R8C/38T-A Group

Touch API Reference (R8C/38T-A Group)

R01AN0744EJ0100
Rev.1.00
May 23, 2013

Summary

Touch panel microcomputer R8C/3xT-A group builds hardware (TSCU: sensor control unit) that perceives the contact of the human body by measuring the stray capacity generated between the touch electrode and the human body into.

This specifications described the external specification concerning API(Application Program Interface) for the touch processing.

Target device

R8C/38T-A group

Contents

1. Summary

1.1 Touch API Overview

Touch API is comprised of "Base API" and "User API". "Base API" controls TSCU measurement and Judgement process for touch in R8C/38T-A Group. "User API" supports the acquisition of information and the settings by user.

1.2 Touch API function

Touch API functions are as follows.

- TSCU interrupt process
- Moving addition value of count from TSCU measurement calculation
- TSCU measurement startup
- Judgement process for touch or not
- Drift correction
- Automatic calibration
- Multi Touch Canceller
- Wheel position detection
- Slider position detection
- Getting touch position on touch key
- Getting touch position on slider/wheel
- Start/Stop of TSCU measurement
- Drift correction setting

2. Source & Header Files

The source files and the header files constituting Touch API are as follows.

Table 2-1 Source & Header Files List

No	File name	Remarks
1	touch_control.c	This file defines Base API for Touch key.
2	touch_user_API.c	This source file defines User API.
3	touch_interrupt.c	This file defines TSCU interrupt process.
4	slider_control.c	This file defines Base API for Slider position detection.
5	wheel_control.c	This file defines Base API for Wheel position detection.
6	touch_control.h	This file is a header file for touch_control.c.
7	touch_user_API.h	This file is a header file for touch_user_API.c.
8	touch_interrupt.h	This file is a header file for touch_interrupt.c.
9	slider_control.h	This file is a header file for slider_control.c.
10	wheel_control.h	This file is a header file for wheel_control.c.

3. Touch API List

Table 3-1 Macro definition List

Chapter.	Macro name	Remarks
4.1	TSCU_INV_NOISE	Conditional compilation to switch the countermeasure against the inverter noise for TSCU
4.2	MULTI_CANCEL	Conditional compilation to build Multi Touch Canceller
4.3	MULTI_START_CH	Start channel of Multi Touch Canceller
4.4	MULTI_END_CH	End channel of Multi Touch Canceller
4.5	SLIDER_USE	Conditional compilation to build a Slider module
4.6	WHEEL_USE	Conditional compilation to build a Wheel module
4.7	MAX_CH	Maximum channel number
4.8	DF_TSIERn *1	Initial value for TSCU Input Enable Registers
4.9	DF_CHxx_REF *2	Initial value of Reference count value
4.10	DF_CHxx_THR *2	Initial value of Threshold count value for judgement of touch or not
4.11	DF_CHxx_HYS *2	Initial value of Hysteresis value of the threshold count
4.12	DF_MSA_DATA	Initial value of Maximum successive ON count
4.13	DF_ACCUMULATION	Initial value of Accumulated judgement count
4.14	DF_DCI_DRIFT	Initial value of Drift correction interval
4.15	WORKBENCH_HEWSVR_ENABLE	Conditional compilation to build a control module for communication with Workbench using HewTargetServer
4.16	SUPPORT_UART	Conditional compilation to build a control module for communication with Workbench using UART

*1. Certainly set all DF_TSIERn. (n = 0, 1, 2)
*2. xx = “00” - the biggest numerical value of Touch CH.

Table 3-2 Base API List

Chapter	API name	Remarks
5.1	void TouchDtcInitialSet(void);	Initialization of DTC registers
5.2	void TouchDataInitial(void);	Initialization of RAM related to touch operation
5.3	uint8_t CheckReadFlashData(void)	Reading function from DATA FLASH
5.4	void TouchDataInitial2(void);	RAM is initialized based on the data obtained from the DATA FLASH
5.5	void TscuInitial(void);	Initialization of TSCU registers
5.6	void TscuInterrup(void);	TSCU Interrupt handling function
5.7	void TscuMeasure(void);	Judgement for touch or not control
5.8	void CheckWriteStatusFlashData(void);	Writing function to DATA FLASH
5.9	void FtAddMakeAve(void);	Moving addition value of count calculation
5.10	uint8_t SetTouchSensor(void);	TSCU measurement boot control
5.11	void MakeCthr(void);	Threshold count value calculation
5.12	void MultiCancel(void); *3	Multi Touch Canceller control
5.13	void OnOffJudgement(void);	Judgement for touch or not
5.14	void Slider(void); *4	Slider position detection
5.15	void SWheel(void); *5	Wheel position detection
5.16	void CorrectSub(uint16_t s_dci1);	Drift correction control
5.17	void MsrCalibration(void);	Calibration control

*3. When MULTI_CANCEL is defined, you can use MultiCancel ().
*4. When SLIDER_USE is defined, you can use Slider ().
*5. When WHEEL_USE is defined, you can use SWheel ().

Table 3-3 User API List

Chapter	API name	Remarks
6.1	TOUCH_ONOFF_STATUS_E GetTouchOnOff(void);	Get the status of touch position in Touch key
6.2	TOUCH_ONOFF_STATUS_E GetWheelPosition(void); *6	Get the status of touch position in Wheel
6.3	TOUCH_ONOFF_STATUS_E GetSliderPosition(void); *7	Get the status of touch position in Slider
6.4	MODE_TSCU_MEASURE_E SetTscuMode(TSCU_MODE_E mode);	Start/Stop TSCU Measurement
6.5	uint8_t SetTscuDcen(DRIFT_ENABLE_E sw);	Set Touch CH having an effect of Drift correction.

*6. When WHEEL_USE is defined, you can use GetWheelPosition ().
*7. When SLIDER_USE is defined, you can use GetSliderPosition ().

4. Macro definition

Change Macro definition defined in touch_control.h according to your application.

4.1 TSCU_INV_NOISE

Remarks

Select the TSCU Setting from the followings.

- Normal setting. Operation clock is 5 MHz.
- Custom setting (Countermeasure against the inverter noise). Operation clock is 20 MHz.

