RA6M2 グループ

静電容量タッチ低消費電力ガイド

要旨

本アプリケーションノートは、RA6M2 に搭載された非同期汎用タイマ（以下：AGT）機能と低消費電力モード（ディープソフトウェアスタンバイモード）を用いて、静電容量タッチ計測の低消費間欠動作について説明します。

動作確認デバイス

RA6M2 グループ

目次

1. 仕様

1.1 プロジェクト説明

本アプリケーションノートで解説するサンプルコードは、RA6M2 グループ 静電容量タッチ評価システム（RTK0EG0021S01001BJ）で動作を確認したプロジェクトです。このプロジェクトの設定は RA6M2 グループ 静電容量タッチ評価システムに実装されている R7FA6M2AF3CFB に合わせています。その他のデバイスの場合は、プロジェクトの設定でデバイスを変更してご使用ください。

1.2 周辺機能

表 1-1 にサンプルコードで使用する周辺機能を示します。

表 1-1　周辺機能と用途

周辺機能	用途
静電容量式タッチセンシングユニット（CTSU）	- タッチ電極に発生する静電容量を計測
データトランスファコントローラ（DTC）	- タッチ計測時の CTSUSSC、CTSUSO0、CTSUSO1 の設定値を RAM から CTSU レジスタに転送 - CTSUSC、CTSURC カウンタを CTSU レジスタから RAM に転送
非同期汎用タイマ（AGT）	- ソフトウェアスタンバイモードからスヌーズモードへ切り替え - ディープソフトウェアスタンバイモードを解除
イベントリンクコントローラ（ELC）	- スヌーズエントリで CTSU 計測を開始

1.3 CPU 動作モード

表 1-2 にサンプルコードで使用する CPU 動作モードを記載します。

表 1-2　CPU 動作モード

CPU 動作モード	遷移条件
通常モード （MCU 起動/タッチ計測開始）	ディープソフトウェアスタンバイモードから非同期汎用タイマ 100ms 経過
ソフトウェアスタンバイモード	r_lpm ドライバの API から WFI 命令を実行
スヌーズモード	ソフトウェアスタンバイモードから AGT1 アンダーフローでスヌーズモードに遷移
通常モード （タッチ計測終了処理 + タッチオン/オフ判定処理）	スヌーズモードから CTSU 計測終了割り込みで遷移
ディープソフトウェアスタンバイモード	r_lpm ドライバの API から WFI 命令を実行

1.4　CTSU 動作状態

表 1-3 にサンプルコードで使用する CTSU 動作状態を記載します。

表 1-3　CTSU 動作状態

CTSU 動作状態	遷移条件
停止	- リセット状態
動作（サスペンド状態）	- タッチ計測開始処理で CTSU 待機時省電力を有効
動作	- タッチ計測開始
停止	- タッチ計測終了処理・タッチオン/オフ判定処理に先立ち、CTSU をモジュールストップ状態へ遷移
停止（不定）	- ディープソフトウェアスタンバイモード

1.5　レジスタ設定

リセット後の値から変更したレジスタの設定内容を以下に示します。

表 1-4 レジスタ設定

機能	レジスタ名	設定値	備考
I/O ポート	P000PFS	0x00000000	
	P001PFS	0x00000000	
	P002PFS	0x00000000	
	P003PFS	0x00000000	
	P004PFS	0x00000000	
	P005PFS	0x00000000	
	P006PFS	0x00000000	
	P007PFS	0x00000000	
	P108PFS	0x00000000	
	P109PFS	0x00000000	
	P110PFS	0x00000000	
	P201PFS	0x00000000	
	P300PFS	0x00000000	
バッテリバックアップ機能	VBTICTLR	0x00	
リアルタイムクロック（RTC）	RCR2	0x00	RA6M2 グループ ユーザーズマニュアル ハードウェア編 26.6.7「リアルタイムクロックを使用しない場合の初期化手順」に従って設定。
	RCR4	0x01	RCKSEL ビット（カウントソース選択）: 1 LOCO を選択
低消費電力モード	SBYCR	0x8000	各ビットの設定は以下のとおり。 OPE ビット（出力ポート許可）: 0 アドレスバスとバス制御信号をハイインピーダンス状態に設定 SSBY ビット（ソフトウェアスタンバイ）: 1 ディープソフトウェアスタンバイモード

	SNZCR	0x82	各ビットの設定は以下のとおり。 （スヌーズモードの場合） RXDREQEN ビット（RXD0 スヌーズ要求許可）: 0 ソフトウェアスタンバイモード時に RXD0 の立ち下がりエッジを無視 SNZDTCEN ビット（スヌーズモード時の DTC 許可）: 1 スヌーズモード時に DTC 動作を許可 SNZE（スヌーズモード許可）: 1 スヌーズモードを許可
	DPSBYCR	0x80	各ビットの設定は以下のとおり。 DEEPCUT ビット（電源制御）: 0 スタンバイ SRAM、低速オンチップオシレータ、AGTn、および USBFS レジューム検出部へ電源を供給する IOKEEP ビット（I/O ポート保持）: 0 ディープソフトウェアスタンバイモード解除時に、I/O ポートをリセット状態にクリア DPSBY ビット（ディープソフトウェアスタンバイ）: 1 ディープソフトウェアスタンバイモード
クロック発生回路	SOSCCR	0x01	SOSTP ビット（サブクロック発振器停止）: 1 サブクロック発信器停止

1.6　ファイル構成

表 1-5 に、RA コンフィグレータ、QE for Capacitive Touch で生成したサンプルコードから追加、変更したファイルを示します。

表 1-5　サンプルコードで追加・変更したファイル

ファイル名	処理・設定概要	備考
qe_touch_sample.c	メイン処理	変更ファイル
r_captouch_lpm.c	スタンバイ SRAM 制御	追加ファイル
r_captouch_lpm.h	スタンバイ SRAM 制御関数の定義	追加ファイル
r_user_warm_start.c	リセット要因、ディープソフトウェアスタンバイモード解除要因の検出	追加ファイル
r_user_warm_start.h	リセット要因、ディープソフトウェアスタンバイモード解除要因の検出要因の定義	追加ファイル

2. 動作確認条件

サンプルコードは、表 2-1 の条件で動作を確認しています。

表 2-1　動作確認条件

項目	内容
使用マイコン	R7FA6M2AF3CFB（RA6M2 グループ）
CPU 動作周波数	20MHz　高速オンチップオシレータ（HOCO） 32kHz　低速オンチップオシレータ（LOCO）
動作電圧	5.0V
ターゲットボード	RA6M2 グループ 静電容量タッチ評価システム （製品型名：RTK0EG0021S01001BJ）
開発環境	e^2 studio (2021-10)
C コンパイラ	GCC ARM Embedded（9.3.1.20200408）
動作モード	シングルチップモード
デバッガ	E2 エミュレータ Lite
FSP のバージョン	Ver.3.4.0
サンプルコードのバージョン	Ver.1.00

3.　ソフトウェア説明

サンプルコードでは、表 3-1 で示すドライバ、ミドルウェアを使用して、以下の動作を行います。

1. パワーオンによるリセット解除後に rm_touch ミドルウェアをオープンし、オフセット調整、初期キャリブレーションを実行してタッチボタンのタッチオン/オフ判定を動作可能な状態にします。
2. rm_touch ミドルウェアに関連する情報をスタンバイ SRAM に退避します。
3. ディープソフトウェアスタンバイモードに遷移します。
4. ディープソフトウェアスタンバイモードからリセット解除後にスタンバイ SRAM に退避した rm_touch ミドルウェアに関連する情報を RAM にロードします。
5. ソフトウェアスタンバイモードに遷移します。
6. AGT カウンタアンダーフローでスヌーズモードに遷移し、スヌーズモード遷移をトリガーに CTSU 計測を開始します。
7. CTSU 計測終了割り込みで、スヌーズモードから通常モードに復帰します。
8. タッチボタンのタッチオンを検出時、ユーザ LED（LED3）を点灯します。
9. 以降 2 から 9 を繰り返し、タッチオンを 5 回連続で検出時にユーザ LED を点滅します。

3.1　動作イメージ

図 3-1 にサンプルコードの処理概要に応じた CPU 動作モードと CTSU 動作状態のイメージを示します。

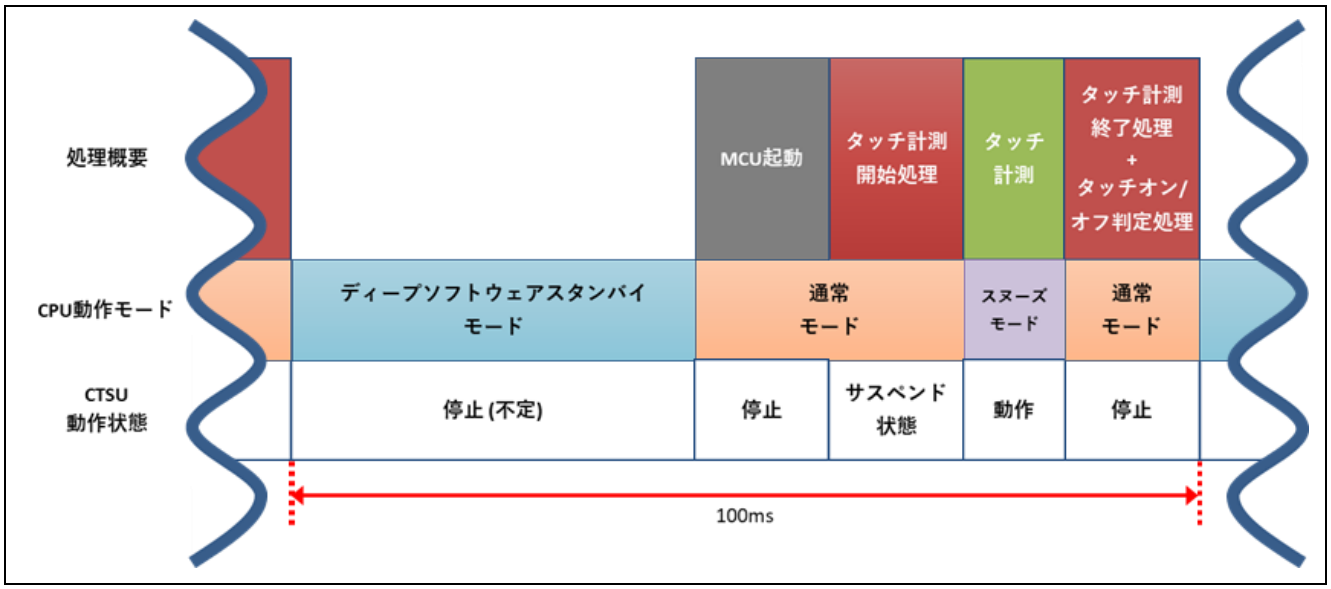

図 3-1　動作イメージ

3.2　FSP ドライバ、ミドルウェア

サンプルコードでは、表 3-1 に示す FSP ドライバ、ミドルウェアを使用しています。

表 3-1　FSP ドライバ、ミドルウェア一覧

ドライバ、ミドルウェア	バージョン
ボードサポートパッケージ（BSP）モジュール	3.4.0
CTSU Driver (r_ctsu)	3.4.0
Capacitive Touch Middleware (rm_touch)	3.4.0
I/O Port Driver (r_ioport)	3.4.0
Timer Driver (r_agt)	3.4.0
Low Power Mode Driver (r_lpm)	3.4.0
Event Link Controller (r_elc)	3.4.0
DTC Driver (r_dtc)	3.4.0

