Application Note

Document No.: AN1090

APM32 TSC Touch Module

Version: V1.0

Contents

1. Introduction

1.1. Overview

This application manual introduces the APM32 TSC touch module to users, including hardware design principles and software design methods, in order to help users implement basic functions such as keys, linear, and rotary. For information about chips and TSC register, please refer to the user manual and datasheet on our official website.

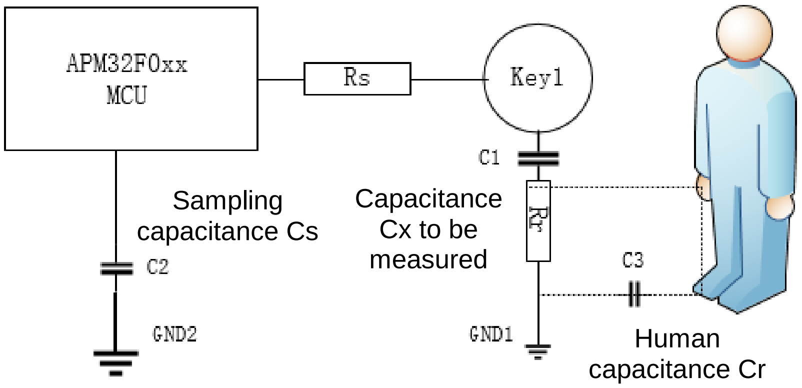
This manual takes the APM32F072VB model in the APM32F0xx series as an example for description.

1.2. Terms

Term	Description
TSC	Touch sensing controller peripheral
Cs	Sampling capacitance
Cx	Sensor capacitance
Channel	Channel, basic acquisition port, corresponding to hardware GPIO.
Block	Blocks, multiple channels collected simultaneously
Object	Any touch sensor (such as touchkey, linear or rotary)
LinRot	Linear or rotary touch sensor (multiple channels)
TouchKey	Key sensor (single channel)
Meas	Current signal measured on a channel
Delat	Difference between measured value and reference value
ECS	Environment change system
DTO	Detection time-out
DXS	Detection exclusion system
Filter	Noise filters

1.3. Principle

Charge transfer and acquisition principles:



When it is touched by human hand, the loop introduces Cr, and the capacitance of the key to ground increases.

Figure 1-3

1. Use the charge storage characteristics of the capacitor.
2. The capacitance Cx to be measured on the electrode charges the sampling capacitance Cs.
3. In the process of charge transfer, mount the analog switch in the hardware GPIO.
4. Repeat the process of charge transfer until the voltage on the sampling capacitance Cs reaches the threshold value of GPIO.
5. The number of charge transfers required to reach the threshold directly represents the capacitance value of the electrode to be measured.
6. When the electrode is touched, the capacitance of the sensor to the ground increases, the stored charge in the electrode increases (Cx), therefore the number of charge transfers required for the voltage of the sampling capacitance (Cs) to reach the threshold value of GPIO decreases, and the measured value decreases.
7. When the measured value is below the threshold value, TSC conducts touch detection.

1.4. Functional Characteristics

1. Based on experienced surface charge transfer principle.
2. Every three capacitance sensing channels to be measured are equipped with one sampling capacitance to reduce system overhead.
3. Depending on different model series, 6~8 analog I/O groups are embedded, supporting up to 3*6~3*8 channels.
4. Each I/O group corresponds to a counter to record the current measured value.
5. The sampling capacitance pins and channels in each I/O group can be configured.
6. The software sets the maximum number of transfers to avoid DTO status.
7. Set a special measurement completion flag and a maximum error flag, either of which can trigger interrupts.
8. This TSC supports such sensing methods as keys, linear, and rotary.
9. Improve sensitivity and anti-jamming capability through DXS (detection exclusion), ECS (environment change system), Filter (noise filters), etc.
10. It supports simple API programs for users to configure and view the operating status.

Analog I/O group:

Group No.	Number of channels for each group of capacitive sensors		
	APM32F072Vx	APM32F072Rx	APM32F072Cx
G1	3	3	3
G2	3	3	3
G3	3	3	2
G4	3	3	3
G5	3	3	3
G6	3	3	3
G7	3	0	0
G8	3	0	0
Total number of capacitive sensor channels	24	18	17

Table 1-4

2. Hardware Design

2.1. Measurement Circuit

Take an example of measurement circuit design when keys are touched, and the linear and rotary sensors are the same, except that the electrode position is different, as shown in the figure:

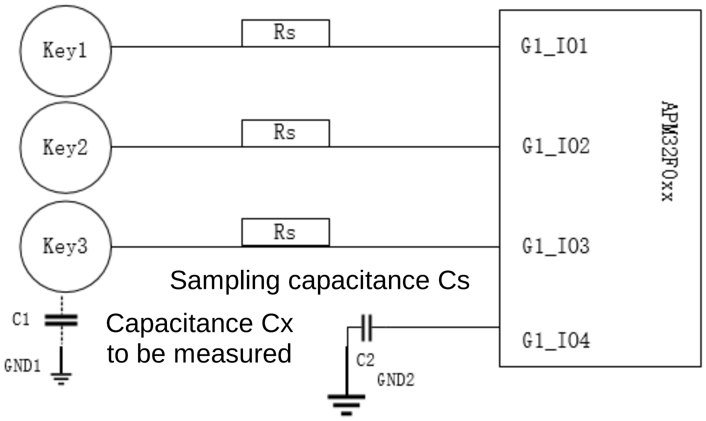


Figure 2-1

Note: A 470 Ω -10k Ω resistor is added near the MCU pin for ESD protection and anti-noise filtering. Each electrode is connected in series with a 10K Rs to improve ESD robustness.

2.2. Dielectric Constant

The panel is the main part of the capacitive medium between the finger and the electrode. Its dielectric constant (εr) can be used to distinguish panel materials. The table 2-3 shows the electric field propagation capability inside the materials. The larger the dielectric constant is, the stronger the propagation capability is.

Material	εr
Air	1.00059
Glass	4 to 10
Sapphire glass	9 to 11
Mica	4 to 8
Nylon	3
Organic glass	3.4
Polyethylene	2.2
Polystyrene	2.56
Polyester (PET)	3.7
FR4 (glass fiber + epoxy resin)	4.2
PMMA (polymethyl methacrylate)	2.6 to 4

Table 2-2

2.3. Sensitivity

$$T_V = t/\varepsilon_r$$

For panel material and thickness (T), t is the thickness of the dielectric medium.

Tv is the equivalent vacuum thickness of electric field conduction of the material. The smaller the value is, the easier it is for the electric field to penetrate.

For panels with the same Tv value, the sensitivity of the keys is the same.

Ways to improve sensitivity:

$$C = \varepsilon_r\varepsilon_0（A/d）$$

Increase the contact area between the finger and the sensor (A).

Reduce the panel thickness (d).

Select the electrolyte with a larger value ε.

The GND layer shall not be too close to the sensor.

Avoid using metal coatings on sensor accessories.

$C = \varepsilon\ r\ \varepsilon\ 0\ (A/d)$

2.4. Touch Keys

Each key occupies one touch channel and can be designed as an independent key or matrix key.

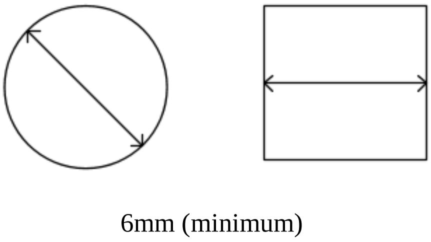


Figure 2-4-1

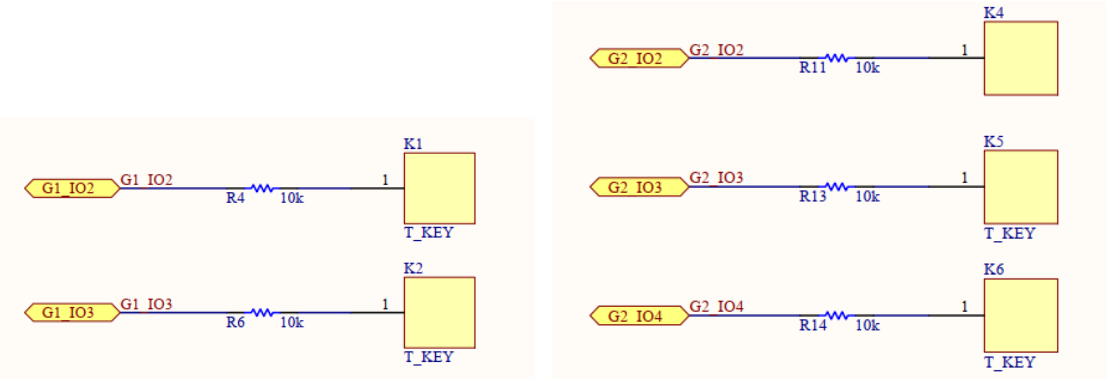


Figure 2-4-2

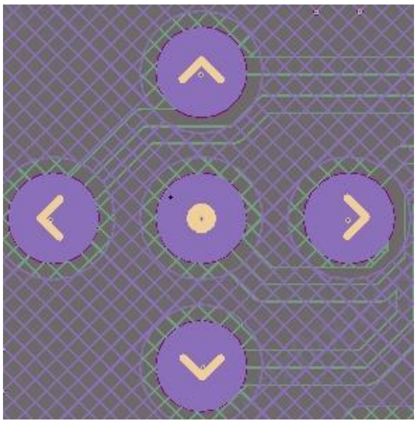


Figure 2-4-3

2.5. Linear Touch

The linear touch sensor shall adopt half-channel electrode method, occupying at least 3 touch channels. The software supports the slider with a resolution range of 3 to 65535. The actual number of GPIO occupied and the selected resolution need to be adjusted according to actual needs.

