R01AN4838JJ0103
Rev.1.03
2020.05.27

RE01 1500KB、256KB グループ
CMSIS ドライバ R_LPM 仕様書

要旨

本書では、RE01 1500KB、256KB グループ CMSIS Driver Package の低消費ドライバ R_LPM の仕様を説明します。

対象デバイス

RE01 1500KB グループ

RE01 256KB グループ

目次

1. 概要

CMSIS Driver Package は 1500KB グループ、256KB グループ毎に提供されます。本書では、特に記述のない限り、RE01 1500KB グループと RE01 256KB グループの R_LPM ドライバは同じ仕様です。

本書における略語一覧を表 1-1 に、関連文書一覧を表 1-2 に示します。

表 1-1 略語一覧

名称	略語
RENESAS-DRIVER R_LPM	R_LPM ドライバ
RENESAS-DRIVER R_SYSTEM	R_SYSTEM ドライバ
RE01 1500KB グループ ユーザーズマニュアル ハードウェア編	1500KB グループ UMH
RE01 256KB グループ ユーザーズマニュアル ハードウェア編	256KB グループ UMH
スヌーズ	Snooze
ソフトウェアスタンバイ	SSTBY
ディープソフトウェアスタンバイ	DSTBY
低リーク電流モード	BackBias モード、VBB
動作モード（スタンバイではない状態）	OPE

表 1-2 関連文書一覧

文書名	文書番号
R7F0E01 グループ (1.5M バイトフラッシュメモリ搭載製品)ユーザーズマニュアル ハードウェア編	R01UH0796
R7F0E01 グループ (256KB フラッシュメモリ搭載製品)ユーザーズマニュアル ハードウェア編	R01UH0894
RE01 1500KB,256KB グループ CMSIS パッケージを用いた開発スタートアップガイド	R01AN4660

2. ソフトウェアコンポーネントの内部構成

R_LPM ドライバは、消費電力低減の各機能に対し図 2-1 に示す関数で構成されます。

R_LPM ドライバはユーザが設定可能なコンフィギュレーションに従い、R_SYSTEM ドライバの関数を呼び出します。R_SYSTEM ドライバの関数の呼び出し条件および関数名を図 2-1 に示します。

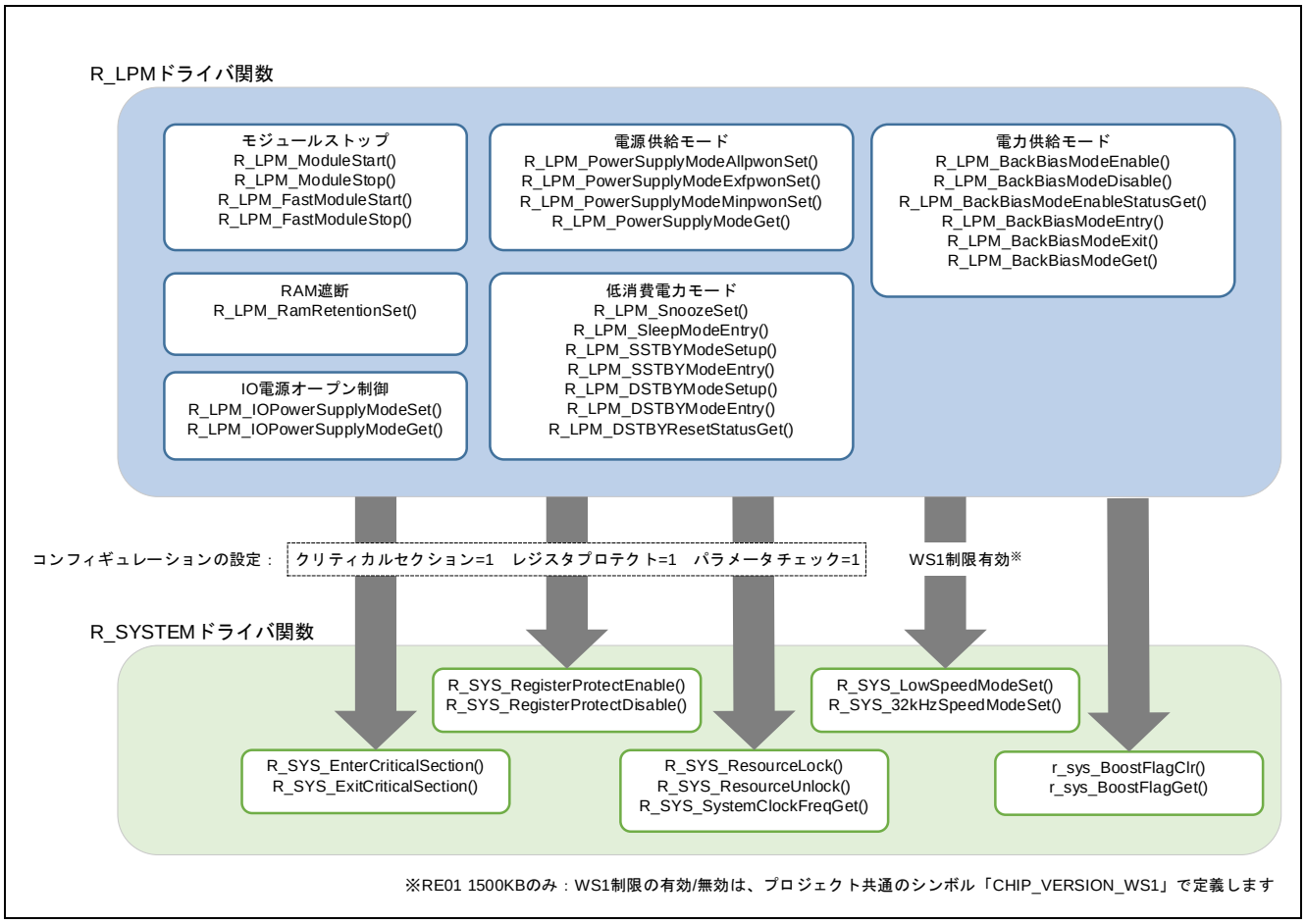

図 2-1 R_LPM ドライバ関数と R_SYSTEM ドライバ関数の関係

2.1　ファイル構成

R_LPM ドライバは CMSIS Driver Package の Device HAL に該当し、ベンダ独自ファイル格納ディレクトリ内の r_lpm_api.c、r_lpm_api.h、r_lpm_cfg.h の 3 個のファイルで構成されます。各ファイルの役割を表 2-1 に示します。RE01 1500KB グループ CMSIS Driver Package における R_LPM ドライバのファイル構成を図 2-2 に青線で示します。ハードウェア差異に伴い、256KB グループ CMSIS Driver Package の構成要素は異なりますが、図 2-2 の青枠で示す R_LPM ドライバのファイル構成は 1500KB グループと同じです。

表 2-1 R_LPM ドライバ 各ファイルの役割

ファイル名	内容
r_lpm_api.c	ドライバソースファイルです ドライバ関数の実態を用意します R_LPM ドライバを使用する場合は、本ファイルをビルドする必要があります
r_lpm_api.h	ドライバヘッダファイルです ユーザが参照可能なマクロ／型／プロトタイプ宣言を用意します R_LPM ドライバを使用する場合は、本ファイルをインクルードする必要があります
r_lpm_cfg.h	コンフィギュレーション定義ファイルです ユーザが設定可能なコンフィギュレーション定義を用意します

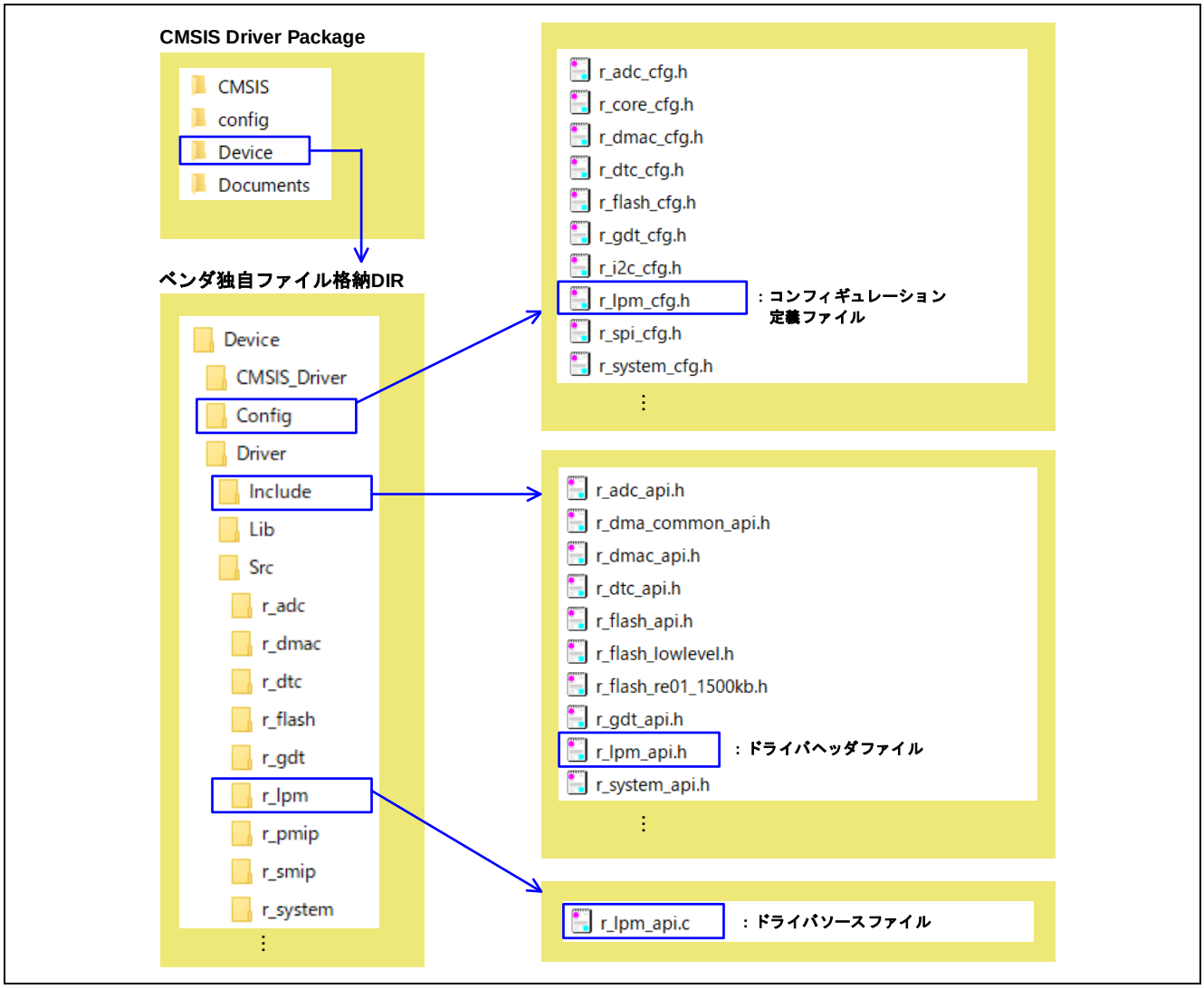

図 2-2 CMSIS Driver Package における R_LPM ドライバファイル構成

3. ソフトウェアコンポーネントの内部動作

R_LPM ドライバは消費電力低減機能を実現します。本章では、モード遷移により消費電力を低減する機能の R_LPM ドライバ関数呼び出し手順を示します。R_LPM ドライバ以外の処理は手順フローには記載せず、R_LPM ドライバ関数実行前の条件や、実行頻度、関数の終了条件などはコメントで記載しています。

3.1 電力制御モード

BackBias モードへ遷移する手順を図 3-1 に示します。

R_LPM_BackBiasModeEntry()関数を使用して BackBias モードへ遷移する場合は、R_LPM_BackBiasModeEnable()関数を使用して BackBias 制御のセットアップを開始し、R_LPM_BackBiasModeEnableStatusGet()関数を使用して BackBias 制御のセットアップの完了を確認する必要があります。R_LPM ドライバで唯一順守する必要のある手順です。

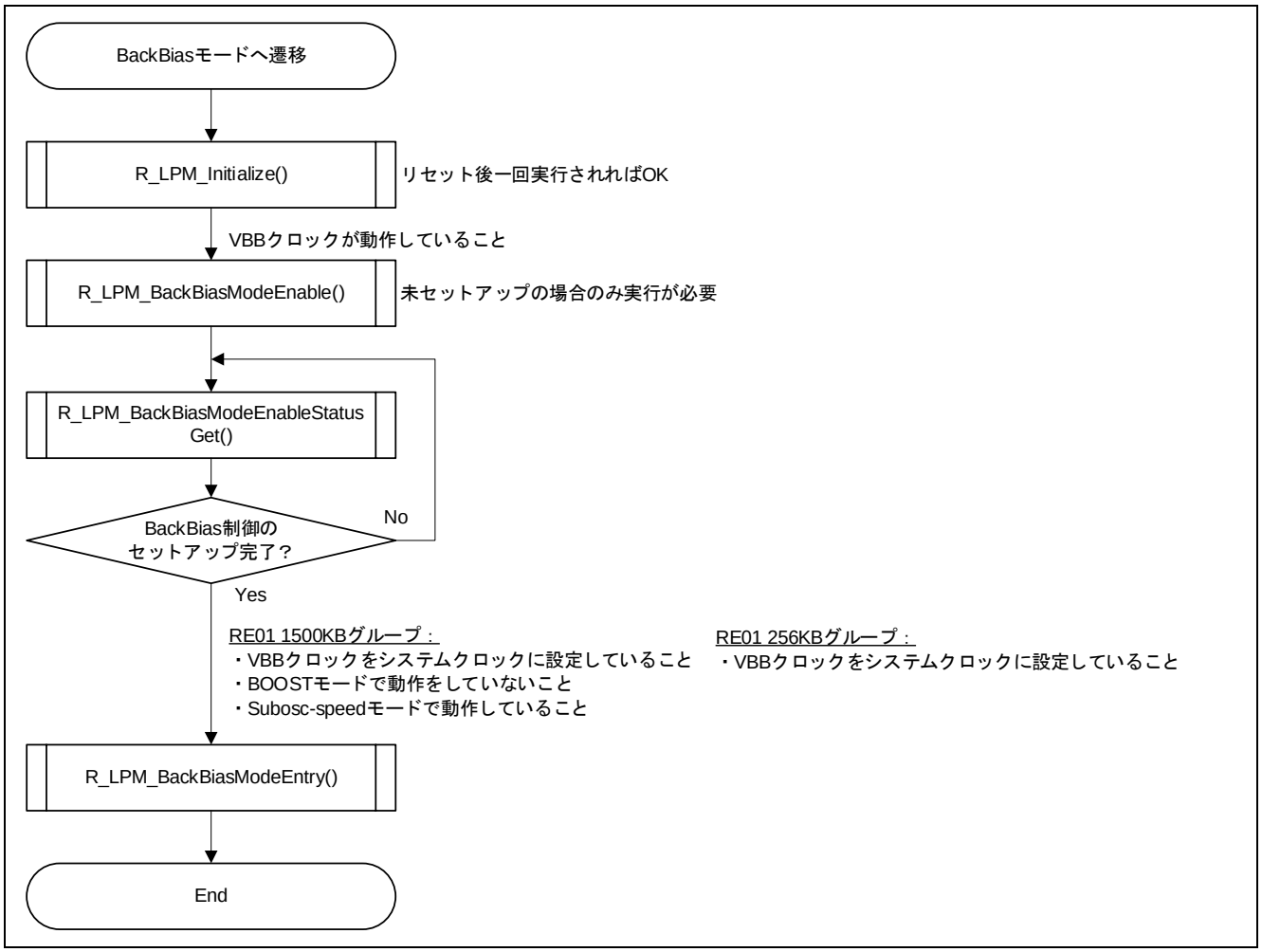

図 3-1 BackBias モードへの遷移手順

3.2　電源供給モード

各電源供給モードへの遷移手順を図 3-2 に示します。

RE01 1500KB ハードウェアは、ALLPWON モードと MINPWON モード間の直接遷移をサポートしていませんが、R_LPM ドライバは関数内で 2 段階の遷移を行うことで、これらのモード遷移をサポートします。ユーザは動作中のモードを意識せずに、遷移先のモードを指定することができます。

RE01 256KB ハードウェアは、ALLPWON モードと MINPWON モード間の直接遷移をサポートします。RE01 1500KB の R_LPM ドライバと同様に、遷移先のモードを指定することが可能です。

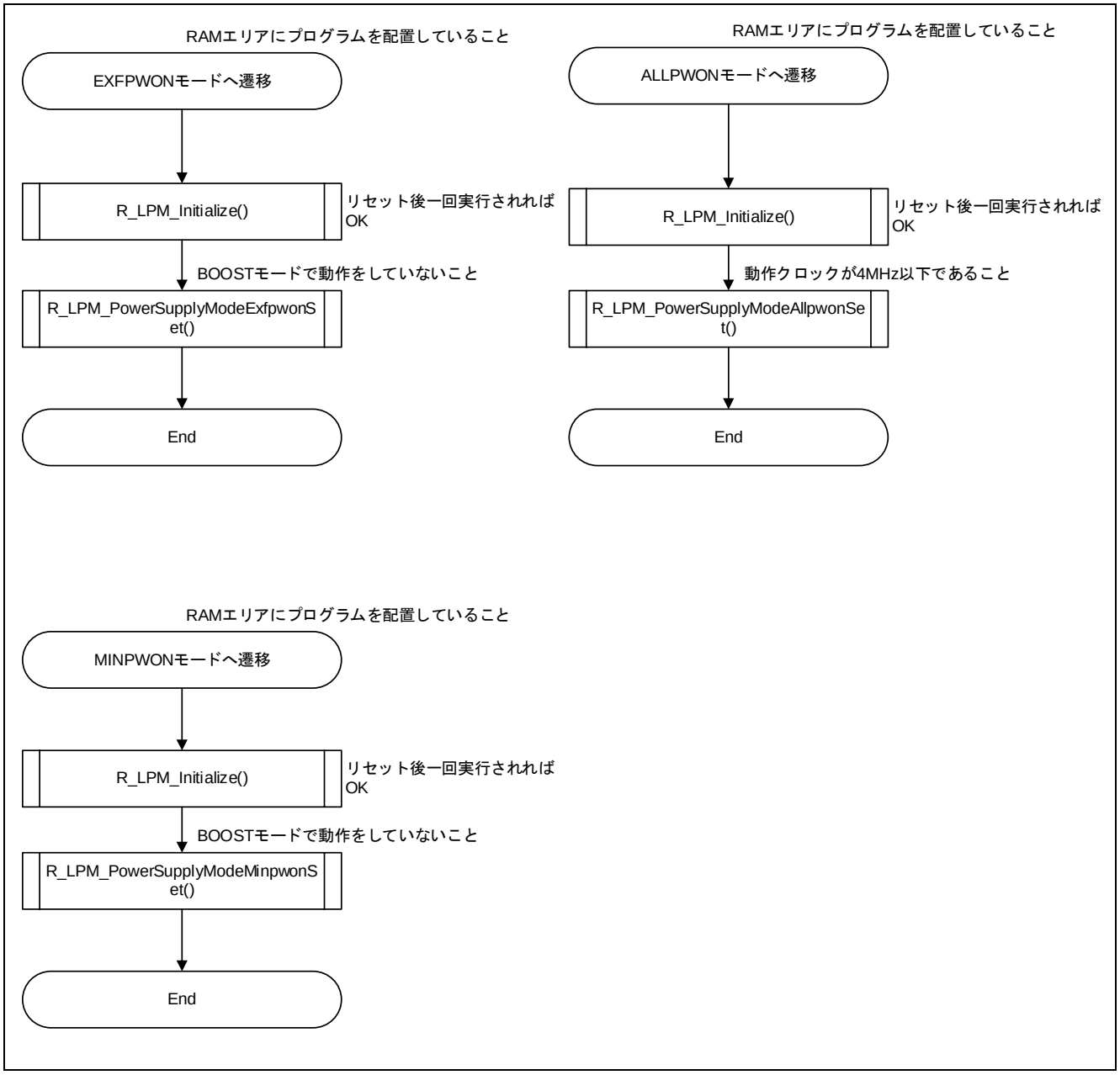

図 3-2 各電源供給モードへの遷移手順

3.3　低消費電力モード

各低消費電力モードへの遷移手順を図 3-3 に示します。

SSTBY モードおよび DSTBY モードへ遷移する前に、R_LPM_SSTBYModeSetup()関数および R_LPM_DSTBYModeSetup()関数を使用して低消費電力モードからの復帰要因などの動作条件を設定します。動作条件を設定しなかった場合は、ハードウェアの初期値相当の条件でモード遷移を行います。

各 Entry 関数は、関数内で低消費電力モードに遷移するため一旦 CPU が停止します。低消費電力モードからの復帰要因を検出すると CPU が再開し、各 Entry 関数が終了します。ただし、R_LPM_DSTBYModeEntry()関数は、復帰要因を検出するとリセットが解除されるため関数は終了せずに、リセットベクタにジャンプします。

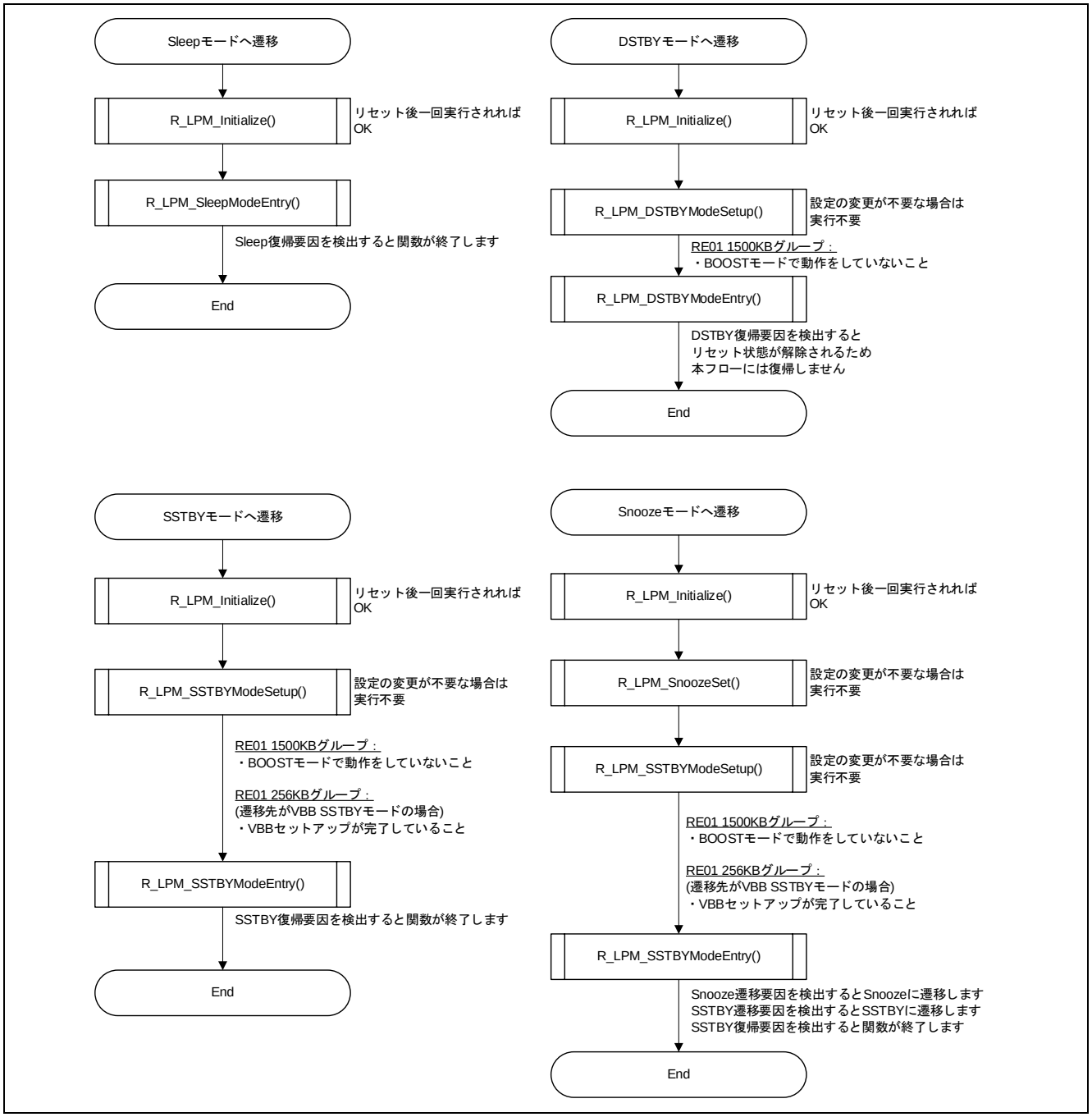

図 3-3 各低消費電力モードへの遷移手順

4. ソフトウェアユニット詳細情報

4.1 コンフィングレーション

R_LPM ドライバは、ユーザが設定可能なコンフィギュレーションを r_lpm_cfg.h ファイルに用意します。

4.1.1 パラメータチェック

R_LPM ドライバにおけるパラメータチェックの有効/無効を設定します。

名称：LPM_CFG_PARAM_CHECKING_ENABLE

表 4-1 LPM_CFG_PARAM_CHECKING_ENBLE の設定

設定値	内容
0	パラメータチェックを無効にします 関数仕様に記載しているエラー条件の検出を行いません ただし「ｘｘｘをハードウェアへ設定できない」に該当するエラーは、本コンフィギュレーションの設定に関わらず常に検出します
1 （初期値）	パラメータチェックを有効にします 関数仕様に記載しているエラー条件の検出を行います

