R8C/33T グループ

タッチ処理 API 説明（R8C/33T グループ）

R01AN0745JJ0100
Rev.1.00
2011.9.6

要旨

タッチパネルマイコン R8C/33T グループは、タッチ電極と人体の間に発生する浮遊容量を測定することで人体の接触を感知するハードウエア（タッチセンサーコントロールユニット、以下 SCU）を内蔵しています。

本仕様書は、タッチ処理用 API（Application Program Interface；以下 API）に関する外部仕様について記述したものです。

対象デバイス

R8C/33T グループ

目次

1. 概要

1.1 API 構成

本APIはR8C/33T・タッチ動作としてSCU計測、タッチ・ソフトウェア処理を制御するためのAPIとして「組み込みAPI」と、ユーザが設定、情報取得可能な「ユーザAPI」があります。
以下その説明を行います。

1.2 機能概要

本APIは以下に示す機能を持ちます。

- SCU割り込み処理機能
- SCU計測値の移動加算機能
- SCU計測開始制御
- タッチキーON/OFF判定機能
- ドリフト補正機能
- 初期キャリブレーション機能
- マルチタッチキャンセル機能
- ホイール・デコード機能
- スライダ・デコード機能
- タッチキーON/OFF結果のコールバック機能
- ホイール、スライダの位置データ・コールバック機能
- タッチ動作のスタート／ストップ制御
- 各電極に対するドリフト補正処理のON/OFF設定機能

2. ファイル構成

表 2-1に API で使用するファイルを示します。

表 2-1　ファイル一覧

項	ファイル名	機能概要
1	**touch_control.c**	組み込み API。タッチキーS/W 処理全般。
2	**touch_user_API.c**	ユーザ API。タッチの情報取得と設定処理。
3	**touch_interrupt.c**	SCU 計測終了時の割り込み処理。
4	**slider_control.c**	スライダのサンプルコード。
5	**wheel_control.c**	ホイールのサンプルコード。
6	**touch_control.h**	[touch_control.c]のヘッダファイル。
7	**touch_user_API.h**	[touch_user_API.c]のヘッダファイル。
8	**touch_interrupt.h**	[touch_interrupt.c]のヘッダファイル。
9	**slider_control.h**	[slider_control.c]のヘッダファイル。
10	**wheel_control.h**	[wheel_control.c]のヘッダファイル。

3. API 機能

表 3-1に設定が必要なカスタマイズ設定、表 3-2に主な組み込み API、表 3-3にユーザ API を示します。

表 3-1　カスタマイズ設定一覧

項	API 名	機能概要
4.1	**SCU_INV_NOISE**	インバーターノイズ対応の ON/OFF 切り替え
4.2	**MULTI_CANCEL**	マルチタッチキャンセル対応の ON/OFF 切り替え
4.3	**MULTI_START_CH**	マルチタッチキャンセル機能の開始電極 CH
4.4	**MULTI_END_CH**	マルチタッチキャンセル機能の終了電極 CH
4.5	**SLIDER_USE**	スライダ対応の ON/OFF 切り替え
4.6	**WHEEL_USE**	ホイール対応の ON/OFF 切り替え
4.7	**MAX_CH**	計測するタッチ電極 CH の最大値
4.8	**DF_TSIERn**　※1	タッチ電極／汎用端子の切り替え
4.9	**DF_CHxx_REF**　※2	タッチ ON/OFF 判定の基準値
4.10	**DF_CHxx_THR**　※2	タッチ ON/OFF 判断のしきい値
4.11	**DF_CHxx_HYS**　※2	タッチ OFF のヒステリシス
4.12	**DF_MSA_DATA**	連続 ON キャンセルの回数設定
4.13	**DF_ACCUMULATION**	連続 ON/OFF の回数設定 連続 ON：下位 8bit、連続 OFF：上位 8bit
4.14	**DF_DCI_DRIFT**	ドリフトの回数設定
4.15	**WORKBENCH_HEWSVR_ENABLE**	HEW Target Server で動作する Workbench の ON/OFF 切り替え
4.16	**SUPPORT_UART**	UART 経由で動作する Workbench の ON/OFF 切り替え

※1：n=0～2 で 3 カ所の設定が必要です。

※2：xx は最大 CH 数分を設定して下さい。

表 3-2　組み込み API 一覧

項	API 名	機能概要
5.1	**void TouchDtcInitialSet(void);**	SCU 計測に関係した DTC の初期設定
5.2	**void TouchDataInitial(void);**	タッチ動作に関連した RAM の初期化
5.3	**uint8_t CheckReadFlashData(void);**	データフラッシュからの読み出し関数
5.4	**void TouchDataInitial2(void);**	データフラッシュから Read した内容をベースに RAM の初期化
5.5	**void ScuInitial(void);**	SCUレジスタの初期化
5.6	**void ScuInterrup(void);**	SCU 計測が終了すると発生する割り込み。 割り込み内で計測終了状態を設定。
5.7	**void ScuMeasure(void);**	タッチ処理のメイン関数。
5.8	**void CheckWriteStatusFlashData(void);**	データフラッシュへの書き込み関数
5.9	**void FtAddMakeAve(void);**	移動加算処理
5.10	**uint8_t SetTouchSensor(void);**	SCU計測のスタート制御
5.11	**void MakeCthr(void);**	タッチ ON/OFF 判定のしきい値をセットする処理
5.12	**void MultiCancel(void);**	マルチタッチをキャンセルする処理　※3
5.13	**void OnOffJudgement(void);**	タッチ ON/OFF の判断処理
5.14	**void Slider(void);**	スライダ・デコード処理　※4
5.15	**void SWheel(void);**	ホイール・デコード処理　※5
5.16	**void CorrectSub(uint16_t s_dci1);**	ドリフト補正処理
5.17	**void MsrCalibration(void);**	基準値を設定する初期キャリブレーション

※3：マルチタッチキャンセル処理が不要ならば、組み込む必要がありません。

※4：スライダ処理が不要ならば、組み込む必要がありません。（タッチキーには不要です。）

※5：ホイール処理が不要ならば、組み込む必要がありません。（タッチキーには不要です。）

表 3-3　ユーザ API 一覧

項	API 名	機能概要
6.1	**TOUCH_ONOFF_STATUS_E GetTouchOnOff(void);**	タッチ ON/OFF データが有効かステータスを返す。
6.2	**TOUCH_ONOFF_STATUS_E GetWheelPosition(void);**	ホイールの位置データが有効かステータスを返す。※6
6.3	**TOUCH_ONOFF_STATUS_E GetSliderPosition(void);**	スライダの位置データが有効かステータスを返す。※7
6.4	**MODE_SCU_MEASURE_E SetScuMode(SCU_MODE_E mode);**	タッチ計測開始と停止の設定を行う。
6.5	**uint8_t SetScuDcen(DRIFT_ENABLE_E sw);**	各電極のドリフト補正処理 ON/OFF の設定を行う。

