TRAVEO™ファミリのリセット手順およびローパワーモードにおける RAM データ保持方法

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では、TRAVEO™ T2G ファミリ MCU でソフトウェアリセットまたはローパワーモード移行の発生の際、確実に RAM 保持を行うための手順を説明します。

対象者

本書は TRAVEO™ T2G ファミリを使用するすべての人を対象にします。

目次

1 はじめに

このアプリケーションノートでは、インフィニオン TRAVEO™ T2G ファミリの CYT2/CYT3/CYT4 シリーズ MCU のリセット、およびローパワーモード移行における RAM 保持を保証するための手順を説明します。

TRAVEO™ T2G ファミリでは、RAM のアクセス状態に関係なく非同期でリセットが発生します。したがって、リセットが動作中に発生した場合、RAM データは破壊される場合があります。さらに、デバイスのパワーモードが、Active からローパワーモードへ移行する場合も RAM 保持のための適切な手順を行う必要があります。このドキュメントでは、システム設計においてソフトウェアリセットまたはローパワーモードへ移行後に RAM データを確実に保持する手順について説明します。ただし、Hibernate モードにおいては、RAM データは保持できません。そのため、RAM データはアプリケーションフラッシュへ一旦移行する必要があります。Active モードへ復帰後、RAM データをアプリケーションフラッシュから RAM へ戻す必要があります。この場合、移行データをバックアップメモリデータと定義します。

このアプリケーションノートで使用される機能と用語については、architecture technical reference manual (TRM) の"SRAM Interface"章および"Work Flash"章を参照してください。

2 RAM 保持手順概要

2.1 リセット手順

図 1 に、リセットが発生した際の RAM 保持のフローを示します。この例は RAM0 データ保持の場合を示します。

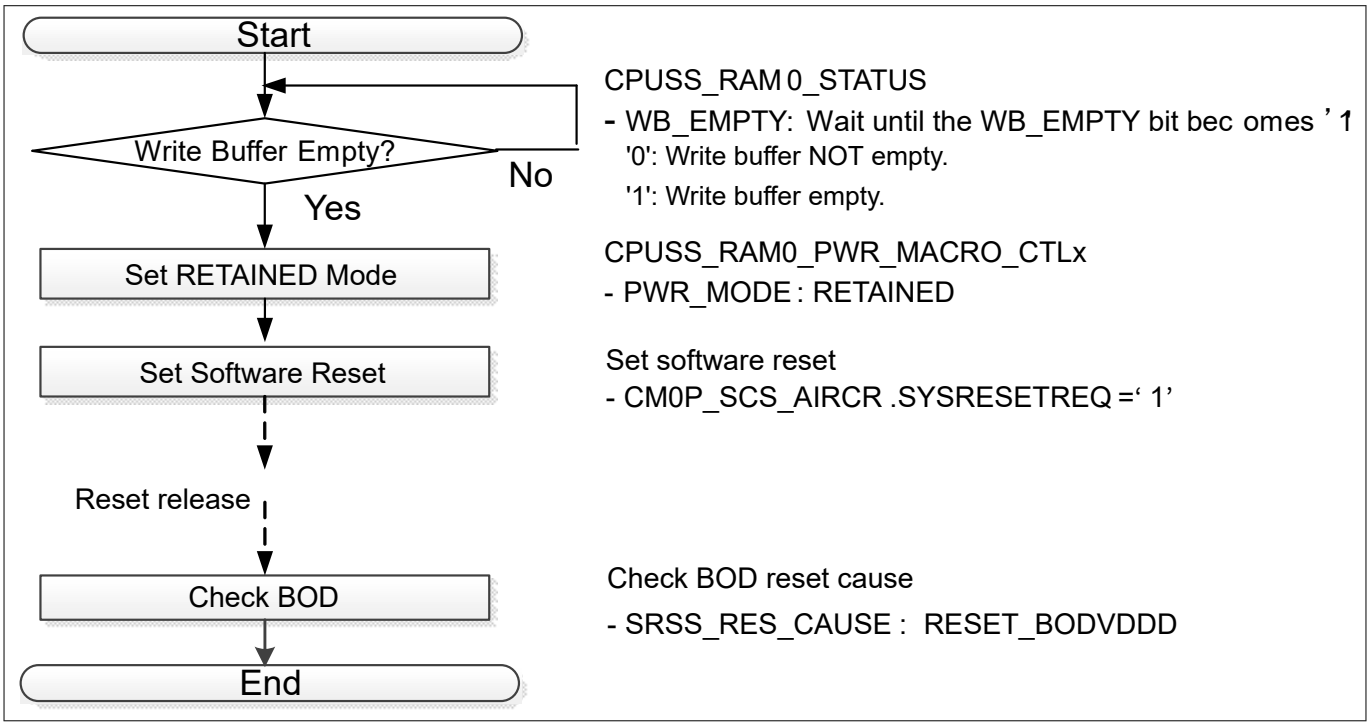

図 1 RAM0 保持手順例

はじめに、RAM0 のケースでは、CPUSS_RAM0_STATUS レジスタの WB_EMPTY ビットにより書き込みバッファステータスを確認します。WB_EMPTY ビットは、書き込みバッファ内のデータの有無を示します。

CYT2 シリーズの MCU では、ECC は 32 ビットデータに付加されます。したがって、部分的な AHB-Lite 書込み (8 ビット/16 ビット) が RAM に行われると、欠損データは RAM から読み出されます。そして、欠損データと部分的な書込みデータをマージし 32 ビットの完全なデータを生成します。ECC は、32 ビットの完全なデータに対し計算され、その 32 ビットデータを RAM へ上書きします。

CYT3/CYT4 シリーズの MCU には、AXI バスインタフェースがあります。AXI バスインタフェースの ECC は 64 ビットデータに付加されます。そのため、部分的な書込み (8 ビット/16 ビット/32 ビット) が RAM に行われると、欠損データは RAM から読み出されます。そして、欠損データと部分的な書込みデータをマージし 64 ビットの完全なデータを生成します。ECC は、64 ビットの完全なデータに対して計算されます。

このような動作に書き込みバッファが使用されます。このため、RAM へまだ書き込まれていないデータが、書き込みバッファに存在している可能性があります。書き込みバッファにあるデータの未書込み状態を回避するため、書き込みバッファのステータスを確認する必要があります。

もし有効なデータが存在する場合、RAM0 へ書き込むまで待ちます。書き込みバッファに有効なデータが無い場合、CPUSS_RAM0_PWR_MACRO_CTLx レジスタの PWR_MODE ビットを RETAINED モードに設定してください。最後にソフトウェアリセットを設定してリセットを発生します。この手順によって、リセットによる RAM 保持が可能です。しかし、電源電圧が Brown-Out Detection (BOD: 2.7 V) レベルより低下した場合、RAM0 データは保持できません。そのため、BOD が発生しなかったことをリセット復帰後に確認する必要があります。

表 1 に RAM0 ステータスレジスタを示します。ソフトウェアリセットが発生する前に WB_EMPTY ビットが、‘1’に設定されていることを確認する必要があります。

表 1 RAM0 ステータスレジスタ

レジスタ	ビットフィールド	ビット値	説明
CPUSS_RAM0_STATUS	WB_EMPTY [0]	0	書き込みバッファは空ではありません
		1	書き込みバッファは空です

表 2 は、1 つのマクロでシステム RAM0 の電源ステータスを制御する電源制御レジスタを示します。

表 2 電源制御レジスタ

レジスタ	ビットフィールド	ビット値	説明
CPUSS_RAM0_PWR_MACRO_CTLx	PWR_MODE [1:0][1)]	0	OFF モード: SRAM を OFF にします。これによりアレイ電源と周辺電源の両方が OFF になります。SRAM メモリの内容は失われます。
		1	予約済み
		2	RETAINED モード: RETAINED モードで SRAM を保持します。これにより、SRAM 周辺電源は OFF になりますが、メモリの内容を保持するためにアレイ電源は ON です。SRAM の内容は、DeepSleep システム電源モードで保持されます。
		3	ENABLE モード: 通常動作のため SRAM は有効です。SRAM の内容は、DeepSleep システム電源モードで保持されます。(初期値)

このレジスタは、CPUSS システムの RAM0 コントローラ用です。これは RAM0 RETAINED モードの設定に使用されます。

2.2 ローパワーモード (DeepSleep モード) 移行手順

図 2 は、ローパワーモード移行における RAM 保持のフローを示します。この方法では、ローパワーモードへ移行する際に RAM 保持のための設定が行われます。Write バッファのステータス確認、および RAM の RETAINED モード設定手順は同じです。MCU がローパワーモードになった際、メインプログラムの実行は停止します。もし、Wakeup 割込みが発生したら、MCU は Active モードに復帰します。

1 PWR_MODE ビット フィールドを設定するには、CPUSS_RAM0_PWR_MACRO_CTLx レジスタのワード アクセスを使用します。詳細については、registers TRM を参照してください。

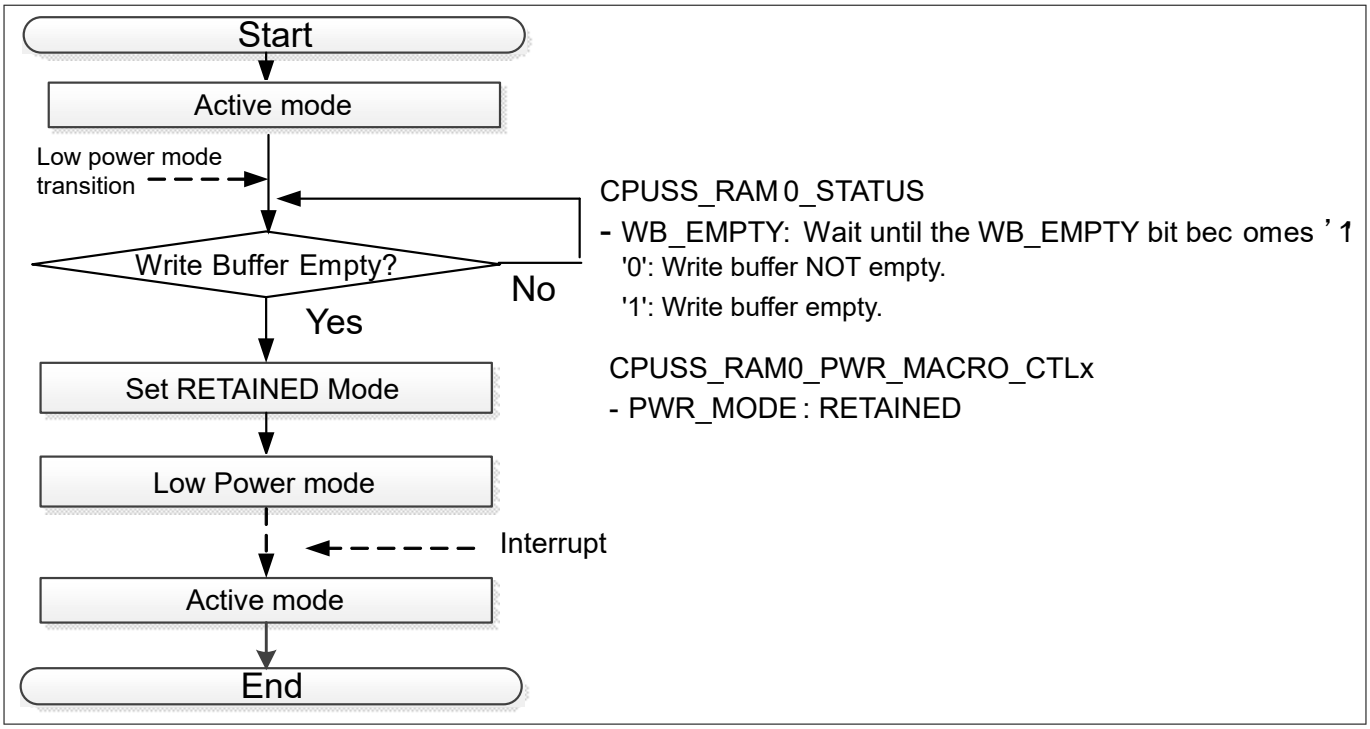

図 2　　ローパワーモードの RAM0 保持手順例

2.3　　ローパワーモード(Hibernate モード) 移行手順

図 3 に、Hibernate モード移行においてバックアップメモリデータ がどのようにバックアップされるかを示します。この方法では、Hibernate モードに移行するとき、RAM のバックアップメモリデータがアプリケーションフラッシュに移行されます。MCU が Hibernate モードに入るとき、メインプログラムの実行が停止します。もし、Hibernate wakeup リセットが発生すると、MCU は、Active モードに復帰します。そして、バックアップメモリデータは、アプリケーションフラッシュから RAM に移行されます。RAM からアプリケーションフラッシュへのデータ移行およびその逆の移行は、ユーザソフトウェアによって制御されます。

意図しないバックアップデータの上書きを避けるために、データバックアップ中に許可していない他のプログラムがアプリケーションフラッシュと RAM のバックアップエリアにアクセスしないように考慮すべきです。さらに、RAM とアプリケーションフラッシュ間のデータ移行時間が、システム要件を満たしていることを確認する必要があります。

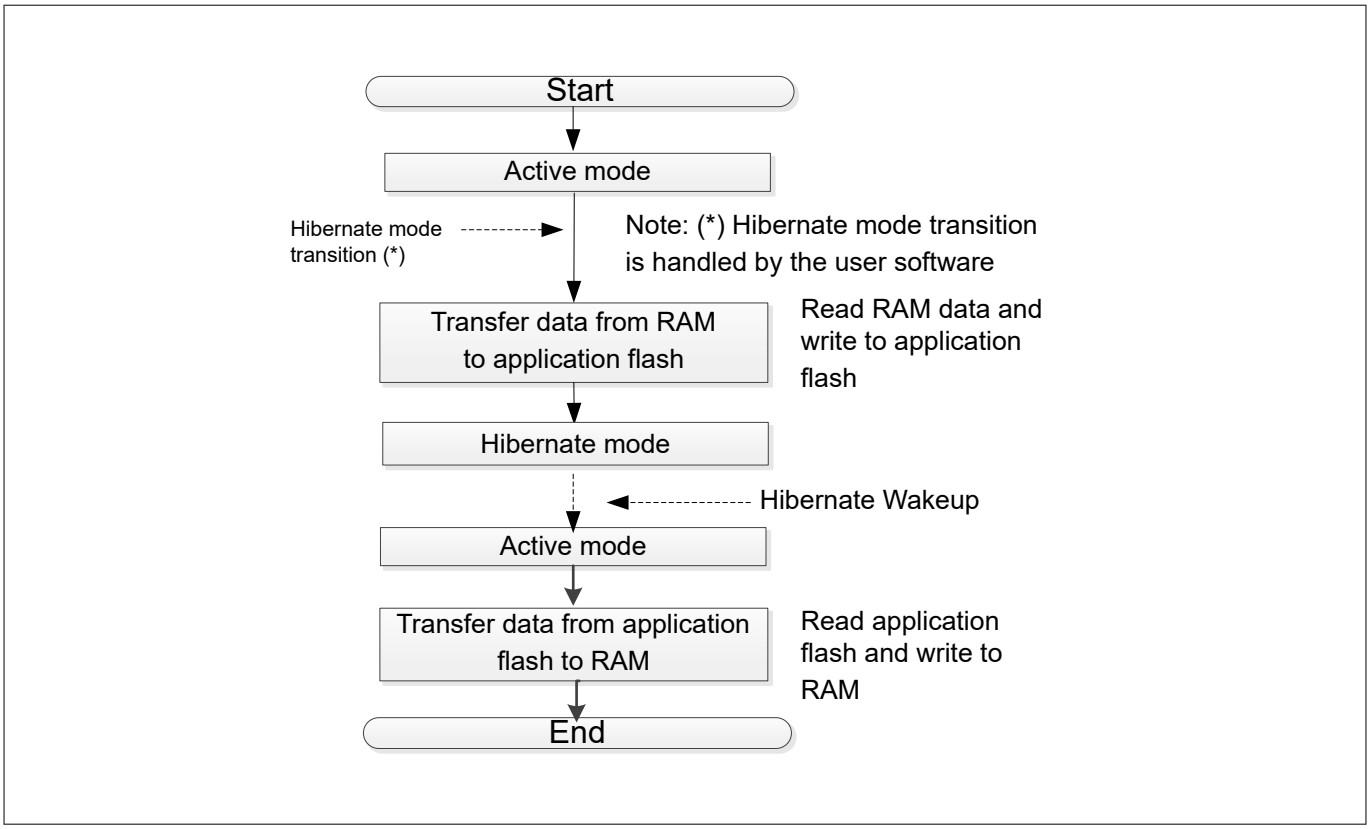

図 3　　Hibernate モードにおけるバックアップ手順例

3 リセットにおける RAM 保持手順

ここでは、2 つのリセット手順例をブロックダイヤグラム, タイミングチャート, およびフローチャートで示します。1 つは、低電圧検出 (LVD) 割込みを使用します。もう 1 つは、外部 LVD IC の外部リセット入力信号を使用します。

