

# TRAVEO™ T2G ファミリ: 外部電源設計ガイド

## 本書について

### 適用範囲と目的

このアプリケーションノートは、TRAVEO™ T2G ファミリ CYT3/4/6 シリーズの外部電源設計の手順を説明します。

### 対象者

本書は TRAVEO™ T2G ファミリ CYT3/4/6 シリーズを使用するすべての人を対象とします。

### 関連製品ファミリ

CYT3/CYT4/CYT6 シリーズ

## 目次

## 目次

	本書について .....	1
	目次 .....	2
<b>1</b>	はじめに .....	4
<b>2</b>	<b>TRAVERO™ T2G 電源システム</b> .....	5
2.1	ブロックダイアグラム .....	5
2.1.1	内部レギュレータと REGHC および PMIC コントローラ .....	6
2.2	CYT3B/4B/6B シリーズと CYT3D/4D シリーズの違い .....	6
2.2.1	外部電源構成の違い .....	6
2.2.2	レジスタ名の違い .....	7
2.3	外部電源と内部レギュレータ間のハンドオーバ .....	7
2.3.1	PMIC 使用時の内部レギュレータ設定 .....	8
2.3.2	システムコール API .....	9
2.3.3	ハンドオーバシーケンスの概要 .....	18
2.3.4	SwitchOverRegulators API の Non-blocking call 処理 .....	19
2.3.4.1	REGHC_SEQ_BUSY フラグの動作 .....	20
2.3.4.2	DeepSleep レギュレータの構成 .....	20
2.3.4.3	Non-blocking call でのハンドオーバ処理例 .....	21
2.3.5	各ユースケースでの LoadRegulatorTrims 処理 .....	22
2.4	外部電源の設定 .....	24
<b>3</b>	<b>パストランジスタ</b> .....	26
3.1	ハードウェア構成 .....	26
3.2	ハンドオーバタイミングチャート .....	26
3.3	ソフトウェアフロー .....	27
3.3.1	内部レギュレータからパストランジスタへのハンドオーバ .....	27
3.3.2	DeepSleep への遷移と復帰 .....	29
3.3.3	パストランジスタから内部レギュレータへのハンドオーバ .....	31
3.4	部品選定 .....	31
3.5	レイアウト設計ガイドライン .....	33
<b>4</b>	<b>PMIC (スイッチングレギュレータ)</b> .....	34
4.1	PMIC 仕様要件 .....	35
4.2	推奨する PMIC トポロジ .....	37
4.3	PMIC 直接接続 .....	38
4.3.1	ハードウェア構成 .....	39
4.3.2	ケース 1 .....	40
4.3.2.1	ハンドオーバタイミングチャート .....	40
4.3.2.2	ソフトウェアフロー .....	41
4.3.3	ケース 2 .....	44

## 目次

4.3.3.1	ハンドオーバタイミングチャート	44
4.3.3.2	ソフトウェアフロー	45
4.3.4	ケース 3	49
4.3.4.1	ハンドオーバタイミングチャート	49
4.3.4.2	ソフトウェアフロー	51
4.3.5	ケース 4	53
4.3.5.1	ハンドオーバタイミングチャート	53
4.3.5.2	ソフトウェアフロー	54
4.4	ロードスイッチを使用した PMIC	57
4.4.1	ケース 1	58
4.4.1.1	ハードウェア構成	58
4.4.1.2	ハンドオーバタイミングチャート	59
4.4.1.3	ソフトウェアフロー	60
4.4.2	ケース 2	63
4.4.2.1	ハードウェア構成	63
4.4.2.2	PMIC のハンドオーバタイミングチャート	65
4.4.2.3	ソフトウェアフロー	66
4.4.3	ケース 3	68
4.4.3.1	ソフトウェアフロー	70
4.5	部品選定	74
4.5.1	PMIC の周辺部品	74
4.5.2	ロードスイッチ	74
4.6	レイアウト設計ガイドライン	75
5	<b>Appendix A. ハードウェアシーケンス</b>	76
	用語集	78
	関連ドキュメント	79
	改訂履歴	80
	免責事項	81

## 1 はじめに

### 1 はじめに

このドキュメントは、TRAVEO™ T2G MCU ファミリ CYT3, CYT4, および CYT6 シリーズの外部電源を設計する手順について説明します。CYT3B/4B/6B シリーズは、ハイエンドボディコントロールユニットなどの自動車システムを対象とした TRAVEO™ T2G MCU ボディコントローラファミリです。CYT3D および CYT4D シリーズは、インストルメントクラスターなどの自動車システムを対象とした TRAVEO™ T2G MCU クラスター 2D ファミリです。

これらの MCU には 2 つの内部レギュレータ (Active レギュレータ、DeepSleep レギュレータ)があります。加えて、CYT3B/CYT4B/CYT6B シリーズには高電流レギュレータコントローラ (REGHC)、そして CYT3D/CYT4D シリーズには PMIC コントローラがあります。REGHC または PMIC コントローラは、外部電源システム制御に使用されます。このドキュメントでは、REGHC または PMIC コントローラを使用する外部電源システム設計での推奨事項と制限事項について説明します。また、さまざまなユースケースでの内部レギュレータと外部電源間のハンドオーバー手順についても説明します。

各シリーズでサポートされる外部電源構成については、[表 1](#) を参照してください。

デバイスの機能と関連設定の詳細は、TRAVEO™ T2G アーキテクチャテクニカルリファレンスマニュアル(TRM)[\[2\]](#)、および専用のデバイスデータシート[\[1\]](#)を参照してください。

## 2 TRAVEO™ T2G 電源システム

### 2 TRAVEO™ T2G 電源システム

以下に TRAVEO™ T2G 電源システム機能について示します。

- 2.7 V～5.5 V の  $V_{DD}$  電源電圧範囲
- コア電源 <sup>1)</sup>  $V_{CCD}$
- 複数のオンチップレギュレータ
  - 低消費電流時に Active または Sleep 電源モードで MCU に電源供給する Active レギュレータ
  - DeepSleep 電源モード時に周辺機能に電源供給する DeepSleep レギュレータ。PMIC は、DeepSleep 電源モードで電源供給するよう設定できます。その場合、DeepSleep レギュレータを無効にできます。
- REGHC。パストランジスタを使用するか PMIC を制御することで、Active および Sleep 電源モードでより高い負荷電流をサポートします。
- PMIC コントローラ。PMIC を制御することで、Active および Sleep 電源モードでより高い負荷電流をサポートします。

#### 2.1 ブロックダイアグラム

TRAVEO™ T2G には、2 つの内部レギュレータ (Active レギュレータ、DeepSleep レギュレータ) があります。また、外部電源用の REGHC または PMIC コントローラがあります。REGHC または PMIC コントローラは、DRV\_VOUT および EXT\_PS\_CTL[0:2] の 4 つの端子を介して、 $V_{CCD}$  を制御します。図 1 に、TRAVEO™ T2G デバイスの電源システムを示します。

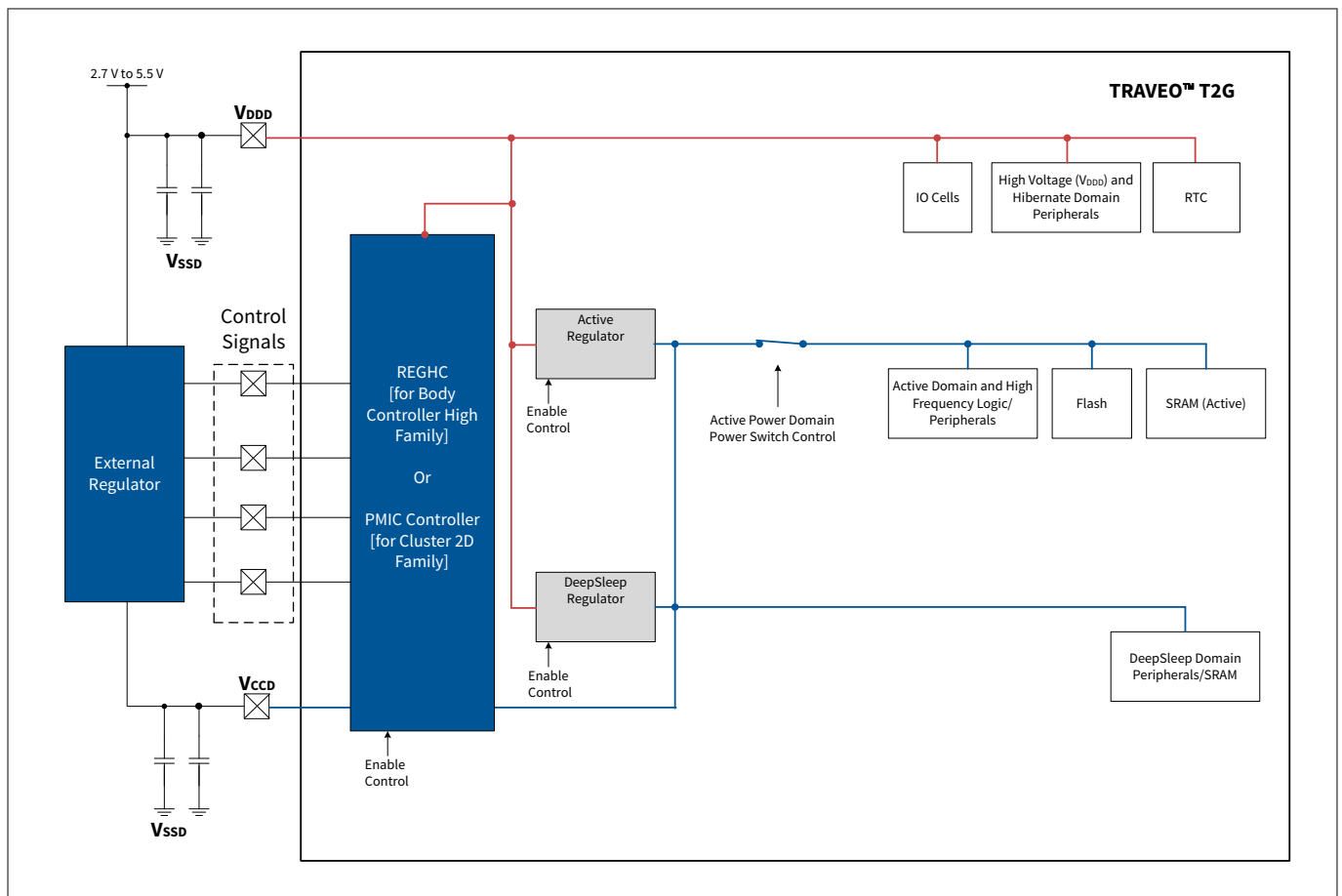


図 1 TRAVEO™ T2G 電源システム

<sup>1</sup> 指定されるコア電源電圧についてはデータシートを参照してください。

## 2 TRAVEO™ T2G 電源システム

### 2.1.1 内部レギュレータと REGHC および PMIC コントローラ

$V_{CCD}$  はコア電源に接続されます。ただし、Active レギュレータの負荷電流が 300 mA を超える場合、REGHC または PMIC コントローラを介して外部から  $V_{CCD}$  を供給する必要があります。Hibernate モードでは、すべてのレギュレータがオフになり、 $V_{CCD}$  は駆動されないことに注意してください。Hibernate 回路は直接  $V_{DD}$  から駆動します。詳細については、アーキテクチャ TRM [2] の Device Power Modes 章を参照してください。デバイスは、外部電源へのハンドオーバー前に Active レギュレータから開始します。

#### Active レギュレータ

Active または Sleep 電源モードで MCU に電源供給するリニア低ドロップアウト (LDO) レギュレータです。このレギュレータは、Active または Sleep 電源モード中に  $V_{DD}$  からコア電圧 ( $V_{CCD}$ ) を生成します。最大 300 mA の負荷電流をサポートし、電源オンおよび高電圧 (HV) リセット (XRES, POR, BOD, OVD, OCD, HIBERNATE ウェイクアップ) でのデバイス起動時、または外部電源から内部レギュレータへのハンドオーバーで動作します。負荷電流が 300 mA 未満の場合、Active レギュレータは低速動作で MCU の動作に対応できます。PMIC が有効な場合、レジスタ設定によってこのレギュレータを有効または無効にできます。Active レギュレータが有効な場合、Active レギュレータ内の過電流検出 (OCD) 機能を使用できます。

#### DeepSleep レギュレータ

DeepSleep 電源モード中にコア電圧 ( $V_{CCD}$ ) に電源供給するリニア LDO レギュレータです。Active レギュレータと比較して駆動能力は低く (最大 18 mA)、消費電流も低いです。DeepSleep 電源モード中の電源を DeepSleep レギュレータまたは PMIC より選択できます。PMIC は、DeepSleep 電源モード中に有効にできます。

#### REGHC または PMIC コントローラ

MCU は、REGHC または PMIC コントローラと外部コンポーネントを使用して、より高い負荷電流をサポートします。REGHC または PMIC コントローラは、外部電源と内部レギュレータ間のハンドオーバーを制御します。REGHC または PMIC コントローラの設定および制御ピンは、HV リセット (XRES, POR, BOD, OVD, OCD, HIBERNATE ウェイクアップ) によってのみ初期化され、低電圧 (LV) リセットでは初期化されません。標準の GPIO ピンが外部電源制御に使用される場合、それらは LV リセットによって初期化される場合があり、再設定する必要があることに注意してください。詳細については、アーキテクチャ TRM [2] を参照してください。

## 2.2 CYT3B/4B/6B シリーズと CYT3D/4D シリーズの違い

ここでは、CYT3B/4B/6B (ボディコントローラハイ) シリーズと CYT3D/4D (クラスタ 2D) シリーズの違いについて説明します。

### 2.2.1 外部電源構成の違い

前述のように、外部電源制御用に CYT3B/4B/6B シリーズは REGHC、CYT3D/4D シリーズは PMIC コントローラがあります。表 1 に、CYT3B/4B/6B シリーズと CYT3D/4D シリーズの違いを示します。

表 1 外部電源構成の差分

項	CYT3B/4B/6B シリーズ	CYT3D/4D シリーズ
コントローラ	REGHC	PMIC コントローラ
サポートする外部電源構成	パストランジスタ PMIC 直接接続 ロードスイッチを使用した PMIC	PMIC 直接接続
制御ピン	表 3 参照	注参照

注: CYT6B シリーズは PMIC のみに設定できます。

## 2 TRAVEO™ T2G 電源システム

### 2.2.2 レジスタ名の違い

CYT3B/4B/6B と CYT3D/4D シリーズでは、レジスタ名が異なります。表 2 に、各シリーズのレジスタを比較します。本書では、CYT3B/4B/6B シリーズの表記を使用します。CYT3D/4D シリーズを使用する際は、表 2 に従って読み替えてください。レジスタの詳細はレジスタ TRM [3] を参照してください。

表 2 レジスタ差分

CYT3B/4B/6B シリーズ		CYT3D/4D シリーズ	
レジスタ名	ビットフィールド名	レジスタ名	ビットフィールド名
PWR_CTL2	DPSLP_REG_DIS	PWR_CTL2	DPSLP_REG_DIS
PWR_REGHC_CTL	REGHC_CONFIGURED	PWR_PMIC_CTL	PMIC_CONFIGURED
	REGHC_PMIC_STATUS_WAIT		PMIC_STATUS_WAIT
	REGHC_PMIC_STATUS_POLARITY		PMIC_STATUS_POLARITY
	REGHC_PMIC_STATUS_INEN		PMIC_STATUS_INEN
	REGHC_PMIC_CTL_POLARITY		PMIC_CTL_POLARITY
	REGHC_PMIC_CTL_OUTEN		PMIC_CTL_OUTEN
	REGHC_PMIC_RADJ		非対応
	REGHC_PMIC_USE_RADJ		非対応
	REGHC_PMIC_USE_LINREG		PMIC_USE_LINREG
	REGHC_VADJ		PMIC_VADJ
	REGHC_PMIC_DRV_VOUT		PMIC_VREF
	REGHC_MODE		非対応
PWR_REGHC_STATUS	REGHC_SEQ_BUSY	PWR_PMIC_STATUS	PMIC_SEQ_BUSY
	REGHC_PMIC_STATUS_OK		PMIC_STATUS_OK
	REGHC_ENABLED		PMIC_ENABLED
PWR_REGHC_CTL2	REGHC_EN	PWR_PMIC_CTL2	PMIC_EN
PWR_REGHC_CTL4	REGHC_PMIC_DPSLP	PWR_PMIC_CTL4	PMIC_DPSLP
	REGHC_PMIC_VADJ_DIS		PMIC_VADJ_DIS

### 2.3 外部電源と内部レギュレータ間のハンドオーバー

ここでは、外部電源と内部レギュレータ間のハンドオーバーの概要について説明します。詳細については、[パストランジスタ](#)および [PMIC \(スイッチングレギュレータ\)](#)を参照してください。

REGHC または PMIC コントローラは、初期状態では無効で、API によって有効にする必要があります。Active レギュレータは、REGHC または PMIC コントローラが設定され、有効になり動作準備できるまでコア電流をサポートします。コア電流はハンドオーバーが完了するまで Active レギュレータの制限を超えてはいけません。API については、[システムコール API](#) を参照してください。

REGHC には、高負荷電流を駆動するため、パストランジスタ向けと PMIC 向けの 2 つの設定があります。PMIC コントローラは PMIC 設定のみです。

表 3 に、各設定での REGHC ピンを示します。注に、PMIC コントローラピンを示します。

## 2 TRAVEO™ T2G 電源システム

表 3 REGHC ピン

ピン名	パストランジスタ		PMIC	
	入出力	説明	入出力	説明
DRV_VOUT	出力	パストランジスタのベース	未使用	-
EXT_PS_CTL0	入力	電流検出抵抗のプラス側端子	入力	PMIC からのリセット出力 (RO) またはパワーグッド (PG) 信号入力
EXT_PS_CTL1	入力	電流検出抵抗のマイナス側端子	出力	PMIC イネーブル出力
EXT_PS_CTL2	未使用	-	出力 (オプション)	一部 PMIC のリセット出力調整

**注:** CYT6B シリーズは PMIC 専用には設定されています。また、CYT6B シリーズには DRV\_VOUT ピンがありません。

表 4 PMIC コントローラピン

ピン名	PMIC		
	入出力	説明	REGHC ピンとの関係
PMIC_STATUS	入力	PMIC からのリセット出力 (RO) またはパワーグッド (PG) 信号入力	EXT_PS_CTL0
PMIC_EN	出力	PMIC イネーブル出力	EXT_PS_CTL1

**注:** CYT3D/4D シリーズには、DRV\_VOUT ピンおよび EXT\_PS\_CTL2 ピンはありません。

以下の場合に、外部電源と内部レギュレータ間のハンドオーバーが行われます。

- 専用の API とレジスタを使用し、ユーザソフトウェアが内部レギュレータと外部電源を切り替えます。
- OFF または XRES 状態および HIBERNATE への遷移によってハードウェアにより REGHC または PMIC コントローラは無効になります。リセットが解除されると、MCU は Active レギュレータで動作を開始します。ソフトウェアは、デバイス起動後に再度 REGHC または PMIC コントローラを有効にします。OFF および XRES 状態については、アーキテクチャ TRM [2] の Device Power Modes 章を参照してください。

### 2.3.1 PMIC 使用時の内部レギュレータ設定

REGHC または PMIC コントローラを PMIC に設定する場合、Active および DeepSleep レギュレータを有効または無効にできます。

PMIC と Active レギュレータの両方を同時に有効にした場合、PMIC から電源供給時に、Active レギュレータの OCD 機能を使用できます。OCD は、Active レギュレータの一部です。したがって、OCD 機能は、Active レギュレータが有効な場合にのみ使用できます。DeepSleep 電源モードなど省電力モードでは、Active レギュレータはオフとなり OCD もオフされます。MCU は、PMIC が制御範囲から外れた、または高速の負荷電流増加に対応できないといった VCCD 電圧低下を検出します。これは、PMIC に OCD 機能が無い場合に有効です。Active レギュレータが無効な場合、この機能は無効です。



## 2 TRAVEO™ T2G 電源システム

DeepSleep レギュレータは、DeepSleep 電源モードでのみコア電源を供給します。レジスタにより DeepSleep 電源モードで PMIC または DeepSleep レギュレータのいずれかが MCU に電源供給するかを設定できます。システムに応じて表 5 に示される設定を選択できます。

**表 5 PMIC での内部レギュレータ設定**

ユースケース	設定	Active レギュレータ (OCD 含む)	DeepSleep レギュレータ (OCD 含む)
OCD 機能なし PMIC を使用 DeepSleep 電源モードで PMIC は無効	PMIC から電源供給時、OCD は有効です。 DeepSleep 電源モードでは DeepSleep レギュレータが電源供給します。	有効	有効
OCD 機能付き PMIC を使用 DeepSleep 電源モードで PMIC は無効	PMIC から電源供給時、OCD は無効です。 DeepSleep 電源モードでは DeepSleep レギュレータが電源供給します。	無効	有効
OCD 機能付き PMIC を使用 DeepSleep 電源モードで PMIC は有効	PMIC から電源供給時、OCD は無効です。 DeepSleep 電源モードでは PMIC が電源供給します。	無効	有効 / 無効

動作例は、[PMIC 直接接続](#)で説明します。

**注:** 外部電源の過電流検出に OCD 機能を使用することは推奨しません。OCD 機能は Active レギュレータの一部のため直接 VCCD の電源ラインに実装されていません。

**注:** DeepSleep レギュレータを無効 (PWR\_CTL2.DPSLP\_REG\_DIS = "1") に設定した場合、PMIC から内部レギュレータへのハンドオーバーはできません。システムが PMIC へのハンドオーバー後、再び内部レギュレータにハンドオーバーする場合、DeepSleep レギュレータを有効にする必要があります。PWR\_CTL2.DPSLP\_REG\_DIS は、ブロック図での SwitchOverRegulators API またはソフトウェアで設定できます。

### 2.3.2 システムコール API

上記のとおり、外部電源と内部レギュレータ間でハンドオーバーする場合、専用のシステムコール API とレジスタを使用します。

TRAVEO™ T2G デバイスには、ブート ROM と SROM API を含むスーパーバイザリ ROM があります。SROM API は Arm® Cortex®-M0+ (CM0+) で使用するよう設計され、フラッシュプログラミングやライフサイクルステージの変更など特定の操作に使用します。また、API は、外部電源と内部レギュレータ間のハンドオーバーにも使用します。API は、API パラメータに応じたレジスタ設定と処理により、REGHC または PMIC コントローラを適切に制御し、ハンドオーバーを実行します。API の詳細については、アーキテクチャ TRM [2] の Nonvolatile Memory Programming 章を参照してください。

**注:** レジスタへの直接書き込みではなく CM0+ からシステムコールを使用してハンドオーバーを実行することを推奨します。

以下にハンドオーバーに使用する API を示します。

## 2 TRAVERO™ T2G 電源システム

### ConfigureRegulator API:

SwitchOverRegulators API によって、外部電源と内部レギュレータ間のハンドオーバーの前に最初に呼び出されて、要求されるレギュレータに設定します。PWR\_REGHC\_CTL.REGHC\_CONFIGURED = 0 の場合に、この API は必要です。表 6 に、この API パラメータを示します。

表 6 ConfigureRegulator API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
IPC_STRUCT.DATA0	SRAM_SCRATCH_AD DR1	[31:0]	API パラメータ SRAM_SCRATCH1 が保存されるアドレス。32 ビット境界である必要があります。
IPC_STRUCT.DATA1	SRAM_SCRATCH_AD DR2	[31:0]	API パラメータ SRAM_SCRATCH2 が保存されるアドレス。32 ビット境界である必要があります。
SRAM_SCRATCH1	Opcode	[31:24]	ConfigureRegulator API のオペコード 0x15
	RadjValue	[15:13]	このフィールドは、PMIC のリセット電圧調整設定に使用します。 詳細は、レジスタ TRM [3] の PWR_REGHC_CTL.REGHC_PMIC_RADJ を参照してください。 このフィールドは、CYT3D/4D シリーズでは無効です。
	VADJ trim value	[12:8]	このフィールドは REGHC から取得する必要な電圧出力調整に使用されます。Operation mode (bit[1]) が外部トランジスタの場合、このフィールドに 16(0x10)を設定してください。 詳細は、レジスタ TRM [3] の PWR_REGHC_CTL.REGHC_VADJ を参照してください。
	Vadj	[7]	このビットは REGHC_PMIC_VADJ_DIS を設定します。 TRAVERO™ T2G では、PMIC を使用した構成時 DRV_VOUT ピンは未使用のためこのビットは "1" に設定してください。 詳細は、レジスタ TRM [3] の PWR_REGHC_CTL. REGHC_PMIC_VADJ_DIS を参照してください。
	UseRadj	[6]	このビットは、PMIC のリセット電圧調整の使用を設定します。 0: RADJ を使用しません 1: RADJ を使用します 詳細は、レジスタ TRM [3] の PWR_REGHC_CTL. REGHC_PMIC_USE_RADJ を参照してください。 このフィールドは、CYT3D/4D シリーズでは無効です。

(続く)

## 2 TRAVEO™ T2G 電源システム

表 6 (続き) ConfigureRegulator API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
	UseLinReg	[5]	<p>このビットは、外部 PMIC モード中に Active レギュレータを有効にします。このビットは Operating mode が”1” (外部 PMIC)の時、有効です。</p> <p>0: PMIC 有効後、内部 Active レギュレータは、無効です。</p> <p>1: PMIC 有効後、内部 Active レギュレータは、有効のままです。</p> <p>詳細は、レジスタ TRM [3]の PWR_REGHC_CTL.REGHC_PMIC_USE_LINREG を参照してください。</p> <p>SwitchOverRegulators API がブロッキングコールで呼び出された場合、このビットの構成によって PWR_CTL2.DPSLP_REG_DIS がセットされる場合があります。</p>
	DeepSleep	[4]	<p>このビットは、DeepSleep 電源モード中の PMIC の動作を設定します。このビットは Operating mode が”1” (外部 PMIC)の時、有効です。</p> <p>0: DeepSleep 電源モードで PMIC は無効です。</p> <p>1: DeepSleep 電源モードで PMIC は有効です。(DeepSleep レギュレータは DeepSleep 電源モードで無効です)</p> <p>詳細は、レジスタ TRM [3]の PWR_REGHC_CTL4.REGHC_PMIC_DPSLP を参照してください。</p> <p>SwitchOverRegulators API がブロッキングコールで呼び出された場合、このビットの構成によって PWR_CTL2.DPSLP_REG_DIS がセットされる場合があります。</p> <p>このフィールドは、CYT3D/4D シリーズでは無効です。</p>

(続く)

## 2 TRAVERO™ T2G 電源システム

表 6 (続き) ConfigureRegulator API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
	Reset Polarity	[3]	<p>PMIC のエラー条件を設定します。(PMIC リセット出力 (RO) の有効レベルまたは PMIC PG 信号の無効レベル)これは、RO または PG 信号入力に基づいてリセットを発行するために使用します。</p> <p>この要因によってリセットされた場合、RES_CAUSE.RESET_PMIC に示されます。</p> <p>このビットは Operating mode が”1” (外部 PMIC) の時、有効です。</p> <p>0: RO または PG 信号入力が”0”の時、リセットが発生します。(RO または PG 信号が”1”の時、パワーグッド状態を示します)</p> <p>1: RO または PG 信号入力が”1”の時、リセットが発生します。(RO または PG 信号が”0”の時、パワーグッド状態を示します)</p> <p>詳細は、レジスタ TRM [3]の PWR_REGHC_CTL.REGHC_PMIC_STATUS_POLARITY を参照してください。また、この API は PWR_REGHC_CTL.REGHC_PMIC_STATUS_INEN ビットも設定します。</p>
	Enable Polarity	[2]	<p>PMIC を有効にするための極性を設定します。REGHC は REGHC_PMIC_CTL_POLARITY を使用して PMIC を有効にし、補数を使用し PMIC を無効にします。</p> <p>このビットは Operating mode が”1” (外部 PMIC) の時、有効です。</p> <p>0: “0”で PMIC は有効です。</p> <p>1: “1”で PMIC は有効です。</p> <p>詳細は、レジスタ TRM [3]の PWR_REGHC_CTL.REGHC_PMIC_CTL_POLARITY を参照してください。また、この API は PWR_REGHC_CTL.REGHC_PMIC_CTL_OUTEN ビットも設定します。</p>
	Operating Mode	[1]	<p>このビットは、パストランジスタまたは PMIC を指定します。</p> <p>0: 外部トランジスタ</p> <p>1: 外部 PMIC</p> <p>詳細は、レジスタ TRM [3]の PWR_REGHC_CTL.REGHC_MODE を参照してください。</p> <p>このビットは、SwitchOverRegulators API の”Operating Mode”と一致する必要があります。</p> <p>REGHC のないデバイスでは、Operating mode は無効です。</p>

(続く)

## 2 TRAVEO™ T2G 電源システム

表 6 (続き) ConfigureRegulator API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
	その他	-	未使用
SRAM_SCRATCH2	WaitCount	[8:0]	<p>PMIC ステータスが OK 後、4 <math>\mu</math>s 刻みの待機カウントを設定します。これは、ハードウェアシーケンサによって使用され、内部レギュレータを無効にする前に追加時間を設定できます。</p> <p>詳細は、レジスタ TRM [3] の PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT を参照してください。</p> <p>ConfigureRegulator API は、WaitCount[8:0]をサポートし、WaitCount[9]をサポートしないことに注意してください。</p> <p>したがって、0x1FF を超える WaitCount が必要な場合、ConfigureRegulator API の成功後、ユーザソフトウェアは、PWR_REGHC_CTL.REGHC_PMIC_STATUS_WAIT[29:20]を設定する必要があります。</p>

0x1FF を超える WaitCount を設定する場合、アプリケーションソフトウェアより PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを直接設定してください。図 2 に、REGHC\_PMIC\_STATUS\_WAIT 設定フローを示します。

## 2 TRAVEO™ T2G 電源システム

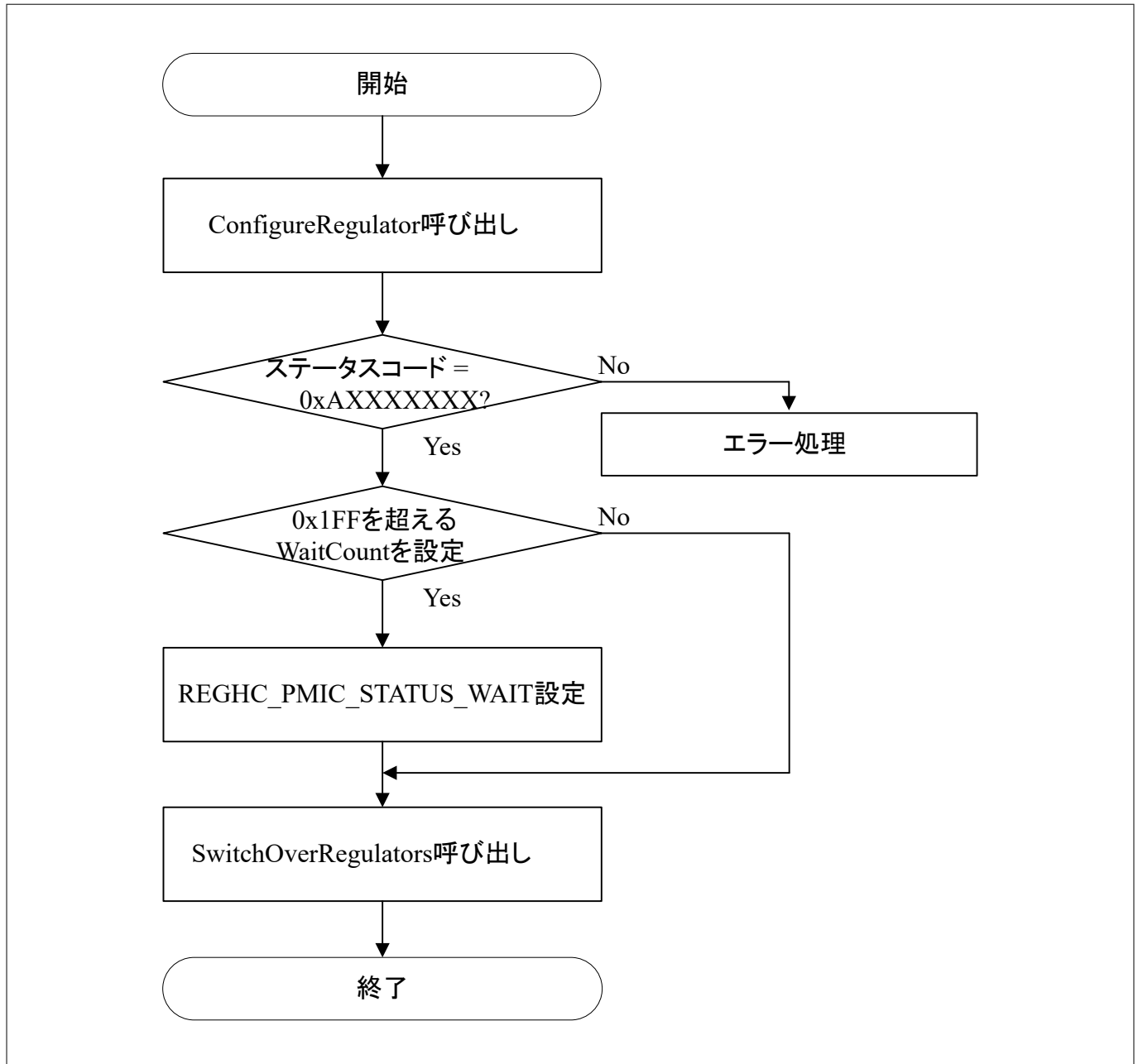


図 2 REGHC\_PMIC\_STATUS\_WAIT 設定フロー

このレジスタは、ConfigureRegulator API の完了後、SwitchOverRegulators API の呼び出し前に設定する必要があります。

**注:** ConfigureRegulator API を呼び出すとき、アプリケーションソフトウェアが WaitCount を設定する場合であっても、SRAM\_SCRATCH\_ADDR2 が格納される IPC\_STRUCT.DATA1 は有効なユーザ RAM 領域を指定する必要があります。無効なアドレスを設定した場合、バスエラーが発生する場合があります。

### SwitchOverRegulators API:

Active 電源モードで、コア電圧供給 VCCD を外部電源と内部レギュレータのいずれかに選択するために使用します。

外部電源が選択される場合、この API は 1 回だけ呼び出す必要があり、REGHC または PMIC コントローラは、内部 DeepSleep レギュレータと外部レギュレータ間のハンドオーバーを自動的に制御します。消費電流が Active レ