Example

```
- Normal settings
  // #define  TSCU_INV_NOISE // Comment-out
             or
  #undef TSCU_INV_NOISE  // TSCU_INV_NOISE is disabled using #undef


- Custom settings (Countermeasure against the inverter noise)
  #define  TSCU_INV_NOISE // TSCU_INV_NOISE is enabled
```

4.2 MULTI_CANCEL

Remarks

This macro is conditional compilation to build a control module for Multi Touch Canceller. When MULTI_CANCEL is defined, user can use the API of Multi Touch Canceller. Specify the influence range of Multi Touch Canceller to MULTI_START_CH and MULTI_END_CH.

Note

Multi Touch Canceller prohibits the simultaneous touch of touch keys more than two.

Example

```
- Multi Touch Canceller is disabled.
  // #define  MULTI_CANCEL   // Comment-out
             or
  #undef MULTI_CANCEL  // MULTI_CANCEL is disabled using #undef


- Multi Touch Canceller is enabled.
  #define  MULTI_CANCEL   // MULTI_CANCEL is enabled
```

4.3 MULTI_START_CH

Remarks
Define the start channel of touch keys processed by Multi Touch Canceller.

Note
Define MULTI_START_CH to meet the following conditions.
MULTI_START_CH < MULTI_END_CH

Value range
0 - 34: R8C/36T-A or R8C/38T-A

Example
```
-  Start channel of Multi Touch Canceller
#define MULTI_START_CH 24
```

4.4 MULTI_END_CH

Remarks
Define the end channel of touch keys processed by Multi Touch Canceller.

Note
Define MULTI_END_CH to meet the following conditions.
MULTI_START_CH < MULTI_END_CH

Value range
1 - 35: R8C/36T-A、R8C/38T-A

Example
```
-  End channel of Multi Touch Canceller
#define MULTI_END_CH 35
```

4.5 SLIDER_USE

Remarks

This macro is conditional compilation to build a control module for Slider position detection. When SLIDER_USE is defined, user can use the API of Slider position detection.

Example

```
- Slider function is disabled.
  // #define SLIDER_USE     // Comment-out
              or
  #undef SLIDER_USE// SLIDER_USE is disabled using #undef


- Slider function is enabled.
  #define SLIDER_USE     // SLIDER_USE is enabled
```

4.6 WHEEL_USE

Remarks

This macro is conditional compilation to build a control module for Wheel position detection. When WHEEL_USE is defined, user can use the API of Wheel position detection.

Example

```
- Wheel function is disabled.
  // #define WHEEL_USE   // Comment-out
              or
  #undef WHEEL_USE // WHEEL_USE is disabled using #undef


- Wheel function is enabled.
  #define WHEEL_USE   // WHEEL_USE is enabled
```

4.7 MAX_CH

Remarks

Define numerical value that added one to the largest number of Touch CH.

Value range

0: Do not use

1 - 35: R8C/36T-A、R8C/38T-A

Example

- Using channel-0, channel-3, channel-4, channel 6 as touch electrode.

```
#define MAX_CH 7    // CH6(largest number of Touch CH) + 1.
```

4.8 DF_TSIERn

Remarks
Select a use of Touch CH from Touch sensor pin and I/O port.

Note
n = 0, 1, 2

Relationship between the bit pattern and Touch CH is as follows.

Table 4-1 DF_TSIER0

	b15															b0
Touch CH	15 *	14 *	13 *	12 *	11	10	9 *	8	7	6	5	4	3	2	1	0

* Set to 0 at the time of use of R8C/36T-A.

Table 4-2 DF_TSIER1

	b15															b0
Touch CH	31	30 *	29 *	28	27	26 *	25	24	23	22	21	20	19	18	17	16

* Set to 0 at the time of use of R8C/36T-A.

Table 4-3 DF_TSIER2

	b15															b0
Touch CH	*	*	*	*	*	*	*	*	*	*	*	*	35	34	33	32

* Set to 0

Value range
0: I/O port

1: Touch sensor pin

Example
- Using channel-0, channel-3, channel-4, channel 6 as touch electrode.

```
#define DF_TSIER0 0x5b
#define DF_TSEER1 0x00
#define DF_TSIER2 0x00
```

4.9 DF_CHxx_REF

Remarks
Define the initial value of the reference count value according to Touch CH to use as a Touch electrode.

Note
- xx expresses two columns of channel numbers.

Value range
0 - 65535

Example
- Reference count value of channel-8

```
#define DF_CH08_REF 308
```

- Reference count value of channel-16

```
#define DF_CH16_REF 316
```

4.10 DF_CHxx_THR

Remarks
Define the initial value of the Threshold count value for judgement of touch or not according to Touch CH to use as a Touch electrode.

Note
- xx expresses two columns of channel numbers.

Value range
0 - 65535

Example
- Threshold count value of channel-8

```
#define DF_CH08_THR 58
```

- Threshold count value of channel-16

```
#define DF_CH16_THR 66
```

4.11 DF_CHxx_HYS

Remarks
Define the initial value of the hysteresis value of the threshold count according to Touch CH to use as a Touch electrode.

Note
- xx expresses two columns of channel numbers.

Value range
0 - 65535

Example
- Hysteresis value of channel-8

```
#define DF_CH08_HYS 4
```

- Hysteresis value of channel-16

```
#define DF_CH16_HYS 5
```

4.12 DF_MSA_DATA

Remarks
Define the initial value of MSA.

Note
When the touch judgement is continued, the judgement becomes forcibly non-touch judgement.

Value range
0: MSA does not function.

1 - 255: MSA functions.

Example
- MSA does not function

```
#define DF_MSA_DATA 0
```

4.13 DF_ACCUMULATION

Remarks
Define the initial value of ACD Off to On and ACD On to Off.

Note
- ACD Off to On
  When a count value drops the threshold count value the N times, the count value is judged touch. (N is the value of ACD Off to On)

- ACD On to Off
  When a count value exceeds the threshold count value the N times, the count value is judged non-touch. (N is the value of ACD On to Off)

Example
- `ACD Off to On = 0Ah, ACD On to Off = 05h`

```
#define DF_ACCUMULATION 0x050A
```

4.14 DF_DCI_DRIFT

Remarks
Define the initial value of the interval to execute Drift correction.

Note
Drift correction corrects the reference count value according to environment.

Value range
`0 - 65535`

Example
- `Drift correction interval is 32`

```
#define DF_DCI_DRIFT      32
```

4.15 WORKBENCH_HEWSVR_ENABLE

Remarks

This macro is conditional compilation to build a control module for communication with Workbench using HewTargetServer.