3.1 LVD 割込みを使用したリセット

このケースでは、LVD1 と LVD2 を使用します。LVD1 は、システム低レベル電圧検知で、リセットは RAM0 保持が保証された状態で発生します。また、LVD1 は VDDD 電源電圧の低下を検知ために使用されます。ユーザアプリケーションは、LVD2 の確認により開始します。LVD2 は、立ち上りトリップポイントの設定により VDDD 電源電圧の復旧確認のために使用されます。

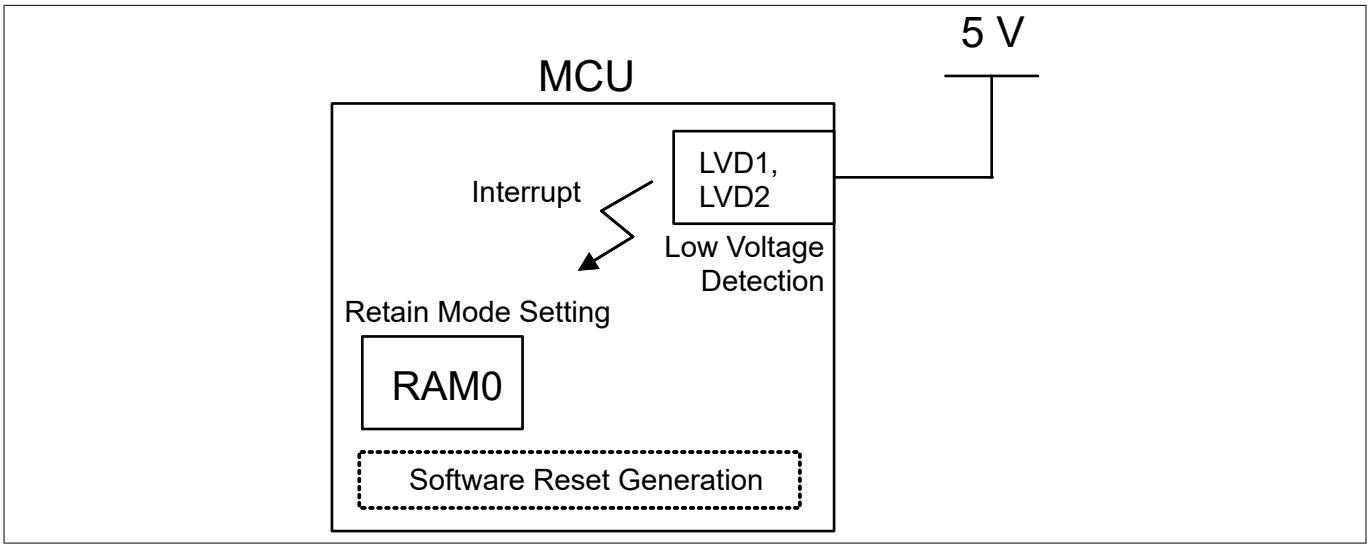

図 4 LVD 割込み方法による RAM0 保持のためのリセット手順のブロックダイヤグラム

図 4 において、LVD1 の立ち下りエッジ検知により、割込みが発生します。割込み発生後、RAM0 ステータスを確認し、RETAINED モードに設定する必要があります。このステップの後、ソフトウェアリセットを行います。MCU がリセット復旧後に、VDDD が LVD2 立ち上りエッジを超えていることを、SRSS_INTR レジスタで確認してください。最後に、BOD リセットが発生していないことを確認してください。もし、BOD リセットが発生していた場合、RAM0 データ保持は保証されません。したがって、RAM0 データを破棄する必要があります。

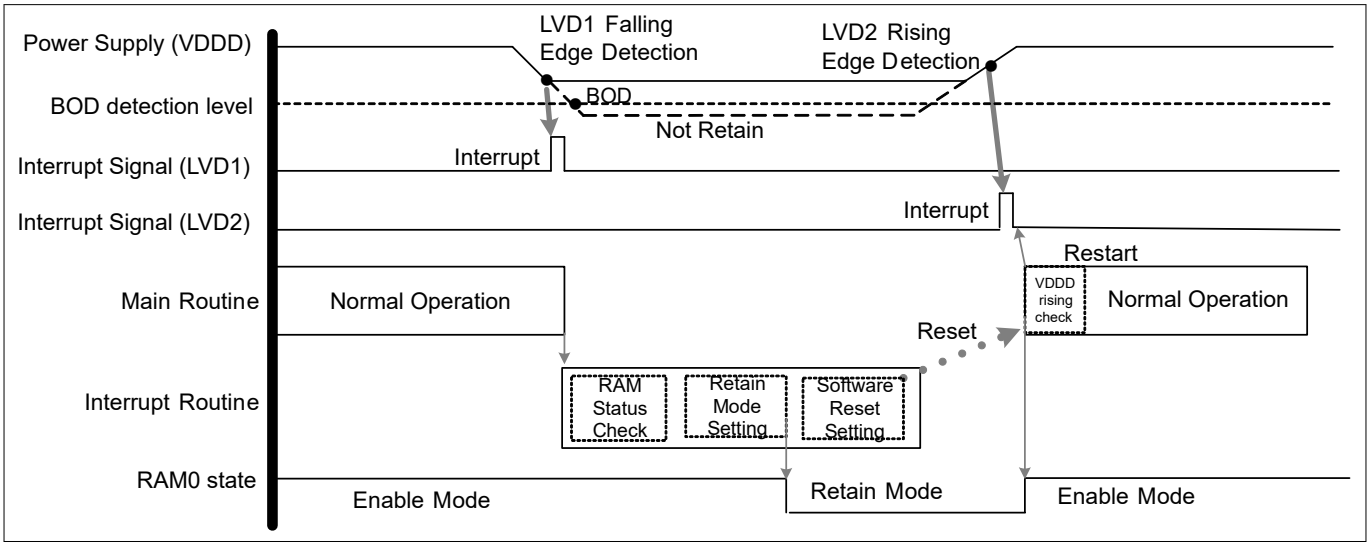

図 5 LVD 割込みによる RAM0 リセット手順タイミングチャート例

図 5 のように、VDDD が低下すると、割込みルーチンがコールされます。割込みルーチンは、RAM0 ステータス, RETAINED モード設定, およびソフトウェアリセット設定を行ってリセットを発生します。リセット完了後、LVD2 の検知によって VDDD の立ち上りを確認して通常動作に戻ります。

3.1.1 使用事例

ここでは、サンプルドライバライブラリ (SDL) を使用した使用事例に基づいて、LVD 割込みを使用してリセットを設定する方法について説明します。このアプリケーションノートに記載されているプログラムコードは、SDL に含まれるものです。SDL については、その他の参考資料を参照してください。

SDL には設定部とドライバ部があります。設定部では、主に目的の動作をさせるためのパラメーター値を設定します。ドライバ部は設定部のパラメーター値に基づいて各レジスタを設定します。このサンプルプログラムは CYT2B7 シリーズ用です。

使用事例:

- LVD1 の設定: 立ち下り検出時の閾値 4.7 V
- LVD2 の設定: 立ち上り検出時の閾値 4.9 V
- RAM モードの設定: 保持モード
- リセットの設定: ソフトウェア リセット

図 6 に、LVD 割込みによる RAM0 のリセット手順のフロー例を示します。

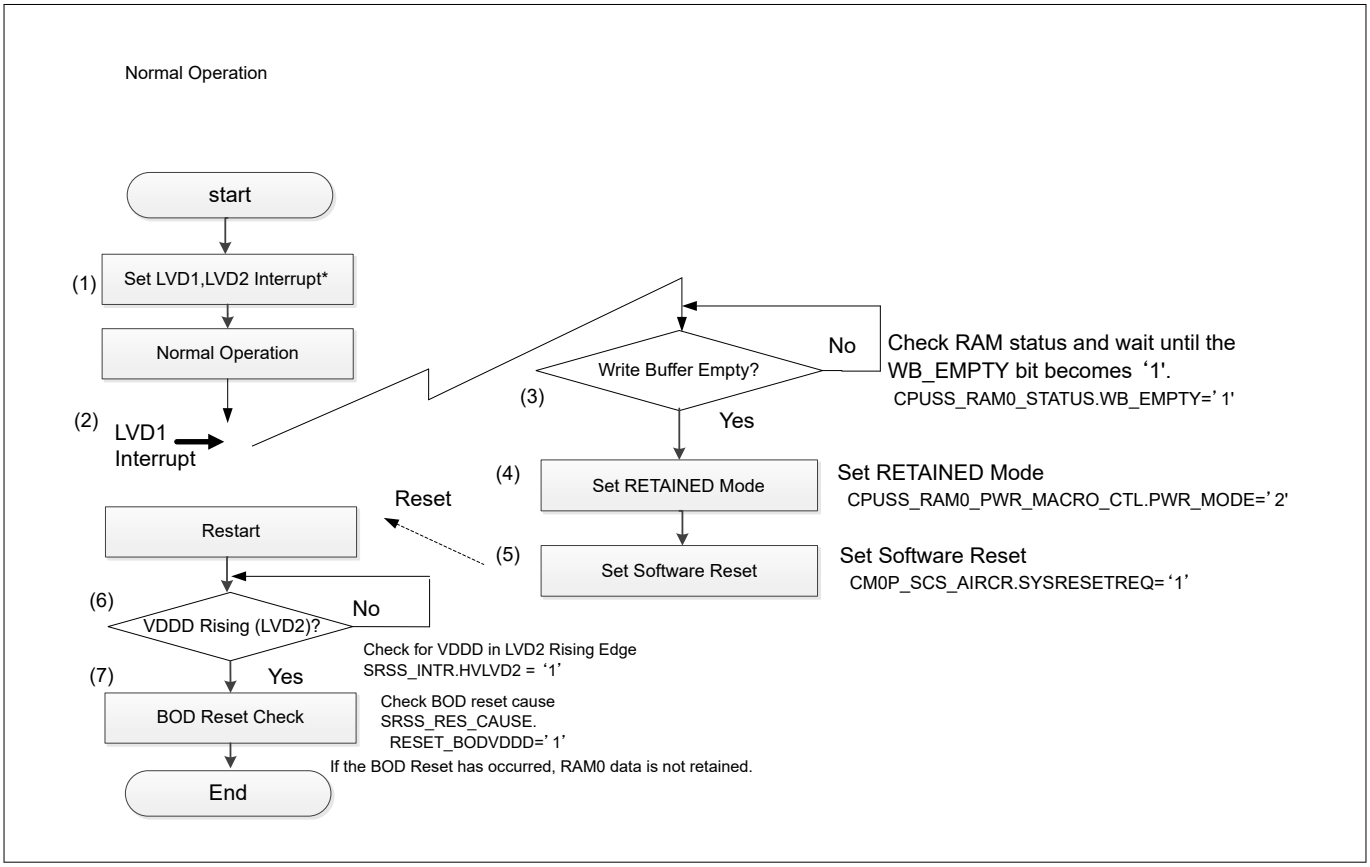

図 6 LVD 割込みよる RAM0 リセット手順フローチャート例

* 割込み設定の詳細については、architecture TRM の"Interrupts"章を参照してください。

1. LVD1 および LVD2 の割込みを有効にしてください。
2. VDDD の低下により低電圧が検出されると、LVD1 は割込みを CPU に対して発生します。
3. CPU が LVD 割き込みを検出したら、書き込みバッファのステータスを確認してください。

書き込みバッファにデータが存在する場合 (WB_EMPTY = '0')、書き込みバッファのデータが無くなるまで待機してください。

4. 書き込みバッファにデータが無くなると (WB_EMPTY = '1')、CPU は RETAINED モードに設定します。
5. CPU はソフトウェアリセットを発生します。
6. CPU は VDDD 電源電圧の復旧を確認します。
7. VDDD が回復したら、CPU は BOD リセットが発生していないことを確認します。BOD リセットが発生していない場合、RAM0 のデータは保持されます。しかし、BOD リセットが発生した場合、RAM0 データを破棄する必要があります。

LVD 割込み設定を使用したリセットにおける SDL の設定部のパラメーターを表 3 に、関数を表 4 に示します。

表 3 LVD 割込みパラメーターを使用したイニシャルリセット一覧

パラメーター	説明	値
user_led0_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led0_port_pin_cfg.Hsiom	I/O ピン配線の接続	USER_LED0_PIN_MUX
user_led0_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led0_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led0_port_pin_cfg. vtrip	入力バッファ モード	0ul,
user_led0_port_pin_cfg. slewRate	スルーレート	0ul
user_led0_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led1_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led1_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED1_PIN_MUX
user_led1_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led1_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led1_port_pin_cfg.Vtrip	入力バッファ モード	0ul
user_led1_port_pin_cfg. slewRate	スルーレート	0ul
user_led1_port_pin_cfg. driveSel	GPIO 駆動強度	0ul

表 4 LVD 割込み設定関数を使用したリセットの一覧

関数	説明	値
Cy_SysReset_GetResetReason(void)	最新のリセット要因を返します	-
Cy_LVD_ClearInterruptMask (cy_en_lvd_type_select_t lvdType)	LVD 割込みを無効にします。 lvdType: LVD タイプ	CY_LVD_TYPE_1 (for LVD1) CY_LVD_TYPE_2 (for LVD2)

(続く)

表 4 (続き) LVD 割込み設定関数を使用したリセットの一覧

関数	説明	値
Cy_LVD_SetThreshold (cy_en_lvd_type_select_t lvdType, cy_en_lvd_tripsel_t threshold)	VDDD 電圧を監視するための閾値を設定します。 lvdType: LVD タイプ threshold: 閾値	CY_LVD_TYPE_1(for LVD1), CY_LVD_THRESHOLD_4_7_V (for LVD1) CY_LVD_TYPE_2 (for LVD2), CY_LVD_THRESHOLD_4_9_V (for LVD2)
Cy_LVD_SetActionSelect (cy_en_lvd_type_select_t lvdType, cy_en_lvd_action_select_t lvdActionSelect)	プログラムされた方向で閾値を超えたときに実行されるアクションを設定します。 lvdType: LVD タイプ lvdActionSelect: アクション	CY_LVD_TYPE_1(for LVD1), CY_LVD_ACTION_INTR (for LVD1, LVD2) CY_LVD_TYPE_2(for LVD2),
Cy_LVD_SetEdgeSelect (cy_en_lvd_type_select_t lvdType, cy_en_lvd_edge_select_t lvdEdgeSelect)	閾値を超えたときにエッジトリガを設定します。 lvdType: LVD タイプ lvdEdgeSelect: エッジ選択	CY_LVD_TYPE_1(for LVD1), CY_LVD_EDGE_FALLING (for LVD1) CY_LVD_TYPE_2(for LVD2), CY_LVD_EDGE_RISING (for LVD2)
Cy_LVD_Enable (cy_en_lvd_type_select_t lvdType)	LVD の出力を有効にします。 lvdType: LVD タイプ	CY_LVD_TYPE_1(for LVD1) CY_LVD_TYPE_2(for LVD2)
Cy_LVD_SetInterrupt (cy_en_lvd_type_select_t lvdType)	LVD の割込みを生成するためにデバイスをトリガします。 lvdType: LVD タイプ	CY_LVD_TYPE_1(for LVD1) CY_LVD_TYPE_2(for LVD2)
void Cy_LVD_SetInterruptMask (cy_en_lvd_type_select_t lvdType)	LVD 割込みを有効にします。 lvdType: LVD タイプ	CY_LVD_TYPE_1(for LVD1) CY_LVD_TYPE_2(for LVD2)
void Cy_LVD_ClearInterrupt (cy_en_lvd_type_select_t lvdType)	LVD 割込みをクリアします。 lvdType: LVD タイプ	CY_LVD_TYPE_1(for LVD1) CY_LVD_TYPE_2(for LVD2)
Cy_Cpu_SramWriteBufferStatus (cy_en_cpu_sram_macro_t sramType)	書き込みバッファの状態を確認。 sramType: SRAM タイプ	CY_CPU_SRAM0
Cy_Cpu_SramPowerModeSet (cy_en_cpu_sram_macro_t sramType, cy_en_cpu_sram_power_mode_t pwrMode, uint8_t sramMacro)	SRAM 電力モードを設定します。 sramType: SRAM タイプ pwrMode: SRAM 電力モード sramMacro: 電力制御レジスタ インデックス	CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul

Code Listing 1 に、LVD 割込みを使用したリセットの設定部のプログラム例を示します。

このアプリケーション ノートでは、LED 操作に関連する関数は記載していません。

以下の説明は、SDL のドライバ部のレジスタ表記を理解するのに役立ちます。

- SRSS-> unPWR_LVD_CTL レジスタは、register TRM に記載されている PWR_LVD_CTL レジスタです。その他のレジスタについても、同様の意味です。

レジスタ表記の結合と構造の詳細については、hdr/rev_x/ip の cyip_srss_v2.h を参照してください。

Code Listing 1 LVD 割込みを使用したリセットのプログラム例

```
/* Define for LED0 */
/* Define the port settings */
#define USER_LED0_PORT           CY_LED0_PORT
#define USER_LED0_PIN            CY_LED0_PIN
#define USER_LED0_PIN_MUX        CY_LED0_PIN_MUX

/* Define for LED1 */
/* Define the port settings */
#define USER_LED1_PORT           CY_LED1_PORT
#define USER_LED1_PIN            CY_LED1_PIN
#define USER_LED1_PIN_MUX        CY_LED1_PIN_MUX


/* Set to gpio for LED0 */
/* Configure the port setting parameters for LED0. See 表 3. */
cy_stc_gpio_pin_config_t user_led0_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = USER_LED0_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};
/* Set to gpio for LED1 */
/* Configure the port setting parameters for LED1. See 表 3.*/
cy_stc_gpio_pin_config_t user_led1_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = USER_LED1_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};


/* Set to LVD interrupt */
/* Configure the interrupt structure parameters */
const  cy_stc_sysint_irq_t irq_cfg=
{
  .sysIntSrc  = srss_interrupt_IRQn,
  .intIdx     = CPUIntIdx2_IRQn,
  .isEnabled  = true,
};
/* VDDD checking flag */
bool VDDD_rising_check = false;
```

```
void LVD_IntHandler(void)    //Interrupt routine for LVD1 and LVD2
{
  uint32_t intStatus_4_7v;
  uint32_t intStatus_4_9v;
  uint32_t LVD_Status_4_7v_th;
  uint32_t LVD_Status_4_9v_th;

  /* LVD1 interrupt cause */
  intStatus_4_7v = Cy_LVD_GetInterruptStatusMasked(CY_LVD_TYPE_1);
  /* LVD2 interrupt cause */
  intStatus_4_9v = Cy_LVD_GetInterruptStatusMasked(CY_LVD_TYPE_2);

  /* If LVD_Status_4_7v_th is "1", it is above the threshold. If it is "0", it is below the
threshold. The LVD_Status_4_9v_th is also same. */
  LVD_Status_4_7v_th = Cy_LVD_GetStatus(CY_LVD_TYPE_1);
  LVD_Status_4_9v_th = Cy_LVD_GetStatus(CY_LVD_TYPE_2);

  /* Clear to interrupt LVD1 */
  Cy_LVD_ClearInterrupt(CY_LVD_TYPE_1);
  /* Clear to interrupt LVD2 */
  Cy_LVD_ClearInterrupt(CY_LVD_TYPE_2);

  /* LVD1 interrupt case */
  /* (2) LVD1 detection.*/
  if(((intStatus_4_7v >> 1) == 1ul) && (LVD_Status_4_7v_th == 0ul))
  {
    /* RAM status check */
    /* (3) Check the write buffer status. See Code Listing 11.*/
    while(0ul==(Cy_Cpu_SramWriteBufferStatus(CY_CPU_SRAM0)));

    /* RAM retain mode setting */
    /* (4) Sets the RETAINED mode. See Code Listing 12.*/
    Cy_Cpu_SramPowerModeSet(CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul);

    /* Software reset */
    NVIC_SystemReset();     //(5) Set software reset.
  }



  /* LVD2 interrupt case */
  else if(((intStatus_4_9v >> 2) == 1ul) && (LVD_Status_4_9v_th ==  1ul) &&  ((intStatus_4_7v
>> 1) == 0ul))
  {
    /* VDDD was returned to rising point */
    VDDD_rising_check = true;
  }
  else
  {
  }
```

```
}


int main(void)
{
  uint32_t tRstReason = 0ul;

  __enable_irq();

  SystemInit();

  /* Read the RESET reason */
  /* (1) Set for LVD1, LVD2 interrupt.*/
  /*Read for reset cause in restart. See Code Listing 2.*/
  tRstReason = Cy_SysReset_GetResetReason();    //

  /* (1) Set for LVD1, LVD2 interrupt.*/
  /* Clear to interrupt mask for LVD1  See Code Listing 3*/
  Cy_LVD_ClearInterruptMask(CY_LVD_TYPE_1);


  /* LVD1,LVD2 settings */
  /* (1) Set for LVD1, LVD2 interrupt.*/
  /* Set the LVD1 threshold  See Code Listing 4.*/
  Cy_LVD_SetThreshold(CY_LVD_TYPE_1, CY_LVD_THRESHOLD_4_7_V);



  /* Action Select for LVD1 */
  /* (1) Set for LVD1, LVD2 interrupt.*/
  /* Set the action for LVD1. See Code Listing 5.*/
  Cy_LVD_SetActionSelect(CY_LVD_TYPE_1, CY_LVD_ACTION_INTR);



  /* (1) Set for LVD1, LVD2 interrupt.*/
  /* Edge Select for LVD1  See Code Listing 6.*/
  Cy_LVD_SetEdgeSelect(CY_LVD_TYPE_1, CY_LVD_EDGE_FALLING);



  /* (1) Set for LVD1, LVD2 interrupt.*/
  /* Enable LVD1  See Code Listing 7*/
  Cy_LVD_Enable(CY_LVD_TYPE_1);



  /* Set interupt for LVD1 */
  /* (1) Set for LVD1, LVD2 interrupt.*/
  Cy_LVD_SetInterrupt(CY_LVD_TYPE_1);     //Enable for LVD1 interrupt. See Code Listing 8.
  Cy_LVD_SetInterruptMask(CY_LVD_TYPE_1);    //Set to interrupt mask for LVD1. See Code Listing
9.
```

```
/* (1) Set for LVD1, LVD2 interrupt.*/
/* Clear to interrupt LVD1  See Code Listing 10*/
Cy_LVD_ClearInterrupt(CY_LVD_TYPE_1);



/* Clear to interrupt mask for LVD2 */
/* (1) Set for LVD1, LVD2 interrupt.*/
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_ClearInterruptMask(CY_LVD_TYPE_2);

/* Set the LVD2 threshold */
/* (1) Set for LVD1, LVD2 interrupt.*/
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_SetThreshold(CY_LVD_TYPE_2, CY_LVD_THRESHOLD_4_9_V);

/* Action Select for LVD2 */
/* (1) Set for LVD1, LVD2 interrupt.*/
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_SetActionSelect(CY_LVD_TYPE_2, CY_LVD_ACTION_INTR);

/* Edge Select for LVD2 */
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_SetEdgeSelect(CY_LVD_TYPE_2, CY_LVD_EDGE_RISING);

/* Enable LVD2 */
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_Enable(CY_LVD_TYPE_2);

/* Set interupt for LVD1 */
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_SetInterrupt(CY_LVD_TYPE_2);
Cy_LVD_SetInterruptMask(CY_LVD_TYPE_2);

/* Clear to interrupt LVD2 */
/* For LVD2 settings, refer to the LVD1.*/
Cy_LVD_ClearInterrupt(CY_LVD_TYPE_2);

/* Initialize in GPIO for LED */
/* Set for LED0, LED1.*/
Cy_GPIO_Pin_Init(USER_LED0_PORT, USER_LED0_PIN, &user_led0_port_pin_cfg);
Cy_GPIO_Pin_Init(USER_LED1_PORT, USER_LED1_PIN, &user_led1_port_pin_cfg);



/* Interrupt settings */
/* Set for interrupt routine.*/
Cy_SysInt_InitIRQ(&irq_cfg);
Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, LVD_IntHandler);
NVIC_SetPriority(CPUIntIdx2_IRQn, 0ul);
NVIC_ClearPendingIRQ(CPUIntIdx2_IRQn);
NVIC_EnableIRQ(CPUIntIdx2_IRQn);
```

```
  for(;;)
  {
    /* VDDD return checking point */
    /* (6) Check for VDDD risig in restart by LVD2 detection.*/
    if(VDDD_rising_check == true){

      /* After the LVD2 interrupt routine occurred,                 */
      /* when this program entered the code here, VDDD was returned.*/
      VDDD_rising_check = false;

      /* LED0 blink */
      for(uint16_t i = 0ul; i < 50ul; i++)
      {
        Cy_SysTick_DelayInUs(50000ul);
        Cy_GPIO_Inv(USER_LED0_PORT, USER_LED0_PIN);    //If VDDD was retuned in restart, the
LED0 blink.
      }
    }

    /* Check for BOD reset generation */
    if( ( tRstReason & CY_SYSRESET_BODVDDD ) == CY_SYSRESET_BODVDDD )    //(7) Check BOD reset
cause in restart.
    {
      /* If the BOD occurs, RAM data was not retained.*/
      /* LED1 turn on */
      Cy_GPIO_Write(USER_LED1_PORT, USER_LED1_PIN, 1);    //If BOD reset was generated, the
LED1 turn on.
    }
  }
}
```

Code Listing 2～Code Listing 12 に、ドライバ部で LVD 割込みを使用したリセットを設定するプログラムの例を示します。

Code Listing 2 ドライバ部の SysReset_GetResetReason のプログラム例

```
uint32_t Cy_SysReset_GetResetReason(void)
{
  return(SRSS->unRES_CAUSE.u32Register);    //(7) Check BOD reset cause.
}
```

Code Listing 3 ドライバ部の割込みマスクをクリアするプログラム例

```
void Cy_LVD_ClearInterruptMask(cy_en_lvd_type_select_t lvdType)
{
  /* Clear the interrupt mask.*/
  if(lvdType == CY_LVD_TYPE_1)
  {
    SRSS->unSRSS_INTR_MASK.u32Register &= (uint32_t) ~SRSS_SRSS_INTR_MASK_HVLVD1_Msk;
  }
  else
  {
  SRSS->unSRSS_INTR_MASK.u32Register &= (uint32_t) ~SRSS_SRSS_INTR_MASK_HVLVD2_Msk;
  }
}
```

Code Listing 4 ドライバ部の LVD SetThreshold のプログラム例

```
void Cy_LVD_SetThreshold(cy_en_lvd_type_select_t lvdType, cy_en_lvd_tripsel_t threshold)
{
  CY_ASSERT(CY_LVD_CHECK_TRIPSEL(threshold));

  if(lvdType == CY_LVD_TYPE_1)
  {
    /* Set the threshold.*/
    SRSS->unPWR_LVD_CTL.u32Register =  _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL.u32Register,
SRSS_PWR_LVD_CTL_HVLVD1_TRIPSEL_HT, threshold);
  }
  else
  {
    SRSS->unPWR_LVD_CTL2.u32Register = _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL2.u32Register,
SRSS_PWR_LVD_CTL2_HVLVD2_TRIPSEL_HT, threshold);
  }
}
```

Code Listing 5 ドライバ部の LVD SetActionSelect のプログラム例

```
void Cy_LVD_SetActionSelect(cy_en_lvd_type_select_t lvdType, cy_en_lvd_action_select_t
lvdActionSelect)
{
  CY_ASSERT(CY_LVD_CHECK_ACTION_CFG(lvdActionSelect));

  if(lvdType == CY_LVD_TYPE_1)
  {
    /*Set the action.*/
    SRSS->unPWR_LVD_CTL.u32Register = _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL.u32Register,
SRSS_PWR_LVD_CTL_HVLVD1_ACTION, lvdActionSelect);
  }
  else
  {
    SRSS->unPWR_LVD_CTL2.u32Register = _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL2.u32Register,
SRSS_PWR_LVD_CTL2_HVLVD2_ACTION, lvdActionSelect);
  }
}
```

Code Listing 6 ドライバ部の LVD SetEdgeSelect のプログラム例

```
void Cy_LVD_SetEdgeSelect(cy_en_lvd_type_select_t lvdType, cy_en_lvd_edge_select_t
lvdEdgeSelect)
{
  CY_ASSERT(CY_LVD_CHECK_EDGE_CFG(lvdEdgeSelect));

  if(lvdType == CY_LVD_TYPE_1)
  {
    /* Select the detection edge.*/
    SRSS->unPWR_LVD_CTL.u32Register = _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL.u32Register,
SRSS_PWR_LVD_CTL_HVLVD1_EDGE_SEL, lvdEdgeSelect);
  }
  else
  {
    SRSS->unPWR_LVD_CTL2.u32Register = _CLR_SET_FLD32U(SRSS->unPWR_LVD_CTL2.u32Register,
SRSS_PWR_LVD_CTL2_HVLVD2_EDGE_SEL, lvdEdgeSelect);
  }
}
```

Code Listing 7 ドライバ部の LVD Enable のプログラム例

```
void Cy_LVD_Enable(cy_en_lvd_type_select_t lvdType)
{
  if(lvdType == CY_LVD_TYPE_1)
  {
    /* Enable for LVD1.*/
    SRSS->unPWR_LVD_CTL.u32Register |= SRSS_PWR_LVD_CTL_HVLVD1_EN_HT_Msk;
  }
  else
  {
    SRSS->unPWR_LVD_CTL2.u32Register |= SRSS_PWR_LVD_CTL2_HVLVD2_EN_HT_Msk;
  }
}
```

Code Listing 8 ドライバ部の LVD SetInterrupt のプログラム例

```
void Cy_LVD_SetInterrupt(cy_en_lvd_type_select_t lvdType)
{
  if(lvdType == CY_LVD_TYPE_1)
  {
    /*Enable for LVD1 interrupt.*/
    SRSS->unSRSS_INTR_SET.u32Register = SRSS_SRSS_INTR_SET_HVLVD1_Msk;
  }
  else
  {
    SRSS->unSRSS_INTR_SET.u32Register = SRSS_SRSS_INTR_SET_HVLVD2_Msk;
  }
}
```

Code Listing 9 ドライバ部の LVD SetInterruptMask のプログラム例

```
void Cy_LVD_SetInterruptMask(cy_en_lvd_type_select_t lvdType)
{
  if(lvdType == CY_LVD_TYPE_1)
  {
    /*Set the interrupt mask for LVD1.*/
    SRSS->unSRSS_INTR_MASK.u32Register |= SRSS_SRSS_INTR_MASK_HVLVD1_Msk;
  }
  else
  {
    SRSS->unSRSS_INTR_MASK.u32Register |= SRSS_SRSS_INTR_MASK_HVLVD2_Msk;
  }
}
```