※6：ホイールの API（SWheel）が有効で使用しているときのみ、ご使用下さい。

※7：スライダの API（Slider）が有効で使用しているときのみ、ご使用下さい。

4. カスタマイズ設定

[touch_control.h]ファイル内にある下記の define を設定して下さい。

4.1 SCU_INV_NOISE

概要

SCU 計測を動作クロック 5MHz の通常計測か、20MHz のノイズ対策用の計測かを切り替える制御をします。主にインバーターをターゲットにしたノイズ対策で、SCU レジスタの設定を変更します。

レジスタに設定される詳細内容については「5.4 **void ScuInitial(void);**」を参照して下さい。

設定値

[#define SCU_INV_NOISE]無効　：　通常計測（システムクロック 20MHz に対して f4/5MHz の設定）

[#define SCU_INV_NOISE]有効　：　ノイズ対策計測（システムクロック 20MHz に対して f1/20MHz の設定）

4.2 MULTI_CANCEL

概要

2 つ以上のキー同時押し（マルチタッチ）に対して、キーON させない機能です。製品の仕様、水濡れ対策等で使用して下さい。対応させるキーCH については、下記の「MULTI_START_CH」、「MULTI_END_CH」で設定して下さい。

設定値

[#define MULTI_CANCEL]無効　：　マルチタッチキャンセル機能を無効。

[#define MULTI_CANCEL]有効　：　マルチタッチキャンセル機能を有効。

4.3 MULTI_START_CH

概要

マルチタッチキャンセル機能で、対象となるキーの開始 CH を設定して下さい。設定値は終了 CH である、「MULTI_END_CH」よりも小さな値を設定して下さい。

設定値

・0～16：R8C/33T

・0～20：R8C/3JT

4.4 MULTI_END_CH

概要

マルチタッチキャンセル機能で、対象となるキーの終了 CH を設定して下さい。設定値は開始 CH である、「MULTI_START_CH」よりも大きな値を設定して下さい。

設定値

・1～17：R8C/33T

・1～21：R8C/3JT

4.5 SLIDER_USE

概要

スライダ機能を ON/OFF させるスイッチです。設定を有効にして HEW のビルドを行うとプログラムがリンクされて動作可能な状態になります。本 API はサンプルコードなので、電極の使用数、解像度は仕様に合わせて設定して下さい。

設定値

[#define SLIDER_USE]無効　：　スライダ機能 OFF（使用不可能）

[#define SLIDER_USE]有効　：　スライダ機能 ON（使用可能）

4.6 WHEEL_USE

概要

ホイール機能を ON/OFF させるスイッチです。設定を有効にして HEW のビルドを行うとプログラムがリンクされて動作可能な状態になります。本 API はサンプルコードなので、電極は 4ch 固定、解像度は仕様に合わせて設定して下さい。

設定値

[#define WHEEL_USE]無効　：　ホイール機能 OFF（使用不可能）

[#define WHEEL_USE]有効　：　ホイール機能 ON（使用可能）

4.7 MAX_CH

概要

SCU 計測する CH（電極）数、タッチ ON/OFF、ドリフト補正処理等で使用する RAM サイズを最適にする制御をします。タッチで使用する CH の「番号＋１」の値を設定して下さい。

設定値

・0：タッチ電極無し

・1～17：タッチ電極有り（R8C/33T）、1～21：タッチ電極有り（R8C/3JT）

例）　CH0、CH1、CH3、CH4、**CH6** を電極で使用した場合　→　「7」設定

4.8 DF_TSIERn

概要

マイコンの端子をタッチ電極として使用するか、汎用端子として使用するか切り替えを制御します。データ内容は「タッチセンサ入力許可レジスタ n」と同じビット構成で設定して下さい。

設定値

・0：汎用端子
・1：タッチ電極

・DF_TSIER0

b7							b0
CH07	CH06	CH05	CH04	CH03	CH02	CH01	CH00

・DF_TSIER1

b7							b0
CH15	CH14	CH13	CH12	CH11	CH10	CH09	CH08

・DF_TSIER2

b7				b2		b0
		CH21 *	CH20 *	CH19 *	CH18 *	CH17	CH16

*：R8C/33T「0」設定

4.9 DF_CHxx_REF

概要

タッチ ON/OFF 判定に使用する「基準値」の初期値を設定して下さい。その際、使用するチャンネル分を設定して下さい。

（サンプルコードは初期キャリブレーションで基準値を決定していますので、本設定は未使用です。）

設定値

・0～65535（0x0000～0xFFFF）

4.10 DF_CHxx_THR

概要

タッチ ON/OFF 判定に使用する「ON/OFF 判定値」の初期値を設定して下さい。その際、使用するチャンネル分を設定して下さい。

設定値

・0～65535（0x0000～0xFFFF）

4.11 DF_CHxx_HYS

概要

タッチ ON/OFF 判定に使用する「ヒステリシス値」の初期値を設定して下さい。その際、使用するチャンネル分を設定して下さい。

設定値

・0～65535（0x0000～0xFFFF）

4.12 DF_MSA_DATA

概要

連続 ON キャンセル機能の、ON 判定回数を設定して下さい。

「連続 ON キャンセル機能」とは、ある一定時間タッチ ON だった場合、強制的にタッチ OFF 状態に設定する機能です。

設定値

・0：未使用

・1～255（0x01～0xFF）：動作許可。[DF_ONRESET_CNT]と掛け合わせた回数を設定。

4.13 DF_ACCUMULATION

概要

連続 ON/OFF 機能の、連続回数を設定して下さい。

「連続 ON/OFF 機能」とは、タッチキーON（OFF）を決定するのに、ある一定の連続回数で決定する機能です。

設定値

・ON：下位 8bit（0x00～0xFF）

・OFF：上位 8bit（0x00～0xFF）

4.14 DF_DCI_DRIFT

概要

ドリフト補正機能の、SCU 計測値加算回数を設定して下さい。

「ドリフト補正機能」とは、ある一定回数の SCU 計測値を平均化して、「基準値」を変更する機能です。周辺環境に基準値を追従させて、誤動作を防止します。

設定値

・0～65535（0x0000～0xFFFF）

4.15 WORKBENCH_HEWSVR_ENABLE

概要

Hew Target Server で動作する Workbench の ON/OFF スイッチです。設定を有効にして HEW のビルドを行うとプログラムがリンクされて動作可能な状態になります。