3.2.1　スタンバイ SRAM に関する注意点

スタンバイ SRAM を使用するにあたって、ユーザーズマニュアルに記載されているスタンバイ SRAM の SRAM アドレス、SRAM 容量に従って、リンカースクリプトファイルのセクション情報にスタンバイ SRAM のセクション定義を追加してください。

サンプルコードでは、script¥fsp.ld がリンカースクリプトファイルに該当します。

3.3　定数一覧

ユーザが変更可能な定数を表 3-2 に示します。

表 3-2　定数一覧（qe_touch_sample.c）

定数名	設定値	内容
DETECTED_PRESSES	5U	タッチオン検出回数
TOUCH_BUTTON_00	1U	タッチボタンのタッチオン/オフ確認向けビット位置
NO_DETECTED_PRESS	0U	タッチオン非検出
LED_LOOP_MAX	10U	タッチオン検出時の LED 制御回数
LED_CYCLE_MS	50U	タッチオン検出時の LED 点灯サイクル
RTC_STABLIZATION_TIME_US	190U	RTC クロック安定待ち時間
DEEP_STBY_TIME	0xcc0	ディープソフトウェアスタンバイモード時間

表 3-3　定数一覧（r_user_warm_start.h）

定数名	設定値	内容
IWDT_RESET_ENABLED	0	0: 独立ウォッチドッグタイマリセットを無効にします 1: 独立ウォッチドッグタイマリセットを有効にします
WDT_RESET_ENABLED	0	0: ウォッチドッグタイマリセットを無効にします 1: ウォッチドッグタイマリセットを有効にします
VOLTAGE_MONITOR_0_RESET_ENABLED	0	0: 電圧監視 0 リセットを無効にします 1: 電圧監視 0 リセットを有効にします
VOLTAGE_MONITOR_1_RESET_ENABLED	0	0: 電圧監視 1 リセットを無効にします 1: 電圧監視 1 リセットを有効にします
VOLTAGE_MONITOR_2_RESET_ENABLED	0	0: 電圧監視 2 リセットを無効にします 1: 電圧監視 2 リセットを有効にします
SRAM_PARITY_RESET_ENABLED	0	0: SRAM パリティエラーリセットを無効にします 1: SRAM パリティエラーリセットを有効にします
SRAM_ECC_RESET_ENABLED	0	0: SRAM ECC エラーリセットを無効にします 1: SRAM ECC エラーリセットを有効にします
BUS_MPU_SLAVE_RESET_ENABLED	0	0: バススレーブ MPU エラーリセットを無効にします 1: バススレーブ MPU エラーリセットを有効にします
BUS_MPU_MASTER_RESET_ENABLED	0	0: バスマスタ MPU エラーリセットを無効にします 1: バスマスタ MPU エラーリセットを有効にします
CPU_SP_RESET_ENABLED	0	0: スタックポインタエラーリセットを無効にします 1: スタックポインタエラーリセットを有効にします
CPU_DEEP_SW_STBY_RESET_ENABLED	1	0: ディープソフトウェアスタンバイモード解除によるリセットを無効にします 1: ディープソフトウェアスタンバイモード解除によるリセットを有効にします
CPU_SOFTWARE_RESET_ENABLED	1	0: ソフトウェアリセットを無効にします 1: ソフトウェアリセットを有効にします

3.4 構造体一覧

サンプルコードで定義する構造体を以下に示します。

3.4.1 リセット要因（r_user_warm_start.h）

```
/* Reset source */
typedef struct st_reset_source
{
    union
    {
        volatile uint16_t all;

        struct
        {
            volatile uint16_t power_on_reset_detect_flag : 1; /* Power-On reset
*/
            volatile uint16_t voltage_monitor_0_reset_detect_flag : 1; /*
Voltage monitor 0 reset */
            volatile uint16_t voltage_monitor_1_reset_detect_flag : 1; /*
Voltage monitor 1 reset */
            volatile uint16_t voltage_monitor_2_reset_detect_flag : 1; /*
Voltage monitor 2 reset */
            volatile uint16_t deep_software_standby_reset_flag : 1; /* Deep
Software Standby reset */
            volatile uint16_t independent_watchdog_timer_reset_detect_flag : 1;
/* IWDT underflow/refresh error reset */
            volatile uint16_t watchdog_timer_reset_detect_flag : 1; /* WDT
underflow/refresh error reset */
            volatile uint16_t software_reset_detect_flag : 1; /* Software reset
*/
            volatile uint16_t sram_parity_error_reset_detect_flag : 1; /* SRAM
parity error reset */
            volatile uint16_t sram_ecc_error_reset_detect_flag : 1; /* SRAM ECC
error reset */
            volatile uint16_t bus_slave_mpu_error_reset_detect_flag : 1; /* MPU
bus slave error reset */
            volatile uint16_t bus_master_mpu_error_reset_detect_flag : 1; /* MPU
bus master error reset */
            volatile uint16_t sp_error_reset_detect_flag : 1; /* CPU stack
pointer monitor reset */
            volatile uint16_t cold_warm_start_determination_flag : 1; /*
Cold/Warm Start detection */
            volatile uint16_t : 2;
        } flag;
    };
} reset_source_t;
```

3.4.2 ディープソフトウェアスタンバイモード解除要因（r_user_warm_start.h）

```
/* Deep Software Standby cancel source */
typedef struct st_dp_sw_stby_source
{
    union
    {
        volatile uint32_t all;

        struct
        {
            volatile uint32_t irq0_ds : 1;   /* IRQ0 */
            volatile uint32_t irq1_ds : 1;   /* IRQ1 */
            volatile uint32_t irq2_ds : 1;   /* IRQ2 */
            volatile uint32_t irq3_ds : 1;   /* IRQ3 */
            volatile uint32_t irq4_ds : 1;   /* IRQ4 */
            volatile uint32_t irq5_ds : 1;   /* IRQ5 */
            volatile uint32_t irq6_ds : 1;   /* IRQ6 */
            volatile uint32_t irq7_ds : 1;   /* IRQ7 */
            volatile uint32_t irq8_ds : 1;   /* IRQ8 */
            volatile uint32_t irq9_ds : 1;   /* IRQ9 */
            volatile uint32_t irq10_ds : 1;  /* IRQ10 */
            volatile uint32_t irq11_ds : 1;  /* IRQ11 */
            volatile uint32_t irq12_ds : 1;  /* IRQ12 */
            volatile uint32_t irq13_ds : 1;  /* IRQ13 */
            volatile uint32_t irq14_ds : 1;  /* IRQ14 */
            volatile uint32_t irq15_ds : 1;  /* IRQ15 */
            volatile uint32_t lvd1_ds : 1;   /* LVD1 */
            volatile uint32_t lvd2_ds : 1;   /* LVD2 */
            volatile uint32_t rtcint_ds : 1; /* RTC Interval */
            volatile uint32_t rtcalm_ds : 1; /* RTC Alarm */
            volatile uint32_t nmi_ds : 1;    /* NMI */
            volatile uint32_t usbf_ds : 1;   /* USBFS */
            volatile uint32_t usbh_ds : 1;   /* USBHS */
            volatile uint32_t agt1_ds : 2;   /* AGT1 */
        } flag;
    };
} dp_sw_stby_source_t;
```

3.5　変数一覧

サンプルコードで追加・変更したファイルで定義する変数を以下に示します。

表 3-4 変数一覧（qe_touch_main.c）

型	変数名	内容
static uint8_t	g_button_status[DETECTED_PRESSES]	タッチボタンの過去 5 回のタッチオン/オフ判定結果を格納する。
static uint32_t	g_button_status_index	タッチボタンの過去 5 回のタッチオン/オフ判定結果の格納先のインデックスを示す。
uint64_t	button_status	タッチボタンのタッチオン/オフ判定結果を示す。
uint64_t	last_button_status	前回のタッチボタンのタッチオン/オフ判定結果を示す。

表 3-5 変数一覧（r_user_warm_start.c）

型	変数名	内容
volatile reset_source_t	g_reset_source	リセット要因を示す。
volatile dp_sw_stby_source_t	g_dp_sw_stby_source	ディープソフトウェアスタンバイモードの解除要因を示す。

表 3-6 変数一覧（r_captouch_lpm.c）

型	変数名	内容
static uint16	g_power_on_reset_symbol	パワーオンリセットの有無を示す。
static uint16_t	g_touch_button_threshold_ssram[TOUCH_CFG_NUM_BUTTONS]	タッチボタンのタッチしきい値を示す。
static uint16_t	g_touch_button_hysteresis_ssram[TOUCH_CFG_NUM_BUTTONS]	タッチボタンのヒステリシスを示す。
static uint16_t	g_touch_button_reference_ssram[TOUCH_CFG_NUM_BUTTONS]	タッチボタンの基準値を示す。
static uint16_t	g_touch_button_on_count_ssram[TOUCH_CFG_NUM_BUTTONS]	タッチボタンのタッチオンのカウンタを示す。
static uint16_t	g_touch_button_off_count_ssram[TOUCH_CFG_NUM_BUTTONS]	タッチボタンのタッチオフのカウンタを示す。
static uint32_t	g_touch_button_drift_buf_ssram[TOUCH_CFG_NUM_BUTTONS]	ドリフト補正を実行するまでのタッチボタンのカウント値の総計を示す。
static uint16_t	g_touch_button_drift_count_ssram[TOUCH_CFG_NUM_BUTTONS]	ドリフト補正を実行するまでのタッチボタンの CTSU 計測の回数を示す。
static int32_t	g_ctsu_tuning_diff_ssram[CTSU_CFG_NUM_SELF_ELEMENTS + CTSU_CFG_NUM_MUTUAL_ELEMENTS]	オフセット調整、初期キャリブレーション実行時におけるカウント値と理想的なカウント値との差分を示す。

static ctsu_ctsuwr_t	g_ctsu_ctsuwr_ssram[(CTSU_CFG_NUM_SELF_ELEMENTS + CTSU_CFG_NUM_MUTUAL_ELEMENTS) * CTSU_MULTI_NUM]	タッチボタンの CTSUSSC、CTSUSO0、CTSUSO1 レジスタの設定値を示す。
static ctsu_self_buf_t	g_ctsu_self_raw_ssram[CTSU_CFG_NUM_SELF_ELEMENTS * CTSU_MULTI_NUM]	タッチボタンの CTSUSC、CTSURC レジスタの値を示す。
static uint16_t	g_ctsu_self_data_ssram[CTSU_CFG_NUM_SELF_ELEMENTS]	タッチボタンのカウント値を示す。
static ctsu_correction_info_t	g_ctsu_correction_info_ssram	ICO 補正譲歩を示す。
static ctsu_instance_ctrl_t	g_qe_ctsu_ctrl_config_ssram	r_ctsu 制御向け構成情報を示す。
static touch_instance_ctrl_t	g_qe_touch_ctrl_config_ssram	rm_touch 制御向け構成情報を示す。
static transfer_info_t	gp_dtc_vector_table_ssram[CTSU_VECTOR_TABLE_ENTRIES]	DTC ベクターテーブルの情報を示す。
static uint32_t	dtc_vector_table_address_ssram	ベクターテーブルのアドレスを示す。
static dtc_instance_ctrl_t	g_transfer0_ctrl_ssram	CTSUWR 割り込み向け DTC 構成情報を示す。
static dtc_instance_ctrl_t	g_transfer1_ctrl_ssram	CTSURD 割り込み向け DTC 構成情報を示す。