## 2 TRAVEO™ T2G 電源システム

ギュレータの許容範囲を超える可能性があるため、アプリケーションコアの起動前にこの API を呼び出す必要があります。この API は ConfigureRegulator API の後に呼び出してください。

表 7 に、この API パラメータを示します。

**表 7 SwitchOverRegulators API パラメータ**

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
IPC_STRUCT.DATA	SRAM_SCRATCH_A DDR	[31:0]	API パラメータが保存されるアドレス。32 ビット境界である必要があります。
SRAM_SCRATCH	Opcode	[31:24]	SwitchOverRegulators API のオペコード 0x11
	Blocking	[23:16]	このフィールドは CM0+コアをブロックまたは非ブロックするかを定義します。 0: Non-blocking call 1: Blocking call Blocking call は、ハンドオーバが完了した場合にのみ Syscall が完了します。Non-blocking call は、ハンドオーバ完了前であってもステータスを返します。つまり、Non-blocking call を使用する場合、ユーザソフトウェアはハンドオーバが完了するまで待つ必要があります。詳細は、 <a href="#">SwitchOverRegulators API の Non-blocking call 処理</a> を参照してください。 この API が Blocking call で呼び出された場合、内部レギュレータは、外部電源ハンドオーバに適した状態に変更されます。この API が Nonblocking call で呼び出された場合、状態は変更されないことに注意してください。 加えて、この API が Blocking call で呼び出され PMIC が DeepSleep 電源モードで有効な場合 (UseLinReg = "0" および DeepSleep = "1"に ConfigureRegulator API によって構成されている場合)、PWR_CTL2.DPSLP_REG_DIS は "1"に設定されます。
	Select regulator	[15:8]	このビットは、Active レギュレータから外部電源へ、または外部電源から Active レギュレータへのハンドオーバを決定するために使用します。 0: 外部電源に切り替えます。 1: Active レギュレータに切り替えます。
	Operating Mode	[1]	このビットは、パストランジスタまたは PMIC を指定します。 0: 外部トランジスタ 1: 外部 PMIC 詳細は、レジスタ TRM [3]の PWR_REGHC_CTL.REGHC_MODE を参照してください。 このビットは、ConfigureRegulator API の Operating Mode と一致する必要があります。 REGHC のないデバイスでは、Operating mode は無効です。

(続く)



## 2 TRAVEO™ T2G 電源システム

表 7 (続き) SwitchOverRegulators API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
	その他	-	未使用

### LoadRegulatorTrims API:

DeepSleep 電源モードへの移行およびハンドオーバー中の内部レギュレータ出力電圧を調整するために使用します。表 8 および表 9 に、この API のパラメータを示します。この API は、SwitchOverRegulators API を Blocking call で使用する場合を除いて、負荷遷移で外部レギュレータと内部レギュレータを切り替えることに呼び出す必要があります。

表 8 LoadRegulatorTrims API パラメータ

レジスタ/ SRAM_SCRATCH	名称	ビット	説明
IPC_STRUCT.DATA	SRAM_SCRATCH_A DDR	[31:0]	API パラメータが保存されるアドレス。32 ビット境界である必要があります。
SRAM_SCRATCH	Opcode	[31:24]	LoadRegulatorTrims API のオペコード 0x16
	ユースケース	[3:2]	このフィールドは LoadRegulatorTrims API が呼び出されるユースケースを指定します。 0: Force trim setting。レギュレータ設定にかかわらず要求されたトリム値を設定します。 1: DeepSleep Entry ユースケース 2: DeepSleep Exit ユースケース 3: Reset Recovery ユースケース ユースケースの詳細は、表 9 を参照してください。
	Operating Mode	[1]	このビットは内部レギュレータの出力電圧を指定します。 0: 内部レギュレータ 1: 外部電源 このフィールドはユースケースが"0"の時のみ有効です。
	その他	-	未使用

表 9 LoadRegulatorTrims API のユースケース

ユースケース	説明
Force trim setting	このユースケースでは、レギュレータ設定をバイパスして必要な出力電圧を設定する必要があります。出力状態については表 10 を参照してください。
DeepSleep Entry	DeepSleep 電源モード移行前に、DeepSleep レギュレータの出力電圧を変更する必要があります、ただし、DeepSleep 電源モードで DeepSleep レギュレータがオフに設定されている場合、このユースケースは無効です。
DeepSleep Exit	システムが DeepSleep 電源モードから復帰する場合に使用します。 DeepSleep 電源モードに入る前に DeepSleep Entry が実行されている必要があります。

(続く)



## 2 TRAVEO™ T2G 電源システム

表 9 (続き) LoadRegulatorTrims API のユースケース

ユースケース	説明
Reset Recovery	ハンドオーバー中にリセット(LV リセット)が発生した場合でも、電源の遷移には影響しませんが、アプリケーションはリセットされます。この場合、リセット解除後にこのユースケースを使用し、内部レギュレータの出力状態が要件に従っていることを確認する必要があります。

**注:** Hibernate 電源モードに移行する場合、システムは事前動作なしに直接 Hibernate 電源モードに移行できます。

SwitchOverRegulators および LoadRegulatorTrims API は、表 10 に示すように、ハンドオーバー移行ケースにしたがい、内部レギュレータの出力状態を変更します。内部レギュレータの出力電圧は、出力状態にしたがって適用されます。

表 10 コア電源移行後の内部レギュレータの出力状態

ハンドオーバー移行のケース	内部レギュレータの出力状態	説明
内部レギュレータから外部電源	外部状態	コアは PMIC から電源供給されます。PMIC と内部レギュレータはオンです。
	オフ	コアは PMIC から電源供給されます。PMIC はオン、内部レギュレータはオフです。
外部電源から内部レギュレータ	内部状態	コアは内部レギュレータから電源供給されます。PMIC はオフ、内部レギュレータはオンです。

API の処理が完了後、ステータスコードが SRAM\_SCRATCH アドレスに返されます。表 11 に、ステータスコードを示します。

表 11 ステータスコード

ステータスコード	説明	処理例
0xAXXXXXXX	API は正常に完了しました。 X: Don't care	-
0xF00000E0	レギュレータはマニュアルモードに設定されます。	ConfigureRegulator API を呼び出します。
0xF00000E1	レギュレータはハンドオーバー中です。 - 完了を待ちます。	API 完了を待ちます。
0xF00000E2	レギュレータはすでに有効になっています。	追加の処理はありません。
0xF00000E3	レギュレータは ConfigureRegulator によって設定されません。	ConfigureRegulator API を呼び出します。
0xF00000E4	ConfigureRegulator API 以外の OpMode パラメータで syscall が呼び出された場合、SwitchOverRegulators API によって返されます。	正しい組み合わせで API を呼び出します。

## 2 TRAVEO™ T2G 電源システム

これらの意図しないエラーが再発生する場合、チップ全体のリセットなどのシステム設計にしたがって適切なエラー処理を実行してください。

### 2.3.3 ハンドオーバーシーケンスの概要

ここでは API を使用した、ハンドオーバーシーケンス例と、REGHC 動作のタイミングチャートについて説明します。

図 3 に、内部レギュレータと外部電源間のハンドオーバーシーケンスの概要を示します。このシーケンスは、LoadRegulatorTrims, ConfigureRegulator, および Blocking call での SwitchOverRegulators API が使用されます。

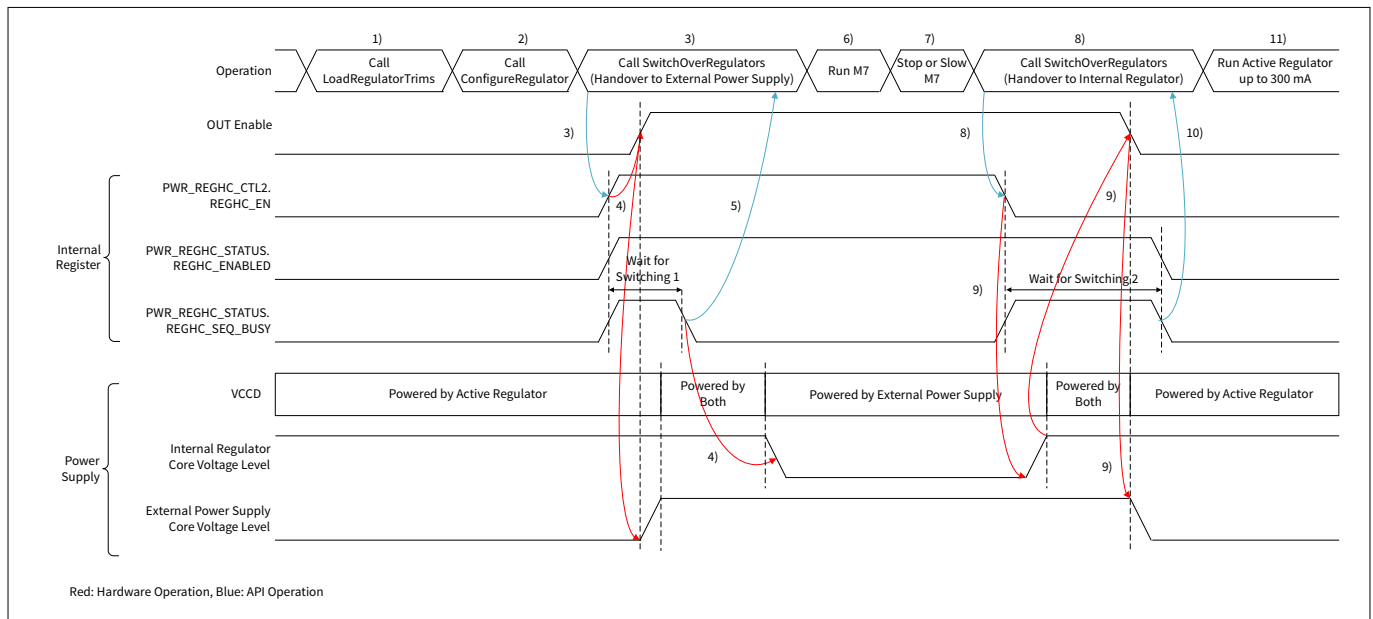


図 3 内部レギュレータと外部電源間のハンドオーバーシーケンス

起動中、MCU は内部レギュレータで開始します。

内部レギュレータから外部電源へのハンドオーバ

1. SW: PWR\_REGHC\_CTL.REGHC\_CONFIGURED が”1”に設定されている場合、Reset recovery ユースケースで LoadRegulatorTrims API を呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. SW: パストランジスタまたは PMIC で REGHC 設定のために ConfigureRegulator API を呼び出してください。外部電源に応じて Operating mode を選択してください。

注: REGHC のないデバイスでは、Operating mode は無効です。

3. SW: SwitchOverRegulators API を呼び出し、ハンドオーバを実行してください。ConfigureRegulator API で選択した Operating mode を選択してください。次に PWR\_REGHC\_CTL2.REGHC\_EN を”1”に設定してください。
4. HW: 外部電源が有効になります。ConfigureRegulator API に応じて、内部レギュレータが無効になるか、外部状態に設定されます。
5. SW: SwitchOverRegulators API がステータスコードを返します。Blocking call で呼び出された場合、API はハンドオーバの完了を待ちます。

“切り替え 1 待ち”は、外部電源が完全に準備されるまでの待機時間です。パストランジスタ設定では、待機時間は 15 us 以内に完了します。PMIC 設定では、待機時間はパワーグッド状態信号と PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT の値に依存します。

ソフトウェアによって REGHC\_PMIC\_STATUS\_WAIT をプログラムできます。したがって、カウンタが長いほど、ハードウェアシーケンサが内部 Active レギュレータを有効にする時間が長くなります。この機能は、PMIC の起動前までの時間を待機できます。詳細については、レジスタ TRM [3]を参照してください。

6. SwitchOverRegulators API が正常に完了後、外部電源により MCU は高負荷で実行できます。

## 2 TRAVEO™ T2G 電源システム

**注:** 動作クロック周波数は、VCCD 電圧のアンダーシュートを回避するために、アプリケーションソフトウェアによって段階的に増加させる必要があります。

### 外部電源から内部レギュレータへのハンドオーバー

7. MCU は、ハンドオーバーを実行する前に、内部レギュレータの仕様範囲の消費電流に対応したクロック設定で実行する必要があります。
8. SW: SwitchOverRegulators API を呼び出してください。ConfigureRegulator API で選択した Operating mode を選択してください。次に PWR\_REGHC\_CTL2.REGHC\_EN を"0"に設定してください。
9. HW: PWR\_REGHC\_CTL2.REGHC\_EN が"0"に設定されると内部レギュレータが有効になります。内部レギュレータが電源供給を開始すると出力エナブルがディassertされ、外部電源が無効になります。
10. SW: SwitchOverRegulators API がステータスコードを返します。Blocking call で呼び出された場合、API はハンドオーバーの完了を待ちます。

“切り替え 2 待ち”は、内部レギュレータの準備ができるまでの時間待機時間です。パストランジスタ設定では、待機時間は 10 us 以内に完了します。PMIC 設定では、待機時間はパワーグッド状態に依存します。

SwitchOverRegulators API が正常に完了後、MCU は内部レギュレータで実行されます。

**注:** “切り替え 1 待ち”および“切り替え 2 待ち”で指定された時間内にハンドオーバーが完了しない、または意図しない API エラーが再発生する場合、WDT を使用するか、外部システム制御での XRES\_L トリガ要求によって、電源システムを含むチップ全体のリセットをすることを推奨します。

### 2.3.4 SwitchOverRegulators API の Non-blocking call 処理

SwitchOverRegulators API には、Blocking call と Non-blocking call の 2 つの呼び出しモードがあります。Blocking call により SwitchOverRegulators API を呼び出した場合、API はハンドオーバーの完了待ち、および内部レギュレータ出力を適切な状態に変更します。さらに、PMIC が DeepSleep 電源モードで有効 (ConfigureRegulator API の UseLinReg = "0" および DeepSleep = "1") の場合、API は PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定します。

しかしながら、Non-blocking call により SwitchOverRegulators API を呼び出した場合、API は、ハンドオーバー操作が開始される前であってもステータスを返します。さらに、内部レギュレータの出力状態を変更せず、PWR\_CTL2.DPSLP\_REG\_DIS は設定されません。

ここでは、Non-blocking call での SwitchOverRegulators API 処理について説明します。表 12 に、Blocking call と Non-blocking call の違いを示します。

**表 12 Blocking call と Non-blocking call の違い**

動作	Blocking call	Non-blocking call
ハンドオーバー完了待ち	SwitchOverRegulators API で処理	アプリケーションソフトウェアで処理
内部レギュレータ出力状態	SwitchOverRegulators API で処理	LoadRegulatorTrims API 実行が必要
DeepSleep レギュレータの設定	SwitchOverRegulators API で処理 ConfigureRegulator API で UseLinReg = "0" および DeepSleep = "1"に設定された場合、SwitchOverRegulators API は PWR_CTL2.DPSLP_REG_DIS を"1"に設定します。	アプリケーションソフトウェアで処理 ConfigureRegulator API で UseLinReg = "0" および DeepSleep = "1"に設定された場合でも、SwitchOverRegulators API は PWR_CTL2.DPSLP_REG_DIS を"1"に設定しません。

## 2 TRAVEO™ T2G 電源システム

### 2.3.4.1 REGHC\_SEQ\_BUSY フラグの動作

ここでは、ユーザソフトウェアによるハンドオーバーの完了を確認する方法について説明します。コア電圧の内部レギュレータと PMIC 間の自動遷移の状態は、REGHC\_SEQ\_BUSY 状態フラグで示されます。

#### 遷移: 内部レギュレータから外部電源へ

PMIC 有効後、PG 信号がディアサートされている場合、フラグは"1"です。PG 信号がアサートされるとフラグは"0"に設定されます。フラグは、遷移シーケンスが正しく完了したことを示します。

**注:** ソフトスタート中に PG 信号がアサートされると、PMIC は完全に動作する前に内部レギュレータがオフになり、VCCD 電圧がアンダースhootする場合があります。この問題を軽減するために遅延時間を設定できます。フラグは、PMIC の PG 信号がアサートされた後も、プログラム可能な遅延時間によって"0"になりません。この遅延は、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT によって設定できます。

図 4 に、PMIC のイネーブル信号は Active HIGH でイネーブル、PMIC からの PG 信号が Active HIGH でパワーグッド状態の場合の REGHC\_SEQ\_BUSY の動作を示します。

#### 遷移: 外部電源から内部レギュレータへ

PMIC 無効後も、PG 信号がアサートされている場合、フラグは"1"です。その後、PG 信号がディアサートされると、フラグは"0"に設定されます。フラグは、遷移シーケンスが正しく完了したことを示します (内部レギュレータへの遷移が完了した)。

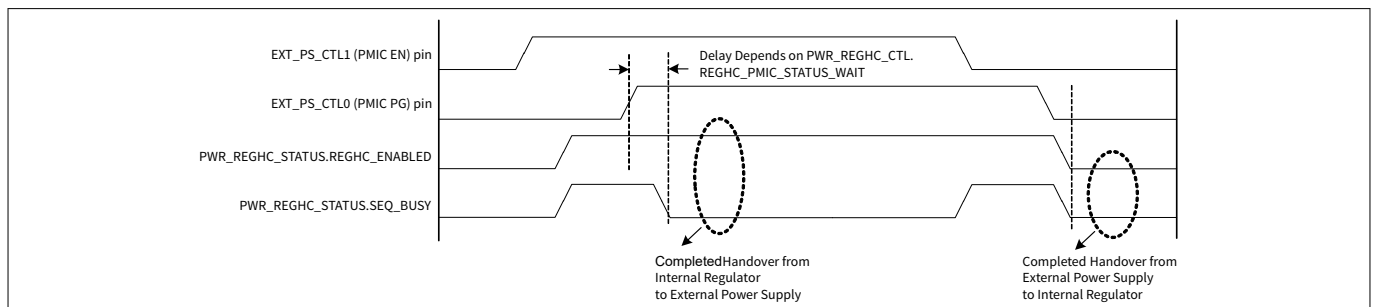


図 4 REGHC\_SEQ\_BUSY 動作

- 内部レギュレータから外部レギュレータへのハンドオーバー完了状態

PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0, PWR\_REGHC\_STATUS.REGHC\_ENABLED = 1.

- 外部レギュレータから内部レギュレータへのハンドオーバー完了状態

PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0, PWR\_REGHC\_STATUS.REGHC\_ENABLED = 0.

### 2.3.4.2 DeepSleep レギュレータの構成

SwitchOverRegulators API の Blocking call と Non-blocking call での内部レギュレータ構成の違い、およびそれらの使用方法について説明します。

MCU は、Active レギュレータを無効にし、DeepSleep 電源モードで PMIC を有効に設定すると DeepSleep レギュレータを無効 (PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定) にできます。

SwitchOverRegulators API は以下の場合、DeepSleep レギュレータを無効に設定します。

- ConfigureRegulator API によって、UseLinReg = "0"および DeepSleep = "1"に設定する
- SwitchOverRegulators API が Blocking call で呼び出される

ConfigureRegulator API によって、UseLinReg = "0"および DeepSleep = "1"以外の構成、または Non-blocking call で SwitchOverRegulators API を呼び出した場合、PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定しません。

上述のように、DeepSleep レギュレータが無効 (PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定) の場合、SwitchOverRegulators API を使用して PMIC から内部レギュレータへのハンドオーバーはできません。システムが

## 2 TRAVERO™ T2G 電源システム

SwitchOverRegulators API によって PMIC から内部レギュレータへのハンドオーバーを行う場合、PMIC へのハンドオーバー時に Non-blocking call で SwitchOverRegulators API を呼び出す必要があります。

### 2.3.4.3 Non-blocking call でのハンドオーバー処理例

図 5 に、Non-blocking call で SwitchOverRegulators API 使用して内部レギュレータから PMIC へのハンドオーバー例を示します。

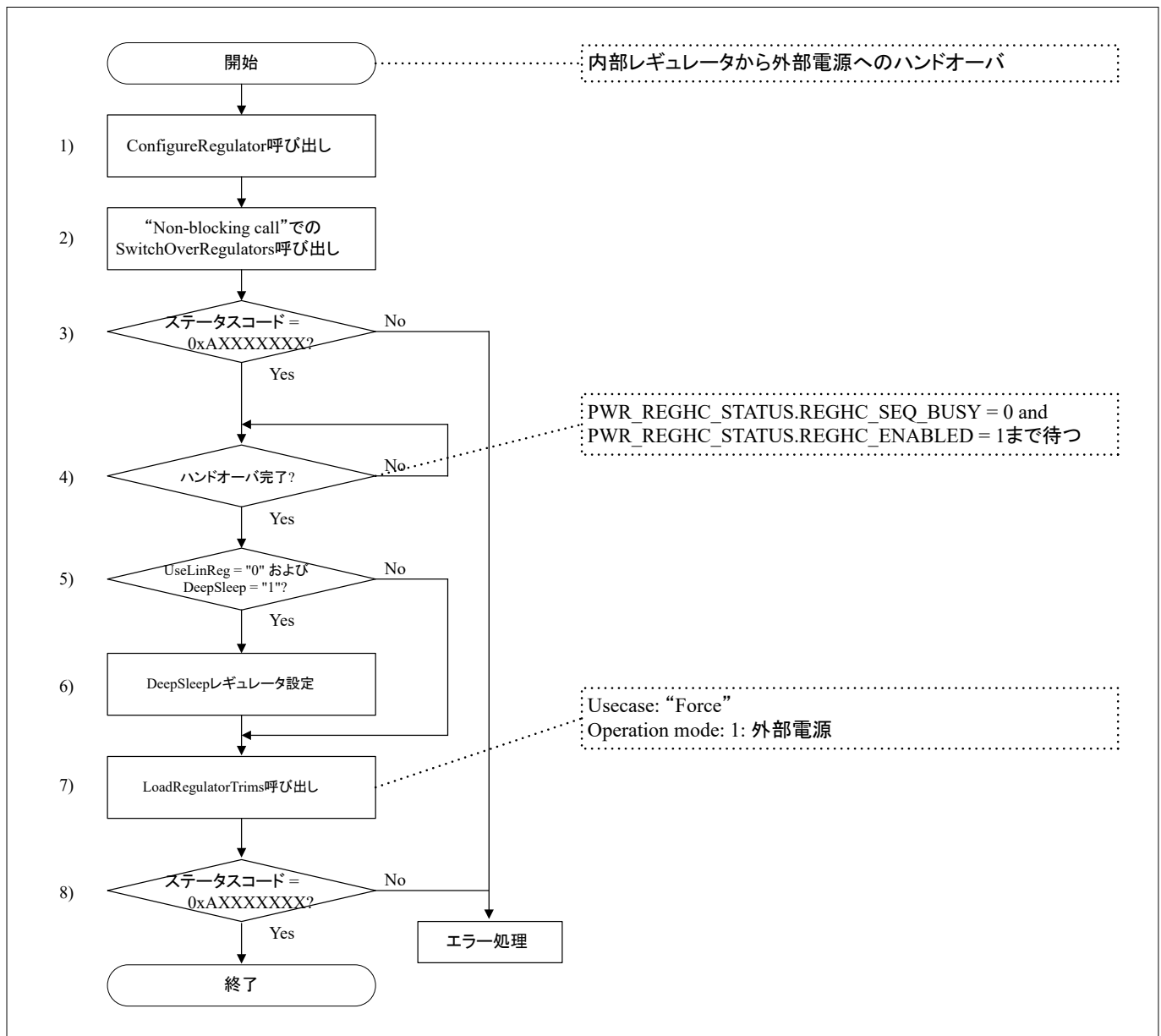


図 5 Non-blocking call での内部レギュレータから PMIC へのハンドオーバー例

1. 内部レギュレータを設定するため、ConfigureRegulator API を呼び出してください。
2. Non-blocking call で SwitchOverRegulators API を呼び出してください。
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。ただし、ハンドオーバーは完了していません。
4. ハンドオーバー完了のため PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0 および PWR\_REGHC\_STATUS.REGHC\_ENABLED = 1 になるまで待ってください。
5. PMIC が DeepSleep 電源モードで有効かどうか確認してください。



## 2 TRAVEO™ T2G 電源システム

6. PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定してください。DeepSleep レギュレータは無効になります。ただし、PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定した場合、PMIC から内部レギュレータへのハンドオーバを SwitchOverRegulators API を使用して再度実行することはできません。したがって、システムが API を使用して内部レギュレータに戻す場合は、PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定しないでください。
7. ユースケースを Force trim setting、および Operating mode を外部電源にして LoadRegulatorTrims API を呼び出してください。内部レギュレータの出力が外部状態に変更されます。  
Non-blocking call で SwitchOverRegulators API を呼び出した場合、内部レギュレータの出力状態が適切に変更されない場合があります。この場合は、LoadRegulatorTrims API の呼び出しによって内部レギュレータの出力状態を変更してください。
8. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。ただし、ハンドオーバは完了していません。

### 2.3.5 各ユースケースでの LoadRegulatorTrims 処理

LoadRegulatorTrims API は、内部レギュレータの出力状態を変更するために使用し、4 タイプのユースケース設定があります。詳細については、[LoadRegulatorTrims](#) を参照してください。

#### Force trim setting

現在のレギュレータ設定にかかわらず、Operating mode で指定された内部レギュレータ状態を設定します。多くの場合、Blocking call での SwitchOverRegulators API、およびその他ユースケースでの LoadRegulatorTrims API の呼び出しによって適切に制御されます。

#### DeepSleep entry

DeepSleep 電源モード移行前に内部レギュレータの出力状態を内部状態に設定します。したがって、DeepSleep 電源モードに移行する前に、DeepSleep Entry ユースケースで LoadRegulatorTrims API を実行する必要があります。ただし、DeepSleep 電源モードで DeepSleep レギュレータがオフの設定ではこのユースケースは不要です。

#### DeepSleep exit

DeepSleep 電源モードから Active 電源モードに移行後、内部レギュレータの出力状態を外部状態に設定します。システムが DeepSleep 電源モードから復帰し、PMIC へのハンドオーバが完了した後、DeepSleep Exit ユースケースで LoadRegulatorTrims API を実行する必要があります。これは、DeepSleep Entry ユースケースでの LoadRegulatorTrims API を使用して DeepSleep 電源モードへの移行が実行された場合にのみ必要です。

#### Reset recovery

リセット後、現在の REGHC または PMIC コントローラ状態に応じて適切な内部レギュレータ出力状態を設定します。REGHC または PMIC コントローラ設定は LV リセット後も維持されますが、内部レギュレータ出力状態は初期化されます。リセット解除後に、このユースケースを使用して、現在の状況に応じて内部レギュレータの出力状態を適用する必要があります。GPIO 設定など一部の設定も初期化されることに注意してください。

図 6 に、PMIC を使用したリセット後の Reset Recovery ユースケースの LoadRegulatorTrims API フロー例を示します。

## 2 TRAVEO™ T2G 電源システム

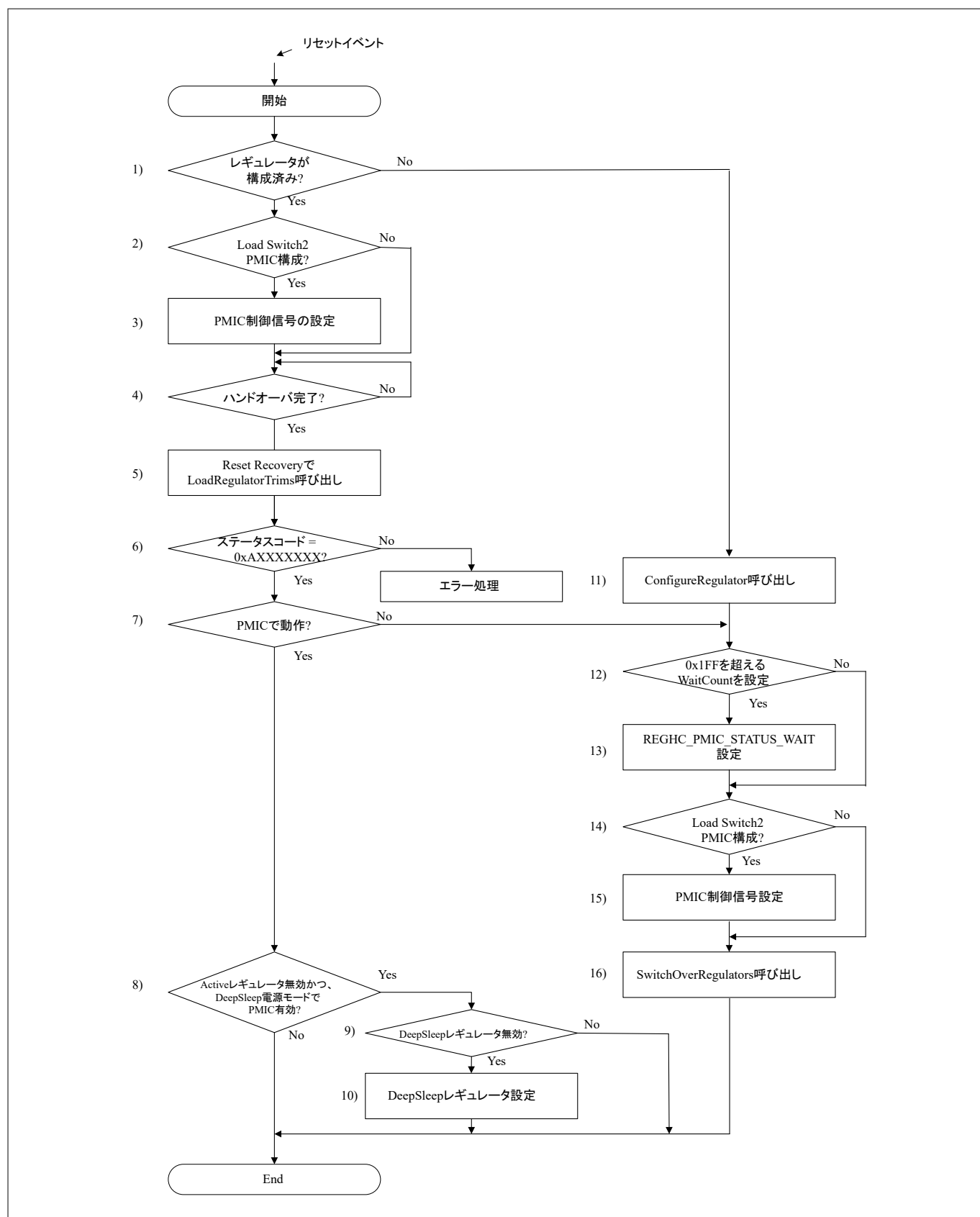


図 6 LoadRegulatorTrims API の Reset recovery 例

1. REGHC または PMIC コントローラ設定が完了済みかどうか確認してください。

## 2 TRAVEO™ T2G 電源システム

PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“1”の場合、設定はすでに完了し、ConfigureRegulator API の呼び出しは不要です。PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。REGHC 設定がされていない場合は(11)に進んでください。

2. **ロードスイッチ 2 構成での PMIC** 設定かどうか確認してください。そうでない場合は、(4)に進んでください。
3. PMIC イネーブル制御のため GPIO を設定してください。この場合、LV リセット前の PMIC イネーブル状態が再設定されます。以下に GPIO 再設定の例を示します。
  - a.  $\sim$  (PWR\_REGHC\_CTL2.REGHC\_EN ^ PWR\_REGHC\_CTL.REGHC\_PMIC\_CTL\_POLARITY) の値を GPIO データレジスタに書き込んでください。
  - b. GPIO を出力に設定してください。  
GPIO データレジスタへの書き込み後、GPIO 出力への設定が必要です。手順が異なる場合、PMIC に意図しない出力がされる場合があります。  
異なる設定ではこの処理は不要です。
4. ハードウェアがハンドオーバ完了するまで待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
5. Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。  
内部レギュレータの出力状態が適切に設定されます。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、内部レギュレータ出力状態は外部状態に、“0”の場合は、内部レギュレータ出力状態は内部状態に設定されます。
6. LoadRegulatorTrims API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
7. MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(12)へ進んでください。
8. Active レギュレータが無効かつ DeepSleep 電源モードで PMIC が有効かどうかを確認してください (PWR\_REGHC\_CTL.REGHC\_PMIC\_USE\_LINREG = “0”かつ PWR\_REGHC\_CTL4.REGHC\_PMIC\_DPSLP = “1”)。そうでない場合は、終了に進んでください。
9. DeepSleep レギュレータを“無効”に設定するかどうか確認してください。そうでない場合は、終了に進んでください。
10. PWR\_CTL2.DPSLP\_REG\_DIS を“1”に設定してください。
11. ConfigureRegulator API を呼び出しレギュレータ設定してください。
12. WaitCout を 0x1FF より大きい値に設定する必要があるか確認してください。そうでない場合は、(14)に進んでください。
13. アプリケーションソフトウェアより、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT を設定してください。詳細については、**システムコール API** の ConfigureRegulator API を参照してください。
14. **ロードスイッチ 2 構成での PMIC** 設定かどうか確認してください。そうでない場合は、(16)に進んでください。
15. PMIC イネーブル信号を GPIO に設定してください。
16. PMIC にハンドオーバするため SwitchOverRegulators API を呼び出してください。

### 2.4 外部電源の設定

REGHC の外部電源の設定にはパストランジスタと PMIC の 2 つがあります。

**表 13** は外部電源設定の特長について示します。



## 2 TRAVEO™ T2G 電源システム

表 13 外部電源設定の特長

タイプ	部品コスト	変換効率
パストランジスタ	低	低
PMIC	高	高

高効率な電源が要求される場合、PMIC を選択することを推奨します。ただし、PMIC を使用する場合は、[PMIC \(スイッチングレギュレータ\)](#)で説明されている予防策が必要です。

各シリーズでサポートされる外部電源構成については、[表 1](#) を参照してください。

### 3 パストランジスタ

## 3 パストランジスタ

### 3.1 ハードウェア構成

図 7 に、パストランジスタの使用例を示します。パストランジスタ, VCCD および VDDD の平滑コンデンサ, および センス抵抗で構成されます。MCU は、EXT\_PS\_CTL0 および EXT\_PS\_CTL1 からの入力電圧を検出し、DRV\_VOUT を制御します。CYT3B/4B シリーズのみこの設定をサポートします。CYT6B シリーズはこの設定をサポートしていません。