4.1.2 クリティカルセクション

R_LPM ドライバにおけるクリティカルセクション制御の有効/無効を設定します。

レジスタの一部ビットを変更するために、レジスタ値を読み出し後一部変更して再設定を行うような場合は、途中で割り込みが入らないようクリティカルセクションの制御を行う必要があります。

名称：LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE

表 4-2 LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE の設定

設定値	内容
0	クリティカルセクションの制御を無効にします
1 （初期値）	下記 R_SYSTEM ドライバ関数を使用したクリティカルセクションの制御を有効にします R_SYS_EnterCriticalSection()関数 R_SYS_ExitCriticalSection()関数

4.1.3　レジスタライトプロテクション

R_LPM ドライバにおけるレジスタライトプロテクション制御の有効/無効を設定します。

消費電力低減機能はレジスタライトプロテクション対象レジスタを持ちます。対象レジスタに書き込みを行う場合は、レジスタライトプロテクションの制御を行う必要があります。

名称：LPM_CFG_REGISTER_PROTECTION_ENABLE

表 4-3 LPM_CFG_REGISTER_PROTECTION_ENABLE の設定

設定値	内容
0	レジスタライトプロテクションの制御を無効にします
1 （初期値）	下記 R_SYSTEM ドライバ関数を使用したレジスタライトプロテクションの制御を有効にします R_SYS_RegisterProtectEnable()関数 R_SYS_RegisterProtectDisable()関数

4.1.4 関数の RAM 配置

R_LPM ドライバの特定関数を RAM で実行するための設定を行います。

Flash を遮断中に実行するプログラムは、RAM に配置し RAM で実行する必要があります。

関数の RAM 配置を設定するコンフィギュレーションは、関数ごとに定義を持ちます。

名称：LPM_CFG_SECTION_xxx

xxx には関数名をすべて大文字で記載

例）　R_LPM_Initialize 関数　→　LPM_CFG_SECTION_R_LPM_INITIALIZE

表 4-4 LPM_CFG_SECTION_xxx の設定

設定値	内容
SYSTEM_SECTION_CODE	関数を RAM に配置しません
SYSTEM_SECTION_RAM_FUNC	関数を RAM に配置します

表 4-5 各関数の RAM 配置初期状態

番号	関数名	RAM 配置
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　マクロ／型定義

R_LPM ドライバは、ユーザが参照可能なマクロおよび型定義を r_lpm_api.h フィルに用意します。

高速化を考慮し、定義値はハードウェアレジスタに直接設定できる値とします。

低消費電力モードからの復帰要因設定のように、複数の定義を選択する必要がある用途に対しては、#define 定義を使用します。#define 定義された値をレジスタに設定する際は、設定対象レジスタの設定可能ビットを定義した LPM_xxx_ALL との論理積をレジスタに設定することで、定義されていない値がユーザから設定された場合に、予約ビット等に書き込まれないようにします。

4.2.1　バージョン取得

バージョン取得で使用する定義を図 4-1 に示します。

R_LPM_GetVresion 関数の戻り値の定義(Version 1.23 の場合)

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

図 4-1 バージョン取得で使用する定義

4.2.2　モジュールストップ

(1)　RE01 1500KB グループ

モジュールストップで使用する定義を図 4-2 に示します。

R_LPM_ModuleStart 関数の引数 module の型
R_LPM_ModuleStop 関数の引数 module の型

```
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;
```

R_LPM_FastModuleStart 関数の引数 module_a の型
R_LPM_FastModuleStop 関数の引数 module_a の型

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

R_LPM_FastModuleStart 関数の引数 module_b の型
R_LPM_FastModuleStop 関数の引数 module_b の型

```
#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)
```

R_LPM_FastModuleStart 関数の引数 module_c の型
R_LPM_FastModuleStop 関数の引数 module_c の型

```
#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)
```

R_LPM_FastModuleStart 関数の引数 module_d の型
R_LPM_FastModuleStop 関数の引数 module_d の型

```
#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          (0x00001100UL)
#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)
```

図 4-2 モジュールストップで使用する定義

(2) RE01 256KB グループ

モジュールストップで使用する定義を図 4-3 に示します。

R_LPM_ModuleStart 関数の引数 module の型
R_LPM_ModuleStop 関数の引数 module の型

```
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;
```

R_LPM_FastModuleStart 関数の引数 module_a の型
R_LPM_FastModuleStop 関数の引数 module_a の型

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

R_LPM_FastModuleStart 関数の引数 module_b の型
R_LPM_FastModuleStop 関数の引数 module_b の型

```
#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)
```

R_LPM_FastModuleStart 関数の引数 module_c の型
R_LPM_FastModuleStop 関数の引数 module_c の型

```
#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)
```

R_LPM_FastModuleStart 関数の引数 module_d の型
R_LPM_FastModuleStop 関数の引数 module_d の型

```
#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)
```

図 4-3 モジュールストップで使用する定義

4.2.3 RAM 遮断

(1) RE01 1500KB グループ

RAM 遮断で使用する定義を図 4-4 に示します。

R_LPM_RamRetentionSet 関数の引数 area の定義

```
#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)
```

図 4-4 RAM 遮断で使用する定義

(2) RE01 256KB グループ

RAM 遮断で使用する定義を図 4-5 に示します。

R_LPM_RamRetentionSet 関数の引数 area の定義

```
#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)
```

図 4-5 RAM 遮断で使用する定義

4.2.4　IO 電源オープン制御

(1)　RE01 1500KB グループ

IO 電源オープン制御で使用する定義を図 4-6 に示します。

R_LPM_IOPowerSupplyModeSet 関数の引数 vocr の定義
R_LPM_IOPowerSupplyModeGet 関数の戻り値の定義

```
#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)
```

図 4-6 IO 電源オープン制御で使用する定義

(2)　RE01 256KB グループ

IO 電源オープン制御で使用する定義を図 4-7 に示します。

R_LPM_IOPowerSupplyModeSet 関数の引数 vocr の定義
R_LPM_IOPowerSupplyModeGet 関数の戻り値の定義

```
#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)
```

図 4-7 IO 電源オープン制御で使用する定義

4.2.5　電源供給モード

電源供給モードで使用する定義を図 4-8 に示します。

R_LPM_PowerSupplyModeGet 関数の戻り値の型

```
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;
```

図 4-8 電源供給モードで使用する定義

4.2.6　低消費電力モード

(1)　RE01 1500KB グループ

スヌーズで使用する定義を図 4-9 に示します。

R_LPM_SnoozeSet 関数の引数 p_cfg の型

```
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_ADOMAT        (0x08UL)
#define LPM_SNZ_SSTBY_ADOUMT        (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;
```

図 4-9 スヌーズで使用する定義

SSTBY モードで使用する定義を図 4-10 に示します。

R_LPM_SSTBYModeSetup 関数の引数 p_cfg の型

```
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)
```

図 4-10 SSTBY モードで使用する定義

DSTBY モードで使用する定義を図 4-11 に示します。

R_LPM_DSTBYModeSetup 関数の引数 p_cfg の型

```
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)
```

R_LPM_DSTBYModeEntry 関数の引数 io の型

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

R_LPM_DSTBYResetStatusGet 関数の戻り値の定義

```
#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)
```

図 4-11 DSTBY モードで使用する定義

(2)　RE01 256KB グループ

スヌーズで使用する定義を図 4-12 に示します。

R_LPM_SnoozeSet 関数の引数 p_cfg の型

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

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;

#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)

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;

#define LPM_SNZ_SSTBY_AGT1UD        (0x01UL)
#define LPM_SNZ_SSTBY_DTCZ          (0x02UL)
#define LPM_SNZ_SSTBY_DTCN          (0x04UL)
#define LPM_SNZ_SSTBY_ADOMAT        (0x08UL)
#define LPM_SNZ_SSTBY_ADOUMT        (0x10UL)
#define LPM_SNZ_SSTBY_SCIO          (0x80UL)
#define LPM_SNZ_SSTBY_NONE          (0x00UL)
#define LPM_SNZ_SSTBY_ALL           (0x9FUL)

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_SCIO_AM         = 0x79,
    LPM_SNZ_WUP_SCIO_RXI_OR_ERI = 0x7A,
    LPM_SNZ_WUP_NONE            = 0x00
} e_lpm_snz_wup_t;
```

図 4-12 スヌーズで使用する定義

SSTBY モードで使用する定義を図 4-13 に示します。

R_LPM_SSTBYModeSetup 関数の引数 p_cfg の型

```
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)
```

図 4-13 SSTBY モードで使用する定義

DSTBY モードで使用する定義を図 4-14 に示します。

R_LPM_DSTBYModeSetup 関数の引数 p_cfg の型

```
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)
```

R_LPM_DSTBYModeEntry 関数の引数 io の型

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

R_LPM_DSTBYResetStatusGet 関数の戻り値の定義