- r_captouch_lpm.c の変数は、すべてスタンバイ SRAM のアドレスに割り付けています。

3.6　関数一覧

サンプルコードで追加した関数について説明します。

3.6.1　qe_touch_main ()

qe_touch_main ()	
概　要	メイン処理
宣　言	void qe_touch_main (void)
説　明	タッチ計測およびディープソフトウェアスタンバイモードへの遷移を制御します。 リセット要因によって、以下のように制御します。 - パワーオンリセット、デバッガによるソフトウェアリセット、リセット端子 1. タッチミドルウェアオープン 2. 初期キャリブレーション（初期オフセット調整） 3. タッチミドルウェア関連データをスタンバイ SRAM へセーブ 4. ディープソフトウェアスタンバイに遷移（AGT カウンタアンダーフローで解除） - 上記以外 1. タッチミドルウェア関連データをスタンバイ SRAM からロード 2. タッチ計測開始準備 3. ソフトウェアスタンバイモード、スヌーズモードに遷移 4. タッチ計測開始 5. タッチボタンのタッチオンを検出でユーザ LED を点滅 6. タッチミドルウェア関連データをスタンバイ SRAM へセーブ 7. ディープソフトウェアスタンバイに遷移（AGT カウンタアンダーフローで解除）
引　数	- -
リターン値	-

3.6.2　r_captouch_low_power_scan ()

r_captouch_low_power_scan ()	
概　要	タッチ計測制御
宣　言	void r_captouch_low_power_scan (void)
説　明	タッチ計測を以下のように制御します。 1. タッチ計測を開始 2. AGT カウントを開始 3. ソフトウェアスタンバイモードに遷移。 4. AGT カウントを停止
引　数	- -
リターン値	-

3.6.3　r_captouch_low_power_disable_rtc ()

r_captouch_low_power_disable_rtc ()		
概　要	RTC 無効化	
宣　言	void r_captouch_low_power_disable_rtc (void)	
説　明	RTC 内のレジスタを初期化	
引　数	-	-
リターン値	-	

3.6.4　R_BSP_WarmStart ()

R_BSP_WarmStart()		
概　要	リセット要因、ディープソフトウェアスタンバイモード解除要因を検出	
宣　言	void R_BSP_WarmStart (bsp_warm_start_event_t event)	
説　明	リセット要因、ディープソフトウェアスタンバイモードの解除要因を検出します。	
引　数	bsp_warm_start_event_t event	ワームスタート・イベント BSP_WARM_SRART_RESET:　リセット直後 BSP_WARM_START_POST_CLOCK: クロック設定後 BSP_WARM_START_POST_C: C ランタイム初期化後
リターン値	-	

3.6.5　R_CAPTOUCH_LPM_Save ()

R_CAPTOUCH_LPM_Save ()		
概　要	スタンバイ SRAM セーブ	
宣　言	void R_CAPTOUCH_LPM_Save (void)	
説　明	タッチミドルウェア関連データをスタンバイ SRAM にセーブします。	
引　数	-	-
リターン値	-	

3.6.6　R_CAPTOUCH_LPM_Load ()

R_CAPTOUCH_LPM_Load ()		
概　要	スタンバイ SRAM ロード	
宣　言	void R_CAPTOUCH_LPM_Load (void)	
説　明	タッチミドルウェア関連データをスタンバイ SRAM からロードします。	
引　数	-	-
リターン値	-	

3.6.7　R_CAPTOUCH_LPM_IsPowerOnReset

R_CAPTOUCH_LPM_IsPowerOnReset ()		
概　要	パワーオンリセットの有無を返す	
宣　言	void R_CAPTOUCH_LPM_IsPowerOnReset (void)	
説　明	パワーオンリセットの有無を返します。	
引　数	-	-
リターン値	-	

4. フローチャート

4.1 qe_touch_main ()