設定値

・[#define WORKBENCH_HEWSVR_ENABLE]無効：HEW Target Server で動作する Workbench 機能 OFF（使用不可）

・[#define WORKBENCH_HEWSVR_ENABLE]有効：HEW Target Server で動作する Workbench 機能 ON（使用可能）

4.16 SUPPORT_UART

概要

UART で動作する Workbench の ON/OFF スイッチです。設定を有効にして HEW のビルドを行うとプログラムがリンクされて動作可能な状態になります。

設定値

・[#define SUPPORT_UART]無効：UART で動作する Workbench 機能 OFF（使用不可）

・[#define SUPPORT_UART]有効：UART で動作する Workbench 機能 ON（使用可能）

5.　組み込み API 仕様

5.1　void TouchDtcInitialSet(void);

概要

DTCはSCU計測時に、計測値をレジスタからメモリへ自動転送するために使用します。ユーザはDTCに対してカスタマイズ値を設定するだけですが、下記の注意事項があります。

・ リピートエリアに指定したアドレスの初期値の下位8 ビットを“00h”にしてください。
（R8C/33T、3JT・ハードウェアマニュアル　「13.3.5 リピートモード」に記載）

表 5-1　DTC の主な設定内容

設定項目	内容
転送モード選択ビット	リピートモード
デスティネーションアドレス制御ビット	加算
1回の起動で転送するデータブロックサイズ	4バイト
DTCのデータ転送回数	ユーザ設定の最大チャンネル数（MAX_CH）
DTC割り込み要因	SCUデータ転送要求

条件

初期設定の処理群からコールして下さい。**ScuInitial**より前のタイミングでコールして下さい。

関数プロトタイプ

```
void TouchDtcInitialSet( void )
```

引数

無し

戻り値

無し

使用例

```
void main(void)
{
      :
   TouchDtcInitialSet();
   ScuInitial();
      :
         while(1){                          // Main Loop
                  :
                 ScuMeasure();
                  :
         }
}
```

5.2　void TouchDataInitial(void);

概要

タッチAPIに関連するRAMの初期値を設定します。

条件

初期設定の処理群からコールして下さい。**ScuInitial**より前のタイミングでコールして下さい。

関数プロトタイプ

```
void TouchDataInitial( void )
```

引数

無し

戻り値

無し

使用例

```
void main(void)
{
      :
  TouchDataInitial();
  ScuInitial();
      :
        while(1){                          // Main Loop
                :
                ScuMeasure();
                :
        }
}
```

5.3　void CheckReadFlashData(void);

概要

タッチAPIに関連するデータフラッシュのデータ内容をRAMにセットします。データフラッシュにデータが無い場合は、ROMテーブルの初期値をRAMにセットします。
設定しているデータ内容は下記の通りです。

表 5-2　データフラッシュの設定内容

設定項目	内容
Ch_para_Ref[MAX_CH]	ON/OFF判定の基準値
Ch_para_Thr[MAX_CH]	ON判定値
Ch_para_Hys[MAX_CH]	ヒステリシス値
Msa	連続ONキャンセル機能の連続回数
Mode	動作モード（ドリフト補正ON/OFF切り替え）
Acd	累積判定カウント（上位8bit：非タッチ、下位8bit：タッチ）
Dci	ドリフト補正間隔
chaxA_selectdata[3]	CHxA0／ CHxA1選択値（0: CHxA0、1: CHxA1）
Athr	マルチタッチキャンセル機能のON/OFFしきい値

条件

初期設定の処理群からコールして下さい。データフラッシュの内容を設定する**TouchDataInitial2**、**ScuInitial**より前のタイミングでコールして下さい。

関数プロトタイプ

```
uint8_t CheckReadFlashData( void )
```

引数

無し

戻り値

無し

（「5.3 **void CheckReadFlashData(void);**」の続き）

使用例

```
void main(void)
{
        :
    result = CheckReadFlashData();
    TouchDataInitial2();
    ScuInitial();
        :
            while(1){                           // Main Loop
                        :
                    ScuMeasure();
                        :
            }
}
```

5.4　void TouchDataInitial2(void);

概要

タッチAPIに関連するRAMの初期値を設定します。主にデータフラッシュで保持されていたデータ内容をRAMにセットします。

条件

初期設定の処理群からコールして下さい。データフラッシュの内容を読み込む**CheckReadFlashData**の後、さらに**ScuInitial**より前のタイミングでコールして下さい。

関数プロトタイプ

```
void TouchDataInitial2( void )
```

引数

無し

戻り値

無し

使用例

```
void main(void)
{
      :
  result = CheckReadFlashData();
  TouchDataInitial2();
  ScuInitial();
      :
        while(1){                          // Main Loop
                  :
                ScuMeasure();
                  :
        }
}
```

5.5　void ScuInitial(void);

概要

SCUレジスタの初期値を設定します。SCU計測はクロック5MHzで動作する「通常動作」と、20MHzで動作する「ノイズ対策動作」の2種類を用意してあり、[touch_control.h]のヘッダファイルで切り替え可能になっています。

表 5-3　SCU の主な設定内容

設定項目	内容（通常計測）	内容（ノイズ対策計測）
カウントソース	F4（CPUクロック20MHzに対して5MHz）	F1（20MHz）
SCU割り込み	許可	許可
Pre計測	無し	無し
ランダム計測	無し	無し
多数決計測	無し	無し
SCU計測トリガ	ソフトウェアトリガ	ソフトウェアトリガ
区間1	128サイクル	128サイクル
区間2	1サイクル	8サイクル
区間3	1サイクル	4サイクル
区間4	1サイクル	1サイクル
区間5	1サイクル	スキップ（無し）
区間6	1サイクル	6サイクル
モード選択	スキャンモード	スキャンモード
チャンネル設定	ユーザ設定値（MAX_CH）	ユーザ設定値（MAX_CH）
転送先アドレス	Scudata	Scudata
セカンダリカウンタ	7回	32回
割り込み優先レベル	レベル1	レベル1