Code Listing 10 ドライバ部の LVD ClearInterrupt のプログラム例

```
void Cy_LVD_ClearInterrupt(cy_en_lvd_type_select_t lvdType)
{
if(lvdType == CY_LVD_TYPE_1)
  {
    /*Clear the interrupt.*/
    SRSS->unSRSS_INTR.u32Register = SRSS_SRSS_INTR_HVLVD1_Msk;
  }
  else
  {
    SRSS->unSRSS_INTR.u32Register = SRSS_SRSS_INTR_HVLVD2_Msk;
  }

  (void) SRSS->unSRSS_INTR.u32Register;
}
```

Code Listing 11 ドライバ部の CPU SramWriteBufferStatus のプログラム例

```
bool Cy_Cpu_SramWriteBufferStatus(cy_en_cpu_sram_macro_t sramType)
{
  bool wb_status = false;

  switch(sramType)
  {
    case CY_CPU_SRAM0:
    /*(3) Check for buffer status*/
    wb_status = CPUSS->unRAM0_STATUS.stcField.u1WB_EMPTY;
    break;
    case CY_CPU_SRAM1:
    wb_status = CPUSS->unRAM1_STATUS.stcField.u1WB_EMPTY;
    break;
    case CY_CPU_SRAM2:
    wb_status = CPUSS->unRAM2_STATUS.stcField.u1WB_EMPTY;
    break;
    default:
    break;
  }

  return wb_status;
}
```

Code Listing 12 ドライバ部の CPU SramPowerModeSet のプログラム例

```
void Cy_Cpu_SramPowerModeSet(cy_en_cpu_sram_macro_t sramType, cy_en_cpu_sram_power_mode_t
pwrMode, uint8_t sramMacro)
{
  switch(sramType)
  {
    case CY_CPU_SRAM0:
    /*(4) Sets the RETAINED mode.*/
    CPUSS->unRAM0_PWR_MACRO_CTL[sramMacro].stcField.u2PWR_MODE = pwrMode;
    break;
    case CY_CPU_SRAM1:
    CPUSS->unRAM1_PWR_CTL.stcField.u2PWR_MODE = pwrMode;
    break;
    case CY_CPU_SRAM2:
    CPUSS->unRAM2_PWR_CTL.stcField.u2PWR_MODE = pwrMode;
    break;
    default:
    break;
  }
}
```

3.2 外部リセットを使用したリセット

このケースでは、低電圧は、外部 LVD IC によって検出されます。図 7 に、外部 LVD IC から GPIO ピンへの入力信号を使用した RAM0 保持のリセット手順のブロックダイヤグラムを示します。

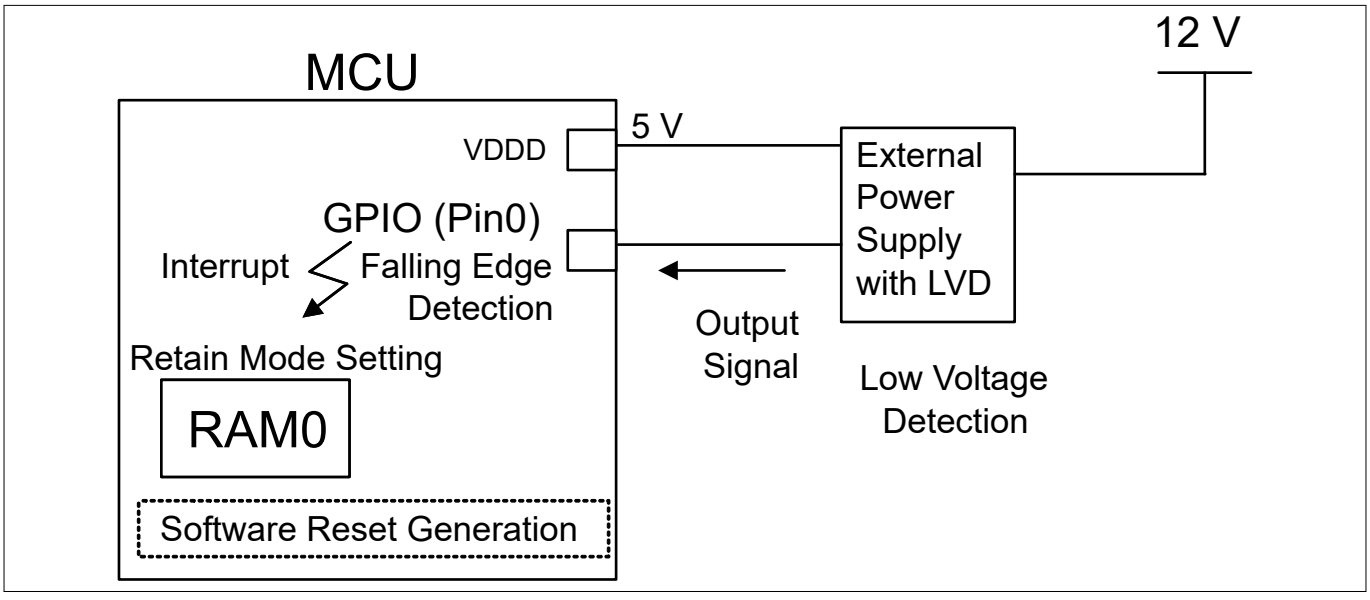

図 7 GPIO への外部 LVD IC 入力信号による RAM0 保持リセット手順のブロックダイヤグラム

図 7 では、この IC の出力は GPIO Pin0 に接続されています。この Pin0 は、割込みピンに設定され、LVD を使用するケースと同様に割込みを発生できます。CYT3 と CYT4 の場合、電源電圧は 5 V と 1.15 V であり、外部 LVD IC は電源電圧の 5 V と 1.15 V の両方のモニタリングに使われます。

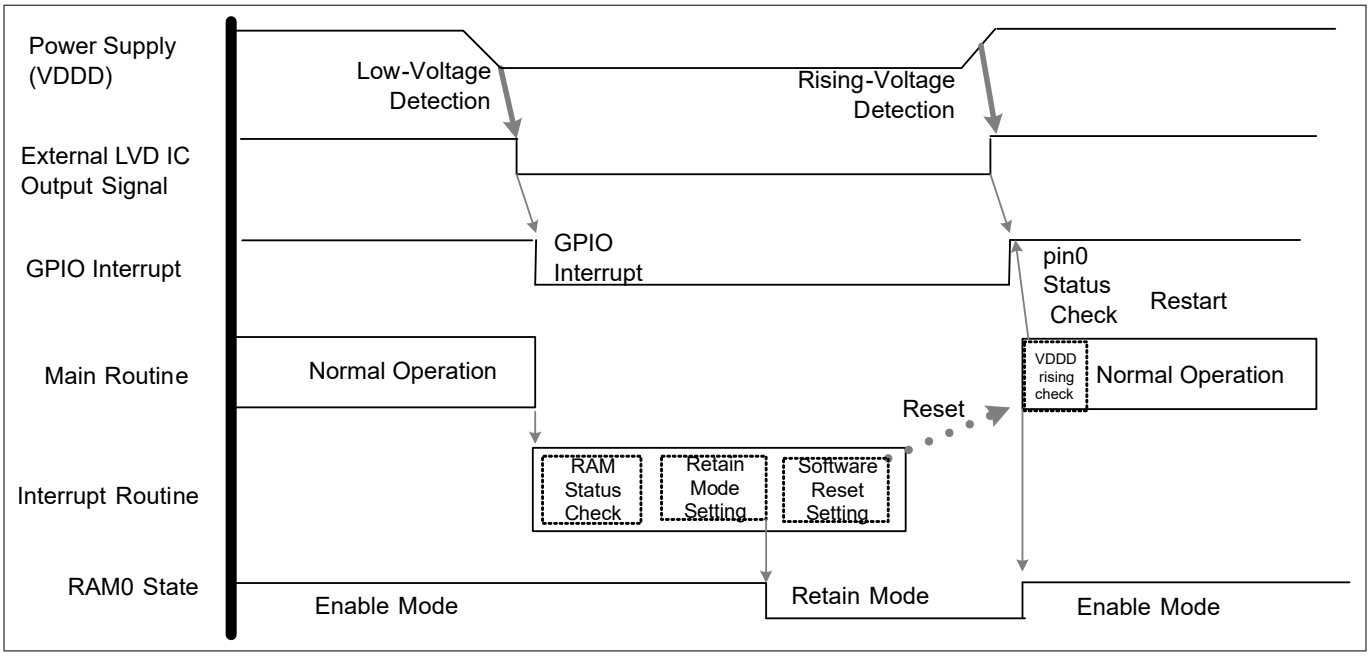

図 8　　外部 LVD IC での RAM0 リセット手順タイミングチャート例

図 8 では、VDDD 上の外部 LVD IC で低電圧を検知した後、外部 LVC IC 出力は LOW になります。GPIO Pin0 は、立ち下りエッジを検知し、割込みを発生します。割込み発生時、MCU はメインルーチンから割込みルーチンに移行します。割込みルーチンは、RAM0 ステータス, RETAINED モード設定, およびソフトウェアリセット設定を行ってリセットを発生します。MCU がリセットから復帰後、GPIO Pin0 が LVD IC 出力と接続されているため、GPIO Pin0 ステータスで VDDD の立ち上りを確認し、BOD リセットが発生していないことを確認します。

3.2.1　　使用事例

ここでは、サンプルドライバライブラリ (SDL) を使用した使用事例に基づいて、外部リセットを使用してリセットを設定する方法について説明します。このアプリケーションノートに記載されているプログラムコードは、SDL に含まれるものです。SDL については、その他の参考資料を参照してください。

使用事例:

- 外部入力信号の設定: ボタン押し
- 外部入力ポートの設定: P6_5
- GPIO 割込みの設定: 入力信号の立ち上り
- RAM モードの設定: 保持モード
- リセットの設定: ソフトウェア リセット

図 9 に、GPIO への外部 LVD IC 入力信号を使用した RAM0 保持でのリセット手順のフロー例を示します。

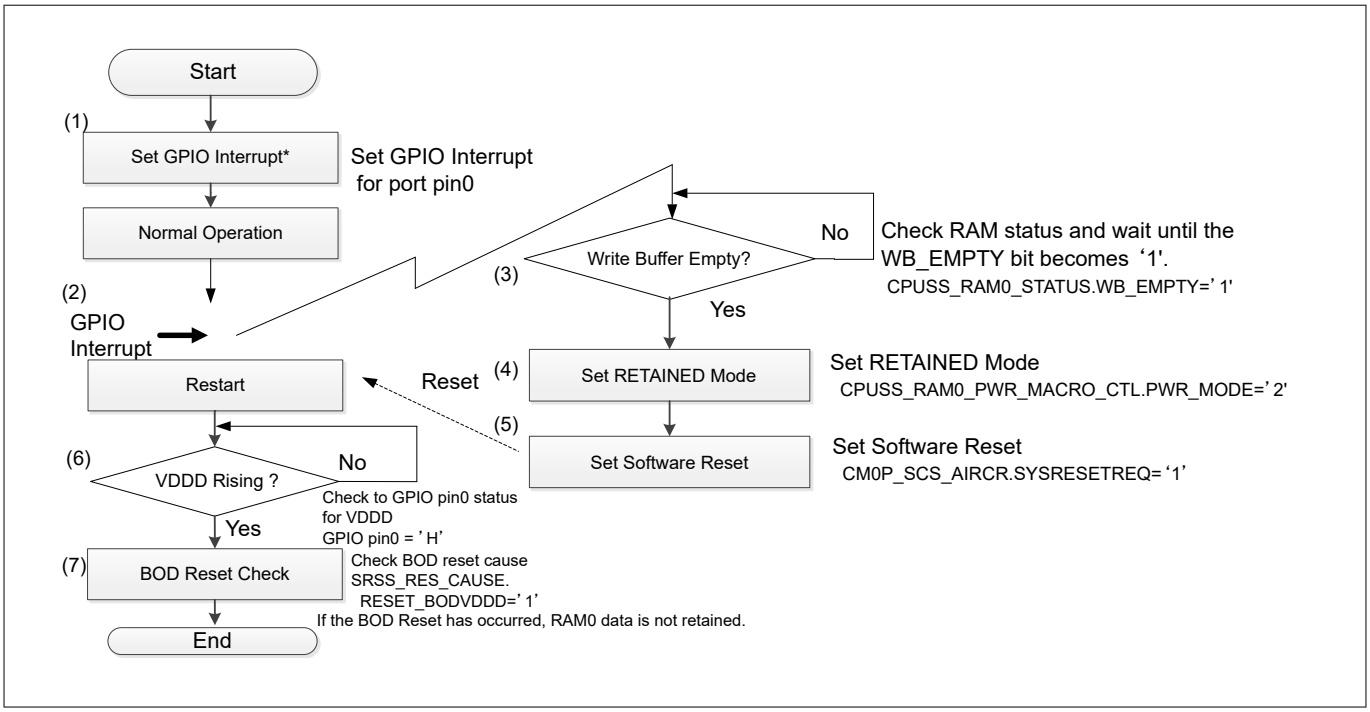

図 9　　GPIO への外部 LVD IC 入力信号での RAM0 保持リセット手順例

* 割込み設定の詳細については、architecture TRM の"Interrupts"章を参照してください。

1. GPIO Pin0 の割込みを有効にしてください。
2. GPIO Pin0 が外部 LVD IC からの入力信号を検知すると、GPIO は割込みを CPU に対して発生します。
3. CPU が GPIO 割込み検出したら、書き込みバッファのステータスを確認してください。
   書き込みバッファにデータが存在する場合 (WB_EMPTY = '0')、書き込みバッファのデータが無くなるまで待機してください。
4. 書き込みバッファにデータが無い場合 (WB_EMPTY = '1')、CPU は RETAINED モードを設定します。
5. CPU はソフトウェアリセットを発生します。
6. CPU は VDDD 電源電圧の復旧を確認します。
7. VDDD が回復すると、CPU は BOD リセットが発生していないことを確認します。BOD リセットが発生していない場合、RAM0 のデータは保持されます。しかし、BOD リセットが発生した場合、RAM0 データを破棄する必要があります。

外部リセット設定を使用したリセットにおける SDL の設定部のパラメーターを表 5 に、関数を表 6 に示します。

表 5　　外部リセットを使用した初期リセットパラメーター一覧

パラメーター	説明	値
user_button_port_pin_cfg. outVal	ピン出力状態	0ul
user_button_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_HIGHZ
user_button_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_BUTTON_PIN_MUX
user_button_port_pin_cfg. intEdge	IRQ をトリガするエッジ	CY_GPIO_INTR_FALLING
user_button_port_pin_cfg. intMask	マスク エッジ割込み	1ul
user_button_port_pin_cfg. vtrip	入力バッファ モード	0ul,
user_button_port_pin_cfg. slewRate	スルーレート	0ul

(続く)

表 5 (続き) 外部リセットを使用した初期リセットパラメーター一覧

パラメーター	説明	値
user_button_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led0_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led0_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED0_PIN_MUX
user_led0_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led0_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led0_port_pin_cfg. vtrip	入力バッファ モード	0ul,
user_led0_port_pin_cfg. slewRate	スルーレート	0ul
user_led0_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led1_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led1_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED1_PIN_MUX
user_led1_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led1_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led1_port_pin_cfg)Vtrip	入力バッファ モード	0ul
user_led1_port_pin_cfg. slewRate	スルーレート	0ul
user_led1_port_pin_cfg. driveSel	GPIO 駆動強度	0ul