Example

```
- Communication function with Workbench using HewTargetServer is disabled.
  // #define WORKBENCH_HEWSVR_ENABLE     // Comment-out
             or
  #undef WORKBENCH_HEWSVR_ENABLE   // WORKBENCH_HEWSVR_ENABLE is disabled using
#undef


-  Communication function with Workbench using HewTargetServer is enabled.
  #define WORKBENCH_HEWSVR_EANBLE  // WORKBENCH_HEWSVR_ENABLE is enabled
```

4.16 SUPPORT_UART

Remarks

This macro is conditional compilation to build a control module for communication with Workbench using UART.

Example

```
- Communication control module is disabled.
  // #define SUPPORT_UART    // Comment-out
             or
  #undef SUPPORT_UART  // SUPPORT_UART is disabled using #undef


-  Communication control module is enabled.
   #define SUPPORT_UART   // SUPPORT_UART is enabled
```

5. Basic API Reference

5.1 TouchDtcInitialSet

Remarks

Touch API uses DTC to transfer measured value from registers to RAM.
The main settings about DTC are as follows.

Table 5-1 DTC Registers and Settings

Item	Setting value
Transfer mode	Repeat
Destination address control	Add
DTC block size	4 byte
DTC transfer control	MAX_CH
DTC Activation	TSCU DTC activation

Notes

Notes on DTC is as follows.

The lower 8 bits of the initial value for the repeat area address must be 00h.

Refer to [13.5.5 Repeat Mode] in “R8C/36T-A Group User’s Manual: Hardware”(R01UH0240EJ) or “R8C/38T-A Group User’s Manual: Hardware” (R01UH0241EJ) for detail.

Requirements

- Call this API from a initialization routine.
- Call this API before TscuInitial().

Declaration

```
void TouchDtcInitialSet( void )
```

Parameters

nothing

Return value

nothing

Examples

```
void main( void )
{
        :
   TouchDtcInitialSet();
   TcuInitial();
        :
   while(1){       // Main Loop
            :
        TscuMeasure();
            :
   }
}
```

5.2 TouchDataInitial

Remarks

This API initializes global variables used in touch API.

Requirements

- Call this API from a initialization routine.
- Call this API before TscuInitial().

Declaration

```
void TouchDataInitial( void )
```

Parameters

nothing

Return value

nothing

Examples

```
void main(void)
{
        :
   TouchDataInitial();
   TscuInitial();
        :
   while(1){       // Main Loop
           :
        TscuMeasure();
           :
   }
}
```

5.3 CheckReadFlashData

Remarks

This API reads data to be used in Touch API from DATA FLASH, and stores the data to RAM. An initial value of ROM table is set in RAM if there is no data in Data Flash. The RAM to store the data read from DATA FLASH is as follows.

Table 5-2 the RAM to store the data read from DATA FLASH

RAM	Remarks
Ch_para_Ref[MAX_CH]	Reference count value
Ch_para_Thr[MAX_CH]	Threshold count value for judgement of touch or not
Ch_para_Hys[MAX_CH]	Hysteresis value of the threshold count
Msa	The value of Maximum successive ON count
Mode	Function mode
Acd	The value of Accumulated judgement count
Dci	The value of Drift correction interval
chaxA_selectdata[3]	The value of CHxA (0: CHxA0, 1: CHxA1)
Athr	Threshold value of Multi Touch Canceller

Requirements

- Call this API from a initialization routine.
- Call this API before TouchDataInitial2() and after TscuInitial().

Declaration

```
uint8_t CheckReadFlashData ( void )
```

Parameters

nothing

Return value

nothing

Examples

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

5.4 TouchDataInitial2

Remarks
This API initializes global variables to store values saved in DATA FLASH.

Requirements
- Call this API from a initialization routine.
- Call this API before TscuInitial() and after CheckReadFlashData().

Declaration
```
void TouchDataInitial2( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void main(void)
{
        :
   result = CheckReadFlashData();
   TouchDataInitial2();
   TscuInitial();
        :
   while(1){       // Main Loop
           :
        TscuMeasure();
           :
   }
}
```

5.5 TscuInitial

Remarks

This API sets TSCU registers. The TSCU measurement supports “Normal measurement” (Operation clock is 5MHz) and “Custom measurement” (Countermeasure against noise is implemented. Operation clock is 20MHz). Please refer to [4.1 **TSCU_INV_NOISE**] about the change of the Normal measurement and the Custom measurement.

The setting of TSCU registers is as follows.

Table 5-3 TSCU Registers and Settings

Item	Normal measurement	Custom measurement
Count source	f4 (5 MHz - f1 clock divided by 4)	f1 (20 MHz)
TSCU interruption	Enable	Enable
PRE measurement	None	None
Random measurement	None	None
Majority measurement	None	None
TSCU measurement start trigger	Software trigger	Software trigger
Period 1	128 cycles	128 cycles
Period 2	1 cycle	8 cycles
Period 3	1 cycle	4 cycles
Period 4	1 cycle	1 cycle
Period 5	1 cycle	Skip
Period 6	1 cycle	6 cycles
Measurement mode	Scan mode	Scan mode
Channel select	MAX_CH - 1	MAX_CH - 1
Transfer destination address	Scudata	Scudata
Secondary counter	7 times	32 times
TSCU interrupt level	level 1	level 1

Requirements

- Call this API from a initialization routine.
- Call this API before starting of TSCU measurement.

Declaration

```
void TscuInitial( void )
```

Parameters

nothing

Return value

nothing

Examples

```
void main(void)
{
        :
    TscuInitial();
    SetTouchSensor();
        :
    while(1){       // Main Loop
            :
        TscuMeasure();
            :
    }
}
```

5.6 TscuInterrupt

Remarks

This API is a interrupt process for a interrupt that is generated after TSCU measurement finishes and updates TSCU Measurement Mode. Usually, this API is called when TSCU Measurement Mode is MD_TSCU_RUN (TSCU measurement is running), and changes TSCU Measurement Mode into MD_TSCU_FINISH (TSCU measurement finish).

When TSCU Measurement Mode is MD_TSCU_STOP (TSCU is stopped), this API does not change the TSCU Measurement Mode. When TSCU Measurement Mode is not MD_TSCU_RUN and is not MD_TSCU_STOP, this API changes the TSCU Measurement Mode into MD_TSCU_READY (TSCU measurement is ready) and starts TSCU measurement.

Requirements

- TSCU interrupt is generated after TSCU measurement finishes.
- This API clears TSCU interrupt request flag to generate the TSCU interrupt again.