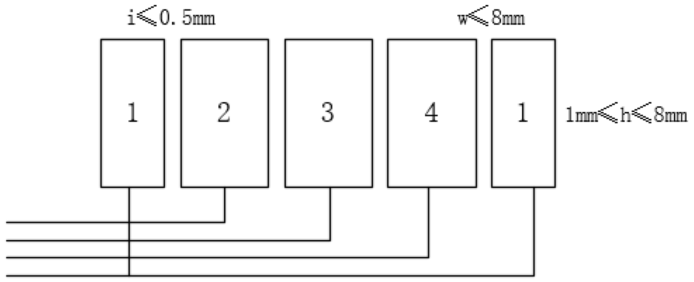


Figure 2-5-1

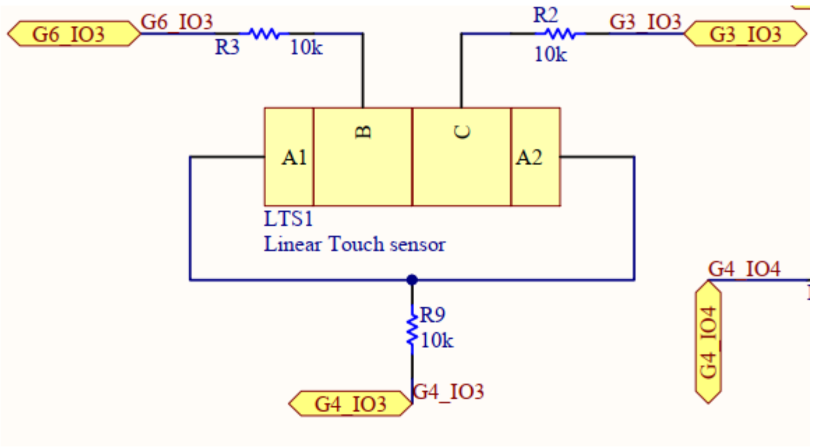


Figure 2-5-2

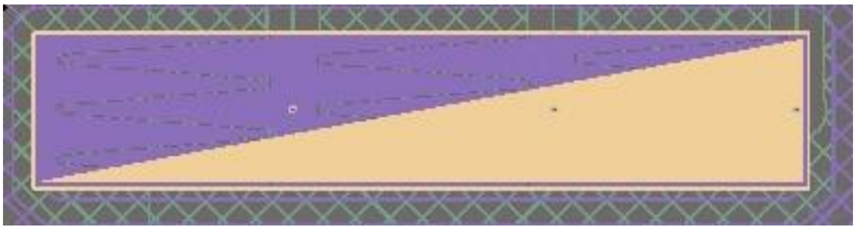


Figure 2-5-3

2.6. Rotary Touch

The rotary touch sensor shall adopt three-electrode rotation, and it is also suitable for five-electrode and eight-electrode rotation, but it occupies at least three touch channels. The software supports the slider with a resolution range of 3 to 65535. The actual number of GPIO occupied and the selected resolution need to be adjusted according to actual needs.