表 6 外部リセット設定を使用したリセット関数一覧

関数	説明	値
Cy_GPIO_Pin_Init (volatile stc_GPIO_PRT_t *base, uint32_t pinNum, const cy_stc_gpio_pin_config_t *config)	ピンのすべてのピン設定を初期化。 *base: ピンのポート レジスタ ベース アドレスへのポインタ pinNum: ポート レジスタ内のピン ビット フィールドの位置 *config: ピン コンフィギュレーション構造のベース アドレスへのポインタ	USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg
Cy_SysReset_GetResetReason(void)	最新のリセット要因を返します	-
Cy_Cpu_SramWriteBufferStatus (cy_en_cpu_sram_macro_t sramType)	書き込みバッファの状態を確認 sramType: SRAM タイプ	CY_CPU_SRAM0
Cy_Cpu_SramPowerModeSet (cy_en_cpu_sram_macro_t sramType, cy_en_cpu_sram_power_mode_t pwrMode, uint8_t sramMacro)	SRAM 電力モードを設定 sramType: SRAM タイプ pwrMode: SRAM 電力モード sramMacro: 電力制御レジスタ インデックス	CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul

Code Listing 13 に、外部リセットを使用したリセット用の設定部のプログラム例を示します。

Code Listing 13 外部リセットを使用したリセットのプログラム例

```
/* Define GPIO button (P6_5) */
/* This button is used as an input signal assumed external LVD IC. */
/* Define the port settings.*/
#define USER_BUTTON_PORT        CY_CB_BUTTON_PORT
#define USER_BUTTON_PIN         CY_CB_BUTTON_PIN
#define USER_BUTTON_PIN_MUX     CY_CB_BUTTON_PIN_MUX
#define USER_BUTTON_IRQ         CY_CB_BUTTON_IRQN

/* Define for LED0 */
/* Define the port settings.*/
#define USER_LED0_PORT          CY_LED0_PORT
#define USER_LED0_PIN           CY_LED0_PIN
#define USER_LED0_PIN_MUX       CY_LED0_PIN_MUX

/* Define for LED1 */
/* Define the port settings.*/
#define USER_LED1_PORT          CY_LED1_PORT
#define USER_LED1_PIN           CY_LED1_PIN
#define USER_LED1_PIN_MUX       CY_LED1_PIN_MUX

/* Set to button for input signal  */
/* Configure the port setting parameters for button. See 表 5.*/
const cy_stc_gpio_pin_config_t user_button_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_HIGHZ,
  .hsiom     = USER_BUTTON_PIN_MUX,
  .intEdge   = CY_GPIO_INTR_FALLING,
  .intMask   = 1ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};




/* LED0 gpio configuration */
/* Configure the port setting parameters for LED0. See 表 5.*/
cy_stc_gpio_pin_config_t user_led0_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = CY_LED0_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};
```

```
/* LED1 gpio configuration */
/*Configure the port setting parameters for LED1. See 表 5.*/
cy_stc_gpio_pin_config_t user_led1_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = CY_LED1_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};

/* Set to GPIO Button interrupt for falling edge */
/* Configure the interrupt structure parameters.*/
const cy_stc_sysint_irq_t irq_cfg =
{
  .sysIntSrc  = USER_BUTTON_IRQ,
  .intIdx     = CPUIntIdx3_IRQn,
  .isEnabled  = true,
};

void GPIO_IntHandler(void)    //(2) GPIO Interrupt routine.
{
  uint32_t intStatus;

  /* If falling edge detected */
  intStatus = Cy_GPIO_GetInterruptStatusMasked(USER_BUTTON_PORT, USER_BUTTON_PIN);
  if (intStatus != 0ul)
  {
    /* RAM staus check */
    /*(3) Check the write buffer status. See Code Listing 11.*/
    while(0ul==(Cy_Cpu_SramWriteBufferStatus(CY_CPU_SRAM0)));



    /* RAM retain mode setting */
    /*(4) Set the RETAINED mode. See Code Listing 12.*/
    Cy_Cpu_SramPowerModeSet(CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul);

    /* Software reset */
    /* (5) Set software reset.*/
    NVIC_SystemReset();

    /* Clear to interrupt */
    Cy_GPIO_ClearInterrupt(USER_BUTTON_PORT, USER_BUTTON_PIN);
  }
}



int main(void)
```

```
{
  uint32_t tRstReason = 0ul;

  __enable_irq(); /* Enable global interrupts. */

  SystemInit();

  /* Port initialization */
  /*Set to GPIO for button settings in assuming signal input from an external LVD IC. See Code
Listing 14.*/
  Cy_GPIO_Pin_Init(USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg);

  /* LED port initialization */
  /* Set for LED0, LED1.*/
  Cy_GPIO_Pin_Init(USER_LED0_PORT, USER_LED0_PIN, &user_led0_port_pin_cfg);
  Cy_GPIO_Pin_Init(USER_LED1_PORT, USER_LED1_PIN, &user_led1_port_pin_cfg);


  /* Read the RESET reason */
  /* Check to reset cause in restart. See Code Listing 2.*/
  tRstReason = Cy_SysReset_GetResetReason();



  /* Check for software reset generation by LVD */
  if( ( tRstReason & CY_SYSRESET_SOFT ) == CY_SYSRESET_SOFT )
  {
    /* VDDD rising check */
    /* (6) Check to GPIO button pin for VDDD rising in restart.*/
    while(0ul==(Cy_GPIO_Read(USER_BUTTON_PORT, USER_BUTTON_PIN)));



    /* LED1 turn on */
    /* In the VDDD return, LED1 turn on .*/
    Cy_GPIO_Write(USER_LED1_PORT, USER_LED1_PIN, 1ul);
  }

  /* Check for BOD reset generation */
  /* (7) Check BOD reset cause in restart.*/
  if( ( tRstReason & CY_SYSRESET_BODVDDD ) == CY_SYSRESET_BODVDDD )
  {
    /* When the here code entered, RAM data was not retained.*/
  }

  /* GPIO interrupt settings */
  /* (1) Set GPIO interrupt.*/
  Cy_SysInt_InitIRQ(&irq_cfg);
  Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, GPIO_IntHandler);
  NVIC_SetPriority(irq_cfg.intIdx, 3ul);
  NVIC_EnableIRQ(irq_cfg.intIdx);

  for(;;)
```

```
    {
      /* Waiting 0.5 [s] in the LED0 */
      /* The LED0 blink in normal operation.*/
      Cy_SysTick_DelayInUs(500000ul);
      Cy_GPIO_Inv(USER_LED0_PORT, USER_LED0_PIN);
    }
  }
```

Code Listing 14 に、ドライバ部で外部リセットを使用したリセットを設定するプログラム例を示します。

Code Listing 14 ドライバ部で GPIO ピン初期化のプログラム例

```
cy_en_gpio_status_t Cy_GPIO_Pin_Init(volatile stc_GPIO_PRT_t *base, uint32_t pinNum, const
cy_stc_gpio_pin_config_t *config)
{
/* Set for GPIO pin.*/
  cy_en_gpio_status_t status = CY_GPIO_SUCCESS;

  if((NULL != base) && (NULL != config))
  {
    Cy_GPIO_Write(base, pinNum, config->outVal);
    Cy_GPIO_SetHSIOM(base, pinNum, config->hsiom);
    Cy_GPIO_SetVtrip(base, pinNum, config->vtrip);
    Cy_GPIO_SetSlewRate(base, pinNum, config->slewRate);
    Cy_GPIO_SetDriveSel(base, pinNum, config->driveSel);
    Cy_GPIO_SetDrivemode(base, pinNum, config->driveMode);
    Cy_GPIO_SetInterruptEdge(base, pinNum, config->intEdge);
    Cy_GPIO_ClearInterrupt(base, pinNum);
    Cy_GPIO_SetInterruptMask(base, pinNum, config->intMask);
  }
  else
  {
    status = CY_GPIO_BAD_PARAM;
  }

  return(status);
}
```

4 ローパワーモードにおける RAM 保持手順

ここでは、ローパワーモード移行手順例をブロック ダイヤグラム, タイミング チャート, およびフロー チャートで示します。

MCU には、デバイスパワーモードがあります。デバイスパワーモードには、Active モード, Sleep モード, DeepSleep モード, および Hibernate モードがあります。Sleep モード, DeepSleep モード, および Hibernate モードは、ローパワーモードです。このアプリケーションノートでは、Deepsleep モードの場合を説明します。

詳細については、Architecture TRM の"Device Power Modes"章を参照してください。

4.1 DeepSleep モードの使用

このケースでは、DeepSleep モード移行が使用されます。DeepSleep モードの設定は、メインプログラムが実行している Active モードで行われます。

図 10 に、RAM0 保持での DeepSleep モード移行手順のブロックダイヤグラムを示します。

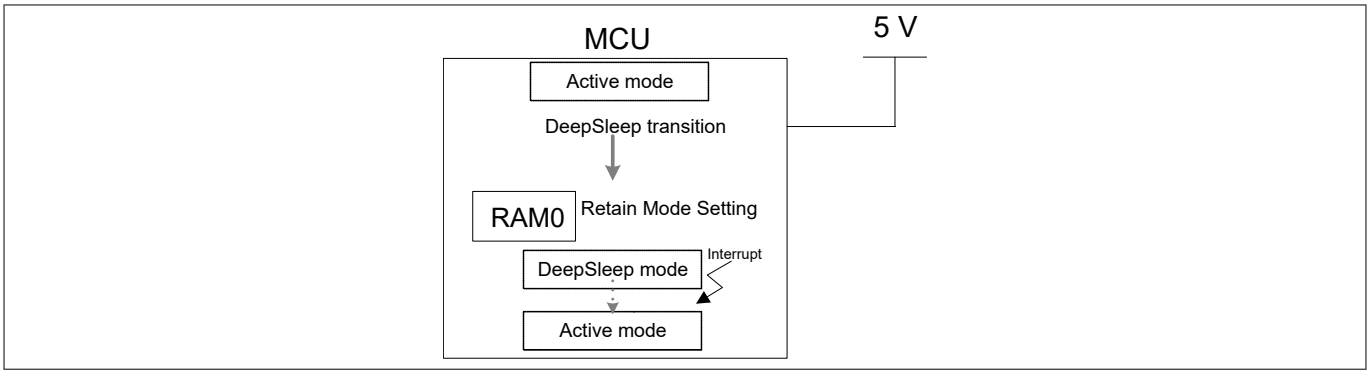

図 10 RAM0 保持での DeepSleep モード 移行手順のブロックダイヤグラム

図 10 では、まず Active モードから DeepSleep モードへ移行します。この移行中に RAM0 ステータスを確認して、RETAINED モードを設定します。次に MCU は DeepSleep モードになります。DeepSleep モード中に割込みが発生すると MCU は Active モードに復帰します。

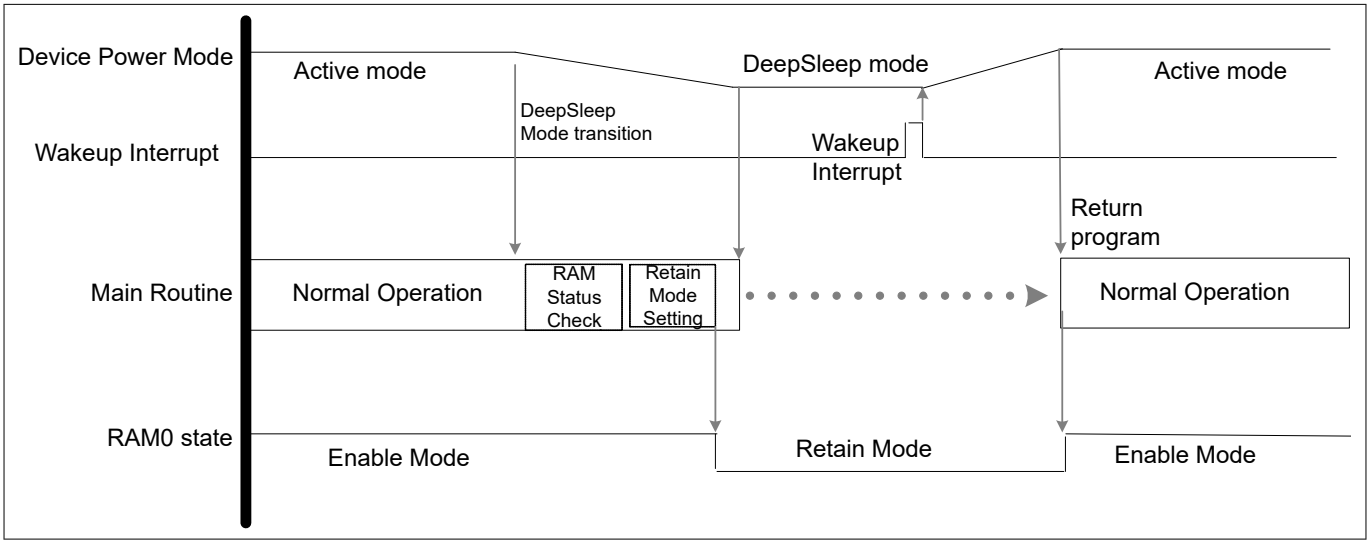

図 11 RAM0 保持での DeepSleep モード移行手順タイミングチャート例

図 11 において、Active モードから DeepSleep モードへ移行する際、メインルーチンは RAM0 ステータスを確認して、RETAINED モードに設定します。その後、MCU は DeepSleep モードへ移行し、メインルーチンは停止します。それから DeepSleep モードでは、割込みが発生した際、Active モードへ復帰します。メインルーチンはプログラムの実行を再開します。

4.1.1 使用事例

ここでは、サンプルドライバライブラリ (SDL) を使用した使用事例に基づいて、DeepSleep モードへの移行を設定する方法について説明します。このアプリケーションノートに記載されているプログラムコードは、SDL に含まれるものです。SDL については、その他の参考資料を参照してください。

使用事例:

- 低電力モード移行の設定: DeepSleep モード
- RAM モードの設定: 保持モード
- Active モードへ復帰の設定: ウェイクアップ入力信号割込み
- ウェイクアップ入力ポートの設定: P6_5

図 12 に、RAM0 保持での DeepSleep モード移行手順のフロー例を示します。

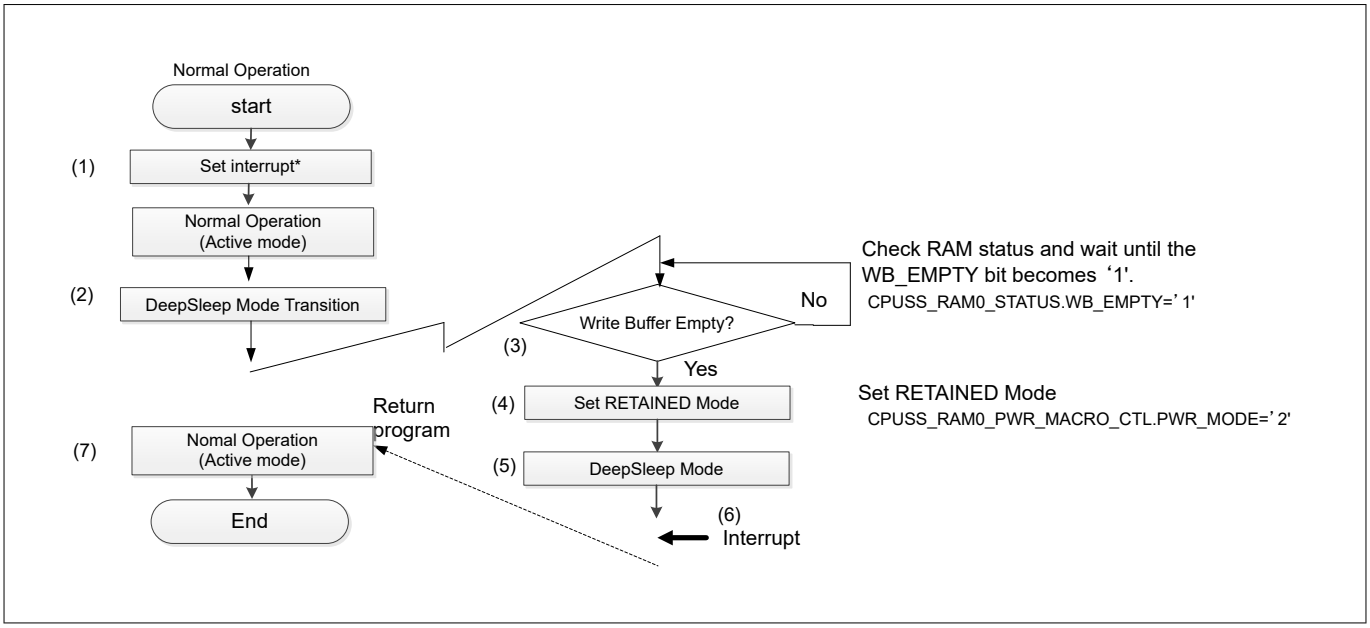

図 12 RAM0 保持での DeepSleep モード移行手順のフローチャート例

* 割込み設定の詳細については、Architecture TRM の“Interrupts”章を参照してください。

1. 割込みを設定してください。
2. 通常動作の Active モードにおいて、Sleep モードを設定してください。
3. 書き込みバッファステータスを確認してください。書き込みバッファにデータが存在する場合 (WB_EMPTY = '0')、書き込みバッファのデータが無くなるまで待機してください。
4. 書き込みバッファにデータが無い場合 (WB_EMPTY = '1')、CPU は RETAINED モードを設定します。
5. MCU は、DeepSleep モードに移行します。
6. 割込みが発生すると、MCU は DeepSleep モードから Active モードに変更します。
7. プログラムの実行が再開します。