Declaration

```
void TscuInterrupt( void )
```

Parameters

nothing

Return value

nothing

Examples

```
#pragma INTERRUPT TscuInterrupt
void TscuInterrupt ( void )
{
   if( md_tscu_measure == MD_TSCU_RUN ){
        md_tscu_measure = MD_TSCU_FINISH;
   }else
   if( md_tscu_measure == MD_TSCU_STOP ){
        md_tscu_measure = MD_TSCU_STOP;
   }else{
        md_tscu_measure = MD_TSCU_READY;
        SetTouchSensor();
   }
   tscucr0_addr.bit.tscue = OFF;
   tscufr_addr.bit.sif = OFF;
}
```

5.7 TscuMeasure

Remarks
This API controls the following functions.

FtAddMakeAve:	Moving addition value of count calculation
SetTouchSensor:	TSCU measurement start
MakeCthr:	Threshold count value calculation
MultiCancel:	Multi Touch Canceller control
OnOffJudgement:	Judgement for touch or not
Slider:	Slider position detection
SWheel:	Wheel position detection
CorrectSub:	Drift correction control
MsrCalibration:	Auto calibration

Requirements
- This API is called from main() and works when TSCU Measurement Mode is finishes.
- When primary counter overflows, this API does not execute the judgement process for touch and Drift correction. Then this API re-starts TSCU measurement and auto calibration.

Declaration
```
void TscuMeasure( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void main(void)
{
   while(1){
           :
        TscuMeasure();
           :
   }
}
```

5.8 CheckWriteStatusFlashData

Remarks
This API writes data to be used in Touch API to DATA FLASH. Please refer to [5.3 CheckReadFlashData] about the RAM to store the data read from DATA FLASH.

Requirements
This API is called from main() and works when Workbench requested an update of DATA FLASH

Declaration
```
void CheckWriteStatusFlashData( void )
```

Parameters
nothing

Return value
nothing

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

5.9 FtAddMakeAve

Remarks
This API executes the “Moving addition value of count” to TSCU measurement result and calculates count values.

Requirements
- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure()
- Call this API before MakeCthr().

Declaration
```
void FtAddMakeAve( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void TscuMeasure( void )
{
    FtAddMakeAve();
       :
}
```

5.10 SetTouchSensor

Remarks
This API starts TSCU measurement. This API changes TSCU Measurement Mode into MD_TSCU_RUN after TSCU measurement starts.

Requirements
- When TSCU Measurement Mode is MD_TSCU_READY, this API starts TSCU measurement .
- When a start of TSCU measurement failed, this API returns 0(TSCU measurement stop).

Declaration
```
uint8_t SetTouchSensor( void )
```

Parameters
nothing

Return value
- 0 TSCU measurement is stopped
- 1 TSCU measurement is started

Examples
```
void TscuMeasure( void )
{
        :
   md_tscu_measure = MD_TSCU_READY;
   SetTouchSensor();
        :
}
```

5.11 MakeCthr

Remarks
This API calculates a Dcount and the threshold count value for judgement touch or not. Dcount is differences between reference count value and count value.

Requirements
- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after FtAddMakeAve().

Declaration
```
void MakeCthr( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void TscuMeasure( void )
{
        :
   FtAddMakeAve();
   MakeCthr();
        :
}
```

5.12 MultiCancel

Remarks
This API executes Multi Touch Canceller. Define the target of the Multi Touch Canceller to MULTI_START_CH and MULTI_END_CH, and define MULTI_CANCEL to build a control module for Multi Touch Canceller.

Requirements
- When TSCUMeasurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after MakeCtrh() and before OnOffJudgement().

Declaration
```
void MultiCancel( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void TscuMeasure( void )
{
        :
   MakeCthr();
   MultiCancel();
   OnOffJugment();
        :
}
```

5.13 OnOffJudgement

Remarks
This API judges touch or non-touch of the touch key and stores the judgement results to BDATA.

Requirements
- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after MakeCthr().

Declaration
```
void OnOffJudgement( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void TscuMeasure( void )
{
   :
   MakeCthr ();
        :
   OnOffJudgement();
        :
}
```

5.14 Slider

Remarks

This API detects the touch position on the slider. Change the number of the Touch CH constructing the slider and resolution according to target system. This API supports two types of resolution, and stores the decoded result of the basic resolution to “sldposition_raw”, and stores the decoded result of the user resolution to “sldposition_r”.

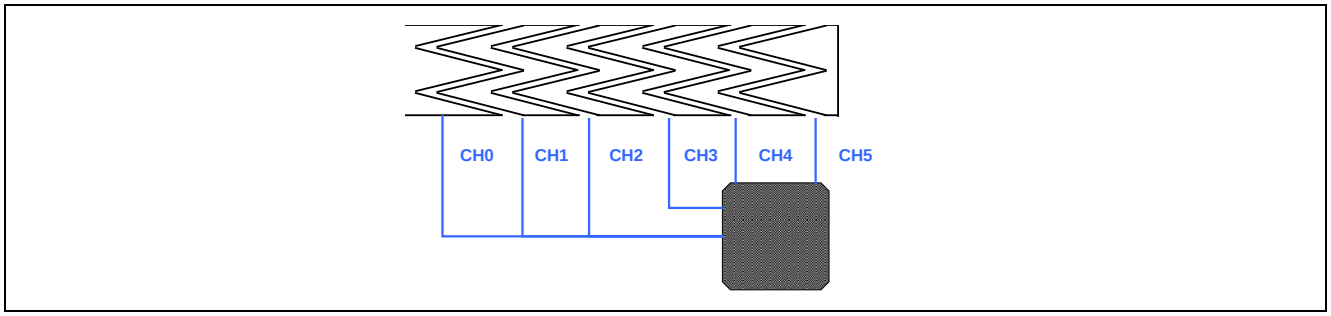

Figure 5-1 Slider Image

Requirements

- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after OnOffJudgement().
- Add slider_control.c and slider_control.h to your application software and define SLIDER_USE to build a control module for Slider().

Declaration

```
void Slider( void )
```

Parameters

nothing

Return value

nothing

Examples

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

5.15 SWheel

Remarks

This API detects the touch position on the wheel. This API divides the wheel into 72 parts and shows a touch position with numerical value (1 - 72). WPOSn (n is from 1 to 4) expresses Touch CH.

This API supports two types of resolution, and stores the decoded result of the basic resolution to “diff_angle_4ch”, and stores the decoded result of the user resolution to “wheel_sw”.

