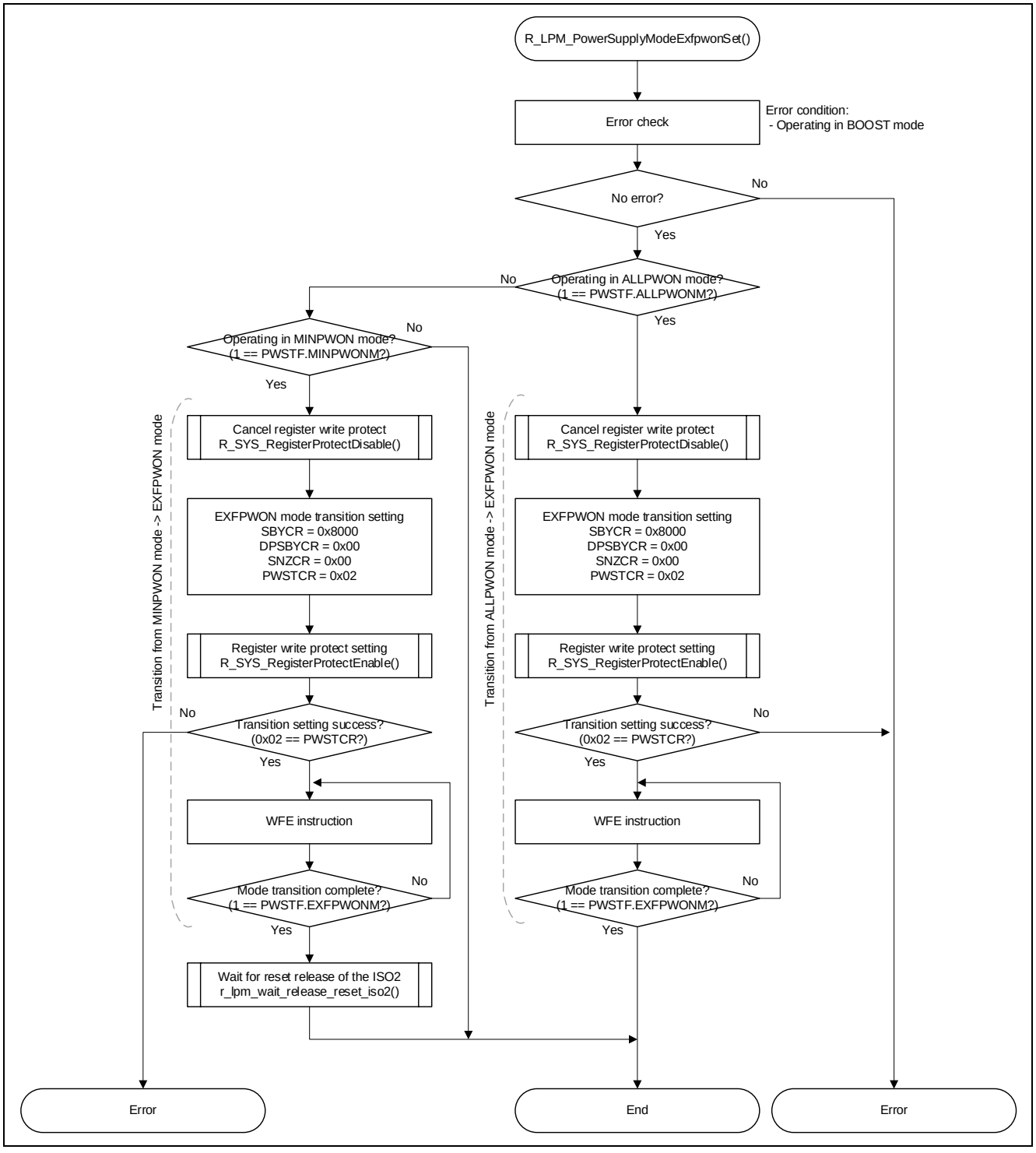

Figure 4-31 R_LPM_PowerSupplyModeExfpwonSet Function Processing Flow

4.3.15 R_LPM_PowerSupplyModeMinpwonSet Function

Table 4-20 Specification of the R_LPM_PowerSupplyModeMinpwonSet Function

Outline	int32_t R_LPM_PowerSupplyModeMinpwonSet(void)
Description	This function supports power mode transition to MINPWON. *1 <RE01 1500KB Group> When this function is executed in the state of ALLPWON mode, this function makes a transition to ALLPWON mode or EXFPWON mode and then makes transition from EXFPWON mode to MINPWON mode.
Input	None
Return value	Success (0) : Transition to the specified power supply mode.
	Error (-1) : Not transition to the specified power supply mode. An error occurs if the following conditions are detected. — Power supply mode transition can not be set to hardware. — It is operating in BOOST mode.
Remark	-