条件

初期設定の処理群からコールして下さい。**SetTouchSensor**でSCU計測を開始する前のタイミングでコールして下さい。

関数プロトタイプ

```
void ScuInitial( void )
```

引数

無し

戻り値

無し

（「5.4 **void ScuInitial(void);**」の続き）

使用例

```
void main(void)
{
      :
  ScuInitial();
  SetTouchSensor();
      :
        while(1){                          // Main Loop
                  :
                ScuMeasure();
                  :
        }
}
```

5.6　void ScuInterrupt(void);

概要

SCU計測終了時に発生する割り込みです。「SCU計測終了」の内部モードを設定します。
正常ならSCU計測中モード（MD_SCU_RUN）から発生して、SCU計測終了モード（MD_SCU_FINISH）を設定します。ドライバが意図しないタイミングで割り込みが発生した場合は、エラー処理を行います。
具体的にはSCU停止モード（MD_SCU_STOP）から発生した場合は、モード保持させます。その他の場合にはSCU計測を再スタートさせます。

条件

SCU計測終了時に割り込みが発生します。再度SCU割り込みを発生させるためには「SIF」SCU割り込み要求フラグのクリアが必要です。

関数プロトタイプ

```
void ScuInterrupt ( void )
```

引数

無し

戻り値

無し

使用例

```
#pragma		INTERRUPT	ScuInterrupt
void ScuInterrupt ( void )
{
	if( md_scu_measure == MD_SCU_RUN ){
		md_scu_measure = MD_SCU_FINISH;
	}else
	if( md_scu_measure == MD_SCU_STOP ){
		md_scu_measure = MD_SCU_STOP;
	}else{
		md_scu_measure = MD_SCU_READY;
		SetTouchSensor();
	}
	scucr0_addr.bit.scue = OFF;
	scufr_addr.bit.sif = OFF;
}
```

5.7　void ScuMeasure(void);

概要

SCU計測終了後に、タッチー連の処理をコールします。主に以下のAPIを制御します。

- FtAddMakeAve　／　移動加算処理
- SetTouchSensor　／　SCU計測の起動
- MakeCthr　／　ON/OFF判定値の設定
- MultiCancel　／　マルチタッチキャンセル機能
- OnOffJudgement　／　ON/OFF判定処理
- Slider　／　スライダ・デコード処理
- SWheel　／　ホイール・デコード処理
- CorrectSub　／　ドリフト補正処理
- MsrCalibration　／　基準値を設定する初期キャリブレーション

条件

メインの動作タイミングでコールされて、SCU計測の終了時のみ処理を実行します。オーバーフローフラグ（プライマリカウンターがオーバー）がセットされている時は処理せずにSCU計測を再スタートさせます。初期キャリブレーション中にオーバーフローフラグが設定されていると、キャリブレーション自体を最初から再スタートさせます。

関数プロトタイプ

```
void ScuMeasure( void )
```

引数

無し

戻り値

無し

使用例

```
void main(void)
{
        while(1){
                :
                ScuMeasure();
                :
        }
}
```

5.8　void CheckWriteStatusFlashData(void);

概要

タッチAPIに関連する、ある特定のRAMにセットしたデータをデータフラッシュへ書き込みます。設定しているデータ内容は「5.3　uint8_t CheckReadFlashData(void)」と同じです。

条件

メインの動作タイミングでコールされて、Workbenchから要求があった時のみ処理を実行します。

関数プロトタイプ

```
void CheckWriteStatusFlashData( void )
```

引数

無し

戻り値

無し

使用例

```
void main(void)
{
        while(1){
                :
                CheckWriteStatusFlashData();
                :
        }
}
```

5.9　void FtAddMakeAve(void);

概要

SCU計測値を汎用RAMへ保持させて移動加算処理を行います。保持させる際には一番古いデータの上書して、4回分の加算を実行します。加算値は「計測値」として、ON/OFF判定、ドリフト補正処理で使用します。

条件

ScuMeasureからSCU計測終了時のみコールされます。**ScuMeasure**がコールされたら一番最初に本APIをコールして下さい。

関数プロトタイプ

```
void FtAddMakeAve( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        FtAddMakeAve();
         :
}
```

5.10　uint8_t SetTouchSensor(void);

概要

SCU計測のスタートを設定します。S/W内部モードをSCU計測中（MD_SCU_RUN）に設定します。

条件

S/W内部モードがSCU計測準備OKのモード（MD_SCU_READY ）の時のみ処理を実行します。SCUスタートが未設定の場合、エラーを返します。

関数プロトタイプ

```
uint8_t SetTouchSensor( void )
```

引数

無し

戻り値

0：SCU スタート未設定

1：SCU スタート設定

使用例

```
void ScuMeasure( void )
{
        :
    md_scu_measure = MD_SCU_READY;
    SetTouchSensor();
        :
}
```

5.11　void MakeCthr(void);

概要

スライダー、ホイール、同時押しキャンセル等のアプリケーションで使用するDcount（基準からの差分値）と、タッチキーのON/OFF判定をするためのCthr（しきい値）を算出します。

条件

ScuMeasureからSCU計測終了時のみコールされます。**FtAddMakeAve**の後でコールして下さい。

関数プロトタイプ

```
void MakeCthr( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        :
        FtAddMakeAve();
        MakeCthr ();
        :
}
```

5.12　void MultiCancel(void);

概要

複数の同時押しタッチ（マルチタッチ）をONさせない機能です。複数のキーをグループで確認し、ON/OFFしきい値を調整する事で実現させます。オプション扱いのサンプルコードなので、どの電極を使用するか設定が必要です。Difineで有効にすると使用可能となります。

条件

ScuMeasureからSCU計測終了時のみコールされます。**MakeCthr**の後、**OnOffJugment**の前でコールして下さい。

関数プロトタイプ

```
void MultiCancel( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        :
    MakeCthr ();
    MultiCancel ();
    OnOffJugment();
        :
}
```

5.13　void OnOffJudgement(void);

概要

タッチキーのON/OFF判定を行います。ONからOFF状態へ判断する時は「ヒステリシス値」も一緒に判断します。仮判定したON/OFF状態に対して、「累積判定カウント（ON/OFF）」、「連続ONキャンセル」の判断を加えて最終判断をします。結果はBDATAに、1チャンネル1ビットで設定します。