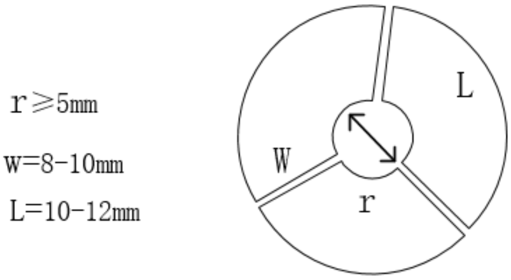


Figure 2-6-1

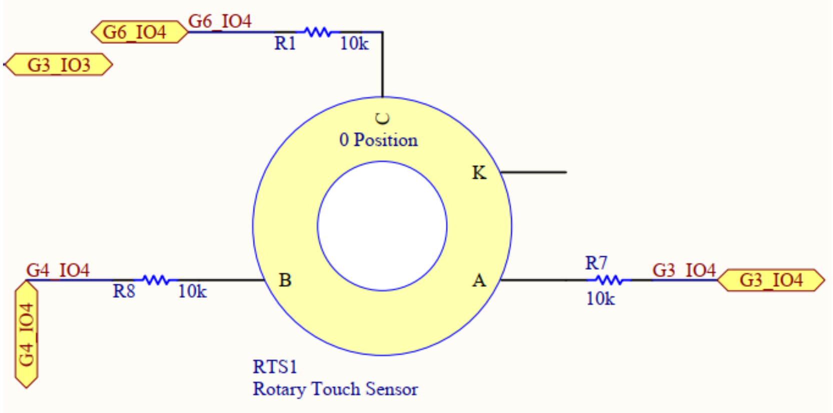


Figure 2-6-2

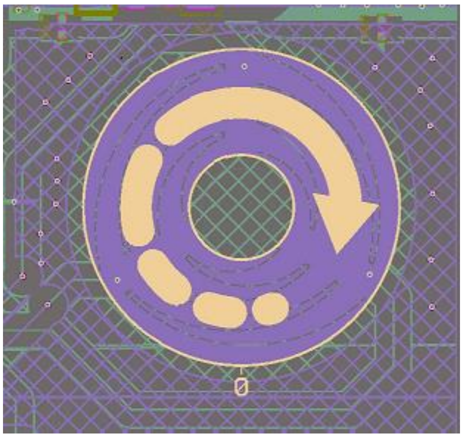


Figure 2-6-3

2.7. Development Board

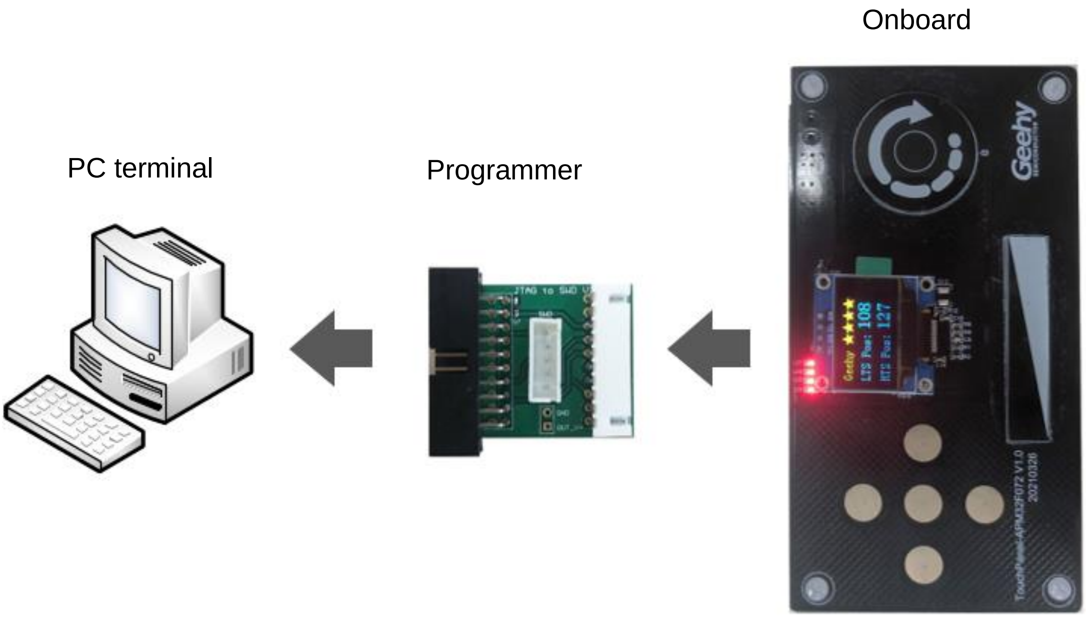


Figure 2-7-1

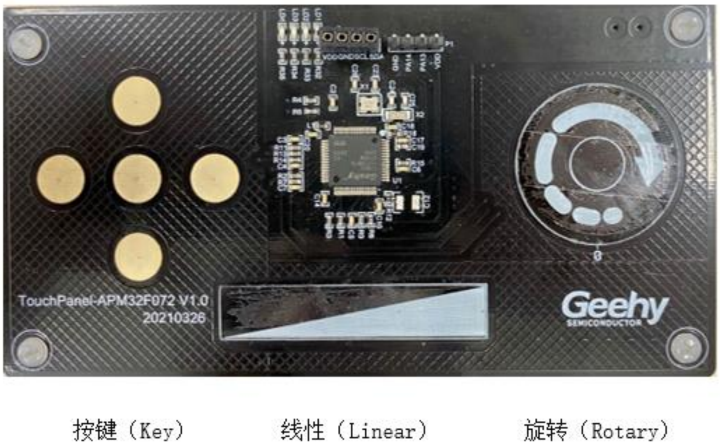
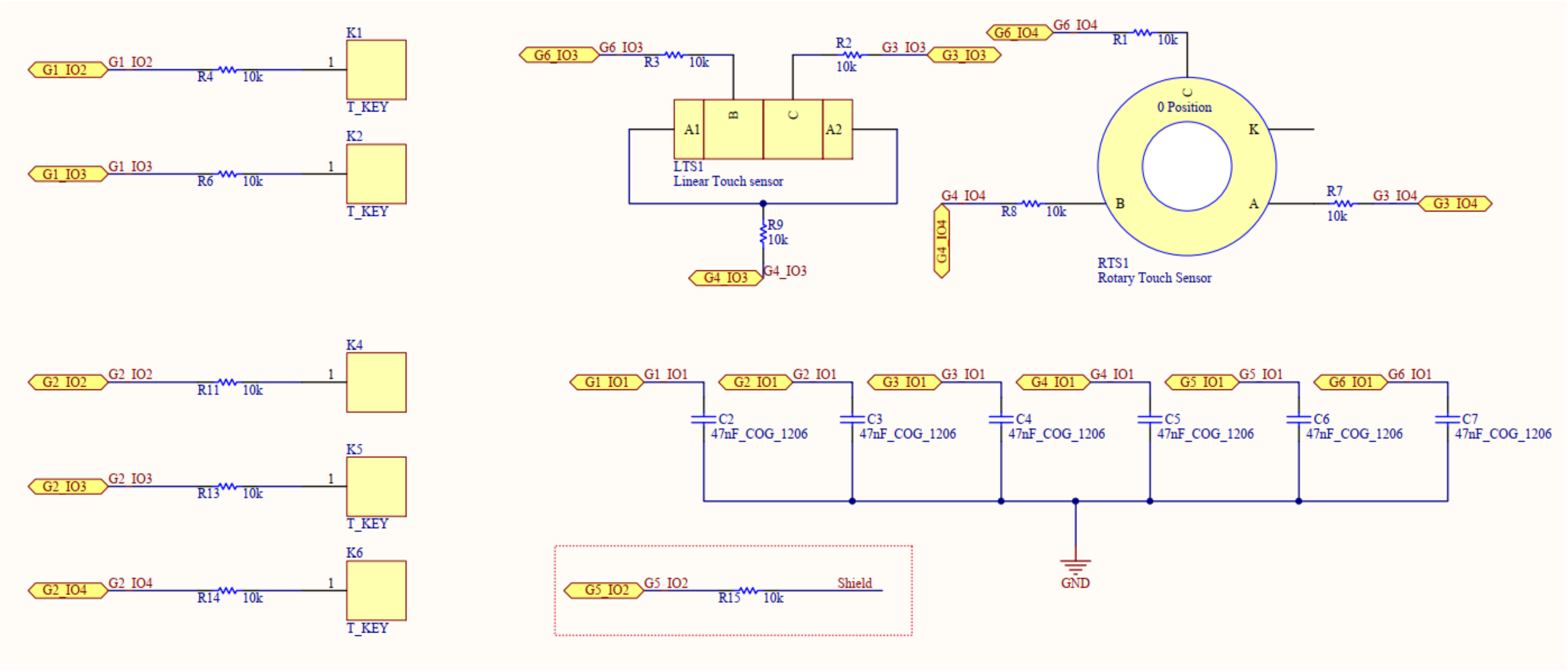
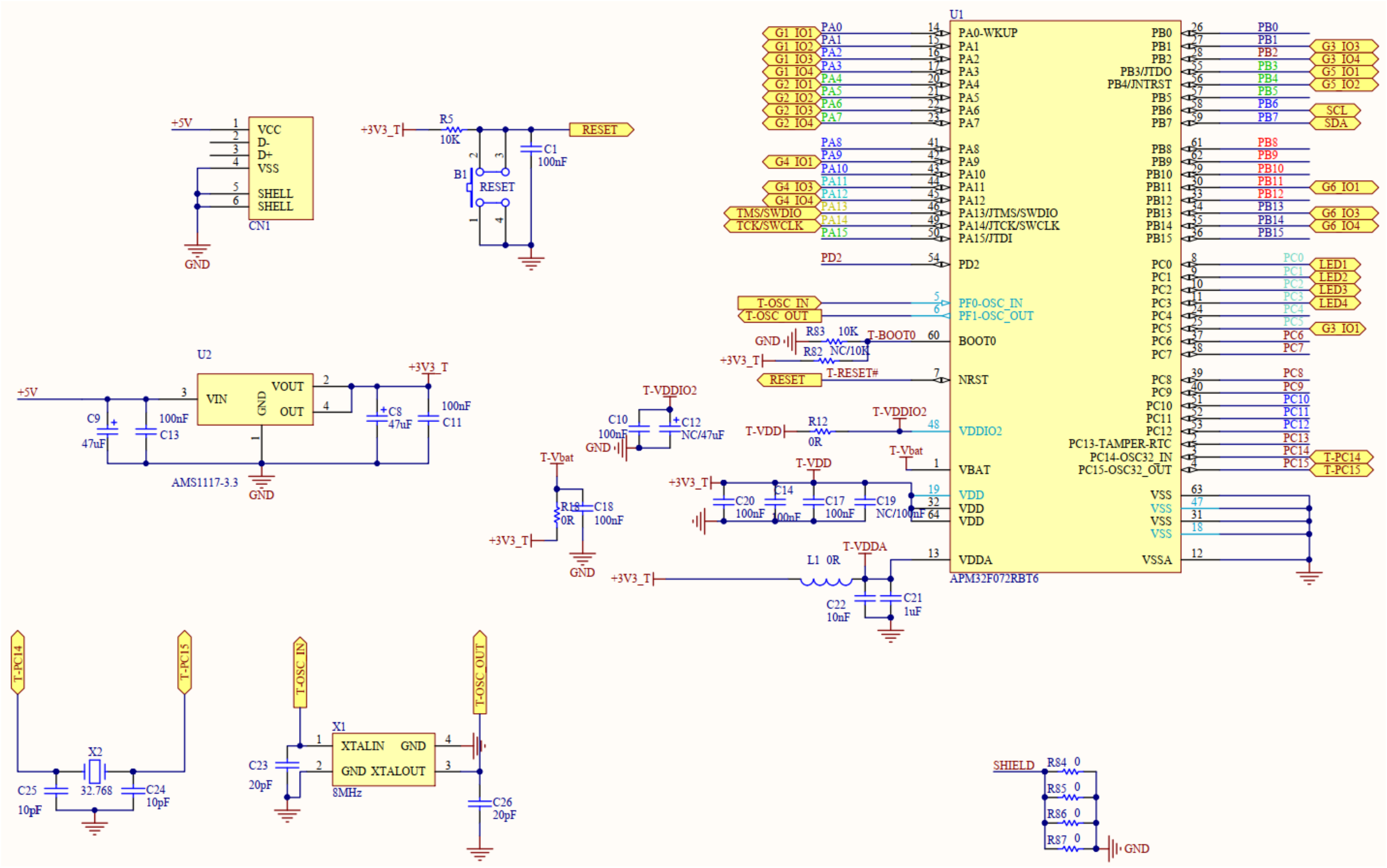