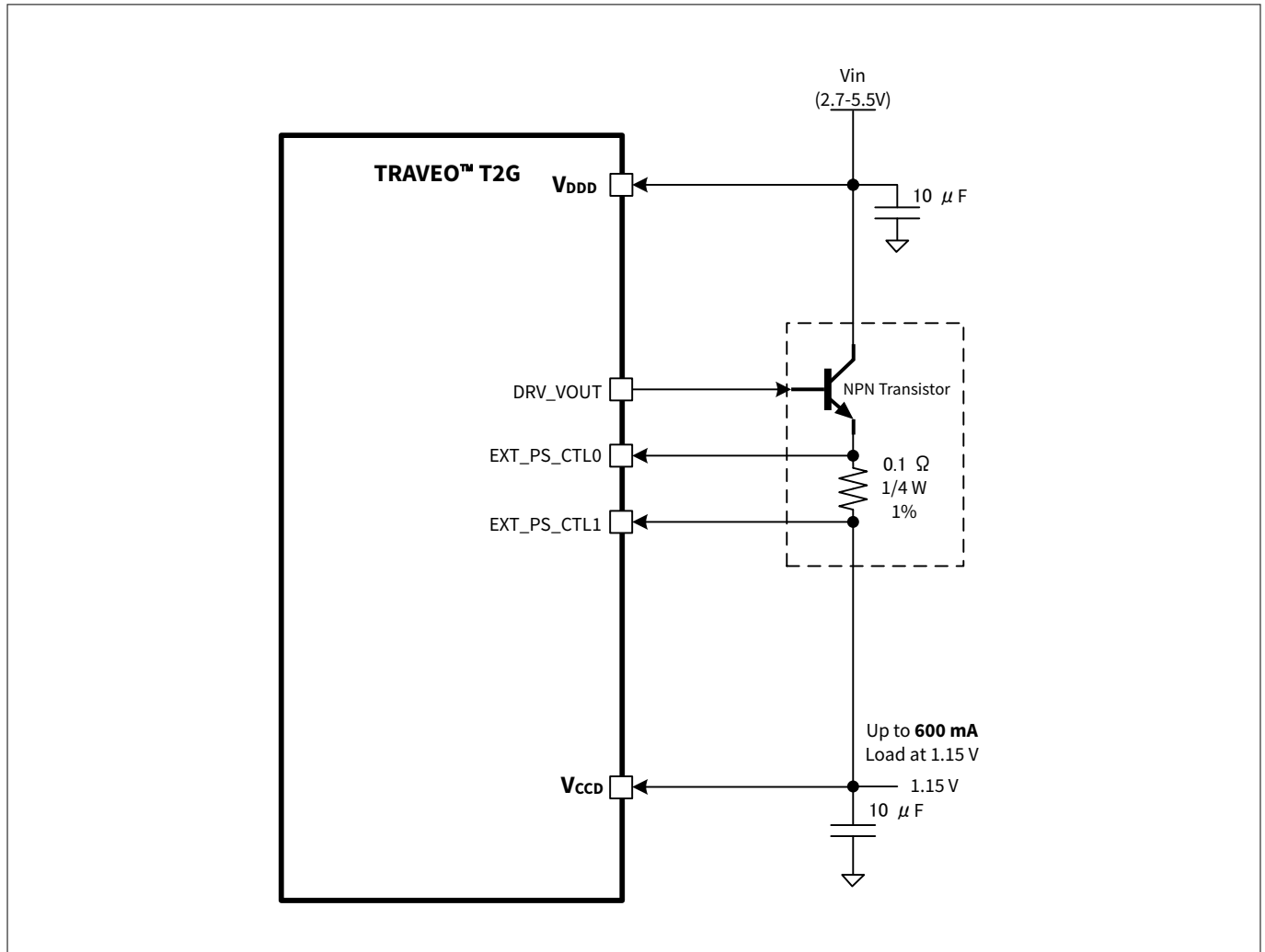


図 7 パストランジスタハードウェア構成

### 3.2 ハンドオーバータイミングチャート

図 8 に、ハンドオーバーシーケンスの例を示します。この例は、レギュレータ設定、内部レギュレータからパストランジスタへのハンドオーバー、DeepSleep 電源モードへの移行、DeepSleep 電源モードからの復帰、およびパストランジスタから内部レギュレータへのハンドオーバーが含まれます。

### 3 パストランジスタ

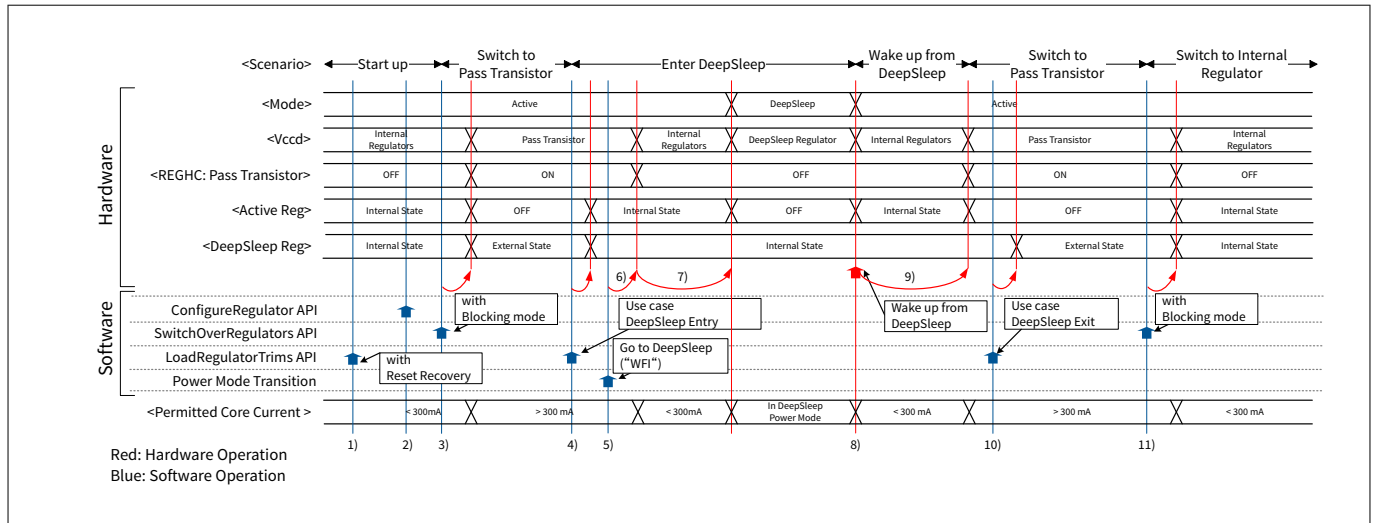


図 8 ハンドオーバーシーケンス例

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が”1”に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、パストランジスタで REGHC を設定するために ConfigureRegulator API を呼び出してください。
3. 内部レギュレータからパストランジスタへハンドオーバーするため SwitchOverRegulators API を Blocking call で呼び出してください。パストランジスタが有効になり、Active レギュレータが無効になります。DeepSleep レギュレータは外部状態に変更されます。
4. 内部レギュレータを内部状態に変更するために DeepSleep Entry ユースケースで LoadRegulatorTrims API を呼び出してください。  
Active および DeepSleep レギュレータは内部状態に変更されます。
5. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。その後、WFI 命令を使用して DeepSleep 電源モードへの移行を実行できます。
6. ソフトウェアが DeepSleep 電源モードへの移行を実行すると、ハードウェアはパストランジスタから内部レギュレータに切り替わります。
7. MCU は、内部レギュレータへの切り替えが完了すると、DeepSleep 電源モードへ移行します。
8. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生し、その後、Active レギュレータは内部状態で開始されます。
9. Active 電源モードへの移行が完了すると、パストランジスタが有効となり、Active レギュレータがオフされます。
10. 内部レギュレータを外部状態に変更するために DeepSleep Exit ユースケースで LoadRegulatorTrims API を呼び出してください。  
DeepSleep レギュレータは外部状態に変更されます。
11. パストランジスタから内部レギュレータへハンドオーバーするため SwitchOverRegulators API を呼び出してください。  
パストランジスタは無効になります。Active および DeepSleep レギュレータは内部状態に変更されます。

### 3.3 ソフトウェアフロー

#### 3.3.1 内部レギュレータからパストランジスタへのハンドオーバー

図 9 に、内部レギュレータからパストランジスタへのハンドオーバーのソフトウェアフローを示します。

### 3 パストランジスタ

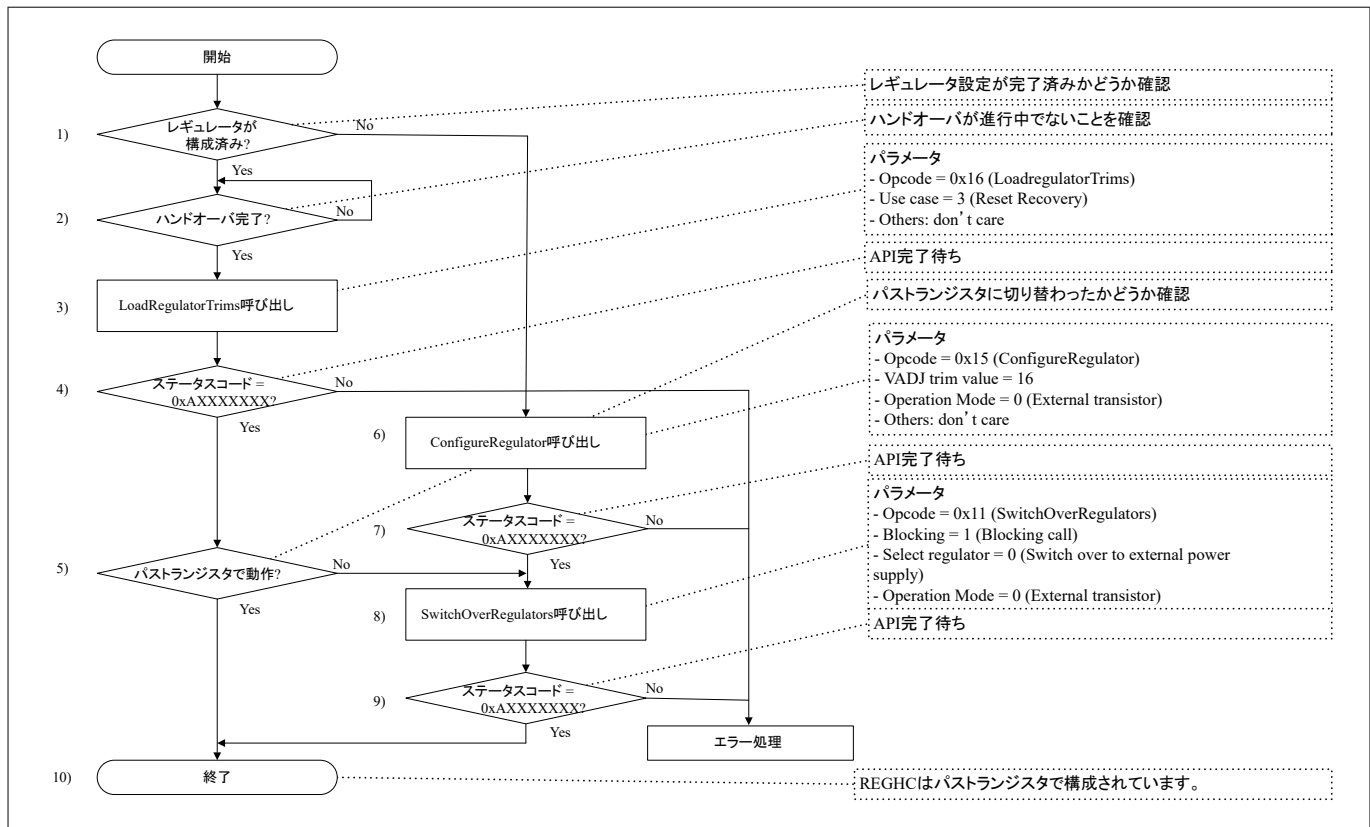


図 9 内部レギュレータからパストランジスタへのハンドオーバーフロー

- レギュレータ設定が完了済みかどうか確認してください。既に REGHC がセットアップされている場合、再構成せずに有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(6)へ進んでください。
- ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
- Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
- LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしたがって適切なエラー処理をしてください。
- MCU がパストランジスタで動作しているかどうか確認してください。  
PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、パストランジスタが電源供給しています。(10)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(8)へ進んでください。
- 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
  - VADJ trim value = 16 (1.15V 電圧出力のため)。詳細については、レジスタ TRM [2]を参照してください。
  - Operating Mode = 0 (外部トランジスタ)
  - その他 = Don't care

## 3 パストランジスタ

**注:** `SRAM_SCRATCH_ADDR2` が格納される `IPC_STRUCT.DATA1` には、有効なユーザ RAM 領域を設定してください。無効なアドレスを設定した場合、バスエラーが発生する場合があります。

7. `ConfigureRegulator` API の完了を待ってください。ステータスコードが `0xAXXXXXXX` ではない場合、システムにしがって適切なエラー処理をしてください。  
以下の手順に従って、パストランジスタで `REGHC` を有効にしてください。
8. 以下のパラメータで `SwitchOverRegulators` API を呼び出してください。
  - `Opcode = 0x11` (`SwitchOverRegulators` API 呼び出し)
  - `Blocking = 1` (Blocking call)
  - `Select regulator = 0` (外部電源に切り替えます)
  - `Operating Mode = 0` (外部トランジスタ)
9. `SwitchOverRegulators` API の完了を待ってください。ステータスコードが `0xAXXXXXXX` ではない場合、システムにしがって適切なエラー処理をしてください。
10. デバイスはパストランジスタの `REGHC` 構成で動作しています。  
既に `REGHC` がセットアップされている場合、再構成せずに有効にできます。

### 3.3.2 DeepSleep への遷移と復帰

図 10 に、DeepSleep 電源モードへの遷移を示します。

### 3 パストランジスタ

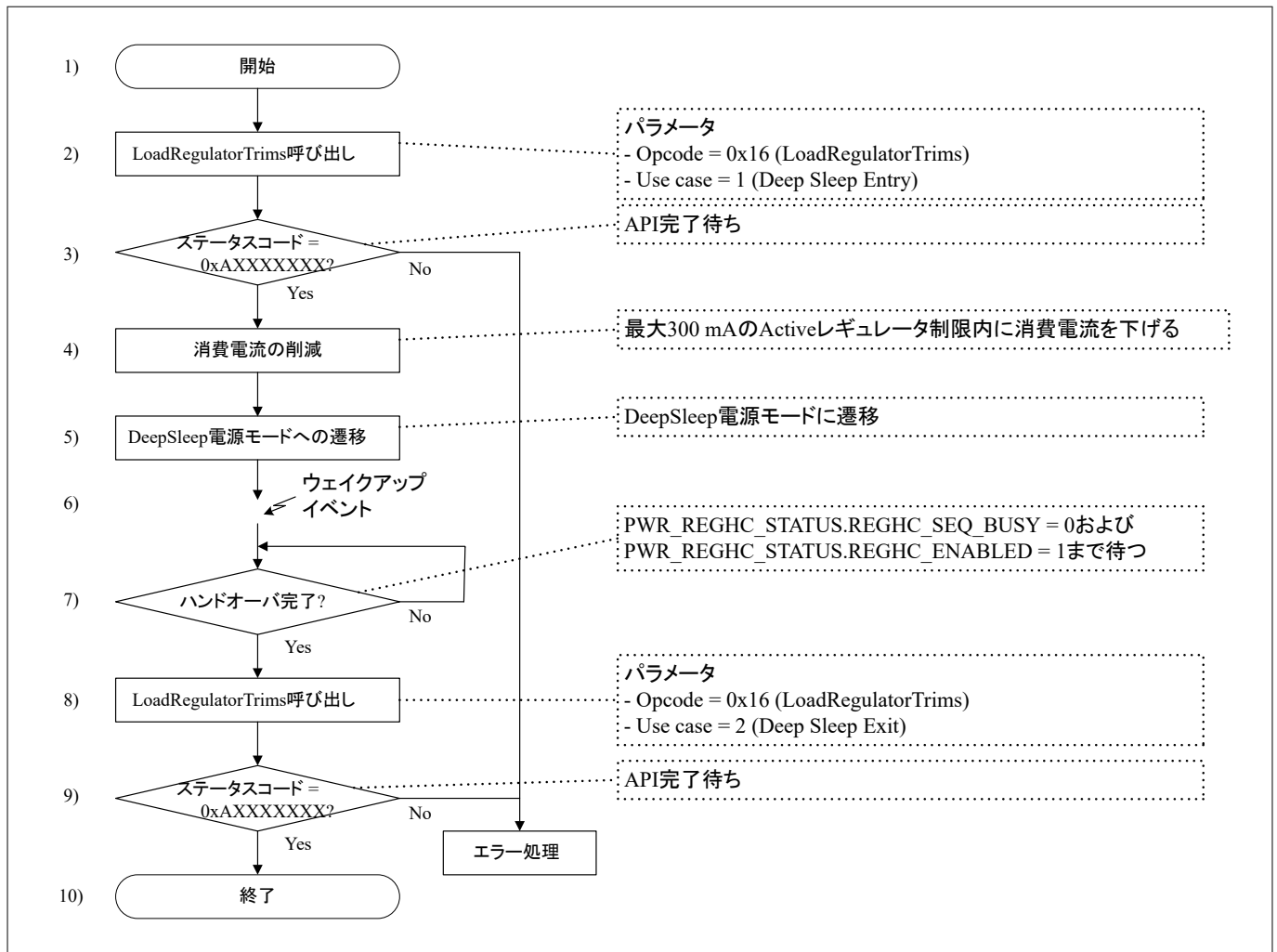


図 10 DeepSleep 遷移と復帰のフロー (パストランジスタ)

1. MCU は、パストランジスタから電源供給されています。
2. DeepSleep 電源モード移行前に以下のパラメータで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 1 (DeepSleep Entry)
3. LoadRegulatorTrims API の完了を待ってください。ステータスコードが”0xAxxxxxxx”以外の場合、システムに応じて適切なエラー処理をしてください。
4. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
5. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。MCU は内部レギュレータへのハンドオーバーを開始し、パストランジスタを無効にします。その後、Active レギュレータは無効になり、DeepSleep 電源モード中 DeepSleep レギュレータが電源供給します。MCU は DeepSleep レギュレータから電源供給されます。
6. ウェイクアップイベントによって MCU は Active 電源モードに遷移します。次にハードウェアによってパストランジスタが有効になります。
7. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = “1”まで待ってください。
8. 以下のパラメータで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは外部状態に設定されます。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 2 (DeepSleep Exit)

### 3 パストランジスタ

9. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
10. MCU は、パストランジスタから電源供給されています。

#### 3.3.3 パストランジスタから内部レギュレータへのハンドオーバー

図 11 に、REGHC によって制御されるパストランジスタモードから内部レギュレータへのハンドオーバーフローを示します。

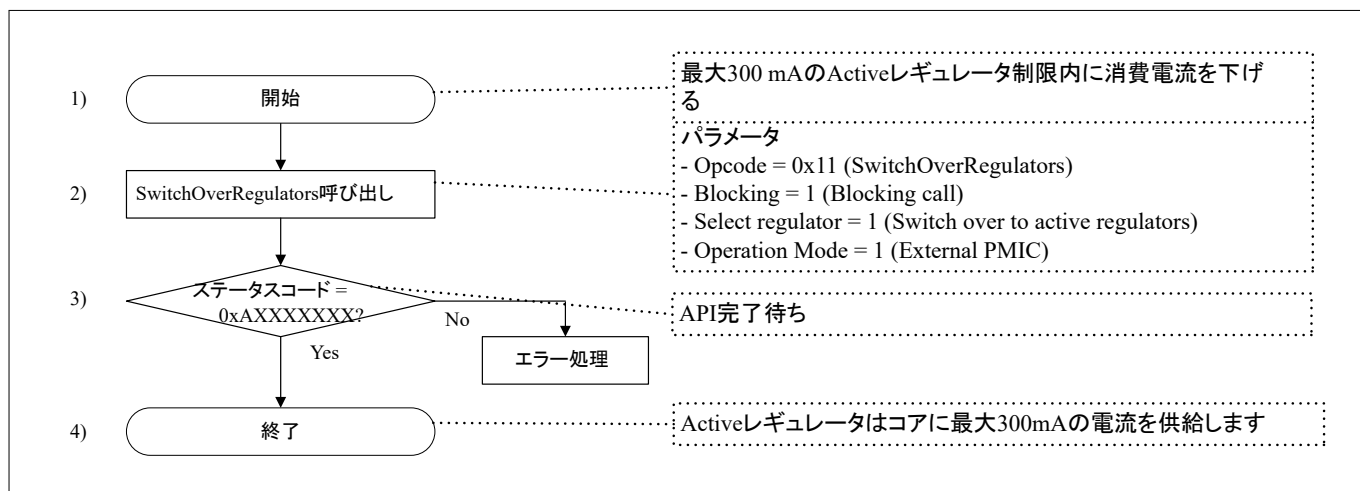


図 11 パストランジスタから内部レギュレータへのハンドオーバーフロー

パストランジスタ構成の REGHC から内部レギュレータにハンドオーバーする場合、

1. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
2. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode = 0 (外部トランジスタ)
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. デバイスは、Active レギュレータで動作しています。

### 3.4 部品選定

パストランジスタの部品選定は以下のとおりです。

#### $V_{CCD}$ の平滑コンデンサ

静電容量については、デバイスデータシート [1]にある Recommended Operating Conditions の「Smoothing capacitor」を参照してください。X7R、許容誤差 $\pm 10\%$ の温度特性で 1.15 V の DC バイアスの影響を受けないセラミックコンデンサを選択してください。

表 14 に推奨する平滑コンデンサを示します。

表 14  $V_{CCD}$  の推奨平滑コンデンサ

特性	製品番号	個数	ベンダ
10 $\mu$ F, 6.3V, X7R, $\pm 10\%$	CGA4J1X7R0J106K	1	TDK

電流センス抵抗: 0.1 オーム、1/4 W、 $\pm 1\%$

### 3 パストランジスタ

#### 外部トランジスタ種類: NPN

表 15 に、外部トランジスタの要件を示します。

表 15 外部トランジスタ要件 (種類: NPN)

パラメータ	記号	最小値	単位	条件
静的順電流伝達率	$H_{FE}$	100	-	$I_C = 1\text{ A}$ , $V_{CE} = 1\text{ V}$
コレクタ-エミッタ間電圧	$V_{CEO}$	10	V	-
トランジション周波数	$f_T$	100	MHz	-
コレクタ電流	$I_C$	1	A	-
コレクタ電流損失	$P_{CD}$	2	W	-

また、定格温度を超えてはいけません。ジャンクション温度を見積もる必要があります。

動作条件下でのパストランジスタ損失は式 1 で計算できます。

式 1

$$P_{Loss} = (V_{DDD\_max} - V_{CCD\_min}) \times I_{VCCD\_max}$$

ここで、

- $P_{Loss}$ : パストランジスタの損失 (W)
- $V_{DDD\_max}$ : 最大  $V_{DDD}$  電圧 (V)
- $V_{CCD\_min}$ : 最小  $V_{CCD}$  電圧 (V)
- $I_{VCCD\_max}$ : 最大  $V_{CCD}$  負荷電流 (A)

パストランジスタの最大電流損失は  $P_{Loss}$  以上である必要があります。

パストランジスタのジャンクション温度は式 2 で計算できます。

式 2

$$T_J = P_{LOSS} \times \theta_{JA} + T_A$$

ここで、

- $T_J$ : パストランジスタのジャンクション温度 (°C)。これは、定格温度以下とする必要があります。
- $T_A$ : 周囲温度 (°C)
- $P_{Loss}$ : パストランジスタの損失 (W)
- $\theta_{JA}$ : 接合部から周囲温度までの熱抵抗 (°C/W)

表 16 に最大  $\theta_{JA}$  の例を示します。

表 16 最大  $\theta_{JA}$  の例

$T_{A\max}$ (°C)	$V_{DDD}$	$P_{Loss\max}$ (W)	$\theta_{JA}$ (°C/W)
+125	5V 電源ライン	2.64	< 9.5
+105			< 17.0
+85			< 24.6
+125	3.3V 電源ライン	1.50	< 16.7
+105			< 30.0
+85			< 43.3

注: 最大  $V_{DDD}$  電圧 = 5.5 V/3.6 V、最小  $V_{CCD}$  電圧 = 1.10 V、最大  $V_{CCD}$  負荷電流 = 0.6 A



## 3 パストランジスタ

パストランジスタによる損失は、 $V_{DD}$  に 3.3 V を供給することで低減できます。

### 3.5 レイアウト設計ガイドライン

以下に、PCB レイアウト設計のためのガイドラインを示します。

- $V_{CCD}$  平滑コンデンサは、MCU の近くに配置し、かつ電流センス抵抗、パストランジスタ、および  $V_{DD}$  平滑コンデンサのとなりに配置してください。
- $DRV\_VOUT$  はノイズの影響を受けやすいため、ノイズ発生源の近くに配線しないでください。
- $EXT\_PS\_CTL0$  および  $EXT\_PS\_CTL1$  端子と電流センス抵抗間の配線は、電源配線から分離し、互いに平行かつ近接して配線してください。この信号はノイズの影響を受けやすいため、ノイズ発生源からできるだけ遠ざけ、最短で配線してください。
- パストランジスタは、動作条件によって多くの熱を発生します。熱が放熱パッド (EP) からボードに放熱されるようにボードをレイアウトしてください。ランド領域は、パストランジスタの実装面と PCB の底面の EP を覆うように配置します。EP のフットプリントのビアは、電源層を含む PCB の各層の EP を覆うように配置したランド領域に電氣的に接続し、熱的に結合します。その他の注意事項については、NPN トランジスタのサプライヤが示すレイアウトガイドラインにしたがってください。

## 4 PMIC (スイッチングレギュレータ)

### 4 PMIC (スイッチングレギュレータ)

ここでは、PMIC を使用して  $V_{CCD}$  に電源供給する方法について説明します。

図 12 に、PMIC 接続の概要を示します。

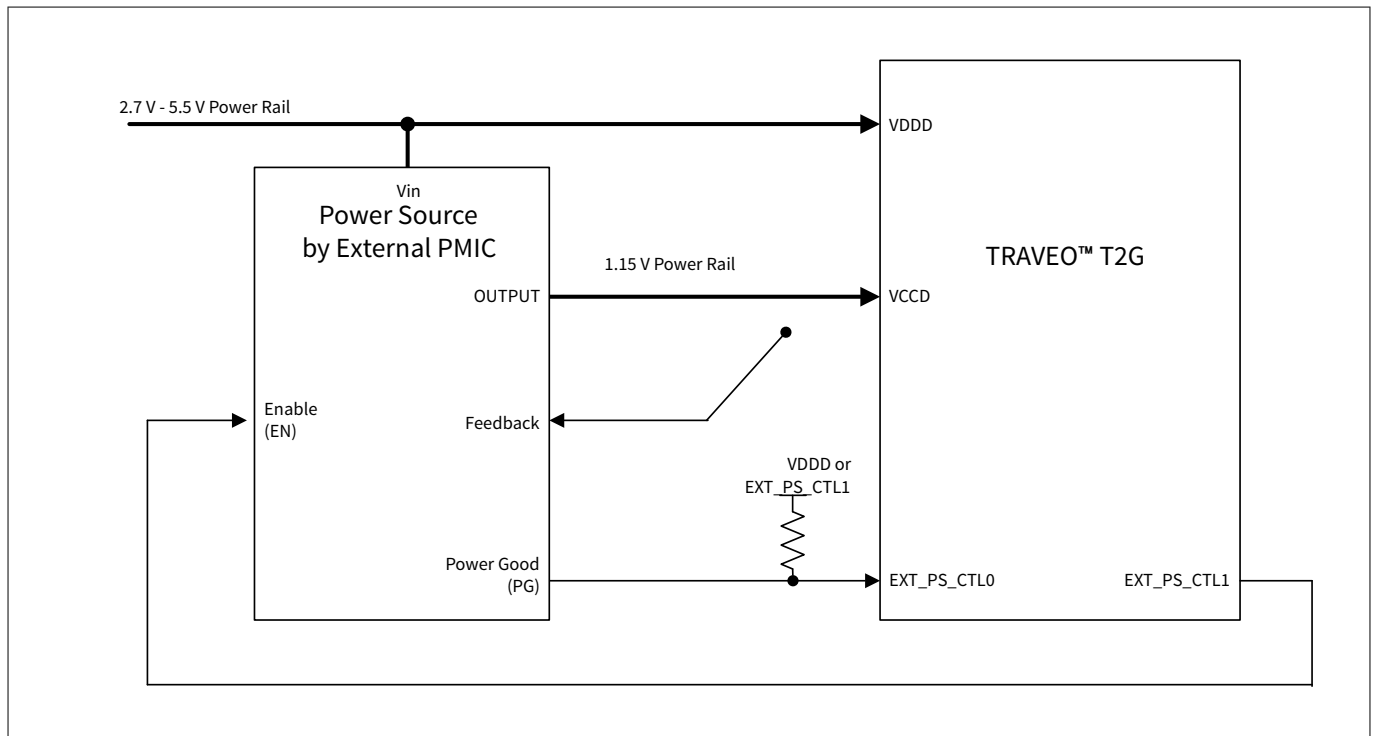


図 12 MCU と PMIC 間の接続

- PMIC 出力(アウトプット)は MCU の  $V_{CCD}$  に接続します。
- PMIC のフィードバックは、MCU の  $V_{CCD}$  に接続します。
- PMIC のイネーブル(EN)は、MCU の EXT\_PS\_CTL1 に接続します。EN が解放され電圧が特定されない場合はプルアップまたはプルダウン抵抗が必要です。
- PMIC のパワーグッド(PG)は、MCU の EXT\_PS\_CTL0 に接続します。PG がオープンドレインの場合、プルアップ抵抗が必要です。

図 13 に、 $V_{CCD}$  電源が接続された場合の、基本的なハンドオーバーシーケンスを示します。

## 4 PMIC (スイッチングレギュレータ)

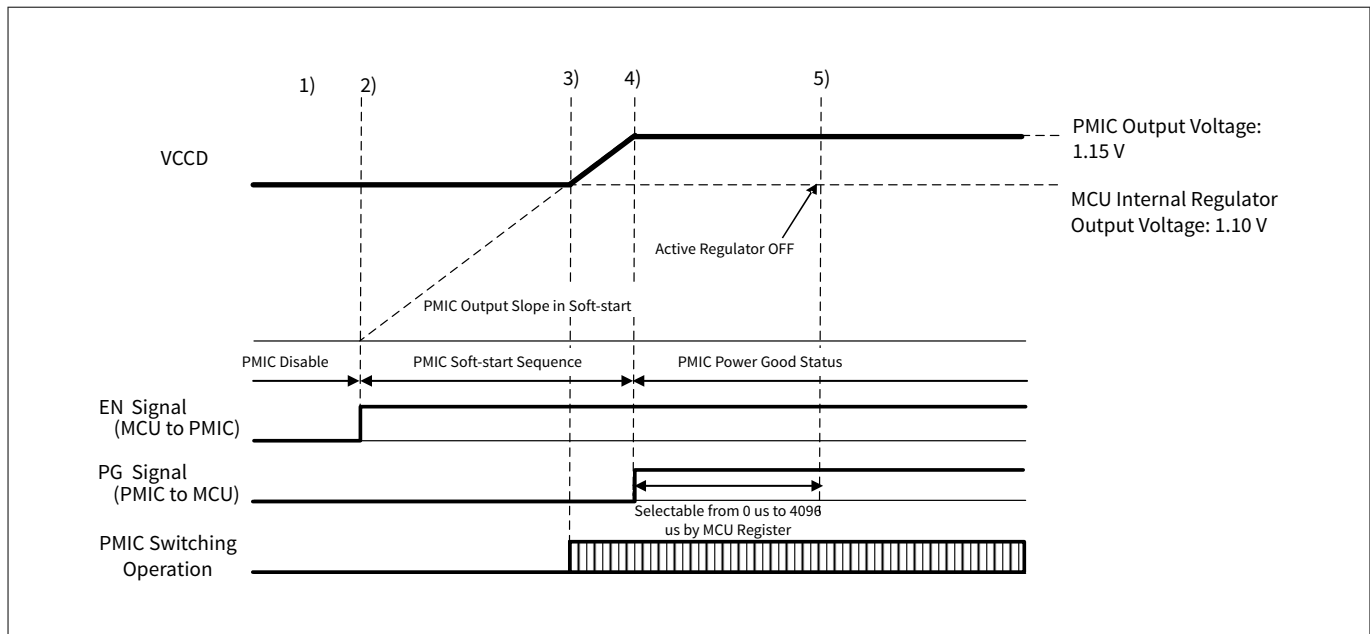


図 13 基本的な VCCD 電源のハンドオーバシーケンス

1. 内部レギュレータの Active レギュレータは、VCCD 電源ラインに電源供給します。
2. MCU は PMIC を有効にします。ただし、ソフトスタートシーケンス時は PMIC 出力の目標電圧が Active レギュレータの VCCD 出力電圧よりも低いため、PMIC はスイッチング動作を開始しません。
3. ソフトスタートシーケンスでの PMIC 目標電圧と Active レギュレータの VCCD 出力電圧が交差します。そのとき PMIC はスイッチング動作を開始し、VCCD 電源ラインに電源供給します。
4. VCCD 電圧が定常状態の PMIC 目標電圧まで上昇し、PMIC ソフトスタートシーケンスが完了すると、PMIC は PG 信号をドロップダウンします。
5. MCU は、PG 信号のパワーグッド状態を検出すると、Active レギュレータをオフにします。

### 4.1 PMIC 仕様要件

MCU は、VCCD 電源供給のため Active レギュレータと PMIC の 2 つの電源を制御します。したがって、PMIC を使用する場合、以下の点を考慮する必要があります。

- PMIC は MCU に正確で安定した電圧を供給する必要があります。
- PMIC に放電機能がある場合、PMIC は DeepSleep 電源モード中に電流を引き込み、消費電流を増加させます。
- VCCD 電源が、Active レギュレータから PMIC に切り替わると、PMIC の引き込み電流が MCU から流れるため電源のハンドオーバによって、電圧降下および過電流が発生する場合があります。
- MCU は PMIC EN 端子で PMIC 有効を制御します。
- MCU は、パワーグッド機能を使用して PMIC 電圧を検出します。