(1) RE01 1500KB Group

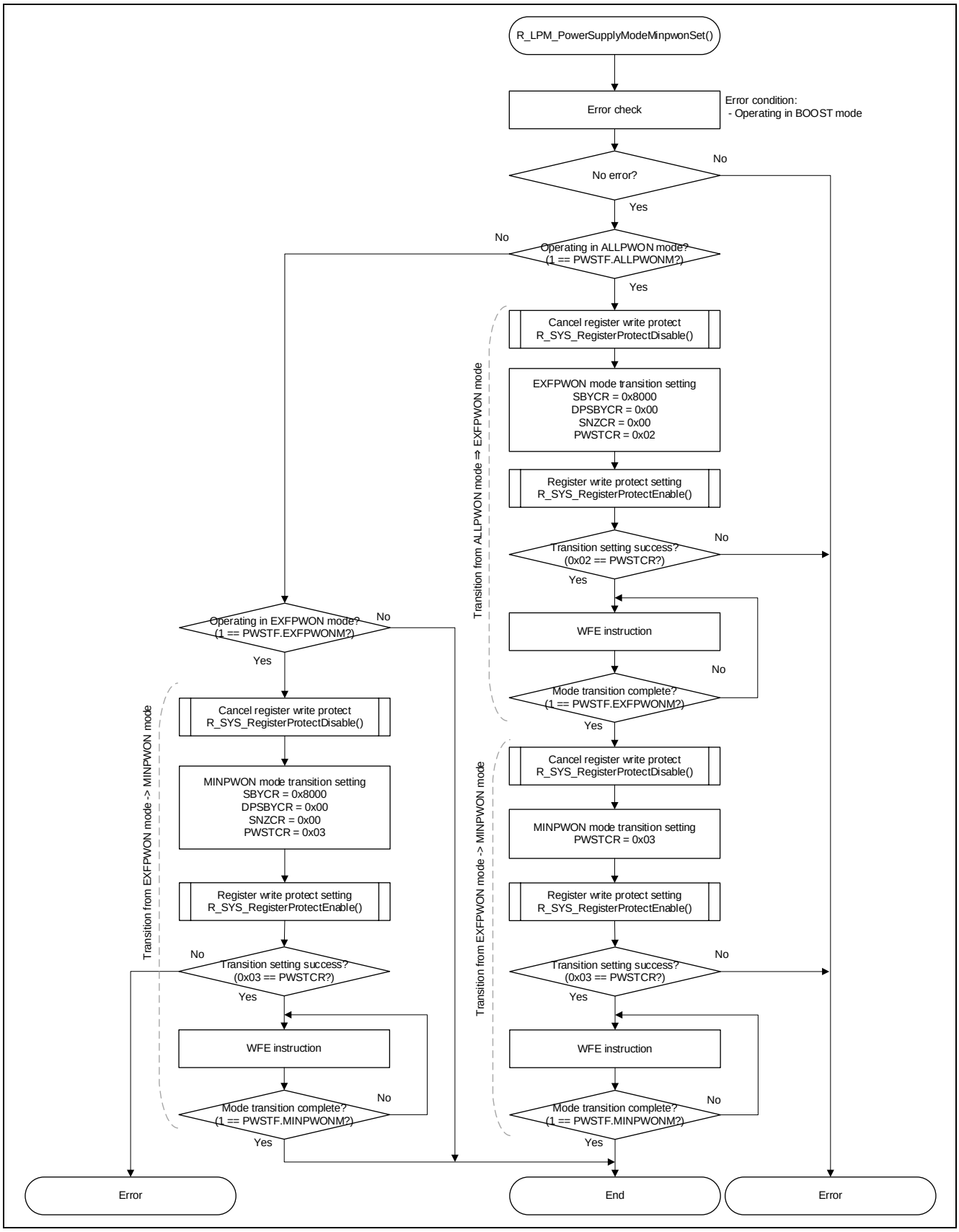

Figure 4-32 R_LPM_PowerSupplyModeMinpwonSet Function Processing Flow

(2) RE01 256KB Group

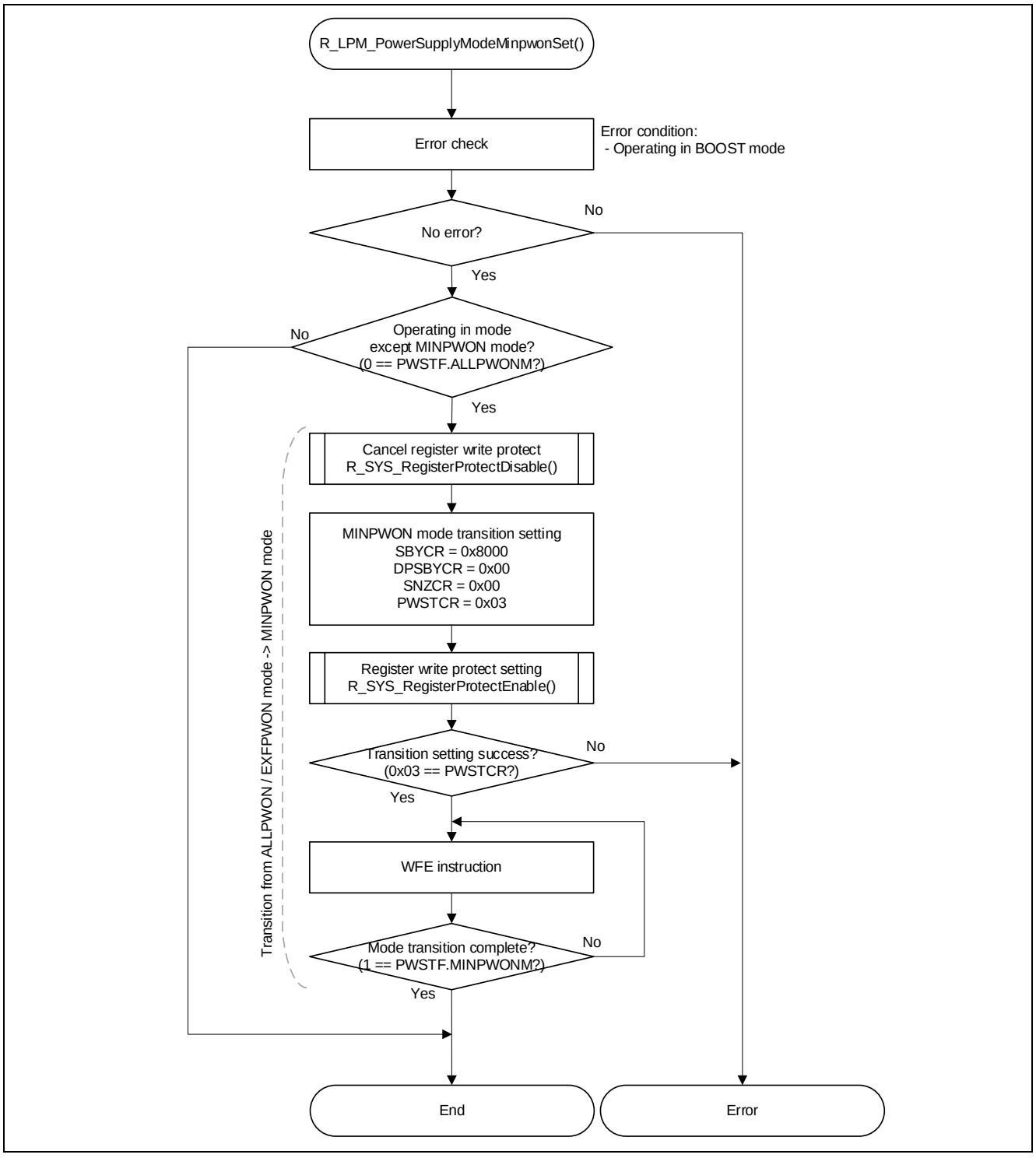

Figure 4-33 R_LPM_PowerSupplyModeMinpwonSet Function Processing Flow

4.3.16 R_LPM_PowerSupplyModeGet Function

Table 4-21 Specification of the R_LPM_PowerSupplyModeGet Function

Outline	e_lpm_power_supply_mode_t R_LPM_PowerSupplyModeGet(void)
Description	This function gets the state of power supply mode.
Input	None
Return value	LPM_POWER_SUPPLY_MODE_ALLPWON : Operating in ALLPWON mode
	LPM_POWER_SUPPLY_MODE_EXFPWON : Operating in EXFPWON mode
	LPM_POWER_SUPPLY_MODE_MINPWON : Operating in MINPWON mode
Remark	-

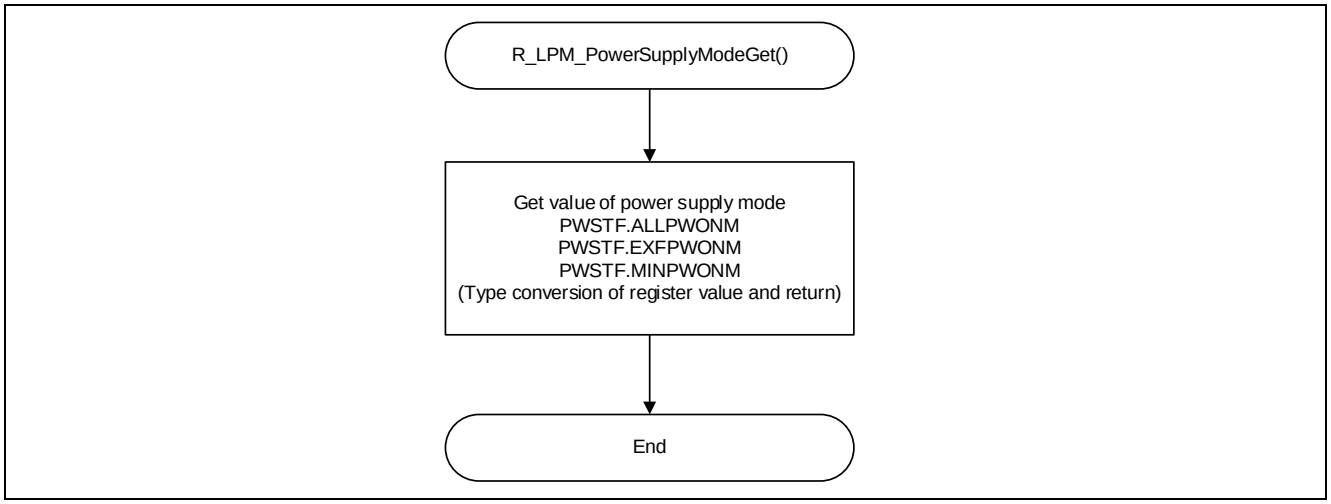

Figure 4-34 R_LPM_PowerSupplyModeGet Function Processing Flow

4.3.17 R_LPM_SnoozeSet Function

Table 4-22 Specification of the R_LPM_SnoozeSet Function

Outline	int32_t R_LPM_SnoozeSet(st_lpm_snooze_cfg_t * p_cfg)
Description	This function sets operating conditions in Snooze mode. *1
Input Note	st_lpm_snooze_cfg_t * p_cfg[in] Pointer of the structure storing the operating condition of Deep Software Standby mode.
Return value	Success (0) : Setting of operating condition in Snooze mode succeeded.
	Error (-1) : Setting of operating condition in Snooze mode failed. An error occurs if the following conditions are detected. — The input parameter "p_cfg" is NULL. — The input parameter "p_cfg->enable" exceeds the definition range of e_lpm_snz_enable_t. — The input parameter "p_cfg->req_rx0" exceeds the definition range of e_lpm_snz_rx0_falling_edge_t. — The input parameter "p_cfg->dtc_enable" exceeds the definition range of e_lpm_snz_dtc_enable_t. — The same event is set for the return factor from Snooze and transition factor from Snooze to Software Standby mode (For DTC and ADC).
Remark	-

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.6 Low Power Consumption Mode.

As reference information, the register that sets the SSTBY / Snooze / OPE mode transition factor is shown. In addition to the event set in the input parameter “p_cfg-> wup” of this function during snooze, when the event set in the input parameter “p_cfg-> wup” of the R_LPM_SSTBYModeSetup function is detected, status of CPU returns from snooze to OPE.