条件

ScuMeasureからSCU計測終了時のみコールされます。**MekeCthr**の後でコールして下さい。

関数プロトタイプ

```
void OnOffJudgement( void )
```

引数

無し

戻り値

無し

使用例

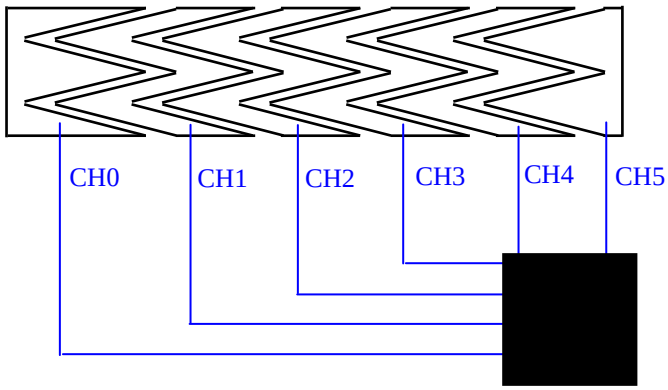
```
void ScuMeasure( void )
{
        :
        MakeCthr ();
        :
        OnOffJudgement();
        :
}
```

5.14　void Slider(void);

概要

スライダのデコードを行います。本APIはサンプルコードのため、スライダで使用する電極CH、CH数、分解能を設定することでカスタマイズが可能です。

分解能は2種類のデータがあり、基礎解像度の【sldposition_raw】、ユーザ設定解像度の【sldposition_r】があります。



条件

ScuMeasureからSCU計測終了時のみコールされます。タッチON中は動作しない仕様条件により、**OnOffJudgement**の後でコールして下さい。

[slider_control.c], [slider_control.h]ファイルを追加して下さい。

関数プロトタイプ

```
void Slider( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure ( void )
{
        :
    OnOffJudgement();
        :
    Slider();
        :
}
```

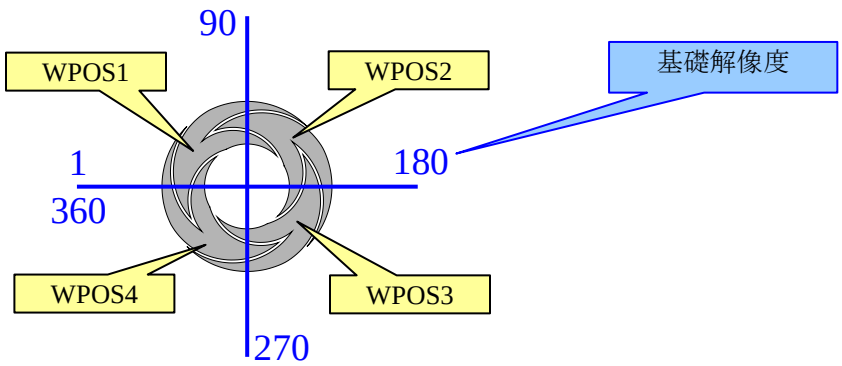
5.15　void SWheel(void);

概要

スライダー電極形状のホイール・デコード処理を行います。WPOS1～WPOS4の電極とマイコンのCH端子は、任意で指定することが可能です。[**wheel_control.h**]ファイルで定義して下さい。

分解能は2種類のデータがあり、基礎解像度の【diff_angle_4ch】、ユーザ設定解像度の【wheel_sw】があります。

S/W処理、電極形状の詳細内容についてはアプリケーションノートを参照して下さい。



条件

ScuMeasureからSCU計測終了時のみコールされます。タッチON中は動作しない仕様条件により、**OnOffJudgement**の後でコールして下さい。

[wheel_control.c], [wheel_control.h]ファイルを追加して下さい。

関数プロトタイプ

```
void SWheel ( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        :
    OnOffJudgement();
        :
    SWheel();
        :
}
```

5.16　void CorrectSub(uint16_t s_dci1);

概要

ドリフト補正処理を制御します。ユーザ設定の回数分、計測値を加算して平均値を算出します。算出された平均値は最新の基準値（Nref）として設定されます。

条件

ScuMeasureからSCU計測終了時のみコールされます。タッチON中は動作しない仕様条件により、**OnOffJudgement**の後でコールして下さい。

関数プロトタイプ

```
void CorrectSub(uint16_t s_dci1)
```

引数

・s_dci1：平均化のデータ取得回数

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        :
        OnOffJudgement();
        CorrectSub(s_dci);
        :
}
```

5.17　void MsrCalibration(void);

概要

初期キャリブレーション処理として、基準値（Nref）の初期値を設定します。タッチ計測値は通常のSCU計測を4回動作させ、2回分の計測値を組み合わせて低い値を基準値とします。設定した基準値は「FtAddMakeAve」で使用する移動加算処理の仮データ（4回分）として設定します。

条件

ScuMeasureからSCU計測終了時でキャリブレーションモード時のみコールされます。

関数プロトタイプ

```
void MsrCalibration( void )
```

引数

無し

戻り値

無し

使用例

```
void ScuMeasure( void )
{
        :
        if (meascal == 0 ) {                    // 通常計測、キャリブレーションモードの判断
                :
        }else{                                  // <- キャリブレーションモード
                MsrCalibration();
        }
        :
}
```

6. ユーザ API 仕様

6.1 TOUCH_ONOFF_STATUS_E GetTouchOnOff(void);

概要

アプリケーションよりコールが行われると、電極毎のタッチON/OFFデータ【BDATA】をチェック可能かどうか、ステータスを返します。チェック可能なステータス【DATA_OK】が得られたら【BDATA】でON/OFF情報を確認して下さい。

条件

コールされた時点のステータスを返します。計測停止、初期キャリブレーション中、オーバーフローエラー発生時はエラーステータスを返します。

関数プロトタイプ

```
TOUCH_ONOFF_STATUS_E GetTouchOnOff( void );
```

引数

無し

戻り値

・0x00：DATA_OK／タッチのON/OFFデータのチェック可。下記【BDATA】を参照下さい。
・0xFF：STOP_MODE／タッチ計測停止ステータス
・0xFE：OVER_MODE／オーバーフローエラー発生中
・0xFD：CALIB_MODE／初期キャリブレーション中

■ BDATA／0：タッチON、1：タッチOFF

・BDATA[0]

b7							b0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

・BDATA[1]

b7							b0
CH15	CH14	CH13	CH12	CH11	CH10	CH09	CH08

・BDATA[2]

b7							b0
－	－	CH21	CH20	CH19	CH18	CH17	CH16

（R8C/33T　：　CH18～21　「0」固定）

使用例