Figure 2-7-2

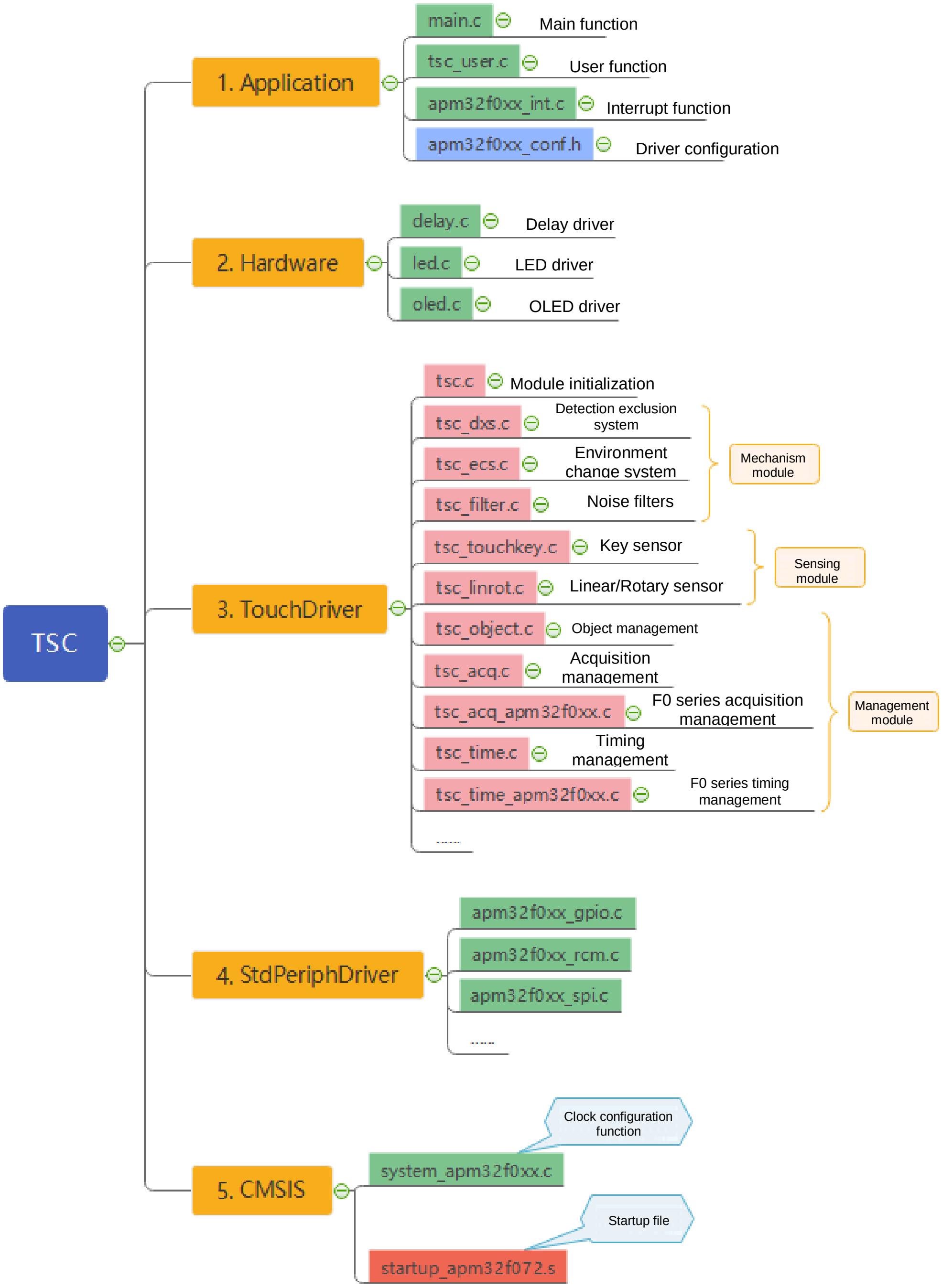
2.8. Schematic Diagram




U1
APM32F072RBT6
PA0-WKUP
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13/JTMS/SWDIO
PA14/JTCK/SWCLK
PA15/JTDI
PD2
PF0-OSC_IN
PF1-OSC_OUT
BOOT0
NRST
VDDIO2
VBAT
VDD
VDD
VDD
VDDA
PB0
PB1
PB2
PB3/JTDO
PB4/JNTRST
PB5
PB6
PB7
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9
PC10
PC11
PC12
PC13-TAMPER-RTC
PC14-OSC32_IN
PC15-OSC32_OUT
VSS
VSS
VSS
VSS
VSSA
G1 IO1
G1 IO2
G1 IO3
G1 IO4
G2 IO1
G2 IO2
G2 IO3
G2 IO4
G4 IO1
G4 IO3
G4 IO4
TMS/SWDIO
TCK/SWCLK
T-OSC IN
T-OSC OUT
G3 IO3
G3 IO4
G5 IO1
G5 IO2
SCL
SDA
G6 IO1
G6 IO3
G6 IO4
LED1
LED2
LED3
LED4
G3 IO1
T-PC14
T-PC15
R83 10K
R82 NC/10K
T-BOOT0
GND
+3V3_T
RESET
T-RESET#
T-VDD
R12 0R
T-VDDIO2
T-Vbat
C20 100nF
C14 100nF
C17 100nF
C19 NC/100nF
T-VDDA
L1 0R
C22 10nF
C21 1uF
CN1
VCC
D-
D+
VSS
SHELL
SHELL
+5V
R5 10K
C1 100nF
B1
RESET
U2
VIN
GND
VOUT
OUT
AMS1117-3.3
C9 47uF
C13 100nF
C8 47uF
C11 100nF
+3V3_T
C10 100nF
C12 NC/47uF
R18 0R
C18 100nF
T-PC14
T-PC15
X2 32.768
C25 10pF
C24 10pF
T-OSC IN
T-OSC OUT
X1
XTALIN
GND
GND
XTALOUT
8MHz
C23 20pF
C26 20pF
SHIELD
R84 0
R85 0
R86 0
R87 0
GND

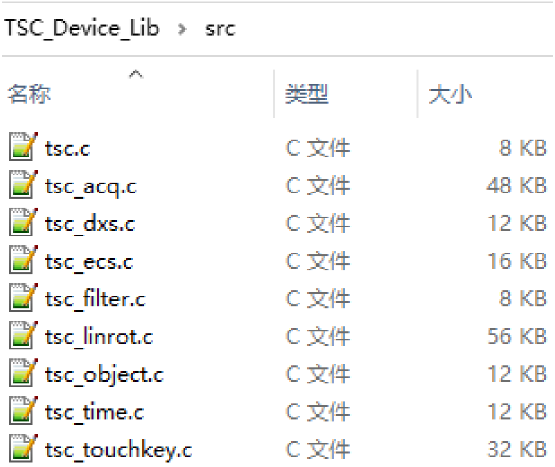
3. Directory Architecture

3.1. Project Directory



3.2. File Directory

a) Source file

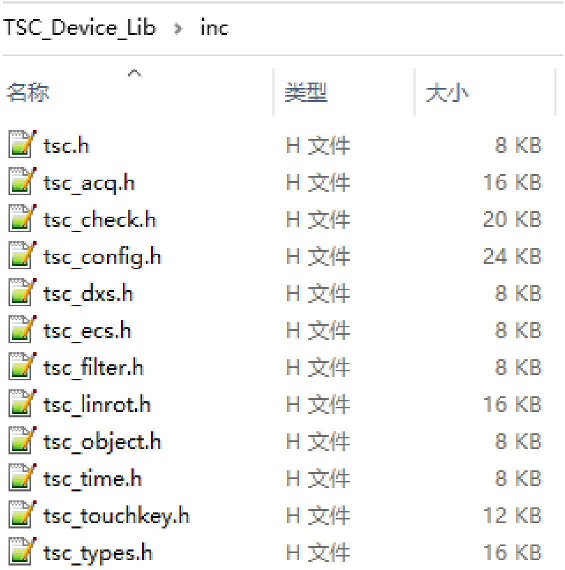


TSC_Device_Lib › src

名称	类型	大小
tsc.c	C 文件	8 KB
tsc_acq.c	C 文件	48 KB
tsc_dxs.c	C 文件	12 KB
tsc_ecs.c	C 文件	16 KB
tsc_filter.c	C 文件	8 KB
tsc_linrot.c	C 文件	56 KB
tsc_object.c	C 文件	12 KB
tsc_time.c	C 文件	12 KB
tsc_touchkey.c	C 文件	32 KB

Figure 3-2-a

b) Header file



TSC_Device_Lib › inc

名称	类型	大小
tsc.h	H 文件	8 KB
tsc_acq.h	H 文件	16 KB
tsc_check.h	H 文件	20 KB
tsc_config.h	H 文件	24 KB
tsc_dxs.h	H 文件	8 KB
tsc_ecs.h	H 文件	8 KB
tsc_filter.h	H 文件	8 KB
tsc_linrot.h	H 文件	16 KB
tsc_object.h	H 文件	8 KB
tsc_time.h	H 文件	8 KB
tsc_touchkey.h	H 文件	12 KB
tsc_types.h	H 文件	16 KB

Figure 3-2-b

3.3. Hierarchical Structure

Figure 3-3 shows the hierarchical differentiation of the entire touch control:

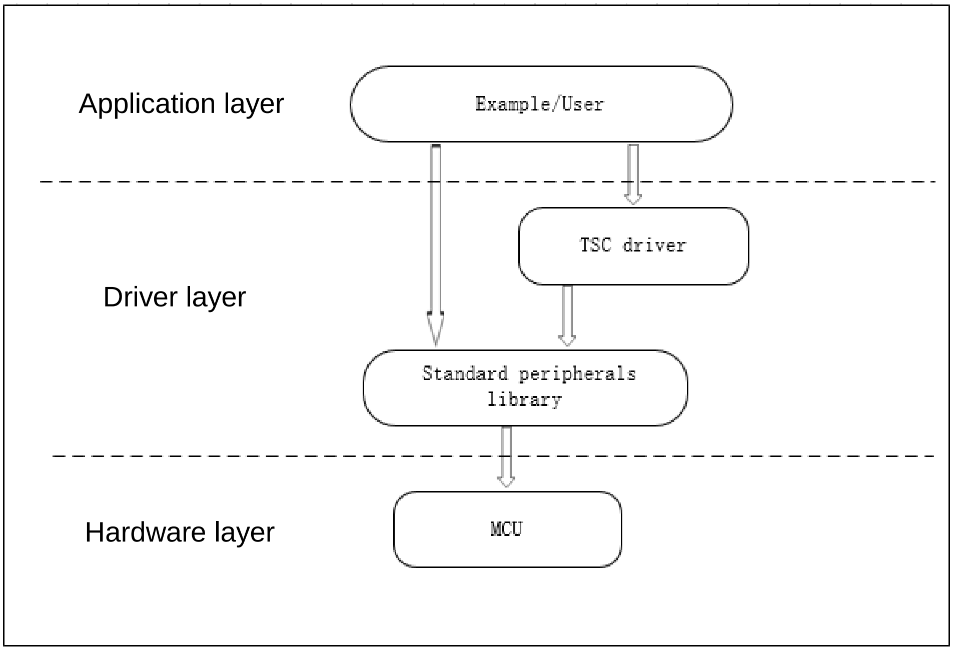


Figure 3-3

Application layer: ..\Example
Driver layer: ..\Library\APM32F0xx_StdPeriphDriver
..\Library\TSC_Device_Lib

3.4. Driver Layer

Figure 3-4 shows the calling relationship of the driver layer touch module in detail:

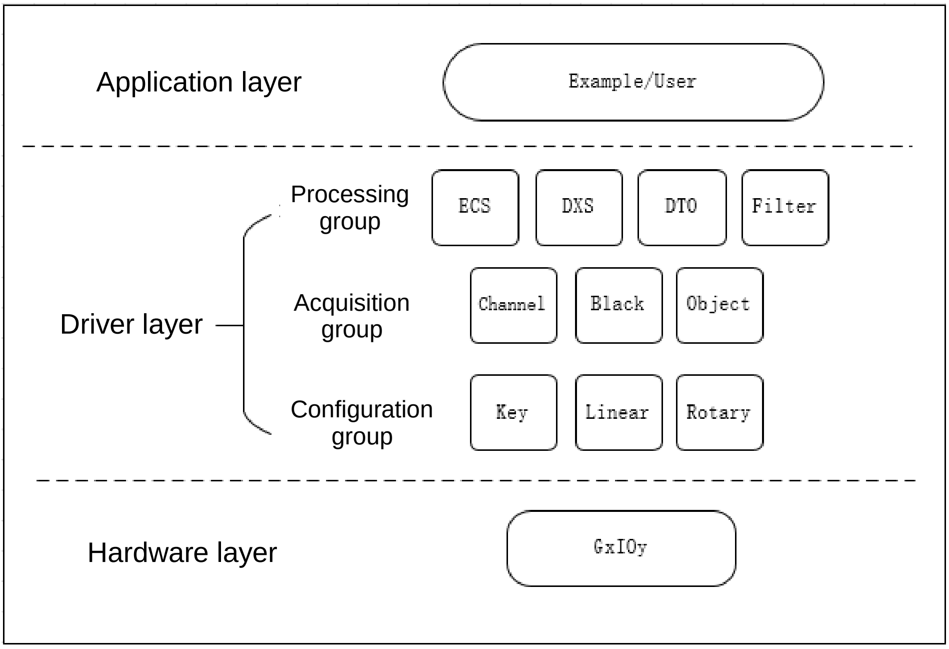


Figure 3-4

Processing group:	tsc_ecs.c \ tsc_dxs.c \ tsc_time.c \ tsc_filter.c \ …
Acquisition group:	tsc.c \ tsc_acq.c \ tsc_object.c \ …
Configuration group:	tsc_touchkey.c \ tsc_linrot.c \ …

4. Software Design

4.1. Channel

Channel is a parameter used to store some basic information.

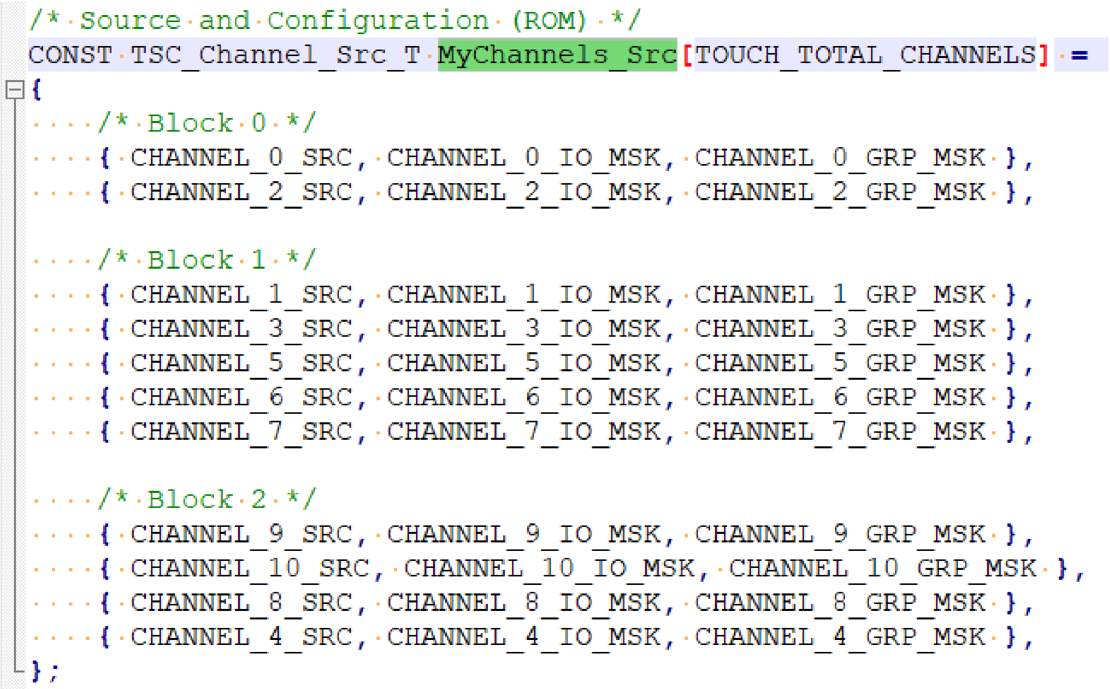
4.1.1. Parameter

#define TOUCH_TOTAL_CHANNELS

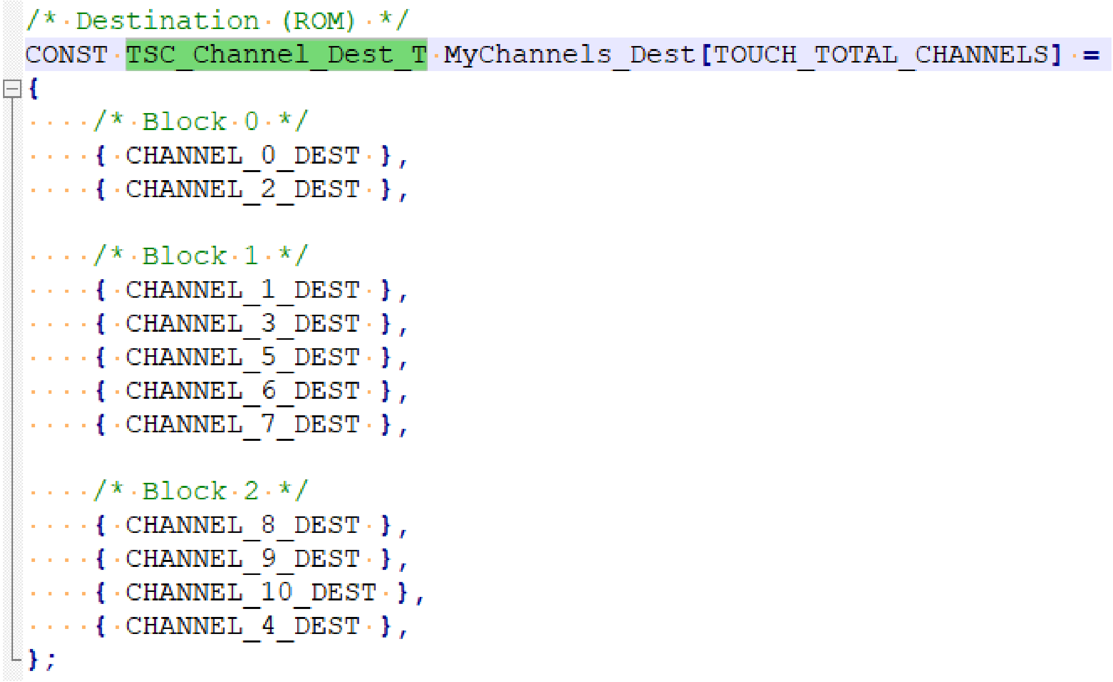
4.1.2. Enumeration Structure

TSC_Channel_Src_T: Channel source structure, including GROUPx value, GxIOy value, and IOGXCR register value.
TSC_Channel_Dest_T: Channel destination structure.
TSC_Channel_Data_T: Channel data structure, including flag bits, reference values, differences, and sampling values.