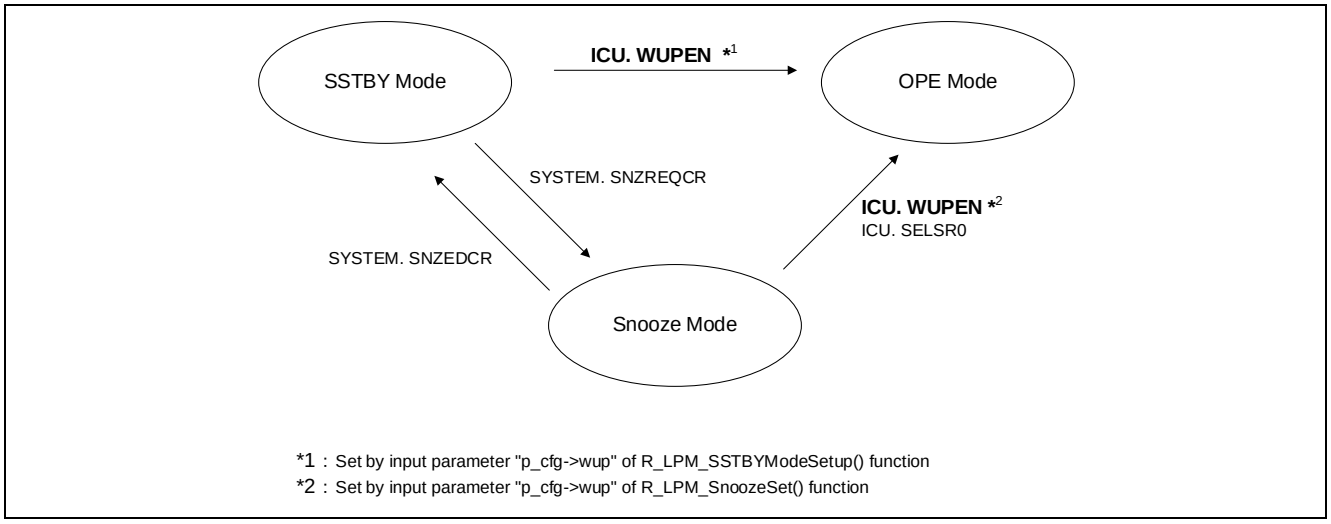

Figure 4-35 Register Which Sets SSTBY/Snooze/OPE Mode Transition Factor

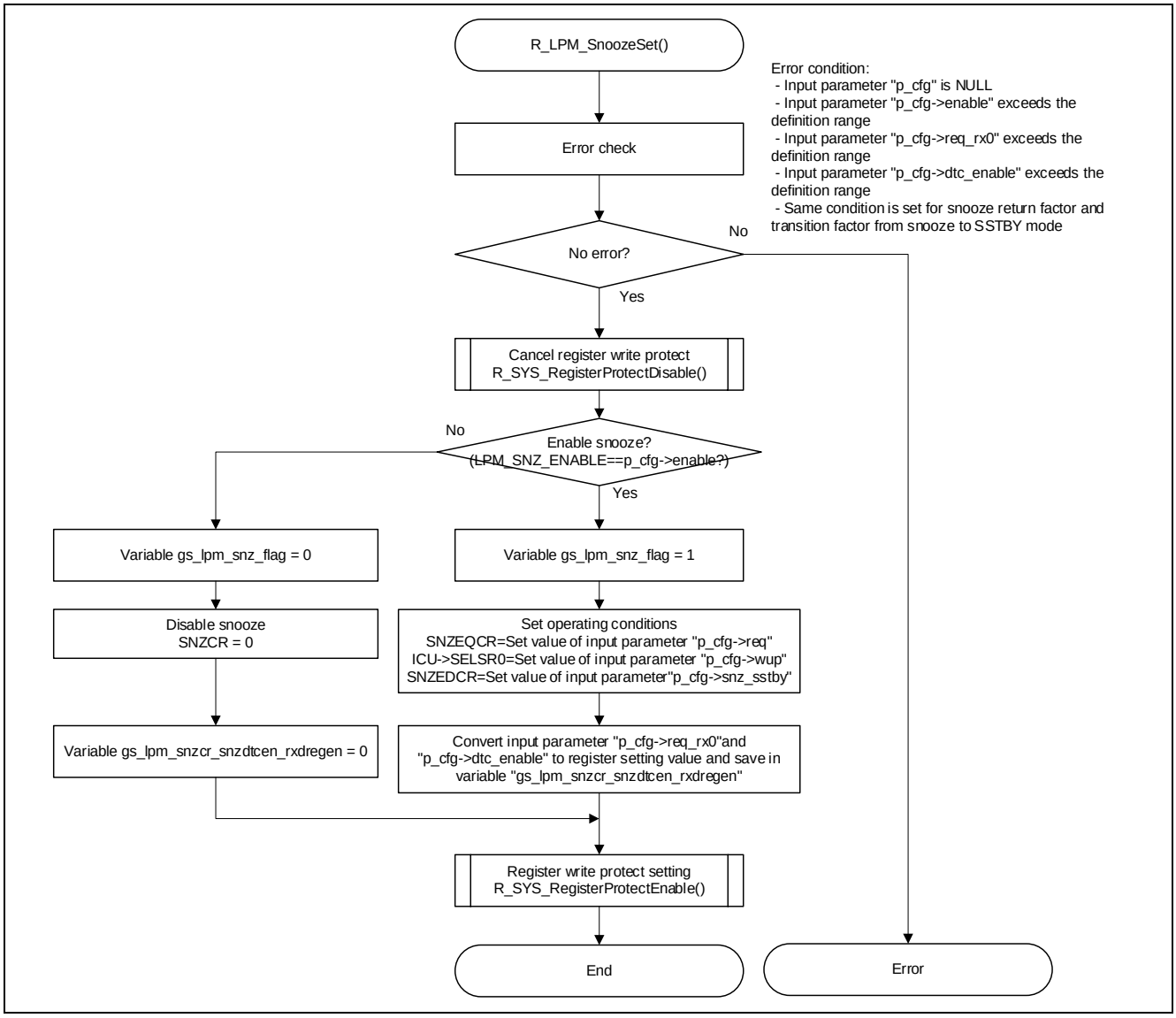

Figure 4-36 R_LPM_SnoozeSet Function Processing Flow

4.3.18 R_LPM_SleepModeEntry Function

Table 4-23 Specification of the R_LPM_SleepModeEntry Function

Outline	void R_LPM_SleepModeEntry(void)
Description	This function transitions to Sleep mode. *1 In this function, it transitions to Sleep mode. This function terminates when returning from Sleep mode.
Input	None
Return value	None
Remark	-

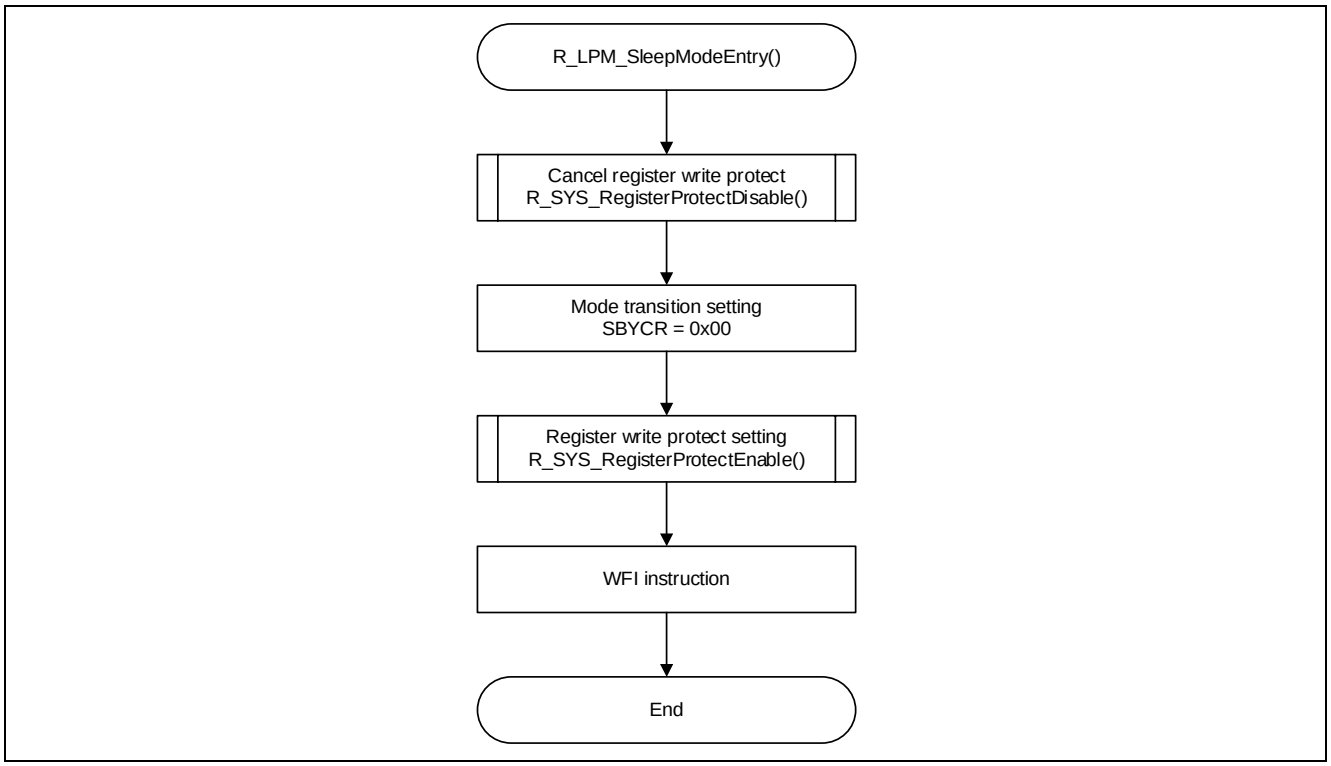

Figure 4-37 R_LPM_SleepModeEntry Function Processing Flow

4.3.19 R_LPM_SSTBYModeSetup Function

Table 4-24 Specification of the R_LPM_SSTBYModeSetup Function

Outline	int32_t R_LPM_SSTBYModeSetup(st_lpm_sstby_cfg_t * p_cfg)
Description	This function sets operating conditions in Software Standby mode. *1 <RE01 1500KB Gourp> Setting of power supply mode must be same both after transition to Software Standby mode and before transition to Operation from Software Standby mode.
Input Note	st_lpm_sstby_cfg_t * p_cfg[in] Set pointer of the structure storing the operating condition of Software Standby mode.
Return value	Success (0) : Setting of operating condition in Software Standby mode succeeded.
	Error (-1) : Setting of operating condition in Software Standby mode failed. An error occurs if the following conditions are detected. <RE01 1500KB Group> — The input parameter "p_cfg" is NULL. — The input parameter "p_cfg->ope_sstby" exceeds the definition range of e_lpm_ope_to_sstby_power_supply_mode_t. — The input parameter "p_cfg->sstby_ope" exceeds the definition range of e_lpm_sstby_to_ope_power_supply_mode_t. — The power supply mode after transition to Software Standby mode does not match the power supply mode before transition from Software Standby mode to operating mode. <RE01 256KB Group> — The input parameter "p_cfg" is NULL. — The input parameter "p_cfg->speed" exceeds the definition range of e_lpm_sstby_speed_mode_t. — The input parameter "p_cfg->power_supply" exceeds the definition range of e_lpm_sstby_power_supply_mode_t.
Remark	<RE01 256KB Group> For the CCC interval interrupt and the WUPT interval interrupt, the macro definition of the return factor from SSTBY mode set by the input parameter "p_cfg->wup" is defined with the same value.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.6 Low Power Consumption Mode.