```
void main(void)
{
                :
        If( DATA_OK == GetTouchOnOff() ){
                Check_touch_onoff();                    //      check touch on/off process
        }
                :
}
```

6.2　TOUCH_ONOFF_STATUS_E GetWheelPosition(void);

概要

アプリケーションよりコールされると、ホイールの位置データ【diff_angle_4ch】、【wheel_sw】をチェック可能かどうか、ステータスを返します。チェック可能なステータス【DATA_OK】が得られたら【diff_angle_4ch】、【wheel_sw】で位置情報を確認して下さい。

条件

コールされた時点のステータスを返します。計測停止、初期キャリブレーション中、オーバーフローエラー発生時はエラーステータスを返します。

関数プロトタイプ

```
TOUCH_ONOFF_STATUS_E GetWheelPosition( void );
```

引数

無し

戻り値

・0x00：DATA_OK／ホイールの位置データのチェック可。下記【diff_angle_4ch】、【wheel_sw】を参照下さい。
・0xFF：STOP_MODE／タッチ計測停止ステータス
・0xFE：OVER_MODE／オーバーフローエラー発生中
・0xFD：CALIB_MODE／初期キャリブレーション中

●diff_Angle_4ch　（0：データ無し、1～360：データ有り）

電極 4ch に対する角度データ

●wheel_sw　（-32767～+32767）

[0]を起点として右回り、左回りで角度を加算した値

使用例

```
void main(void)
{                   :
        If( DATA_OK == GetWheelPosition() ){
                Check_wheel_position();         //      check wheel position process
        }
                :
}
```

6.3 TOUCH_ONOFF_STATUS_E GetSliderPosition(void);

概要

アプリケーションよりコールされると、スライダの位置データ【sldposition_raw】、【sldposition_r】をチェック可能かどうか、ステータスを返します。チェック可能なステータス【DATA_OK】が得られたら【sldposition_raw】、【sldposition_r】で位置情報を確認して下さい。

条件

コールされた時点のステータスを返します。計測停止、初期キャリブレーション中、オーバーフローエラー発生時はエラーステータスを返します。

関数プロトタイプ

```
TOUCH_ONOFF_STATUS_E GetSliderPosition( void );
```

引数

無し

戻り値

・0x00：DATA_OK／スライダの位置データのチェック可。下記【sldposition_raw】、【sldposition_r】を参照下さい。
・0xFF：STOP_MODE／タッチ計測停止ステータス
・0xFE：OVER_MODE／オーバーフローエラー発生中
・0xFD：CALIB_MODE／初期キャリブレーション中

●sldposition_raw　（0xFFFFFFFF:データ無し、1～360：データ有り）

電極 6ch に対する位置データ（例えば 360 分割）

●sldposition_r　（0xFFFF：データ無し、1～36：データ有り）

上記[sldposition_raw]を 1/10 で丸め込み

使用例

```
void main(void)
{
        If( DATA_OK == GetSliderPosition() ){
                Check_slider_position();//        check slider position process
        }
                :
}
```

6.4　MODE_SCU_MEASURE_E SetScuMode(SCU_MODE_E mode);

概要

タッチ動作の開始と停止の設定を行います。設定した結果、その時の内部モードを返します。

条件

タッチ動作停止時に「開始要求」を出すと、タッチ動作を開始させます。タッチ動作中に「停止要求」を出すと、H/WのSCU計測、タッチデコードを中止します。また「開始要求」を出すと、計測処理を継続します（処理無し）。

関数プロトタイプ

```
MODE_SCU_MEASURE_E SetTscuMode( TSCU_MODE_E mode );
```

引数

SCU動作モード

・0x00：MDRQ_SCU_STOP（タッチ動作停止要求）
・0x01：MDRQ_SCU_START（タッチ動作開始要求）

戻り値

SCU 動作のステータス

・0x00：MD_SCU_STOP（タッチ処理・停止中）
・0x01：MD_SCU_READY（SCU 計測・開始 OK 状態）
・0x02：MD_SCU_RUN（SCU 計測中）
・0x03：MD_SCU_FINISH（SCU 計測完了状態でデコード待ち。）

使用例

```
void main(void)
{
        MODE_SCU_MEASURE_E scu_mode;
                :
        scu_mode = SetScuMode(MDRQ_SCU_START);
                :
}
```

6.5　uint8_t SetScuDcen(DRIFT_ENABLE_E sw);

概要

各電極のドリフト補正処理ON/OFFの設定を行います。オプションスイッチが用意されていて、全電極に対してON/OFFを容易に設定が可能になっています。

条件

設定した時点から、次のドリフト補正処理で設定を反映させます。ドリフト補正の動作をOFFからONへ切り替えたチャンネルは、ドリフト補正の平均値を算出させる保持データは全てクリアされ、最初からの計測となります。

関数プロトタイプ