```
#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)
```

図 4-14 DSTBY モードで使用する定義

4.2.7　電力制御モード

BackBias モードで使用する定義を図 4-15 に示します。

R_LPM_BackBiasModeEnable 関数の引数 clock の型

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

R_LPM_BackBiasModeEnableStatusGet 関数の戻り値の型

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

R_LPM_BackBiasModeGet 関数の戻り値の型

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

図 4-15 BackBias モードで使用する定義

4.3　関数仕様

R_LPM ドライバの各関数の仕様と処理フローを示します。

本章の関数仕様の表は、Doxygen に記載している内容に相当します。関数仕様の表の説明では、Doxygen と対比させるため低リーク電流モードを BackBias と記載します。

処理フローのエラーチェックは、エラー条件のみ列挙し具体的なチェック方法の記載は省略します。

処理フローの条件分岐には、条件判定に使用する対象を明確にするためレジスタ名および変数名を記載していますが、判定方法は処理フロー内の記述と必ずしも一致しません。

各関数仕様に共通の注釈を以下に示します。

※1. 本関数は、LPM_CFG_REGISTER_PROTECTION_ENABLE に"1"が設定された場合、R_SYSTEM ドライバの R_SYS_RegisterProtectEnable()関数と R_SYS_RegisterProtectDisable()関数を使用します。本関数実行前に、R_SYS_Initialize()関数を実行してください。

※2. 本関数は、LPM_CFG_ENTER_CRITICAL_SECTION_ENABLE に"1"が設定された場合、R_SYSTEM ドライバの R_SYS_EnterCriticalSection()関数と R_SYS_ExitCriticalSection()関数を使用します。本関数実行前に、R_SYS_Initialize()関数を実行してください。

4.3.1　R_LPM_GetVersion 関数

表 4-6 R_LPM_GetVersion 関数仕様

書式	uint32_t　R_LPM_GetVersion(void)
仕様説明	R_LPM ドライバのバージョンを取得します
引数	なし
戻り値	ドライバのバージョン
備考	ー

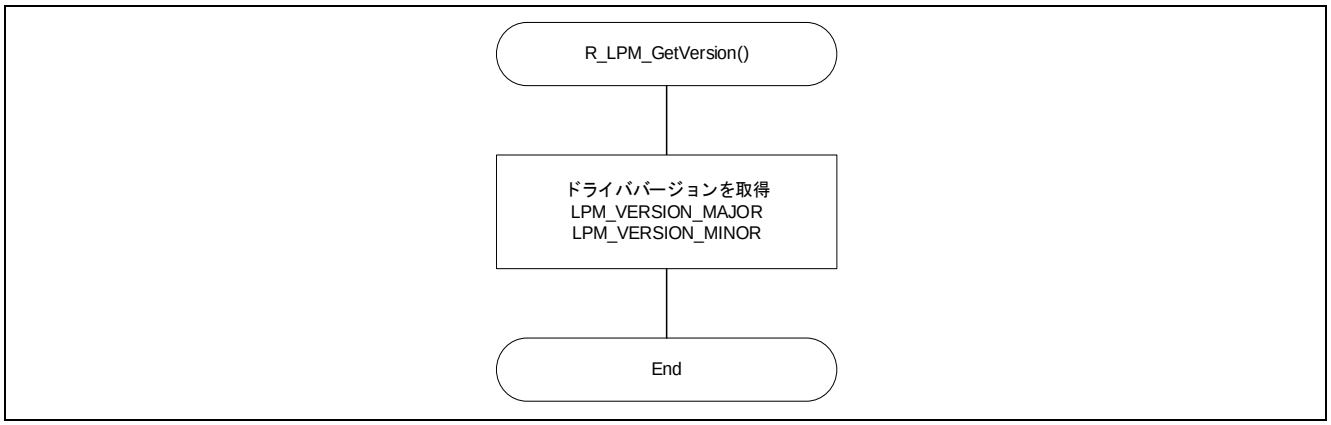

図 4-16 R_L**PM_GetVesion 関数処理フロー**

4.3.2 R_LPM_Initialize 関数

表 4-7 R_LPM_Initialize 関数仕様

書式	void R_LPM_Initialize(void)
仕様説明	ドライバ内の変数の初期化を行います 変数の初期化はリセット後 1 回のみ行い、複数回実行された場合は何もせずに関数を終了します
引数	なし
戻り値	なし
備考	ー

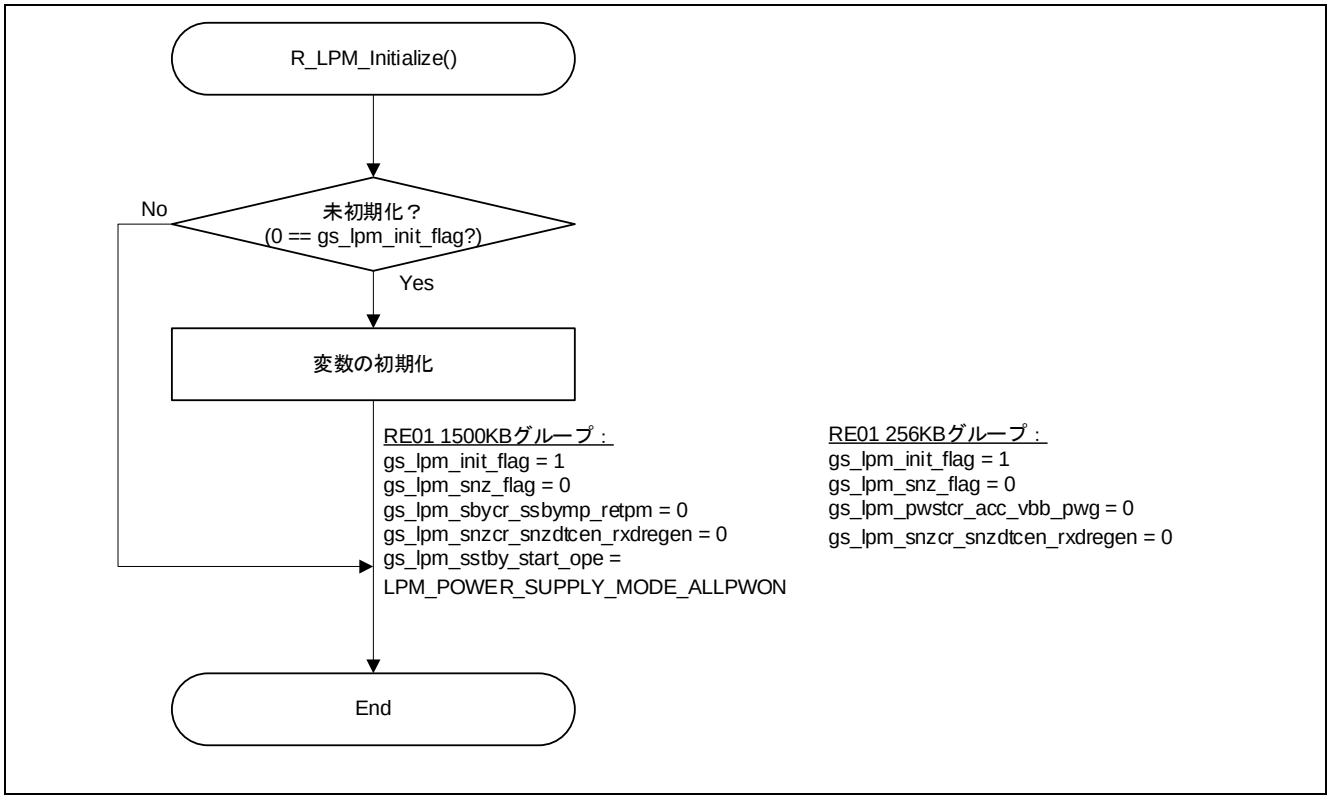

図 4-17 R_LPM_Initialize 関数処理フロー

4.3.3　R_LPM_ModuleStart 関数

表 4-8 R_LPM_ModuleStart 関数仕様

書式	int32_t R_LPM_ModuleStart(e_lpm_mstp_t module)
仕様説明	指定モジュールのモジュールストップ状態を解除します　※2 モジュールストップ状態を解除すると、周辺モジュールへのクロック出力が許可されレジスタの設定が可能になります 1 モジュールのみ指定できます
引数[注]	e_lpm_mstp_t module [入力] 　モジュールストップ状態を解除するモジュールを設定します
戻り値	正常 (0)　　：指定されたモジュールのモジュールストップ状態を解除しました
	エラー (-1)　　：指定されたモジュールのモジュールストップ状態を解除しませんでした 以下の状態を検出するとエラーになります <RE01 1500KB グループ> ― 引数 module は e_lpm_mstp_t の定義範囲を超えている ― 引数 module に LPM_MSTP_USB が指定され、BOOST モードで動作していない <RE01 256KB グループ> ― 引数 module は e_lpm_mstp_t の定義範囲を超えている
備考	ー

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.2 モジュールストップを参照ください。

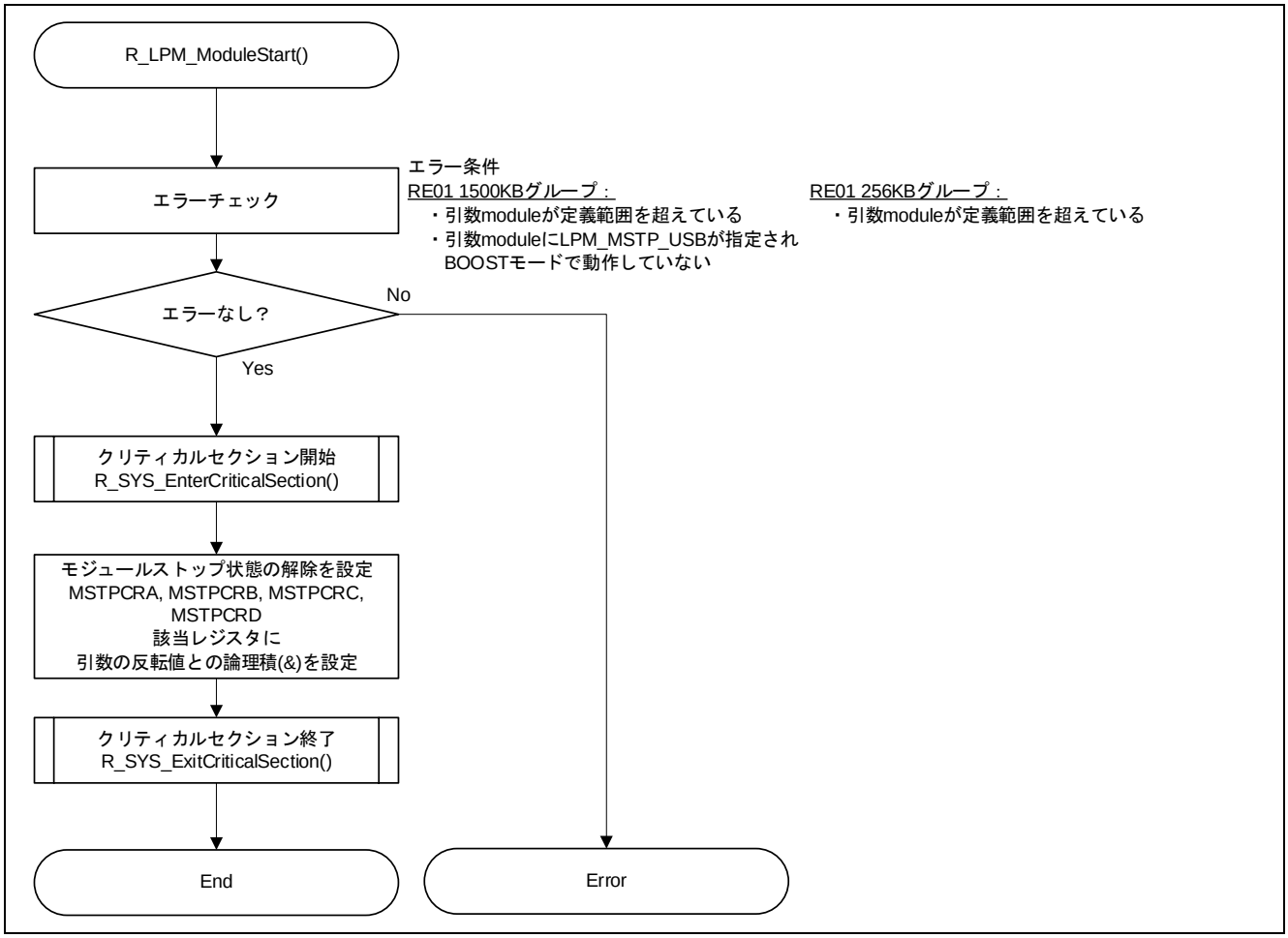

図 4-18 R_LPM_ModuleStart 関数処理フロー

4.3.4　R_LPM_ModuleStop 関数

表 4-9 R_LPM_ModuleStop 関数仕様

書式	int32_t R_LPM_ModuleStop(e_lpm_mstp_t module)
仕様説明	指定モジュールの動作を停止しモジュールストップ状態へ遷移します　※2 モジュールストップ状態へ遷移すると、電力を節約するために周辺モジュールへのクロック出力が禁止されます 1 モジュールのみ指定できます R_SYSTEM ドライバの R_SYS_ResourceLock()関数と R_SYS_ResourceUnLock()関数を使用します 本関数実行前に、R_SYS_Initialize()関数を実行してください
引数[注]	e_lpm_mstp_t module [入力] 　モジュールストップ状態へ遷移するモジュールを設定します
戻り値	正常 (0)　　：指定されたモジュールをモジュールストップ状態へ遷移しました
	エラー (-1)　　：指定されたモジュールをモジュールストップ状態へ遷移しませんでした 以下の状態を検出するとエラーになります ― 引数 module は e_lpm_mstp_t の定義範囲を超えている ― 引数 module で指定したモジュールとモジュールストップレジスタを共有する他の周辺モジュールが動作している 周辺モジュールが動作中かどうかは、R_SYS_ResourceLock 関数の実行状態で判断します
備考	ー

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.2 モジュールストップを参照ください。

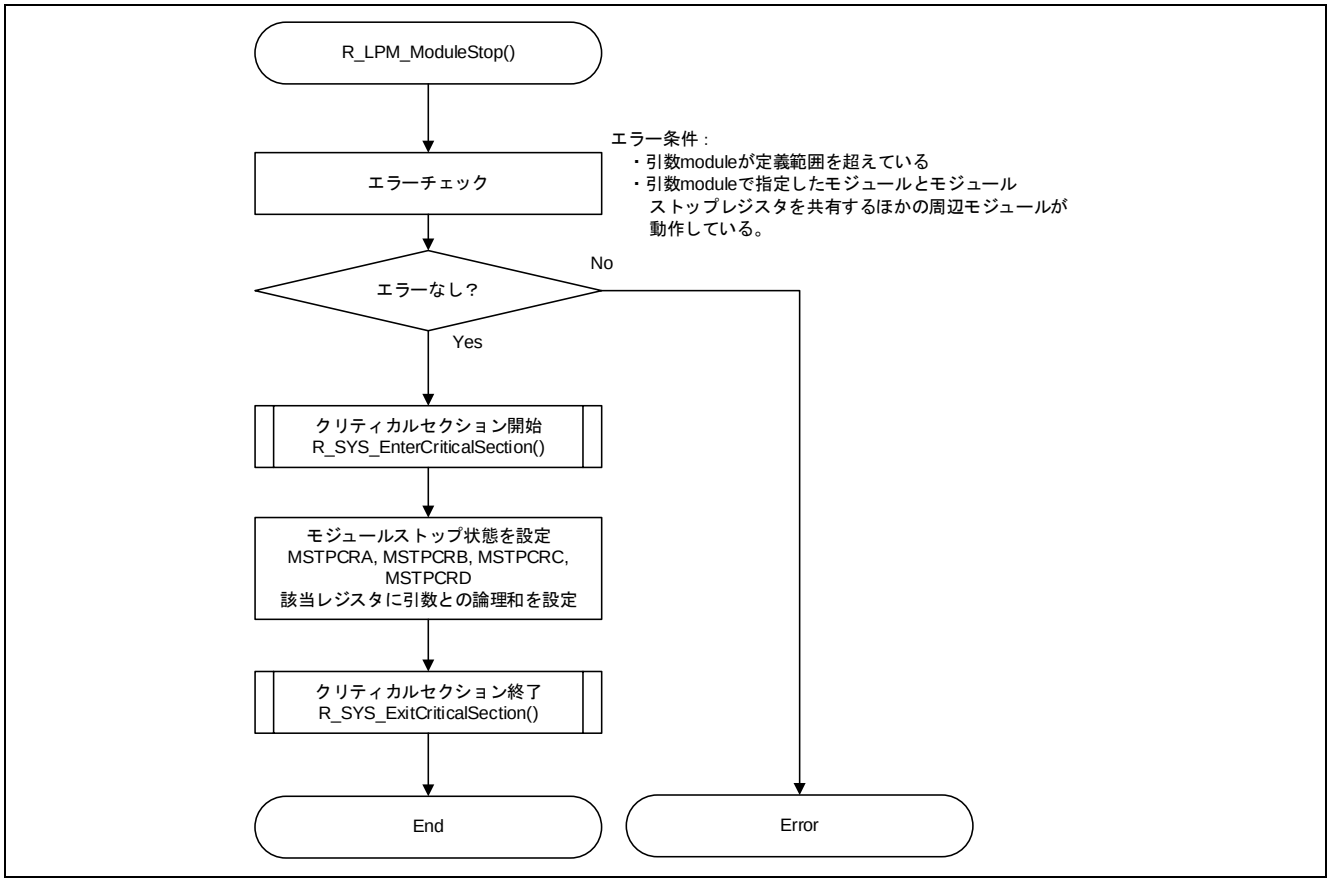

図 4-19 R_LPM_ModuleStop 関数処理フロー

4.3.5　R_LPM_FastModuleStart 関数

表 4-10 R_LPM_FastModuleStart 関数仕様

書式	int32_t R_LPM_FastModuleStart(uint32_t module_a, uint32_t module_b, uint32_t module_c, uint32_t module_d)
仕様説明	複数の指定モジュールのモジュールストップ状態を解除します　※2 モジュールストップ状態を解除すると、周辺モジュールへのクロック出力が許可されレジスタの設定が可能になります 引数 module_a,module_b,module_c,module_d に設定したモジュールのモジュールストップ状態を解除します 各引数には論理和で複数のモジュールを設定することができます
引数[注]	uint32_t module_a[入力]　：LPM_FAST_MSTPT_A で定義されます モジュールストップ状態を解除するモジュールを設定します
	uint32_t module_b[入力]　：LPM_FAST_MSTPT_B で定義されます モジュールストップ状態を解除するモジュールを設定します
	uint32_t module_c[入力]　：LPM_FAST_MSTPT_C で定義されます モジュールストップ状態を解除するモジュールを設定します
	uint32_t module_d[入力]　：LPM_FAST_MSTPT_D で定義されます モジュールストップ状態を解除するモジュールを設定します
戻り値	正常 (0)　：指定されたモジュールのモジュールストップ状態を解除しました
	エラー (-1)　：指定されたモジュールのモジュールストップ状態を解除しませんでした 以下の状態を検出するとエラーになります <RE01 1500KB グループ> ― 引数 module_b に LPM_FAST_MSTP_B_USB が指定され、BOOST モードで動作していない <RE01 256KB グループ> ― エラーなし（本関数の戻り値は常に「正常(0)」となります。）
備考	ー

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.2 モジュールストップを参照ください。

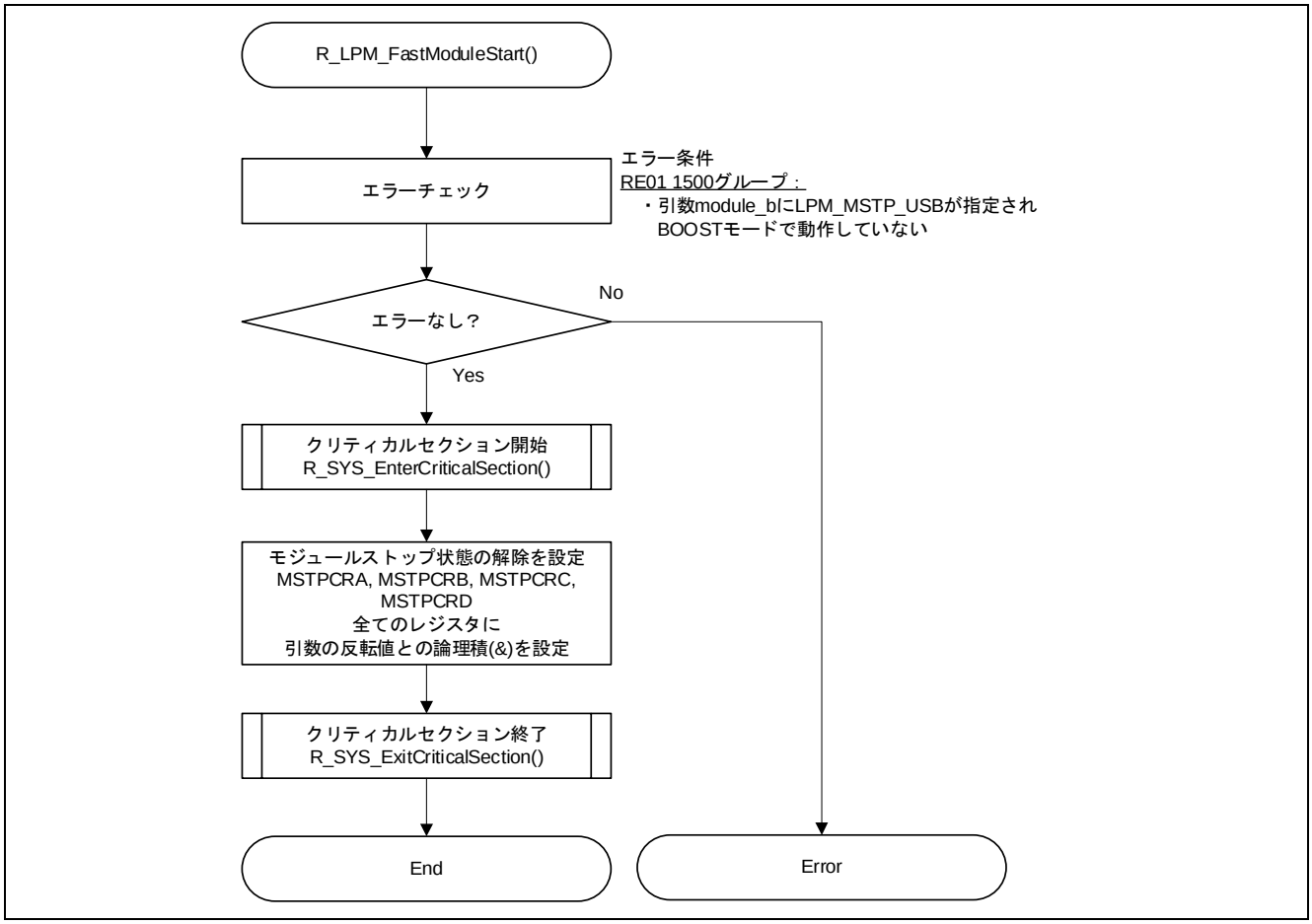

図 4-20 R_LPM_FastModuleStart 関数処理フロー

4.3.6　R_LPM_FastModuleStop 関数

表 4-11 R_LPM_FastModuleStop 関数仕様

書式	int32_t R_LPM_FastModuleStop(uint32_t module_a, uint32_t module_b, uint32_t module_c, uint32_t module_d)
仕様説明	複数の指定モジュールの動作を停止しモジュールストップ状態へ遷移します　※2 モジュールストップ状態へ遷移すると、電力を節約するために周辺モジュールへのクロック出力が停止します 引数 module_a, module_b, module_c, module_d に設定したモジュールをモジュールストップ状態へ遷移します 各引数には論理和で複数のモジュールを設定することができます R_SYSTEM ドライバの R_SYS_ResourceLock()関数と R_SYS_ResourceUnLock()関数を使用します 本関数実行前に、R_SYS_Initialize()関数を実行してください
引数[注]	uint32_t module_a[入力]　：LPM_FAST_MSTPT_A で定義されます モジュールストップ状態へ遷移するモジュールを設定します