フローチャートを示します。

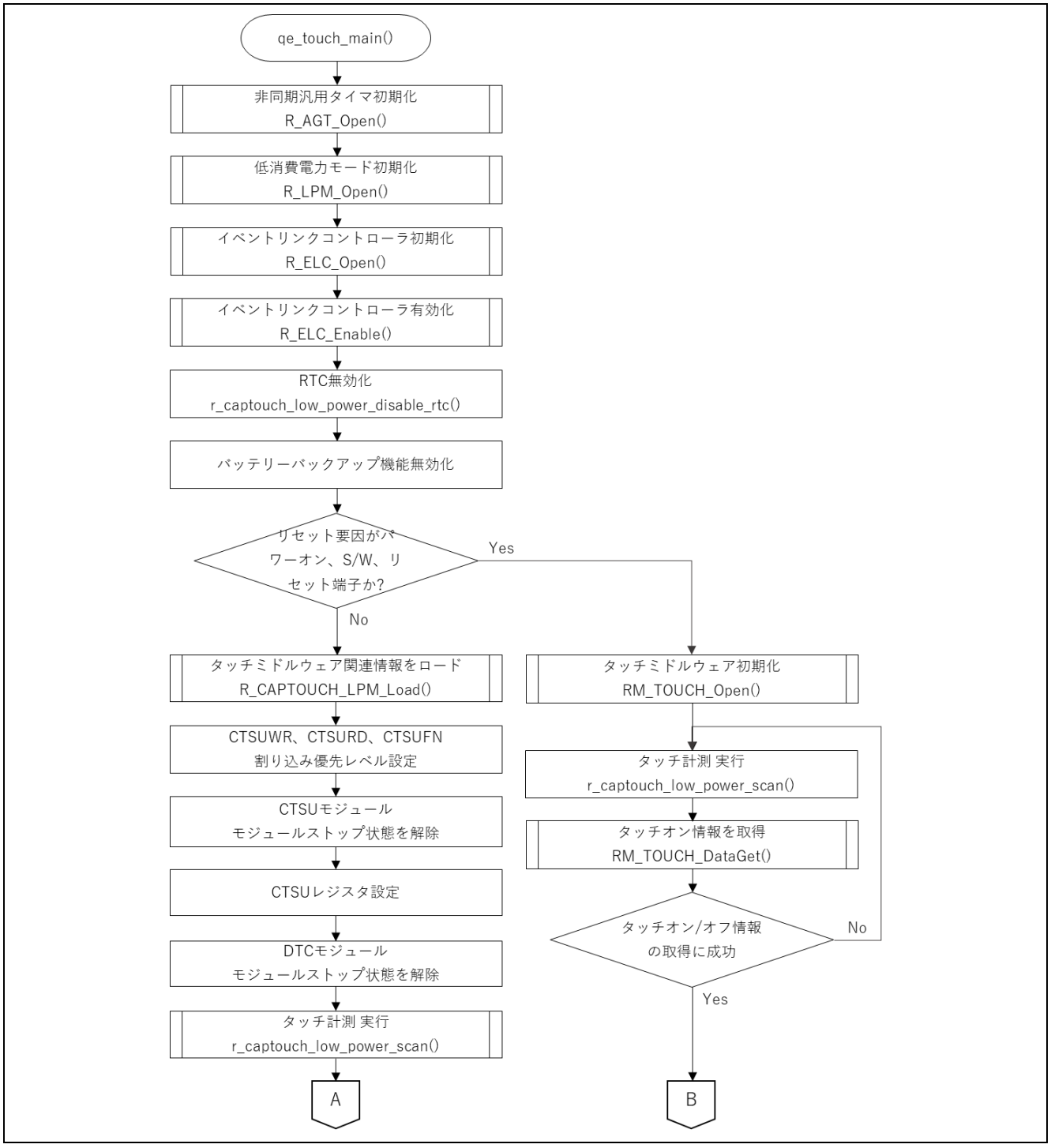

図 4-1 qe_touch_main ()（1/2）

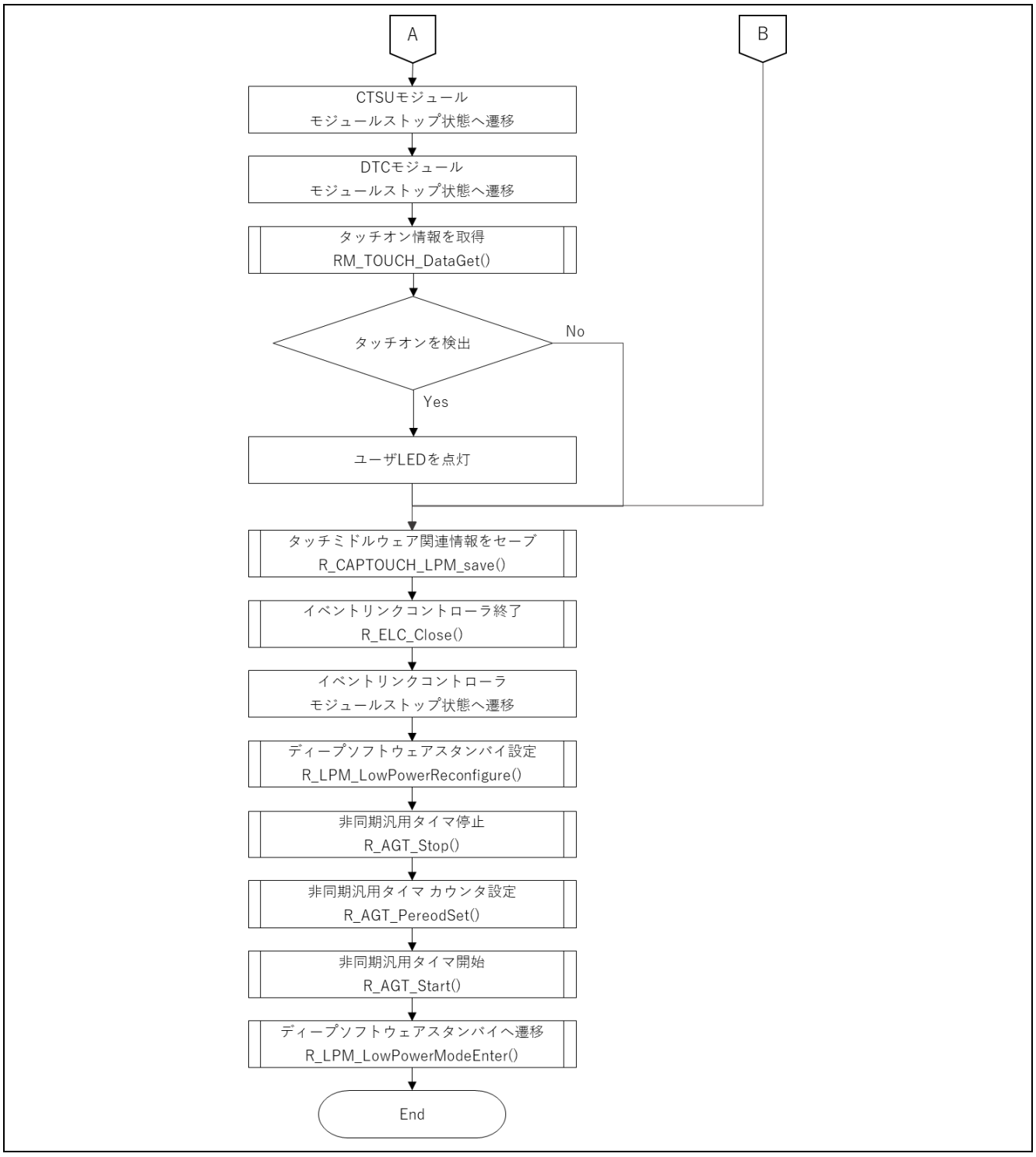

図 4-2　qe_touch_main ()（2/2）

4.2　r_captouch_low_power_scan ()

フローチャートを以下に示します。

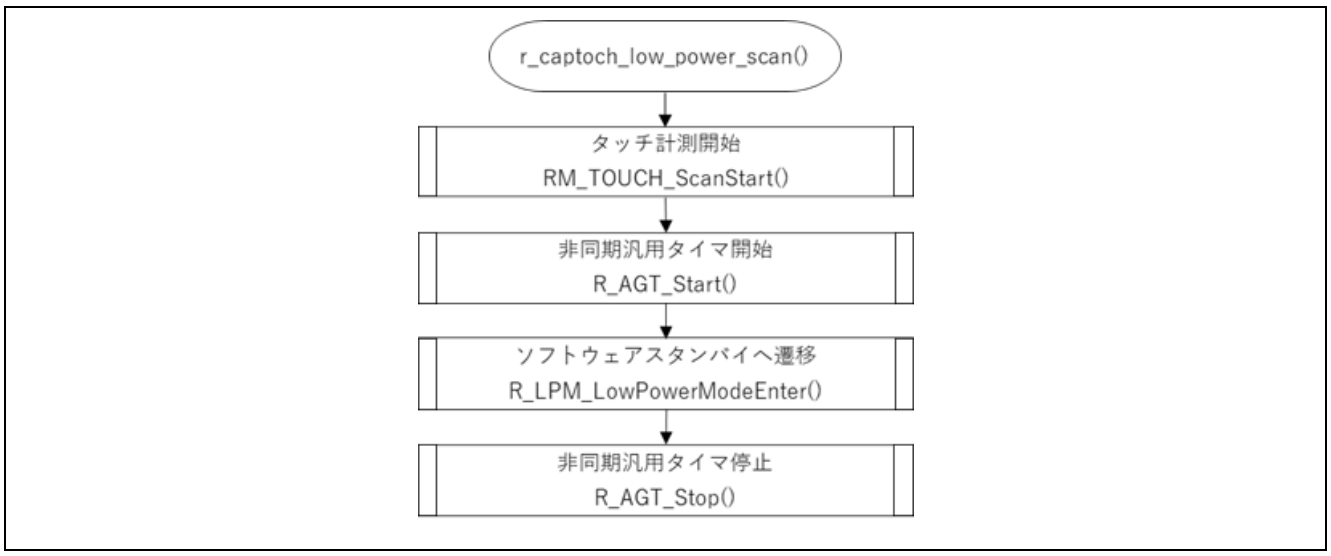

図 4-3 r_captouch_low_power_scan ()

4.3　r_captouch_low_power_disable_rtc ()

フローチャートを以下に示します。

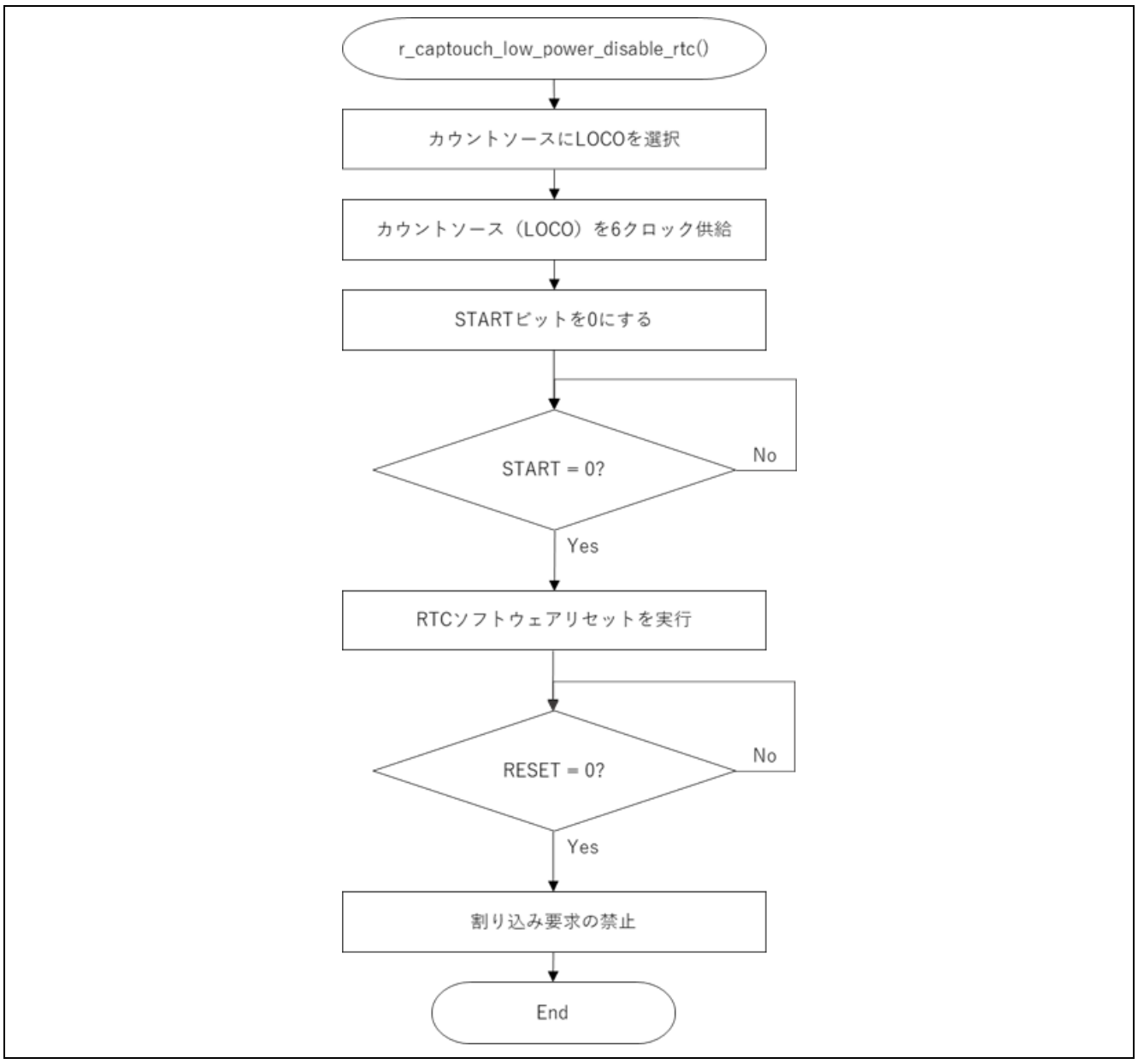

図 4-4　r_captouch_low_power_disable_rtc ()