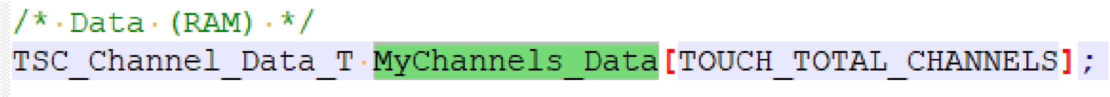
4.1.3. Usage



```
/* Source and Configuration (ROM) */
CONST TSC_Channel_Src_T MyChannels_Src[TOUCH_TOTAL_CHANNELS] =
{
    /* Block 0 */
    { CHANNEL_0_SRC, CHANNEL_0_IO_MSK, CHANNEL_0_GRP_MSK },
    { CHANNEL_2_SRC, CHANNEL_2_IO_MSK, CHANNEL_2_GRP_MSK },

    /* Block 1 */
    { CHANNEL_1_SRC, CHANNEL_1_IO_MSK, CHANNEL_1_GRP_MSK },
    { CHANNEL_3_SRC, CHANNEL_3_IO_MSK, CHANNEL_3_GRP_MSK },
    { CHANNEL_5_SRC, CHANNEL_5_IO_MSK, CHANNEL_5_GRP_MSK },
    { CHANNEL_6_SRC, CHANNEL_6_IO_MSK, CHANNEL_6_GRP_MSK },
    { CHANNEL_7_SRC, CHANNEL_7_IO_MSK, CHANNEL_7_GRP_MSK },

    /* Block 2 */
    { CHANNEL_9_SRC, CHANNEL_9_IO_MSK, CHANNEL_9_GRP_MSK },
    { CHANNEL_10_SRC, CHANNEL_10_IO_MSK, CHANNEL_10_GRP_MSK },
    { CHANNEL_8_SRC, CHANNEL_8_IO_MSK, CHANNEL_8_GRP_MSK },
    { CHANNEL_4_SRC, CHANNEL_4_IO_MSK, CHANNEL_4_GRP_MSK },
};
```



```c
/* Destination (ROM) */
CONST TSC_Channel_Dest_T MyChannels_Dest[TOUCH_TOTAL_CHANNELS] =
{
    /* Block 0 */
    { CHANNEL_0_DEST },
    { CHANNEL_2_DEST },

    /* Block 1 */
    { CHANNEL_1_DEST },
    { CHANNEL_3_DEST },
    { CHANNEL_5_DEST },
    { CHANNEL_6_DEST },
    { CHANNEL_7_DEST },

    /* Block 2 */
    { CHANNEL_8_DEST },
    { CHANNEL_9_DEST },
    { CHANNEL_10_DEST },
    { CHANNEL_4_DEST },
};
```



```c
/* Data (RAM) */
TSC_Channel_Data_T MyChannels_Data[TOUCH_TOTAL_CHANNELS];
```

4.2. Block

One Block needs to acquire multiple channels simultaneously, and multiple blocks can be set.

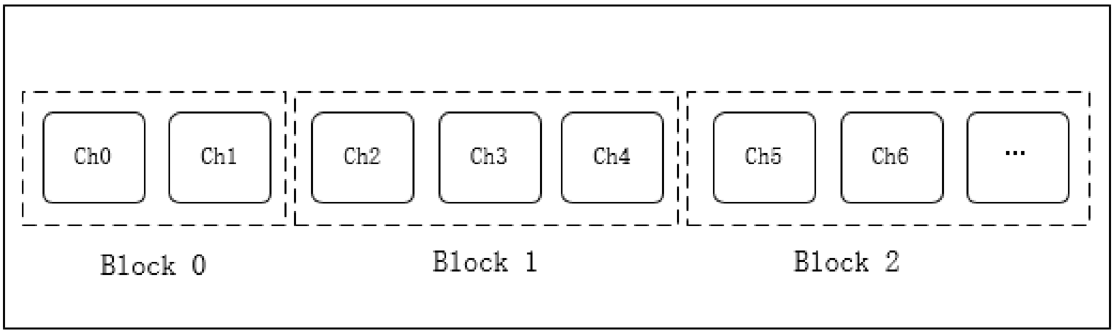


Figure 4-2

4.2.1. Parameter

#define TOUCH_TOTAL_BLOCKS

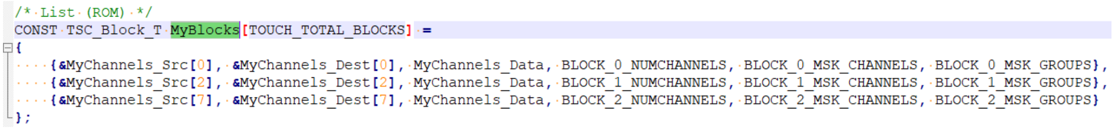
4.2.2. Enumeration Structure

TSC_Block_T: Including the configuration of each channel.

4.2.3. Function Description

Function Name	Purpose
TSC_Acq_ReadBlockResult	Read measured values of all channels and calculate their difference.
TSC_Acq_CalibrateBlock	Calibrate calculated value of block.
TSC_Acq_ConfigBlock	Configure block initialization.
TSC_Acq_StartPerConfigBlock	Start block sampling.
TSC_Acq_WaitBlockEOA	Wait for completion of block sampling.

4.2.4. Usage



```
/* List (ROM) */
CONST TSC_Block_T MyBlocks[TOUCH_TOTAL_BLOCKS] =
{
    {&MyChannels_Src[0], &MyChannels_Dest[0], MyChannels_Data, BLOCK_0_NUMCHANNELS, BLOCK_0_MSK_CHANNELS, BLOCK_0_MSK_GROUPS},
    {&MyChannels_Src[2], &MyChannels_Dest[2], MyChannels_Data, BLOCK_1_NUMCHANNELS, BLOCK_1_MSK_CHANNELS, BLOCK_1_MSK_GROUPS},
    {&MyChannels_Src[7], &MyChannels_Dest[7], MyChannels_Data, BLOCK_2_NUMCHANNELS, BLOCK_2_MSK_CHANNELS, BLOCK_2_MSK_GROUPS}
};
```

4.3. Object

The object is equivalent to a sensor, including key sensor, linear sensor, and rotary sensor supported in the driver file.

4.3.1. Parameter

#define TOUCH_TOTAL_OBJECTS

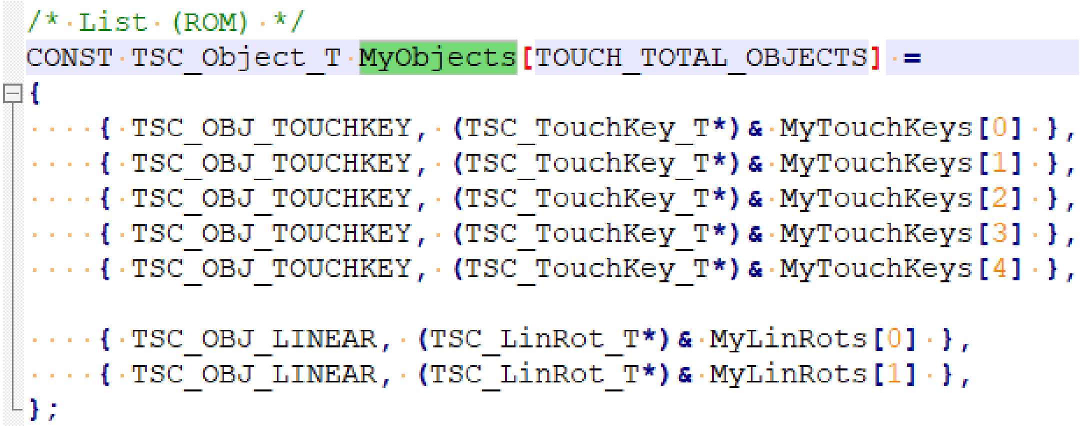
4.3.2. Enumeration Structure

TSC_Object_T: Represents one object.
TSC_ObjectGroup_T: Represents multiple objects of a group.