	uint32_t module_b[入力]　：LPM_FAST_MSTPT_B で定義されます モジュールストップ状態へ遷移するモジュールを設定します
	uint32_t module_c[入力]　：LPM_FAST_MSTPT_C で定義されます モジュールストップ状態へ遷移するモジュールを設定します。
	uint32_t module_d[入力]　：LPM_FAST_MSTPT_D で定義されます モジュールストップ状態へ遷移するモジュールを設定します。
戻り値	正常 (0)　：指定されたモジュールをモジュールストップ状態へ遷移しました
	エラー (-1)　：指定されたモジュールをモジュールストップ状態へ遷移しませんでした 以下の状態を検出するとエラーになります — 引数 module_a, module_b, module_c, module_d で指定したモジュールとモジュールストップレジスタを共有する他の周辺モジュールが動作している 周辺モジュールが動作中かどうかは、R_SYS_ResourceLock 関数の実行状態で判断します
備考	ー

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.2 モジュールストップを参照ください。

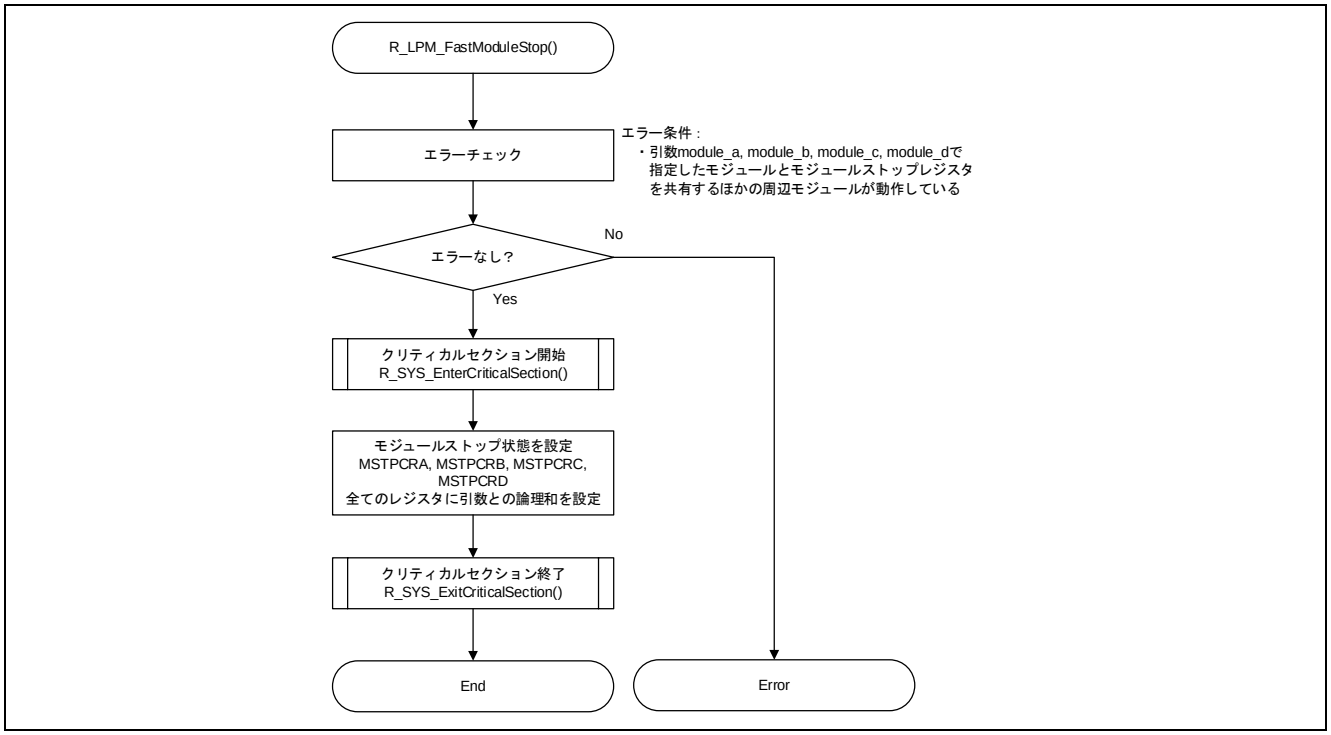

図 4-21 R_LPM_FastModuleStop 関数処理フロー

4.3.7　R_LPM_BackBiasModeEnable 関数

表 4-12 R_LPM_BackBiasModeEnable 関数仕様

書式	int32_t R_LPM_BackBiasModeEnable(e_lpm_vbb_clock_t clock)
仕様説明	BackBias 制御のセットアップを行います　※1 本関数を実行する前に、BackBias モードで使用するクロックを発振させる必要があります 本関数は BackBias 制御を有効にしますが、セットアップの完了待ちは行わないため、本関数を実行後、BackBias モードへ遷移する前に、R_LPM_BackBiasModeEnableStatusGet()関数を実行してセットアップが完了したことを確認する必要があります
引数	e_lpm_vbb_clock_t clock[入力] BackBias 制御のクロックを設定します
戻り値	正常 (0)　：BackBias 制御のセットアップを開始しました
	エラー (-1)　：BackBias 制御のセットアップを開始していません 以下の状態を検出するとエラーになります — 指定されたクロックが発振していない。 — 引数 clock が e_lpm_vbb_clock_t の定義範囲を超えている
備考	ー

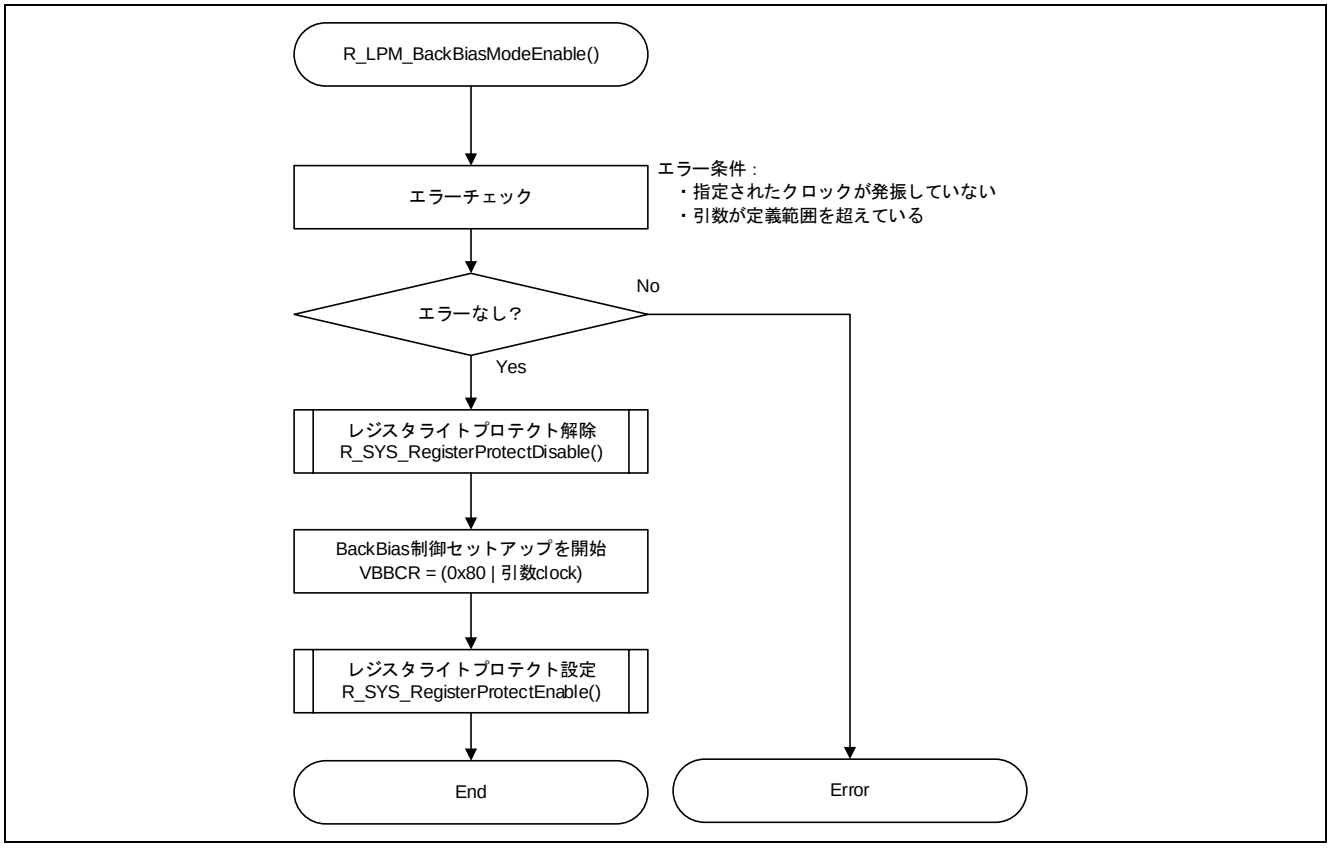

図 4-22 R_LPM_BackBiasModeEnable 関数処理フロー

4.3.8　R_LPM_BackBiasModeDisable 関数

表 4-13 R_LPM_BackBiasModeDisable 関数仕様

書式	int32_t R_LPM_BackBiasModeDisable(void)
仕様説明	BackBias 制御のセットアップ状態を解除します　※1 本関数を実行する前に、BackBias モードでの動作を停止する必要があります
引数	なし
戻り値	正常 (0)　：BackBias 制御のセットアップ状態を解除しました
	エラー (-1)　：BackBias 制御のセットアップ状態を解除していません 以下の状態を検出するとエラーになります ― BackBias モードで動作している
備考	ー

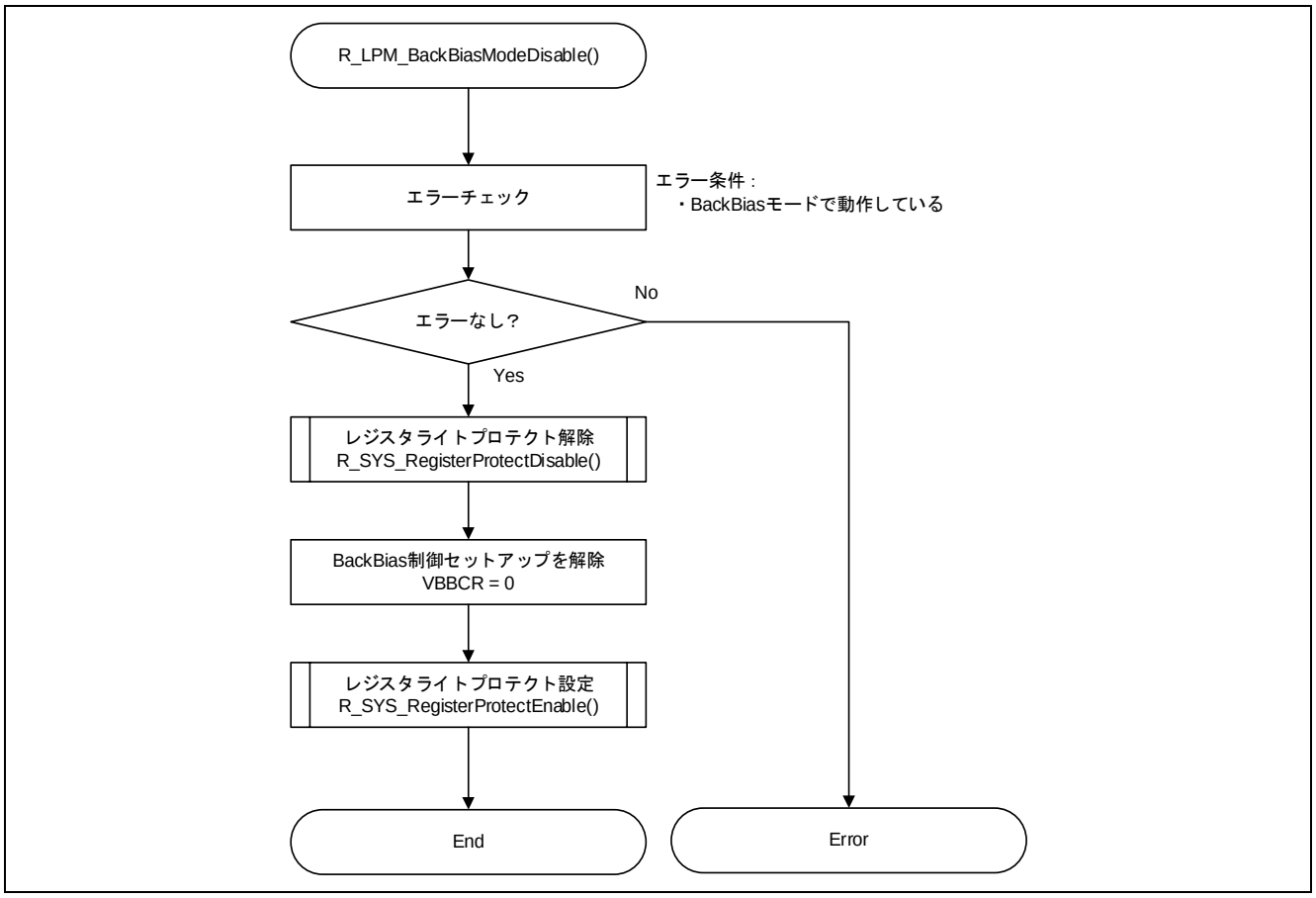

図 4-23 R_LPM_BackBiasModeDisable 関数処理フロー

4.3.9　R_LPM_BackBiasModeEnableStatusGet 関数

表 4-14 R_LPM_BackBiasModeEnableStatusGet 関数仕様

書式	e_lpm_vbb_enable_status_t R_LPM_BackBiasModeEnableStatusGet(void)
仕様説明	BackBias 制御のセットアップ状態を取得します R_LPM_BackBiasModeEnable() 関数を実行した後、BackBias モードに遷移する前に、本関数でセットアップが完了していることを確認してください
引数	なし
戻り値	LPM_VBB_DISABLED　：BackBias 制御のセットアップが完了していません
	LPM_VBB_ENABLED　：BackBias 制御のセットアップが完了しています
備考	ー

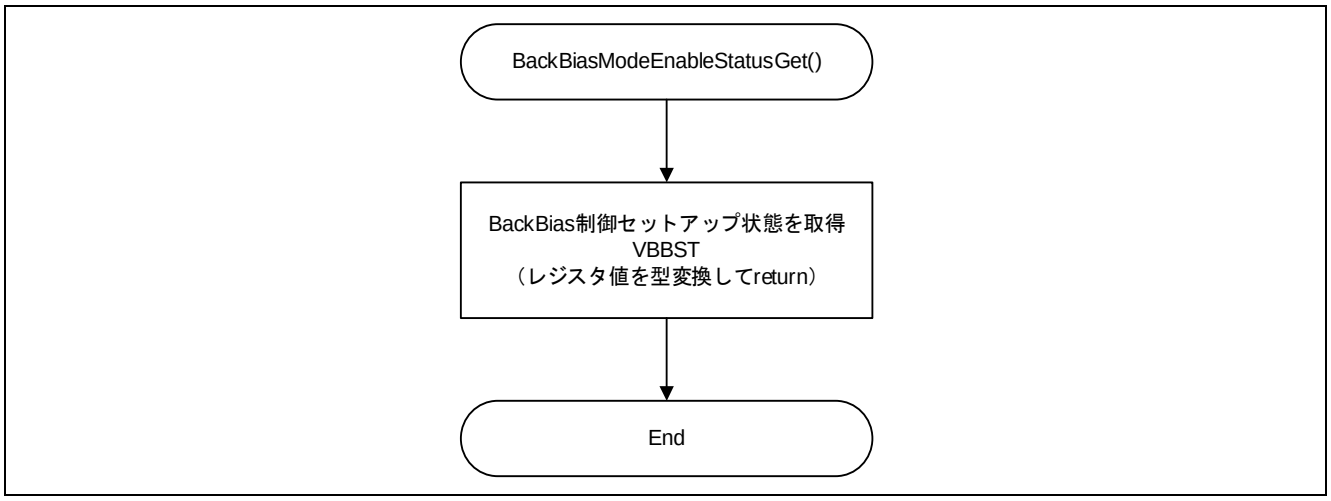

図 4-24 R_LPM_BackBiasModeEnableStatusGet 関数処理フロー

4.3.10　R_LPM_BackBiasModeEntry 関数

表 4-15 R_LPM_BackBiasModeEntry 関数仕様

書式	int32_t R_LPM_BackBiasModeEntry(void)
仕様説明	Back Bias モードに遷移します　※1 本関数を実行する前に、R_LPM_BackBiasModeEnable（）関数を実行してセットアップを完了する必要があります 本関数を実行する前に、R_LPM_BackBiasModeEnable（）関数で設定したクロック且つ Subosc-Speed モード[注]で動作している必要があります 本関数は、R_SYSTEM ドライバの内部関数を使用します。本関数実行前に、R_SYS_Initialize() 関数を実行してください
引数	なし
戻り値	正常 (0)　：BackBias モードに遷移しました
	エラー (-1)　：BackBias モードに遷移していません 以下の状態を検出するとエラーになります <RE01 1500KB グループ> — BackBias 制御のセットアップが完了していない — BOOST モードで動作している — Subosc-Speed モードで動作していない — BackBias 制御のセットアップ時に指定したクロックで動作していない — BackBias モードへの遷移をハードウェアへ設定できない <RE01 256KB グループ> — BackBias 制御のセットアップが完了していない — BackBias 制御のセットアップ時に指定したクロックで動作していない — BackBias モードへの遷移をハードウェアへ設定できない
備考	<RE01 1500KB グループ> 制限事項（No.60） SOSC-Speed で Normal モードにて SSTBY モードへ遷移後、スピード変更を一度もせずに BackBias モードへ遷移すると、SSTBY モードから復帰できません 本関数は、BackBias モードへ遷移する前に、一旦 Low-Speed に切り替えてから再度 SOSC-Speed モードを設定する処理を行います　※2

【注】 RE01 1500KB グループのみ

(1)　RE01 1500KB グループ

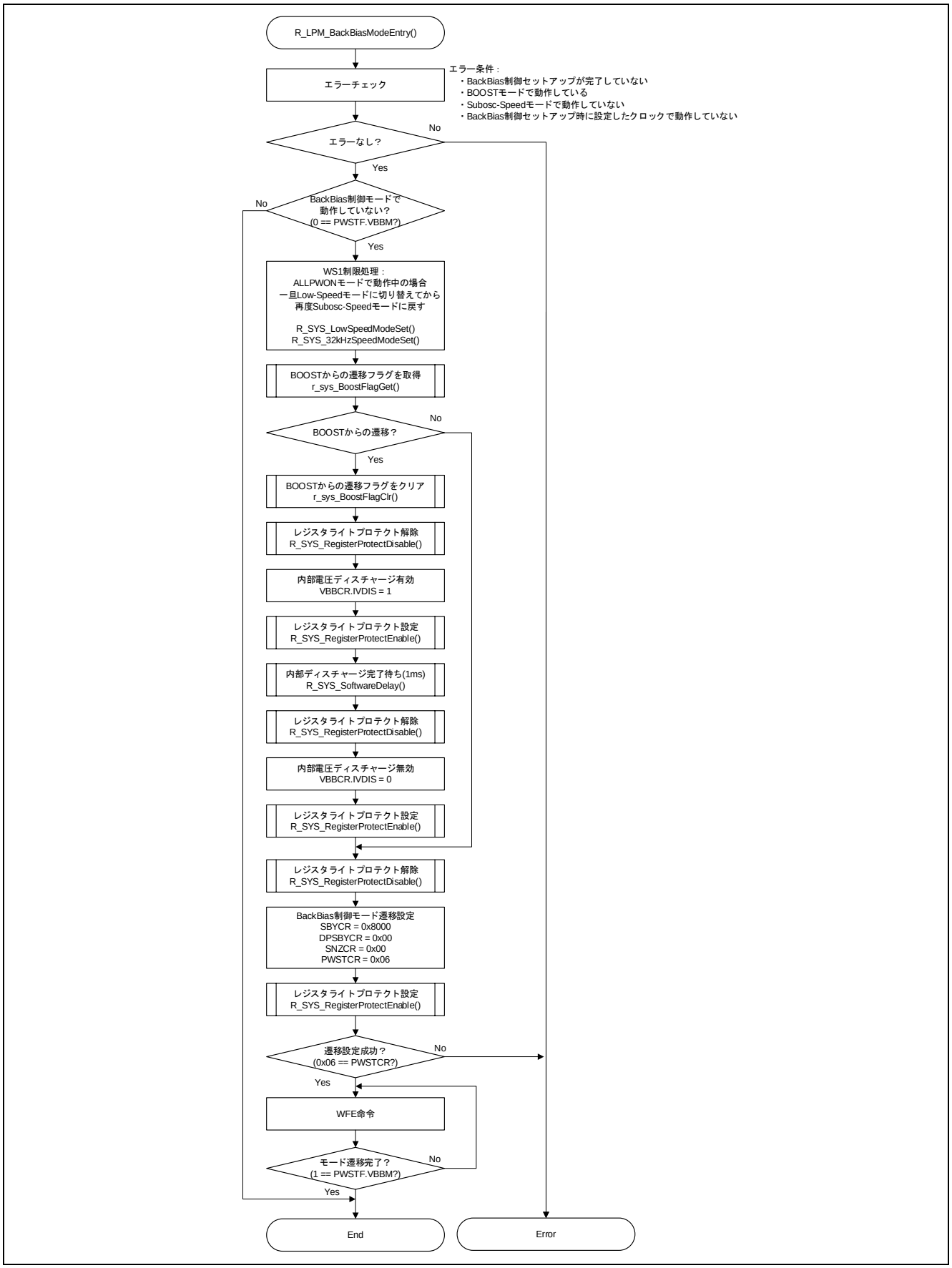

図 4-25 R_LPM_BackBiasModeEntry 関数処理フロー

(2)　RE01 256KB グループ

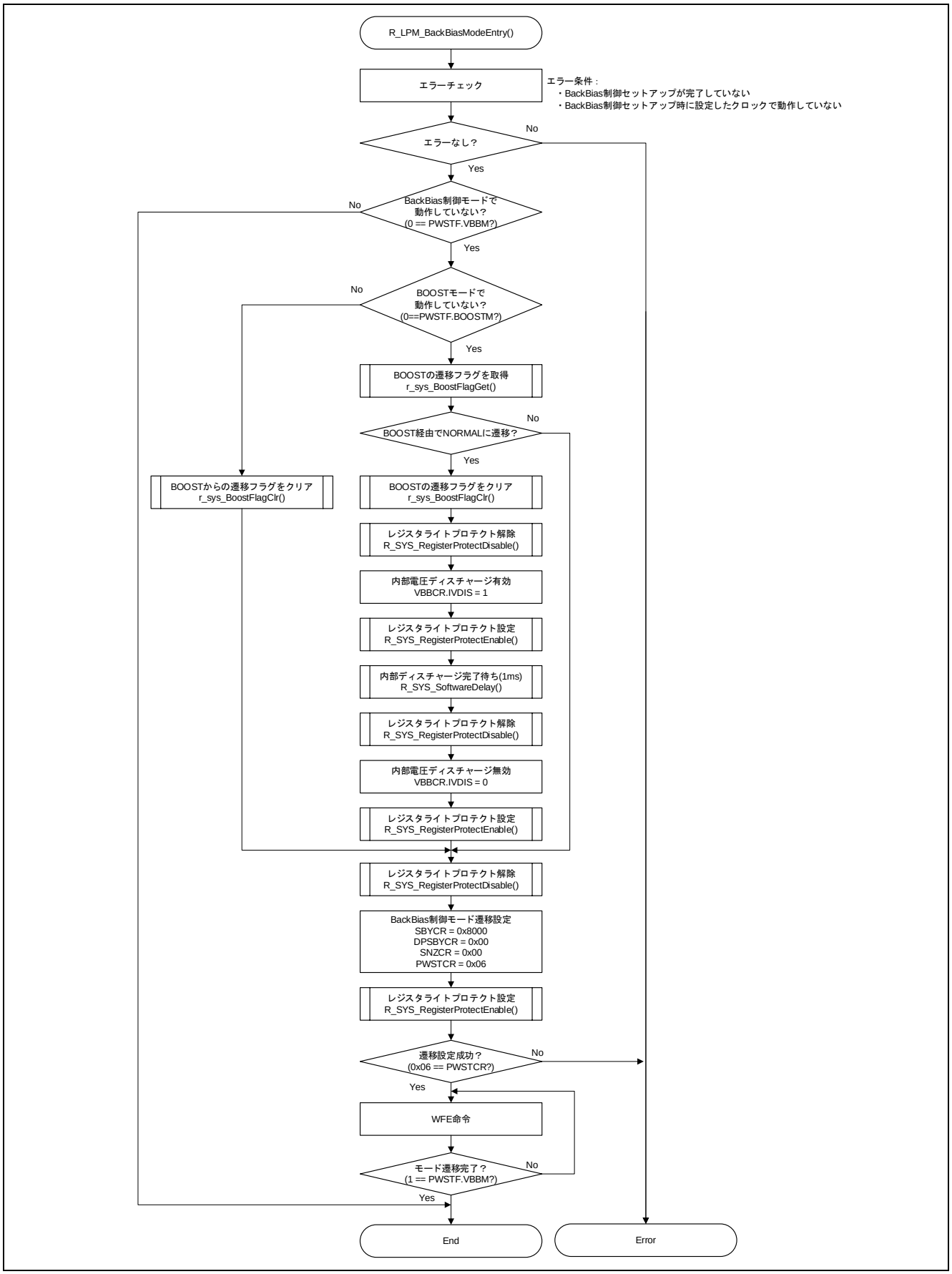

図 4-26 R_LPM_BackBiasModeEntry 関数処理フロー

4.3.11　R_LPM_BackBiasModeExit 関数

表 4-16 R_LPM_BackBiasModeExit 関数仕様

書式	int32_t R_LPM_BackBiasModeExit(void)
仕様説明	Back Bias モードから Normal モードに遷移します　※1
引数	なし
戻り値	正常 (0)　　：BackBias モードから Normal モードに遷移しました
	エラー (-1)　　：BackBias モードから Normal モードに遷移していません 以下の状態を検出するとエラーになります ― Normal モードへの遷移をハードウェアへ設定できない
備考	<RE01 256KB グループ> Back Bias モードから BOOST モードに遷移する場合は、R_SYSTEM ドライバの R_SYS_BoostSpeedModeSet()関数を使用してください

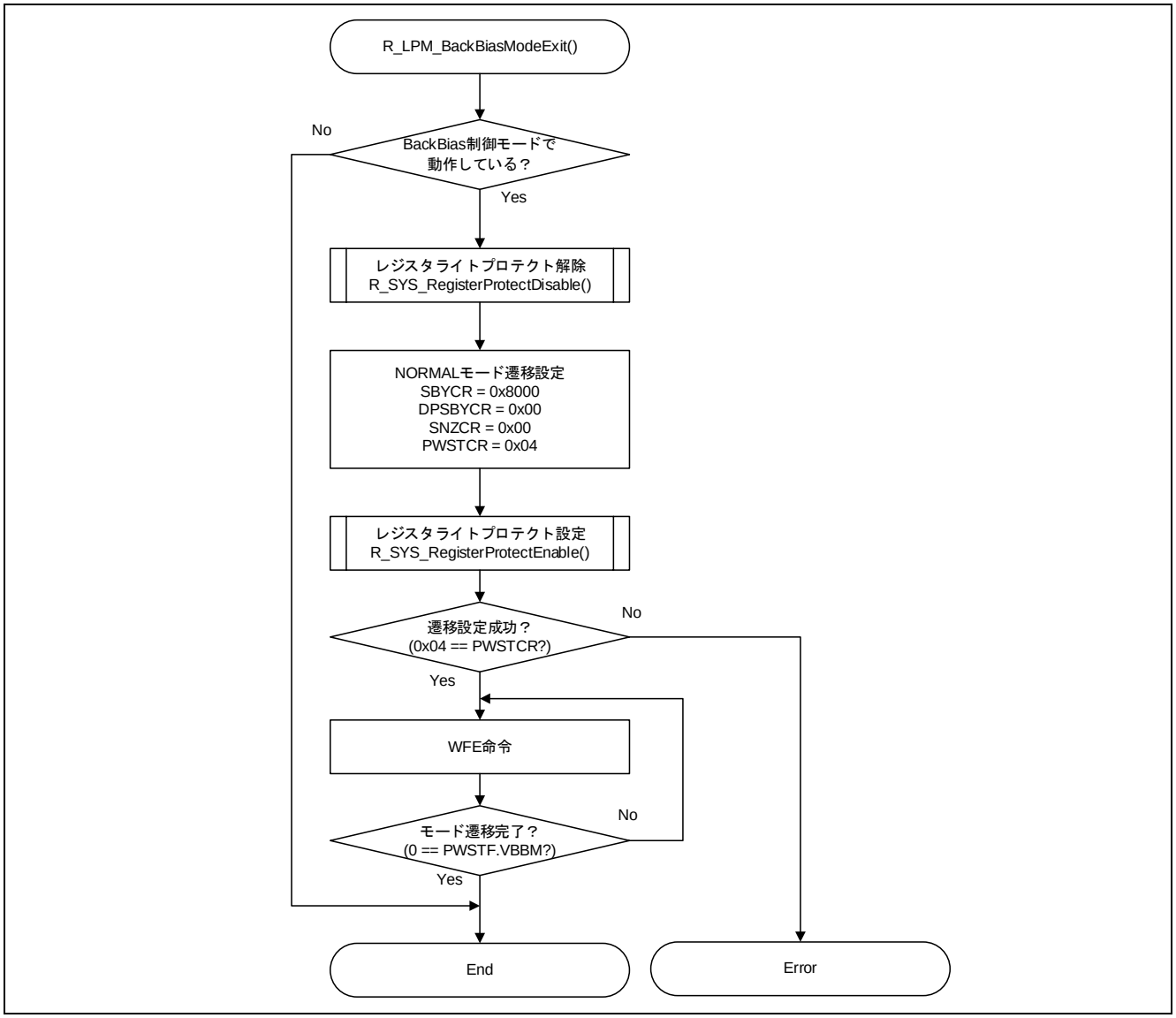

図 4-27 R_LPM_BackBiasModeExit 関数処理フロー

4.3.12　R_LPM_BackBiasModeGet 関数

表 4-17 R_LPM_BackBiasModeGet 関数仕様

書式	e_lpm_backbias_mode_t R_LPM_BackBiasModeGet(void)	
仕様説明	現在の電力制御モードを取得します	
引数	なし	
戻り値	LPM_BACKBIAS_MODE_NORMAL	:BackBias モード以外のモード（Normal もしくは BOOST）で動作しています。
	LPM_BACKBIAS_MODE_VBB	：BackBias モードで動作しています。
備考	ー	

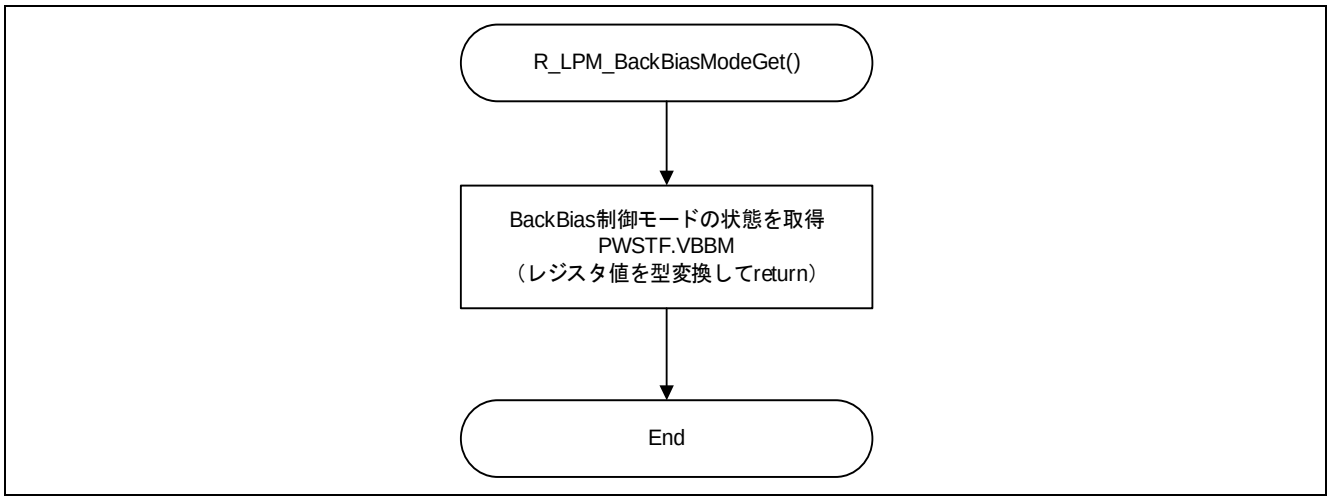

図 4-28 R_LPM_BackBiasModeGet 関数処理フロー

4.3.13 R_LPM_PowerSupplyModeAllpwonSet 関数

表 4-18 R_LPM_PowerSupplyModeAllpwonSet 関数仕様

書式	int32_t R_LPM_PowerSupplyModeAllpwonSet(void)
仕様説明	電源供給モードを ALLPWON モードへ遷移します　※1 <RE01 1500KB グループ> MINPWON モードで本関数を実行した場合、MINPWON モードから EXFPWON モードに遷移後、EXFPWON モードから ALLPWON モードに遷移します
	本関数は、MINPWON モードで停止する ISO2 領域のリセット解除を待つため、MINPWON モードから ALLPWON モードへ遷移後、r_lpm_wait_release_reset_iso2()関数を実行します。本処理は、PCLKB の分周設定が 4 から 64 分周の場合に必要です。r_lpm_wait_release_reset_iso2()関数は、R_LPM ドライバ内で WEAK 関数として実装しています。同一名称の非 WEAK 関数を実装することで、ドライバ内の該当関数を無効にできます