選定された PMIC 機能によって、PMIC と VCCD 端子間の絶縁用ロードスイッチの要否が決まります。表 17 に、PMIC 直接接続またはロードスイッチを使用した PMIC での電源構成での PMIC 機能要件を示します。

#### 4 PMIC (スイッチングレギュレータ)

表 17 PMIC 要件

機能	電源構成の必要機能		説明
	PMIC 直接接続	ロードスイッチを使用した PMIC	
出力電圧精度/電流能力	✓	✓	電圧および電流能力についてはデバイスデータシート [1] の “Recommended Operating Conditions for V <sub>CCD</sub> ” を参照してください。
放電機能なし	✓	対象外	意図しない消費電流を防ぐために PMIC が無効になっている場合、PMIC の放電機能を動作させないでください。
プリバイアス スタートとソフトスタート	✓	対象外	PMIC は MCU の Active レギュレータから電流を引き込んではいけません。
Enable	✓	✓	PMIC 出力を制御するために必要です。
パワーグッド (PG)	✓	✓	PMIC 出力状態を示すために必要です。

#### 放電機能なし

放電機能は、PMIC が無効の場合、V<sub>CCD</sub> とグランド間に抵抗 (例えば 100 オーム程度) を接続することによって、出力容量を放電します。したがって、DeepSleep 電源モードでこのような PMIC が無効になった場合、V<sub>CCD</sub> は放電されます。一部の PMIC は、低電圧ロックアウト(UVLO)の場合でも放電します。放電電流は意図しない電流消費を引き起こす可能性があるため、PMIC には放電機能を持たせないでください。

#### プリバイアススタートとソフトスタート

ソフトスタートは、ランプアップの開始ポイントでの印可電圧レベル (プリバイアススタート) に関係なく、一定のランプアップ時間でスイッチングが開始した場合、突入電流を防ぐために出力電圧を徐々に増加させる機能です。ソフトスタート動作には 2 つのタイプがあります。プリバイアスソフトスタートは、PMIC の出力電圧が Active レギュレータの電圧レベルに達するまで、MCU から電流を引き込まず、スイッチング動作を開始しません。強制 0V ソフトスタートは、V<sub>CCD</sub> 端子を介して MCU から電流を引き込むことにより、出力を 0V から強制的に開始します。使用する PMIC のデータシートを参照するか、PMIC ベンダに問い合わせることによりどちらのソフトスタートが採用されているか確認してください。図 14 に、各ソフトスタートのタイミングチャートを示します。

## 4 PMIC (スイッチングレギュレータ)

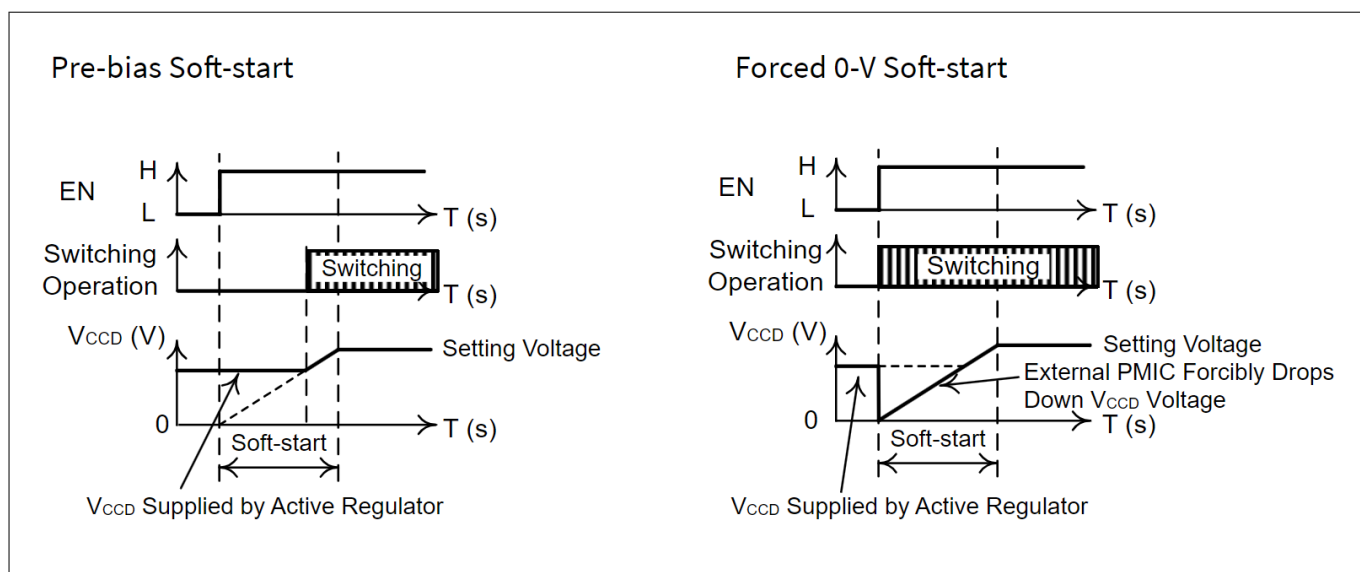


図 14 ソフトスタートタイミングチャート

**注:** 特定の PMIC は、ソフトスタート中に過電圧検出を有効にします。この場合、PMIC はソフトスタート中に過電圧を検出し、PMIC はスイッチングを開始しません。過電圧検出範囲外に達すると、PMIC はスイッチングを開始します。この許容範囲の下限值にて、PMIC は Active レギュレータから供給される VCCD をプルダウンします。このような PMIC は強制 0V ソフトスタートと同じ方法で処理する必要があります。

**注:** 起動時のプリバイアス動作について確認することを強く推奨します。わずかに電圧低下が発生する場合がありますが、電圧は内部 Active レギュレータの動作範囲内を維持する必要があります。

### パワーグッド (PG)

PG は、PMIC 無効中にパワーグッド状態を示してはいけません。さらに、ソフトスタート中、またはターゲット電圧が 100%に達する前に、PMIC は正しく電源供給できないため、この期間はパワーグッド状態を示さないものがより適します。この場合、PMIC が完全に動作する前に Active レギュレータがオフになる場合があります。PMIC が 100%に達する前のソフトスタート中にパワーグッド状態を示す場合、ソフトウェアは PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT を使用して Active レギュレータのオフを遅延させることができます。PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT については各電源構成でのソフトウェアフローを参照してください。

## 4.2 推奨する PMIC トポロジ

表 18 に各ユースケースでの推奨する PMIC トポロジを示します。

表 18 PMIC 要件

ユースケース	トポロジ	利点
DeepSleep 電源モード中に PMIC 無効 ケース 1, ケース 2	強制 PWM モード	スイッチングリップル電圧が小さく、 負荷応答による電圧変動が小さい。 固定スイッチング周波数。
DeepSleep 電源モード中に PMIC 有効 ケース 3, ケース 4	自動 PFM/PWM モード	DeepSleep 電源モード中に VCCD に 外部 PMIC から供給電力を低減する

## 4 PMIC (スイッチングレギュレータ)

### 強制 PWM モードと自動 PWM/PFM モード

強制 PWM モードは、常に PWM 動作のみで動作します。

自動 PWM/PFM モードは、PMIC が負荷状態に応じて PWM 動作と PFM 動作を自動的に切り替える動作です。

このモードでは、高負荷電流状態では PWM で動作し、低負荷電流状態では PFM で動作します。

PWM 動作は、一定の周波数で動作します。スイッチング FET オンのデューティ比を変化させることで、出力電圧を制御します。負荷応答特性も PFM 動作より優れており、出力リップル電圧は PFM 動作より小さいです。さらに、スイッチング周波数が固定されているため、EMI ノイズ解析も PFM 動作と比較して容易です。

PFM 動作は、負荷電流に応じてスイッチング周波数を変更し、出力電圧を制御します。PFM 動作の利点は、低負荷電流状態で変換効率を向上できることです。

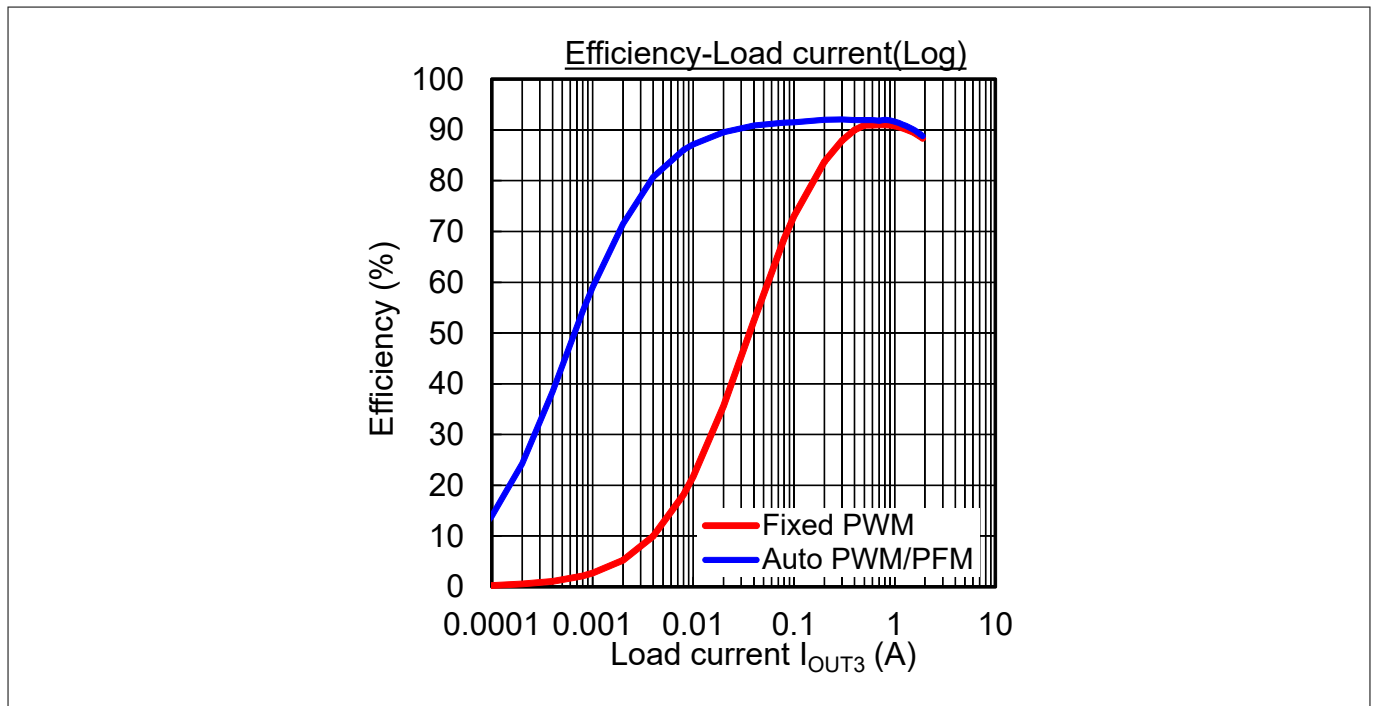


図 15 変換効率の例

**注:** PFM 動作ではハイサイドスイッチのみがアクティブのため、PMIC は電流を引き込みません。したがって、PMIC がソフトスタート中にも PFM モード動作できる場合、ソフトスタート機能はプリバイアスソフトスタートと同様に動作します。ただし、PMIC の自動 PWM/PFM モードは、ソフトスタート時に動作することを確認してください。

**注:** PFM 動作の PMIC は PWM 動作よりも電圧精度が低くなります。

### 4.3 PMIC 直接接続

ここでは、PMIC 直接接続について説明します。内部レギュレータの構成に応じて以下のユースケースについて、PMIC の接続、ハンドオーバータイミングチャート、およびハンドオーバーソフトウェアフロー例を示します。

- **ケース 1:** Active 電源モードで OCD を使用する、DeepSleep 電源モードで PMIC 無効にする
- **ケース 2:** Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 無効にする
- **ケース 3:** Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Blocking Call で SwitchOverRegulators API を使用する)
- **ケース 4:** Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Non-blocking Call で SwitchOverRegulators API を使用する)

## 4 PMIC (スイッチングレギュレータ)

詳細については、[PMIC 使用時の内部レギュレータ設定](#)を参照してください。

**注:** 外部電源の過電流検出にOCD 機能を使用することは推奨しません。OCD 機能はActive レギュレータの一部のため直接VCCD の電源ラインに実装されていません。

**注:** OCD を使用するケースは、Active モードで外部 PMIC 電源によるMCU 動作のみの設定です。OCD 機能は内部レギュレータの一部であるため、例えばPOR 後は有効になります。

**注:** ケース3 では、PMIC へのハンドオーバー後に内部レギュレータに戻すことはできません。内部レギュレータに戻したい場合は、WDT を使用するか、外部システム制御のXRES\_L トリガ要求によって、電源システムを含むチップ全体のリセットをする必要があります。

### 4.3.1 ハードウェア構成

図 16 に、PMIC 直接接続時の回路図を示します。

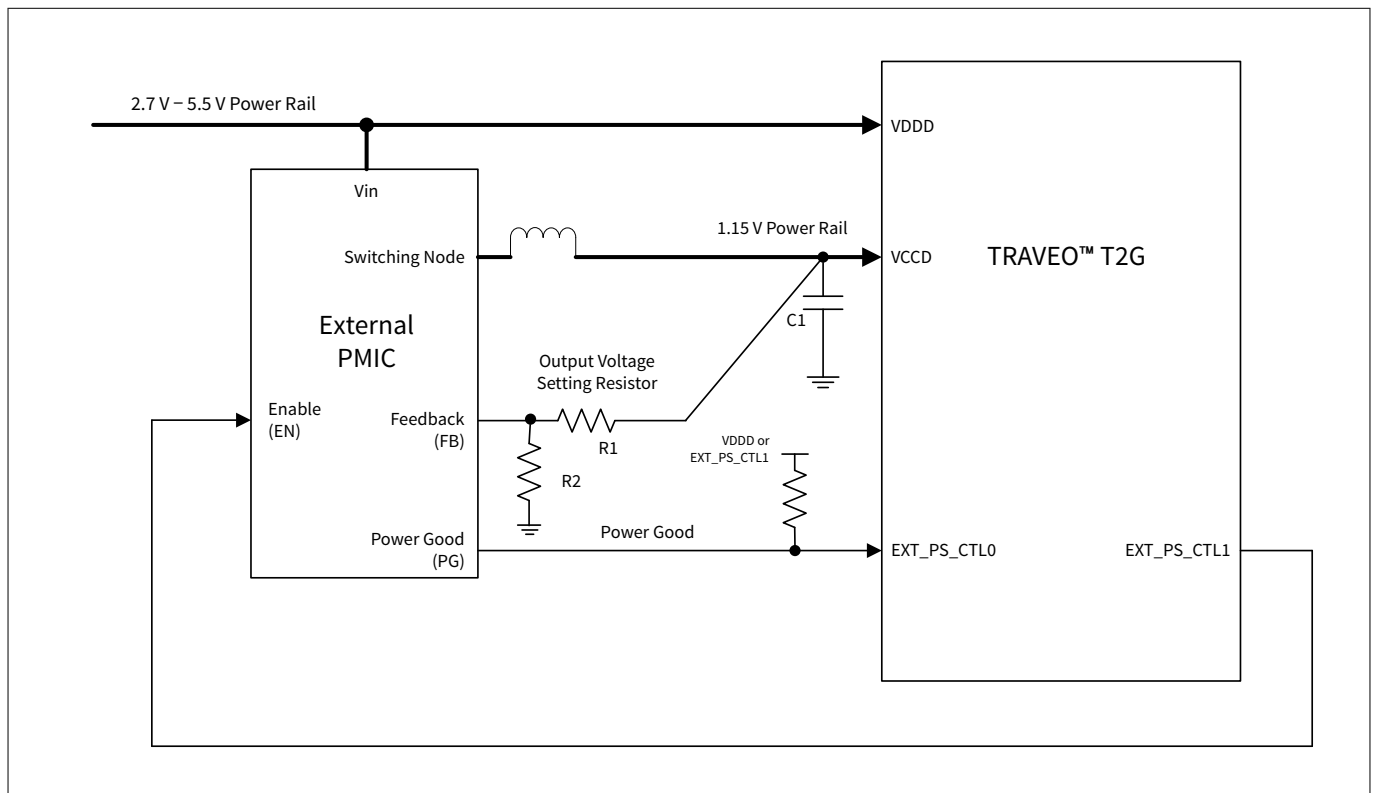


図 16 PMIC 直接接続時の回路図

- PMIC EN の極性は、HIGH で有効です。PMIC PG の極性は、HIGH でパワーグッド状態を示します。RESET\_PMIC は、PMIC の PG が LOW になると生成されます。これは、HV リセットで発行され、PMIC および REGHC または PMIC コントローラ含む MCU と PMIC が初期化されます。
- PMIC の FB 端子は、V<sub>CCD</sub> に接続されます。
- 選択した PMIC に応じて、出力電圧設定抵抗が必要です。
- PMIC の EN 端子は、MCU の EXT\_PS\_CTL1 端子に接続されます。XRES および Hibernate 中に PMIC を無効にするためには、PCB または PMIC にプルダウン抵抗が必要です。
- EN 端子に内部プルダウン抵抗がない場合、POR 中に PMIC を無効にするため外部プルダウン抵抗が必要となります。



## 4 PMIC (スイッチングレギュレータ)

- $V_{CCD}$  コンデンサ (C1) は、MCU 要件に依存します。(静電容量については、デバイスデータシート [1]を参照してください)
- 出力電圧設定抵抗 (R1, R2) は、選択した PMIC に依存します。

ここでは、PMIC イネーブル(EN)端子極性が”1”、PG 状態極性が”1”の場合のソフトウェアフローを説明します。

ConfigureRegulator API の Enable Polarity は PMIC への有効信号の極性を指定し、ConfigureRegulator API の Reset Polarity は PMIC からの PG 状態の極性を指定します。Reset Polarity ビットフィールドは、PG 信号のディアサージョンレベルに対応することに注意してください。

### 4.3.2 ケース 1

ここでは、Active 電源モードで OCD を使用する、DeepSleep 電源モードで PMIC 無効にするケースについて説明します。

**注:** 監視は  $V_{CCD}$  電源ラインの直接シャント実装ではないため、外部電源の過電流監視に OCD を使用することは推奨しません。これは Active レギュレータの不可欠な部分です。

#### 4.3.2.1 ハンドオーバータイミングチャート

ケース 1 では、Active レギュレータは有効 (OCD 使用)、DeepSleep レギュレータは有効 (DeepSleep 電源モードで DeepSleep レギュレータ有効) です。図 17 に、ハンドオーバーシーケンス例を示します。この例は、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバ、DeepSleep 電源モードへの移行、DeepSleep 電源モードからのウェイクアップ、そして PMIC から内部レギュレータへのハンドオーバが含まれます。

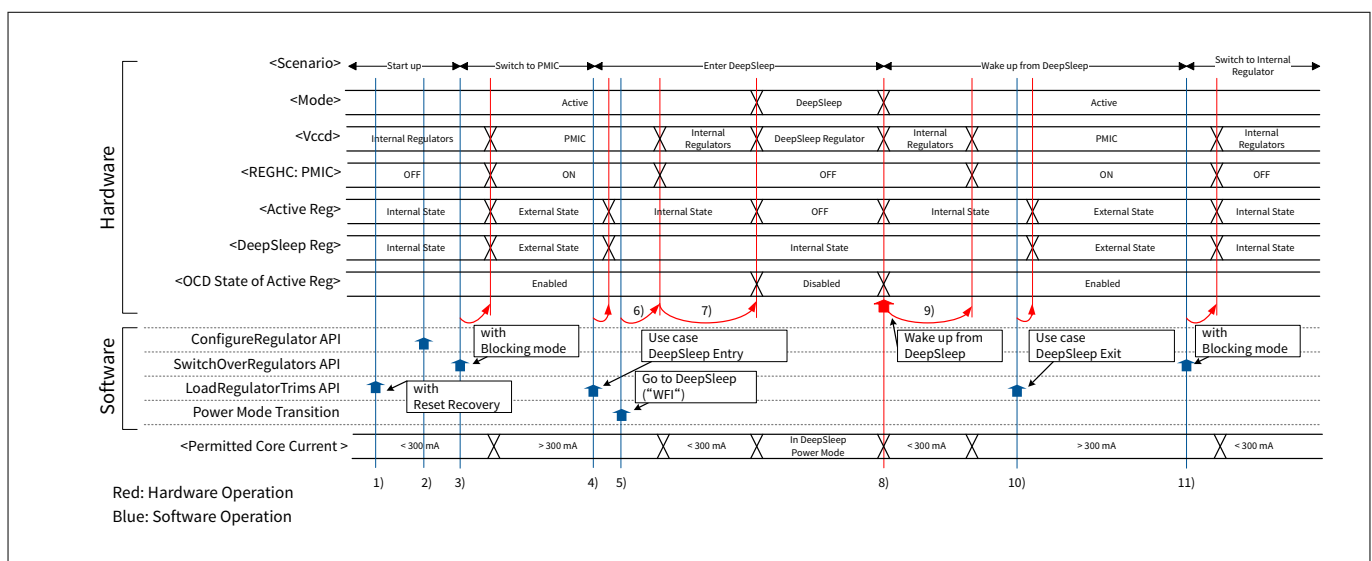


図 17 ハンドオーバーシーケンス例 (ケース 1)

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が”1”に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、Reset Recovery を参照してください。
2. 起動後、PMIC で REGHC または PMIC コントローラを設定するために ConfigureRegulator API を呼び出してください。
3. 内部レギュレータから PMIC へハンドオーバするため SwitchOverRegulators API を呼び出してください。PMIC は有効になります。Active レギュレータは OCD 機能により有効ですが外部状態に変更されます。DeepSleep レギュレータは外部状態に変更されます。
4. 内部レギュレータを内部状態に変更するために DeepSleep Entry ユースケースで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは内部状態に変更されます。



#### 4 PMIC (スイッチングレギュレータ)

5. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。その後、WFI 命令を使用して DeepSleep 電源モードへの移行を実行できます。
6. ソフトウェアが、DeepSleep 電源モードへの移行を実行すると、ハードウェアは PMIC から内部レギュレータに切り替えます。
7. MCU は内部レギュレータへの切り替えが完了すると DeepSleep 電源モードへ遷移します。Active レギュレータはオフになります。
8. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生し、その後、Active レギュレータは内部状態で開始されます。
9. Active 電源モードへの移行が完了すると、PMIC が有効になります。
10. 内部レギュレータを外部状態に変更するために DeepSleep Exit ユースケースで LoadRegulatorTrims API を呼び出してください。  
Active と DeepSleep レギュレータは外部状態に変更されます。
11. PMIC から内部レギュレータにハンドオーバーするため SwitchOverRegulators API を呼び出してください。  
PMIC は無効になります。Active および DeepSleep レギュレータは内部状態に変更されます。

#### 4.3.2.2 ソフトウェアフロー

##### 内部レギュレータから PMIC へのハンドオーバー

図 18 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。

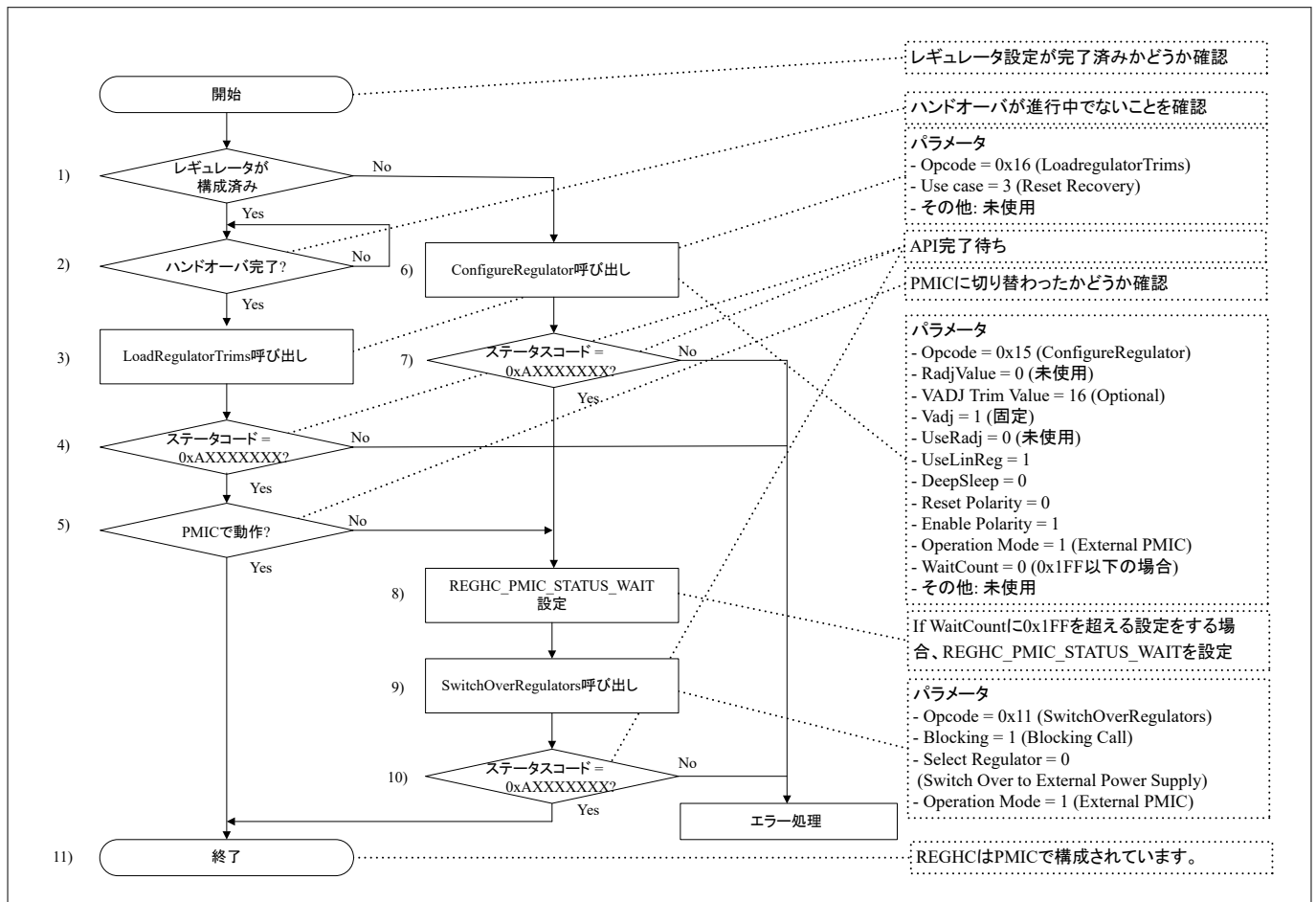


図 18 内部レギュレータから PMIC へのハンドオーバーフロー (ケース 1)

1. レギュレータ設定が完了済みかどうか確認してください。既に REGHC または PMIC コントローラがセットアップされている場合、再設定なしで有効にできます。

#### 4 PMIC (スイッチングレギュレータ)

- PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(6)へ進んでください。
2. ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
  3. Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
    - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
    - Use case = 3 (Reset Recovery)
  4. LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしがって適切なエラー処理をしてください。
  5. MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(11)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(8)へ進んでください。
  6. 以下のパラメータで ConfigureRegulator API を呼び出してください。
    - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
    - RadjValue\*1 = 0 (未使用)
    - VADJ trim value = 16
    - Vadj = 1 (固定)
    - UseRadj<sup>2)</sup> = 0 (未使用)
    - UseLinReg = 1 (PMIC イネーブル後も内部 Active レギュレータは有効です)
    - DeepSleep<sup>2)</sup>\*1 = 0 (PMIC は DeepSleep 電源モードで無効です)
    - Reset Polarity = 0 (PMIC PG 信号は、“0” でパワーバッド状態を示します)
    - Enable Polarity = 1 (PMIC は“1”で有効です)
    - Operating Mode<sup>2)</sup> = 1 (外部 PMIC)
    - WaitCount = 0 (0x1FF 以下)
  7. ConfigureRegulator API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
  8. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。  
以下の手順に従って、PMIC で REGHC または PMIC コントローラを有効にします。
  9. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
    - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
    - Blocking = 1 (Blocking call)
    - Select regulator = 0 (外部電源に切り替えます)
    - Operating Mode\*1 = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。
  10. SwitchOverRegulators API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
  11. デバイスは PMIC で動作します。Active および DeepSleep レギュレータは外部状態で動作します。

#### DeepSleep への遷移と復帰

[図 19](#) に、DeepSleep 電源モードへの遷移を示します。

<sup>2)</sup> CYT3D/4D シリーズでは無効です。

#### 4 PMIC (スイッチングレギュレータ)

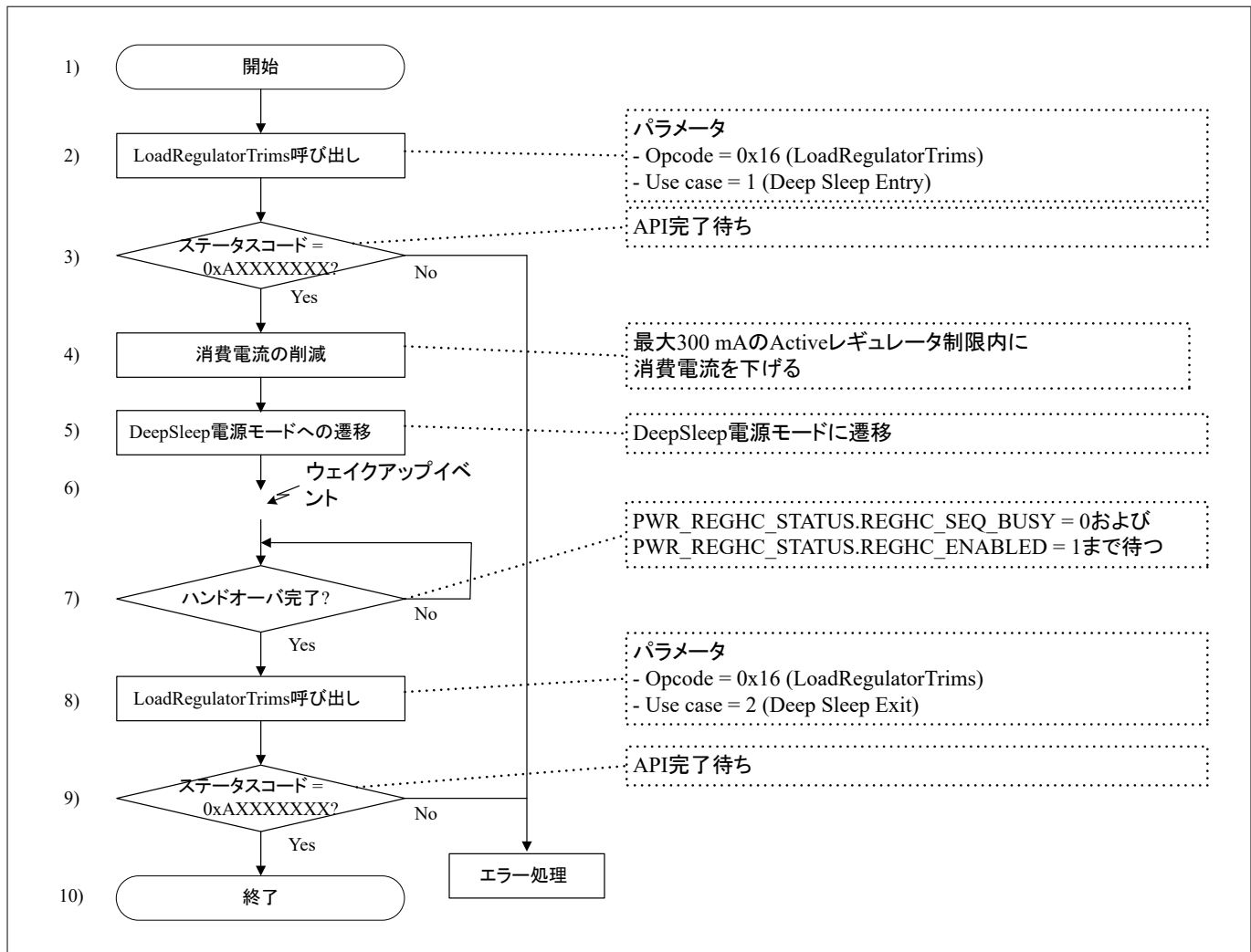


図 19 DeepSleep への遷移と復帰フロー (ケース 1)

1. MCU は PMIC から電源供給されます。
2. DeepSleep 電源モード移行前に以下のパラメータで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 1 (DeepSleep Entry)
3. LoadRegulatorTrims API の完了を待ってください。ステータスコードが "0xAXXXXXXX" 以外の場合、システムに応じて適切なエラー処理をしてください。
4. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
5. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。内部レギュレータへのハンドオーバーを実行し、PMIC を無効にします。次に Active レギュレータは無効、DeepSleep レギュレータは DeepSleep 中に電源供給します。  
MCU は DeepSleep レギュレータから電源供給されます。
6. ウェイクアップイベントのため、MCU は Active 電源モードに遷移します。次に、ハードウェアが PMIC を有効にします。
7. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = "0" および PWR\_REGHC\_STATUS.REGHC\_ENABLED = "1" まで待ってください。

## 4 PMIC (スイッチングレギュレータ)

8. 以下のパラメータで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは外部状態に設定されます。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 2 (DeepSleep Exit)
9. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
10. MCU は PMIC から電源供給されます。