(1) RE01 1500KB Group

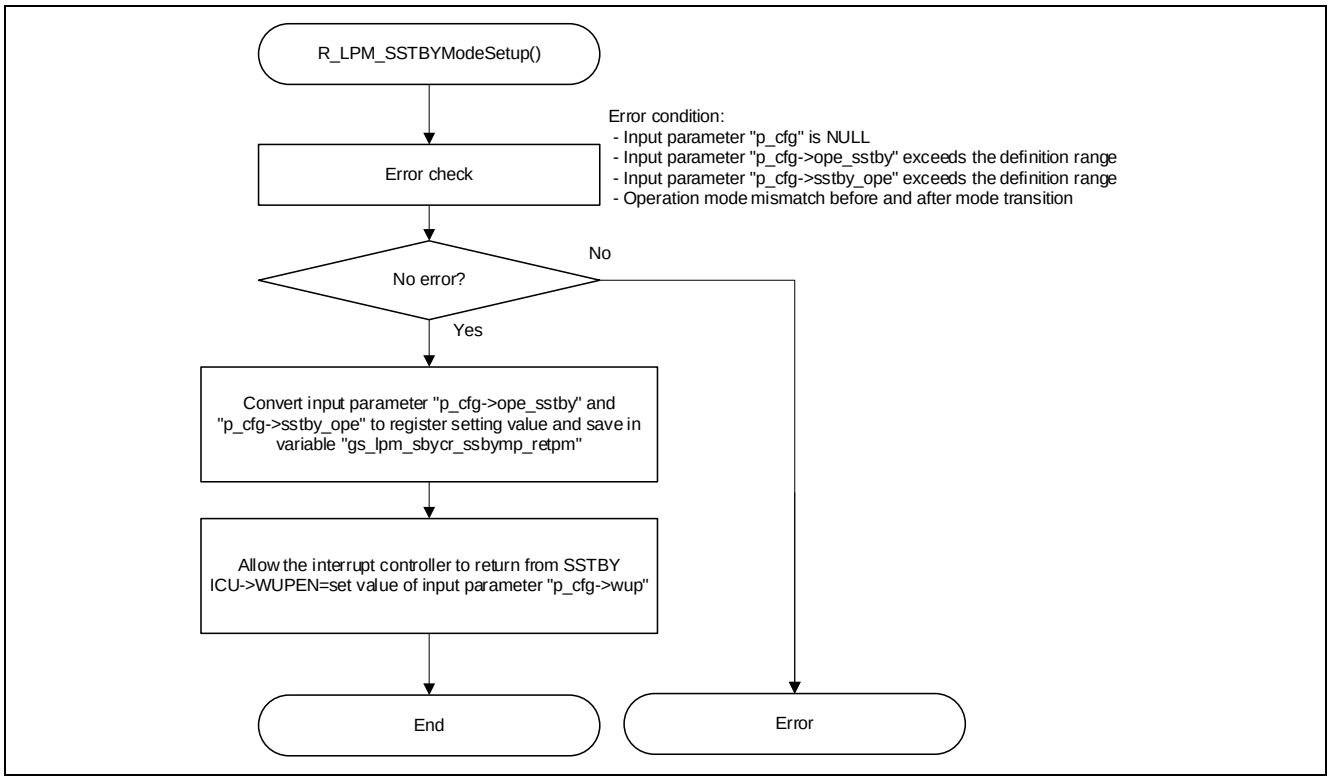

Figure 4-38 R_LPM_SSTBYModeSetup Function Processing Flow

(2) RE01 256KB Group

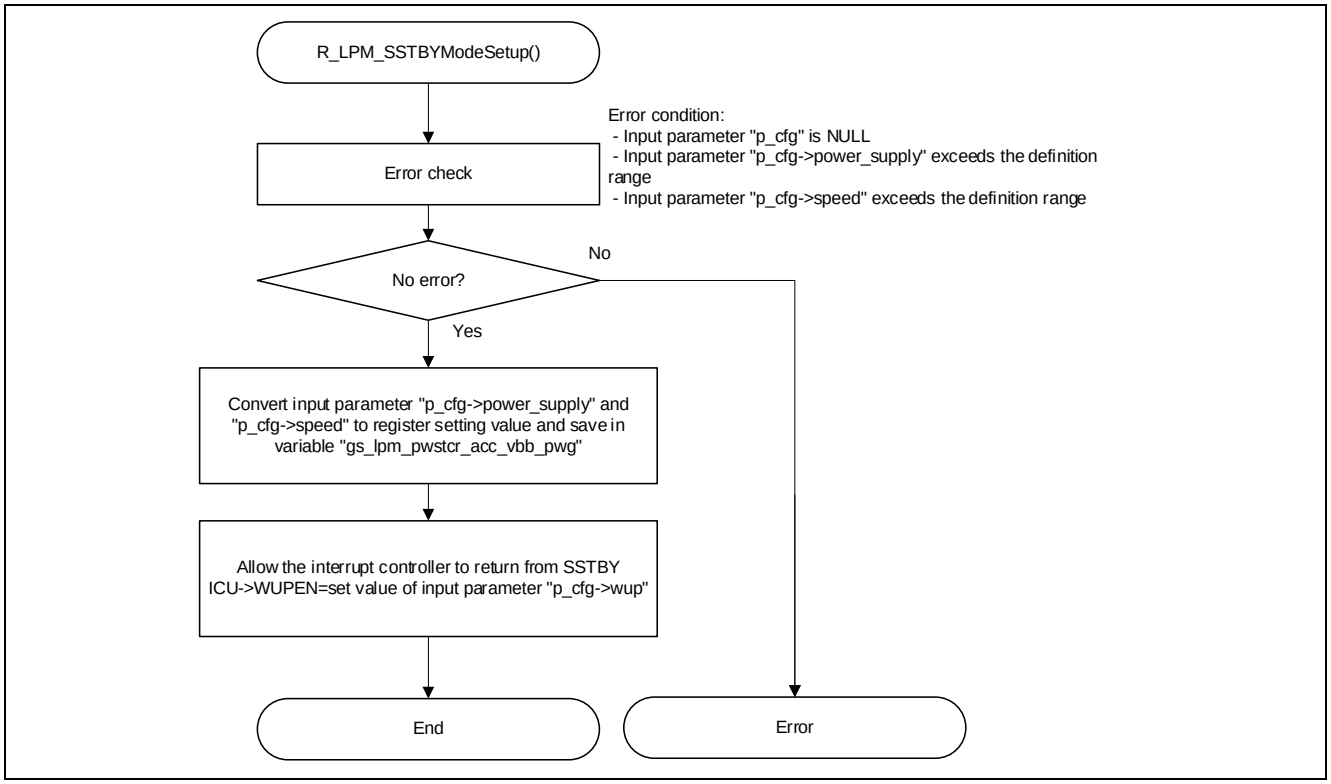

Figure 4-39 R_LPM_SSTBYModeSetup Function Processing Flow

4.3.20 R_LPM_SSTBYModeEntry Function

Table 4-25 Specification of the R_LPM_SSTBYModeEntry Function

Outline	int32_t R_LPM_SSTBYModeEntry(void)
Description	This function transitions to Software Standby mode. *1 In this function, it transitions to Software Standby mode. This function terminates when returning from Software Standby mode. Before this function is executed, it is necessary to execute the R_LPM_SSTBYModeSetup() function and set the operation conditions in Software Standby mode. <RE01 256KB Group> When executing this function in Back Bias mode, the mode transitions to another mode depending on the input parameter "p_cfg->power_supply" set by the R_LPM_SSTBYModeSetup() function as follows. — Transitions to EXFPWON VBB SSTBY mode, when "p_cfg->power_supply" is LPM_OPE_TO_SSTBY_EXFPWON or LPM_OPE_TO_SSTBY_EXFPWON_VBB. — Transitions to MINPWON VBB SSTBY mode, when "p_cfg->power_supply" is LPM_OPE_TO_SSTBY_MINPWON or LPM_OPE_TO_SSTBY_MINPWON_VBB.
	<RE01 256KB Group> This function executes the r_lpm_wait_release_reset_iso2 () function after transition from the MINPWON SSTBY mode to the EXFPWON Ope mode in order to wait for reset release of the ISO2 area stopped in the MINPWON mode. This process is required when the PCLKB division setting is 4 to 64 division. The r_lpm_wait_release_reset_iso2 () function is implemented as a WEAK function in the R_LPM driver. It can disable the corresponding function in R_LPM driver by implementing the non-weak function with the same name.
Input	None
Return value	Success (0) : Software Standby mode transition succeeded.
	Error (-1) : Software Standby mode transition failed. An error occurs if the following conditions are detected. <RE01 1500KB Group> — Operating in BOOST mode. — For the state of the current power supply mode, the state of the power supply mode before the transition to the Software Standby mode set by the R_LPM_SSTBYModeSetup () function is not appropriate. <RE01 256KB Group> — HOCO oscillation frequency is set to 48MHz or 64MHz — When transitioning to VBB SSTBY mode, setup of Back Bias control is incomplete. — For the state of the MINPWON mode, the state of the power supply mode in SSTBY mode set by the R_LPM_SSTBYModeSetup () function is not MINPWON mode. — When operating at ALLPWON, operating clock is over 4MHz.
Remark	<RE01 1500KB group> When using the device of WS1, there are the following restrictions. (Restriction No.34, 35, 36) If snooze is enabled in Subosc-Speed mode and Back Bias operation mode, LPM_SSTBY_TO_OPE_EXFPWON_TO_ALLPWON can not be set as the power supply mode when returning from Software Standby mode, so it returns an error.
	<RE01 256KB group> Because the PWSTCR resister writing operation in this function cannot be executed when HOCO is oscillating at 48MHz or 64MHz (HOCOMCR=02h or 03h), the return value will be "Error (-1)".