Refer to an application note for the details about the wheel control.

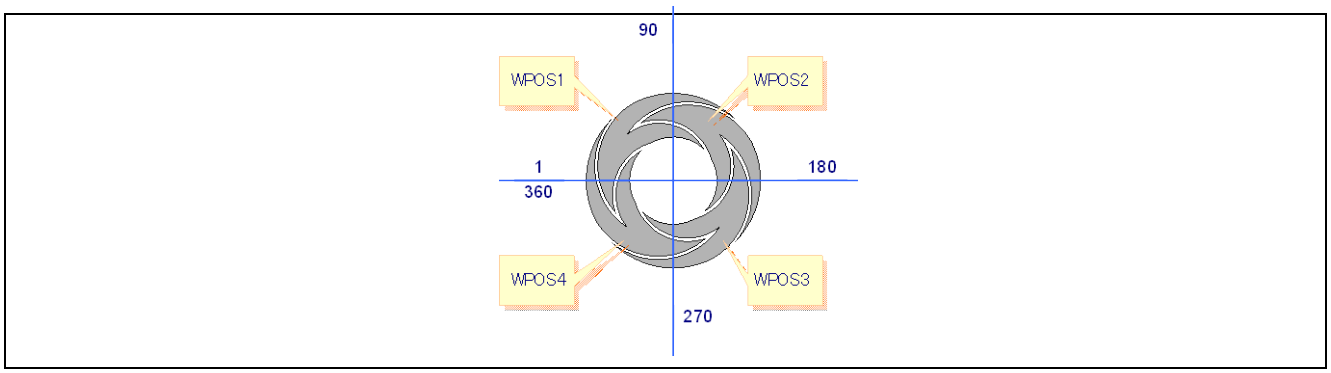

Figure 5-2 Wheel Image

Requirements

- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after OnOffJudgement().
- Add wheel_control.c and wheel_control.h to your application software and define WHEEL_USE to build a control module for SWheel().

Declaration

```
void SWheel( void )
```

Parameters

nothing

Return value

nothing

Examples

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

5.16 CorrectSub

Remarks
This API executes Drift correction, and stores the reference count value to Nref[].

Requirements
- When TSCU Measurement Mode is MD_TSCU_FINISH, this API is called from TscuMeasure().
- Call this API after OnOffJudgement().

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

Parameters
s_dci1 Specifies interval of Drift correction execution

Return value
nothing

Examples
```
void TscuMeasure( void )
{
        :
   OnOffJudgement();
   CorrectSub(s_dci);
        :
}
```

5.17 MsrCalibration

Remarks
This API executes auto calibration.

Requirements
- When TSCU Measurement Mode is MD_TSCU_FINISH and Auto calibration does not finishes, this API is called from TscuMeasure().

Declaration
```
void MsrCalibration( void )
```

Parameters
nothing

Return value
nothing

Examples
```
void TscuMeasure( void )
{
        :
   if (meascal == 0 ) {   // Calibration flag is false
        :
   }else{                 // Calibration flag is true
        MsrCalibration();
   }
        :
}
```

6. User API Reference

6.1 GetTouchOnOff

Remarks

This API returns reference status of BDATA. BDATA is a global variable to store On/Off status of Touch CH. When this API returns DATA_OK, the reference of BDATA is possible.

Declaration

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

Parameters

nothing

Return value

- DATA_OK (0x00) Reference of BDATA is possible
- STOP_MODE (0xFE) TSCU measurement stops
- OVER_MODE (0xFE) Overflow error
- CALIB_MODE (0xFD) Auto calibration functions

Examples

```
void main( void )
{
        :
   If( DATA_OK == GetTouchOnOff() ){
        Check_touch_onoff();    // Function made by user
   }
        :
}
```

BDATA

- BDATA has information that Touch CH is On or Off by a bit unit.
- When a value of the bit is zero, it is shown that the corresponding Touch CH is touched.
- The relations of each bit and Touch CH are as follows.

Declaration

```
TOUCH_EXTERN WORD_ACS_T BDATA[3];
```

- BDATA[0]

b15															b00
CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- BDATA[1]

b15															b00
CH31	CH30	CH29	CH28	CH27	CH26	C25	C24	CH23	CH22	CH21	CH20	CH19	CH18	CH17	CH16

- BDATA[2]

b15															b00
-	-	-	-	-	-	-	-	-	-	-	-	CH35	CH34	CH33	CH32

6.2 GetWheelPosition

Remarks

This API returns reference status of diff_angle_4ch and wheel_sw. diff_angle_4ch is a global variable to store an angle that a wheel is touched. wheel_sw is a global variable to store touch position on the wheel. When this API returns DATA_OK, the reference of diff_angle_4ch and wheel_sw is possible.

Declaration

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

Parameters

nothing

Return value

- DATA_OK (0x00) Reference of diff_angle_4ch and wheel_sw is possible
- STOP_MODE (0xFE) TSCU measurement stops
- OVER_MODE (0xFE) Overflow error
- CALIB_MODE (0xFD) Auto calibration functions

Examples

```
void main(void)
{
        :
   If( DATA_OK == GetWheelPosition() ){
        Check_wheel_positoin(); // Function made by user
   }
        :
}
```

diff_angle_4ch

- diff_angle_4ch stores an angle when the wheel was traced with a finger.
- The range of the angle value is from zero to 360.
- When the value is zero, it is shown that the wheel is not touched.

Declaration

```
WHEEL_EXTERN uint32_t diff_angle_4ch;
```

wheel_sw

- wheel_sw stores the touch position on the wheel.
- The range of the position value is from zero to 72.
- When the value is zero, it is shown that the wheel is not touched.

Declaration

```
WHEEL_EXTERN uint16_t wheel_sw;
```

6.3 GetSliderPosition

Remarks

This API returns reference status of sldposition_raw and sldposition_r. sldposition_raw and sldposition_r are global variables to store touch position on the slider. When this API returns DATA_OK, the reference of sldposition_raw and sldposition_r is possible.

Declaration

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

Parameters

nothing

Return value

- DATA_OK (0x00) Reference of sldposition_raw and sldposition_r is possible
- STOP_MODE (0xFE) TSCU measurement stops
- OVER_MODE (0xFE) Overflow error
- CALIB_MODE (0xFD) Auto calibration functions

Examples

```
void main(void)
{
        :
   If( DATA_OK == GetSliderPosition() ){
        Check_slider_positoin(); // Function made by user
   }
        :
}
```

sldposition_raw

- sldposition_raw stores touch position on the slider.
- When the value of sldposition_raw is 0xFFFFFFFF, it is shown that the slider is not touched.