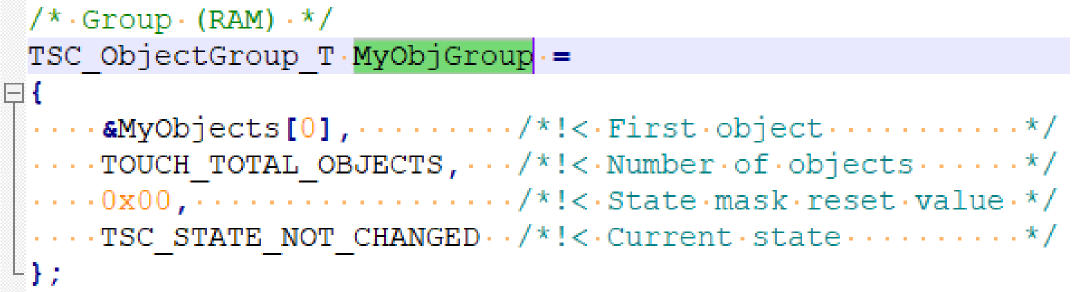
4.3.3. Function Description

Function Name	Purpose
TSC_Obj_ConfigGroup	Configure multiple objects of one group.
TSC_Obj_ProcessGroup	Process multiple objects of one group.
TSC_Obj_ConfigGlobalObj	Configure global object.

4.3.4. Usage



```
/* List (ROM) */
CONST TSC_Object_T MyObjects[TOUCH_TOTAL_OBJECTS] =
{
    { TSC_OBJ_TOUCHKEY, (TSC_TouchKey_T*)& MyTouchKeys[0] },
    { TSC_OBJ_TOUCHKEY, (TSC_TouchKey_T*)& MyTouchKeys[1] },
    { TSC_OBJ_TOUCHKEY, (TSC_TouchKey_T*)& MyTouchKeys[2] },
    { TSC_OBJ_TOUCHKEY, (TSC_TouchKey_T*)& MyTouchKeys[3] },
    { TSC_OBJ_TOUCHKEY, (TSC_TouchKey_T*)& MyTouchKeys[4] },

    { TSC_OBJ_LINEAR, (TSC_LinRot_T*)& MyLinRots[0] },
    { TSC_OBJ_LINEAR, (TSC_LinRot_T*)& MyLinRots[1] },
};
```



```
/* Group (RAM) */
TSC_ObjectGroup_T MyObjGroup =
{
    &MyObjects[0],         /*!< First object           */
    TOUCH_TOTAL_OBJECTS,   /*!< Number of objects      */
    0x00,                  /*!< State mask reset value */
    TSC_STATE_NOT_CHANGED  /*!< Current state          */
};
```

4.4. Key Sensor

Touch Key consists of only one channel, similar to a key, including two states: release and detection.

4.4.1. Parameter

#define TOUCH_TOTAL_TOUCHKEYS

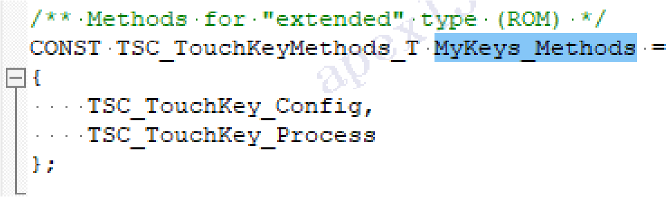
4.4.2. Enumeration Structure

TSC_TouchKeyB_T: Represents basic touch key.
TSC_TouchKey_T: Represents extended touch key, adding methods and state machines.

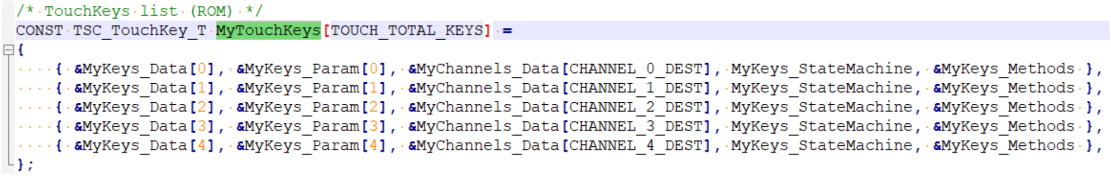
4.4.3. Function Description

Function Name	Purpose
TSC_TouchKey_Config	Configure touch keys.
TSC_TouchKey_Process	Process key state machine.

4.4.4. Usage



```
/** Methods for "extended" type (ROM) */
CONST TSC_TouchKeyMethods_T MyKeys_Methods =
{
    TSC_TouchKey_Config,
    TSC_TouchKey_Process
};
```



```
/* TouchKeys list (ROM) */
CONST TSC_TouchKey_T MyTouchKeys[TOUCH_TOTAL_KEYS] =
{
    { &MyKeys_Data[0], &MyKeys_Param[0], &MyChannels_Data[CHANNEL_0_DEST], MyKeys_StateMachine, &MyKeys_Methods },
    { &MyKeys_Data[1], &MyKeys_Param[1], &MyChannels_Data[CHANNEL_1_DEST], MyKeys_StateMachine, &MyKeys_Methods },
    { &MyKeys_Data[2], &MyKeys_Param[2], &MyChannels_Data[CHANNEL_2_DEST], MyKeys_StateMachine, &MyKeys_Methods },
    { &MyKeys_Data[3], &MyKeys_Param[3], &MyChannels_Data[CHANNEL_3_DEST], MyKeys_StateMachine, &MyKeys_Methods },
    { &MyKeys_Data[4], &MyKeys_Param[4], &MyChannels_Data[CHANNEL_4_DEST], MyKeys_StateMachine, &MyKeys_Methods },
};
```

4.5. Linear/Rotary Sensor

The linear/rotary sensor (LinRot) is composed of a variable number of single channels. The difference between linear and rotary sensors lies in how to organize electrode positions.

Number of channels (n = 1, 3, 4, 5, 6),
Difference coefficient table (MyLinRot0_DeltaCoeff),
Position offset table (TSC_POSOFF_xCH_LIN/ROT_M1/M2/H/D),
Rotary sector calculation (TSC_SCTCOMP_ xCH_LIN/ROT_M1/M2/H/D),
Linear position correction (TSC_POSCORR_
xCH_LIN/ROT_M1/M2/H/D).

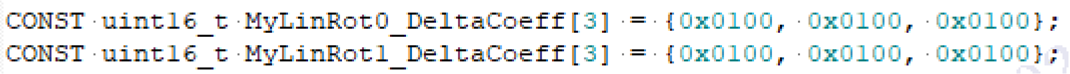
4.5.1. Difference Coefficient Table

The number of channels used to adjust linear sensors is unlimited and they can share the same coefficient table.

The coefficient value is 16 bits. MSB is the integer part, and LSB is the decimal part, for example:

When the coefficient value is 1.50:
0x01 to the MSB
0x0D to the LSB (0.50 x 256 = 12.8 -> rounded to 13 = 0x0D)
When the coefficient value is 0.80:
0x00 to the MSB
0xCD to the LSB (0.80 x 256 = 204.8 -> rounded to 205 = 0xCD)

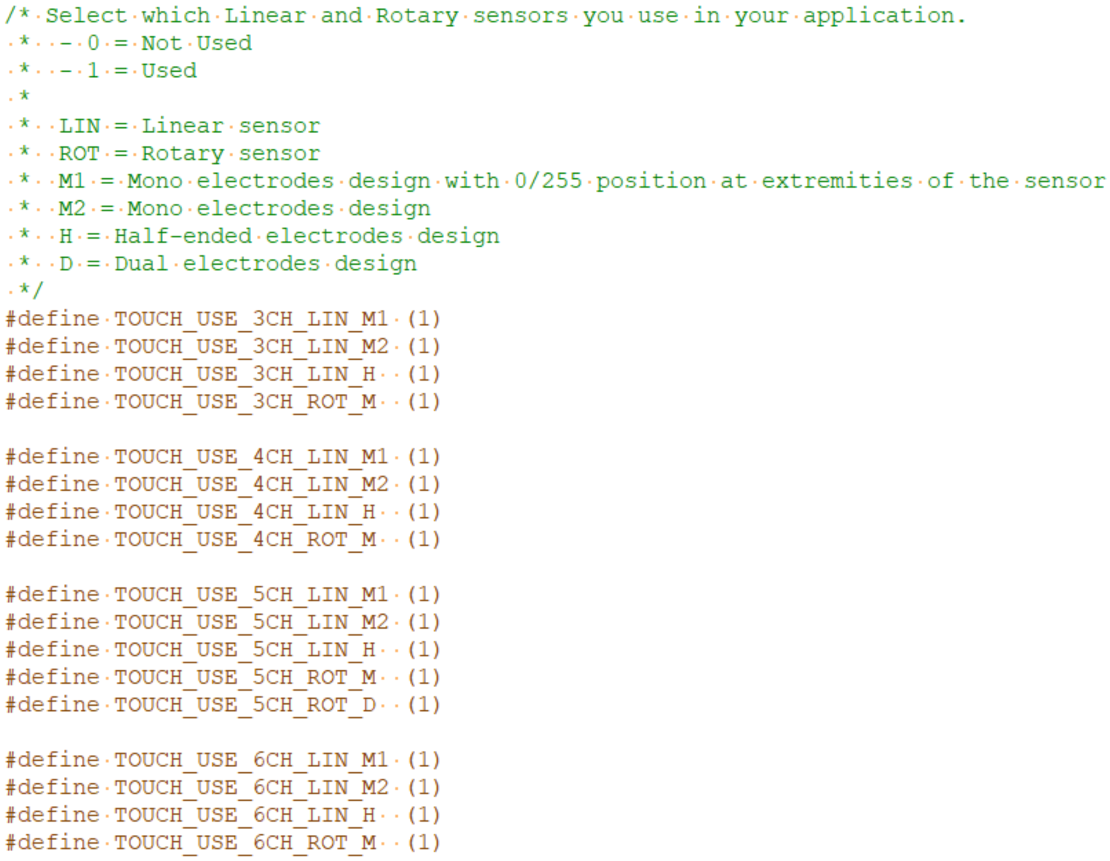
The application coefficient value is 1:



```
CONST uint16_t MyLinRot0_DeltaCoeff[3] = {0x0100, 0x0100, 0x0100};
CONST uint16_t MyLinRot1_DeltaCoeff[3] = {0x0100, 0x0100, 0x0100};
```