(1) RE01 1500KB Group

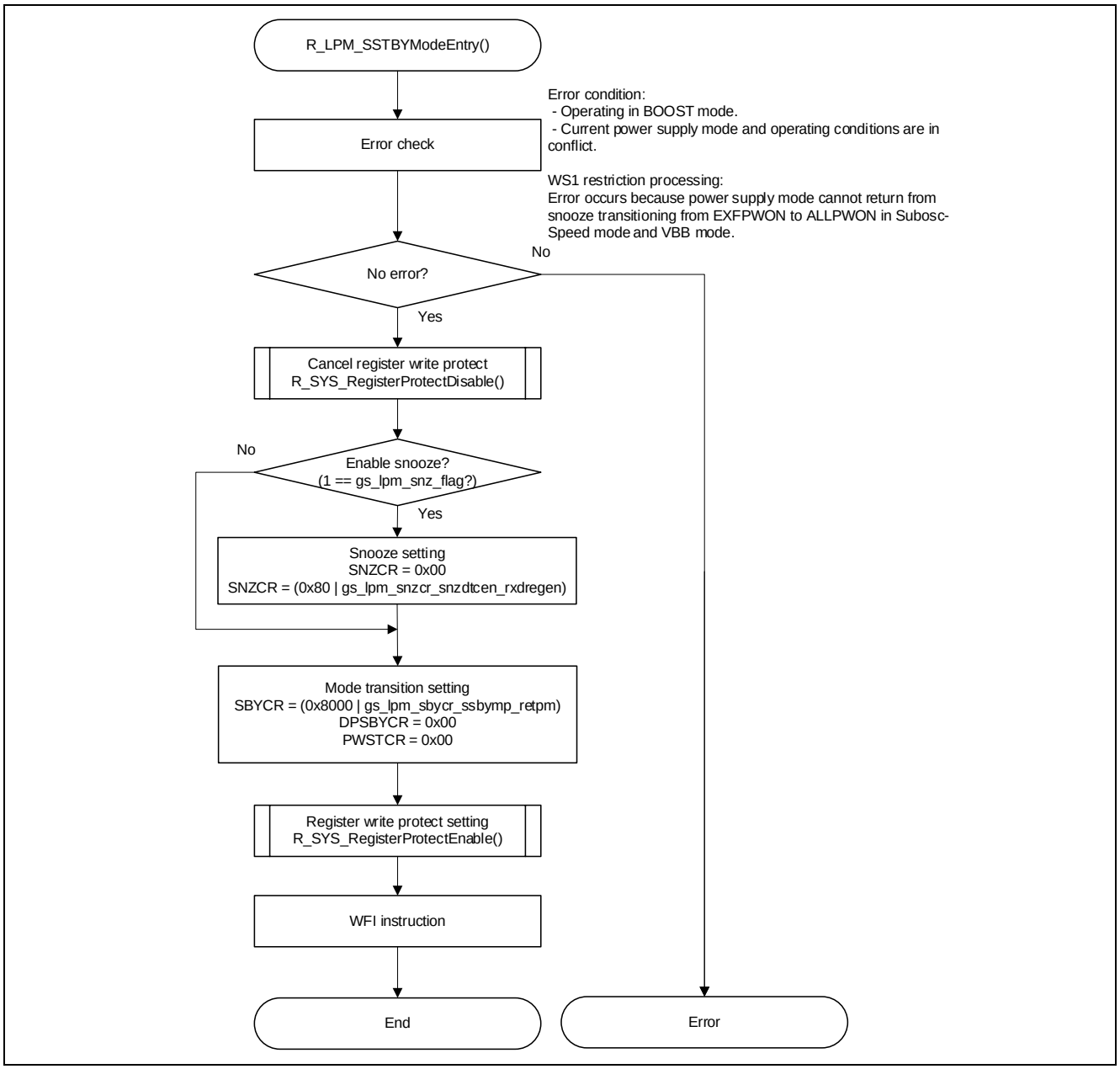

Figure 4-40 R_LPM_SSTBYModeEntry Function Processing Flow

(2) RE01 256KB Group

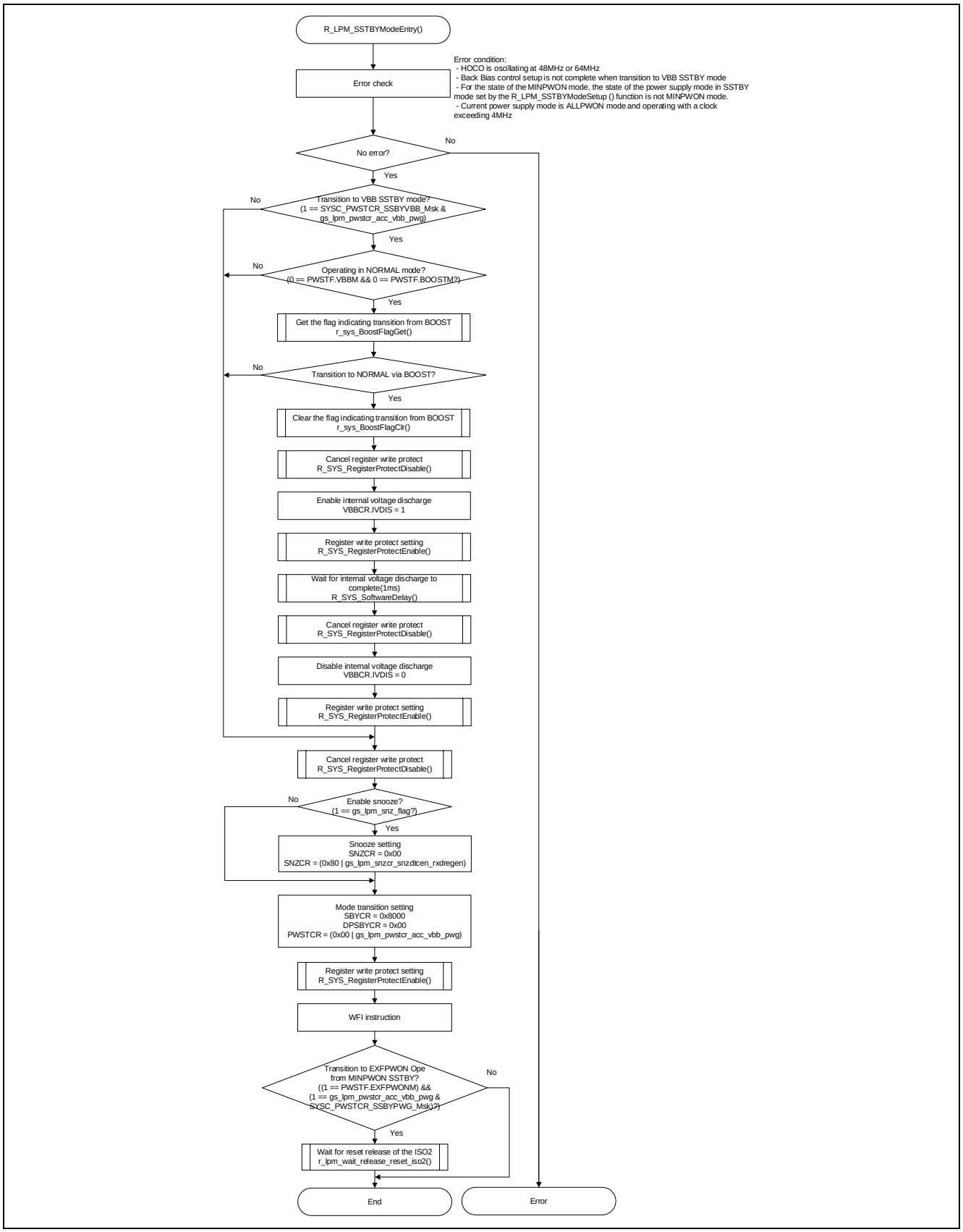

Figure 4-41 R_LPM_SSTBYModeEntry Function Processing Flow

4.3.21 R_LPM_DSTBYModeSetup Function

Table 4-26 Specification of the R_LPM_DSTBYModeSetup Function

Outline	int32_t R_LPM_DSTBYModeSetup(st_lpm_dstby_cfg_t * p_cfg)
Description	This function sets operating conditions in Deep Software Standby mode. *1
Input Note	st_lpm_dstby_cfg_t * p_cfg[in] Pointer of the structure storing the operating condition of Deep Software Standby mode.
Return value	Success (0) : Setting of operating condition in Deep Software Standby mode succeeded.
	Error (-1) : Setting of operating condition in Deep Software Standby mode failed. An error occurs if the following conditions are detected. — The input parameter "p_cfg" is NULL.
Remark	<RE01 256KB Group> For the CCC interval interrupt and the WUPT interval interrupt, the macro definition of the return factor from DSTBY mode set by the input parameter "p_cfg->wup" is defined with the same value.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.6 Low Power Consumption Mode.

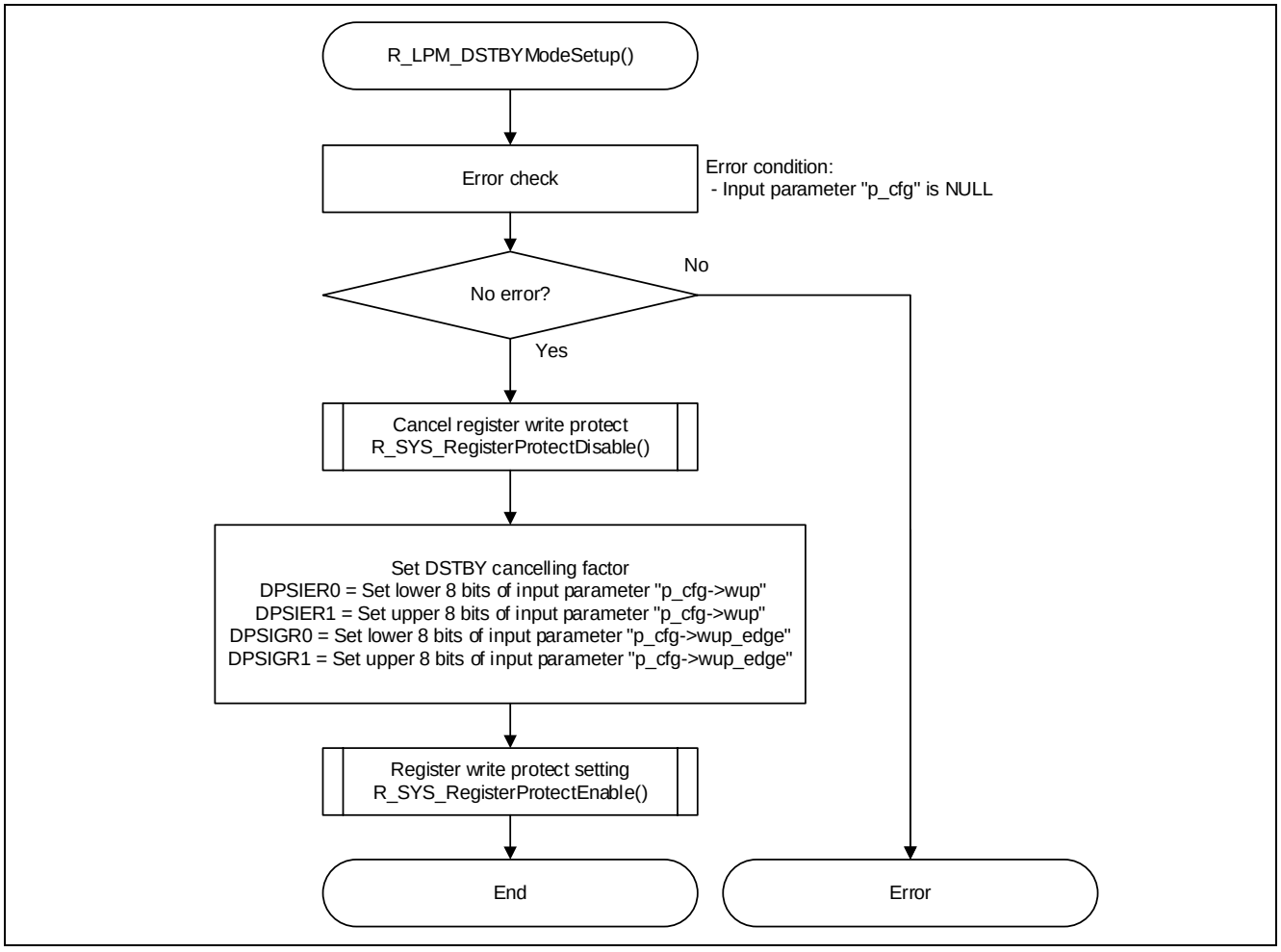

Figure 4-42 R_LPM_DSTBYModeSetup Function Processing Flow

4.3.22 R_LPM_DSTBYModeEntry Function

Table 4-27 Specification of the R_LPM_DSTBYModeEntry Function

Outline	int32_t R_LPM_DSTBYModeEntry(e_lpm_dstby_io_t io)
Description	This function transitions to Deep Software Standby mode. *1 In this function, it transitions to Deep Software Standby mode. Reset is canceled when recovered from Deep Software Standby mode. Before this function is executed, it is necessary to execute the R_LPM_DSTBYModeSetup() function and set the operation conditions in Deep Software Standby mode.
Input	e_lpm_dstby_io_t io[in] Set Pin States when Deep Software Standby Mode is canceled.
Return value	Error (-1) : Deep Software Standby mode transition failed. An error occurs if the following conditions are detected. <RE01 1500KB Group> — Operating in BOOST mode. — The input parameter "io" exceeds the definition range of e_lpm_dstby_io_t. <RE01 256KB Group> — The input parameter "io" exceeds the definition range of e_lpm_dstby_io_t.
Remark	The R_LPM_DSTBYResetStatusGet() function acquires the Deep Software Standby mode cancelling factor. Whether there is a need for initialization after rest of DPSIFR0 and DPSIFR1 which are indicator of cancelling factor is depending on the reset factor. This function clears DPSIFR0 and DPSIFR1 resisters before transitioning to Deep Software Standby mode.
	<RE01 1500KB Group> When using the device of WS1, there are the following restrictions. (Restriction No.51 and 52) If CCC is not set as a wakeup event from Deep Software Standby mode using the R_LPM_DSTBYModeSetup() function, or if the SYOCDCR.DBGEN bit is set to "1", return from Deep Software Standby mode is not possible. Then it returns an error.
	<RE01 1500KB Group> When using the device of WS1, there are the following restrictions. (Restriction No.1) In on-chip debugging state, it can not transit to Deep Software Standby mode.