DeepSleep モード移行設定の SDL の設定部のパラメーターを表 7 に、関数を表 8 に示します。

表 7 初期 DeepSleep モード移行パラメーター一覧

パラメーター	説明	値
user_button_port_pin_cfg. outVal	ピン出力状態	0ul
user_button_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_HIGHZ
user_button_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_BUTTON_PIN_MUX

(続く)

表 7 (続き) 初期 DeepSleep モード移行パラメーター一覧

パラメーター	説明	値
user_button_port_pin_cfg. intEdge	IRQ をトリガするエッジ	CY_GPIO_INTR_FALLING
user_button_port_pin_cfg. intMask	マスク エッジ割込み	1ul
user_button_port_pin_cfg. vtrip	入力バッファ モード	0ul,
user_button_port_pin_cfg. slewRate	スルーレート	0ul
user_button_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED_PIN_MUX
user_led_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led_port_pin_cfg. vtrip	入力バッファ モード	0ul
user_led_port_pin_cfg. slewRate	スルーレート	0ul
user_led_port_pin_cfg. driveSel	GPIO 駆動強度	0ul

表 8 初期 DeepSleep モード移行設定関数一覧

関数	説明	値
Cy_GPIO_Pin_Init (volatile stc_GPIO_PRT_t *base, uint32_t pinNum, const cy_stc_gpio_pin_config_t *config)	ピンのすべてのピン設定を初期化。 *base: ピンのポート レジスタ ベース アドレスへのポインタ pinNum: ポート レジスタ内のピン ビット フィールドの位置 *config: ピン コンフィギュレーション構造のベース アドレスへのポインタ	USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg
Cy_Cpu_SramWriteBufferStatus (cy_en_cpu_sram_macro_t sramType)	sramType: SRAM タイプ	CY_CPU_SRAM0
Cy_Cpu_SramPowerModeSet (cy_en_cpu_sram_macro_t sramType, cy_en_cpu_sram_power_mode_t pwrMode, uint8_t sramMacro)	SRAM 電力モードを設定 sramType: SRAM タイプ pwrMode: SRAM 電力モード sramMacro: 電力制御レジスタ インデックス	CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul
Cy_SysPm_DeepSleep (cy_en_syspm_waitfor_t waitFor)	CPU コアを DeepSleep モードに設定。 waitFor: アクションの待機を選択	CY_SYSPM_WAIT_FOR_INTERRUPT

Code Listing 15 に、DeepSleep モード移行用の設定部のプログラム例を示します。

Code Listing 15 DeepSleep モード移行のプログラム例

```
/* Define for Deepsleep wakeup gpio (P6_5) */
/* Define the port settings*/
#define USER_BUTTON_PORT        CY_CB_BUTTON_PORT
#define USER_BUTTON_PIN         CY_CB_BUTTON_PIN
#define USER_BUTTON_PIN_MUX     CY_CB_BUTTON_PIN_MUX
#define USER_BUTTON_IRQ         CY_CB_BUTTON_IRQN

/* Define for LED0 */
/* Define the port settings*/
#define USER_LED_PORT           CY_LED0_PORT
#define USER_LED_PIN            CY_LED0_PIN
#define USER_LED_PIN_MUX        CY_LED0_PIN_MUX

/* Set to wakeup pin for Deepsleep mode return  */
const cy_stc_gpio_pin_config_t user_button_port_pin_cfg =
{
/* Configure the port setting parameters for wakeup. See 表 7.*/
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_HIGHZ,
  .hsiom     = USER_BUTTON_PIN_MUX,
  .intEdge   = CY_GPIO_INTR_FALLING,
  .intMask   = 1ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};

/* Set to gpio for LED0 */
/* Configure the port setting parameters for LED0. See 表 7.*/
cy_stc_gpio_pin_config_t user_led_port_pin_cfg =
{
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = USER_LED_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};

/* Set to GPIO Button interrupt for Deepsleep wakeup */
/* Configure the interrupt structure parameters.*/
const cy_stc_sysint_irq_t irq_cfg =
{
  .sysIntSrc  = USER_BUTTON_IRQ,
  .intIdx     = CPUIntIdx3_IRQn,
  .isEnabled  = true,
};

/* Deepsleep mode start flag */
```

```
uint8_t Deepsleep_transition = false;

/* Deepsleep mode transition checking */
cy_en_syspm_status_t return_value;


void ButtonIntHandler(void)    //(6) Interrupt routine
{
  uint32_t intStatus;

  /* If falling edge detected */
  intStatus = Cy_GPIO_GetInterruptStatusMasked(USER_BUTTON_PORT, USER_BUTTON_PIN);
  if (intStatus != 0ul)
  {
    /* Clear to interrupt flag */
    Cy_GPIO_ClearInterrupt(USER_BUTTON_PORT, USER_BUTTON_PIN);
  }
}


int main(void)
{
  __enable_irq(); /* Enable global interrupts. */

  SystemInit();

  /* Initialize in GPIO for Deeplsleep wakeup */
  /* Set GPIO pin for DeepSleep wakeup. See Code Listing 14.*/
  Cy_GPIO_Pin_Init(USER_BUTTON_PORT, USER_BUTTON_PIN, &user_button_port_pin_cfg);

  /* Initialize in GPIO for LED0 */
  /* Set for LED0.*/
  Cy_GPIO_Pin_Init(USER_LED_PORT, USER_LED_PIN, &user_led_port_pin_cfg);

  /* Interrupt settings */
  /* (1) Set interrupt.*/
  Cy_SysInt_InitIRQ(&irq_cfg);
  Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, ButtonIntHandler);
  NVIC_SetPriority(irq_cfg.intIdx, 3ul);
  NVIC_EnableIRQ(irq_cfg.intIdx);

  /* Deepsleep mode transition start setting */
  /* (2) Set for Deepsleep mode transition.*/
  Deepsleep_transition = true;



  /* SRAM write buffer status check */
  /* (3) Check the write buffer status. See Code Listing 11.*/
  while(0ul==(Cy_Cpu_SramWriteBufferStatus(CY_CPU_SRAM0)));

  /* RAM retain mode setting */
  /* (4) Sets the RETAINED mode. See Code Listing 12.*/
```

```
    Cy_Cpu_SramPowerModeSet(CY_CPU_SRAM0, CY_CPU_SRAM_PM_RETAINED, 0ul);

    for(;;)
    {
      /* Set to transfer to Deepsleep mode only once */
      if( Deepsleep_transition == true)
      {
        Deepsleep_transition = false;

        /* Transfer to Deepsleep mode */
        /* (5) MCU enters DeepSleep mode. See Code Listing 16.*/
        Cy_SysPm_DeepSleep((cy_en_syspm_waitfor_t)CY_SYSPM_WAIT_FOR_INTERRUPT);
      }

      /* Continuously blink an LED0 to indicate waking up from DeepSleep */
      /* (7) Program execution resumes. The LED0 blink.*/
      Cy_SysTick_DelayInUs(50000ul);
      Cy_GPIO_Inv(USER_LED_PORT, USER_LED_PIN);
    }
  }
```

Code Listing 16 に、ドライバ部で DeepSleep モードへの移行を設定するプログラム例を示します。

Code Listing 16 ドライバ部での SysPm_DeepSleep のプログラム例

```
void cy_en_syspm_status_t Cy_SysPm_DeepSleep(cy_en_syspm_waitfor_t waitFor)
{
  uint32_t interruptState;
  cy_en_syspm_status_t retVal = CY_SYSPM_SUCCESS;

  /* Call the registered callback functions with
  * the CY_SYSPM_CHECK_READY parameter.
  */
  if(0u != currentRegisteredCallbacksNumber)
  {
    retVal = Cy_SysPm_ExecuteCallback(CY_SYSPM_DEEPSLEEP, CY_SYSPM_CHECK_READY);
  }

  /* The device (core) can switch into the deep sleep power mode only when
  *  all executed registered callback functions with the CY_SYSPM_CHECK_READY
  *  parameter returned CY_SYSPM_SUCCESS.
  */
  if(retVal == CY_SYSPM_SUCCESS)
  {
    /* Call the registered callback functions with the CY_SYSPM_BEFORE_TRANSITION
    * parameter. The return value is ignored.
    */
    interruptState = Cy_SysLib_EnterCriticalSection();
    if(0u != currentRegisteredCallbacksNumber)
    {
      (void) Cy_SysPm_ExecuteCallback(CY_SYSPM_DEEPSLEEP, CY_SYSPM_BEFORE_ENTER);
    }

    #if(0u != CY_CPU_CORTEX_M0P)


    /* The CPU enters the deep sleep mode upon execution of WFI/WFE */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;    //(5) MCU enters DeepSleep mode.

    if(waitFor != CY_SYSPM_WAIT_FOR_EVENT)
    {
      __WFI();
    }
    else
    {
      __WFE();
    }
    #else

    /* Repeat WFI/WFE instructions if wake up was not intended.
    */
    do
    {
      SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

      if(waitFor != CY_SYSPM_WAIT_FOR_EVENT)
```

```
        {
          __WFI();
        }
        else
        {
          __WFE();
        }
      } while (0);

      #endif /* (0u != CY_CPU_CORTEX_M0P) */

      Cy_SysLib_ExitCriticalSection(interruptState);

      /* Call the registered callback functions with the CY_SYSPM_AFTER_TRANSITION
      *  parameter. The return value is ignored.
      */
      if(0u != currentRegisteredCallbacksNumber)
      {
        (void) Cy_SysPm_ExecuteCallback(CY_SYSPM_DEEPSLEEP, CY_SYSPM_AFTER_EXIT);
      }
    }
    else
    {
      /* Execute callback functions with the CY_SYSPM_CHECK_FAIL parameter to
      * undo everything done in the callback with the CY_SYSPM_CHECK_READY
      * parameter. The return value is ignored.
      */
      (void) Cy_SysPm_ExecuteCallback(CY_SYSPM_DEEPSLEEP, CY_SYSPM_CHECK_FAIL);
      retVal = CY_SYSPM_FAIL;
    }
    return retVal;
  }
```

4.2 Hibernate モードの使用

このケースでは、Hibernate モード移行が使用されます。Hibernate モードの設定は、メインプログラム実行中の Active モードにおいて実施します。

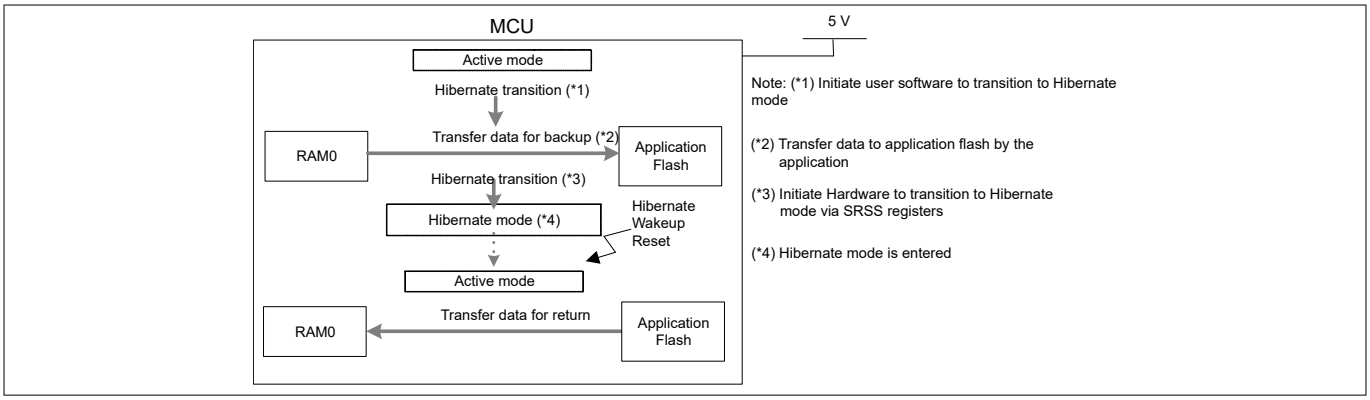

図 13 バックアップメモリデータの Hibernate モードでのバックアップ手順ブロックダイヤグラム

図 13 に、Active モードから Hibernate モードへの移行を示します。この移行の際に、アプリケーションフラッシュへのアクセスがチェックされます。もしアプリケーションフラッシュがアクセスできるようであれば、RAM からバックアップメモリデータは、アプリケーションフラッシュへ移行されます。データ移行が完了した後、ユーザソフトウェアは MCU が Hibernate モードへ入るように SRSS レジスタを設定します。もし Hibernate モードにおいて Hibernate wakeup リセットが発生したら、MCU は Active モードへ復帰します。Active モードへ移った後、バックアップメモリデータは、アプリケーションフラッシュから RAM へ移行されます。

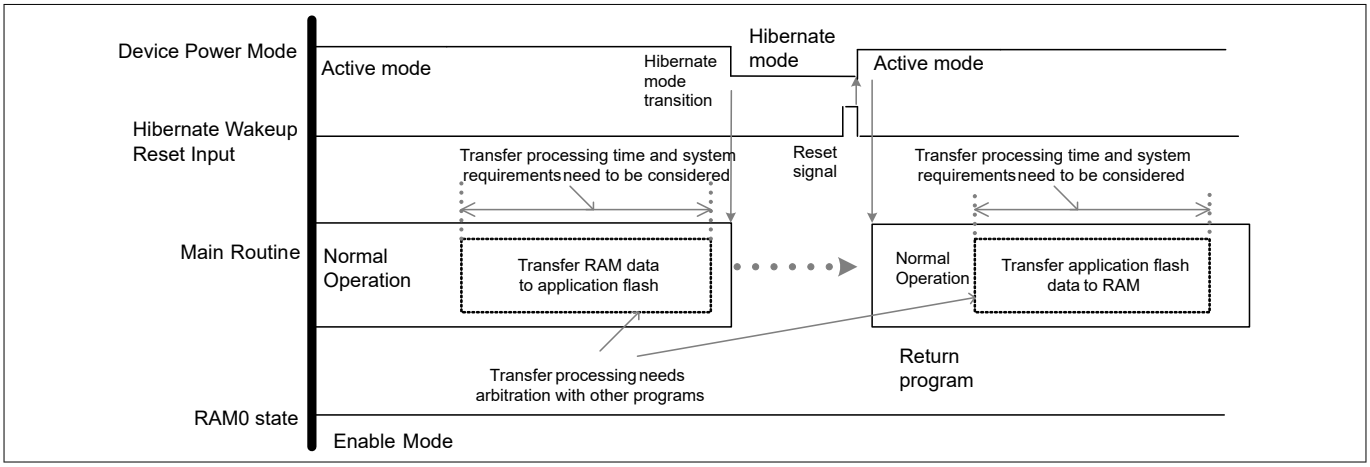

図 14　　バックアップメモリデータの Hibernate モードでのバックアップ手順タイミングチャート例

図 14 において、Active モードから Hibernate モードの移行の際、メインプログラムはバックアップメモリデータを RAM からアプリケーション フラッシュへ移行します。アプリケーション フラッシュに書込み中、ユーザはアクセスを調停しなければいけません。この調停は RAM とアプリケーション フラッシュのアクセスと他のアプリケーションプログラムの間です。この調停はバックアップ アクセス中に他のプログラムがアクセスするのを避けるためのものです。