### PMIC から内部レギュレータへのハンドオーバー

図 20 に、PMIC から内部レギュレータへのハンドオーバーを示します。

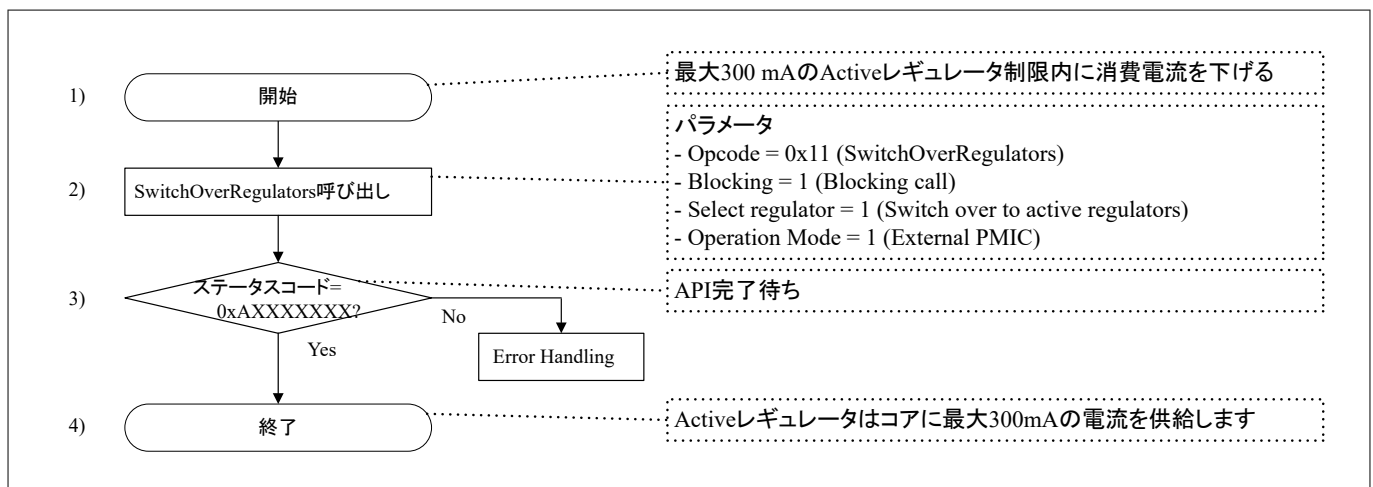


図 20 PMIC から内部レギュレータへのハンドオーバーフロー (ケース 1)

PMIC から内部レギュレータにハンドオーバーする場合

1. 最大 300 mA の Active レギュレータ制限内まで消費電流を下げてください。(クロック、コア、ペリフェラルなどの適切なアプリケーション構成によって)
2. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode<sup>3)</sup> = 1 (外部 PMIC)
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. デバイスは、内部レギュレータで動作しています。

### 4.3.3 ケース 2

ここでは、Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 無効にするケースについて説明します。

#### 4.3.3.1 ハンドオーバータイミングチャート

ケース 2 では、PMIC が MCU に電源供給中は、Active レギュレータは無効 (OCD 未使用)、そして DeepSleep 電源モードで DeepSleep レギュレータは有効 (つまり PMIC は無効) です。図 21 に、ハンドオーバーシーケンスの例を

<sup>3</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

示します。この例は、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバー、DeepSleep 電源モードへの移行、および DeepSleep 電源モードからのウェイクアップが含まれます。

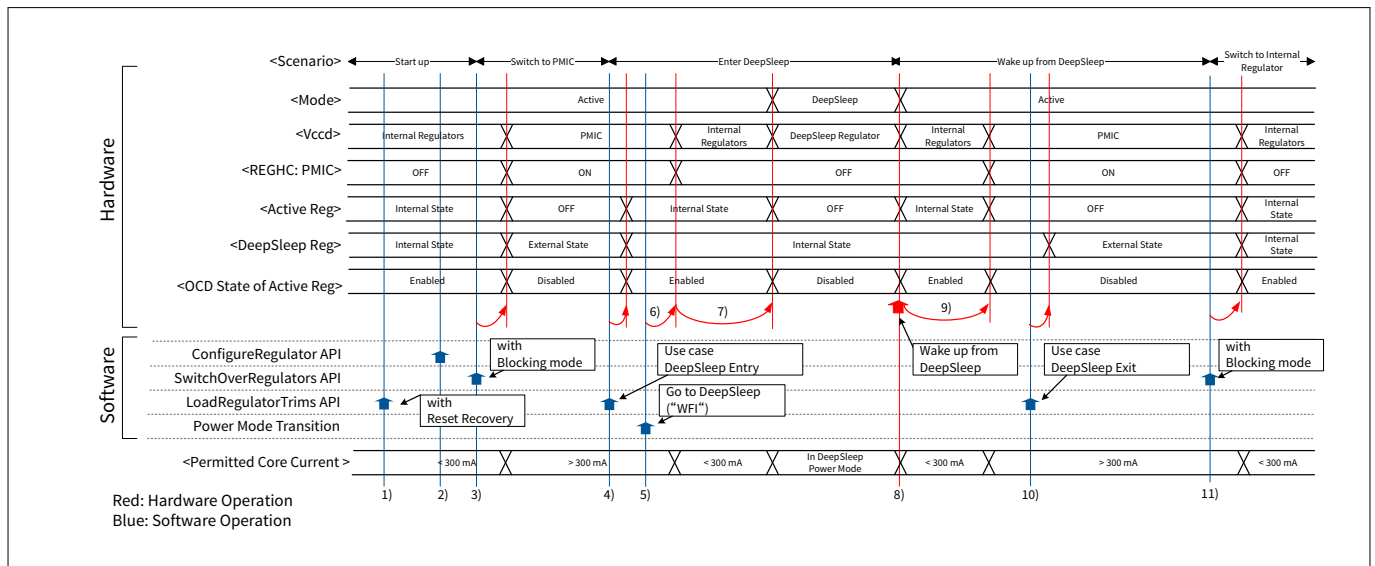


図 21 ハンドオーバーシーケンス例 (ケース 2)

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が"1"に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、PMIC で REGHC または PMIC コントローラを構成するために ConfigureRegulator API を呼び出してください。
3. 内部レギュレータから PMIC へハンドオーバーするため SwitchOverRegulators API を呼び出してください。PMIC は有効になります。OCD 機能を使用しないため、Active レギュレータは無効です。DeepSleep レギュレータは外部状態に変更されます。
4. 内部レギュレータを内部状態に変更するために DeepSleep Entry ユースケースで LoadRegulatorTrims API を呼び出してください。  
Active および DeepSleep レギュレータは、DeepSleep 電源モード遷移前に内部状態に変更されます。
5. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。その後、WFI 命令を使用して DeepSleep 電源モードへの移行を実行します。
6. ソフトウェアが、DeepSleep 電源モードへの移行を実行すると、ハードウェアは PMIC から内部レギュレータに切り替えます。
7. MCU は内部レギュレータへの切り替えが完了すると DeepSleep 電源モードへ遷移します。DeepSleep 電源モードへ遷移後、Active レギュレータはオフになります。
8. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生し、その後、Active レギュレータは内部状態で開始されます。
9. Active 電源モードへの移行が完了すると、PMIC が有効、Active レギュレータは無効になります。
10. レギュレータの外部状態を変更するために DeepSleep Exit ユースケースで LoadRegulatorTrims API を呼び出してください。DeepSleep レギュレータは外部状態に変更されます。
11. PMIC から内部レギュレータにハンドオーバーするため SwitchOverRegulators API を呼び出してください。PMIC は無効になります。Active および DeepSleep レギュレータは内部状態に変更されます。

### 4.3.3.2 ソフトウェアフロー

#### 内部レギュレータから PMIC へのハンドオーバー

図 22 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。



## 4 PMIC (スイッチングレギュレータ)

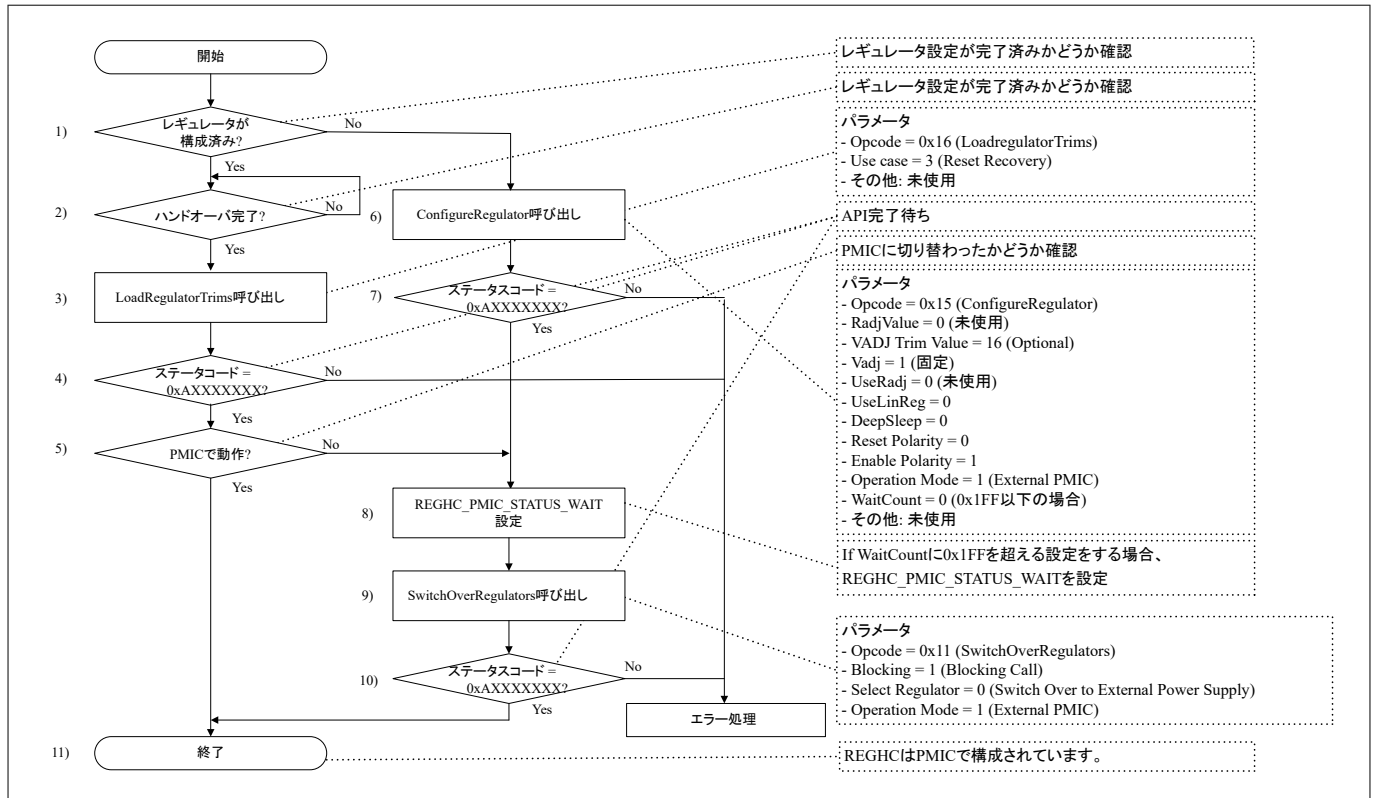


図 22 内部レギュレータから PMIC へのハンドオーバーフロー (ケース 2)

- レギュレータ設定が完了済みかどうか確認してください。既に REGHC または PMIC コントローラがセットアップされている場合、再設定なしで有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(6)へ進んでください。
- ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
- Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
- LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしがって適切なエラー処理をしてください。
- MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(11)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(8)へ進んでください。
- 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
  - RadjValue<sup>4)</sup> = 0 (未使用)
  - VADJ trim value = 16
  - Vadj = 1 (固定)
  - UseRadj<sup>4)</sup> = 0 (未使用)
  - UseLinReg = 0 (PMIC イネーブル後、内部 Active レギュレータは無効です)
  - DeepSleep<sup>4)</sup> = 0 (PMIC は DeepSleep 電源モード中は無効です)

<sup>4</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

- Reset Polarity = 0 (PMIC PG 信号は、"0" でパワーバッド状態を示します)
  - Enable Polarity = 1 (PMIC は"1"で有効です)
  - Operating Mode<sup>4</sup> = 1 (外部 PMIC)
  - WaitCount = 0 (0x1FF 以下)
7. ConfigureRegulator API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
  8. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。  
以下の手順に従って、PMIC で REGHC または PMIC コントローラを有効にします。
  9. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
    - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
    - Blocking = 1 (Blocking call)
    - Select regulator = 0 (外部電源に切り替えます)
    - Operating Mode = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。
  10. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
  11. デバイスは PMIC で動作します。Active および DeepSleep レギュレータは外部状態で動作します。

### DeepSleep への遷移と復帰

[図 23](#) に、DeepSleep 電源モードへの遷移を示します。

<sup>4</sup> CYT3D/4D シリーズでは無効です。



#### 4 PMIC (スイッチングレギュレータ)

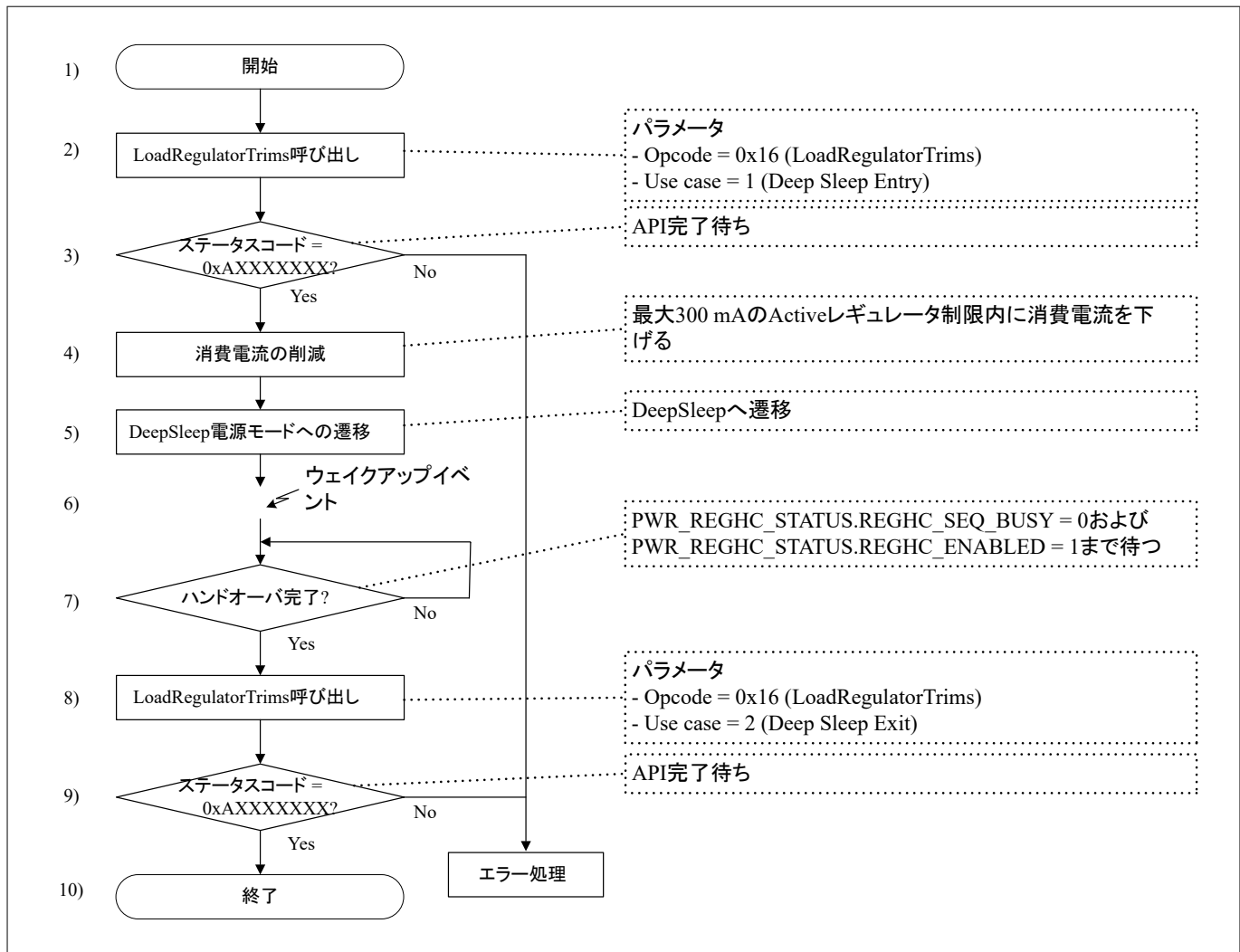


図 23 DeepSleep への遷移と復帰フロー (ケース 2)

1. MCU は PMIC から電源供給されます。
2. DeepSleep 電源モード移行前に以下のパラメータで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 1 (DeepSleep Entry)
3. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
5. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。内部レギュレータへのハンドオーバーを実行し、PMIC を無効にします。次に Active レギュレータは無効、DeepSleep レギュレータは DeepSleep 中に電源供給します。  
MCU は DeepSleep レギュレータから電源供給されます。
6. ウェイクアップイベントのため、MCU は Active 電源モードに遷移します。次に、ハードウェアが PMIC を有効にします。
7. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = "0" および PWR\_REGHC\_STATUS.REGHC\_ENABLED = "1"まで待ってください。

## 4 PMIC (スイッチングレギュレータ)

8. 以下のパラメータで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは外部状態に設定されます。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 2 (DeepSleep Exit)
9. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
10. MCU は PMIC から電源供給されます。

### PMIC から内部レギュレータへのハンドオーバー

図 24 に、PMIC から内部レギュレータへのハンドオーバーを示します。

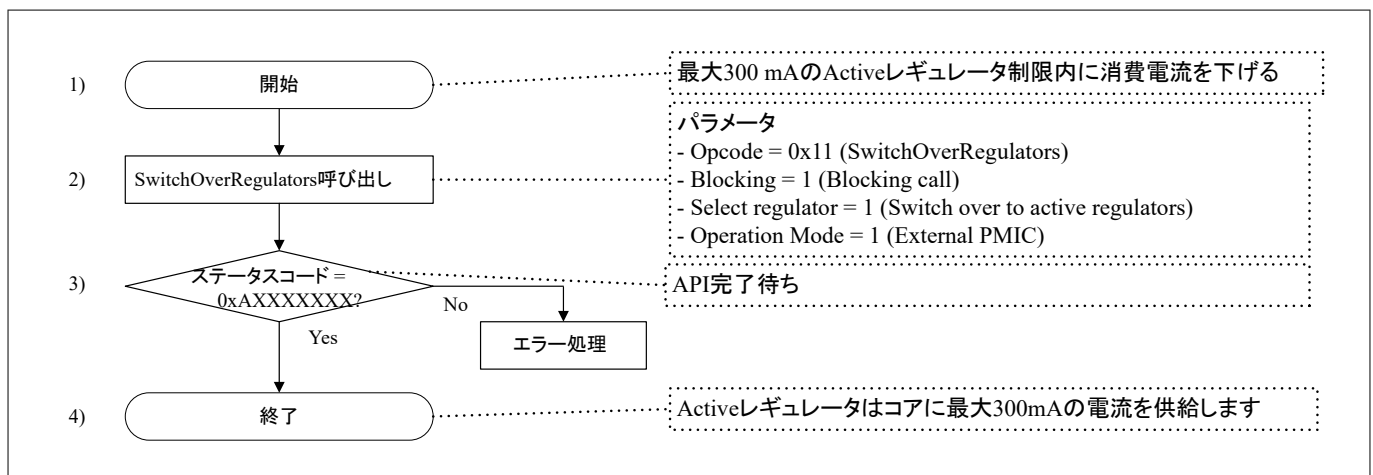


図 24 PMIC から内部レギュレータへのハンドオーバーフロー (ケース 2)

PMIC から内部レギュレータにハンドオーバーする場合

1. 最大 300 mA の Active レギュレータ制限内まで消費電流を下げてください。(クロック、コア、ペリフェラルなどの適切なアプリケーション構成によって)
2. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode<sup>5)</sup> = 1 (外部 PMIC)
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. デバイスは、内部レギュレータで動作しています。

### 4.3.4 ケース 3

ここでは、Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Blocking Call で SwitchOverRegulators API を使用する) ケースについて説明します。

#### 4.3.4.1 ハンドオーバータイミングチャート

ケース 3 では、Active レギュレータは無効 (OCD 未使用)、そして DeepSleep レギュレータは無効です (DeepSleep 電源モード中、PMIC が電源供給します)。図 25 に、ハンドオーバーシーケンス例を示します。この例は、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバー、DeepSleep 電源モードへの移行、および

<sup>5</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

DeepSleep 電源モードからのウェイクアップが含まれます。この設定では、API を使用して PMIC から内部レギュレータに戻すことはできません。内部レギュレータで動作させる場合、チップ全体をリセットする必要があります (HV リセット)。

システムによって SwitchOverRegulators API を使用して PMIC から内部レギュレータへハンドオーバーする必要がある場合、PMIC へのハンドオーバー時に Non-blocking call で SwitchOverRegulators API を呼び出してください。[ケース 4](#) 参照。

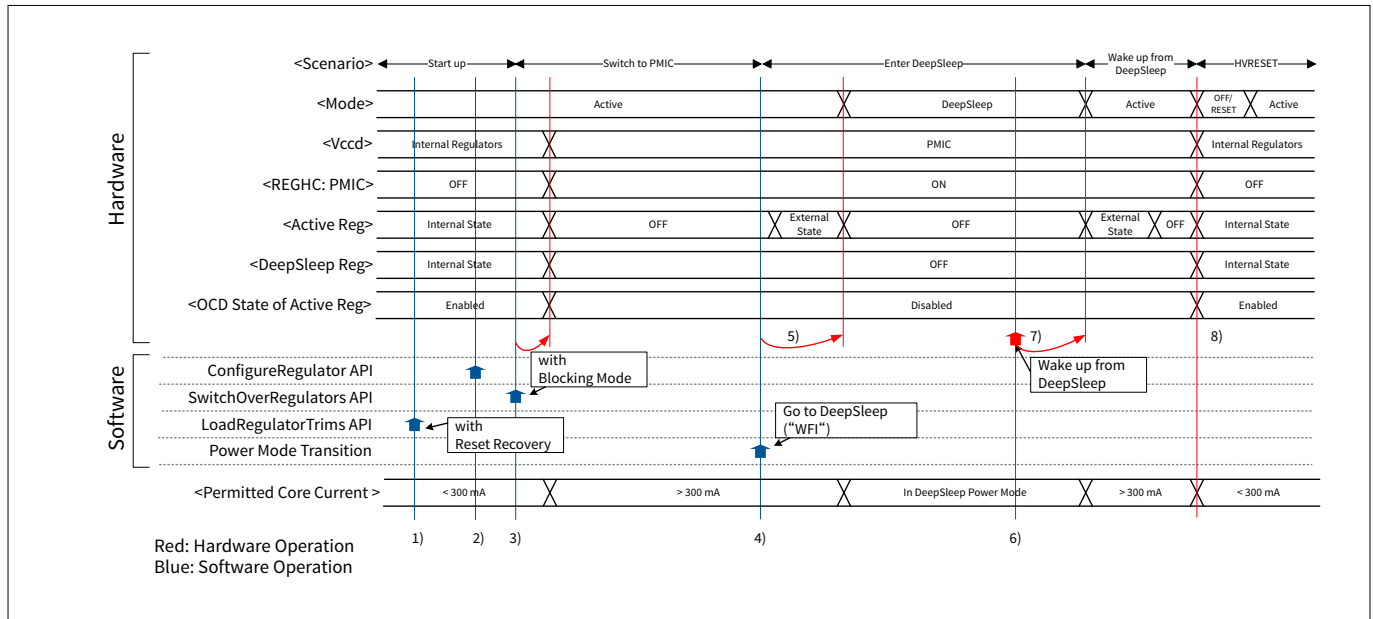


図 25 ハンドオーバーシーケンス例 (ケース 3)

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が "1" に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、PMIC で REGHC または PMIC コントローラを構成するために ConfigureRegulator API を呼び出してください。
3. 内部レギュレータから PMIC へハンドオーバーするため SwitchOverRegulators API を呼び出してください。  
**注:** Blocking call で SwitchOverRegulators API を呼び出した場合、API は PWR\_CTL2.DPSLP\_REG\_DIS を "1" に設定し、PMIC から内部レギュレータに戻すことはできません。

PMIC は有効になります。OCD は使用されず、DeepSleep 電源モードで DeepSleep レギュレータは無効のため Active および DeepSleep レギュレータは無効になります。

4. WFI 命令を使用して DeepSleep 電源モードへの遷移を実行してください。この場合、システムは消費電流の削減なしに DeepSleep 電源モードに移行できます。PMIC は引き続き電源を供給します。
5. MCU は DeepSleep 電源モードへ移行します。  
この場合、SwitchOverRegulators API を呼び出した後、Active と DeepSleep レギュレータは無効になります。電源は常に PMIC から供給されます。したがって、DeepSleep 電源モードまたは Active 電源モードへの遷移に LoadRegulatorTrims API は不要です。
6. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生します。Active および DeepSleep レギュレータは無効です。
7. MCU は Active 電源モードに移行し、PMIC は引き続き電源を供給します。
8. HV リセットが発生すると、電源システムは初期化されます。したがって、リセット解除後、MCU は Active レギュレータで起動します。

## 4 PMIC (スイッチングレギュレータ)

### 4.3.4.2 ソフトウェアフロー

#### 内部レギュレータから PMIC へのハンドオーバー

図 26 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。

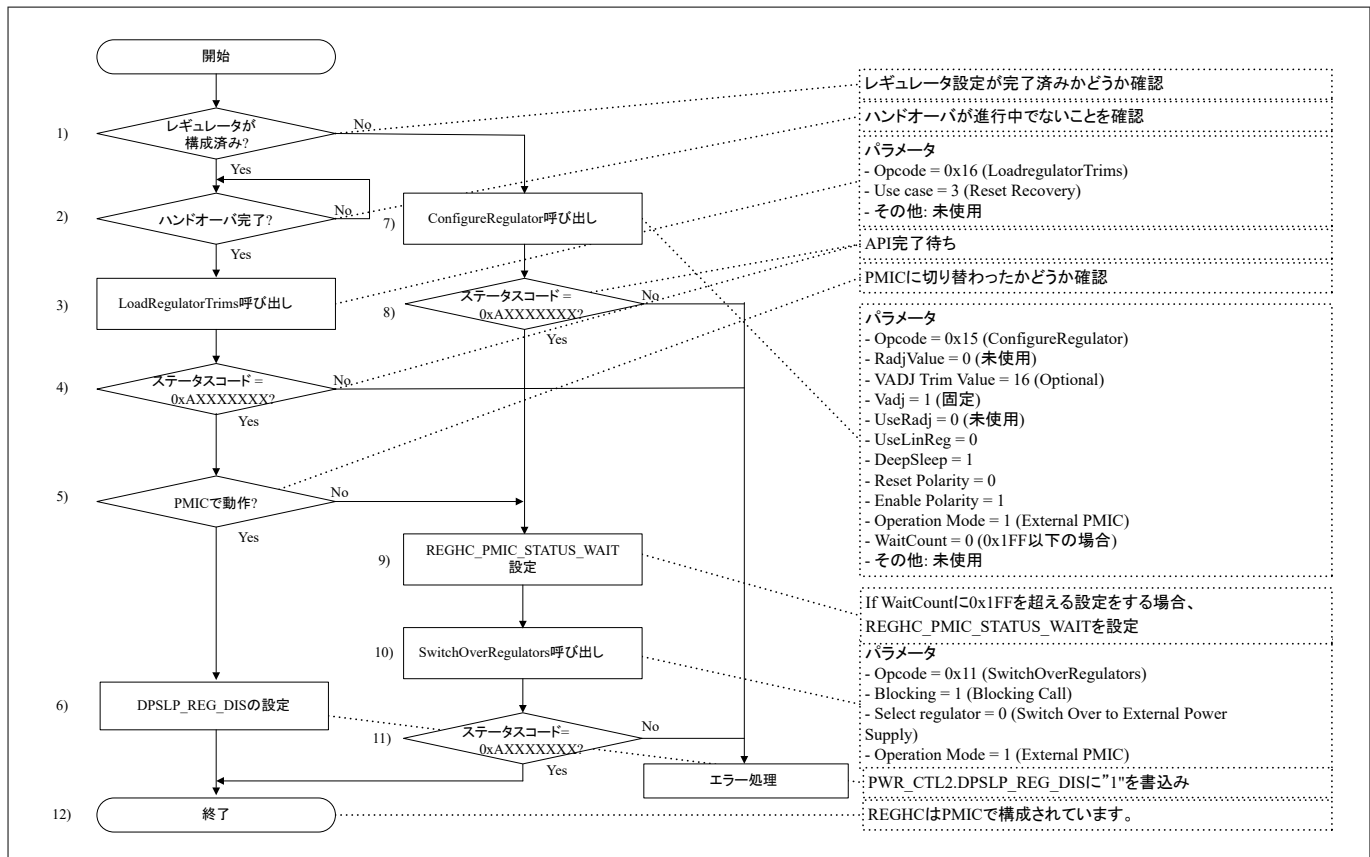


図 26 内部レギュレータから PMIC へのハンドオーバーフロー (ケース 3)

- レギュレータ設定が完了済みかどうか確認してください。既に REGHC または PMIC コントローラがセットアップされている場合、再設定なしで有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(7)へ進んでください。
- ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
- Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
- LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしたがって適切なエラー処理をしてください。
- MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(6)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(9)へ進んでください。
- PWR\_CTL2.DPSLP\_REG\_DIS を“1”に設定してください。  
ケース 3 では、PMIC は DeepSleep 電源モードで有効です。したがって、PMIC へのハンドオーバーが完了している場合、ユーザソフトウェアは PWR\_CTL2.DPSLP\_REG\_DIS を“1”に設定する必要があります。そうでない場合、Blocking call での SwitchOverRegulators API により、PWR\_CTL2.DPSLP\_REG\_DIS が設定されます。

## 4 PMIC (スイッチングレギュレータ)

7. 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
  - RadjValue<sup>6)</sup> = 0 (未使用)
  - VADJ trim value = 16
  - Vadj = 1 (固定)
  - UseRadj<sup>6)</sup> = 0 (未使用)
  - UseLinReg = 0 (PMIC イネーブル後、内部 Active レギュレータは無効です)
  - DeepSleep<sup>6)</sup> = 1 (PMIC は DeepSleep 電源モード中に有効です)
  - Reset Polarity = 0 (PMIC PG 信号は、“0” でパワーバッド状態を示します)
  - Enable Polarity = 1 (PMIC は“1”で有効です)
  - Operating Mode<sup>6)</sup> = 1 (外部 PMIC)
  - WaitCount = 0 (0x1FF 以下)
8. ConfigureRegulator API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
9. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。  
以下の手順に従って、PMIC で REGHC または PMIC コントローラを有効にします。
10. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 0 (外部電源に切り替えます)
  - Operating Mode<sup>6)</sup> = 1 (外部 PMIC)このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。

**注:** この場合、SwitchOverRegulators API は、PWR\_CTL2.DPSLP\_REG\_DIS を“1”に設定します。また、PMIC から内部レギュレータに戻すことはできません。システムが SwitchOverRegulators API によって PMIC から内部レギュレータへのハンドオーバーを実行する必要がある場合、Non-blocking call での SwitchOverRegulators API 呼び出しによって PMIC へのハンドオーバー実行してください。詳細については、[SwitchOverRegulators API の Non-blocking call 処理](#)を参照してください。
11. SwitchOverRegulators API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
12. デバイスは PMIC で動作します。Active および DeepSleep レギュレータは無効です。

### DeepSleep への遷移と復帰

[図 27](#) に、DeepSleep 電源モードへの遷移を示します。このケースでは、追加操作なしに、DeepSleep 電源モードに遷移できます。

<sup>6)</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

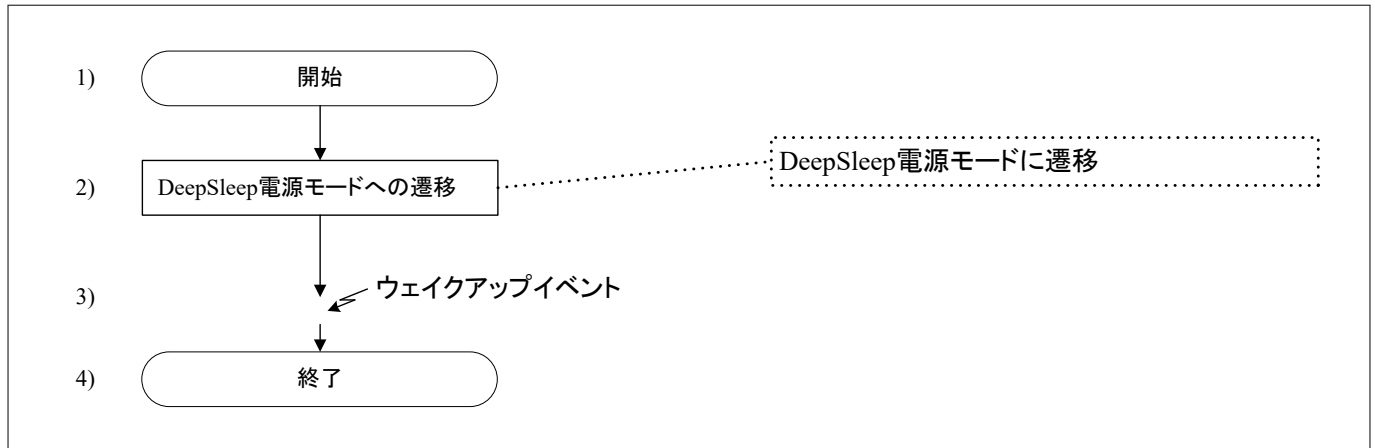


図 27 DeepSleep への遷移と復帰フロー (ケース 3)

1. MCU は PMIC から電源供給されます。
2. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。Active レギュレータは無効のまま、PMIC は引き続き電源を供給します。  
MCU は PMIC から電源供給されます。
3. ウェイクアップイベントのため、MCU は Active 電源モードに遷移します。Active および DeepSleep レギュレータは無効のままです。
4. MCU は PMIC から電源供給されます。

### 4.3.5 ケース 4

ここでは、Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Non-blocking Call での SwitchOverRegulators 使用) ケースについて説明します。

#### 4.3.5.1 ハンドオーバータイミングチャート

ケース 4 では、Active レギュレータは無効 (OCD 未使用)、そして DeepSleep レギュレータは有効、DeepSleep 電源モード中に PMIC が電源供給します。図 28 に、ハンドオーバーシーケンス例を示します。この例は、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバー、および PMIC から内部レギュレータへのハンドオーバーが含まれます。