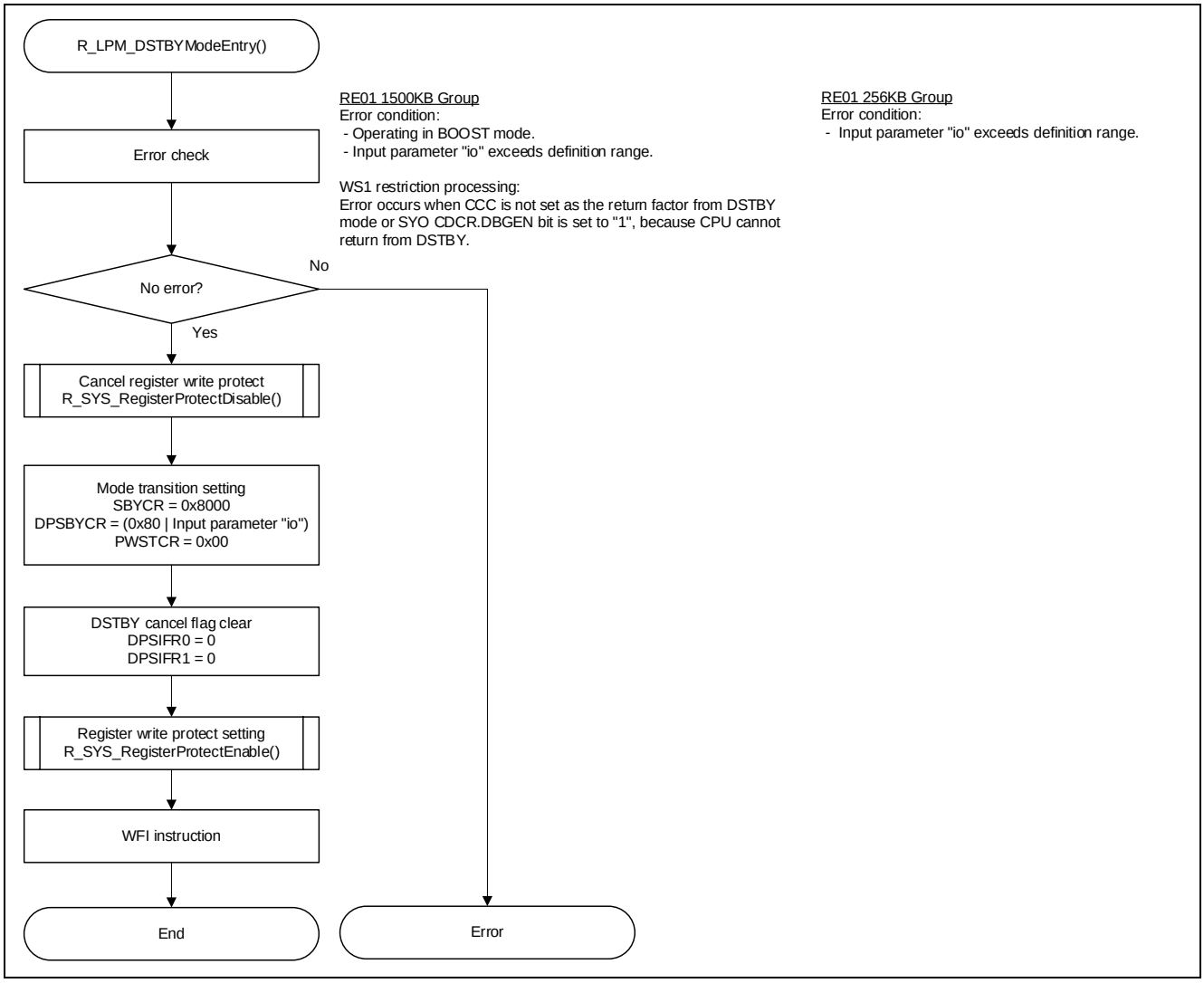

Figure 4-43 R_LPM_DSTBYModeEntry Function Processing Flow

4.3.23 R_LPM_DSTBYResetStatusGet Function

Table 4-28 Specification of the R_LPM_DSTBYResetStatusGet Function

Outline	uint32_t R_LPM_DSTBYResetStatusGet(void)
Description	This function acquires the Deep Software Standby mode cancelling factor.
Input	None
Return value Note	uint32_t : Return the value defined by LPM_DSTBY_RESET. When the following bit field of the return value is 1, return from DSTBY mode due to the relevant factor and the reset was released. <RE01 1500KB Group> — LPM_DSTBY_RESET_IRQ0DS : When the value of bit field is 1, Deep Software Standby was released by IRQ0-DS pin. — LPM_DSTBY_RESET_IRQ1DS : When the value of bit field is 1, Deep Software Standby was released by IRQ1-DS pin. — LPM_DSTBY_RESET_IRQ2DS : When the value of bit field is 1, Deep Software Standby was released by IRQ2-DS pin. — LPM_DSTBY_RESET_IRQ3DS : When the value of bit field is 1, Deep Software Standby was released by IRQ3-DS pin. — LPM_DSTBY_RESET_LVD1 : When the value of bit field is 1, Deep Software Standby was released by voltage monitor 1 signal (LVD1). — LPM_DSTBY_RESET_LVDBAT : When the value of bit field is 1, Deep Software Standby was released by voltage monitor BAT signal (LVDBAT). — LPM_DSTBY_RESET_NMI : When the value of bit field is 1, Deep Software Standby was released by NMI pin. — LPM_DSTBY_RESET_CCC : When the value of bit field is 1, Deep Software Standby was released by CCC interval interrupt signal. <RE01 256KB Group> — LPM_DSTBY_RESET_IRQ0DS : When the value of bit field is 1, Deep Software Standby was released by IRQ0-DS pin. — LPM_DSTBY_RESET_IRQ1DS : When the value of bit field is 1, Deep Software Standby was released by IRQ1-DS pin. — LPM_DSTBY_RESET_IRQ2DS : When the value of bit field is 1, Deep Software Standby was released by IRQ2-DS pin. — LPM_DSTBY_RESET_IRQ3DS : When the value of bit field is 1, Deep Software Standby was released by IRQ3-DS pin. — LPM_DSTBY_RESET_LVD1 : When the value of bit field is 1, Deep Software Standby was released by voltage monitor 1 signal (LVD1). — LPM_DSTBY_RESET_LVDBAT : When the value of bit field is 1, Deep Software Standby was released by voltage monitor BAT signal (LVDBAT). — LPM_DSTBY_RESET_RTCI : When the value of bit field is 1, Deep Software Standby was released by RTC interval interrupt signal. — LPM_DSTBY_RESET_RTCA : When the value of bit field is 1, Deep Software Standby was released by RTC alarm interrupt signal. — LPM_DSTBY_RESET_NMI : When the value of bit field is 1, Deep Software Standby was released by NMI pin. — LPM_DSTBY_RESET_CCC : When the value of bit field is 1, Deep Software Standby was released by CCC interval interrupt signal. — LPM_DSTBY_RESET_WUPT : When the value of bit field is 1, Deep Software Standby was released by WUPT interval interrupt signal.
Remark	<RE01 256KB Group> For the CCC interval interrupt and the WUPT interval interrupt, the macro definition of the return factor from DSTBY mode is defined with the same value.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group.

For details, please refer to 4.2.6 Low Power Consumption Mode.

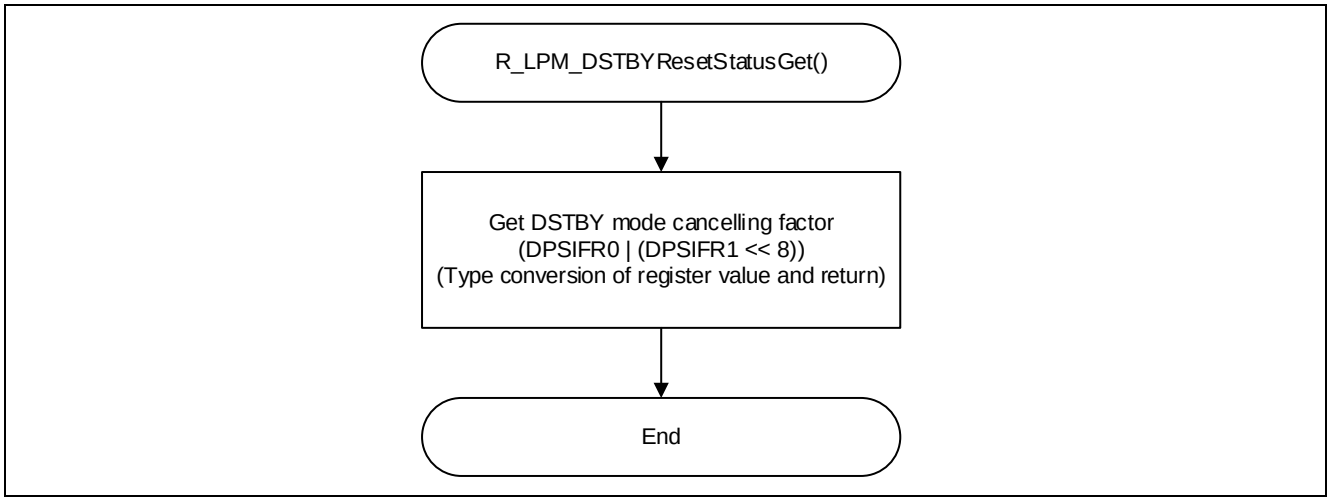

Figure 4-44 R_LPM_DSTBYResetStatusGet Function Processing Flow

4.3.24 R_LPM_RamRetentionSet Function

Table 4-29 Specification of the R_LPM_RamRetentionSet Function

Outline	void R_LPM_RamRetentionSet(uint32_t area)
Description	This function sets the RAM retention area in Software Standby mode. *1
Input Note	uint32_t area[in] : Defined by LPM_RAM_RETENTION_AREA. Set the RAM area to be retention during Software Standby mode to this parameter. Multiple areas can be set to input parameter by logical OR.
Return value	None
Remark	When this input parameter is used to set retention area, it sets corresponding RAMSn bit of RAMSDCR to 0 (retention). Note that this input parameter and the value of RAMSDCR register are inverted.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.3 RAM Cutoff.