調停の後、MCU は Hibernate モードへ移行して、メインルーチンの実行が停止します。Hibernate モードにおいて、Hibernate wakeup リセットが発生したとき、MCU は Active モードへ復帰します。メインルーチンはプログラムの実行を再開します。それによりバックアップメモリ データはアプリケーション フラッシュから RAM へ移行されます。この移行において、他のアプリケーションプログラムとの調停を考慮しなければいけません。

モード移行の時に、バックアップ データにおける RAM とアプリケーション フラッシュ移行時間と復帰はシステム要件を満たす必要があります。

4.2.1　　使用事例

ここでは、サンプル ドライバ ライブラリ (SDL) を使用した使用事例に基づいて、バックアップ メモリ データの Hibernate モード移行を設定する方法について説明します。このアプリケーションノートに記載されているプログラムコードは、SDL に含まれるものです。SDL については、その他の参考資料を参照してください。

使用事例:

- 低電力モード移行の設定: ハイバネートモード
- Active モードへ復帰の設定: ハイバネート復帰リセット
- Hibernate ウェイクアップ リセット ポートの設定: P21_4
- RAM とアプリケーション フラッシュ間の移行データの設定: 0x5A5A5A5A
- RAM アドレスの設定: 0x08000818
- アプリケーション フラッシュ アドレスの設定: 0x14000010

図 15 に、バックアップ メモリ データの Hibernate モード移行のフロー例を示します。

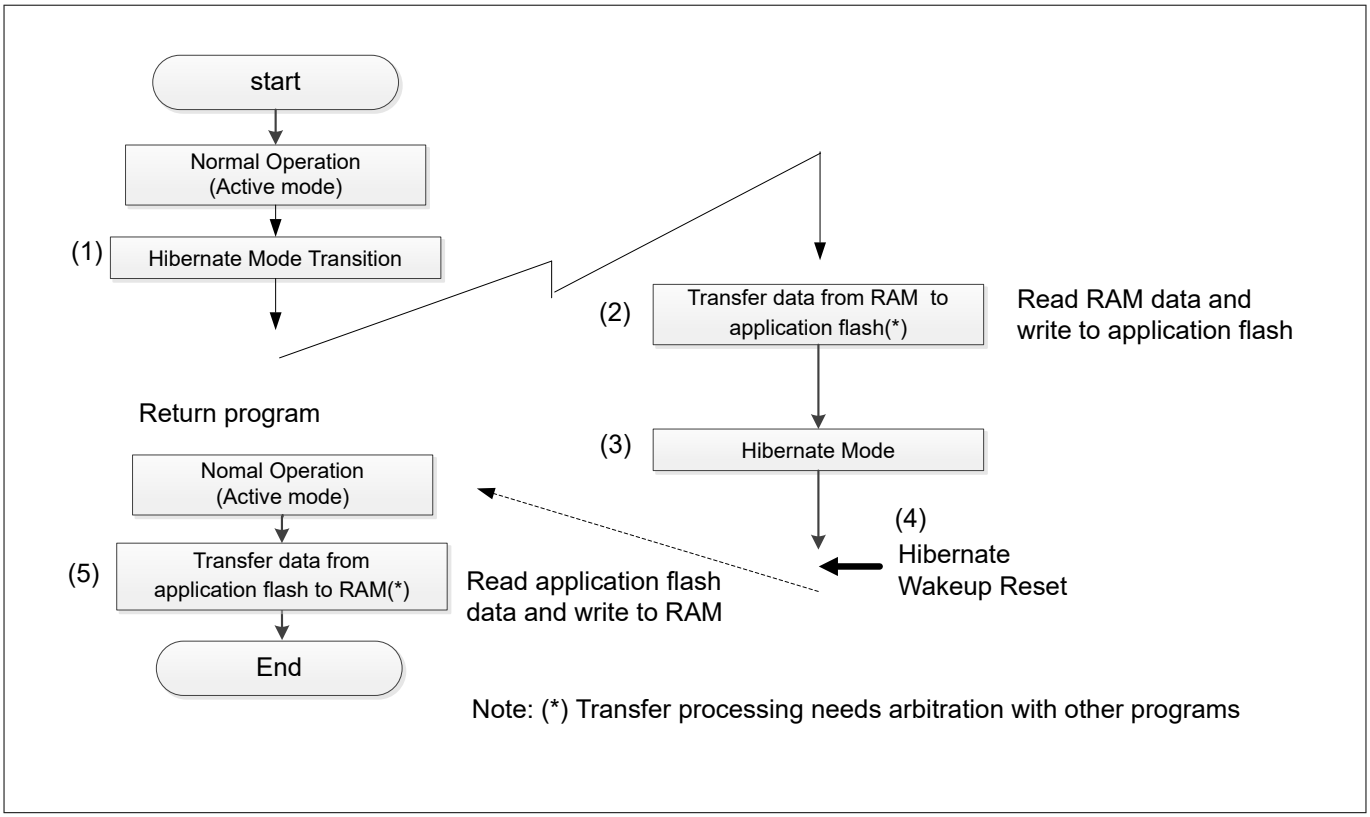

図 15　　バックアップメモリデータの Hibernate モード移行でのバックアップ手順例フローチャート

1. 正常動作中の Active モード時に、Hibernate モードを設定してください。
2. バックアップメモリデータを RAM からアプリケーションフラッシュへ移行してください。(RAM から読み出してアプリケーションフラッシュへ書き込む)
3. MCU は Hibernate モードへ移行します。
4. Hibernate ウェイクアップ リセットが発生したとき、MCU は Hibernate モードから Active モードへ変更します。
5. プログラムの実行が再開します。バックアップメモリデータはアプリケーションフラッシュから RAM へ移行します。(アプリケーションフラッシュへの読み出しと RAM への書き込み)

Hibernate モード移行設定の SDL の設定部のパラメーターを表 9 に、関数を表 10 に示します。

表 9　　バックアップ メモリ データのハイバネート モード移行時の初期パラメーターの一覧

パラメーター	説明	値
Hibernate_wakeup_port_pin_cfg. outVal	ピン出力状態	0ul
Hibernate_wakeup_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_HIGHZ
Hibernate_wakeup_port_pin_cfg. hsiom	I/O ピン配線の接続	WAKEUP_PIN_MUX
Hibernate_wakeup_port_pin_cfg. intEdge	IRQ をトリガするエッジ	CY_GPIO_INTR_RISING
Hibernate_wakeup_port_pin_cfg. intMask	マスク エッジ割込み	1ul
Hibernate_wakeup_port_pin_cfg. vtrip	入力バッファ モード	0ul,

(続く)

表 9 (続き) バックアップ メモリ データのハイバネート モード移行時の初期パラメーターの一覧

パラメーター	説明	値
Hibernate_wakeup_port_pin_cfg. slewRate	スルーレート	0ul
Hibernate_wakeup_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led0_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led0_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED0_PIN_MUX
user_led0_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led0_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led0_port_pin_cfg. vtrip	入力バッファ モード	0ul
user_led0_port_pin_cfg. slewRate	スルーレート	0ul
user_led0_port_pin_cfg. driveSel	GPIO 駆動強度	0ul
user_led1_port_pin_cfg. outVal	ピン出力状態	0ul
user_led_port_pin_cfg. driveMode	GPIO 駆動モード	CY_GPIO_DM_STRONG_IN_OFF
user_led1_port_pin_cfg. hsiom	I/O ピン配線の接続	USER_LED1_PIN_MUX
user_led1_port_pin_cfg. intEdge	IRQ をトリガするエッジ	0ul
user_led1_port_pin_cfg. intMask	マスク エッジ割込み	0ul
user_led1_port_pin_cfg. vtrip	入力バッファ モード	0ul
user_led1_port_pin_cfg. slewRate	スルーレート	0ul
user_led1_port_pin_cfg. driveSel	GPIO 駆動強度	0ul

表 10 バックアップ メモリ データの Hibernate モード移行初期の関数一覧

関数	説明	値
Cy_GPIO_Pin_Init (volatile stc_GPIO_PRT_t *base, uint32_t pinNum, const cy_stc_gpio_pin_config_t *config)	ピンのすべてのピン設定を初期化。 *base: ピンのポート レジスタ ベース アドレスへのポインタ pinNum: ポート レジスタ内のピン ビット フィールドの位置 *config: ピン コンフィギュレーション構造のベース アドレスへのポインタ	WAKEUP_PORT, WAKEUP_PIN, &Hibernate_wakeup_port_pin_cfg
Cy_SysPm_GetIoFreezeStatus(void)	IoFreeze が有効になっているかどうかを確認。	-
Cy_SysPm_IoUnfreeze(void)	IO を解凍。	-

(続く)

表 10 (続き) バックアップ メモリ データの Hibernate モード移行初期の関数一覧

関数	説明	値
Cy_SysReset_IsResetDueToHibWakeup (void)	Hibernate ウェイクアップが原因である場合は、リセット状態を返す。	-
Cy_FlashInit(bool non_blocking)	アプリケーションフラッシュを初期化。 non_blocking: 非ブロッキングモードを使用。	false
Cy_FlashWriteWork (uint32_t writeAddr, const uint32_t* data, cy_en_flash_driver_blocking_t blocking)	アプリケーションフラッシュでアドレス指定する 32 ビット データをプログラム。 writeAddr:書き込まれるアプリケーションフラッシュ セクタのアドレス data: 書き込むデータへのポインタ Blocking: param ブロッキング Blocking モードかどうか	TEST_WF_LS_ADDR, (uint32_t*)&rget_value, CY_FLASH_DRIVER_BLOCKING
Cy_SysPm_SetHibWakeupSource (cy_en_syspm_hib_wakeup_source_t wakeupSource)	Hibernate モードからデバイスをウェイクアップするようにソースを設定。 wakeupSource: ウェイクアップするソース	CY_SYSPM_HIBPIN0_HIGH
Cy_SysPm_Hibernate(void)	デバイスを Hibernate モードに設定。	-

Code Listing 17 に、バックアップ メモリ データの Hibernate モード移行の設定部のプログラム例を示します。

Code Listing 17 バックアップメモリデータ関数の Hibernate モード移行のプログラム例

```
/* Define for hibernate wakeup gpio */
/* Define the wakeup port settings.*/
#define WAKEUP_PORT                    GPIO_PRT21
#define WAKEUP_PIN                     4ul
#define WAKEUP_PIN_MUX                 P21_4_GPIO
#define WAKEUP_IRQ                     ioss_interrupts_gpio_21_IRQn

/* Define the transition data. */
#define RAM_0_DATA                     0x5A5A5A5Aul

/* Access to Application flash address */
/* Define the address for Flash.*/
#define TEST_WF_LS_ADDR                0x14000010ul

/* Set to wakeup pin for Hibernate mode return */
const cy_stc_gpio_pin_config_t Hibernate_wakeup_port_pin_cfg =
{
  /* Configure the port setting parameters for wakeup. See 表 9.*/
 .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_HIGHZ,
  .hsiom     = WAKEUP_PIN_MUX,
  .intEdge   = CY_GPIO_INTR_RISING,
  .intMask   = 1ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};

cy_stc_gpio_pin_config_t user_led0_port_pin_cfg =
{
/  * Configure the port setting parameters for LED0. See 表 9 .*/
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = CY_LED0_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
  .driveSel  = 0ul,
};

cy_stc_gpio_pin_config_t user_led1_port_pin_cfg =
{
  /* Configure the port setting parameters for LED1. See 表 9.*/
  .outVal    = 0ul,
  .driveMode = CY_GPIO_DM_STRONG_IN_OFF,
  .hsiom     = CY_LED1_PIN_MUX,
  .intEdge   = 0ul,
  .intMask   = 0ul,
  .vtrip     = 0ul,
  .slewRate  = 0ul,
```

```
  .driveSel  = 0ul,
};
/* Set to IRQ for hibernate wakeup gpio */
/* Configure the interrupt structure parameters.*/
const cy_stc_sysint_irq_t hibwakeup_irq_cfg =
{
  .sysIntSrc  = WAKEUP_IRQ,
  .intIdx     = CPUIntIdx2_IRQn,
  .isEnabled  = true,
};
/* Hibernate mode start flag */
uint8_t Hibernate_transition = false;

/* SRAM address for access */
/* Define the address for RAM0.*/
uint32_t variable_ram0_address = 0x08000818ul;

void Wakeup_IntHandler(void)      //(4) Interrupt routine for Hibernate wakeup.
{
  uint32_t intStatus;

  /* If wakeup raising edge detected */
  intStatus = Cy_GPIO_GetInterruptStatusMasked(WAKEUP_PORT, WAKEUP_PIN);
  if (intStatus != 0ul)
  {
    /* Clear to interrupt flag */
    Cy_GPIO_ClearInterrupt(WAKEUP_PORT, WAKEUP_PIN);
  }
}

int main(void)
{

  uint32_t RstReason = 0ul;

  __enable_irq(); /* Enable global interrupts. */

  SystemInit();

  /* Initialize for LED0, LED1 */
  /* Set for LED0, LED1.*/
  Cy_GPIO_Pin_Init(CY_LED0_PORT, CY_LED0_PIN, &user_led0_port_pin_cfg);
  Cy_GPIO_Pin_Init(CY_LED1_PORT, CY_LED1_PIN, &user_led1_port_pin_cfg);


  /* Initialize for wakeup pin(P21_4) */
  /* Set GPIO pin for Hibernate wakeup. See Code Listing 14.*/
  Cy_GPIO_Pin_Init(WAKEUP_PORT, WAKEUP_PIN, &Hibernate_wakeup_port_pin_cfg);

  /* Check I/O frozen status, unfreeze if it was frozen.*/
  if(Cy_SysPm_GetIoFreezeStatus())     //Check if the Frozen status of the I/O port. See Code
Listing 18.
  {
```

```
    /* Unfreeze the system */
    /* If it is in freeze status, release it. See Code Listing 19.*/
    Cy_SysPm_IoUnfreeze();
  }

  /* Interrupt settings */
  Cy_SysInt_InitIRQ(&hibwakeup_irq_cfg);
  Cy_SysInt_SetSystemIrqVector(hibwakeup_irq_cfg.sysIntSrc, Wakeup_IntHandler);
  NVIC_SetPriority(hibwakeup_irq_cfg.intIdx, 3ul);
  NVIC_EnableIRQ(hibwakeup_irq_cfg.intIdx);

  /* Check if the Hibernate wakeup reset */
  /* Check if the Hibernate wakeup reset cause in restart. See Code Listing 20.*/
  RstReason = Cy_SysReset_IsResetDueToHibWakeup();

  /* If the cause of RESET was Hibernate Wakeup */
  if(RstReason == true)
  {
    /*_Application flash read data value */
    uint32_t fget_value = 0ul;

    /* Read the data from Application flash(Workflash) */
    /* (5) Read to data in application flash and write to RAM in restart.*/
    fget_value = CY_GET_REG32(TEST_WF_LS_ADDR);

    /* Write the read data to RAM */
    /* (5) Read to data in application flash and write to RAM in restart.*/
    CY_SET_REG32(variable_ram0_address, fget_value);

    /* Check to flash read data */
    if(fget_value == 0x5A5A5A5Aul)
    {
      /* Turn on LED0 to indicate that Hibernate wakeup has occurred */
      for(uint16_t i = 0ul; i < 50ul; i++)     //The LED0 blink several times to indicate
completion of processing after Hibernate wakeup in restart.
      {
        Cy_SysTick_DelayInUs(50000ul);
        Cy_GPIO_Inv(CY_LED0_PORT, CY_LED0_PIN);
      }
    }
  }

  /* Hibernate mode transition start setting */
  /* (1) Set for Hibernate mode transtion.*/
  Hibernate_transition = true;

  /* Prepare the data "0x5A5A5A5A" in RAM0. Write to RAM0 in this data.  */
  CY_SET_REG32(variable_ram0_address, RAM_0_DATA);     //Prepare for transition data.

  for(;;)
  {
    /* Toggle an LED1 to notify MCU is working */
    for(uint16_t i = 0ul; i < 100ul; i++)     //The LED1 blink several times to indicate that
```

```
  it is in normal operation.
      {
        Cy_SysTick_DelayInUs(50000ul);
        Cy_GPIO_Inv(CY_LED1_PORT, CY_LED1_PIN);
      }

      /* RAM read data value */
      uint32_t rget_value = 0ul;

      /* Read in the RAM data "0x5A5A5A5A"  */
      rget_value = CY_GET_REG32(variable_ram0_address);



      /* Initialize to Flash */
      /* Initialize to application flash. See Code Listing 21.*/
      Cy_FlashInit(false);



      /* Write to Application flash(Work flash)  */
      /* Write to application flash. See Code Listing 22.*/
      Cy_FlashWriteWork(TEST_WF_LS_ADDR, (uint32_t*)&rget_value,
      CY_FLASH_DRIVER_BLOCKING);

      /* Enable hibernate wake up source pin */
      /* Set for wakeup source. See Code Listing 23.*/
      Cy_SysPm_SetHibWakeupSource(CY_SYSPM_HIBPIN0_HIGH);

      /* Transfer to Hibernate Mode  */
      /* (3) MCU enters Hibernate mode. See Code Listing 24.*/
      Cy_SysPm_Hibernate();

    }
  }
```