Declaration

```
SLIDER_EXTERN uint32_t sldposition_raw;
```

sldposition_r

- sldposition_r stores touch position on the slider.
- When the value of sldposition_r is 0xFFFF, it is shown that the slider is not touched.

Declaration

```
SLIDER_EXTERN uint16_t sldposition_r;
```

6.4 SetTscuMode

Remarks
This API starts and stops TSCU measurement.

Requirements
- When the stop of TSCU measurement was requested during the executing of TSCU measurement, this API aborts the TSCU measurement.
- When the start of TSCU measurement is requested during the executing of TSCU measurement, TSCU measurement and the other processing (Judgement for touch or not, Drift correction, etc.) are continued.

Declaration
```
MODE_TSCU_MEASURE_E SetTscuMode( TSCU_MODE_E mode );
```

Parameters
mode Specifies start or stop of TSCU measurement.
MDRQ_TSCU_STOP (0x00) - This value requests the start of TSCU measurement
MDRQ_TSCU_START (0x01) - This value requests the stop of TSCU measurement.

Return value
- MD_TSCU_STOP (0x00) TSCU measurement stops
- MD_TSCU_READY (0x01) TSCU measurement is ready
- MD_TSCU_RUN (0x02) TSCU measurement is running
- MD_TSCU_FINISH (0x03) TSCU measurement finishes

Examples
```
void main(void)
{
    MODE_TSCU_MEASURE_E tscu_mode;
        :
    tscu_mode = SetTscuMode(MDRQ_TSCU_START);
        :
}
```

6.5 SetTscuDcen

Remarks
This API validates Drift correction every each Touch CH.

Requirements
- The setting by this API is eflected from the next processing of Drift correction.

Declaration
```
uint8_t SetTscuDcen( DRIFT_ENABLE_E sw );
```

Parameters
sw Specifies a method to validate Drift correction.

DC_NON(0x00)	- Drift correction is invalid with all Touch CH.
DC_ALL(0x01)	- Drift correction is valid with all Touch CH.
DC_ENABLE(0x02)	- Drift correction is set according to user's definition.

Return value
- 0x00 Normal end
- 0x01 Parameter error

Examples
```
void main(void)
{
        :
   SetTscuDcen(DC_ALL);
        :
}
```

7. Touch API Hierarchy Chart

Visually details the relationship of Touch API.

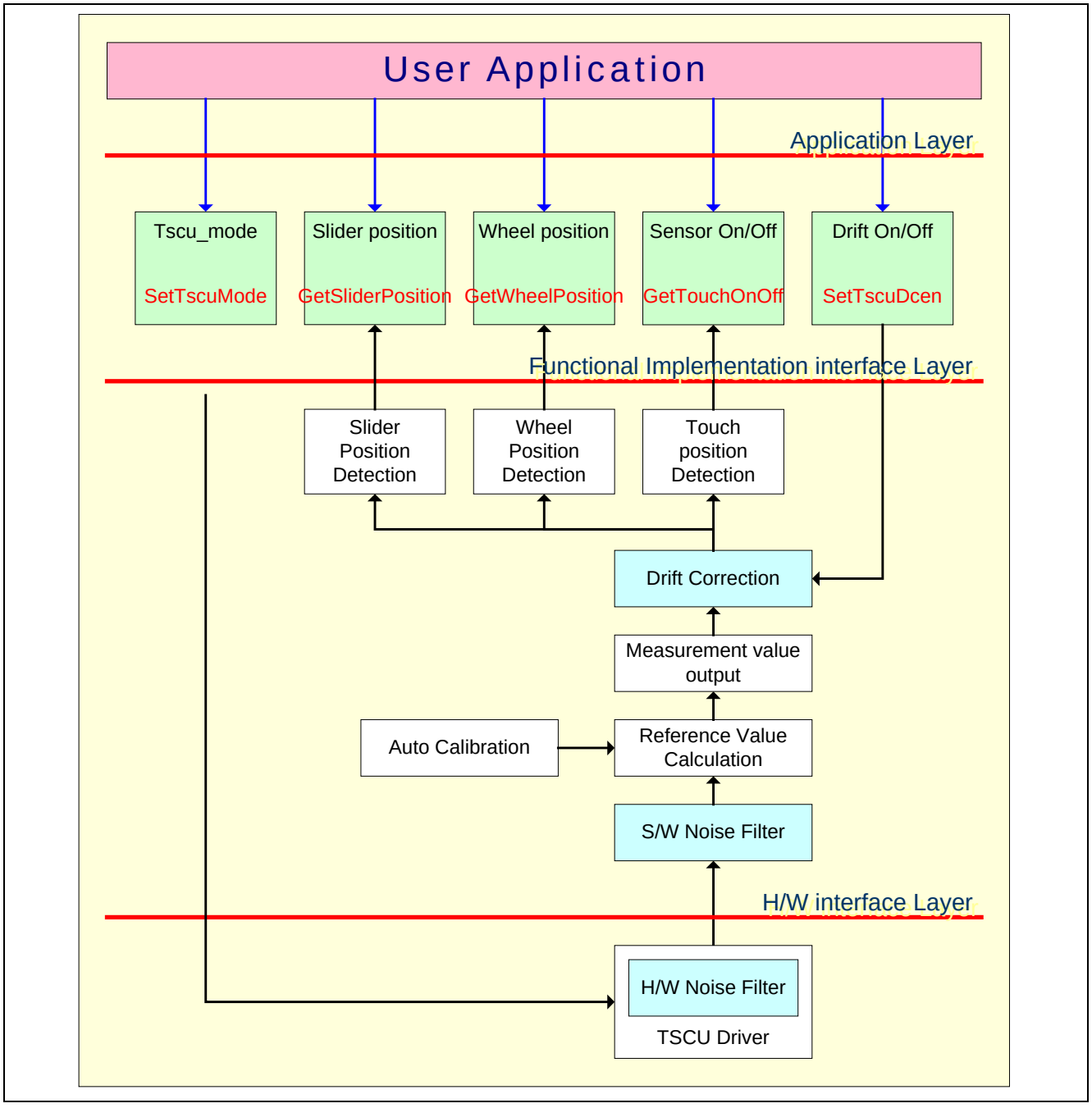

Figure 7-1 Touch API Hierarchy Chart

8. Supplementary explanation

Visually details the flowchart about TSCU measurement by hardware and Judgement process for touch or not by software.