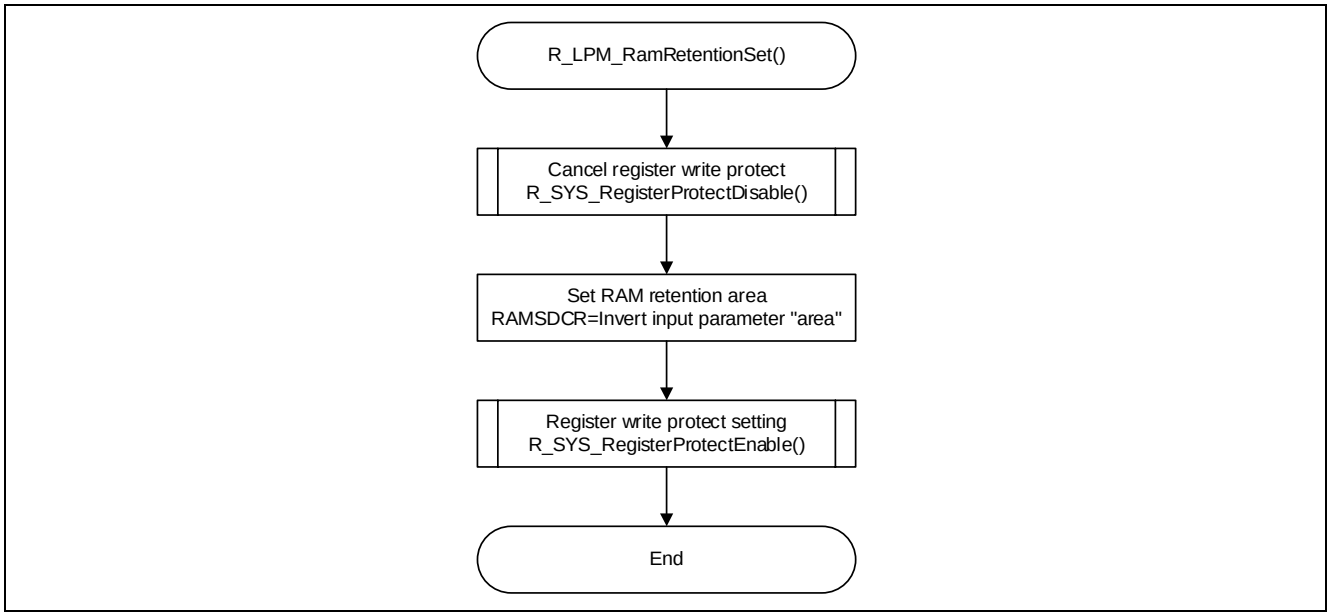

Figure 4-45 R_LPM_RamRetentionSet Function Processing Flow

4.3.25 R_LPM_IOPowerSupplyModeSet Function

Table 4-30 Specification of the R_LPM_IOPowerSupplyModeSet Function

Outline	int32_t R_LPM_IOPowerSupplyModeSet(uint8_t vocr)
Description	If power cannot be supplied in each IO power supply domain, an undefined signal is generated to activate the function to prevent propagation to other domains to which power is supplied. *1 Before this function is executed, it is necessary to execute the R_LPM_ModuleStop() function or the R_LPM_FastModuleStop() function to stop the module belonging to the specified IO power supply domain. <RE01 1500KB Group> AVCC0, AVCC1, IOVCC0, IOVCC1, IOVCC2, IOVCC3, USBVCC, MTDV <RE01 1500KB Group> AVCC0, IOVCC0, IOVCC1
Input Note	uint8_t vocr[in] : Defined by LPM_IOPOWER_SUPPLY. Set the IO power supply domains that are not supplied power to this parameter. Enable the propagation suppression function of generated undefined value in IO power supply domain set by “vocr” to other domains. Multiple IO power supply domains can be set to input parameter by logical OR.
Return value	Success (0) : Setting of the undefined signal propagation suppression function of IO power supply domain succeeded.
	Error (-1) : Setting of the undefined signal propagation suppression function of IO power supply domain failed. An error occurs if the following condition is detected. <RE01 1500KB Group> — A module belonging to the IO power supply domain specified by the input parameter "vocr" is not transitioned to the module stop state. <RE01 256KB Group> — No error. The Return value of this function is always "Success(0)".
Remark	<RE01 1500KB group> There is following restriction when using the WS1 device. (Restriction No.58) VOCR b7 bit definition is inverted.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.4 IO Power Supply Open Control.

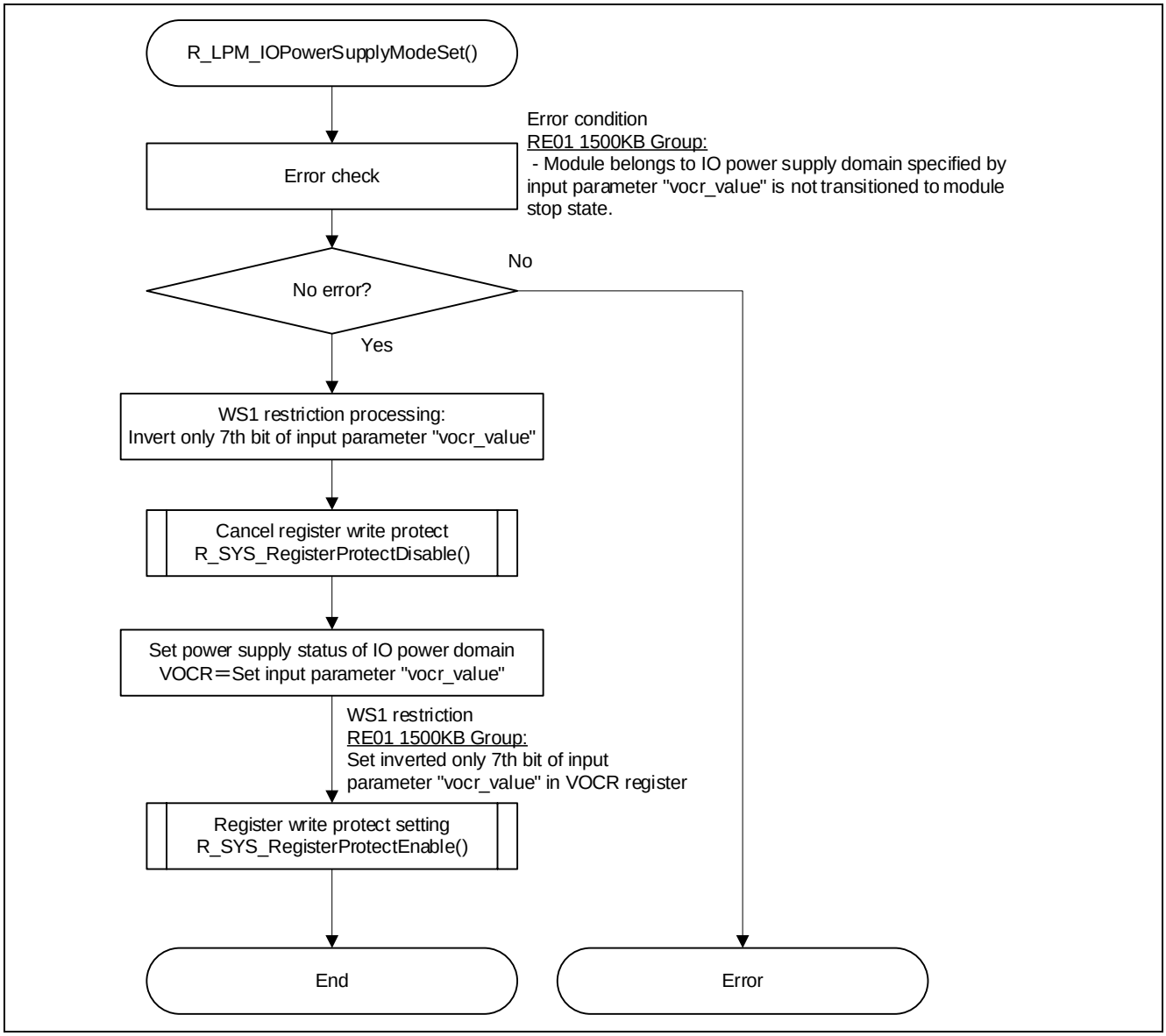

Figure 4-46 R_LPM_IOPowerSupplyModeSet Function Processing Flow

4.3.26 R_LPM_IOPowerSupplyModeGet Function

Table 4-31 Specification of the R_LPM_IOPowerSupplyModeGet Function

Outline	uint8_t R_LPM_IOPowerSupplyModeGet(void)
Description	This function acquires the IO power supply domains which undefined signal propagation suppression function is enabled.
Input	None
Return value Note	uint8_t : Return the value defined by LPM_IOPOWER_SUPPLY. When the bit field below the return value is '1', the undefined signal propagation suppression function of the corresponding IO power supply domain is enabled. <RE01 1500KB group> — LPM_IOPOWER_SUPPLY_AVCC0 : S14AD, TEMPS and VREF — LPM_IOPOWER_SUPPLY_AVCC1 : R12DA and ACMP — LPM_IOPOWER_SUPPLY_IOVCC0 : PORT8 — LPM_IOPOWER_SUPPLY_IOVCC1 : PORT3, PORT6, PORT7 and P202-P204 — LPM_IOPOWER_SUPPLY_IOVCC2 : PORT1 — LPM_IOPOWER_SUPPLY_IOVCC3 : P010-P015 and PORT5 — LPM_IOPOWER_SUPPLY_USBVCC : USB — LPM_IOPOWER_SUPPLY_MTDV : MTDV <RE01 256KB group> — LPM_IOPOWER_SUPPLY_AVCC0 : P000-P007 — LPM_IOPOWER_SUPPLY_IOVCC0 : PORT8 and P010-P015 — LPM_IOPOWER_SUPPLY_IOVCC1 : PORT1, PORT3, PORT5, PORT6, PORT7 and P202-P205
Remark	<RE01 1500KB Group> There is following restriction when using the WS1 device. (Restriction No.58) VOCR b7 bit definition is inverted.

Note: There are differences between the definitions of the RE01 1500KB group and the RE01 256KB group. For details, please refer to 4.2.4 IO Power Supply Open Control.

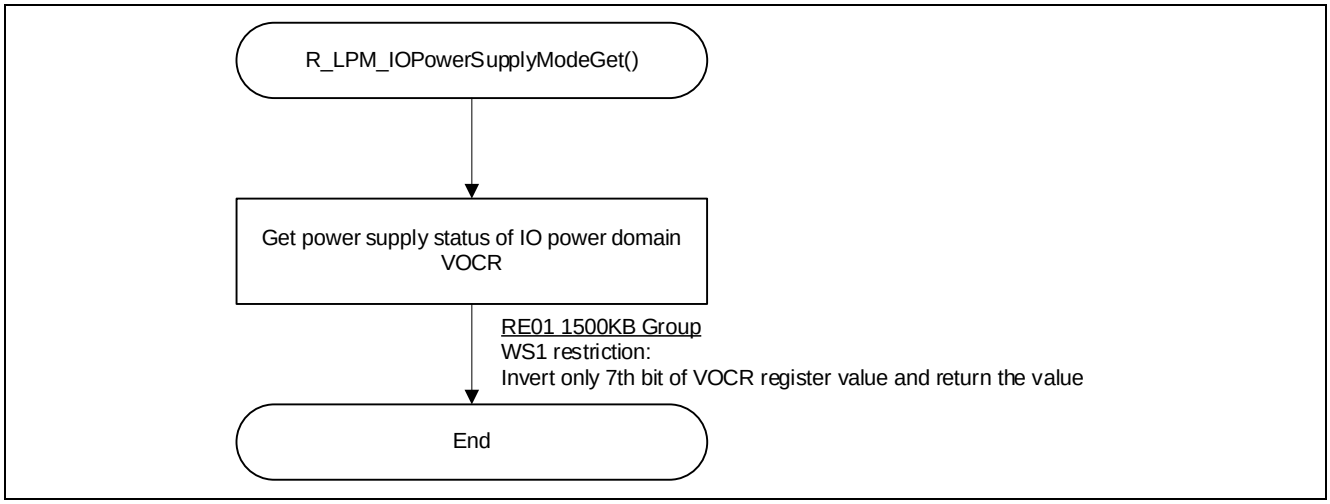

Figure 4-47 R_LPM_IOPowerSupplyModeGet Function Processing Flow

4.4 Interrupt Controller Settings

In addition to register corresponding to reducing power consumption function in chapter 12 of UMH, the R_LPM driver sets up the partial registers corresponding to interrupt controller unit in chapter 16 of UMH to set recovery events from software standby and snooze. The registers of the interrupt controller unit configured by the R_LPM driver are shown in Table 4-32.

Table 4-32 Interrupt Controller Register Settings

Register Name	Description
WUPEN	Set permission / prohibition of return from SSTBY mode for each interrupt
SELSR0	Set the snooze return event

5. Software Migration between Product Groups

This section describes the differences of interface specification of the driver function between the RE01 1500KB group and the RE01 256KB group, the precaution statement of software migration between product groups. Please refer to the Getting Started Guide regarding the example of mode transition.