Code Listing 18～Code Listing 24 に、ドライバ部のバックアップ メモリ データの Hibernate モード移行を設定するプログラム例を示します。

Code Listing 18 ドライバ部の GetIoFreezeStatus のプログラム例

```
__STATIC_INLINE bool Cy_SysPm_GetIoFreezeStatus(void)
{
  return(0u != _FLD2VAL(SRSS_PWR_HIBERNATE_FREEZE, SRSS->unPWR_HIBERNATE.u32Register));
}
```

Code Listing 19 ドライバ部の IoUnfreeze のプログラム例

```
/*Check if the Frozen status of the I/O port.*/
void Cy_SysPm_IoUnfreeze(void)
{
  uint32_t interruptState;
  interruptState = Cy_SysLib_EnterCriticalSection();

  /* Preserve the last reset reason and wakeup polarity. Then, unfreeze I/O:
  * write PWR_HIBERNATE.FREEZE=0, .UNLOCK=0x3A, .HIBERANTE=0,
  */
  /* Unfreeze I/O port.*/
  SRSS->unPWR_HIBERNATE.u32Register = (SRSS->unPWR_HIBERNATE.u32Register &
  CY_SYSPM_PWR_RETAIN_HIBERNATE_STATUS) | CY_SYSPM_PWR_HIBERNATE_UNLOCK;

  /* If Read stands after Write, read this register two times to delay
  *  enough time for internal settling.
  */
  (void) SRSS->unPWR_HIBERNATE.u32Register;
  (void) SRSS->unPWR_HIBERNATE.u32Register;

  /* Lock the hibernate mode:
  * write PWR_HIBERNATE.HIBERNATE=0, UNLOCK=0x00, HIBERANTE=0
  */
  SRSS->unPWR_HIBERNATE.u32Register &= CY_SYSPM_PWR_RETAIN_HIBERNATE_STATUS;

  Cy_SysLib_ExitCriticalSection(interruptState);
}
```

Code Listing 20 ドライバ部の IsResetDueToHibWakeup のプログラム例

```
bool Cy_SysReset_IsResetDueToHibWakeup(void)
{
  bool retReason = false;

  // Contains a 8-bit token that is retained through a HIBERNATE/WAKEUP sequence
  // that can be used by firmware to differentiate WAKEUP from a general RESET event
  // Value 0x1B into the TOKEN is added before jumping to HIBERNATE
  if(0x1Bu == SRSS->unPWR_HIBERNATE.stcField.u8TOKEN)
  {
    retReason = true;    //Check the Hibernate wakeup reset cause
  }

  return retReason;
}
```

Code Listing 21 ドライバ部の FlashInit のプログラム例

```
void Cy_FlashInit(bool non_blocking)
{
  if(non_blocking == true)
  {
    /******  Setting for IPCs    ******/
    Cy_Srom_SetResponseHandler(Cy_FlashHandler, CPUIntIdx3_IRQn);
    NVIC_SetPriority(CPUIntIdx3_IRQn, 3ul);
    NVIC_EnableIRQ(CPUIntIdx3_IRQn);
    g_NB_ModeEnabled = true;
  }
  else
  {
    g_NB_ModeEnabled = false;
  }

  /*  Flash Write Enable   */
  /* Initialization for writing to application Flash.*/
  Cy_Flashc_MainWriteEnable();
  Cy_Flashc_WorkWriteEnable();

  g_completeFlag = true;
}
```

Code Listing 22 ドライバ部の FlashWriteWork のプログラム例

```
void Cy_FlashWriteWork(uint32_t writeAddr, const uint32_t* data, cy_en_flash_driver_blocking_t
blocking)
{
  CY_ASSERT(Cy_Flash_WorkBoundsCheck(writeAddr) == CY_FLASH_IN_BOUNDS);

  uint32_t status;
  cy_stc_flash_programrow_config_t programRowConfig = {0};

  if(blocking == CY_FLASH_DRIVER_NON_BLOCKING)
  {
    CY_ASSERT(g_NB_ModeEnabled == true);

    /* Only for non-blocking operation */
    g_completeFlag = false;
  }

  //  Program work flash
  programRowConfig.blocking = CY_FLASH_PROGRAMROW_BLOCKING;
  programRowConfig.skipBC   = CY_FLASH_PROGRAMROW_SKIP_BLANK_CHECK;
  programRowConfig.dataSize = CY_FLASH_PROGRAMROW_DATA_SIZE_32BIT;
  programRowConfig.dataLoc  = CY_FLASH_PROGRAMROW_DATA_LOCATION_SRAM;
  programRowConfig.intrMask = CY_FLASH_PROGRAMROW_NOT_SET_INTR_MASK;
  programRowConfig.destAddr = (uint32_t*)writeAddr;
  programRowConfig.dataAddr = data;
  /* Write to data in application flash.*/
  status = Cy_Flash_ProgramRow(NULL, &programRowConfig, blocking);
  CY_ASSERT(status == CY_FLASH_DRV_SUCCESS);
}
```

Code Listing 23 ドライバ部の SetHibWakeupSource のプログラム例

```
void Cy_SysPm_SetHibWakeupSource(cy_en_syspm_hib_wakeup_source_t wakeupSource)
{
  /* Reconfigure the wake-up pins and LPComp polarity based on the input */
  if(0u != ((uint32_t) wakeupSource & CY_SYSPM_WAKEUP_LPCOMP0))
  {
  SRSS->unPWR_HIBERNATE.u32Register &=
    ((uint32_t) ~_VAL2FLD(SRSS_PWR_HIBERNATE_POLARITY_HIBPIN, CY_SYSPM_WAKEUP_LPCOMP0_BIT));
  }

            if(0u != ((uint32_t) wakeupSource & CY_SYSPM_WAKEUP_LPCOMP1))
  {
  SRSS->unPWR_HIBERNATE.u32Register &=
    ((uint32_t) ~_VAL2FLD(SRSS_PWR_HIBERNATE_POLARITY_HIBPIN, CY_SYSPM_WAKEUP_LPCOMP1_BIT));
  }

            if(0u != ((uint32_t) wakeupSource & CY_SYSPM_WAKEUP_PIN0))
  {
  SRSS->unPWR_HIBERNATE.u32Register &=
    ((uint32_t) ~_VAL2FLD(SRSS_PWR_HIBERNATE_POLARITY_HIBPIN, CY_SYSPM_WAKEUP_PIN0_BIT));
  }

            if(0u != ((uint32_t) wakeupSource & CY_SYSPM_WAKEUP_PIN1))
  {
    SRSS->unPWR_HIBERNATE.u32Register &=
    ((uint32_t) ~_VAL2FLD(SRSS_PWR_HIBERNATE_POLARITY_HIBPIN, CY_SYSPM_WAKEUP_PIN1_BIT));
  }

  /* Set for wakeup source in pin0.*/
  SRSS->unPWR_HIBERNATE.u32Register |= ((uint32_t) wakeupSource);
}
```

Code Listing 24 ドライバ部の SysPm_Hibernate のプログラム例

```
cy_en_syspm_status_t Cy_SysPm_Hibernate(void)
{
  cy_en_syspm_status_t retVal = CY_SYSPM_SUCCESS;

  /* Call the registered callback functions with the
  * CY_SYSPM_CHECK_READY parameter
  */
  if(0u != currentRegisteredCallbacksNumber)
  {
    retVal = Cy_SysPm_ExecuteCallback(CY_SYSPM_HIBERNATE, CY_SYSPM_CHECK_READY);
  }

  /* The device (core) can switch into Hibernate power mode only when
  *  all executed registered callback functions with CY_SYSPM_CHECK_READY
  *  parameter returned CY_SYSPM_SUCCESS.
  */
  if(retVal == CY_SYSPM_SUCCESS)
  {
    /* Call registered callback functions with CY_SYSPM_BEFORE_TRANSITION
    *  parameter. Return value is ignored.
    */
    (void) Cy_SysLib_EnterCriticalSection();
    if(0u != currentRegisteredCallbacksNumber)
    {
      (void) Cy_SysPm_ExecuteCallback(CY_SYSPM_HIBERNATE, CY_SYSPM_BEFORE_ENTER);
    }

    /* Preserve the token that will retain through a wakeup sequence
    * thus could be used by Cy_SysReset_GetResetReason() to differentiate
    * Wakeup from a general reset event.
    * Preserve the wakeup source(s) configuration.
    */
    /* (3) MCU enters Hibernate mode.*/
    SRSS->unPWR_HIBERNATE.u32Register =
    (SRSS->unPWR_HIBERNATE.u32Register & CY_SYSPM_PWR_WAKEUP_HIB_MASK) |
CY_SYSPM_PWR_TOKEN_HIBERNATE;

    /* All the three writes to hibernate register use the same value:
    *  PWR_HIBERNATE.FREEZE=1, .UNLOCK=0x3A, .HIBERANTE=1,
    */
    SRSS->unPWR_HIBERNATE.u32Register |= CY_SYSPM_PWR_SET_HIBERNATE;

    SRSS->unPWR_HIBERNATE.u32Register |= CY_SYSPM_PWR_SET_HIBERNATE;

    SRSS->unPWR_HIBERNATE.u32Register |= CY_SYSPM_PWR_SET_HIBERNATE;

    /* Wait for transition */
    __WFI();

    /* The callback functions calls with the CY_SYSPM_AFTER_TRANSITION
    * parameter in the hibernate power mode are not applicable as device
```

```
      * wake-up was made on device reboot.
      */

      /* A wakeup from the hibernate is performed by toggling of the wakeup
      * pins, or WDT matches, or Backup domain alarm expires. Depends on what
      * item is configured in the hibernate register. After a wakeup event, a
      * normal Boot procedure occurs.
      * No need to exit from the critical section.
      */
    }
    else
    {
      /* Execute callback functions with the CY_SYSPM_CHECK_FAIL parameter to
      * undo everything done in the callback with the CY_SYSPM_CHECK_READY
      * parameter. The return value is ignored.
      */
      (void) Cy_SysPm_ExecuteCallback(CY_SYSPM_HIBERNATE, CY_SYSPM_CHECK_FAIL);
      retVal = CY_SYSPM_FAIL;
    }
    return retVal;
  }
```

5 用語集

表 11 用語集

用語	説明
CPU	Central Processing Unit (中央演算処理装置)
BOD	Brown-Out Detection。詳細は architecture TRM の"Power Supply and Monitoring"章を参照してください。
LVD	Low-Voltage Detection (低電圧検出)。詳細は architecture TRM の"Power Supply and Monitoring"章を参照してください。
VDDD	Digital power supply。詳細は architecture TRM の"Power Supply and Monitoring"章を参照してください。
GPIO	General purpose input/output (汎用入出力)。詳細は architecture TRM の"IO System"章を参照してください。
ECC	Error Correcting Code (誤り訂正符号)
CPUSS	CPU subsystem (CPU サブシステム)
MCU	Microcontroller Unit (マイクロコントローラー ユニット)

6 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。これらのドキュメントを入手するには、テクニカルサポートに連絡してください。

- デバイスデータシート
  - CYT2B7 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
  - CYT2B9 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
  - 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
  - CYT3DL datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-27763)
- Body Controller Entry ファミリ
  - TRAVEO™ T2G automotive body controller entry family architecture technical reference manual (TRM)
  - TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B7
  - TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B9
- Body Controller High ファミリ
  - TRAVEO™ T2G automotive body controller high family architecture technical reference manual (TRM)
  - 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
- Cluster 2D ファミリ
  - TRAVEO™ T2G automotive cluster 2D family architecture technical reference manual (TRM) (Doc No. 002-25800)
  - 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. 002-29584)

7 その他の参考資料

インフィニオンは、さまざまな周辺機器にアクセスするためのサンプル ソフトウェアとしてスタートアップを含むサンプル ドライバ ライブラリ (SDL) を提供しています。SDL は、AUTOSAR の公式製品ではカバーされていないドライバのリファレンスとしての役割も担っています。SDL は自動車用規格に適合していないため、量産用としては使用できません。このアプリケーションノートに記載されているプログラムコードは、SDL に含まれるものです。SDL を入手するには、テクニカルサポートに連絡してください。

改訂履歴

版数	発行日	変更内容
**	2019-06-19	これは英語版 002-20152 Rev. **を翻訳した日本語版 002-27534 Rev. **です。
英語版*A	2019-10-24	英語版の改訂内容: Added CYT4D Series parts related information in all instances across the document. Added “RAM Retention Procedure in Low-Power Mode”.
*A	2020-08-24	これは英語版 002-20152 Rev. *B を翻訳した日本語版 002-27534 Rev. *A です。英語版の改訂内容: Changed parts (from CYT2B/CYT4B/CYT4D Series to CYT2/CYT4 Series) in all instances across the document. Added CYT3 Series parts related information in all instances across the document.
*B	2021-02-09	これは英語版 002-20152 Rev. *C を翻訳した日本語版 002-27534 Rev. *B です。英語版の改訂内容: Updated RAM Retention Procedure Overview: Added “Low-Power Mode (Hibernate Mode) Transition Procedure”.
*C	2022-08-17	テンプレートの変更を実施。これは英語版 002-20152 Rev. *D を翻訳した日本語版 002-27534 Rev. *C です。
*D	2022-08-17	これは英語版 002-20152 Rev. *E を翻訳した日本語版 002-27534 Rev. *D です。英語版の改訂内容: Updated code examples using SDL.
*E	2024-04-12	これは英語版 002-20152 Rev. *F を翻訳した日本語版 002-27534 Rev. *E です。英語版の改訂内容: Template update; no content updated.

Trademarks

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

Edition 2024-04-12

Published by

Infineon Technologies AG

81726 Munich, Germany

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

Email: erratum@infineon.com

Document reference
IFX-agf1681379995630

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。本製品の使用の前に、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。