引数	なし
戻り値	正常 (0)　：ALLPWON モードへ遷移しました。
	エラー (-1)　：ALLPWON モードへ遷移しませんでした。 以下の状態を検出するとエラーになります — 電源供給モードの遷移をハードウェアへ設定できない — 動作クロックが 4MHz を超えている
備考	ー

(1)　RE01 1500KB グループ

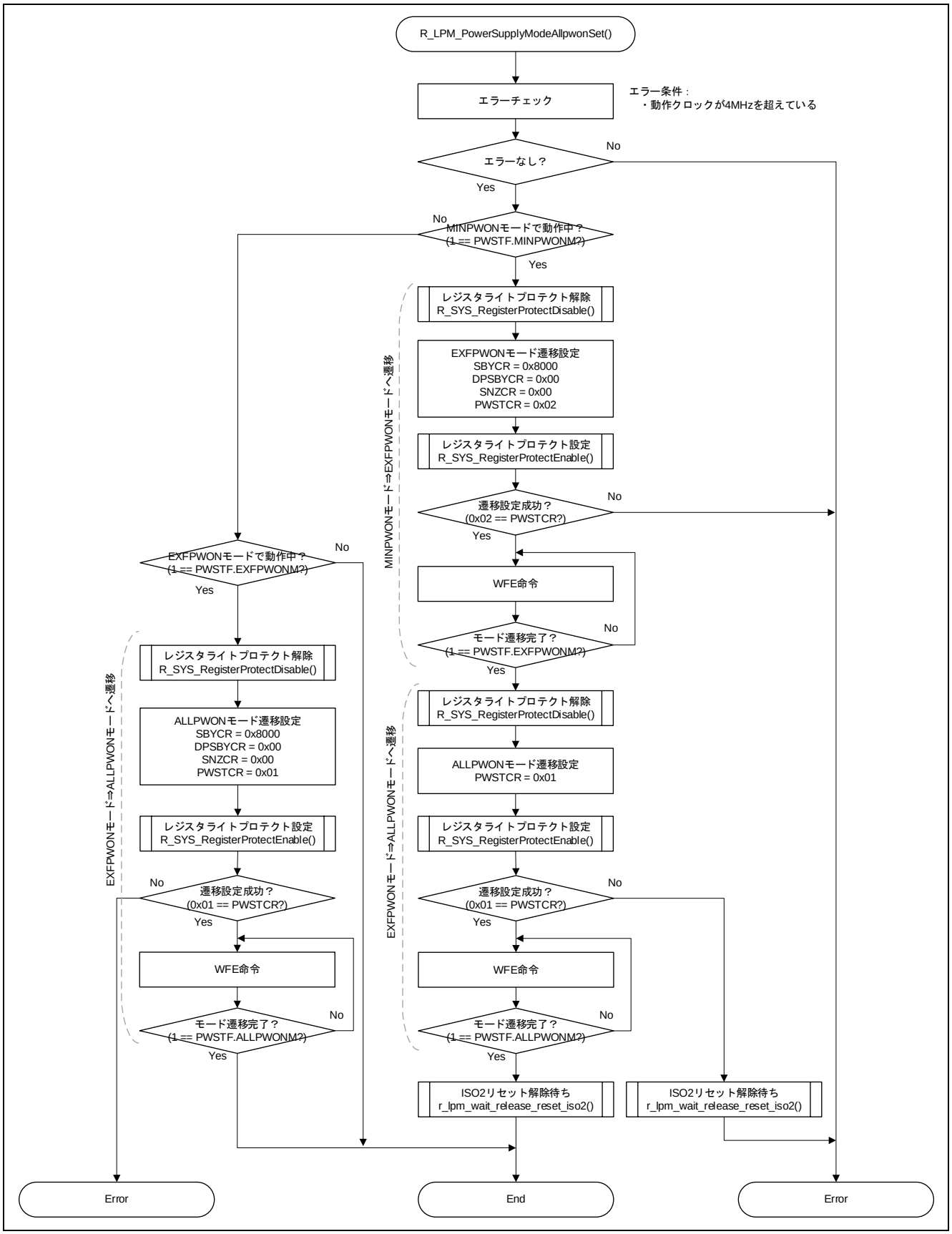

図 4-29 R_LPM_PowerSupplyModeAllpwonSet 関数処理フロー

(2)　RE01 256KB グループ

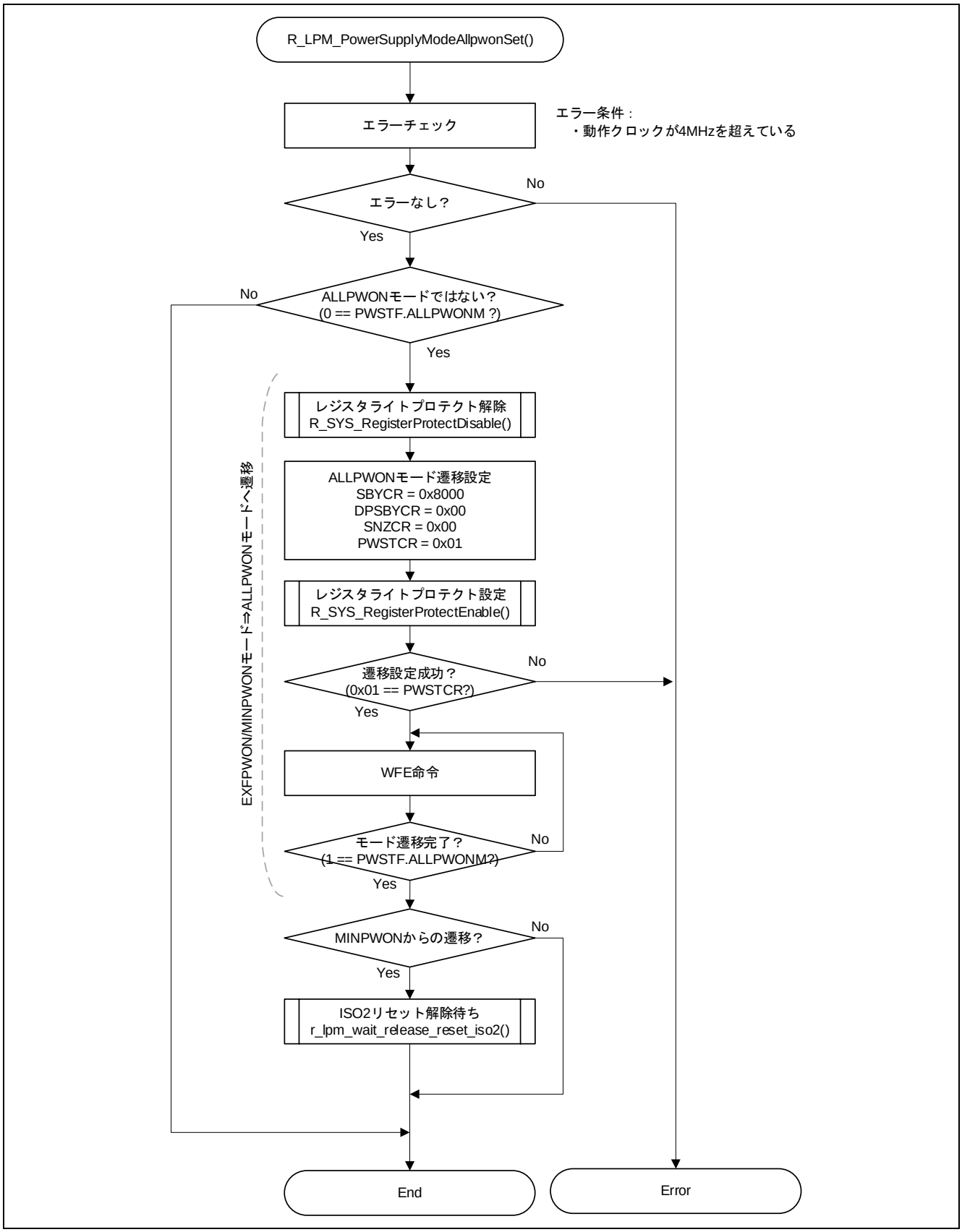

図 4-30 R_LPM_PowerSupplyModeAllpwonSet 関数処理フロー

4.3.14　R_LPM_PowerSupplyModeExfpwonSet 関数

表 4-19 R_LPM_PowerSupplyModeExfpwonSet 関数仕様

書式	int32_t R_LPM_PowerSupplyModeExfpwonSet(void)
仕様説明	電源供給モードを EXFPWON モードへ遷移します　※1
	本関数は、MINPWON モードで停止する ISO2 領域のリセット解除を待つため、MINPWON モードから EXFPWON モードへ遷移後、r_lpm_wait_release_reset_iso2()関数を実行します。本処理は、PCLKB の分周設定が 4 から 64 分周の場合に必要です。r_lpm_wait_release_reset_iso2()関数は、R_LPM ドライバ内で WEAK 関数として実装しています。同一名称の非 WEAK 関数を実装することで、ドライバ内の該当関数を無効にできます
引数	なし
戻り値	正常 (0)　　：EXFPWON モードへ遷移しました
	エラー (-1)　　：EXFPWON モードへ遷移しませんでした 以下の状態を検出するとエラーになります — 電源供給モードの遷移をハードウェアへ設定できない — BOOST モードで動作している
備考	ー

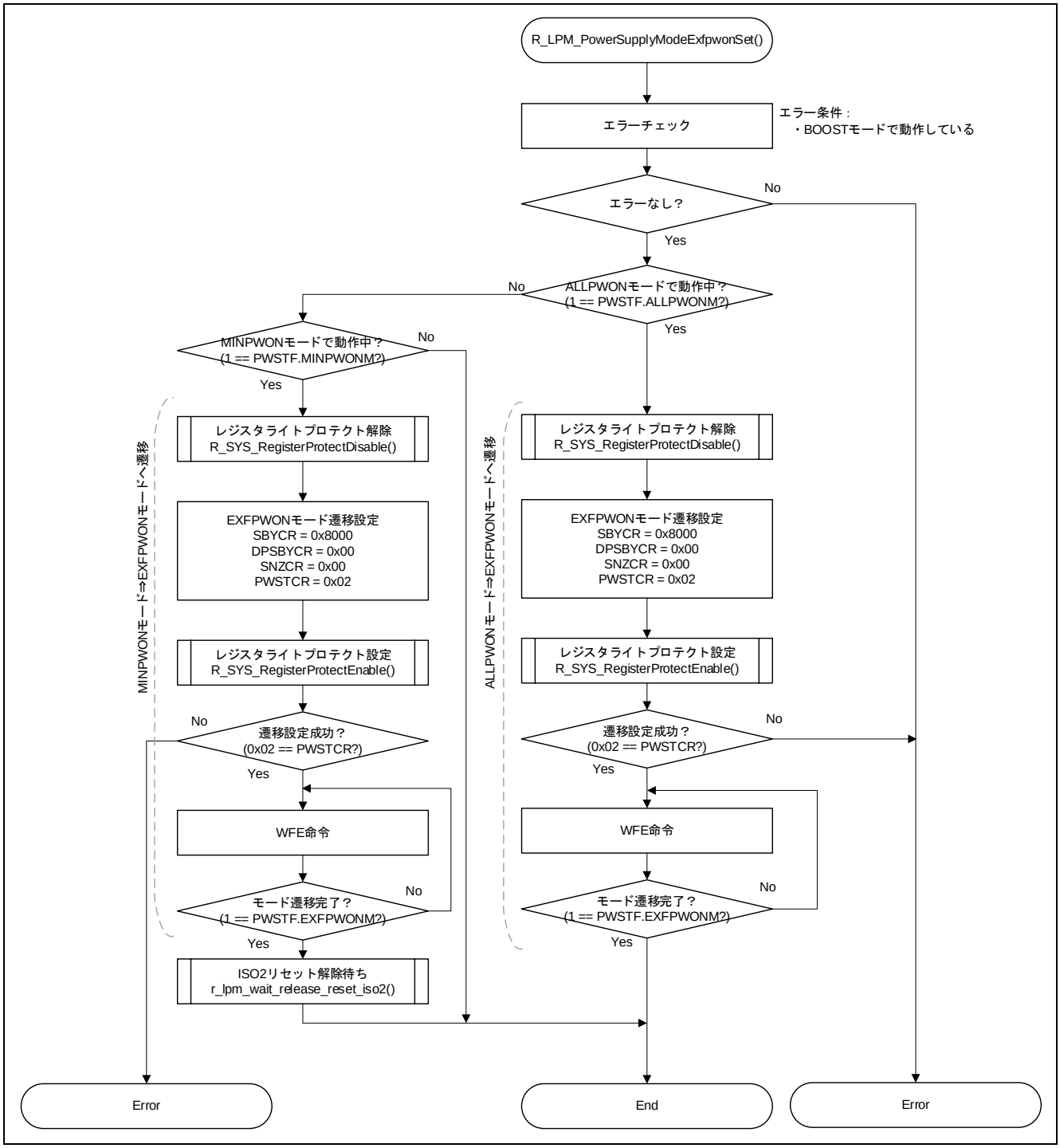

図 4-31 R_LPM_PowerSupplyModeExfpwonSet 関数処理フロー

4.3.15 R_LPM_PowerSupplyModeMinpwonSet 関数

表 4-20 R_LPM_PowerSupplyModeMinpwonSet 関数仕様

書式	int32_t R_LPM_PowerSupplyModeMinpwonSet(void)
仕様説明	電源供給モードを MINPWON モードへ遷移します　※1 <RE01 1500KB グループ> ALLPWON モードで本関数を実行した場合、ALLPWON モードから EXFPWON モードに遷移後、EXFPWON モードから MINPWON モードに遷移します
引数	なし
戻り値	正常 (0)　：MINPWON モードへ遷移しました
	エラー (-1)　：MINPWON モードへ遷移しませんでした 以下の状態を検出するとエラーになります — 電源供給モードの遷移をハードウェアへ設定できない。 — BOOST モードで動作している
備考	ー

(1)　RE01 1500KB グループ

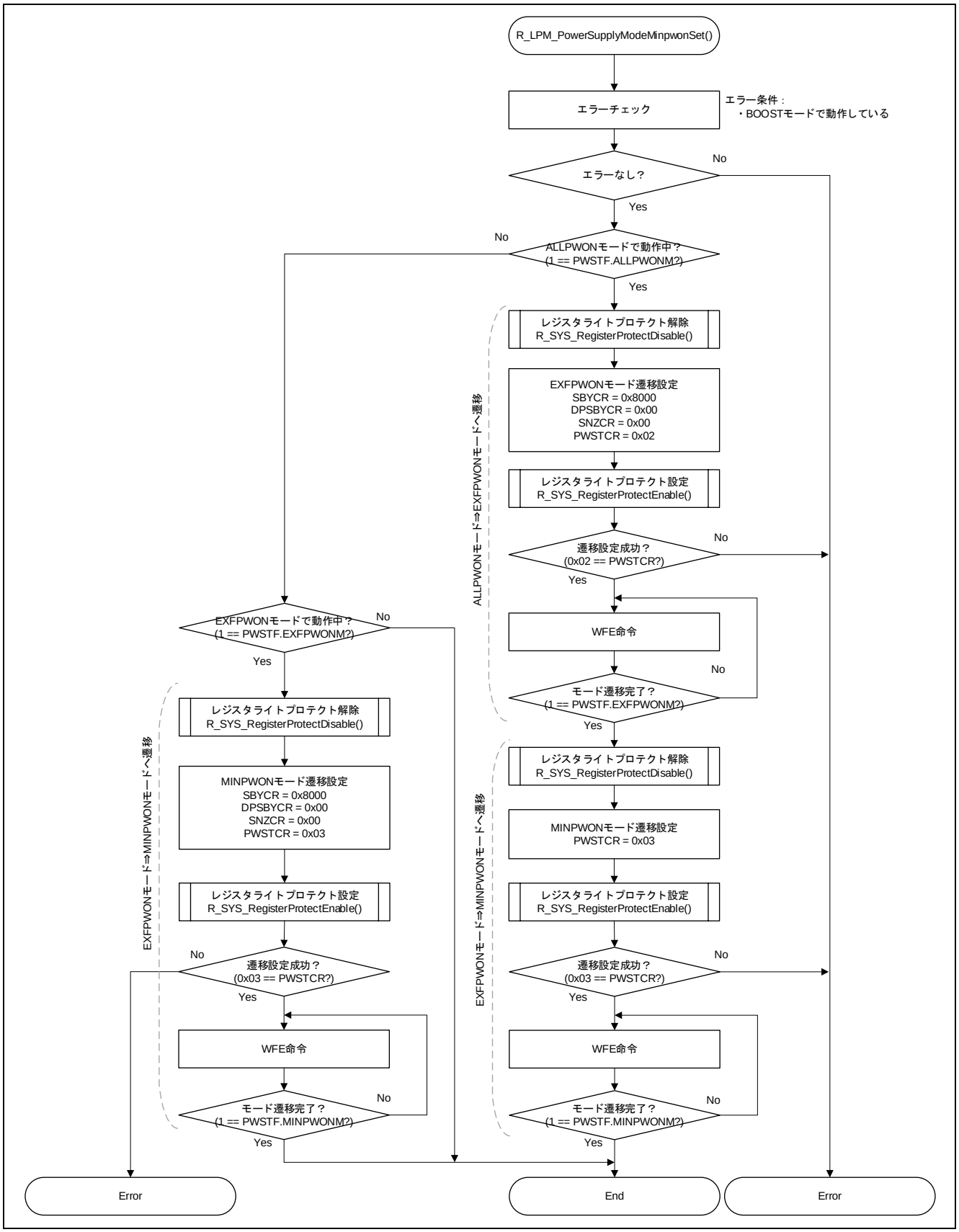

図 4-32 R_LPM_PowerSupplyModeMinpwonSet 関数処理フロー

(2)　RE01 256KB グループ

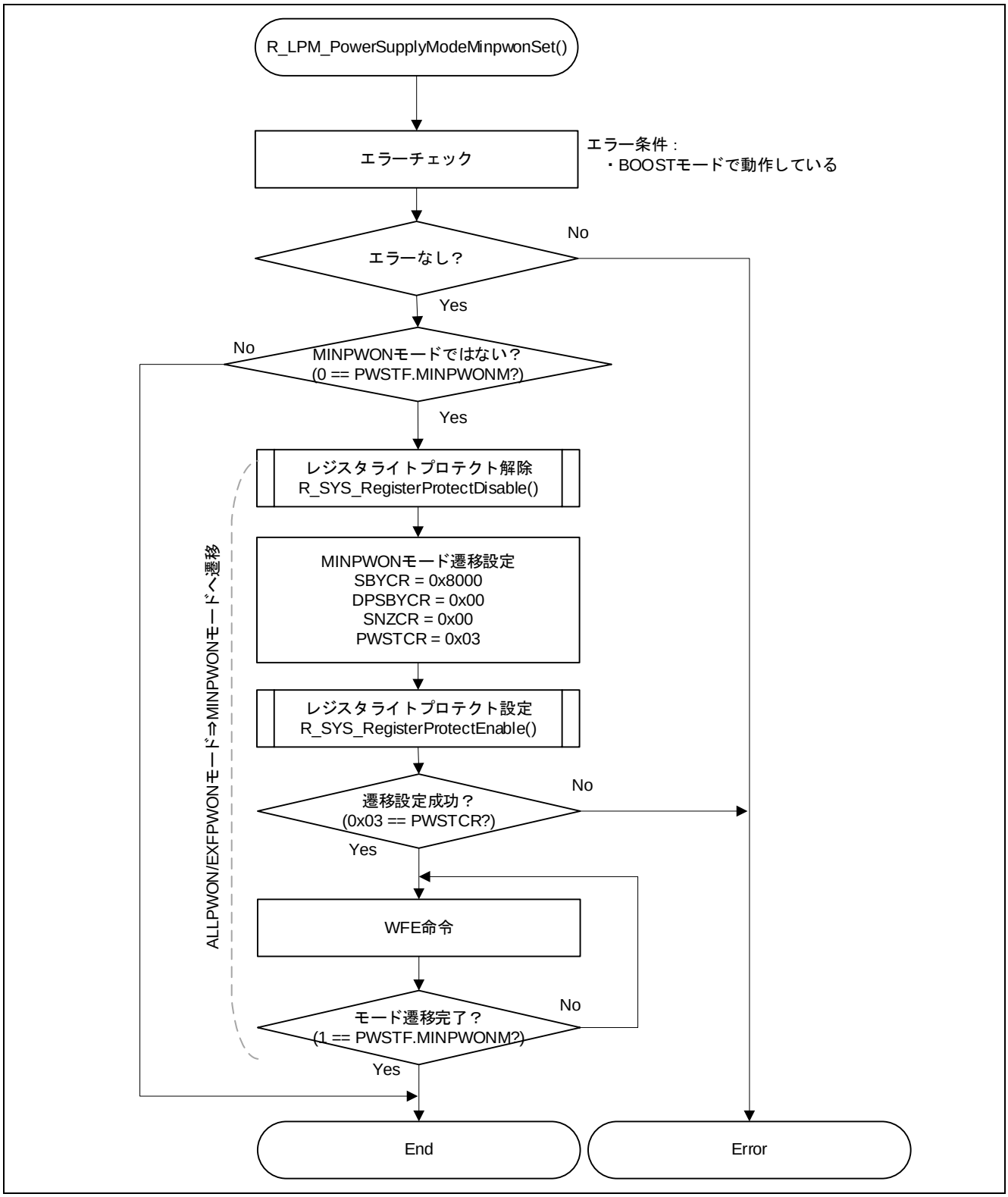

図 4-33 R_LPM_PowerSupplyModeMinpwonSet 関数処理フロー

4.3.16　R_LPM_PowerSupplyModeGet 関数

表 4-21. R_LPM_PowerSupplyModeGet 関数仕様

書式	e_lpm_power_supply_mode_t R_LPM_PowerSupplyModeGet(void)	
仕様説明	現在の電源供給モードを取得します	
引数	なし	
戻り値	LPM_POWER_SUPPLY_MODE_ALLPWON	：ALLPWON モードで動作しています
	LPM_POWER_SUPPLY_MODE_EXFPWON	：EXFPWON モードで動作しています
	LPM_POWER_SUPPLY_MODE_MINPWON	：MINPWON モードで動作しています
備考	ー	

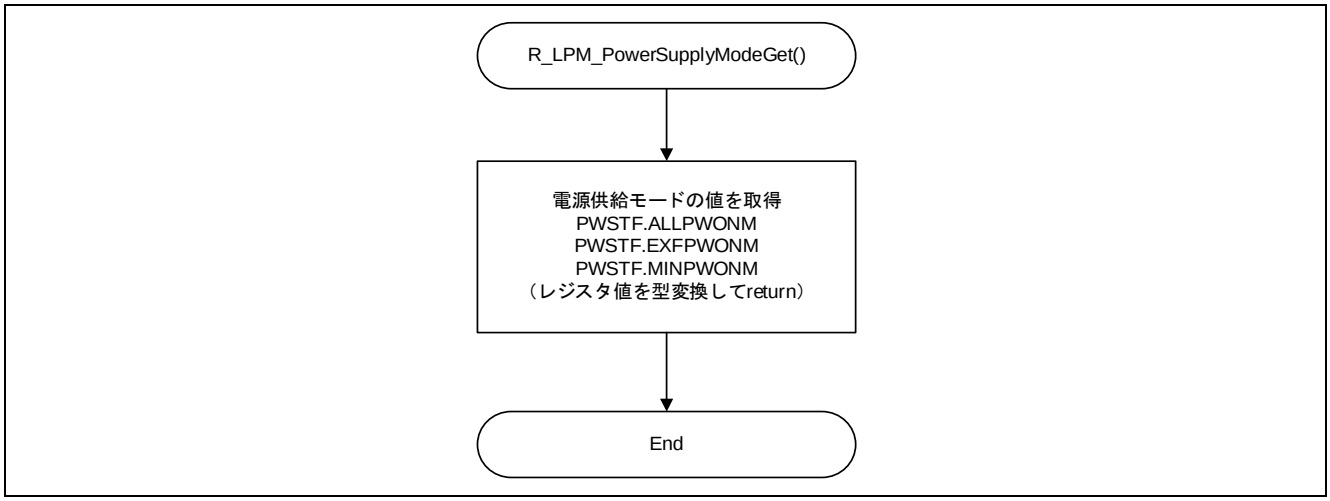

図 4-34 R_LPM_PowerSupplyModeGet 関数処理フロー

4.3.17　R_LPM_SnoozeSet 関数

表 4-22 R_LPM_SnoozeSet 関数仕様

書式	int32_t R_LPM_SnoozeSet(st_lpm_snooze_cfg_t * p_cfg)
仕様説明	スヌーズの動作条件を設定します　※1
引数[注]	st_lpm_snooze_cfg_t * p_cfg[入力] スヌーズの動作条件を格納した構造体 st_lpm_snooze_cfg_t のポインタを設定します
戻り値	正常 (0)　：スヌーズの動作条件を設定しました
	エラー (-1)　：スヌーズの動作条件を設定しませんでした 以下の状態を検出するとエラーになります — 引数 p_cfg が NULL — 引数 p_cfg->enable が e_lpm_snz_enable_t の定義範囲を超えている — 引数 p_cfg->req_rx0 は e_lpm_snz_rx0_falling_edge_t の定義範囲を超えている — 引数 p_cfg->dtc_enable が e_lpm_snz_dtc_enable_t の定義範囲を超えている — スヌーズからの復帰要因とスヌーズから SSTBY モードへの遷移要因に同一条件が設定されている（DTC と ADC が該当）
備考	ー

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.6 低消費電力モードを参照ください。

参考情報として、SSTBY/Snooze/OPE モード遷移要因を設定するレジスタを示します。スヌーズ中に、本関数の引数 p_cfg->wup に設定したイベントに加え、R_LPM_SSTBYModeSetup 関数の引数 p_cfg->wup に設定したイベントを検出するとスヌーズから OPE に復帰します。

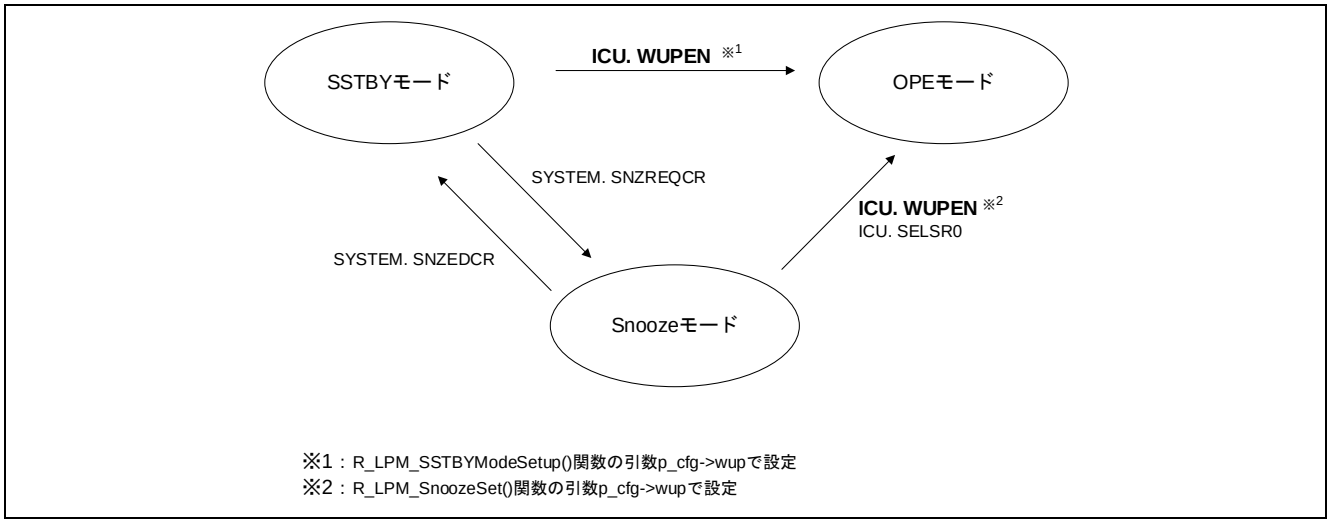

図 4-35 SSTBY/Snooze/OPE モード遷移要因を設定するレジスタ

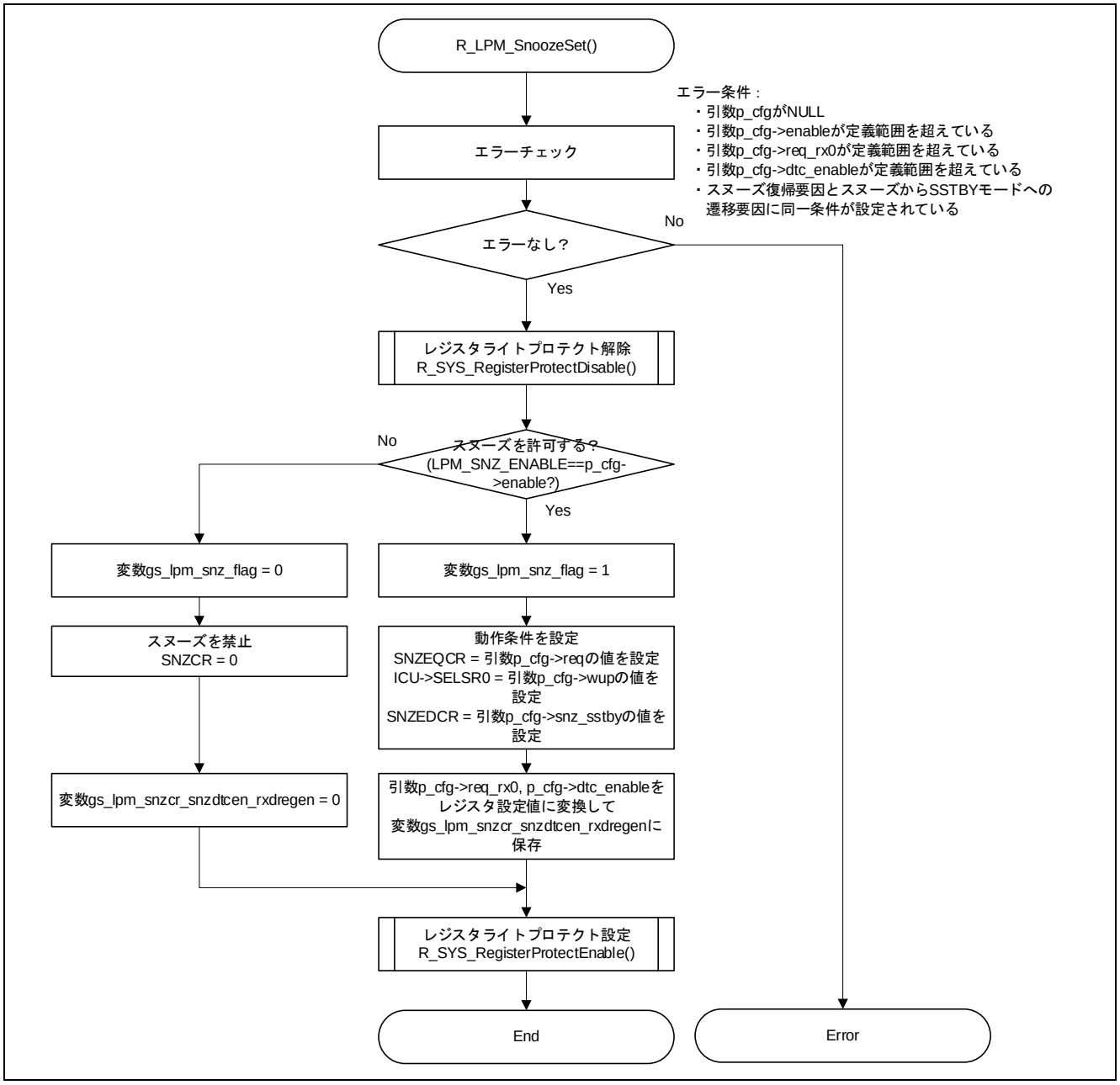

図 4-36 R_LPM_SnoozeSet 関数処理フロー

4.3.18　R_LPM_SleepModeEntry 関数

表 4-23 R_LPM_SleepModeEntry 関数仕様

書式	void R_LPM_SleepModeEntry(void)
仕様説明	スリープモードに遷移します　※1 本関数内でスリープモードに遷移し、スリープから復帰すると本関数が終了します
引数	なし
戻り値	なし
備考	ー

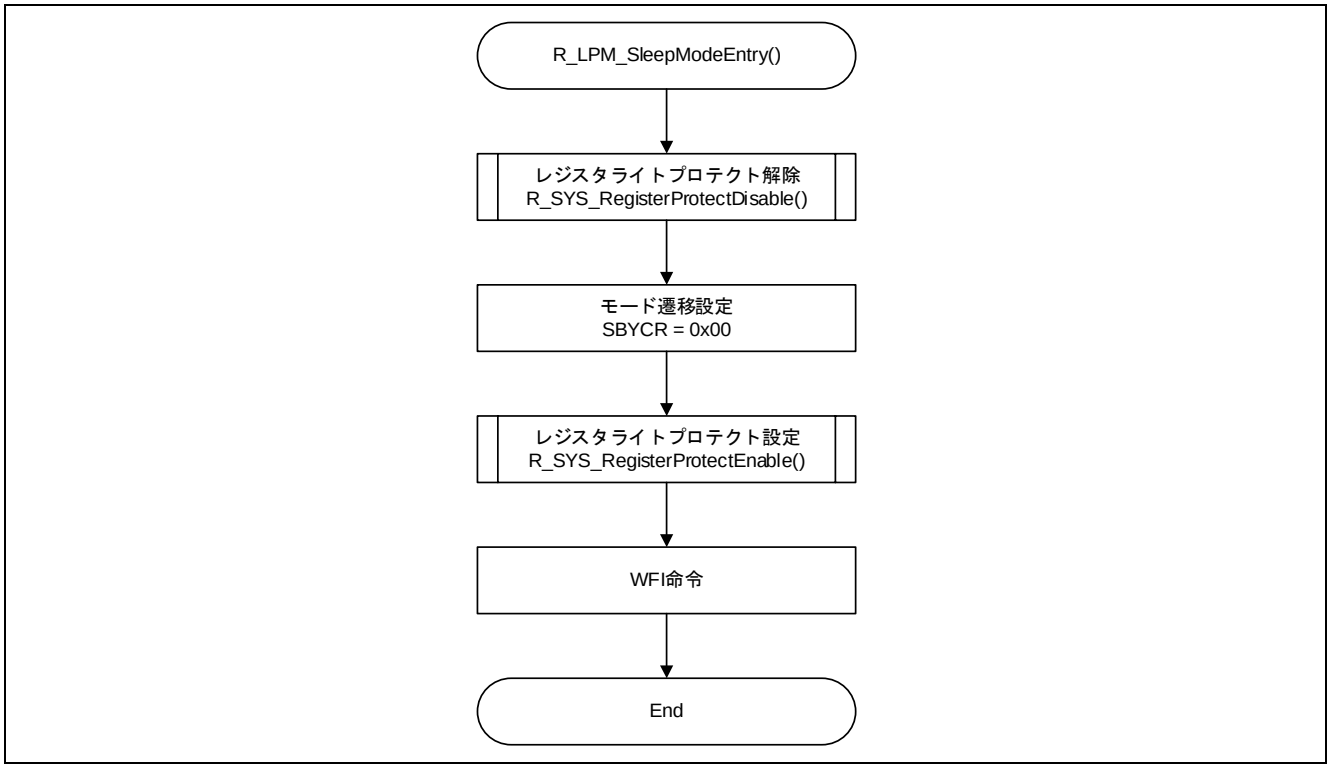

図 4-37 R_LPM_SleepModeEntry 関数処理フロー

4.3.19　R_LPM_SSTBYModeSetup 関数

表 4-24 R_LPM_SSTBYModeSetup 関数仕様

書式	int32_t R_LPM_SSTBYModeSetup(st_lpm_sstby_cfg_t * p_cfg)
仕様説明	SSTBY モードの動作条件を設定します　※1 <RE01 1500KB グループ> SSTBY モードに遷移した後の電源供給モードと、SSTBY モードから動作モードに遷移する前の電源供給モードは同じである必要があります
引数[注]	st_lpm_sstby_cfg_t * p_cfg[入力] SSTBY モードの動作条件を格納した構造体 st_lpm_sstby_cfg_t のポインタを設定します
戻り値	正常 (0)　：SSTBY モードの動作条件を設定しました。
	エラー (-1)　：SSTBY モードの動作条件を設定しませんでした。 以下の状態を検出するとエラーになります <RE01 1500KB グループ> — 引数 p_cfg が NULL — 引数 p_cfg->ope_sstby が e_lpm_ope_to_sstby_power_supply_mode_t の定義範囲を超えている — 引数 p_cfg->sstby_ope が e_lpm_sstby_to_ope_power_supply_mode_t の定義範囲を超えている — SSTBY モードに遷移した後の電源供給モードと、SSTBY モードから動作モードに遷移する前の電源供給モードが一致しない <RE01 256KB グループ> — 引数 p_cfg が NULL — 引数 p_cfg->speed が、e_lpm_sstby_speed_mode_t の定義範囲を超えている — 引数 p_cfg->power_supply が e_lpm _sstby_power_supply_mode_t の定義範囲を超えている
備考	<RE01 256KB グループ> 引数 p_cfg->wup で設定する SSTBY モード復帰要因について、CCC インターバル割り込みと WUPT インターバル割り込みのマクロ定義は併記されます

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.6 低消費電力モードを参照ください。

(1)　RE01 1500KB グループ

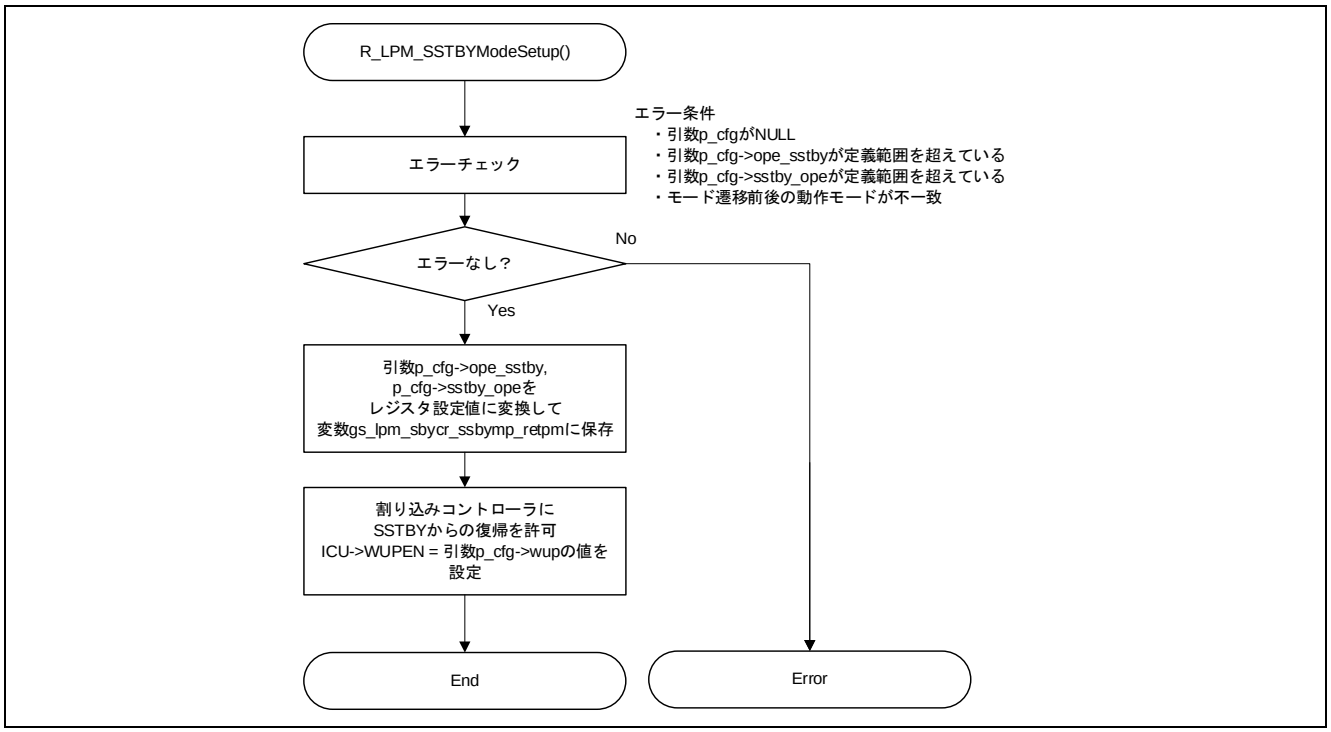

図 4-38 R_LPM_SSTBYModeSetup 関数処理フロー

(2)　RE01 256KB グループ

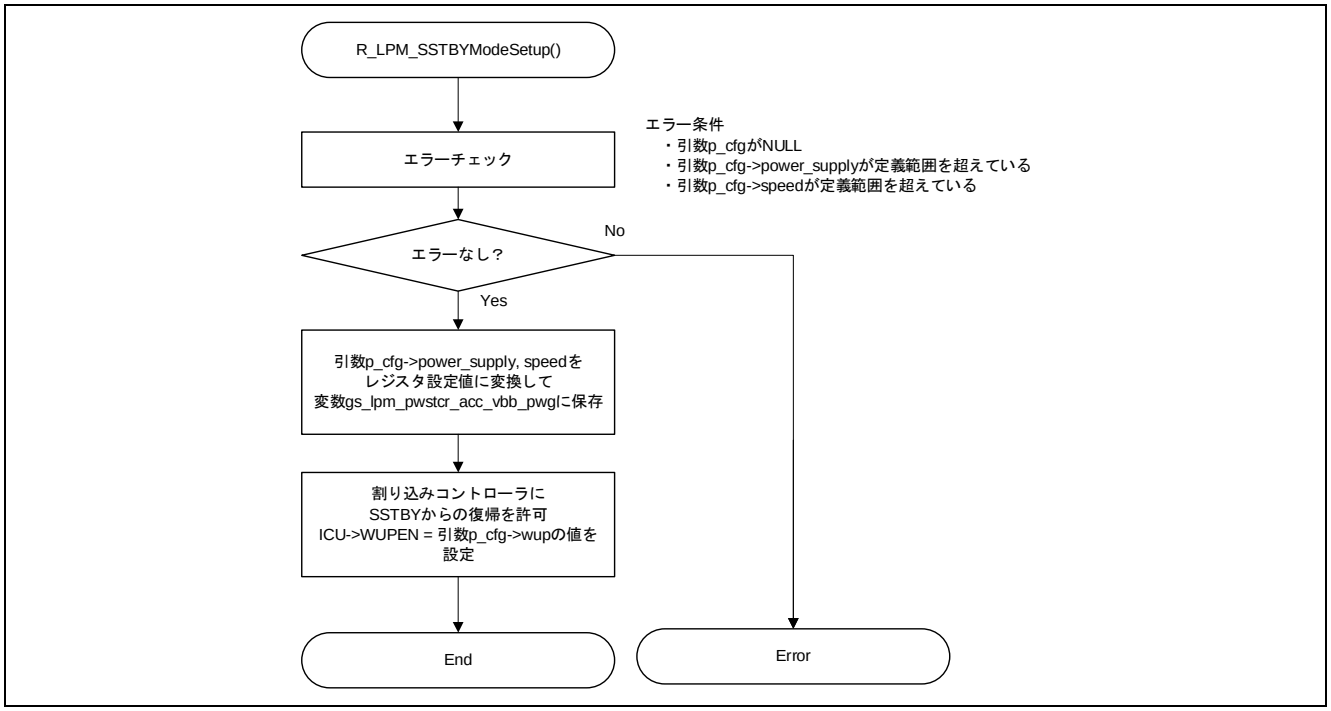

図 4-39 R_LPM_SSTBYModeSetup 関数処理フロー

4.3.20 R_LPM_SSTBYModeEntry 関数

表 4-25 R_LPM_SSTBYModeEntry 関数仕様