5.1 The Differences of Interface Specification of the R_LPM Driver between the RE01 1500KB Group and the RE01 256KB Group

Table 5-1 shows the functions with different interface specification between the RE01 1500KB group and the RE01 256KB group.

For the detail of the macro / type definition of each group, please refer 4.2 Macro / Type Definition.

For the detail of error conditions, please refer 4.3 Specification of the Function.

When using the functions with no compatibility between groups, user software modification is mandatory.

Table 5-1 The Difference of Interface Specification of Each Function between Groups

No.	Function	The difference of macro / type definition by the difference of the hardware specification	The difference of error condition of the functions	No compatibility
1	R_LPM_GetVersion			
2	R_LPM_Initialize			
3	R_LPM_ModuleStart	✔ Note	✔	
4	R_LPM_ModuleStop	✔ Note		
5	R_LPM_FastModuleStart	✔ Note	✔	
6	R_LPM_FastModuleStop	✔ Note		
7	R_LPM_BackBiasModeEnable			
8	R_LPM_BackBiasModeDisable			
9	R_LPM_BackBiasModeEnableStatusGet			
10	R_LPM_BackBiasModeEntry		✔	
11	R_LPM_BackBiasModeExit			
12	R_LPM_BackBiasModeGet			
13	R_LPM_PowerSupplyModeAllpwonSet			
14	R_LPM_PowerSupplyModeExfpwonSet			
15	R_LPM_PowerSupplyModeMinpwonSet			
16	R_LPM_PowerSupplyModeGet			
17	R_LPM_SnoozeSet	✔		
18	R_LPM_SleepModeEntry			
19	R_LPM_SSTBYModeSetup	✔ Note	✔	✔
20	R_LPM_SSTBYModeEntry		✔	
21	R_LPM_DSTBYModeSetup	✔ Note		
22	R_LPM_DSTBYModeEntry		✔	
23	R_LPM_DSTBYResetStatusGet	✔ Note		
24	R_LPM_RamRetentionSet	✔ Note		
25	R_LPM_IOPowerSupplyModeSet	✔ Note	✔	
26	R_LPM_IOPowerSupplyModeGet	✔ Note		

Note: Not only of hardware module differences, but also function specification differences are the causes of interface changes.
e.g. : RTC function is always in "module stop release state" in the RE01 1500KB group, but it is "module stop state" after reset release in the RE01 256KB group.

5.2 The Precaution Statement of Software Migration between Product Groups

Regarding the functions shown in the Table 5-1 that have differences in I / F specifications, the precaution statement of software migration between the RE01 1500KB group and the RE01 256KB group, and the behavior when software migration is not done correctly are shown.

(1) R_LPM_ModuleStart, R_LPM_FastModuleStart：In case of using RTC, WDT or IWDT
R_LPM_ModuleStop, R_LPM_FastModuleStop

RE01 1500KB Group	RE01 256KB Group
After reset, these modules are operating. In case these modules are used as input parameter of R_LPM_ModuleStart or R_LPM_FastModuleStart, compilation error occurs.	After reset, these modules stop. R_LPM_ModuleStart or R_LPM_FastModuleStart must be set to operate these modules.

(2) R_LPM_BackBiasModeEntry：In case of transitioning from BOOST mode to VBB mode

RE01 1500KB Group	RE01 256KB Group
R_LPM_BackBiasModeEntry returns an error, because the direct transition from BOOST mode to VBB mode is not possible. It is necessary to execute R_LPM_BackBiasModeEntry after transition to NORMAL Subosc-Speed mode by using R_SYSTEM driver function.	The direct transition from BOOST mode to VBB mode is possible.

(3) R_LPM_SnoozeSet：In case of setting AGT0, AGT1 as a trigger for canceling snooze mode

RE01 1500KB Group	RE01 256KB Group
AGT0CA and AGT1CA are set.	Not only AGT0CA and AGT1CA but also AGTW0CA and AGTW1CA are set.

(4) R_LPM_SSTBYModeSetup：There are difference in the structure used in SSTBY setting

RE01 1500KB Group	RE01 256KB Group
Specify the power-supply mode of transition source, transition destination and return destination and the return factors from standby mode with argument. (ope_sstby, sstby_ope, wup) Compilation error occurs in case of using the structure of 256KB Group. (Because the member names are different between 1500KB Group and 256KB Group)	No need to specify because it is same power-supply mode of return destination and transition source. Specify the power-supply mode, VBB or not, Fast transition or not, of transition destination, and the return factors from standby mode with argument. (ope_sstby, speed, wup) Compilation error occurs in case of using the structure of 1500KB Group. (Because the member names are different between 1500KB Group and 256KB Group)

(5) R_LPM_SSTBYModeEntry：In case of transition from BOOST mode to SSTBY mode

RE01 1500KB Group	RE01 256KB Group
R_LPM_SSTBYModeEntry returns an error, because the direct transition from BOOST mode to SSTBY mode is not possible. It is necessary to execute R_LPM_SSTBYModeEntry after transition to NORMAL mode by using R_SYSTEM driver function.	The direct transition from BOOST mode to SSTBY mode is possible.

(6) R_LPM_DSTBYModeSetup, R_LPM_DSTBYResetStatusGet：
In case of setting RTCI and RTCA as trigger for canceling DSTBY mode

RE01 1500KB Group	RE01 256KB Group
Compilation error occurs because RTCI, RTCA are not support.	RTCI, RTCA are specified.

In case of setting CCC as a trigger for canceling DSTBY mode

RE01 1500KB Group	RE01 256KB Group
CCC periodic interrupt is specified.	CCC periodic interrupt and WUPT interrupt are specified.

(7) R_LPM_DSTBYModeEntry：In case of transition from BOOST mode to DSTBY mode

RE01 1500KB Group	RE01 256KB Group
R_LPM_DSTBYModeEntry returns an error, because the direct transition from BOOST mode to DSTBY mode is not possible. It is necessary to execute R_LPM_DSTBYModeEntry after transition to NORMAL mode by using R_SYSTEM driver function.	The direct transition from BOOST mode to DSTBY mode is possible.

(8) R_LPM_RamRetentionSet：
In case of setting AREA4 to AREA7 as the SRAM area for cutting off power supply.

RE01 1500KB Group	RE01 256KB Group
The set area is specified as the SRAM area whose power is to be cut off.	Compilation error occurs because outside the setting area.

(9) R_LPM_IOPowerSupplyModeSet, R_LPM_IOPowerSupplyModeGet :
There is a difference in the IO power supply area (some compile errors occur)

Terminal, Module	Power-domain	
	RE01 1500KB Group	RE01 256KB Group
S14AD (P000-P007) TEMPS, VREF	AVCC0	AVCC0
R12DA, ACMP	AVCC1	-
P010-P015	IOVCC3	IOVCC0
PORT1	IOVCC2	IOVCC1
P202-P204, P205	IOVCC1	IOVCC1
PORT3	IOVCC1	IOVCC1
PORT5	IOVCC3	IOVCC1
PORT6, PORT7	IOVCC1	IOVCC1
PORT8	IOVCC0	IOVCC0
USB	USBVCC	-
MTDV	VPM	-

Revision History

Rev.	Date	Description	
		Page	Summary
1.2	May. 10. 2019	-	First edition issued.
0.72	June. 26. 2019	-	Updated to support R_LPM driver version 0.72.
		-	Renamed title and file. Change document version number to R_LPM driver version.
		4,29	Added internal function call of R_SYSTEM driver with support of internal voltage discharge function.
		12	Update version number.
		13	Add DIL to module stop function.
		29,30	Added internal voltage discharge processing to R_LPM_BackBiasModeEntry() function.
1.00	July. 31. 2019	-	Updated to support R_LPM driver version 1.00
		-	Renamed document title and file to correspond to the official series name. Change series and group names in the document.
		-	Correct the typo.
		3	Added Startup Guide to Development Using CMSIS Package to list of related document.
		5	Changed the file configuration to correspond to the official series name.
		34,35	Added waiting for reset release of ISO2 area to R_LPM_PowerSupplyModeAllpwonSet function.
		36,37	Added waiting for reset release of ISO2 area to R_LPM_PowerSupplyModeExfpwonSet function.
	September. 19. 2019	1,5	Changed the package name to RE01 1500KB Group CMSIS Driver Package.
		-	Replace the note on the last page with the latest version.
		12	Change the number used in the version definition example to include the major and minor versions.
1.01	January.31.2020	ALL	Added R_LPM driver specification of RE01 256KB group.
		65	Added 5. Software Migration between Product Groups.
1.02	March.04.2020	3	Table 1-1 Changed abbreviation of 1500KB group UMH to “1500KB Group UMH”. Added 256KB UMH.
			Table1-2 Changed related document of 256KB group. Deleted table note. Changed title of Startup Guide to the latest version with 256KB group added.
		53	Changed description of error condition of HOCO frequency.
		63	Deleted “ADC140, TEMPS and VREF” from LPM_IOPOWER_SUPPLY_AVCC0 of RE01 256KB group with the change of 256KB Group UMH (Rev.0.80).
1.03	May.27.2020	26-28, 30, 49, 52, 57, 60, 62-63, 65	Table4-8 to Table4-11, Table4-22, Table4-24, Table4-26, Table4-28 to Table4-31 Added the note to refer to 4.2 Macro / Type Definition, because the argument definition is different between 1500KB group and 256KB group.

Rev.	Date	Description	
		Page	Summary
		26, 28, 35, 38, 40, 45, 52, 54, 57, 58, 60, 63, 65	Improved description about differences between product groups. Table4-8, Table4-10, Table4-15, Table4-16, Table4-18, Table4-20, Table4-24 to Table4-28, Table4-30, Table4-31
		60	Added description about return value of the function. Table4-28
		68-70	Added the following function specification differences, and overall changed the description. 5.2 R_LPM_SnoozeSet R_LPM_DSTBYModeSet R_LPM_DSTBYModeSet R_LPM_SSTBYModeSetup R_LPM_SSTBYModeEntry
		-	Deleted “5.2.1 The Example of Mode Transition”. Because to describe in the Getting Started Guide.

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.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

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

(Rev.4.0-1 November 2017)

RENESAS

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

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

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

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

Renesas Electronics (China) Co., Ltd.
Room 101-T01, Floor 1, Building 7, Yard No. 7, 8th Street, Shangdi, Haidian District, Beijing 100085, China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

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

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

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

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

Renesas Electronics Malaysia Sdn.Bhd.
Unit No 3A-1 Level 3A Tower 8 UOA Business Park, No 1 Jalan Pengaturcara U1/51A, Seksyen U1, 40150 Shah Alam, Selangor, Malaysia
Tel: +60-3-5022-1288, Fax: +60-3-5022-1290

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 8.0

© 2019 Renesas Electronics Corporation. All rights reserved.