4.4　R_BSP_WarmStart ()

フローチャートを以下に示します。

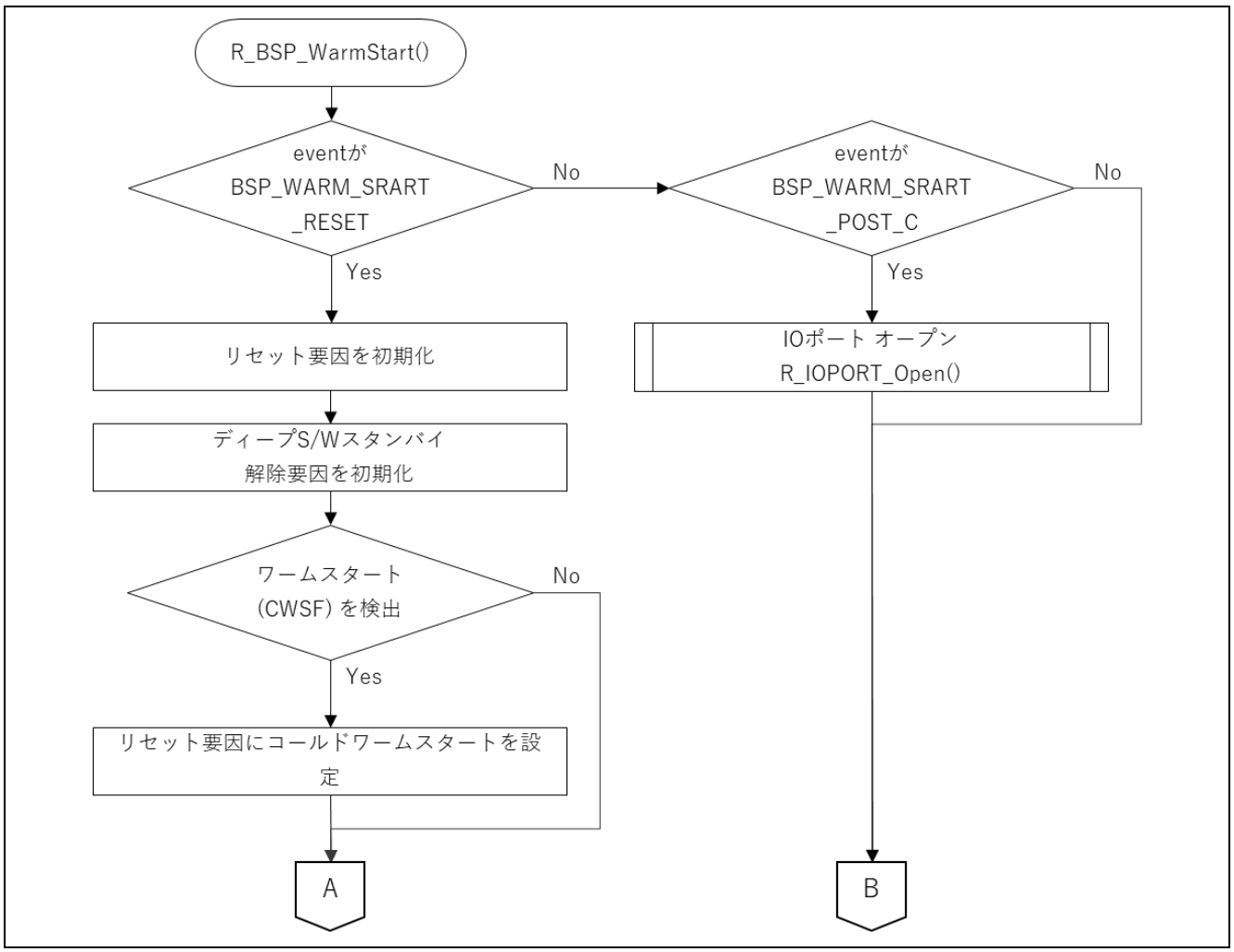

図 4-5　R_BSP_WarmStart ()（1/4）

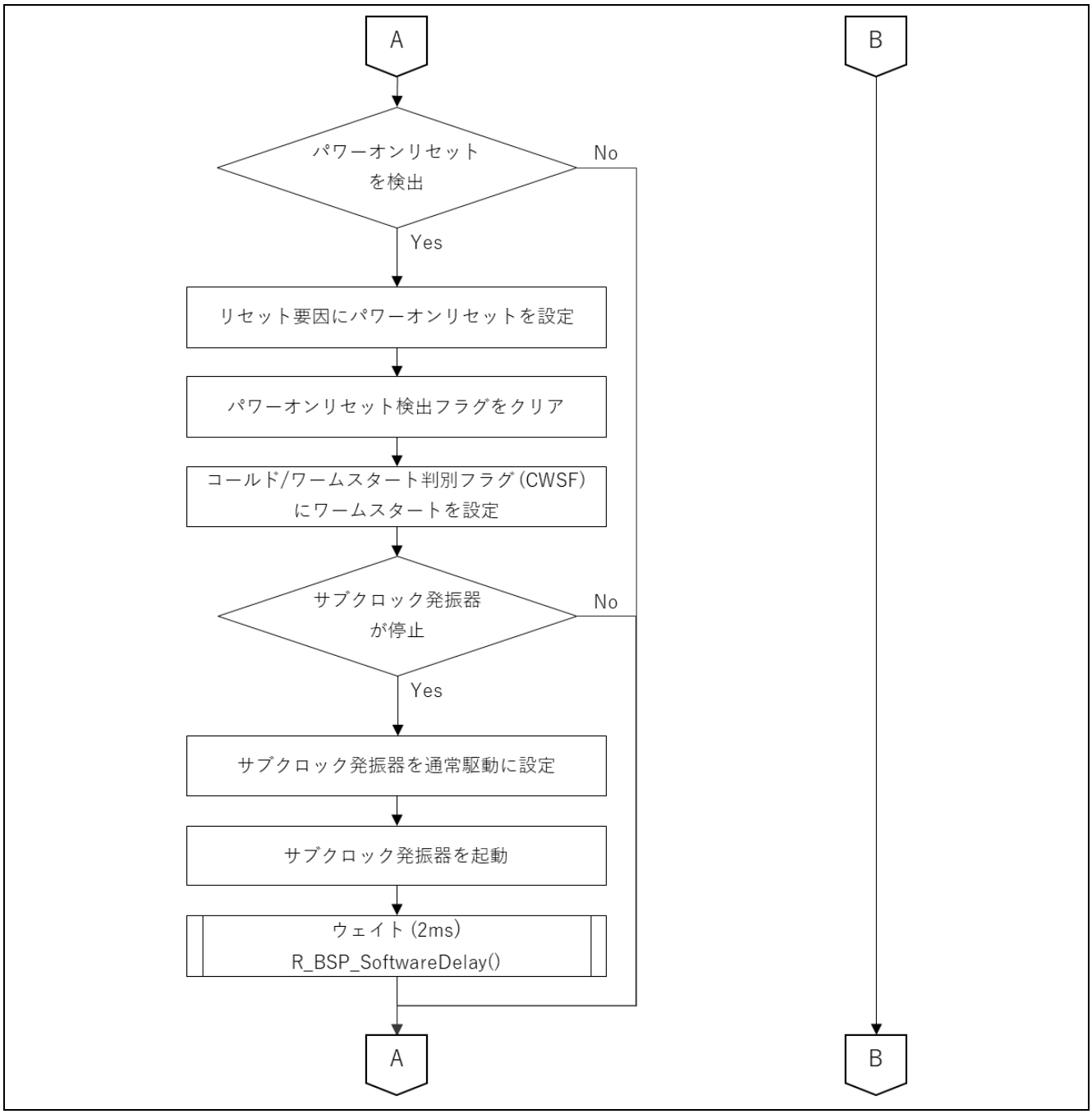

図 4-6 R_BSP_WarmStart ()（2/4）

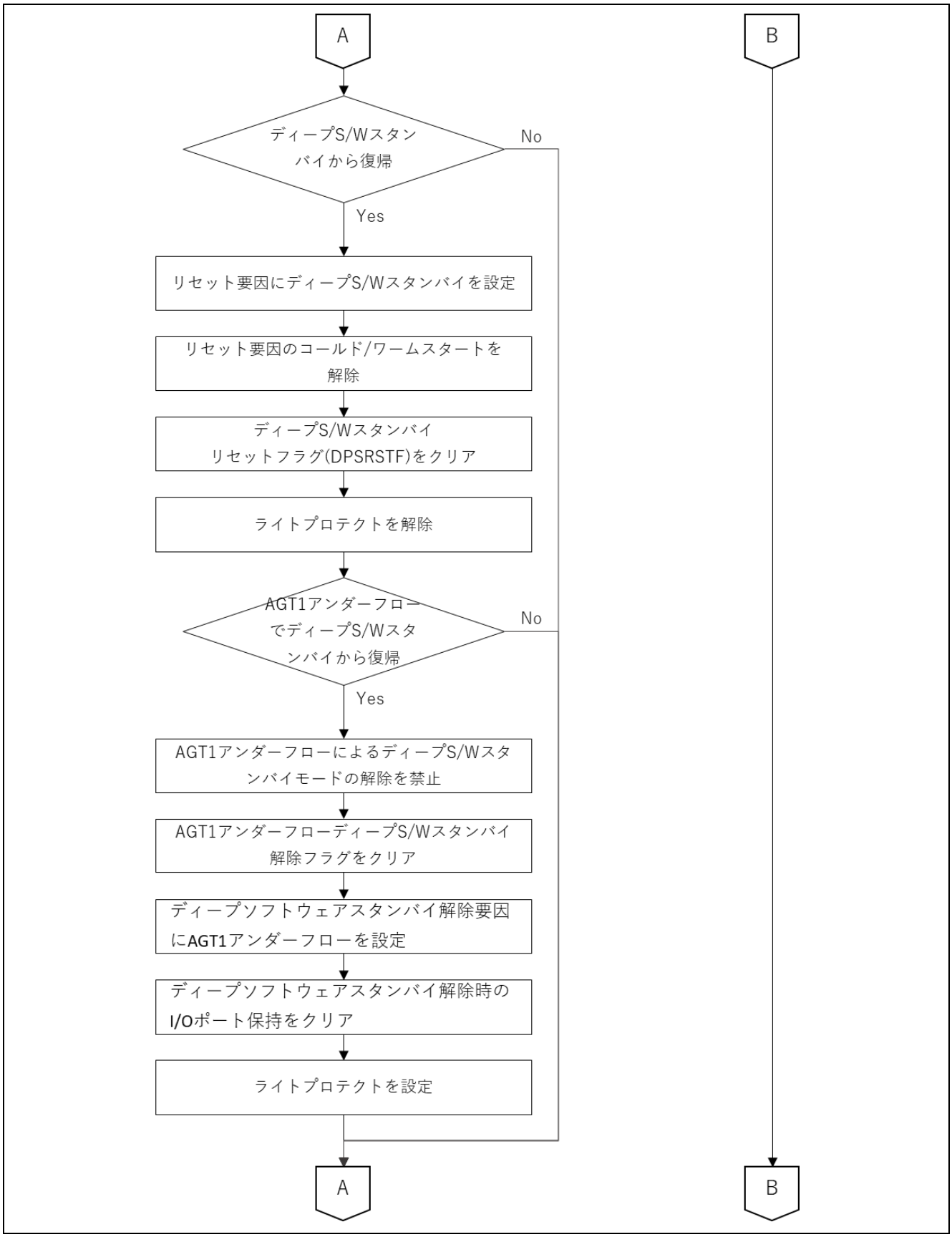

図 4-7 R_BSP_WarmStart ()（3/4）

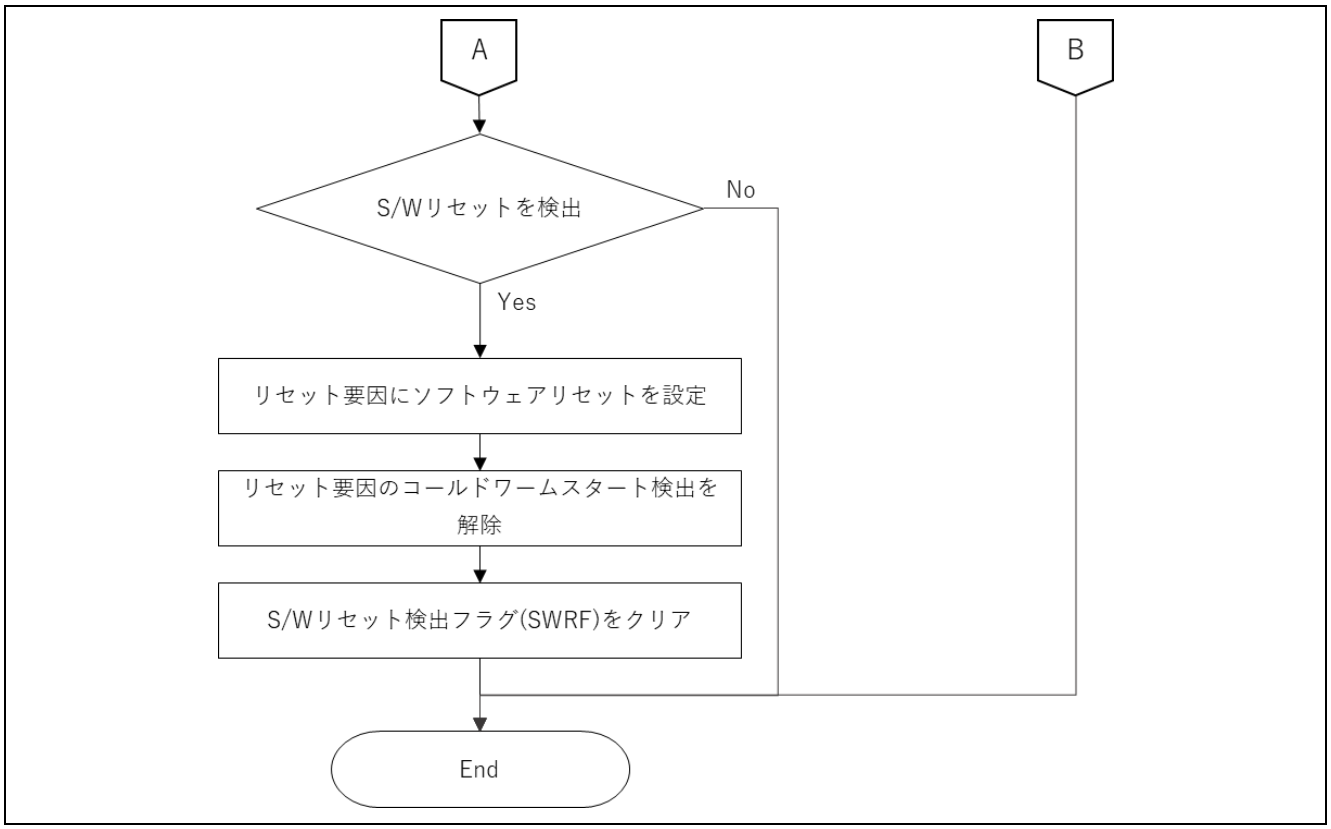

図 4-8 R_BSP_WarmStart ()（4/4）

4.5　R_CAPTOUCH_LPM_Save ()