書式	int32_t R_LPM_SSTBYModeEntry(void)
仕様説明	SSTBY モードに遷移します　※1 本関数内で SSTBY モードに遷移し、SSTBY モードから復帰すると本関数が終了します 本関数を実行する前に、R_LPM_SSTBYModeSetup()関数を実行して SSTBY モードの動作条件を設定する必要があります <RE01 256KB グループ> BackBias モードで本関数を実行した場合、R_LPM_SSTBYModeSetup()関数で設定した引数 p_cfg->power_supply によって以下のモードに遷移します — LPM_OPE_TO_SSTBY_EXFPWON もしくは LPM_OPE_TO_SSTBY_EXFPWON_VBB の場合は EXFPWON VBB SSTBY モードへ遷移 — LPM_OPE_TO_SSTBY_MINPWON もしくは LPM_OPE_TO_SSTBY_MINPWON_VBB の場合は MINPWON VBB SSTBY モードへ遷移
	<RE01 256KB グループ> 本関数は、MINPWON モードで停止する ISO2 領域のリセット解除を待つため、MINPWON SSTBY モードから EXFPWON Ope へ復帰時、r_lpm_wait_release_reset_iso2()関数を実行します。本処理は、PCLKB の分周設定が 4 から 64 分周の場合に必要です。 r_lpm_wait_release_reset_iso2()関数は、R_LPM ドライバ内で WEAK 関数として実装しています。同一名称の非 WEAK 関数を実装することで、ドライバ内の該当関数を無効にできます。
引数	なし
戻り値	正常 (0)　：SSTBY モードへ遷移しました。
	エラー (-1)　：SSTBY モードへ遷移しませんでした。 以下の状態を検出するとエラーになります <RE01 1500KB グループ> — BOOST モードで動作している — 現在の電源供給モードの状態に対して、R_LPM_SSTBYModeSetup()関数で設定された SSTBY モードへ遷移前の電源供給モードの状態が適切ではない <RE01 256KB グループ> — HOCO 発振周波数が 48MHz もしくは 64MHz に設定されている — VBB SSTBY モードへ遷移時、BackBias 制御のセットアップが完了していない — MINPWON モードの状態に対して、R_LPM_SSTBYModeSetup()関数で設定された SSTBY モードの電源供給モードが MINPWON モードでない — ALLPWON で動作している場合、動作クロックが 4MHz を超えている
備考	<RE01 1500KB グループ> WS1 制限事項（No.34, 35, 36） Subosc-Speed モードおよび BackBias モードで、電源供給モードが EXFPWON から ALLPWON へ遷移するスヌーズからの復帰はできないためエラーを返します。
	<RE01 256KB グループ> HOCO が 48MHz または 64MHz で発振しているとき（HOCOMCR = 02H または 03H）、本関数の内部で行っている PWSTCR レジスタへの書込みが出来ません。 そのためエラー値を返します。

(1)　RE01 1500KB グループ

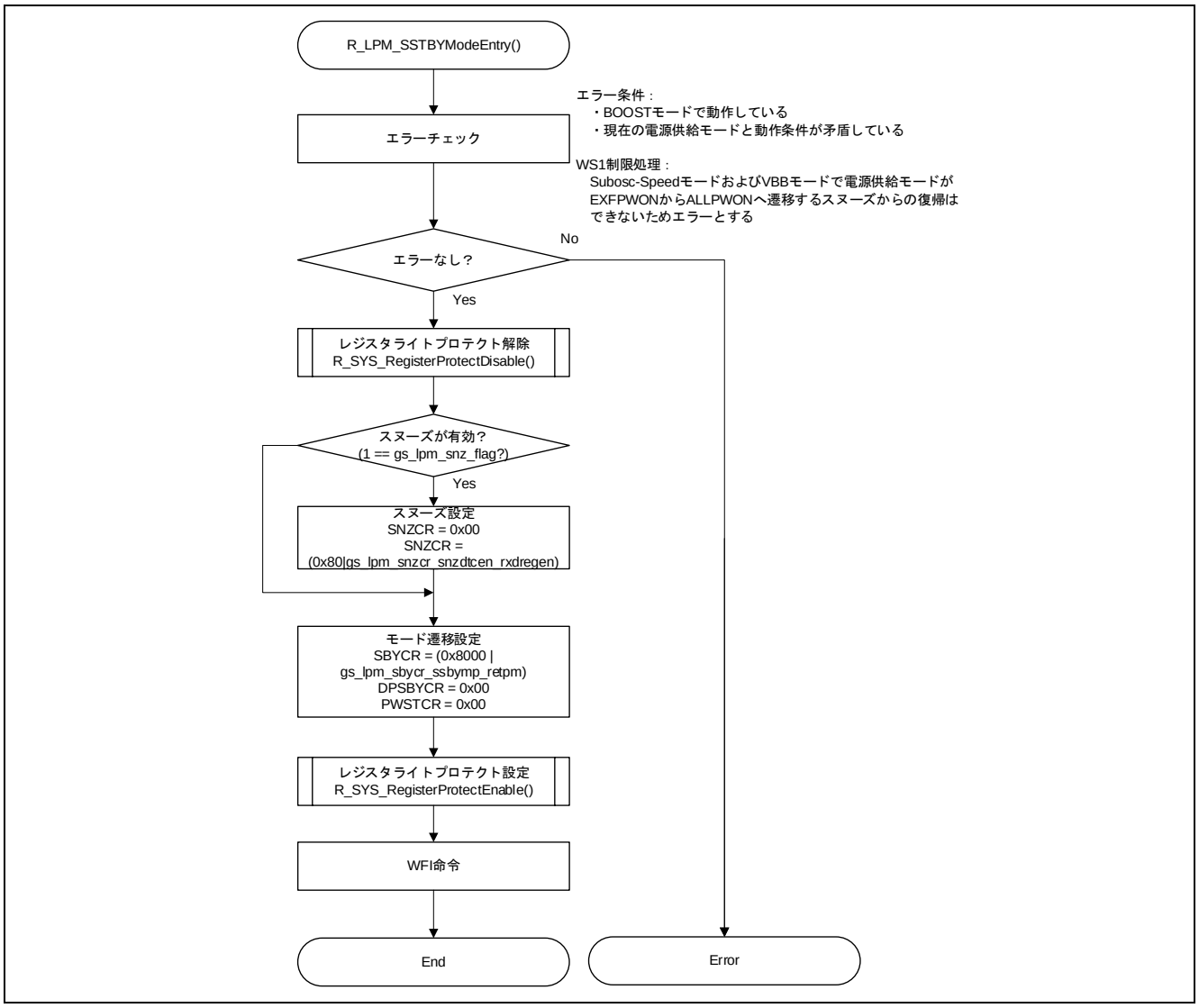

図 4-40 R_LPM_SSTBYModeEntry 関数処理フロー

(2)　RE01 256KB グループ

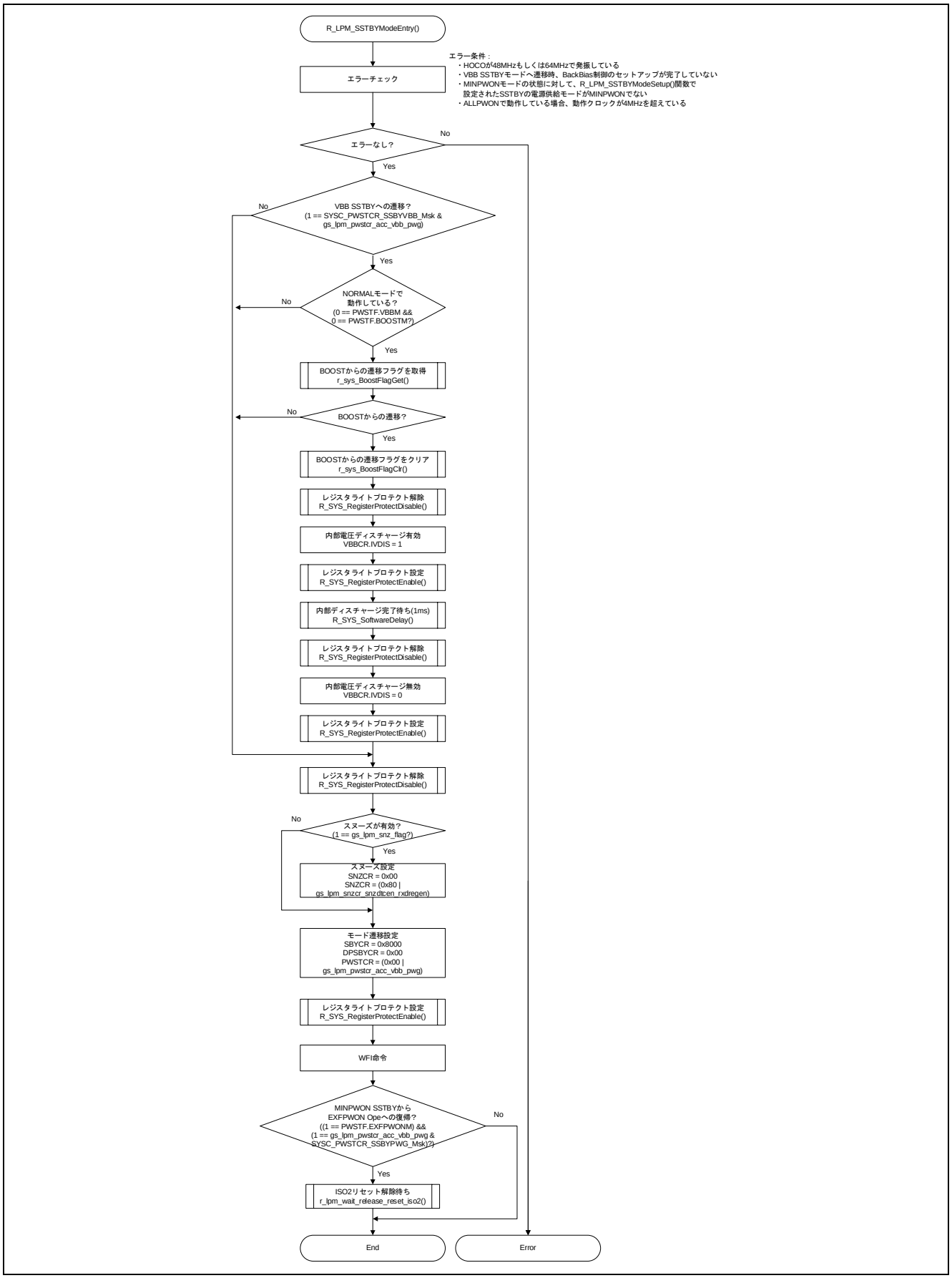

図 4-41 R_LPM_SSTBYModeEntry 関数処理フロー

4.3.21　R_LPM_DSTBYModeSetup 関数

表 4-26 R_LPM_DSTBYModeSetup 関数仕様

書式	int32_t R_LPM_DSTBYModeSetup(st_lpm_dstby_cfg_t * p_cfg)
仕様説明	DSTBY モードの動作条件を設定します　※1
引数[注]	st_lpm_dstby_cfg_t * p_cfg[入力] DSTBY モードの動作条件を格納した構造体 st_lpm_dstby_cfg_t のポインタを設定します
戻り値	正常 (0)　　：DSTBY モードの動作条件を設定しました。
	エラー (-1)　　：DSTBY モードの動作条件を設定しませんでした。 以下の状態を検出するとエラーになります ― 引数 p_cfg が NULL
備考	<RE01 256KB グループ> 引数 p_cfg->wup で設定する DSTBY モード復帰要因について、CCC インターバル割り込みと WUPT インターバル割り込みのマクロ定義は併記されます

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.6 低消費電力モードを参照ください。

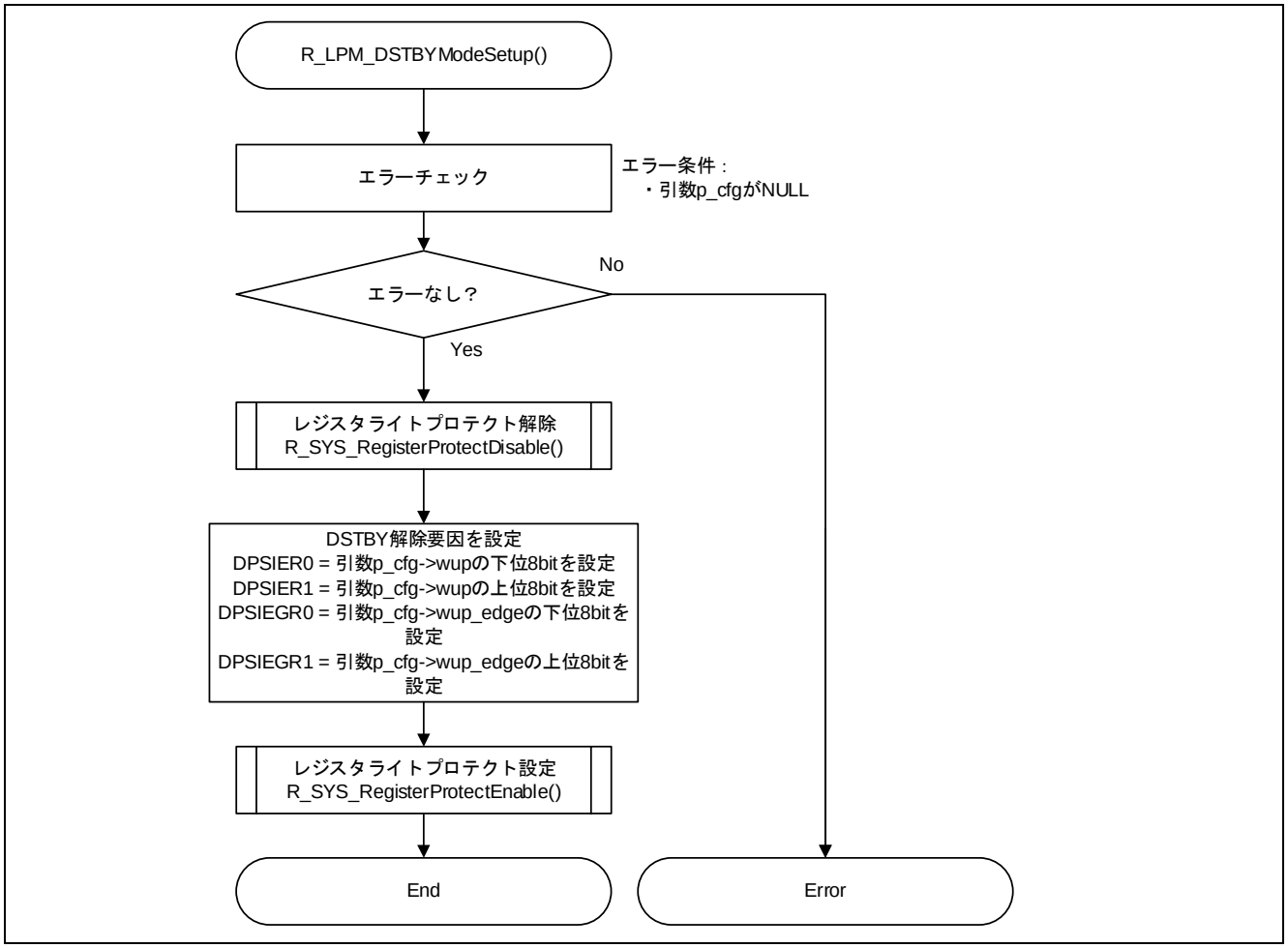

図 4-42 R_LPM_DSTBYModeSetup 関数処理フロー

4.3.22　R_LPM_DSTBYModeEntry 関数

表 4-27 R_LPM_DSTBYModeEntry 関数仕様

書式	int32_t R_LPM_DSTBYModeEntry(e_lpm_dstby_io_t io)
仕様説明	DSTBY モードに遷移します　※1 本関数内で DSTBY モードに遷移し、DSTBY モードから復帰するとリセットが解除されます 本関数を実行する前に、R_LPM_DSTBYModeSetup()関数を実行して DSTBY モードの動作条件を設定する必要があります
引数	e_lpm_dstby_io_t io[入力] DSTBY モードから復帰しリセットが解除された時の端子状態を設定します。
戻り値	エラー (-1)　：DSTBY モードへ遷移しませんでした 以下の状態を検出するとエラーになります <RE01 1500KB グループ> — BOOST モードで動作している — 引数 io が e_lpm_dstby_io_t の定義範囲を超えている <RE01 256KB グループ> — 引数 io が e_lpm_dstby_io_t の定義範囲を超えている
備考	DSTBY モードからの復帰によるリセット解除要因は、R_LPM_DSTBYResetStatusGet()関数を使用して取得できます リセット解除要因を示す DPSIFR0、DPSIFR1 レジスタのリセット後の初期化有無は、リセット要因によって異なります DPSIFR0、DPSIFR1 レジスタは、本関数で DSTBY モードに遷移する前にクリアします
	<RE01 1500KB グループ> WS1 制限事項（No.51, 52） R_LPM_DSTBYModeSetup()関数を使用して DSTBY モードからの復帰要因に CCC を設定していない、もしくは SYOCDCR.DBGEN ビットに"1"が設定されている場合は、DSTBY モードからの復帰ができないためエラーを返します WS1 制限事項（No.1） オンチップデバッグの状態では DSTBY モードへ遷移できませんが、本関数ではエラーを返しません

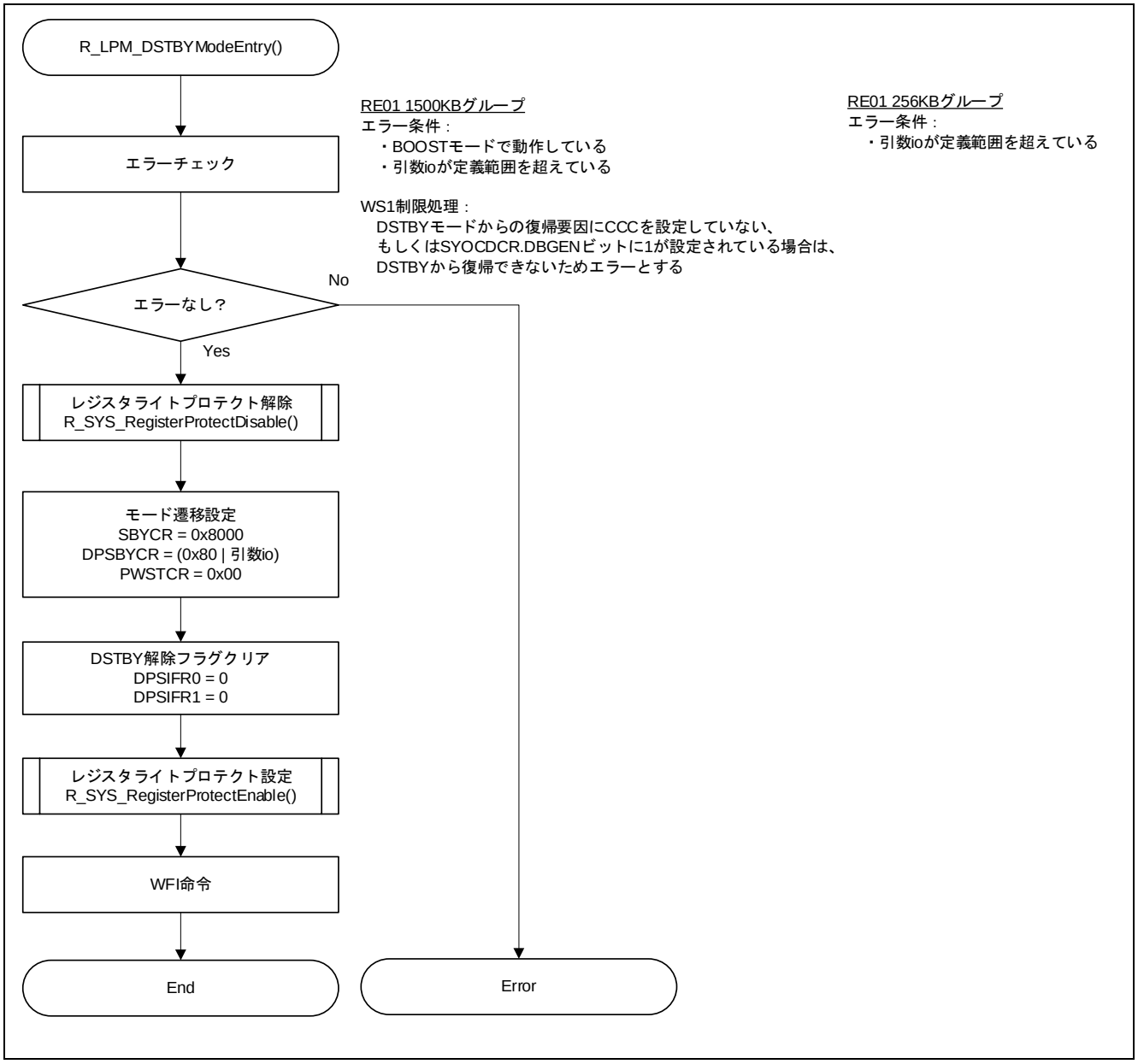

図 4-43 R_LPM_DSTBYModeEntry 関数処理フロー

4.3.23　R_LPM_DSTBYResetStatusGet 関数

表 4-28 R_LPM_DSTBYResetStatusGet 関数仕様

書式	uint32_t R_LPM_DSTBYResetStatusGet(void)
仕様説明	DSTBY モード復帰によるリセット解除要因を取得します。
引数	なし
戻り値[注]	uint32_t　：LPM_DSTBY_RESET によって定義された値を返します。 戻り値の下記ビットフィールドが 1 の場合、該当要因によって DSTBY モードから復帰しリセットが解除されました <RE01 1500KB グループ> — LPM_DSTBY_RESET_IRQ0DS：IRQ0-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ1DS：IRQ1-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ2DS：IRQ2-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ3DS：IRQ3-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_LVD1　：電圧監視 1 信号(LVD1)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_LVDBAT：電圧監視 BAT 信号(LVDBAT)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_NMI　：NMI 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_CCC　：CCC インターバルによって DSTBY モードから復帰 <RE01 256KB グループ> — LPM_DSTBY_RESET_IRQ0DS：IRQ0-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ1DS：IRQ1-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ2DS：IRQ2-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_IRQ3DS：IRQ3-DS 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_LVD1　：電圧監視 1 信号(LVD1)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_LVDBAT：電圧監視 BAT 信号(LVDBAT)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_RTCI　：RTC 周期割り込み信号(RTC_PRD)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_RTCA　：RTC アラーム割り込み信号(RTC_ALM)によって DSTBY モードから復帰 — LPM_DSTBY_RESET_NMI　：NMI 端子によって DSTBY モードから復帰 — LPM_DSTBY_RESET_CCC　：CCC インターバルによって DSTBY モードから復帰 — LPM_DSTBY_RESET_WUPT　：WUPT インターバルによって DSTBY モードから復帰