## 4 PMIC (スイッチングレギュレータ)

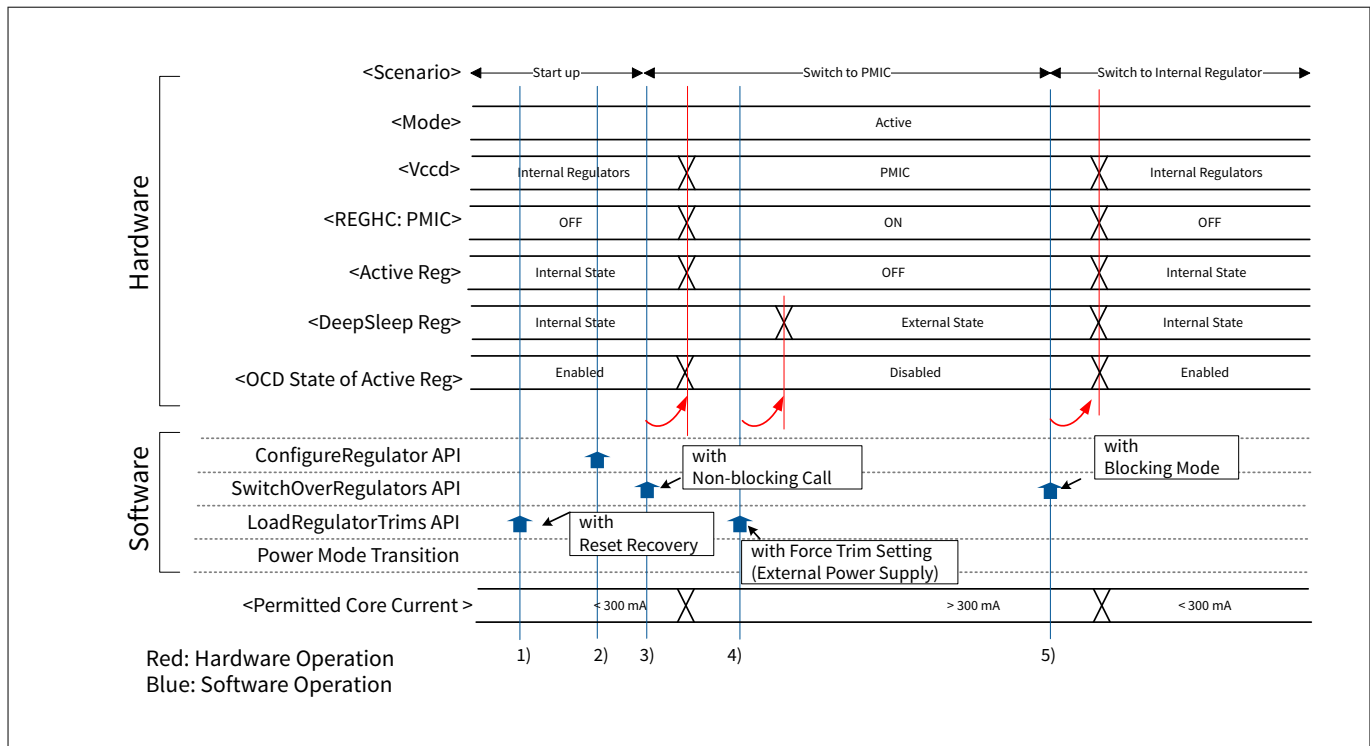


図 28 ハンドオーバーシーケンス例 (ケース 4)

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が”1”に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、PMIC で REGHC または PMIC コントローラを設定するために ConfigureRegulator API を呼び出してください。
3. 内部レギュレータから PMIC へハンドオーバーするため Non-blocking call で SwitchOverRegulators API を呼び出してください。

**注:** Non-blocking call で SwitchOverRegulators API を呼び出した場合、PWR\_CTL2.DPSLP\_REG\_DIS を”1”に設定しません。したがって、SwitchOverRegulators API を使用して、PMIC から内部レギュレータに戻すことができます。ただし、アプリケーションソフトウェアによって PWR\_CTL2.DPSLP\_REG\_DIS を”1”に設定した場合、PMIC から内部レギュレータに戻すことはできません。詳細については、[SwitchOverRegulators API の Non-blocking call 処理](#) を参照してください。

PMIC は有効になります。OCD 機能を使用しないため、Active レギュレータは無効です。

4. レギュレータを外部状態に変更するために Force trim setting ユースケースで LoadRegulatorTrims API を呼び出してください。DeepSleep レギュレータは外部状態に変更されます。
5. PMIC から内部レギュレータにハンドオーバーするため SwitchOverRegulators API を呼び出してください。PMIC は無効になります。Active および DeepSleep レギュレータは内部状態に変更されます。

### 4.3.5.2 ソフトウェアフロー

内部レギュレータから PMIC へのハンドオーバー

図 29 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。



#### 4 PMIC (スイッチングレギュレータ)

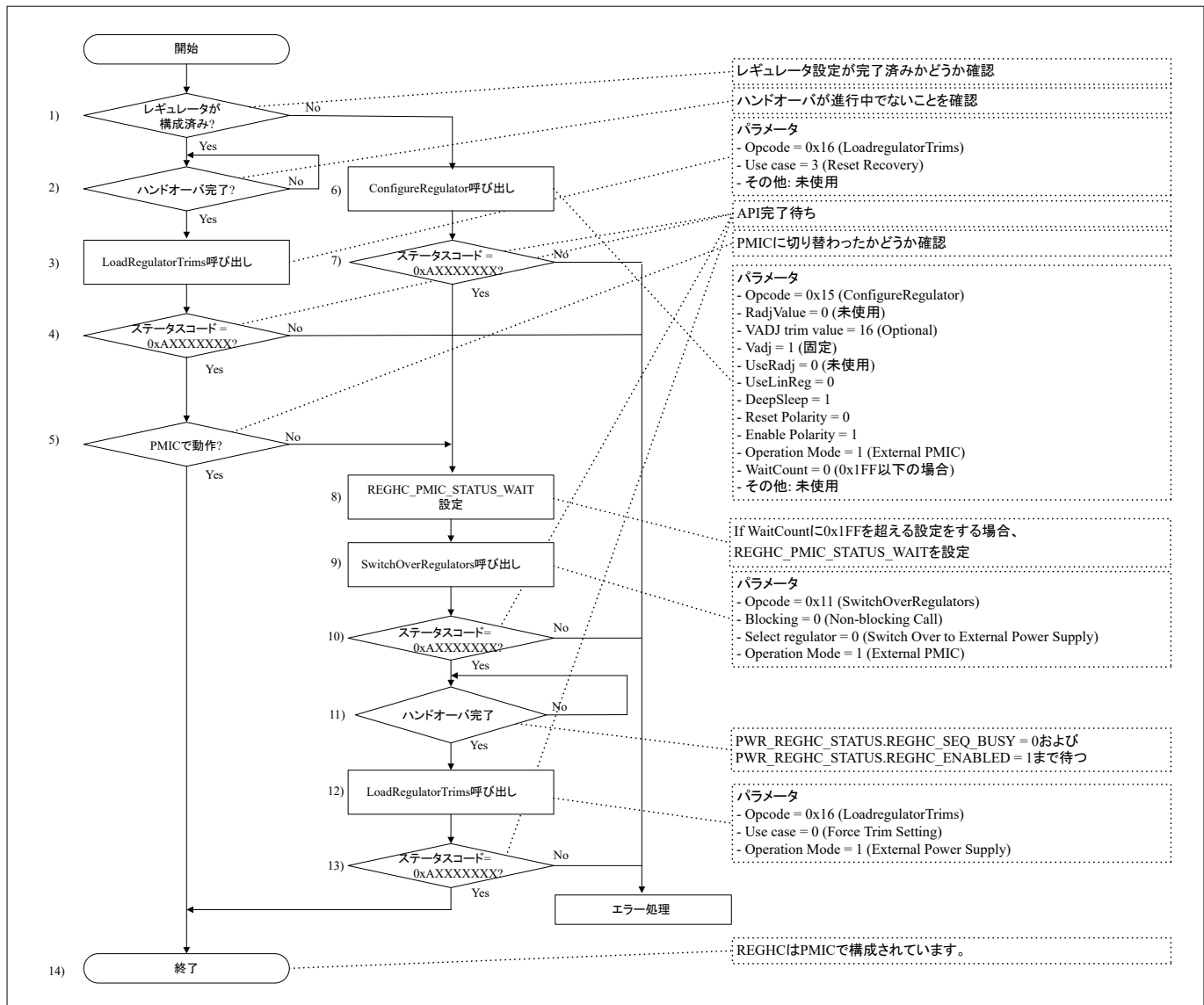


図 29 内部レギュレータから PMIC へのハンドオーバーフロー (ケース 4)

- レギュレータ設定が完了済みかどうか確認してください。既に REGHC または PMIC コントローラがセットアップされている場合、再設定なしで有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(6)へ進んでください。
- ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
- Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
- LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしたがって適切なエラー処理をしてください。
- MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(14)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(8)へ進んでください。
- 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)

## 4 PMIC (スイッチングレギュレータ)

- RadjValue<sup>7)</sup> = 0 (未使用)
  - VADJ trim value = 16
  - Vadj = 1 (固定)
  - UseRadj<sup>7)</sup> = 0 (未使用)
  - UseLinReg = 0 (PMIC イネーブル後、内部 Active レギュレータは無効です)
  - DeepSleep<sup>7)</sup> = 1 (PMIC は DeepSleep 電源モード中に有効です)
  - Reset Polarity = 0 (PMIC PG 信号は、"0" でパワーバッド状態を示します)
  - Enable Polarity = 1 (PMIC は"1"で有効です)
  - Operating Mode<sup>7)</sup> = 1 (外部 PMIC)
  - WaitCount = 0 (0x1FF 以下)
7. ConfigureRegulator API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
  8. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。  
以下の手順に従って、PMIC で REGHC または PMIC コントローラを有効にします。
  9. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
    - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
    - Blocking = 0 (Non-blocking call)
    - Select regulator = 0 (外部電源に切り替えます)
    - Operating Mode<sup>7)</sup> = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。

**注:** ConfigureRegulator API は UseLinReg = "0" および DeepSleep = "1" で設定されますが、Non-blocking call で SwitchOverRegulators を実行します。そのため、SwitchOverRegulators API は PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定しません。
  10. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
  11. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0 および PWR\_REGHC\_STATUS.REGHC\_ENABLED = 1 になるまで待ってください。
  12. Force trim setting で LoadRegulatorTrims API を呼び出してください。
    - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
    - Use case = 0 (Force trim setting)
    - Operating Mode = 1 (外部電源)

これは、内部レギュレータの出力状態を外部状態に変更します。
  13. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
  14. デバイスは PMIC で動作します。Active レギュレータはオフ、DeepSleep レギュレータは外部状態で動作しています。

### PMIC から内部レギュレータへのハンドオーバー

図 30 に、PMIC から内部レギュレータへのハンドオーバーを示します。

<sup>7)</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

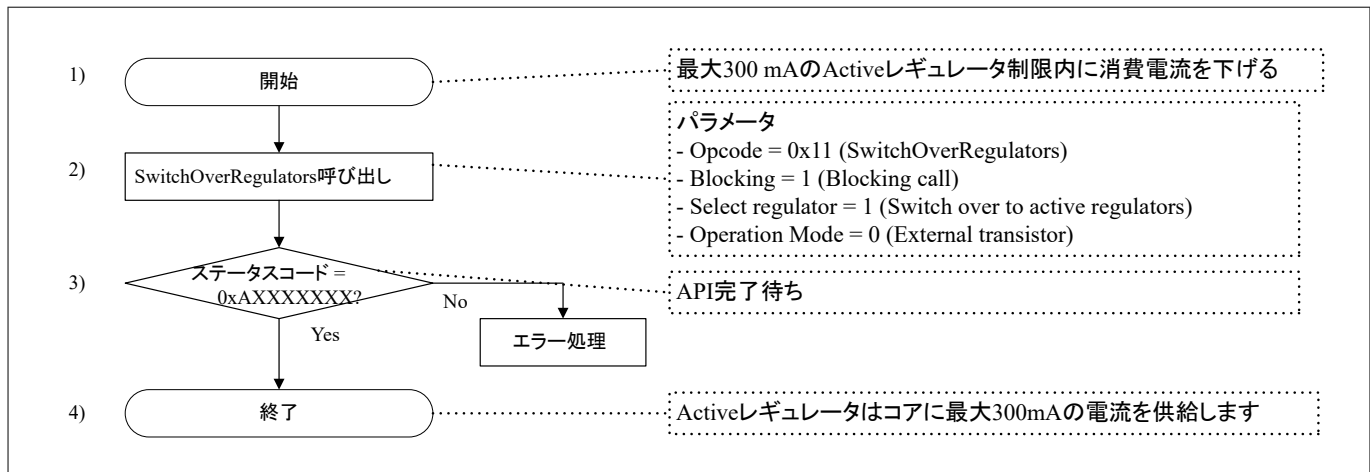


図 30 PMIC から内部レギュレータへのハンドオーバーフロー (ケース 4)

PMIC から内部レギュレータにハンドオーバーする場合

1. 最大 300 mA の Active レギュレータ制限内まで消費電流を下げてください。(クロック、コア、ペリフェラルなどの適切なアプリケーション構成によって)
2. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode<sup>8)</sup> = 1 (外部 PMIC)
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. デバイスは、内部レギュレータで動作しています。

### 4.4 ロードスイッチを使用した PMIC

PMIC 出力と MCU  $V_{CCD}$  間のデカップリングが必要な場合に、コア電源用のロードスイッチの配置について検討します。条件については、[PMIC 仕様要件](#)を参照してください。

一般的に、ロードスイッチを使用した構成は推奨しません。その理由は、すべての動作条件下でのロードスイッチのドレイン-ソース間抵抗による低電圧降下を確実にするための回路の複雑さ、および PMIC 無効時の引き込み電流の懸念のためです。

ここでは、ロードスイッチを使用した PMIC の内部レギュレータ設定に応じた使用例により PMIC 接続、ハンドオーバータイミング、およびハンドオーバーソフトウェアフロー例について説明します。

CYT3B/4B シリーズのみこの設定をサポートします。

- **ケース 1:** ロードスイッチゲートは、PMIC のパワーグッド信号によって制御します。Active 電源モードで OCD は使用され、DeepSleep 電源モードで PMIC は無効です
- **ケース 2:** ロードスイッチゲートは、REGHC(EXT\_PS\_CTL1)ピンによって制御します。Active 電源モードで OCD は使用されず、DeepSleep 電源モードで PMIC は有効です (Blocking Call で SwitchOverRegulators API を使用)
- **ケース 3:** ロードスイッチゲートは、REGHC(EXT\_PS\_CTL1)ピンによって制御します。Active 電源モードで OCD は使用されず、DeepSleep 電源モードで PMIC は有効です (Non-blocking Call で SwitchOverRegulators API を使用)

<sup>8</sup> CYT3D/4D シリーズでは無効です。

## 4 PMIC (スイッチングレギュレータ)

### 4.4.1 ケース 1

ここでは、ロードスイッチ (LSW) を使用したソリューションについて説明します。PMIC を VCCD から絶縁するためにロードスイッチを使用します。このソリューションでは、PMIC がパワーグッド状態を示す場合、PMIC 出力は十分安定している必要があります。

ここでは、Active 電源モードで OCD を使用し、DeepSleep 電源モードで PMIC 無効にするケースについて説明します。ロードスイッチゲートは、PMIC のパワーグッド信号によって制御します。

#### 4.4.1.1 ハードウェア構成

図 31 に、ロードスイッチ 1 構成での PMIC の回路図を示します。この設定では、ロードスイッチのゲートは PG 信号に接続されます。したがって、ロードスイッチは PG 信号によって直接制御されます。

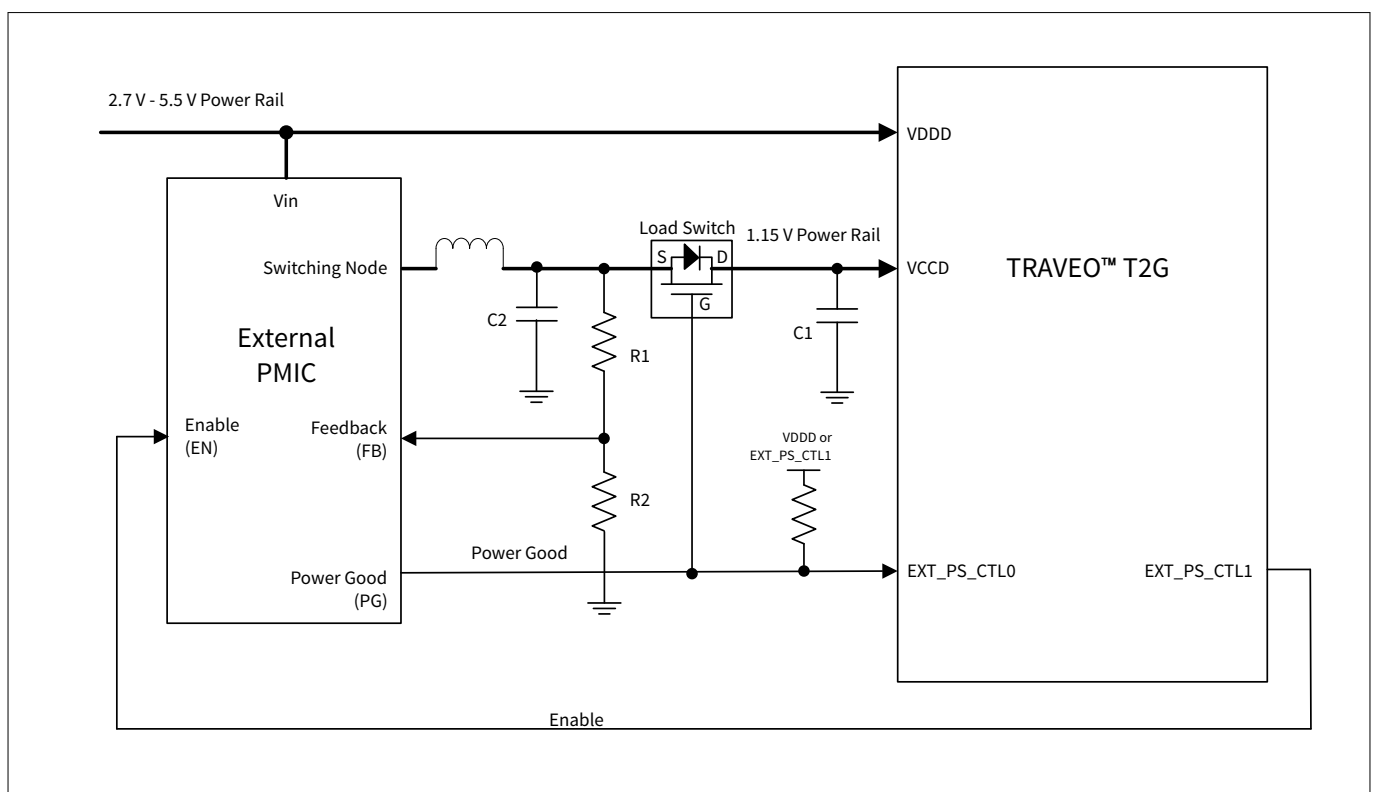


図 31 ロードスイッチ 1 構成での PMIC の回路図

- PMIC EN の極性は、HIGH で有効です。PMIC パワーグッド (PG) の極性は、HIGH で PG を示します。PMIC リセット (RESET\_PMIC) は、PMIC の PG が LOW になると生成されます。MCU は HV リセットを生成します。
- PMIC のフィードバック信号は、ロードスイッチの PMIC 側に接続されます。
- ロードスイッチは PMIC 出力と MCU の V<sub>CCD</sub> 間に接続します。
- ロードスイッチのゲートと PMIC の PG は、MCU の EXT\_PS\_CTL0 に接続されます。ただし、PG は電源 (V<sub>DD</sub>) にプルアップします。

PMIC の Vin 電圧が上昇すると、Vin の過渡電圧と PMIC の PG のハイインピーダンス (Hi-z) 状態によって、PMIC が PG をプルダウンする前に PG 電圧が上昇する場合があります。結果として、意図しないタイミングでロードスイッチがオンする場合があります。意図しないオンを防ぐ方法の 1 つとして、EXT\_PS\_CTL0 (PMIC パワーグッド) のプルアップを VDD ではなく、EXT\_PS\_CTL1 (PMIC EN) に接続することです。

- PMIC の EN ピンは MCU の EXT\_PS\_CTL1 ピンに接続されます。XRES および Hibernate 中に PMIC を無効にするためには、PCB または PMIC にプルダウン抵抗が必要です。

## 4 PMIC (スイッチングレギュレータ)

- EN ピンに内部プルダウン抵抗がない場合、POR 中に PMIC を無効にするために外部プルダウン抵抗を配置する必要があります。
- $V_{CCD}$  コンデンサ (C1) は MCU に依存します。(静電容量についてはデバイスデータシート [1]を参照してください)
- 出力電圧抵抗 (R1, R2) および出力コンデンサ (C2) は、選択した PMIC によって異なります。

以降のソフトウェアフローでは、PMIC イネーブル(EN)端子極性のレジスタ設定は”1”、パワーグッド状態極性は”1”を想定しています。

ConfigureRegulator API の Enable Polarity は PMIC へのイネーブル信号極性を指定し、ConfigureRegulator API の Reset Polarity は PMIC からのパワーグッド状態の極性を指定します。Reset polarity はパワーグッドのディアサート状態を指定することに注意してください。これらの設定は PMIC 仕様によって異なります。

このケースでは、内部レギュレータ設定は OCD を使用し、PMIC は DeepSleep 電源モードで無効です。

### 4.4.1.2 ハンドオーバータイミングチャート

図 32 に、ロードスイッチ 1 構成でのハンドオーバーシーケンス例を示します。この例には、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバー、DeepSleep 電源モードへの移行、DeepSleep 電源モードからのウェイクアップ、および PMIC から内部レギュレータへのハンドオーバーが含まれます。ここでは、PMIC イネーブル(EN)端子極性のレジスタ設定は”1”、パワーグッド状態極性は”1”を想定しています。

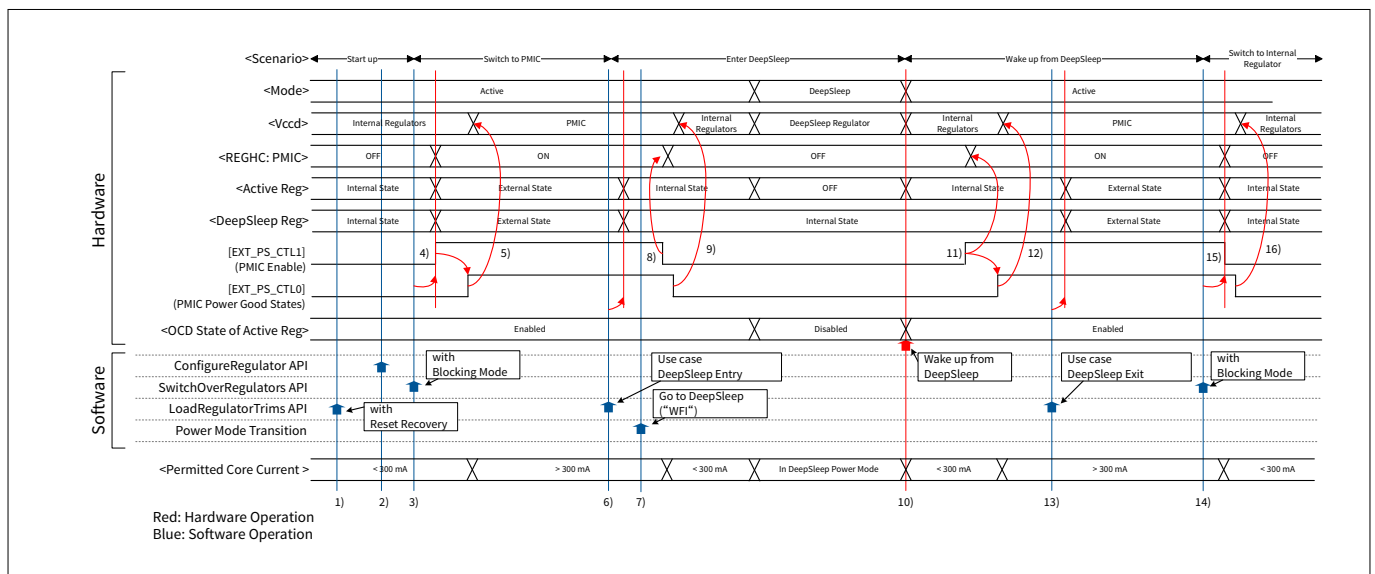


図 32 ロードスイッチ 1 構成での PMIC ハンドオーバーシーケンス例

- PWR\_REGHC\_CTL.REGHC\_CONFIGURED が”1”に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
- 起動後、PMIC で REGHC を設定するために ConfigureRegulator API を呼び出してください。
- 内部レギュレータから PMIC へハンドオーバーするため SwitchOverRegulators API を呼び出してください。Active レギュレータは OCD 機能により有効になり外部状態に変更されます。DeepSleep レギュレータは外部状態に変更されます。SwitchOverRegulators API によって、EXT\_PS\_CTL1 は PMIC 有効信号を出力し、PMIC は有効になります。
- PMIC の準備ができると、EXT\_PS\_CTL0 はパワーグッド状態になります。
- EXT\_PS\_CTL0 が、パワーグッド状態になるとロードスイッチがオンします。PMIC は MCU に電源供給します。
- DeepSleep Entry ユースケースで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは内部状態に変更されます。



## 4 PMIC (スイッチングレギュレータ)

7. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。その後、WFI 命令を使用して DeepSleep 電源モードへの移行を実行できます。
8. DeepSleep 電源モードへの移行により DeepSleep レギュレータへの移行がトリガされます。EXT\_PS\_CTL1 は、PMIC 無効信号を出力し、EXT\_PS\_CTL0 は、パワーグッド状態を示しません。
9. EXT\_PS\_CTL0 が、パワーグッド状態を示さない場合、ロードスイッチはオフされます。内部レギュレータは MCU に電源供給します。
10. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生し、その後、Active レギュレータは内部状態で開始されます。
11. EXT\_PS\_CTL1 は PMIC 有効信号を出力し、PMIC は有効になります。PMIC の準備ができると、EXT\_PS\_CTL0 はパワーグッド状態になります。
12. EXT\_PS\_CTL0 が、パワーグッド状態になるとロードスイッチがオンします。PMIC は MCU に電源供給します。
13. 内部レギュレータは外部状態に変更するために DeepSleep Exit ユースケースで LoadRegulatorTrims API を呼び出してください。
14. PMIC から内部レギュレータにハンドオーバーするため SwitchOverRegulators API を呼び出してください。内部レギュレータは内部状態になります。
15. EXT\_PS\_CTL1 は無効信号を出力し、EXT\_PS\_CTL0 は、パワーグッド状態を示しません。
16. EXT\_PS\_CTL0 が、パワーグッド状態を示さない場合、ロードスイッチはオフされます。内部レギュレータは MCU に電源供給します。

### 4.4.1.3 ソフトウェアフロー

#### 内部レギュレータから PMIC へのハンドオーバー

図 33 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。

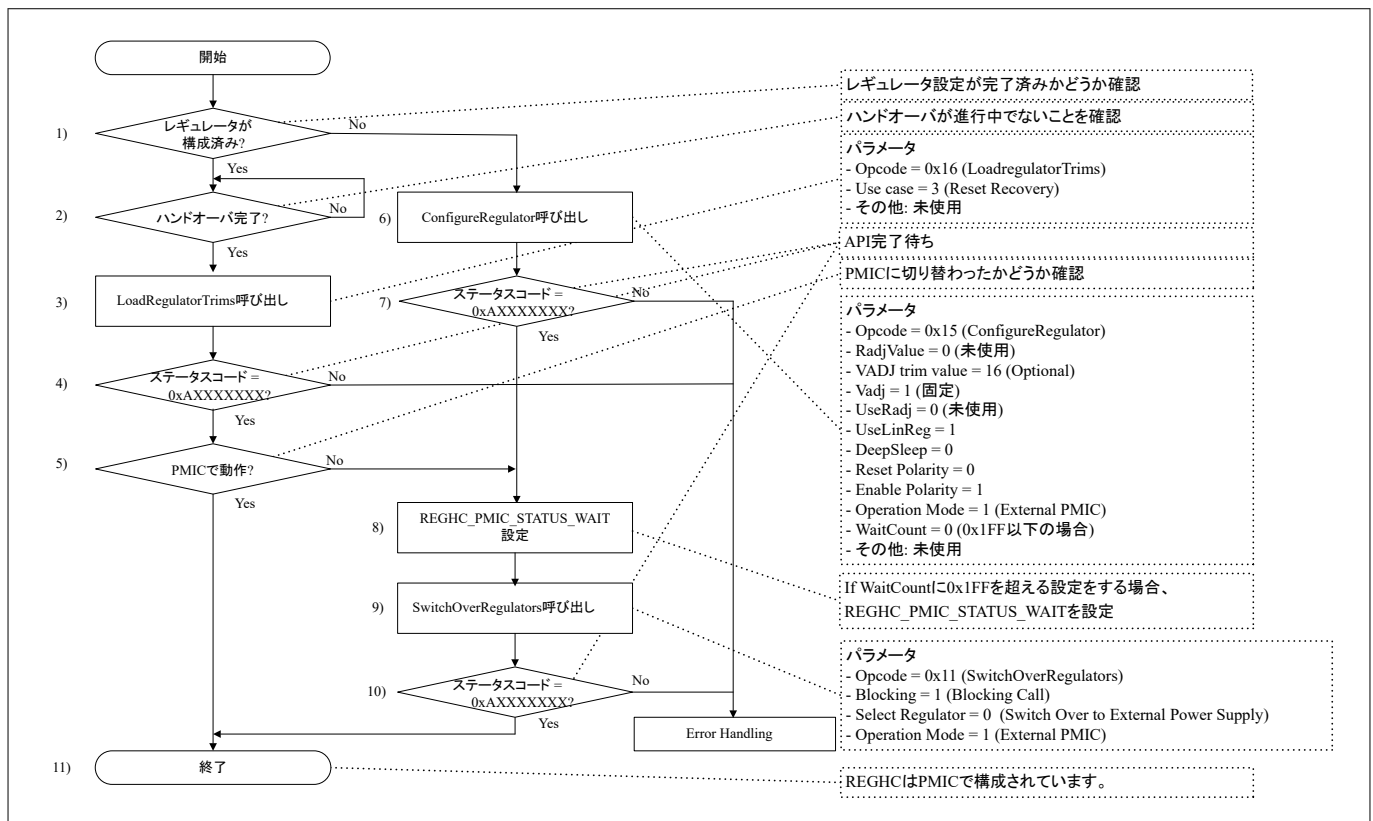



図 33 内部レギュレータからロードスイッチ 1 構成での PMIC へのハンドオーバーフロー

## 4 PMIC (スイッチングレギュレータ)

1. レギュレータ設定が完了済みかどうか確認してください。既に REGHC がセットアップされている場合、再構成せずに有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(6)へ進んでください。
2. ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
3. Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
4. LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしたがって適切なエラー処理をしてください。
5. MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(11)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(8)へ進んでください。
6. 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
  - RadjValue = 0 (未使用)
  - VADJ trim value = 16
  - Vadj = 1 (固定)
  - UseRadj = 0 (未使用)
  - UseLinReg = 1 (PMIC イネーブル後も内部 Active レギュレータは有効です)
  - DeepSleep = 0 (PMIC は、DeepSleep 電源モード中は無効です)
  - Reset Polarity = 0 (PMIC PG 信号は、“0” でパワーバッド状態を示します)
  - Enable Polarity = 1 (PMIC は“1”で有効です)
  - Operating Mode = 1 (外部 PMIC)
  - WaitCount = 0 (0x1FF 以下)
7. ConfigureRegulator API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
8. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。  
以下の手順に従って、PMIC で REGHC を有効にします。
9. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 0 (外部電源に切り替えます)
  - Operating Mode = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。
10. SwitchOverRegulators API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
11. デバイスは PMIC で動作します。Active および DeepSleep レギュレータは外部状態で動作します。

### DeepSleep への遷移と復帰

 34 に、ロードスイッチ 1 構成での PMIC 時の DeepSleep 電源モードへの遷移を示します。



#### 4 PMIC (スイッチングレギュレータ)

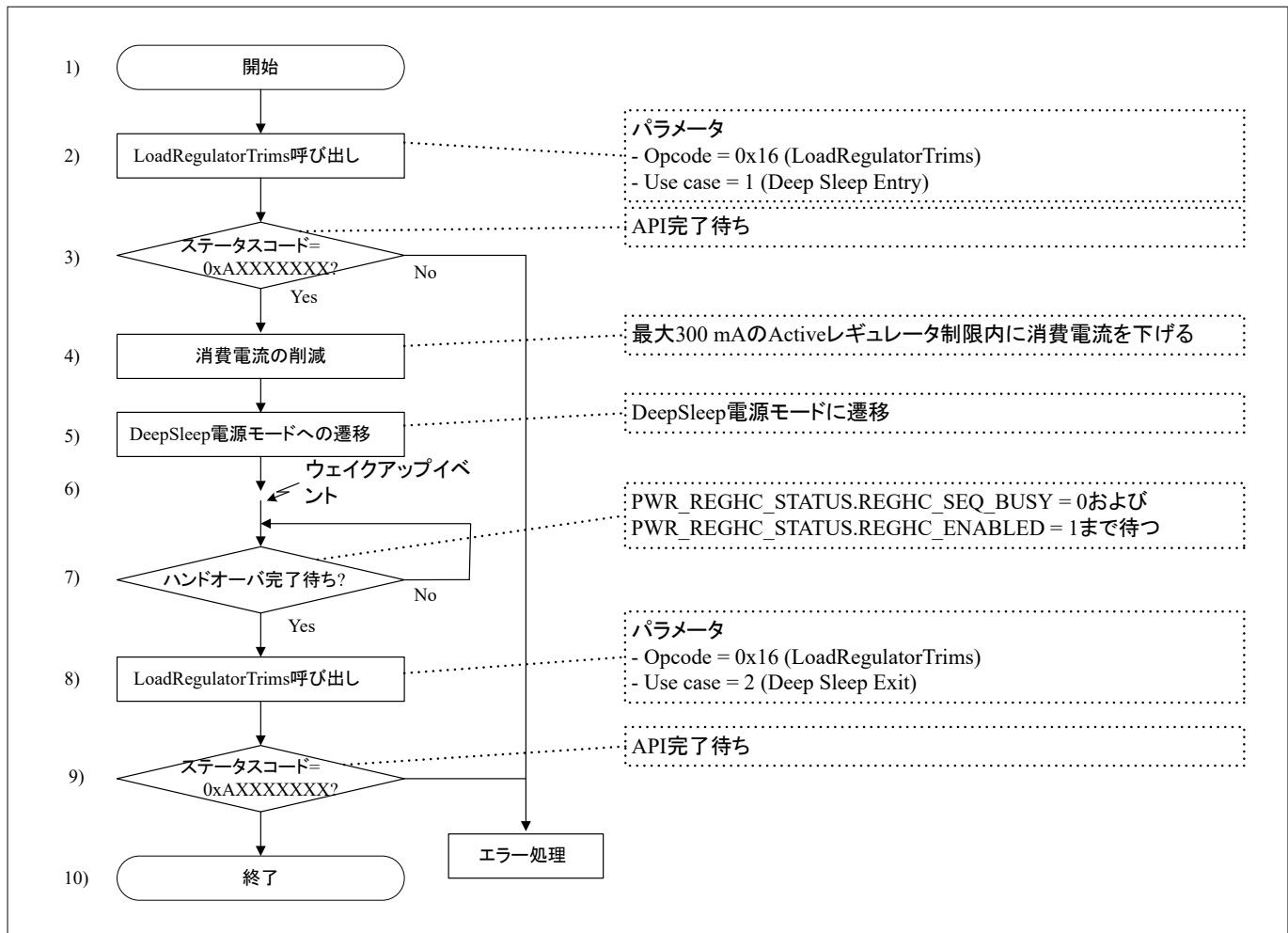


図 34 ロードスイッチ 1 構成での PMIC 時の DeepSleep への遷移と復帰フロー

1. MCU は PMIC から電源供給されます。
2. DeepSleep 電源モード移行前に以下のパラメータで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 1 (DeepSleep Entry)
3. LoadRegulatorTrims API の完了を待ってください。ステータスコードが”0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
4. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
5. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。内部レギュレータへのハンドオーバーを実行し、PMIC を無効にします。次に Active レギュレータは無効、DeepSleep レギュレータは DeepSleep 中に電源供給します。  
MCU は DeepSleep レギュレータから電源供給されます。
6. ウェイクアップイベントのため、MCU は Active 電源モードに遷移します。次に、ハードウェアが PMIC を有効にします。
7. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = “1”まで待ってください。
8. 以下のパラメータで LoadRegulatorTrims API を呼び出してください。Active および DeepSleep レギュレータは外部状態に設定されます。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 2 (DeepSleep Exit)