```
uint8_t SetScuDcen( DRIFT_ENABLE_E sw );
```

引数

・ドリフト補正処理のON/OFF設定値　（0：動作OFF　1：動作ON）
・DC_NON（0x00000000）：全てのチャンネルに対して動作OFFを設定
　→　0x00000000を設定
・DC_ALL（0xFFFFFFFF）：全てのチャンネルに対して動作ONを設定
　→　Nhen.DWORD（レジスタに設定したイネーブルビット）を設定

戻り値

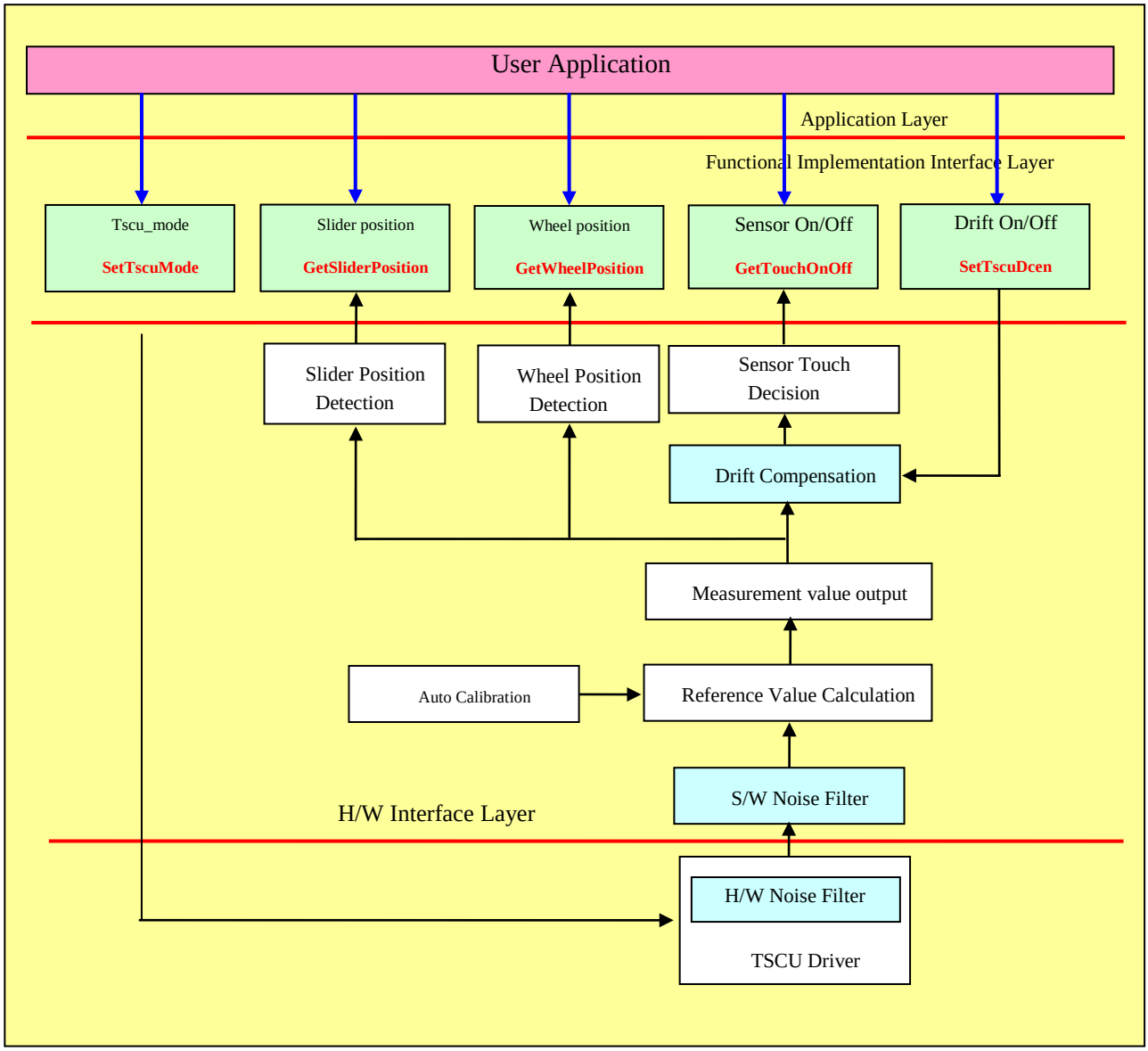
無し

使用例