備考	<RE01 256KB グループ> 戻り値の CCC インターバルと WUPT インターバル割り込みのマクロ定義は併記されます

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.6 低消費電力モードを参照ください。

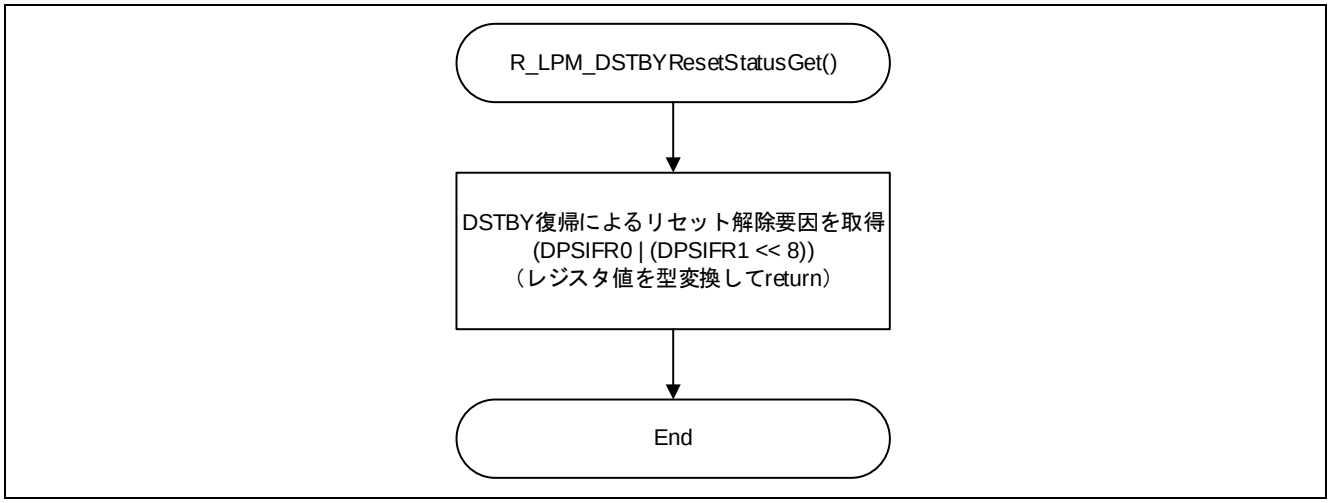

図 4-44 R_LPM_DSTBYResetStatusGet 関数処理フロー

4.3.24　R_LPM_RamRetentionSet 関数

表 4-29 R_LPM_RamRetentionSet 関数仕様

書式	void R_LPM_RamRetentionSet(uint32_t area)
仕様説明	SSTBY モード中の RAM 保持エリアを設定します　※1
引数[注]	uint32_t area[入力]　　：LPM_RAM_RETENTION_AREA で定義されます SSTBY モード中に保持する RAM エリアを設定します area には論理和で複数の RAM エリアを設定することができます
戻り値	なし
備考	本引数で保持エリアに設定すると RAM 遮断コントロールレジスタ（RAMSDCR）の該当する RAMSn ビットが 0（保持）に設定されます。本引数と RAMSDCR レジスタの値は反転しているため、ご注意ください。

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.3 RAM 遮断を参照ください。

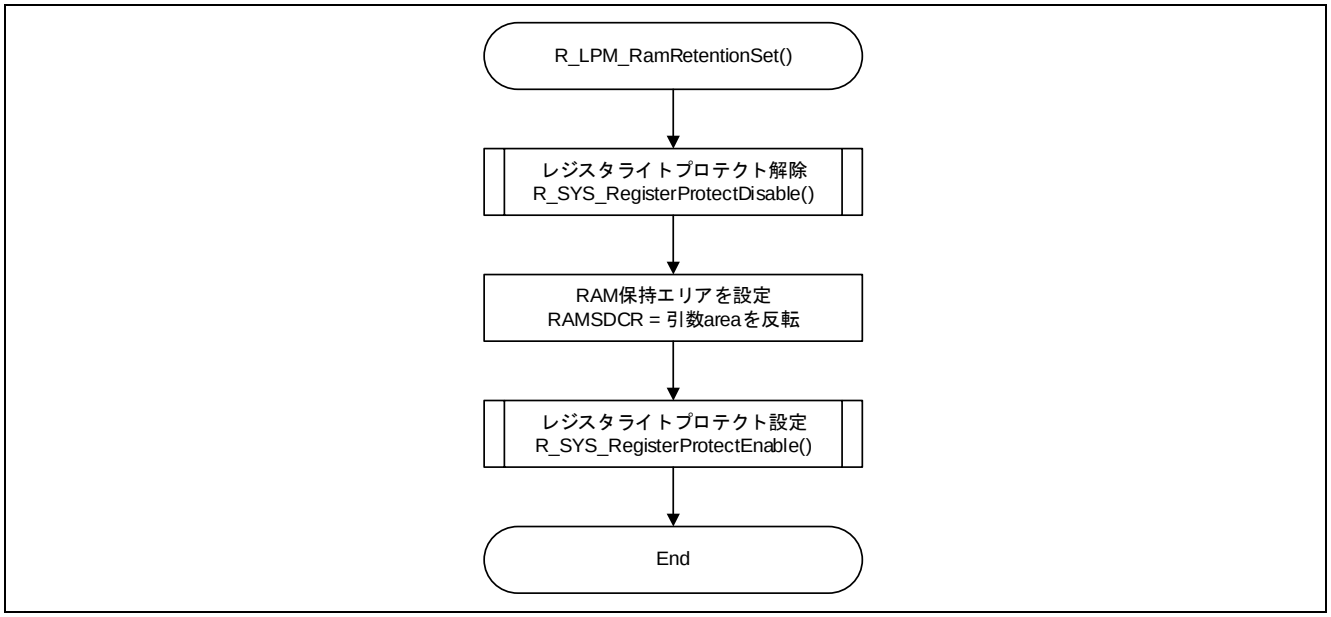

図 4-45 R_LPM_RamRetentionSet 関数処理フロー

4.3.25　R_LPM_IOPowerSupplyModeSet 関数

表 4-30 R_LPM_IOPowerSupplyModeSet 関数仕様

書式	int32_t R_LPM_IOPowerSupplyModeSet(uint8_t vocr)
仕様説明	IO 電源ドメインそれぞれにおいて電源が供給できない場合、不定信号が発生し、電源が供給されている他のドメインへ伝搬することを防ぐための機能を有効にします　※1 本関数を実行する前に、R_LPM_ModuleStop()関数もしくは R_LPM_FastModuleStop()関数を使用して、指定 IO 電源ドメインに属するモジュールをモジュールストップ状態に遷移する必要があります <RE01 1500KB グループ> AVCC0, AVCC1, IOVCC0, IOVCC1, IOVCC2, IOVCC3, USBVCC, MTDV <RE01 256KB グループ> AVCC0, IOVCC0, IOVCC1
引数[注]	uint8_t vocr[入力]　：LPM_IOPOWER_SUPPLY で定義されます 電源非供給の IO 電源ドメインを設定します vocr に設定された IO 電源ドメインで発生した不定値の他ドメインへの伝搬抑止機能を有効にします vocr は論理和で複数の電源非供給の IO 電源ドメインを設定することができます
戻り値	正常 (0)　：IO 電源ドメインの不定値伝搬抑止機能を設定しました。
	エラー (-1)　：IO 電源ドメインの不定値伝搬抑止機能を設定しませんでした。 以下の状態を検出するとエラーになります < RE01 1500KB グループ> ― 引数 vocr で指定した IO 電源ドメインに属するモジュールがモジュールストップ状態に遷移していない < RE01 256KB グループ> ― エラーなし（本関数の戻り値は常に「正常(0)」となります）
備考	<RE01 1500KB グループ> WS1 制限事項（No.58） VOCR の b7 ビット定義反転

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.4 IO 電源オープン制御を参照ください。

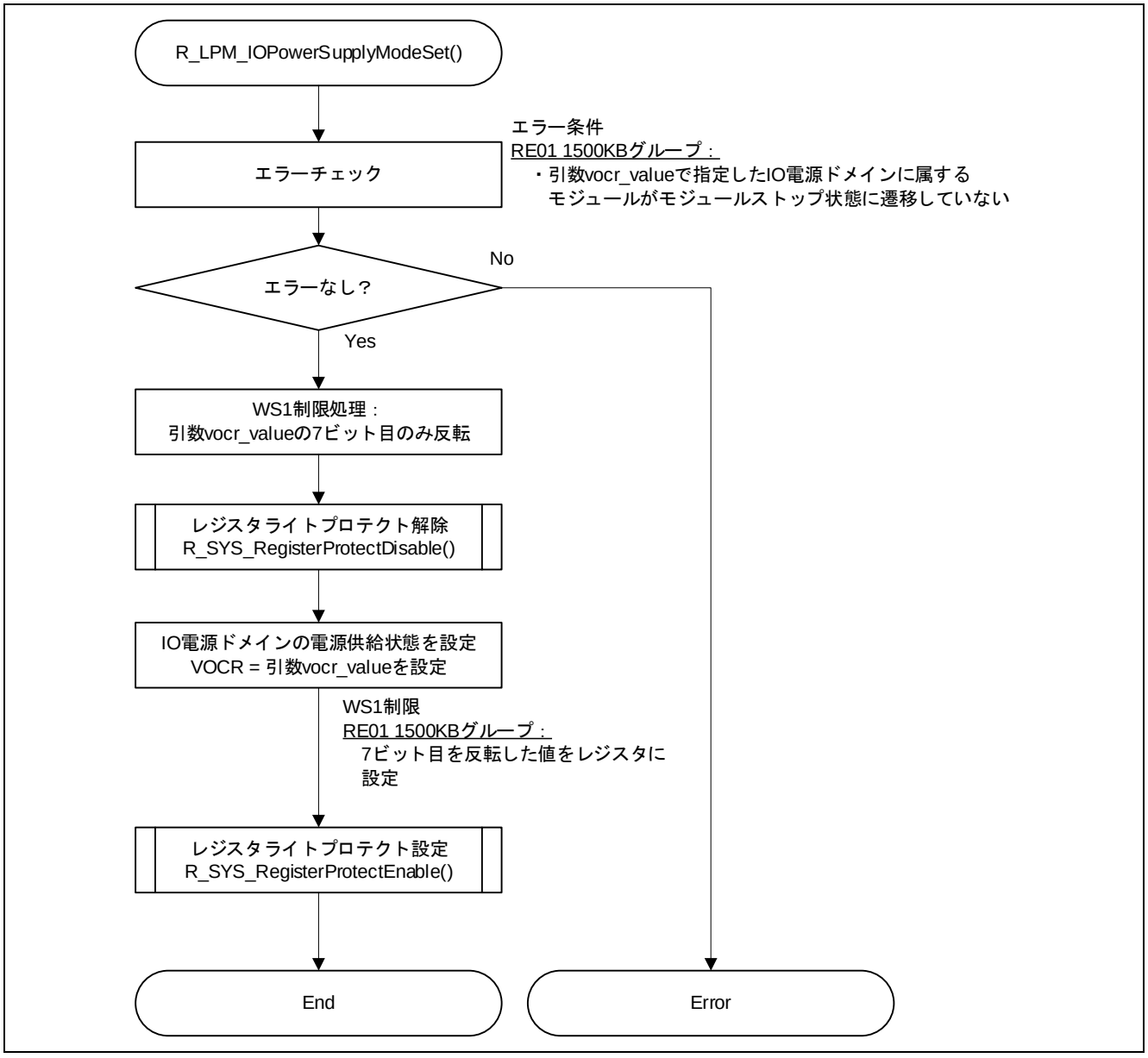

図 4-46 R_LPM_IOPowerSupplyModeSet 関数処理フロー

4.3.26　R_LPM_IOPowerSupplyModeGet 関数

表 4-31 R_LPM_IOPowerSupplyModeGet 関数仕様

書式	uint8_t R_LPM_IOPowerSupplyModeGet(void)
仕様説明	不定値伝搬抑止機能が有効な IO 電源ドメインを取得します
引数	なし
戻り値[注]	uint8_t　：LPM_IOPOWER_SUPPLY で定義された値を返します 戻り値の下記ビットフィールドが 1 の場合、該当 IO 電源ドメインの不定値伝搬抑止機能は有効です <RE01 1500KB グループ> — LPM_IOPOWER_SUPPLY_AVCC0　：S14AD, TEMPS, VREF — LPM_IOPOWER_SUPPLY_AVCC1　：R12DA, ACMP — LPM_IOPOWER_SUPPLY_IOVCC0：PORT8 — LPM_IOPOWER_SUPPLY_IOVCC1：PORT3, PORT6, PORT7, P202-P204 — LPM_IOPOWER_SUPPLY_IOVCC2：PORT1 — LPM_IOPOWER_SUPPLY_IOVCC3：P010-P015, PORT5 — LPM_IOPOWER_SUPPLY_USBVCC：USB — LPM_IOPOWER_SUPPLY_MTDV　：MTDV <RE01 256KB グループ> — LPM_IOPOWER_SUPPLY_AVCC0　：P000-P007 — LPM_IOPOWER_SUPPLY_IOVCC0：PORT8, P010-P015 — LPM_IOPOWER_SUPPLY_IOVCC1：PORT1, PORT3, PORT5, PORT6, PORT7, P202-P205
備考	<RE01 1500KB グループ> WS1 制限事項（No. 58） VOCR の b7 ビット定義反転

【注】 RE01 1500KB グループと RE01 256KB グループで使用する定義に差異があります。
詳細は 4.2.4 IO 電源オープン制御を参照ください。

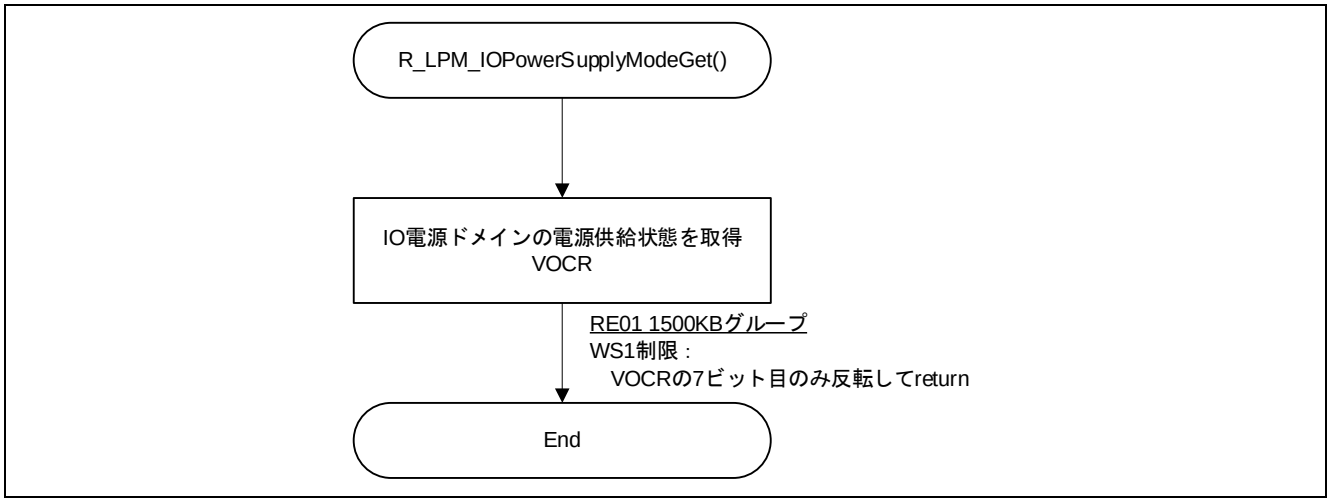

図 4-47 R_LPM_IOPowerSupplyModeGet 関数処理フロー

4.4 割り込みコントローラへの設定

R_LPM ドライバは、UMH の 12 章 消費電力低減機能に該当するレジスタのほかに、ソフトウェアスタンバイおよびスヌーズからの復帰イベント設定のため、UMH の 16 章 割り込みコントローラユニットに該当する一部レジスタへの設定を行います。R_LPM ドライバが設定を行う割り込みコントローラユニットのレジスタを表 4-32 に示します。

表 4-32 割り込みコントローラレジスタへの設定

レジスタ名	内容
WUPEN	各割り込みに対して SSTBY モードからの復帰許可/禁止を設定します
SELSR0	スヌーズからの復帰イベントを設定します

5. 製品グループ間のプログラム移行

1500KB グループと 256KB グループのドライバ関数の I/F 仕様の差異と、ユーザプログラムを製品グループ移行する際の注意事項について説明します。モード遷移例については、開発スタートアップガイドを参照してください。

5.1 1500KB グループ、256KB グループの I/F 仕様差異

表 5-1 に 1500KB グループと 256KB グループで I/F 仕様に差異がある関数を示します。

各グループのマクロ/型定義の詳細は 4.2 マクロ／型定義、エラー条件については 4.3 関数仕様を参照してください。

グループ間で関数の互換性がない関数は、必ずプログラムを変更する必要があります。

表 5-1 各関数のグループ間における I/F 仕様差異

番号	関数名	ハードウェア仕様差異によるマクロ/型定義の差異	関数のエラー条件差異	関数の互換性無し
1	R_LPM_GetVersion			
2	R_LPM_Initialize			
3	R_LPM_ModuleStart	✔注	✔	
4	R_LPM_ModuleStop	✔注		
5	R_LPM_FastModuleStart	✔注	✔	
6	R_LPM_FastModuleStop	✔注		
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	✔注	✔	✔
20	R_LPM_SSTBYModeEntry		✔	
21	R_LPM_DSTBYModeSetup	✔注		
22	R_LPM_DSTBYModeEntry		✔	
23	R_LPM_DSTBYResetStatusGet	✔注		
24	R_LPM_RamRetentionSet	✔注		
25	R_LPM_IOPowerSupplyModeSet	✔注	✔	
26	R_LPM_IOPowerSupplyModeGet	✔注		

注：ハードウェア搭載機能差の他に、機能差による I/F 仕様差異があります。
例）RTC は 1500KB グループでは常にモジュールストップ解除状態ですが、256KB グループではモジュールストップ状態となります

5.2　製品グループ間でプログラム移行する際の注意事項

表 5-1 で示した I/F 仕様差異のある関数について、1500KB グループと 256KB グループでユーザプログラムを移行する際の注意点と、プログラム移行が正しく出来ていない場合の現象を示します。

(1) R_LPM_ModuleStart, R_LPM_FastModuleStart：RTC, WDT, IWDT モジュールを使用する場合
R_LPM_ModuleStop, R_LPM_FastModuleStop

RE01 1500KB グループ	RE01 256KB グループ
初期状態でモジュールが動作しているため、 対象モジュールを設定するとコンパイルエラーが発生する	初期状態でモジュールが停止しているため、 対象モジュールを動作設定する必要がある

(2) R_LPM_BackBiasModeEntry：BOOST から VBB に遷移する場合