8.1 TscuMeasure flowchart

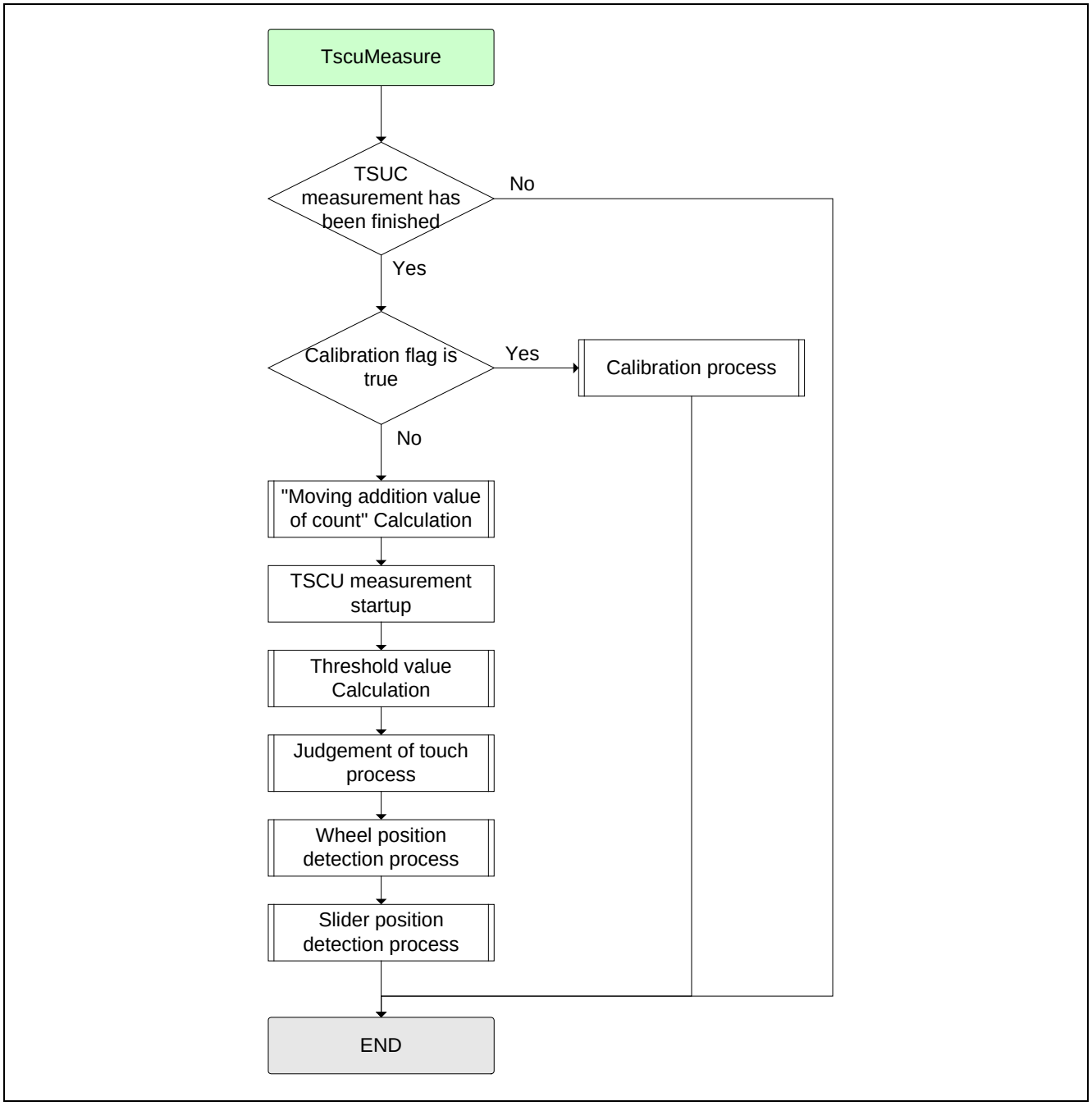

Figure 8-1 TscuMeasure flowchart

8.2 Timing chart of Touch detection

Timing chart about Touch detection is as follows.