フローチャートを以下に示します。

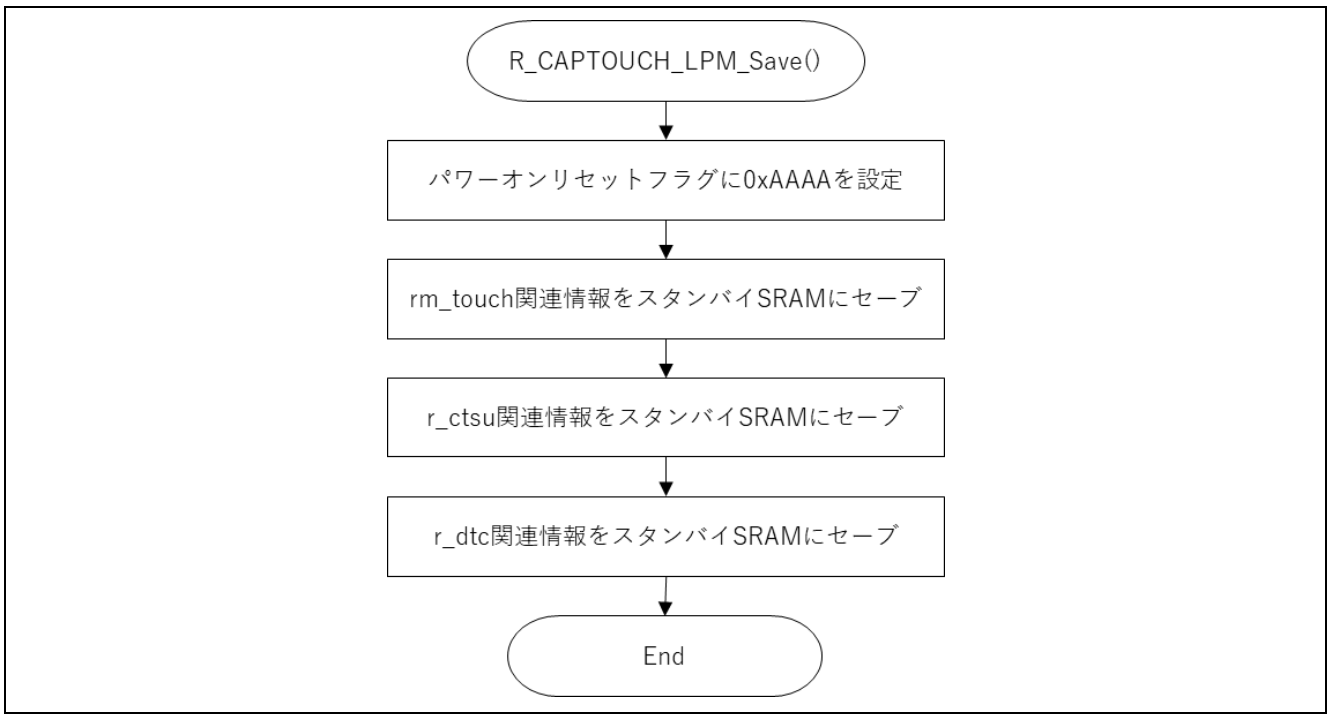

図 4-9　R_CAPTOUCH_LPM_Save ()

4.6　R_CAPTOUCH_LPM_Load ()

フローチャートを以下に示します。

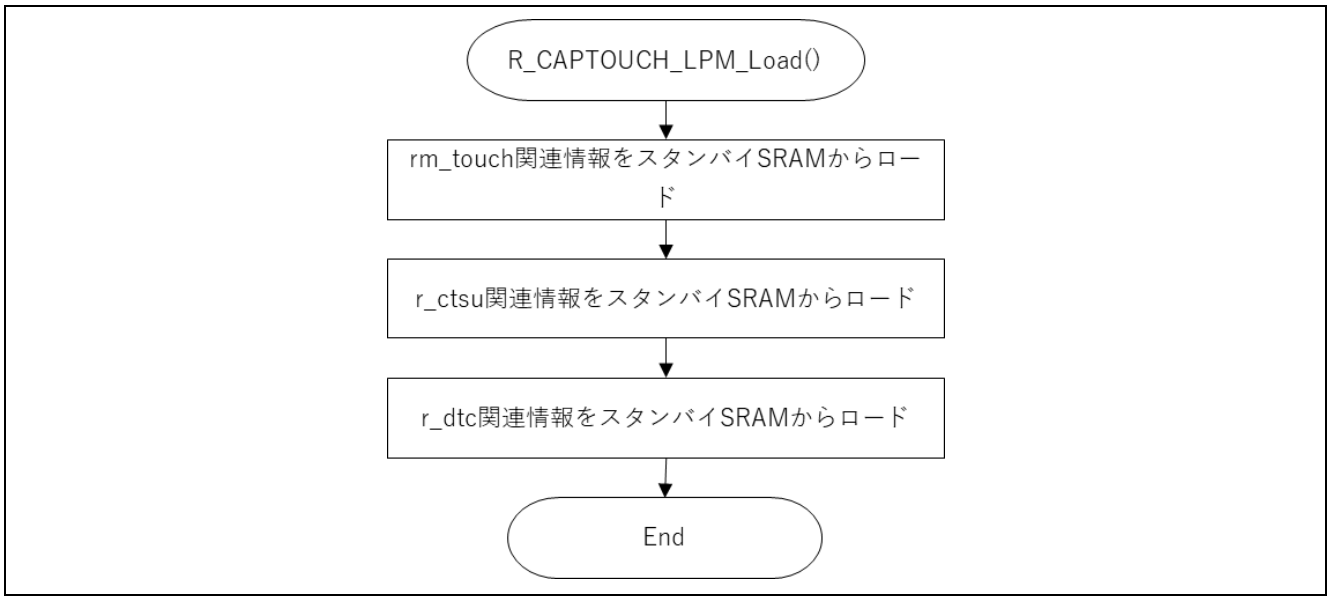

図 4-10　R_CAPTOUCH_LPM_Load ()

4.7　R_CAPTOUCH_LPM_IsPowerOnReset ()

フローチャートを以下に示します。

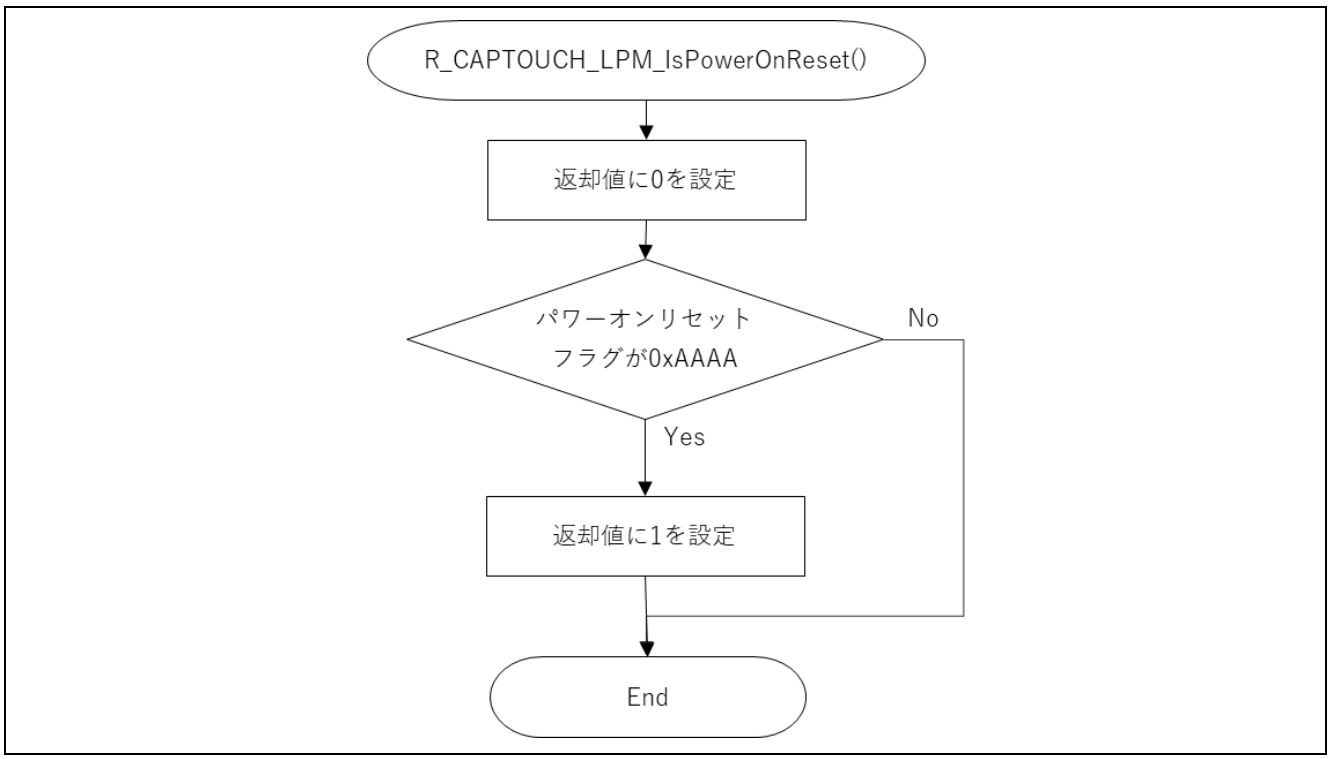

図 4-11　R_CAPTOUCH_LPM_IsPowerOnReset ()