RE01 1500KB グループ	RE01 256KB グループ
BOOST から VBB に遷移できないため、 R_LPM_BackBiasModeEntry()関数からエラーが返る SYS ドライバ関数を使用して、NORMAL モードの Subosc-Speed モードに遷移してから R_LPM_BackBiasModeEntry()関数を実行する必要がある	BOOST から VBB に遷移可能

(3) R_LPM_SnoozeSet：スヌーズ遷移要因に AGT0CA, AGT1CA を設定している場合

RE01 1500KB グループ	RE01 256KB グループ
AGT0CA, AGT1CA が指定される	AGT0CA と AGTW0CA, AGT1CA と AGTW1CA が指定される

(4) R_LPM_SSTBYModeSetup：SSTBY 設定で使用する構造に違いあり

RE01 1500KB グループ	RE01 256KB グループ
遷移元と遷移先、復帰先の電源供給モード、SSTBY からの復帰要因を引数で指定する （ope_sstby, sstby_ope, wup） 256KB グループの構造体を使用した場合は コンパイルエラーが発生する （メンバ名が異なるため）	復帰先と遷移元の電源供給モードは同じであるため指定不要 遷移先の電源供給モード、VBB あり/なし、時短あり/なし、SSTBY からの復帰要因を引数で指定する （ope_sstby, speed, wup） 1500KB グループの構造体を使用した場合は コンパイルエラーが発生する （メンバ名が異なるため）

(5) R_LPM_SSTBYModeEntry：BOOST から SSTBY に遷移する場合

RE01 1500KB グループ	RE01 256KB グループ
BOOST から SSTBY に遷移できないため、 R_LPM_SSTBYModeEntry()関数からエラーが返る SYS ドライバ関数を使用して NORMAL モードに遷移してから R_LPM_SSTBYModeEntry()関数を実行する必要がある	BOOST から SSTBY に遷移可能

(6) R_LPM_DSTBYModeSetup, R_LPM_DSTBYResetStatusGet：
DSTBY からの復帰要因に RTCI, RTCA を設定している場合

RE01 1500KB グループ	RE01 256KB グループ
RTCI, RTCA は対象外のためコンパイルエラーが発生する	RTCI, RTCA が指定される

DSTBY からの復帰要因に CCC を設定している場合

RE01 1500KB グループ	RE01 256KB グループ
CCC の周期割り込みが指定される	CCC の周期割り込みと WUPT 割り込みが指定される

(7) R_LPM_DSTBYModeEntry：BOOST から DSTBY に遷移する場合

RE01 1500KB グループ	RE01 256KB グループ
BOOST から DSTBY に遷移できないため、R_LPM_DSTBYModeEntry()関数からエラーが返る SYS ドライバ関数を使用して NORMAL モードに遷移してから R_LPM_DSTBYModeEntry()関数を実行する必要がある	BOOST から DSTBY に遷移可能

(8) R_LPM_RamRetentionSet：
RAM 遮断領域に AREA4～AREA7（0x20020000H～0x2003FFFFH）を設定する場合

RE01 1500KB グループ	RE01 256KB グループ
設定エリアが指定される	設定エリアは対象外のため、コンパイルエラーが発生する

(9) R_LPM_IOPowerSupplyModeSet, R_LPM_IOPowerSupplyModeGet：
IO 電源供給領域に差異あり（一部コンパイルエラーが発生）

端子・モジュール	電源ドメイン	
	RE01 1500KB グループ	RE01 256KB グループ
S14AD（P000～P007） TEMPS, VREF	AVCC0	AVCC0
R12DA, ACMP	AVCC1	－
P010～P015	IOVCC3	IOVCC0
ポート 1	IOVCC2	IOVCC1
P202～P204, P205	IOVCC1	IOVCC1
ポート 3	IOVCC1	IOVCC1
ポート 5	IOVCC3	IOVCC1
ポート 6, ポート 7	IOVCC1	IOVCC1
ポート 8	IOVCC0	IOVCC0
USB	USBVCC	－
MTDV	VPM	－

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.0	2018.12.21	ー	初版
1.1	2019.3.4	11	4.1.4 表 4-5 R_LPM_GetVersion 関数を追加
		12	4.2.1 バージョン取得 項を追加
		18	4.3.1 R_LPM_GetVersion 項を追加
0.72	2019.6.26	ー	R_LPM ドライバ バージョン 0.72 に対応
		ー	タイトルおよびファイル名を変更 ドキュメント版数を R_LPM ドライババージョンに合わせ変更
		4	内部ディスチャージ対応に伴い R_SYSTEM ドライバの内部関数呼び出しを追加
		13-18	型定義の図フォーマットを変更
		12	バージョン番号を更新
		13	モジュールストップ機能に DIL 追加
		15	スヌーズ要求 AGT1CB を AGT0CA に変更 スヌーズ復帰要因から KEY_INTKR を削除
		16	ソフトウェアスタンバイ復帰要因に IRQ8, IRQ9, LVDBAT を追加 ソフトウェアスタンバイ復帰要因 AGT1CB を AGT0CA に変更
		17	ディープソフトウェアスタンバイ復帰要因および復帰要因エッジに LVDBAT を追加
		17,47	ディープソフトウェアスタンバイのリセットステータスに LVDBAT を追加
		29,30	R_LPM_BackBiasModeEntry 関数に内部ディスチャージ処理を追加
		48	R_LPM_RamRetentionSet 関数の仕様説明を変更 ただし仕様変更はなし
		49	R_LPM_IOPowerSupplyModeSet 関数の仕様説明を変更 ただし仕様変更はなし
		51	R_LPM_IOPowerSupplyModeGet 関数の仕様説明を変更 ただし仕様変更はなし
1.00	2019/7/31	—	R_LPM ドライババージョン 1.00 に対応
		—	シリーズ名の決定に伴う文書タイトルおよびファイル名の変更 文書内のシリーズ名、グループ名の変更
		—	誤記修正
		3	関連文書一覧に、CMSIS パッケージを用いた開発スタートアップガイドを追加
		5	シリーズ名の決定に伴うファイル構成の変更
		33,34	R_LPM_PowerSupplyModeAllpwonSet 関数に ISO2 リセット解除待ちを追加
		35,36	R_LPM_PowerSupplyModeExfpwonSet 関数に ISO2 リセット解除待ちを追加
	2019/9/19	1,5	パッケージ名を RE01 1500KB グループ CMSIS Driver Package に変更
		—	末尾の注意書きを最新版に差し替え
		12	バージョン定義例で使用する番号を、メジャーバージョンとマイナーバージョンを含む数字に変更
1.01	2019/12/20	全体	256KB グループの R_LPM ドライバ仕様を追加
		65	5.1 製品グループ間のプログラム移行 を追加

Rev.	発行日	改訂内容	
		ページ	ポイント
1.02	2020/3/4	3	表 1-1 1500KB UMH の略号を「1500KB グループ UMH」に変更 256KB UMH を追加
			表 1-2 256KB UMH Rev0.50 入稿原稿→UMH0.80 ドキュメント名に変更 脚注を削除 スタートアップガイドの文書名を 256KB グループが追加された最新版に変更
		7	3.2 「ALLPWON モードから MINPWON モードへの遷移、および MINPWON モードから ALLPWON モードへの直接遷移」 →「ALLPWON モードと MINPWON モード間の直接遷移」
		53	4.3.20「HOCO が 48MHz もしくは 64MHz で発振している」→「HOCO 発振周波数が 48MHz もしくは 64MHz に設定されている」
		63	4.3.26 戻り値欄 256KB グループ UMH(Rev.0.80)の記載変更に伴い、256KB グループ LPM_IOPOWER_SUPPLY_AVCC0 から「ADC140, TEMPS, VREF」を削除
1.03	2020/5/27	26-28, 30, 49, 52, 57, 60, 62, 63, 65	表 4-8～表 4-11, 表 4-22, 表 4-24, 表 4-26, 表 4-28～表 4-31, 1500KB グループ, 256KB グループとで引数の定義が異なるため、4.2 マクロ／型定義を参照することの注意書きを追加
		26, 28, 35, 38, 40, 45, 52, 54, 57, 58, 60, 63, 65	製品グループの仕様差異は、注釈から本文記載に変更 表 4-8, 表 4-10, 表 4-15, 表 4-16, 表 4-18, 表 4-20, 表 4-24, 表 4-25, 表 4-26, 表 4-27, 表 4-28, 表 4-30, 表 4-31
		68, 69	5.2 以下の関数の仕様差異を追加し、記載方法を全面変更 R_LPM_SnoozeSet R_LPM_DSTBYModeSet R_LPM_DSTBYModeSet R_LPM_SSTBYModeSetup R_LPM_SSTBYModeEntry
		—	「5.2.1 モード遷移例」削除 開発スタートアップガイドに記載移行するため

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がV_{IL}（Max.）からV_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注1.　本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注2.　本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.4.0-1 2017.11)

RENESAS

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/