## 4 PMIC (スイッチングレギュレータ)

9. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
10. MCU は PMIC から電源供給されます。

### PMIC から内部レギュレータへのハンドオーバー

図 35 に、ロードスイッチ 1 構成での PMIC から内部レギュレータへのハンドオーバーを示します。

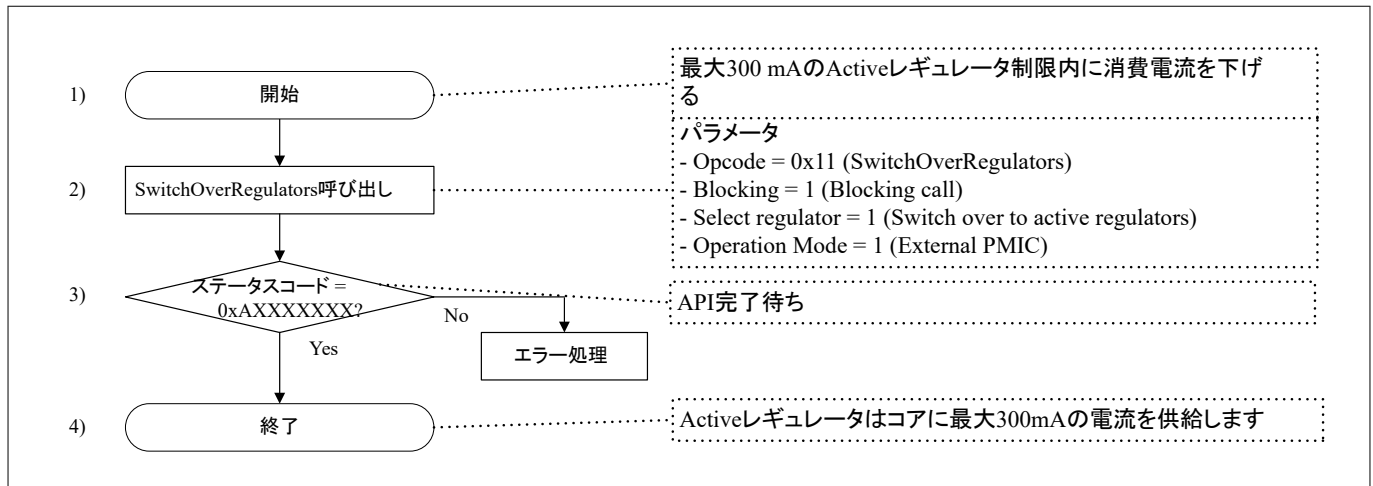


図 35 ロードスイッチ 1 構成での PMIC から内部レギュレータへのハンドオーバーフロー

PMIC から内部レギュレータにハンドオーバーする場合

1. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
2. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode = 1 (外部 PMIC)
3. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
4. デバイスは、内部レギュレータで動作しています。

### 4.4.2 ケース 2

ここでは、Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Blocking Call で SwitchOverRegulators API を使用する) ケースについて説明します。REGHC 信号によってゲート制御するケースについて説明します。

PMIC トポロジの選択に応じて、PMIC を VCCD から絶縁するために、ロードスイッチを使用します。詳細は、[PMIC 仕様要件](#)を参照してください。PMIC 設定での REGHC は、LV リセット (ソフトウェア, FAULT, WDT, MCWDT, CSV) ではなく、HV リセット (XRES, POR, BOD, OVD, OCD, HIBERNATE ウェイクアップ) によって初期化されます。リセットソースについては、アーキテクチャ TRM [2]の「Reset System」を参照してください。

#### 4.4.2.1 ハードウェア構成

図 36 に、ロードスイッチ 2 構成での PMIC の回路図を示します。

#### 4 PMIC (スイッチングレギュレータ)

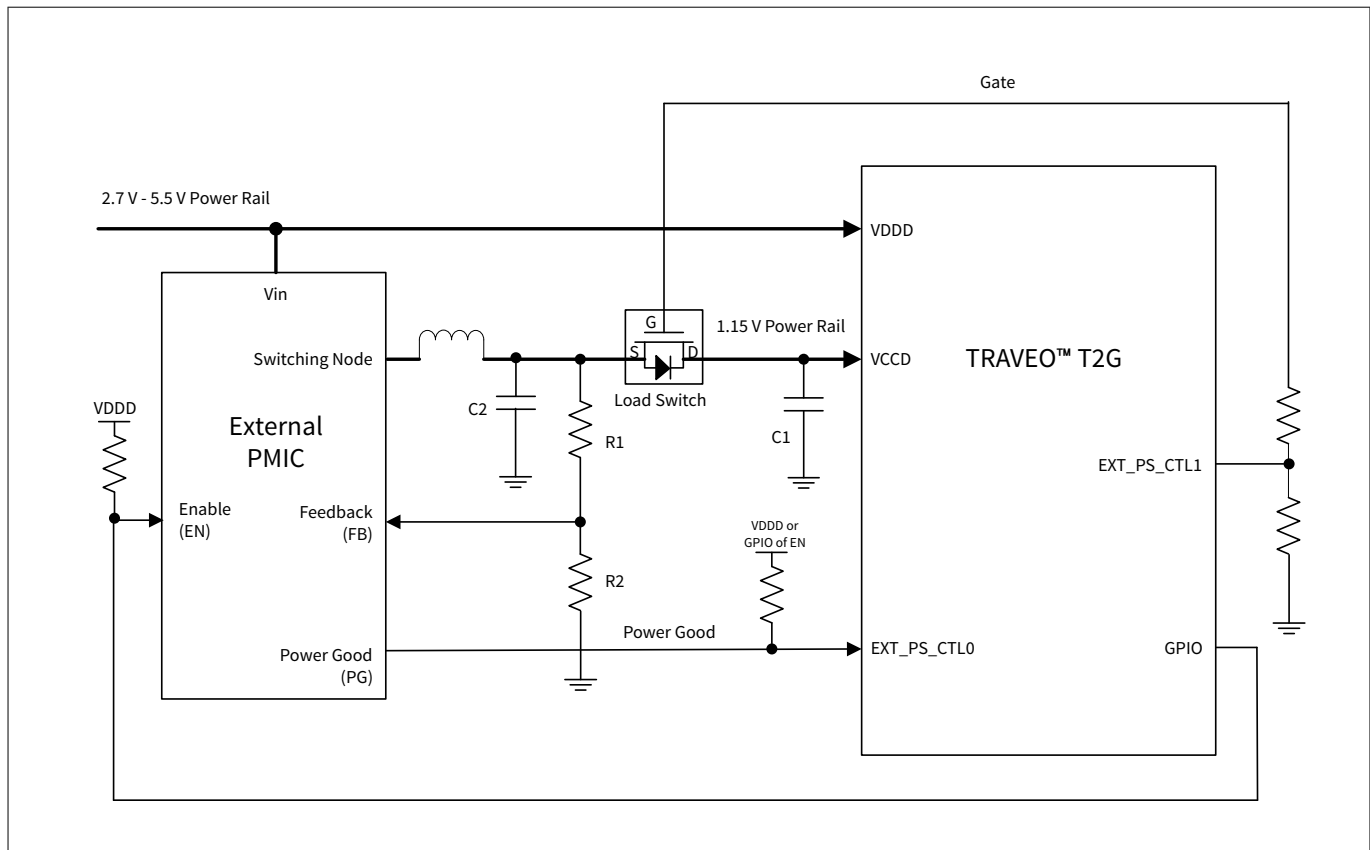


図 36 ロードスイッチ 2 構成での PMIC の回路図

- PMIC EN の極性は、HIGH で有効です。PMIC パワーグッド (PG) の極性は、HIGH で PG を示します。RESET\_PMIC は、PMIC の PG が LOW になると生成されます。これは HV リセットです。HV リセットによって、PMIC 設定の REGHC を含むマイコンが初期化されます。
- PMIC のフィードバック信号は、ロードスイッチの PMIC 側に接続されます。
- ロードスイッチは PMIC 出力と MCU の V<sub>CCD</sub> 間に接続します。
- ロードスイッチのゲートは、MCU の EXT\_PS\_CTL1 に接続されます。
- PMIC の EN 端子は、MCU の GPIO に接続されます。DeepSleep 電源モード中に正しく動作させるために DeepSleep 中も電力を維持する電源ドメインの GPIO を使用します。
- LV リセット中に PMIC を有効にするため、GPIO にプルアップ抵抗が接続されます。
- ロードスイッチのゲート容量の電流制限を 1 mA 以下にするため制限抵抗が、EXT\_PS\_CTL1 とロードスイッチ間に配置されます。
- VCCD コンデンサ (C1) は MCU に依存します。(静電容量についてはデバイスデータシート [1]を参照してください)
- 出力電圧抵抗 (R1, R2) および出力コンデンサ (C2) は、選択した PMIC によって異なります。

**注:** この例では、DeepSleep 電源モード中に消費電流が増加します。これを許容できる場合、このユースケースを使用できます。

以降のソフトウェアフローでは、PMIC イネーブル(EN)端子極性のレジスタ設定は”1”、パワーグッド状態極性は”1”を想定します。

ConfigureRegulator API の”Enable polarity”は PMIC へのイネーブル信号極性を指定し、ConfigureRegulator API の Reset Polarity は PMIC からのパワーグッド状態の極性を指定します。Reset polarity はパワーグッドのディアサート状態を指定することに注意してください。これらの設定は PMIC 仕様によって異なります。

## 4 PMIC (スイッチングレギュレータ)

このケースでは、内部レギュレータ構成は OCD を使用せず、PMIC は DeepSleep 電源モードで有効です。

### 4.4.2.2 PMIC のハンドオーバータイミングチャート

図 37 に、ロードスイッチ 2 構成でのハンドオーバーシーケンス例を示します。この例は、レギュレータ設定、内部レギュレータから PMIC へのハンドオーバー、DeepSleep 電源モードへの移行、および DeepSleep 電源モードからのウェイクアップが含まれます。この設定では、API を使用して PMIC から内部レギュレータに戻すことはできません。内部レギュレータに戻す場合、チップ全体をリセット (HV リセット) が必要です。システムが、SwitchOverRegulators API によって PMIC から内部レギュレータにハンドオーバーを実行する必要がある場合、[ケース 3](#) を参照してください。

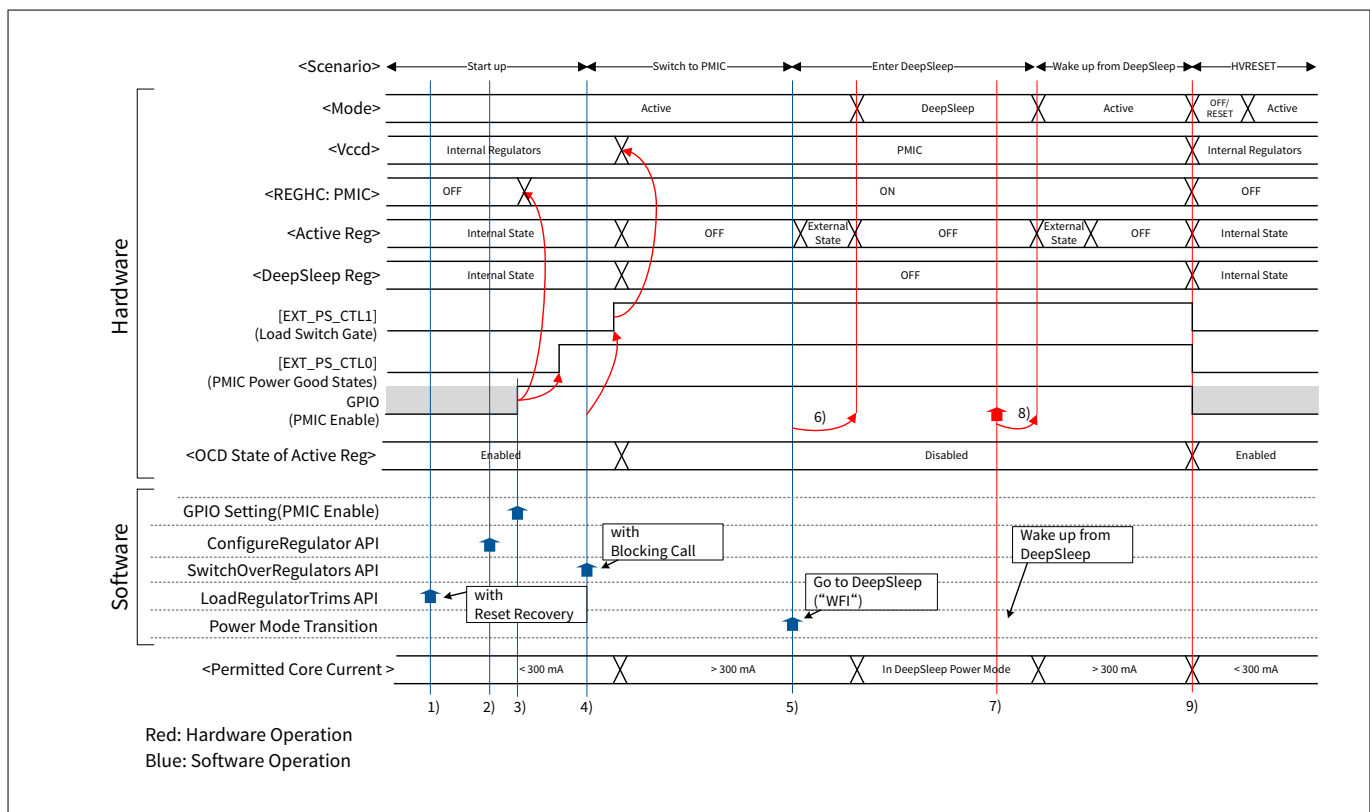


図 37 ロードスイッチ 2 構成での PMIC ハンドオーバーシーケンス例 (ケース 2)

GPIO (PMIC イネーブル端子) の初期状態はピンの終端に依存します。

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が "1" に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、PMIC で REGHC を設定するために ConfigureRegulator API を呼び出してください。
3. ユーザソフトウェアは GPIO を "1" に設定して PMIC を有効にします。PMIC が有効になると EXT\_PS\_CTL0 はパワーグッド状態になります。ここでは、PMIC は有効ですがハンドオーバーは行われません。
4. 内部レギュレータから PMIC へハンドオーバーするため SwitchOverRegulators API を呼び出してください。  
SwitchOverRegulators API によって EXT\_PS\_CTL1 がロードスイッチをオンし、MCU は PMIC から電源供給されます。Active と DeepSleep レギュレータは無効です。  
システムによって SwitchOverRegulators API を使用して PMIC から内部レギュレータへハンドオーバーする必要がある場合、PMIC へのハンドオーバー時に Non-blocking call で SwitchOverRegulators API を呼び出してください。
5. WFI 命令を使用して DeepSleep 電源モードへの遷移を実行できます。この場合、システムは消費電流の削減なしに DeepSleep 電源モードに移行できます。

## 4 PMIC (スイッチングレギュレータ)

DeepSleep 電源モードへの移行時、PMIC は引き続き有効、そして Active と DeepSleep レギュレータは無効です。

6. MCU は DeepSleep 電源モードへ移行します。
7. DeepSleep から Active への電源モード遷移は、ウェイクアップイベント後に発生します。
8. Active 電源モードへの移行が完了時、PMIC は引き続き電源を供給しています。Active および DeepSleep レギュレータは無効です。
9. HV リセットが発生すると、電源システムは初期化されます。そのため、リセット解除後、MCU は Active レギュレータで起動します。

### 4.4.2.3 ソフトウェアフロー

#### 内部レギュレータから PMIC へのハンドオーバー

図 38 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。

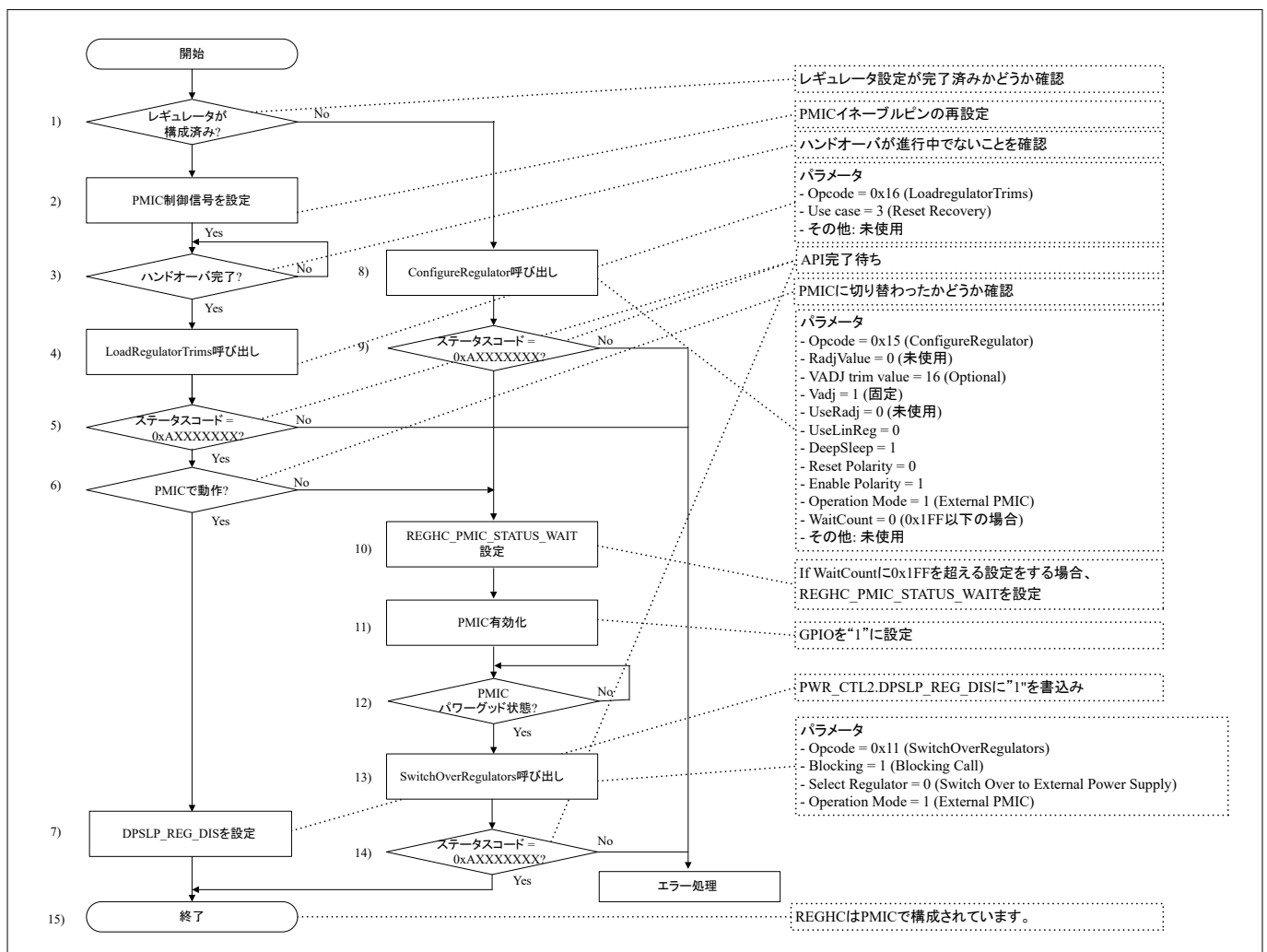


図 38 内部レギュレータからロードスイッチ 2 構成での PMIC へのハンドオーバーフロー (ケース 2)

1. レギュレータ設定が完了済みかどうか確認してください。既に REGHC がセットアップされている場合、再構成せずに有効にできます。  
PWR\_REGHC\_CTL.REGHC\_CONFIGURED が "0" の場合、設定は完了していません。(8)へ進んでください。

## 4 PMIC (スイッチングレギュレータ)

2. PMIC イネーブル制御のため GPIO を設定してください。この場合、LV リセット前の PMIC イネーブル状態が再設定されます。以下に GPIO 再設定の例を示します。
  - a. ~ (PWR\_REGHC\_CTL2.REGHC\_EN ^ PWR\_REGHC\_CTL.REGHC\_PMIC\_CTL\_POLARITY) の値を GPIO データレジスタに書き込んでください。
  - b. GPIO を出力に設定してください。
3. ハンドオーバー完了を待ってください。  
PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = "0" および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
4. Reset Recovery ユースケースで LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 3 (Reset Recovery)
5. LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしがって適切なエラー処理をしてください。
6. MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が"1"の場合、PMIC が電源供給します。(7)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が"0"の場合、内部レギュレータが電源供給します。(12)へ進んでください。
7. PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定してください。  
ケース 2 では、PMIC は DeepSleep 電源モードで有効です。したがって、PMIC へのハンドオーバーが完了している場合、ユーザソフトウェアは PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定する必要があります。そうでない場合、Blocking call での SwitchOverRegulators API により、PWR\_CTL2.DPSLP\_REG\_DIS が設定されます。
8. 以下のパラメータで ConfigureRegulator API を呼び出してください。
  - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
  - RadjValue = 0 (未使用)
  - VADJ trim value = 16
  - Vadj = 1 (固定)
  - UseRadj = 0 (未使用)
  - UseLinReg = 0 (PMIC イネーブル後、内部 Active レギュレータは無効です)
  - DeepSleep = 1 (PMIC は DeepSleep 電源モード中に有効です)
  - Reset Polarity = 0 (PMIC PG 信号は、"0" でパワーバッド状態を示します)
  - Enable Polarity = 1 (PMIC は"1"で有効です)
  - Operating Mode = 1 (外部 PMIC)
  - WaitCount = 0 (0x1FF 以下)

**注:** この設定では、電流制限抵抗が EXT\_PS\_CTL1 (PMIC イネーブル信号出力) とロードスイッチ間に配置されます。この抵抗とロードスイッチの容量によって、ロードスイッチがすぐにオンされない場合があります。したがって、ロードスイッチが完全にオンされるまでハンドオーバー完了を待つ必要があります。このロードスイッチがオンされるまでの待機時間はハードウェアタイマを使用して待つことができます。この待機時間は、ConfigureRegulator API の WaitCount を使用して設定できます。必要な待機時間の設定については、[ロードスイッチ](#)を参照してください。
9. ConfigureRegulator API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
10. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。



## 4 PMIC (スイッチングレギュレータ)

11. PMIC を有効にするために GPIO を “1” に設定してください。その後、PMIC は有効になります。
12. PWR\_REGHC\_STATUS.REGHC\_PMIC\_STATUS\_OK が “1” になるまで待ってください。これは、PMIC がパワーグッド状態になったことを示します。PMIC が十分に安定する前に PMIC がパワーグッド状態を示す場合、ソフトウェアは PMIC 出力を待つ必要があります。  
以下の手順に従って、PMIC で REGHC を有効にします。
13. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 1 (Blocking call)
  - Select regulator = 0 (外部電源に切り替えます)
  - Operating Mode = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。

**注:** システムが SwitchOverRegulators API を使用して PMIC から内部レギュレータに戻す必要がある場合、PMIC へのハンドオーバー実行時に、Non-blocking call で SwitchOverRegulators API を呼び出す必要があります。

14. SwitchOverRegulators API の完了を待ってください。ステータスコードが “0xAXXXXXXX” 以外の場合、システムに応じて適切なエラー処理をしてください。
15. デバイスは PMIC で動作します。Active および DeepSleep レギュレータは無効です。

### DeepSleep への遷移と復帰

図 39 に、ロードスイッチ 2 構成での PMIC 時の DeepSleep 電源モードへの遷移を示します。このケースでは、追加操作なしに、DeepSleep 電源モードに遷移できます。

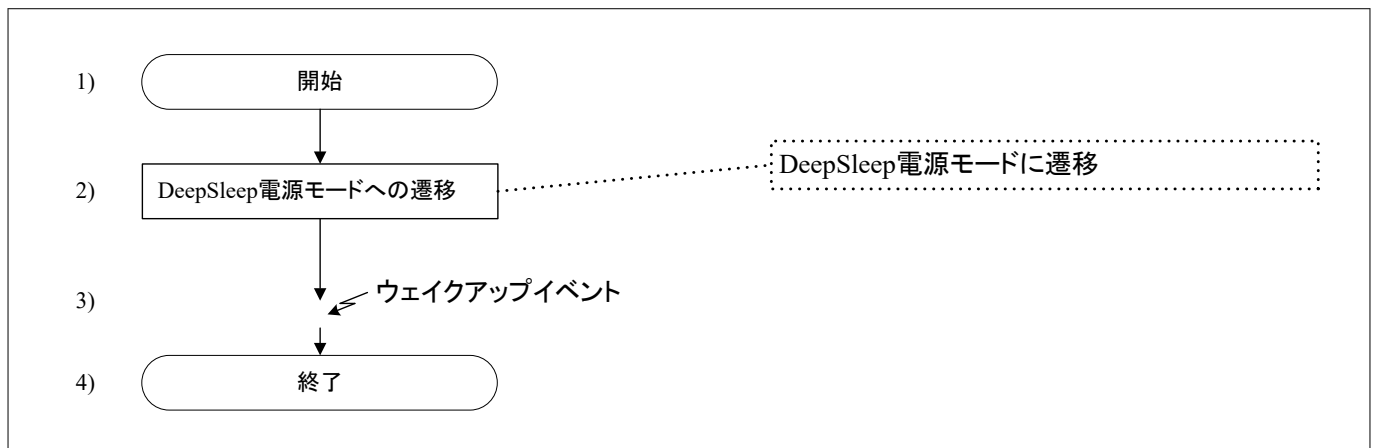


図 39 ロードスイッチ 2 構成での PMIC 時の DeepSleep への遷移と復帰フロー (ケース 2)

1. MCU は PMIC から電源供給されます。
2. WFI 命令を使用して DeepSleep 電源モードへの遷移を開始してください。Active レギュレータは無効、PMIC は引き続き電源供給します。
3. ウェイクアップイベントのため、MCU は Active 電源モードに遷移します。Active レギュレータは外部状態になります。
4. MCU は PMIC から電源供給されます。

### 4.4.3 ケース 3

ここでは、Active 電源モードで OCD を使用しない、DeepSleep 電源モードで PMIC 有効にする (Non-blocking Call での SwitchOverRegulators 使用) ケースについて説明します。REGHC 信号によってゲート制御するケースについて説明します。



## 4 PMIC (スイッチングレギュレータ)

このケースでの回路図は図 36 と同じです。

図 40 に、ロードスイッチ 2 構成でのハンドオーバーシーケンス例を示します。この例は、レギュレータ設定, 内部レギュレータから PMIC へのハンドオーバ, および PMIC から内部レギュレータのハンドオーバが含まれます。

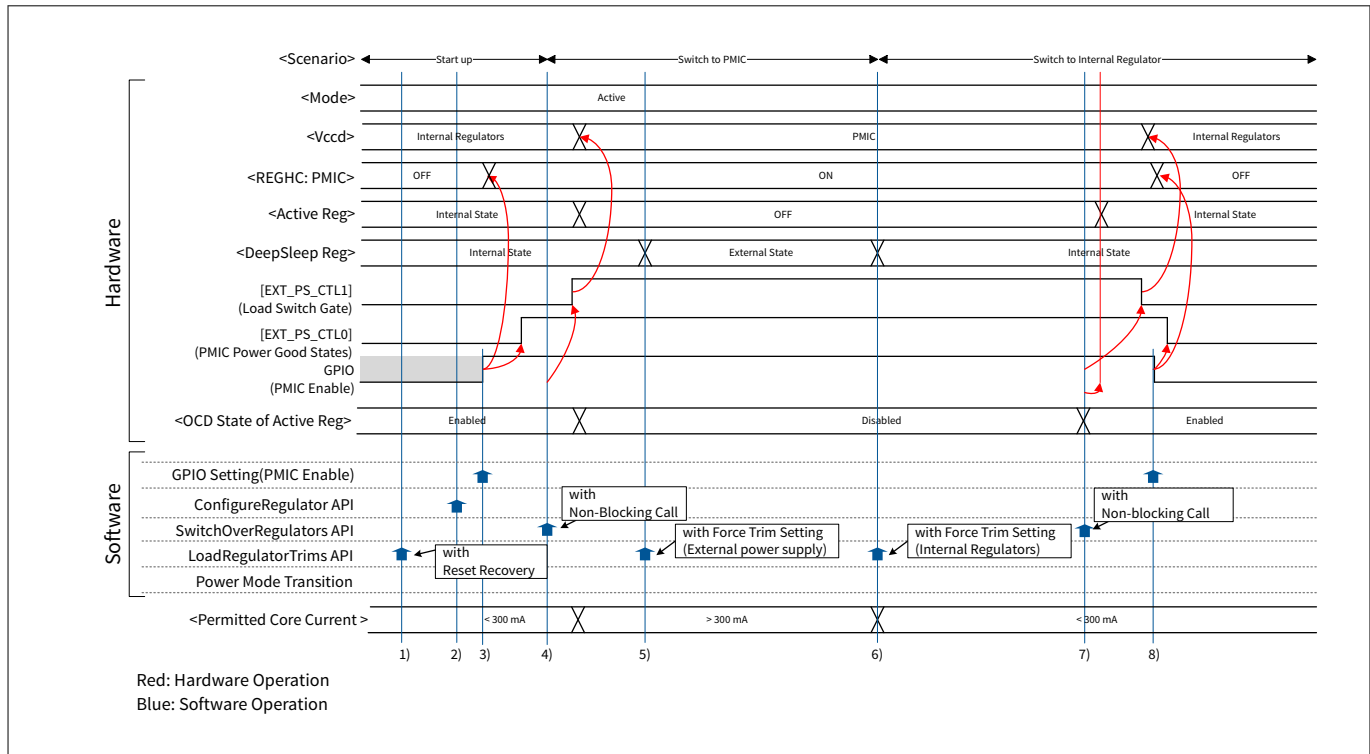


図 40 ロードスイッチ 2 構成での PMIC ハンドオーバーシーケンス例 (ケース 3)

GPIO (PMIC イネーブル端子) の初期状態はピンの終端に依存します。

1. PWR\_REGHC\_CTL.REGHC\_CONFIGURED が"1"に設定されている場合、Reset Recovery ユースケースで LoadRegulatorTrims API 呼び出してください。詳細については、[Reset Recovery](#) を参照してください。
2. 起動後、PMIC で REGHC を設定するために ConfigureRegulator API を呼び出してください。
3. ユーザソフトウェアは GPIO を"1"に設定して PMIC を有効にします。PMIC が有効になると EXT\_PS\_CTL0 はパワーグッド状態になります。ここでは、PMIC は有効ですがハンドオーバは行われません。
4. 内部レギュレータから PMIC へハンドオーバするため SwitchOverRegulators API を呼び出してください。

**注:** Non-blocking call で SwitchOverRegulators API を呼び出した場合、API は PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定しません。したがって、SwitchOverRegulators API を使用して、PMIC から内部レギュレータに戻すことができます。ただし、アプリケーションソフトウェアが PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定した場合は、PMIC から内部レギュレータに戻すことはできません。

PMIC は有効になります。OCD 機能を使用しないため、Active レギュレータは無効です。

5. Force trim setting ユースケース、内部レギュレータ状態を変更するために Operating Mode を外部電源に指定して LoadRegulatorTrims API を呼び出してください。DeepSleep レギュレータは外部状態に変更されます。
6. Force trim setting ユースケース、内部レギュレータ状態を変更するために Operating Mode を内部レギュレータに指定して LoadRegulatorTrims API を呼び出してください。DeepSleep レギュレータは内部状態に変更されます。

## 4 PMIC (スイッチングレギュレータ)

- PMIC から内部レギュレータへハンドオーバーのため Non-blocking call で SwitchOverRegulators API を呼び出してください。その後、EXT\_PS\_CTL1 は、ロードスイッチをオフにします。ユーザソフトウェアはハンドオーバーの完了を待つ必要があります。
- ユーザソフトウェアは、GPIO を“0”に設定して PMIC を無効にします。PMIC が無効になると EXT\_PS\_CTL0 がパワーグッドディassert状態になります。PMIC は無効になります。