5.　消費電力

サンプルコードにおける静電容量タッチ低消費電力動作は、下記の赤枠で示すシステム構成を想定しています。システムが待機状態にあり、静電容量タッチボタン（電源ボタン）のみを 100ms 周期で計測している状態です。

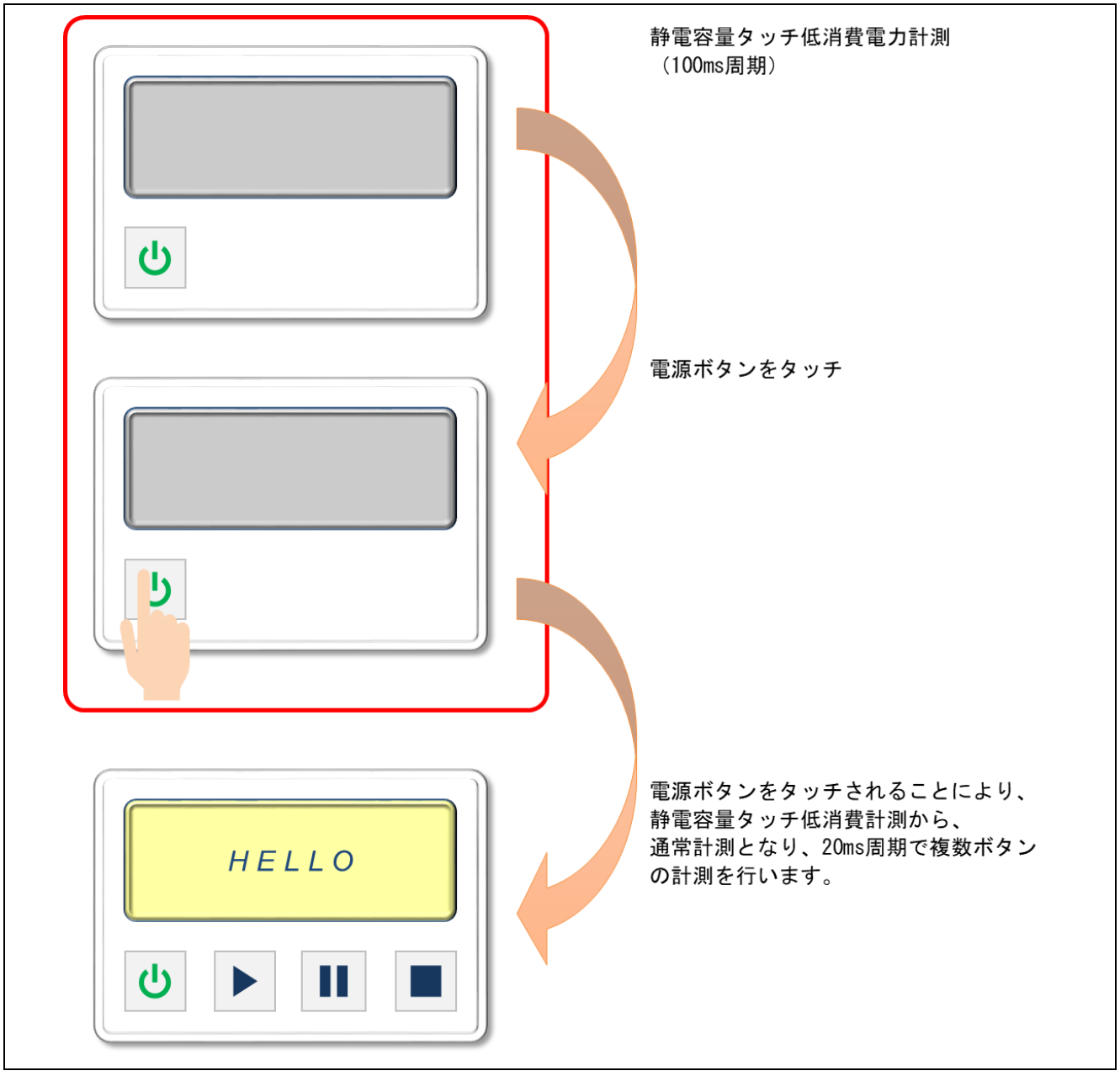

図 5-1　想定システム

5.1　動作条件

表 5-1 に動作条件を示します。

表 5-1　動作条件

項目	内容
CPU 動作周波数	20MHz 高速オンチップオシレータ（HOCO） 32kHz　低速オンチップオシレータ（LOCO）
システムクロック（ICLK）	20MHz
周辺クロックモジュール A（PCLKA）	20MHz
周辺クロックモジュール B（PCLKB）	20MHz
周辺クロックモジュール C（PCLKC）	20MHz
周辺クロックモジュール D（PCLKD）	20MHz
タッチ計測周期	100ms
センサドライブパルス周波数	2MHz
CTSU 計測タッチセンサ	TS9 を使用
CTSU 計測モード	自己容量マルチスキャンモード
CTSU 電源能力	通常出力
CTSU 高域ノイズ低減機能	ON
CTSU スペクトラム拡散分周	2MHz（0001b）
CTSU 計測回数	3

5.2　機器、ソフトウェア

消費電流を計測したときに使用した機器とソフトウェアを以下に示します。

表 5-2　機器、ソフトウェア一覧

種別	名称	用途
デジタルマルチメータ	KEITHLEY/DMM7510	消費電流を計測
安定化電源	KENWOOD/PA18-1.2A	RA6M2 Cap Touch CPU ボードに電源を供給
ソフトウェア	KEITHLEY/KickStart ソフトウェア	KEITHLEY/DMM7510 から消費電流の計測結果を取得し、ログファイルに出力する