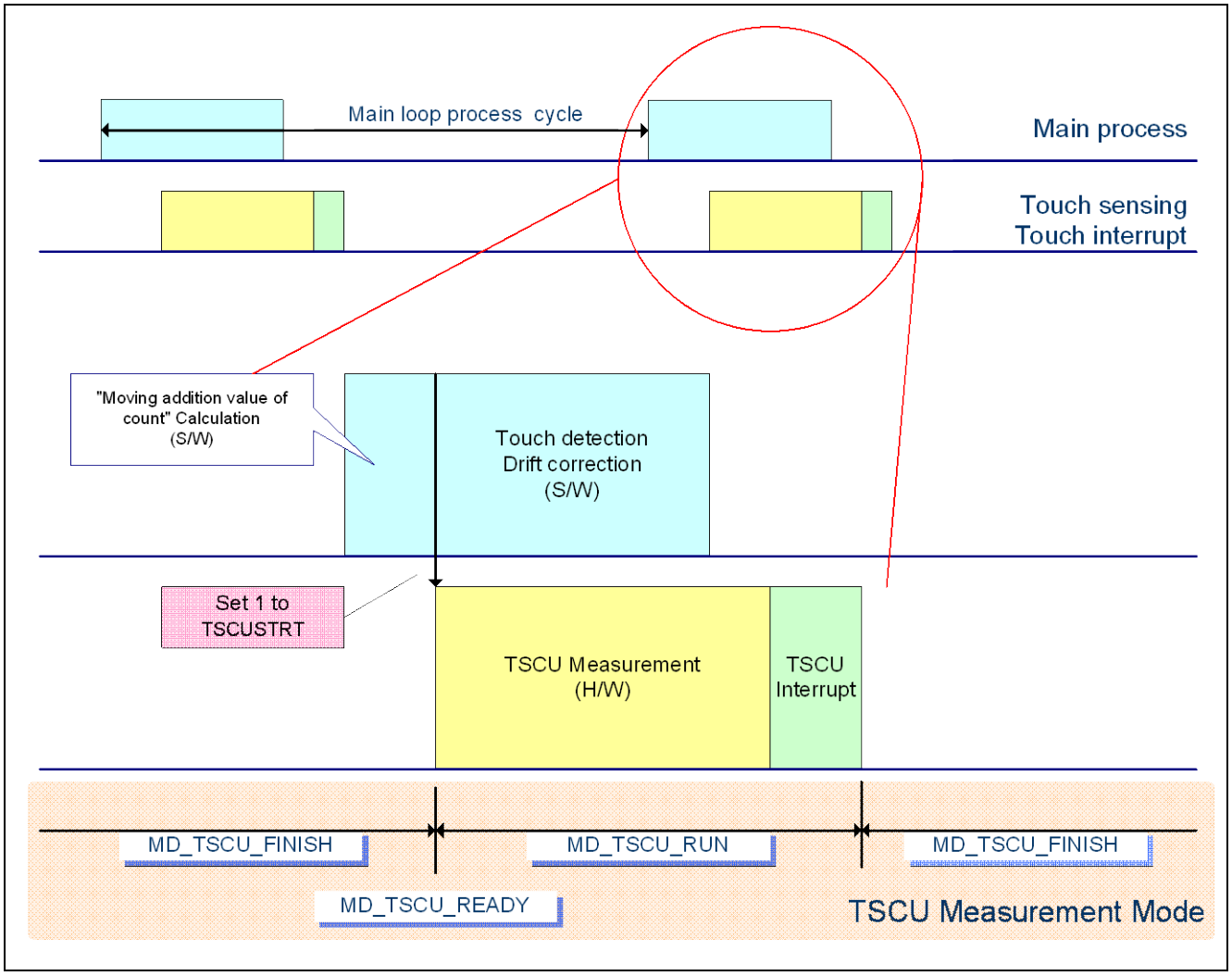

Figure 8-2 Timing chart

8.3 TSCU Measurement Mode

The relationship between TSCU Measurement Mode and the Base API process is as follows.

1. When TSCU Measurement Mode is MD_TSCU_FINISH, the Base API executes the Moving addition value of count.
2. The process of Moving addition value of count saves the result of TSCU measurement. Then the Base API changes TSCU Measurement Mode to MD_TSCU_READY, and starts TSCU measurement.
3. The Base API sets one to TSCUSTRT in TSCU Control Register 0. Then the Base API changes TSCU Measurement Mode to MD_TSCU_RUN.
4. The Base API executes the Judgement process for touch or not, and Drift correction.
5. TSCU measurement is finished, and TSCU interrupt occurs. The TSCU interrupt process changes TSCU Measurement Mode to MD_TSCU_FINISH.

* You can change a timing of the process-3.

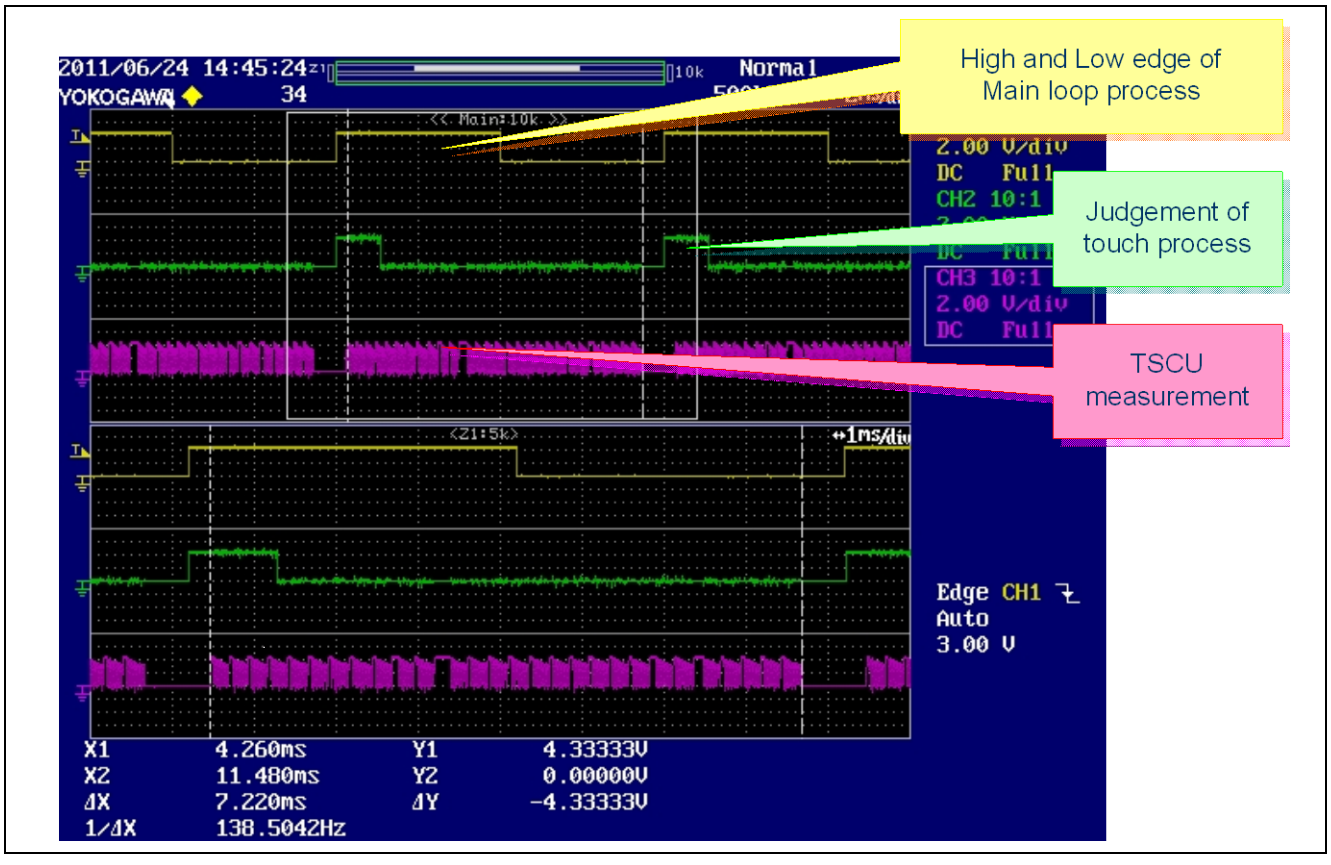

Figure 8-3 Touch process waveform

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Inquiries
http://www.renesas.com/inquiry

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	May.23.2013	—	Numbering change(Content is as same as R010744EJ0100)

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.2