```
void main(void)
{
                :
    SetScuDcen(DC_ALL);
                :
}
```

7. タッチ API 階層

主なAPIの階層を下記に示します。



8. タッチ実動作例

8.1 【ScuMmeasure】概略フローチャート

SCU 計測終了後、ソフトウェアでタッチを処理する内容について、下記の概略フローチャートに示します。

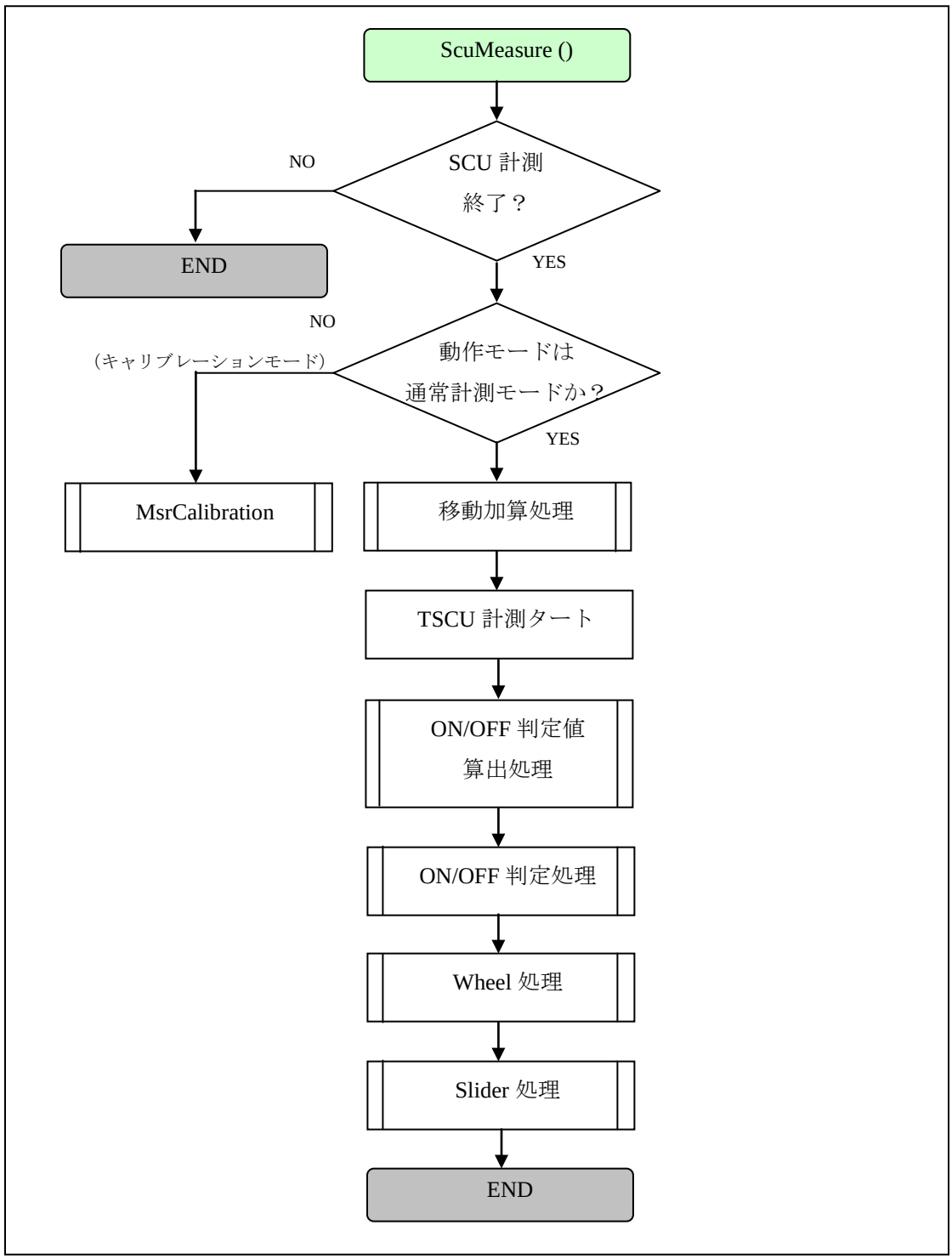


図 8-1　タッチ処理の概略フローチャート

8.2　タッチ処理タイミングチャート

以下にタッチに関連する SCU 計測、SCU 割り込み、タッチ処理（メイン処理）のタイミングチャートを示します。

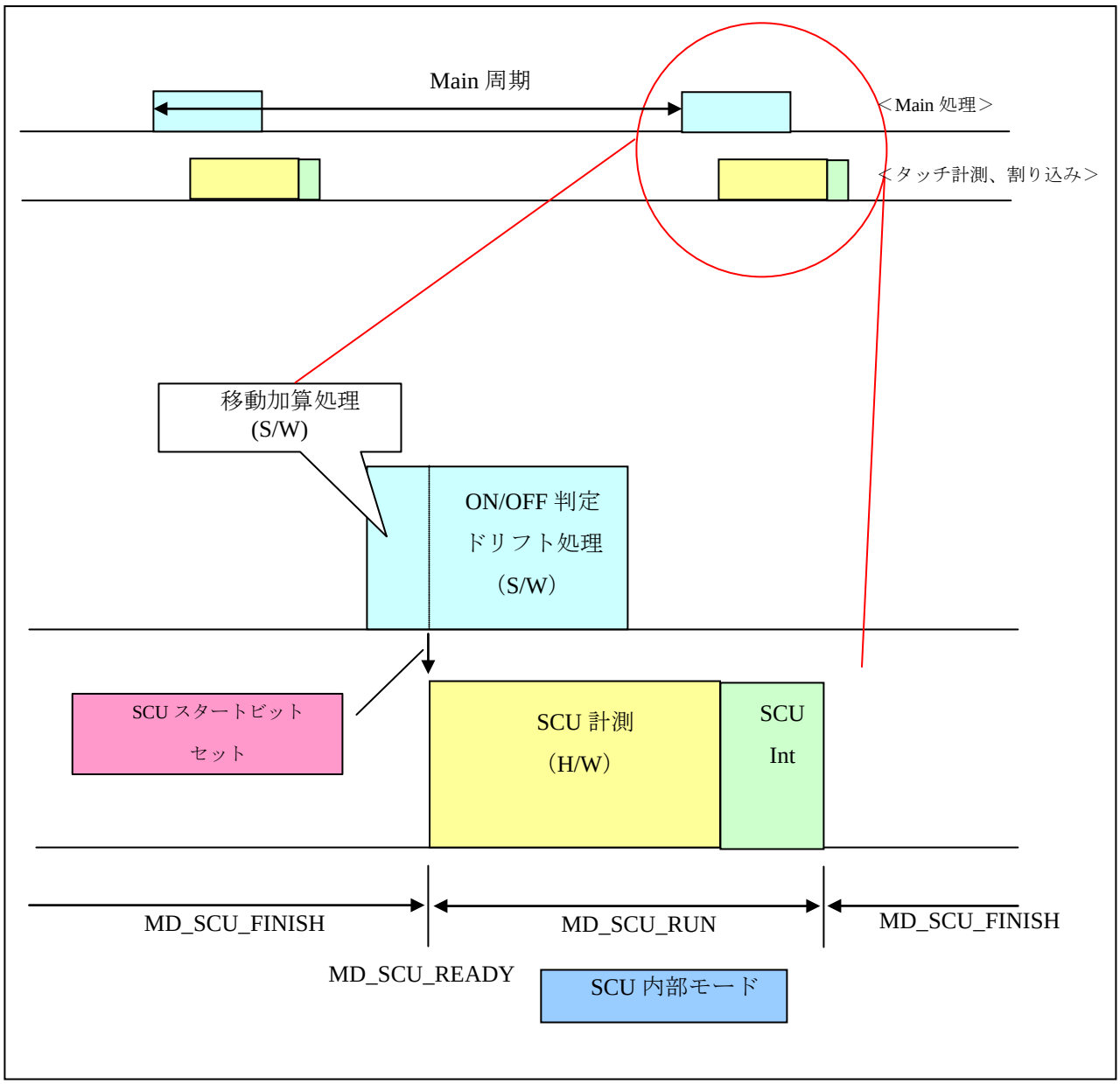


図 8-2　タッチ処理のタイミングチャート

8.3　タッチ処理・内部モードの説明

① メイン処理内で SCU 計測が終了した【MD_SCU_FINISH】モードを確認後、「移動加算処理」を実行。
　　↓
② 「移動加算処理」で SCU 計測値を退避させたら、次の SCU 計測スタートが出来るように【MD_SCU_READY】モード設定。
　　↓
③ 【MD_SCU_READY】モードを確認後、SCU スタートビット“Scustrt”をセットし、【MD_SCU_RUN】モードに設定。　→**SCU 計測（H/W）開始**
　　↓
④ 「ON/OFF 判定」、「ドリフト補正処理」等の一連のタッチ処理を行う。
　　↓
⑤ SCU 計測終了後、SCU 割り込み発生。割り込み内で計測終了の【MD_SCU_FINISH】モードに設定。

・ ③はユーザーの任意のタイミングで変更可能。（例：定周期のタイマ割り込み）
　→　上記設定例は S/W 処理と H/W 計測を同時で行う、最短なタイミング。

図 8-3　タッチ実動作例

ホームページとサポート窓口

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

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

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.9.6	—	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、本文を参照してください。なお、本マニュアルの本文と異なる記載がある場合は、本文の記載が優先するものとします。

1. 未使用端子の処理

   【注意】未使用端子は、本文の「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。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。

2. 電源投入時の処置

   【注意】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. リザーブアドレスのアクセス禁止

   【注意】リザーブアドレスのアクセスを禁止します。

   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレスがあります。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。

4. クロックについて

   【注意】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。

5. 製品間の相違について

   【注意】型名の異なる製品に変更する場合は、事前に問題ないことをご確認下さい。

   同じグループのマイコンでも型名が違うと、内部メモリ、レイアウトパターンの相違などにより、特性が異なる場合があります。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。

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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おりますが、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、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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. 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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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その総株主の議決権の過半数を直接または間接に保有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。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

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

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

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

Colophon 1.0