5.3　RA6M2 Cap Touch CPU ボード・ジャンパ設定

消費電流計測向けの RA6M2 Cap Touch CPU ボードのジャンパ設定を以下に示します。

表 5-3　ジャンパ設定

位置	回路グループ	ジャンパ設定	用途
JP1	電源	オープン	消費電流計測
JP2	電源	1-2 ピン クローズ	DC ジャックから電源を供給

5.4　RA6M2 Cap Touch CPU ボード

RA6M2 Cap Touch CPU ボードの全面を以下に示します。

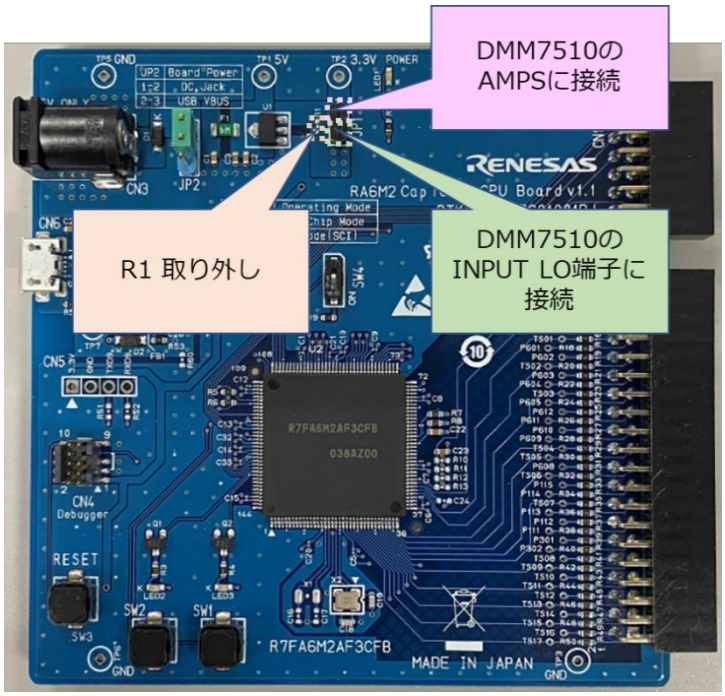

図 5-2　RA6M2 Cap Touch CPU ボード・前面

5.5　消費電流計測環境

消費電流計測を行った計測環境を以下に示します。

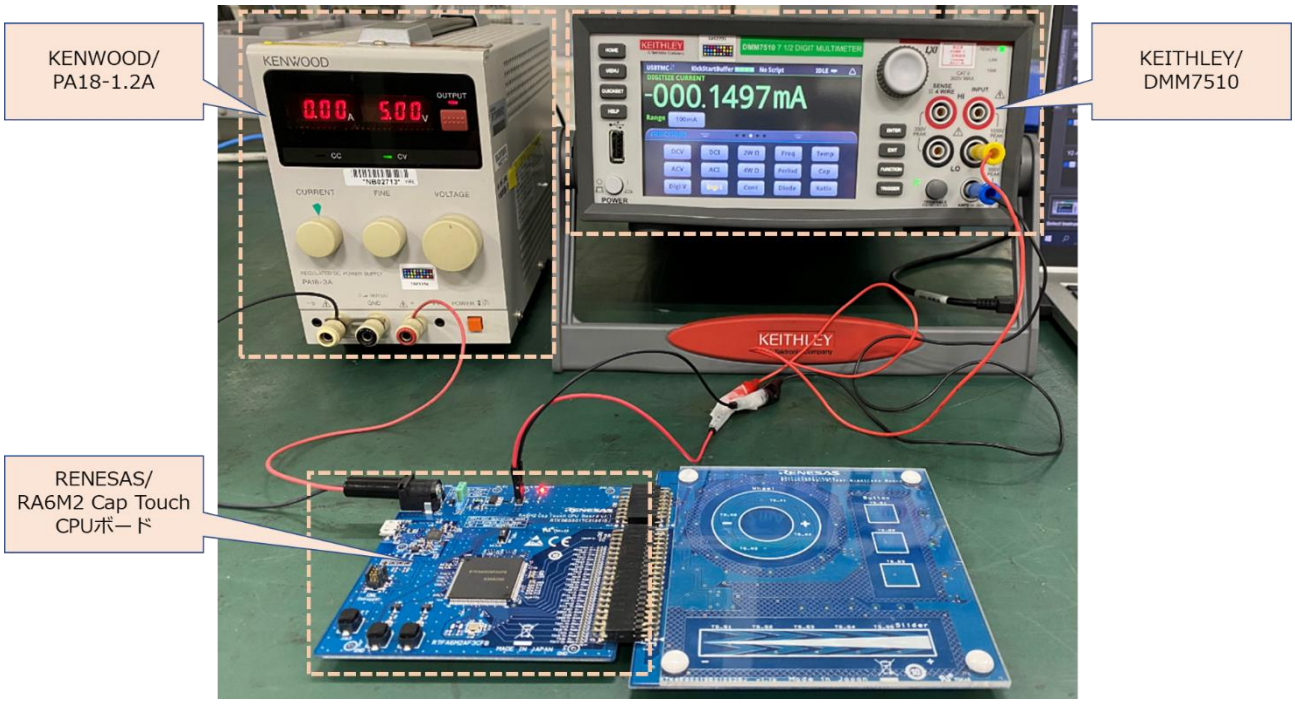

図 5-3　消費電流計測環境

5.6　消費電流計測設定

図 5-4 に、KEITHLEY/KickStart ソフトウェアの消費電流計測の設定を以下に示します。

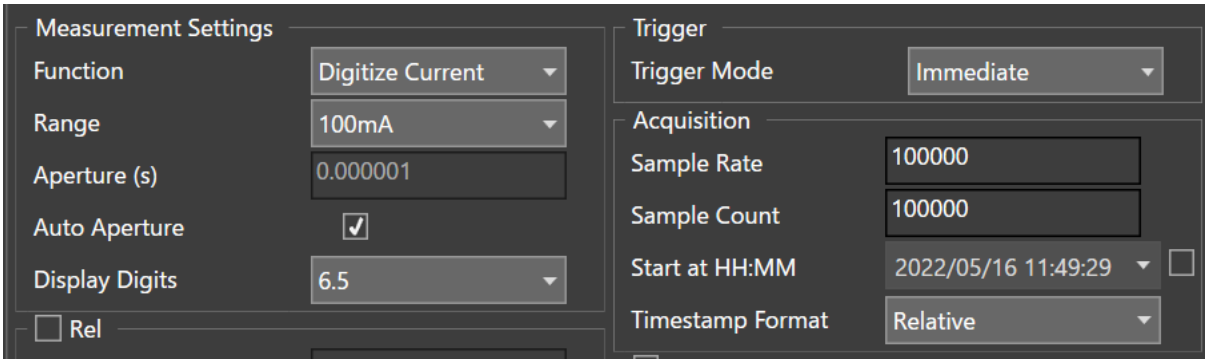

図 5-4　KEITHLEY/KickStart・消費電流計測設定

5.7　消費電流計測結果

図 5-5～図 5-6 に、CPU 動作モードがディープスタンバイモード、通常モード(MCU 起動、タッチ計測開始処理)、スヌーズモード（タッチ計測）、通常モード（タッチ計測終了処理、タッチオン/オフ判定処理）に遷移する際の消費電流波形を示します。

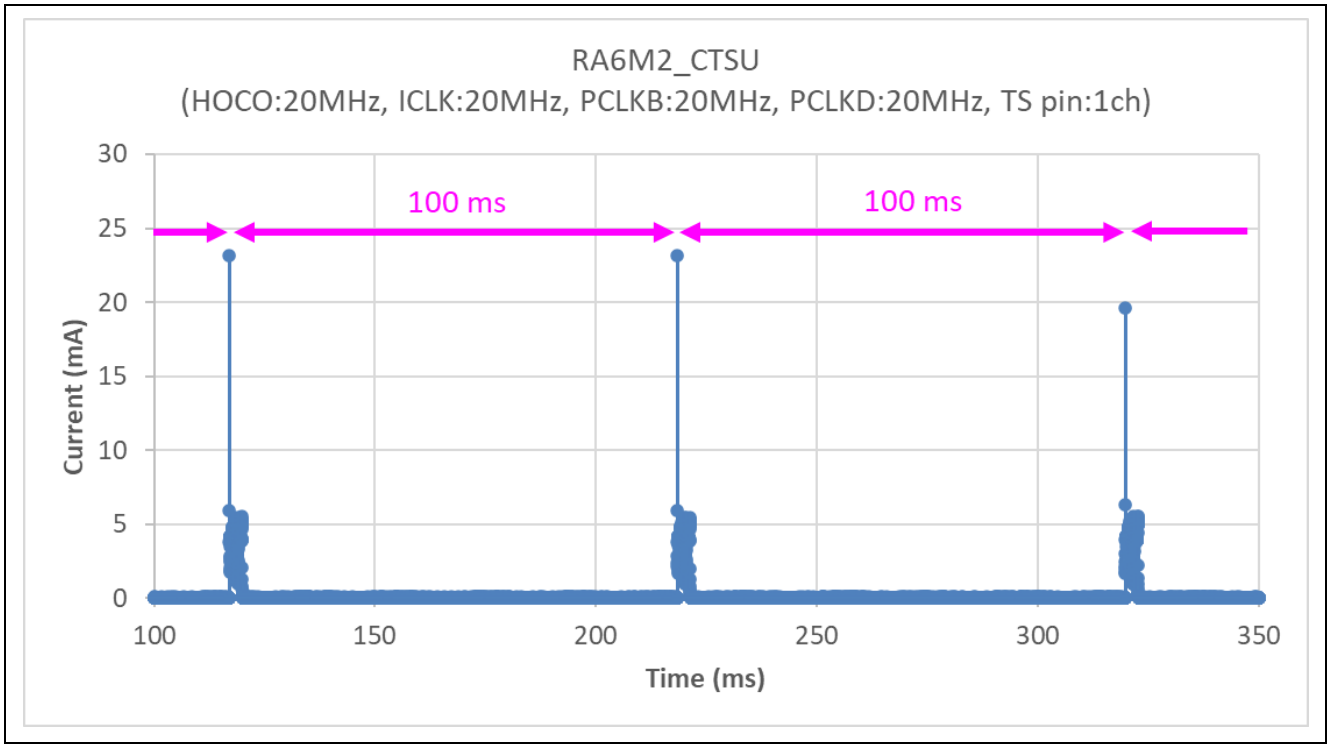

図 5-5　消費電流波形 TS 端子 1ch 計測（1/2）

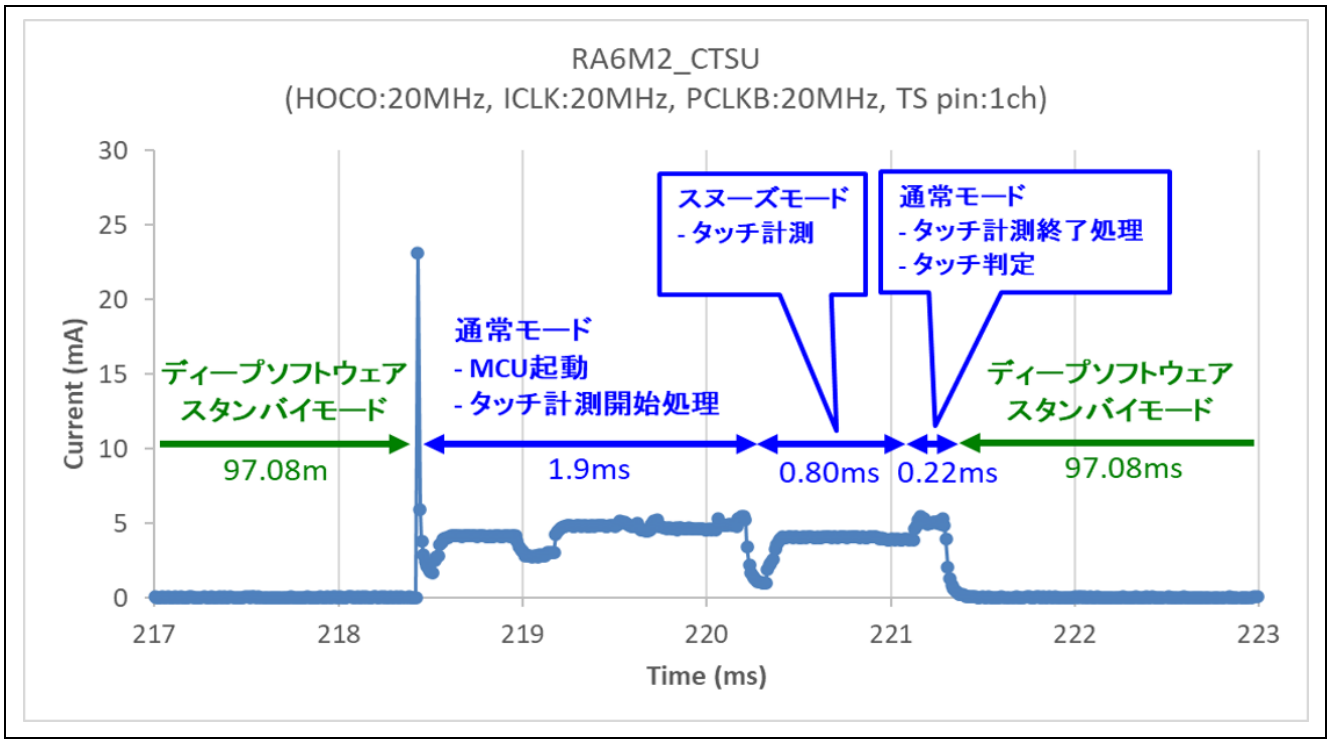

図 5-6　消費電流波形 TS 端子 1ch 計測（2/2）

5.8　消費電流算出結果

TS 端子 1 チャネルをタッチ計測周期 100 ms で測定した平均消費電流を図 5-7 に示します。

図 5-7 消費電流結果（TS 端子 1ch 計測）

タッチ計測周期 100 ms で測定した平均消費電流 = **153.322 µA**

6. 参考ドキュメント

ユーザーズマニュアル：ハードウェア

RA6M2 グループ ユーザーズマニュアル ハードウェア編（R01UH0885）

（最新版をルネサスエレクトロニクスホームページから入手してください。）

テクニカルアップデート／テクニカルニュース

（最新の情報をルネサスエレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：開発環境

（最新版をルネサスエレクトロニクスホームページから入手してください。）

ユーザーズマニュアル：RM6M2 グループ 静電容量タッチ評価システム

（最新版をルネサスエレクトロニクスホームページから入手してください。）

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Jun.30.22	-	初版発行

製品ご使用上の注意事項

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

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

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

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

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。