### 4.4.3.1 ソフトウェアフロー

#### 内部レギュレータから PMIC へのハンドオーバー

図 41 に、内部レギュレータから PMIC へのハンドオーバーフローを示します。

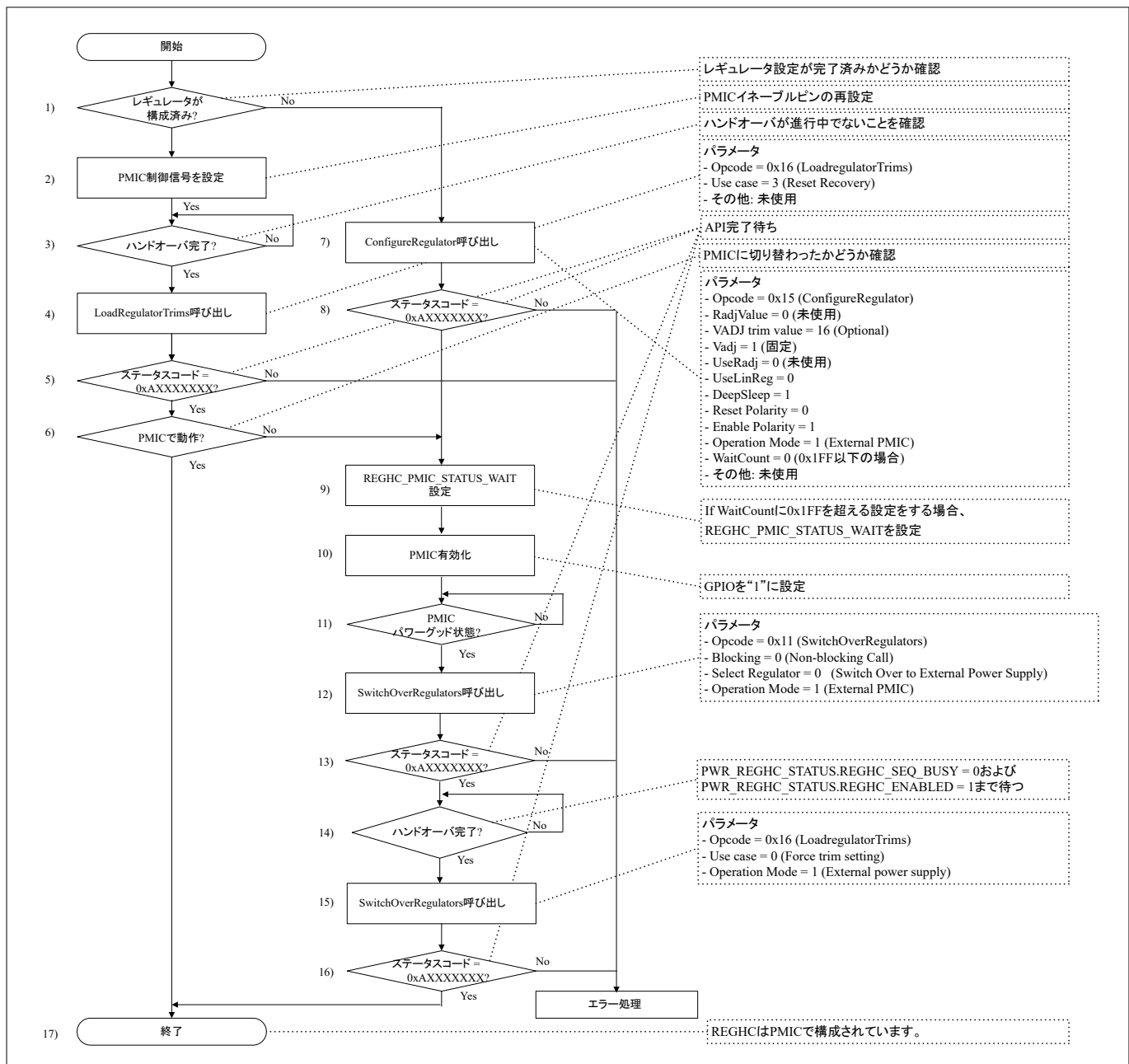


図 41 内部レギュレータからロードスイッチ 2 構成での PMIC へのハンドオーバーフロー (ケース 3)

- レギュレータ設定が完了済みかどうか確認してください。既に REGHC がセットアップされている場合、再構成せずに有効にできます。

## 4 PMIC (スイッチングレギュレータ)

- PWR\_REGHC\_CTL.REGHC\_CONFIGURED が“0”の場合、設定は完了していません。(7)へ進んでください。
2. PMIC イネーブル制御のため GPIO を設定してください。この場合、LV リセット前の PMIC イネーブル状態が再設定されます。以下に GPIO 再設定の例を示します。
    - a.  $\sim$  (PWR\_REGHC\_CTL2.REGHC\_EN ^ PWR\_REGHC\_CTL.REGHC\_PMIC\_CTL\_POLARITY) の値を GPIO データレジスタに書き込んでください。
    - b. GPIO を出力に設定してください。
  3. ハンドオーバー完了を待ってください。  
 PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = “0” および PWR\_REGHC\_STATUS.REGHC\_ENABLED = PWR\_REGHC\_CTL2.REGHC\_EN まで待ってください。
  4. Reset Recovery で LoadRegulatorTrims API を呼び出してください。
    - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
    - Use case = 3 (Reset Recovery)
  5. LoadRegulatorTrims API の完了を待ってください。ステータスコードが 0xAXXXXXXX ではない場合、システムにしがって適切なエラー処理をしてください。
  6. MCU が PMIC で動作中かどうか確認してください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“1”の場合、PMIC が電源供給します。(17)へ進んでください。PWR\_REGHC\_STATUS.REGHC\_ENABLED が“0”の場合、内部レギュレータが電源供給します。(9)へ進んでください。
  7. 以下のパラメータで ConfigureRegulator API を呼び出してください。
    - Opcode = 0x15 (ConfigureRegulator API 呼び出し)
    - RadjValue = 0 (未使用)
    - VADJ trim value = 16
    - Vadj = 1 (固定)
    - UseRadj = 0 (未使用)
    - UseLinReg = 0 (PMIC イネーブル後、内部 Active レギュレータは無効です)
    - DeepSleep = 1 (PMIC は、DeepSleep 電源モード中に有効です)
    - Reset Polarity = 0 (PMIC PG 信号は、“0” でパワーバッド状態を示します)
    - Enable Polarity = 1 (PMIC は“1”で有効です)
    - Operating Mode = 1 (外部 PMIC)
    - WaitCount = 0 (0x1FF 以下)

**注:** この設定では、電流制限抵抗が EXT\_PS\_CTL1 (PMIC イネーブル信号出力) とロードスイッチ間に配置されます。この抵抗とロードスイッチの容量によって、ロードスイッチがすぐにオンされない場合があります。したがって、ロードスイッチが完全にオンされるまでハンドオーバー完了を待つ必要があります。このロードスイッチがオンされるまでの待機時間はハードウェアタイマを使用して待つことができます。この待機時間は、ConfigureRegulator API の WaitCount を使用して設定できます。必要な待機時間の設定については、[ロードスイッチ](#)を参照してください。
  8. ConfigureRegulator API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
  9. 0x1FF を超える WaitCount を設定する必要がある場合、PWR\_REGHC\_CTL.REGHC\_PMIC\_STATUS\_WAIT レジスタを設定してください。そうでない場合、ConfigureRegulator API により設定できるため、このプロセスは不要です。詳細については、[システムコール API](#) の ConfigureRegulator API を参照してください。
  10. PMIC を有効にするために GPIO を“1”に設定してください。その後、PMIC は有効になります。
  11. PWR\_REGHC\_STATUS.REGHC\_PMIC\_STATUS\_OK が“1”になるまで待ってください。これは、PMIC がパワーグッド状態になったことを示します。PMIC が十分に安定する前に PMIC がパワーグッド状態を示す場合、ソフトウェアは PMIC 出力を待つ必要があります。
- 以下の手順に従って、PMIC で REGHC を有効にします。

## 4 PMIC (スイッチングレギュレータ)

12. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 0 (Non-blocking call)
  - Select regulator = 0 (外部電源に切り替えます)
  - Operating Mode = 1 (外部 PMIC) このパラメータは ConfigureRegulator API の Operating Mode と一致する必要があります。

**注:** Non-blocking call での SwitchOverRegulators API 呼び出しは、PWR\_CTL2.DPSLP\_REG\_DIS を"1"に設定しません。詳細については、[ロードスイッチ](#)を参照してください。
13. SwitchOverRegulators API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
14. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0 および PWR\_REGHC\_STATUS.REGHC\_ENABLED = 1 になるまで待ってください。
15. Force trim setting で LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 0 (Force trim setting)
  - Operating Mode = 1 (外部電源)

これは、内部レギュレータの出力状態を外部状態に変更します。
16. LoadRegulatorTrims API の完了を待ってください。ステータスコードが"0xAXXXXXXX"以外の場合、システムに応じて適切なエラー処理をしてください。
17. デバイスは PMIC で動作します。Active レギュレータはオフ、DeepSleep レギュレータは外部状態で動作しています。

### PMIC から内部レギュレータへのハンドオーバー

[図 42](#) に、PMIC から内部レギュレータへのハンドオーバーを示します。

#### 4 PMIC (スイッチングレギュレータ)

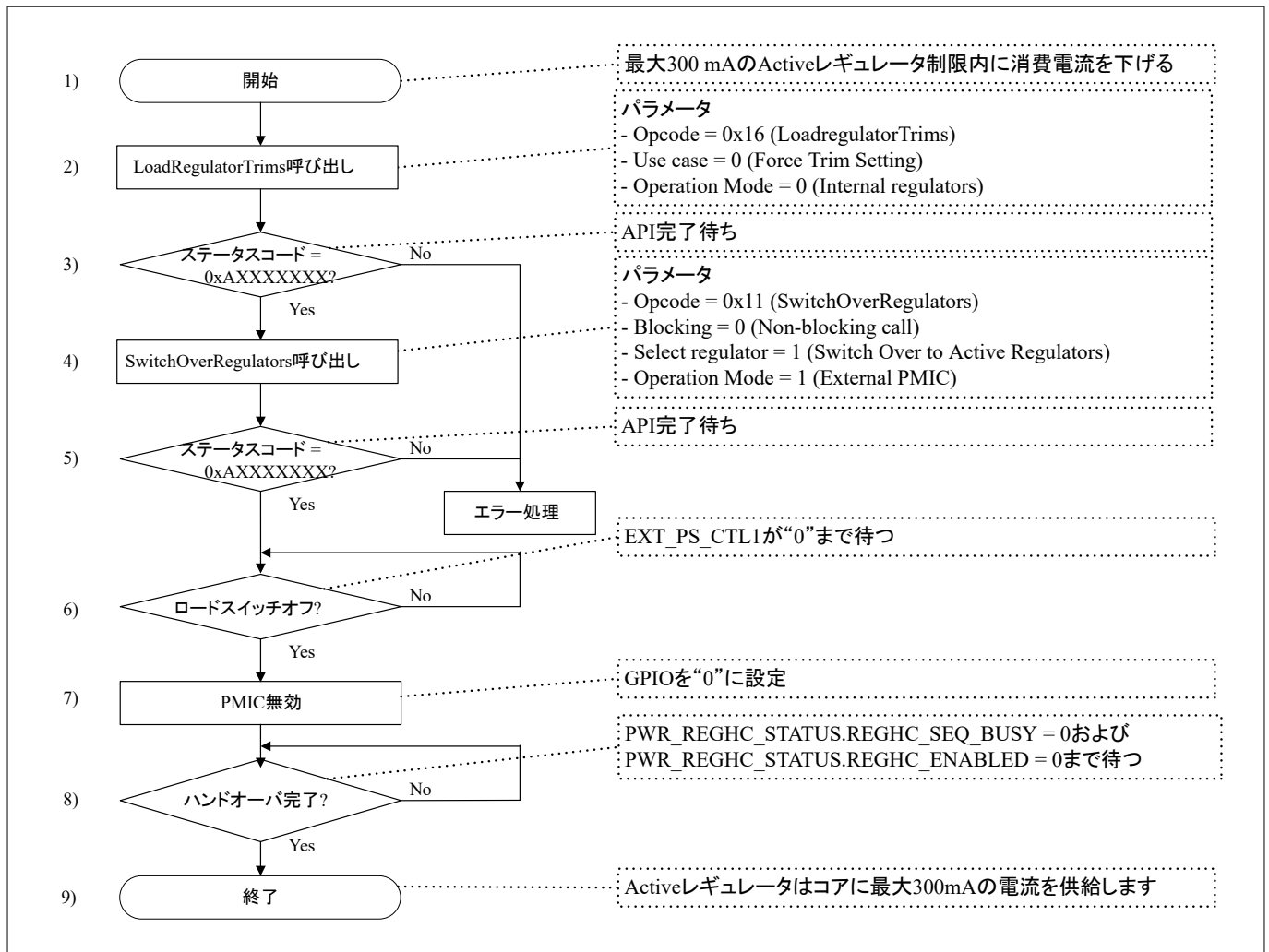


図 42 ロードスイッチ 2 構成での PMIC から内部レギュレータへのハンドオーバーフロー(ケース 3)

PMIC から内部レギュレータにハンドオーバーする場合

1. 最大 300 mA の Active レギュレータの制限まで消費電流を下げてください。
2. Force trim setting で LoadRegulatorTrims API を呼び出してください。
  - Opcode = 0x16 (LoadRegulatorTrims API 呼び出し)
  - Use case = 0 (Force trim setting)
  - Operating Mode = 1 (内部レギュレータ)
 これは内部レギュレータの出力状態を内部状態に変更します。
3. LoadRegulatorTrims API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
4. 以下のパラメータで SwitchOverRegulators API を呼び出してください。
  - Opcode = 0x11 (SwitchOverRegulators API 呼び出し)
  - Blocking = 0 (Non-blocking call)
  - Select regulator = 1 (Active レギュレータに切り替えます)
  - Operating Mode = 1 (外部 PMIC)
5. SwitchOverRegulators API の完了を待ってください。ステータスコードが“0xAXXXXXXX”以外の場合、システムに応じて適切なエラー処理をしてください。
6. ソフトウェアタイマの完了を待ってください。これは、ロードスイッチが完全にオフになるまでの時間を待機します。

## 4 PMIC (スイッチングレギュレータ)

**注:** この設定では、電流制限抵抗が EXT\_PS\_CTL1 (PMIC イネーブル信号出力) とロードスイッチ間に配置されます。この抵抗とロードスイッチの容量によって、ロードスイッチがすぐにオフされない場合があります。したがって、ロードスイッチが完全にオフされるまでハンドオーバー完了を待つ必要があります。ソフトウェアによってタイマを実装しロードスイッチがオフになるまで待ってください。必要な待機時間の設定については、[ロードスイッチ](#)を参照してください。

7. PMIC を無効にするため GPIO を“0”に設定してください。
8. PWR\_REGHC\_STATUS.REGHC\_SEQ\_BUSY = 0 および PWR\_REGHC\_STATUS.REGHC\_ENABLED = 0 になるまで待ってください。
9. デバイスは、内部レギュレータで動作しています。

### 4.5 部品選定

ここでは、部品選定について説明します。

#### 4.5.1 PMIC の周辺部品

PMIC 直接接続では、PMIC の出力容量は  $V_{CCD}$  容量と共有されます。出力コンデンサの選択の詳細については、[部品選定](#)を参照してください。

また、出力電圧設定抵抗は、自身のバイアス電流を考慮する必要があります。このバイアス電流は、DeepSleep 電源モード中であっても発生することに注意してください。

その他の PMIC 部品に関しては、PMIC のデータシートを参照してください。

#### 4.5.2 ロードスイッチ

[表 19](#) にロードスイッチの要件を示します。

**表 19**                      **ロードスイッチ要求仕様**

パラメータ	記号	値	単位
ソース-ドレイン電圧	$V_{DSO}$	$\geq V_{DDD}$	V
ソース-ゲート電圧	$V_{GSO}$	$\geq V_{DDD}$	V
ドレイン電流	$I_D$	$\geq 0.6$ (CYT6BJ を除く) $\geq 0.8$ (CYT6BJ)	A

ロードスイッチの  $R_{ON}$  により、 $V_{CCD}$  電圧降下が発生する場合があります。 $R_{ON}$  による  $V_{CCD}$  電圧降下は、[式 3](#) で計算されます。

式 3

$$V_{CCD\_drop} = R_{ON\_max} \times I_{VCCD\_max}$$

ここで、

- $V_{CCD\_drop}$ :  $V_{CCD}$  電圧降下 (V)
- $R_{ON\_max}$ : ロードスイッチの最大オン抵抗 ( $\Omega$ )
- $I_{VCCD\_max}$ : 最大  $V_{CCD}$  負荷電流 (A)

この電圧降下を含む MCU の  $V_{CCD}$  要件に適合するロードスイッチを選択してください。電流発生時の電圧降下を考慮して、低  $R_{ON}$  を推奨します。

オン状態でのロードスイッチのソース-ゲート電圧は、[式 4](#) のロードスイッチのソース-ゲート最小電圧です。ロードスイッチは、この電圧でオンする必要があります。

式 4



## 4 PMIC (スイッチングレギュレータ)

$$V_{GS\_min} = V_{G\_min} - V_{CCD\_max}$$

- $V_{GS\_min}$ : ロードスイッチのソース-ゲート最小電圧 (V)
- $V_{G\_min}$ : ロードスイッチのグランド-ゲート最小電圧 (V)
  - ロードスイッチ 1 構成での PMIC の場合、 $V_{G\_min}$  はパワーグッドのプルアップ最小電圧です。
  - ロードスイッチ 2 構成での PMIC の場合、 $V_{G\_min}$  は、EXT\_PS\_CTL1="H" の最小電圧です。
- $V_{CCD\_max}$ : 最大  $V_{CCD}$  電圧 (V)

ロードスイッチ 2 構成での PMIC の場合、EXT\_PS\_CTL1 がロードスイッチのゲートを駆動するために一時的に過電流を発生する場合があります。この場合、EXT\_PS\_CTL1 の最大電流内にするために電流制限抵抗が必要です。(詳細は、ボディコントローラハイファミリデータシート [1] の SID23 および SID30 を参照してください。)

ロードスイッチのゲートに必要な抵抗は、式 5 で計算されます。

式 5

$$R_{Load\ switch\ gate} \geq V_{DDD\_max} \div I_{EXT\_PS\_CTL1\_peak}$$

ここで、

- $R_{Load\_switch\_gate}$ : EXT\_PS\_CTL1 電流制限抵抗 ( $\Omega$ )
- $I_{EXT\_PS\_CTL1\_peak}$ : ロードスイッチのゲートを駆動する EXT\_PS\_CTL1 のピーク電流 (A)
- $V_{DDD\_max}$ : 最大  $V_{DDD}$  電圧 (V)

例えば、 $V_{DDD}$  の最大電圧が 5.5 V の場合、電流を 1 mA 以下に制限するためには 5.5 k $\Omega$  以上の抵抗が必要です。

さらに、この電流制限抵抗とロードスイッチの  $C_{iss}$  によってロードスイッチゲートの電圧波形は放電曲線になります。その結果、ロードスイッチのオン/オフ状態の遷移が遅延します。この時定数は式 6 で計算されます。

式 6

$$T_{EXT\_PS\_CTL1} = R_{Load\ switch\ gate} \times C_{iss\_loadswitch}$$

ここで、

- $T_{EXT\_PS\_CTL1}$ : ロードスイッチのゲートと電流制限抵抗の時定数 (s)
- $C_{iss\_load\_switch}$ : ロードスイッチの入力容量 (F)
- $R_{Load\_switch\_gate}$ : EXT\_PS\_CTL1 電流制限抵抗 ( $\Omega$ )

EXT\_PS\_CTL1 高電圧の 95%になるロードスイッチのゲート遅延時間は、 $\tau$  値の 3 倍です。EXT\_PS\_CTL1 高電圧の 5%になるロードスイッチのゲート遅延時間も同様です。

### 4.6 レイアウト設計ガイドライン

以下に、PCB レイアウト設計のためのガイドラインを示します。

- PMIC の出力とロードスイッチは、MCU の  $V_{CCD}$  端子はできるだけ近くに配置してください。
- その他の注意事項については、PMIC サプライヤの PMIC レイアウトガイドラインに従ってください。



## 5 Appendix A. ハードウェアシーケンス

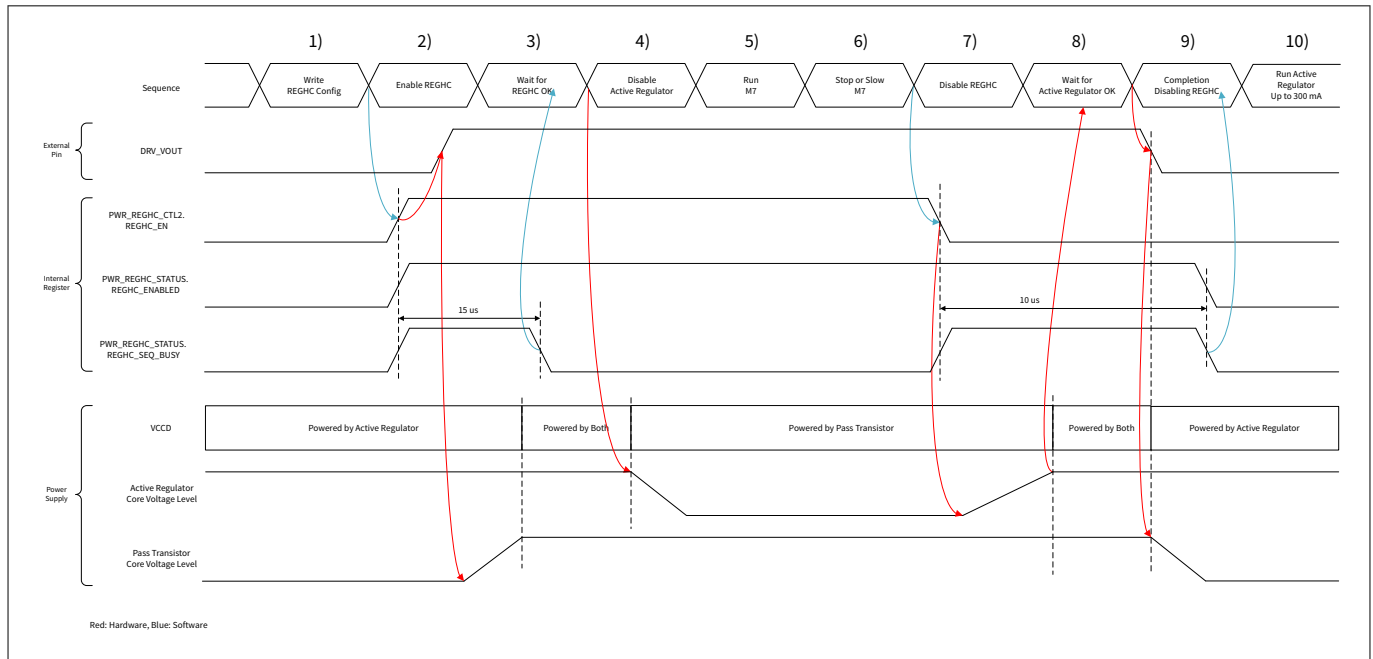
### 5 Appendix A. ハードウェアシーケンス

以下に CYT3B/4B/6B シリーズのシーケンス例を示します。

パストランジスタと Active レギュレータ間のハンドオーバーシーケンス

**注:** CYT6B シリーズはこの設定をサポートしていません。

図 43 に、パストランジスタと Active レギュレータ間のハードウェアのハンドオーバーシーケンスを示します。



**図 43** パストランジスタと Active レギュレータ間のハンドオーバーシーケンス

MCU は起動時 Active レギュレータで開始します。

1. SW: 外部パストランジスタモードで REGHC を設定します。
2. SW: REGHC (パストランジスタ) を有効にします。次に、REGHC は DRV\_VOUT ピンを駆動してパストランジスタ動作を開始します。
3. SW: 外部コア電圧レベルに達するまで、REGHC 完了フラグを待ちます。
4. HW: Active レギュレータを無効にします。
5. MCU は REGHC (パストランジスタ) で動作します。

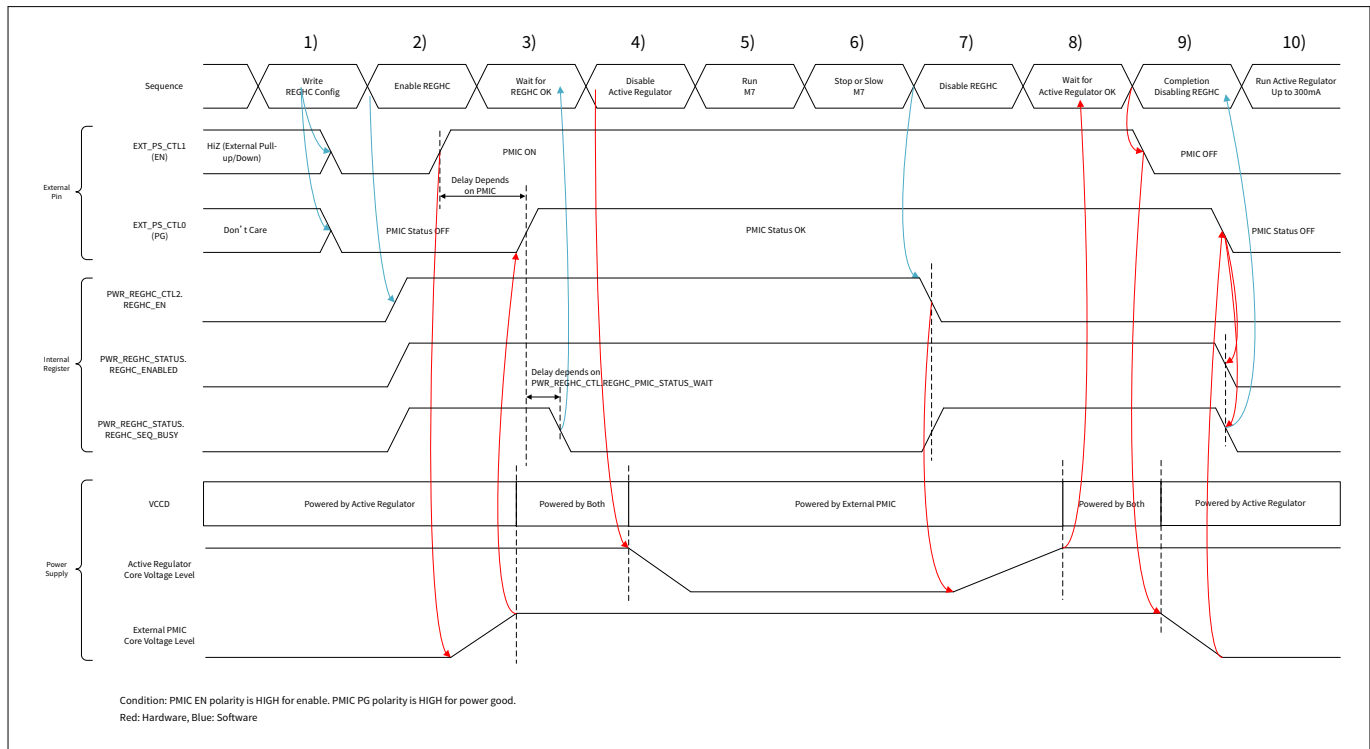
**注:** アプリケーションソフトウェアは、VCCD 電圧のアンダーシュートを回避するために動作クロック周波数を段階的に上げる必要があります。

6. Active レギュレータに移行する前にアプリケーションの消費電流を Active レギュレータの制限内にする必要があります。
7. SW: REGHC (パストランジスタ) を無効にします。
8. HW: Active レギュレータが完全に動作するまで待ちます。
9. HW: REGHC の無効化を完了します。
10. MCU は Active レギュレータで動作します。

## 5 Appendix A. ハードウェアシーケンス

### PMIC と Active レギュレータ間のハンドオーバーシーケンス

図 44 に、PMIC と Active レギュレータ間のハードウェアのハンドオーバーシーケンスを示します。



**図 44** PMIC と Active レギュレータ間のハンドオーバーシーケンス

MCU は起動時 Active レギュレータで開始します。

1. SW: PMIC で REGHC を設定します。
2. SW: PMIC で REGHC を有効にします。その後 PMIC は動作を開始します。
3. SW: REGHC 完了フラグを待ちます。PMIC が有効後、EXT\_PS\_CTL0 がパワーグッド状態になると REGHC 完了フラグがセットされます。
4. ターゲット電圧の 100%に達する前のソフトスタート中に EXT\_PS\_CTL0 がパワーグッド状態になる場合、内部レギュレータがオフになりアプリケーション(例えば PLL オン)の消費電流が直ちに増加すると V<sub>CCD</sub> 電圧がアンダーシュートする場合があります。この問題を軽減するために REGHC\_PMIC\_STATUS\_WAIT を使用して PMIC の安定時間を確保できます。これはハードウェアシーケンサが内部 Active レギュレータを無効にする前の安定待ち時間を設定するために使用できます。ハードウェアは Active レギュレータを無効にします。V<sub>CCD</sub> は外部 PMIC から電源供給されます。
5. MCU は外部 PMIC で動作します。

**注:** アプリケーションソフトウェアは、V<sub>CCD</sub> 電圧のアンダーシュートを回避するために動作クロック周波数を段階的に上げる必要があります。

6. MCU は Active レギュレータに移行する前に Active レギュレータの制限内で動作する必要があります。
7. SW: PMIC から内部レギュレータへのハンドオーバー手順を開始します。REGHC(PMIC)を無効にします。
8. HW: Active レギュレータが完全に動作するまで待ちます。
9. HW: 外部 PMIC の無効化を完了します。
10. MCU は Active レギュレータで動作します。

## 用語集

## 用語集

用語	説明
API	Application programming interface
BOD	Brown-out-detection (電圧低下検出)
BOM	Bill of material (部品コスト)
EP	Exposed pad (放熱パッド)
GPIO	General purpose I/O (汎用入出力)
HVD	High-voltage-detection (高電圧検出)
HV reset	High-voltage reset (高電圧リセット)
LDO regulator	Low-dropout regulator (低ドロップアウトレギュレータ)
LV reset	Low-voltage reset (低電圧リセット)
MCU	Microcontroller unit (マイクロコントローラ)
OCD	Over-current-detection (過電流検出)
OVD	Over-voltage-detection (過電圧検出)
PCB	Printed circuit board (プリント基板)
PG	Power good output signal from PMIC (PMIC パワーグッド出力信号)
PFM	Pulse frequency modulation (パルス周波数変調)
PMC	Power management integrated circuit (パワーマネージメント IC)
PWM	Pulse width modulation (パルス幅変調)
POR	Power-on-reset (パワーオンリセット)
REGHC	High current regulator controller (高電圧レギュレータコントローラ)
SROM	Supervisory ROM (スーパバイザリ ROM)
WFI	Wait for interrupt CPU instruction (割込み待ち CPU 命令)

## 関連ドキュメント

### 関連ドキュメント

以下は、TRAVEO™ T2G ファミリシリーズのデータシートとテクニカルリファレンスマニュアルです。これらの資料の入手については[テクニカルサポート](#)に連絡してください。

- [1] デバイスデータシート:
  - [CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family](#)
  - [CYT4DN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family \(Doc No. 002-24601\)](#)
  - [CYT3BB/4BB datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family](#)
  - [CYT6BJ datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family \(Doc No. 002-33466\)](#)
  - [CYT3DL datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family \(Doc No. 002-27763\)](#)
- [2] アーキテクチャテクニカルリファレンスマニュアル:
  - [TRAVEO™ T2G automotive body controller high family architecture technical reference manual \(TRM\)](#)
  - [TRAVEO™ T2G automotive cluster 2D family architecture technical reference manual \(TRM\) \(Doc No. 002-25800\)](#)
- [3] レジスタテクニカルリファレンスマニュアル:
  - [TRAVEO™ T2G automotive body controller high registers technical reference manual \(TRM\) for CYT4BF](#)
  - [TRAVEO™ T2G automotive body controller high registers technical reference manual \(TRM\) for CYT3BB/4BB](#)
  - [TRAVEO™ T2G automotive body controller high registers technical reference manual \(TRM\) for CYT6BJ \(Doc No. 002-36068\)](#)
  - [TRAVEO™ T2G automotive cluster 2D registers technical reference manual \(TRM\) for CYT4DN \(Doc No. 002-25923\)](#)
  - [TRAVEO™ T2G automotive cluster 2D registers technical reference manual \(TRM\) for CYT3DL \(Doc No. 002-29854\)](#)
- [4] アプリケーションノート:
  - [AN220118 - TRAVEO™ T2G ファミリ MCU スタートアップガイド](#)
  - [AN201006 - Thermal considerations and parameters](#)

## 改訂履歴

## 改訂履歴

版数	発行日	変更内容
**	2020-08-13	これは英語版 002-26698 Rev. **を翻訳した日本語版 Rev. **です。英語版の改訂内容: Initial release
*A	2020-11-09	これは英語版 002-26698 Rev. *A を翻訳した日本語版 Rev. *A です。英語版の改訂内容: Deleted DRV_VOUT function in PMIC configuration Changed WaitCount parameter setting and flow in ConfigureRegulator API Clarification of DeepSleep power mode entry instruction.
*B	2021-02-08	これは英語版 002-26698 Rev. *B を翻訳した日本語版 Rev. *B です。英語版の改訂内容: Added part number.(CYT3 series) Added description for Cluster 2D family
*C	2021-10-11	これは英語版 002-26698 Rev. *C を翻訳した日本語版 Rev. *C です。英語版の改訂内容: Added note in section システムコール API Changed pass transistor configuration
*D	2023-09-06	これは英語版 002-26698 Rev. *D を翻訳した日本語版 Rev. *D です。英語版の改訂内容: Added part number.(CYT6B series) Added a note for CYT6B series below - 外部電源構成の違い - 外部電源と内部レギュレータ間のハンドオーバ - Appendix A. ハードウェアシーケンス Added description for CYT6B series in ハードウェア構成 Change Drain current value in 表 19 Updated the References section
英語版(*E)	-	この版は英語版のみです。英語版の改訂内容: Template update Updated 図 1 Updated 図 38
*E	2024-12-10	これは英語版 002-26698 Rev. *F を翻訳した日本語版 Rev. *E です。英語版の改訂内容: Updated 図 41

## Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2024-12-10**

**Published by**

**Infineon Technologies AG**  
**81726 Munich, Germany**

**© 2024 Infineon Technologies AG**  
**All Rights Reserved.**

**Do you have a question about any aspect of this document?**

**Email: [erratum@infineon.com](mailto:erratum@infineon.com)**

**Document reference**  
**IFX-lrl1686299238044**

## 重要事項

本手引書に記載された情報は、本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

## 警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。