4.5.2. Position Offset Table

Select through relevant macros in the “tsc_conf.h” file.



```
/* Select which Linear and Rotary sensors you use in your application.
 *  - 0 = Not Used
 *  - 1 = Used
 *
 *  LIN = Linear sensor
 *  ROT = Rotary sensor
 *  M1 = Mono electrodes design with 0/255 position at extremities of the sensor
 *  M2 = Mono electrodes design
 *  H = Half-ended electrodes design
 *  D = Dual electrodes design
 */
#define TOUCH_USE_3CH_LIN_M1 (1)
#define TOUCH_USE_3CH_LIN_M2 (1)
#define TOUCH_USE_3CH_LIN_H  (1)
#define TOUCH_USE_3CH_ROT_M  (1)

#define TOUCH_USE_4CH_LIN_M1 (1)
#define TOUCH_USE_4CH_LIN_M2 (1)
#define TOUCH_USE_4CH_LIN_H  (1)
#define TOUCH_USE_4CH_ROT_M  (1)

#define TOUCH_USE_5CH_LIN_M1 (1)
#define TOUCH_USE_5CH_LIN_M2 (1)
#define TOUCH_USE_5CH_LIN_H  (1)
#define TOUCH_USE_5CH_ROT_M  (1)
#define TOUCH_USE_5CH_ROT_D  (1)

#define TOUCH_USE_6CH_LIN_M1 (1)
#define TOUCH_USE_6CH_LIN_M2 (1)
#define TOUCH_USE_6CH_LIN_H  (1)
#define TOUCH_USE_6CH_ROT_M  (1)
```

4.5.3. Electrode Position

It can be divided into single-channel electrode design, dual -channel electrode design, and half-channel electrode design according to the design method.

a) Single-channel electrode

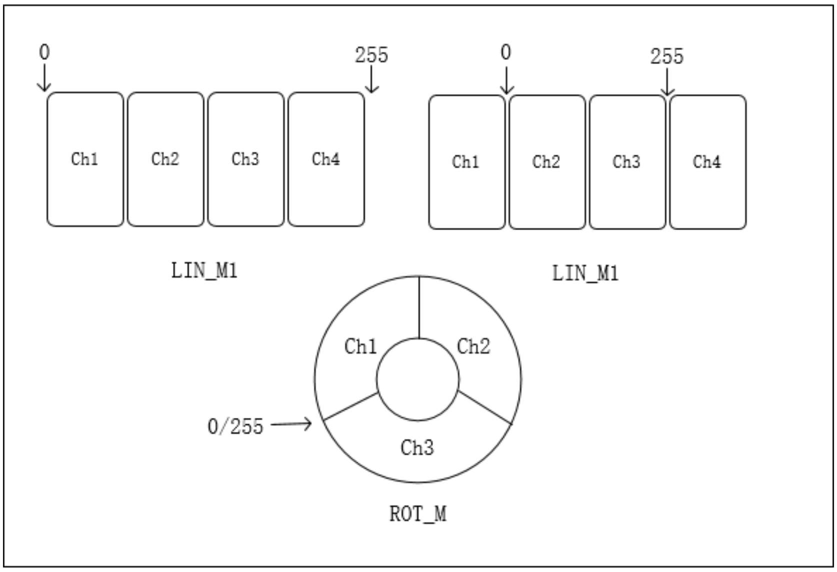


Figure 4-5-3-a

The number of electrodes used is equal to the number of channels, which is suitable for linear and rotary sensors.
Including (CH1, CH2, CH3)
(CH1, CH2, CH3, CH4)
(CH1, CH2, CH3, CH4, CH5)
LIN_M1: The design value of the linear sensor is 0 at the head and 255 at the tail.
LIN_M2: The design value of the linear sensor is 0 between the first and second electrodes, and 255 between the last two electrodes.
ROT_M: The design value of the rotary sensor is 0 at the head and 255 at the tail.

b) Dual-channel electrode

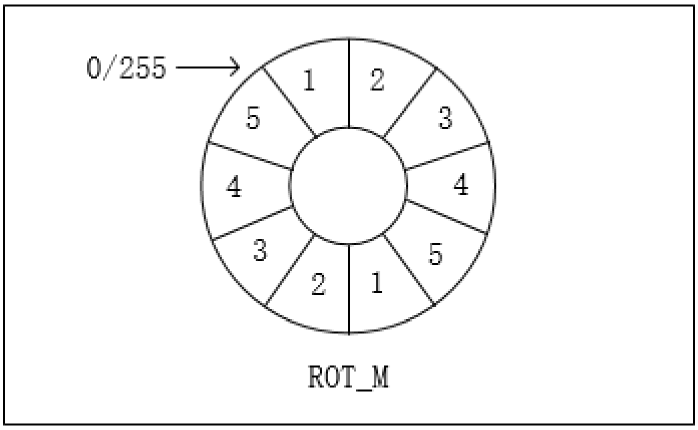


Figure 4-5-3-b

Adopt electrode replication cross combination to increase touch area, which is suitable for rotary sensors. (Only support 5 channels)

Including (CH1, CH2, CH3, CH4, CH5, CH3, CH1, CH4, CH2, CH5)
(CH1, CH2, CH3, CH4, CH5, CH1, CH3, CH5, CH2, CH4)
(CH1, CH2, CH3, CH4, CH5, CH2, CH4, CH1, CH3, CH5)

ROT_D: The design value of the rotary sensor is 0 at the head and 255 at the tail.

c) Half-channel electrode

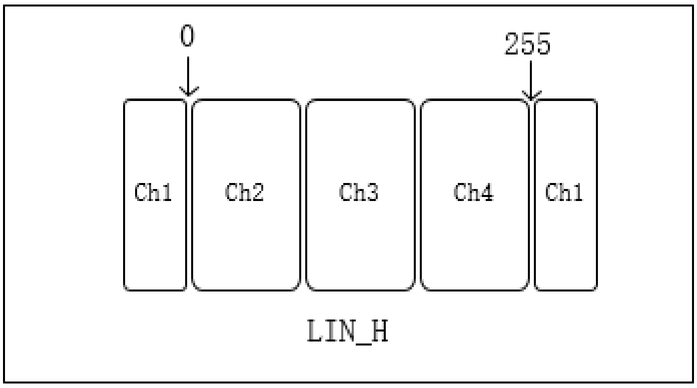


Figure 4-5-3-c

The first electrode is divided into two parts, one is at the foremost and the other is at the rearmost end, only applicable to linear sensors. It is better than single channel.

Including (ch1, CH2, CH3, ch1)
(ch1, CH2, CH3, CH4, ch1)
(ch1, CH2, CH3, CH4, CH5, ch1)

LIN_H: The design value of the linear sensor is 0 between the first and second electrodes, and 255 between the last two electrodes.

4.5.4. Parameter

#define TOUCH_TOTAL_LINROTS

4.5.5. Enumeration Structure

TSC_LinRotB_T: Represents basic linear rotation touch.

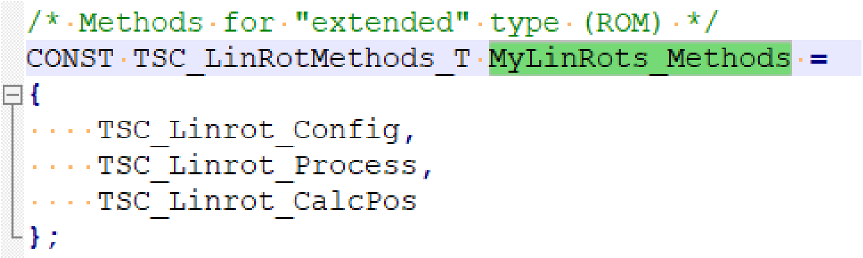
TSC_LinRot_T: Represents extended linear rotation touch, adding methods and state machines.

4.5.6. Function Description

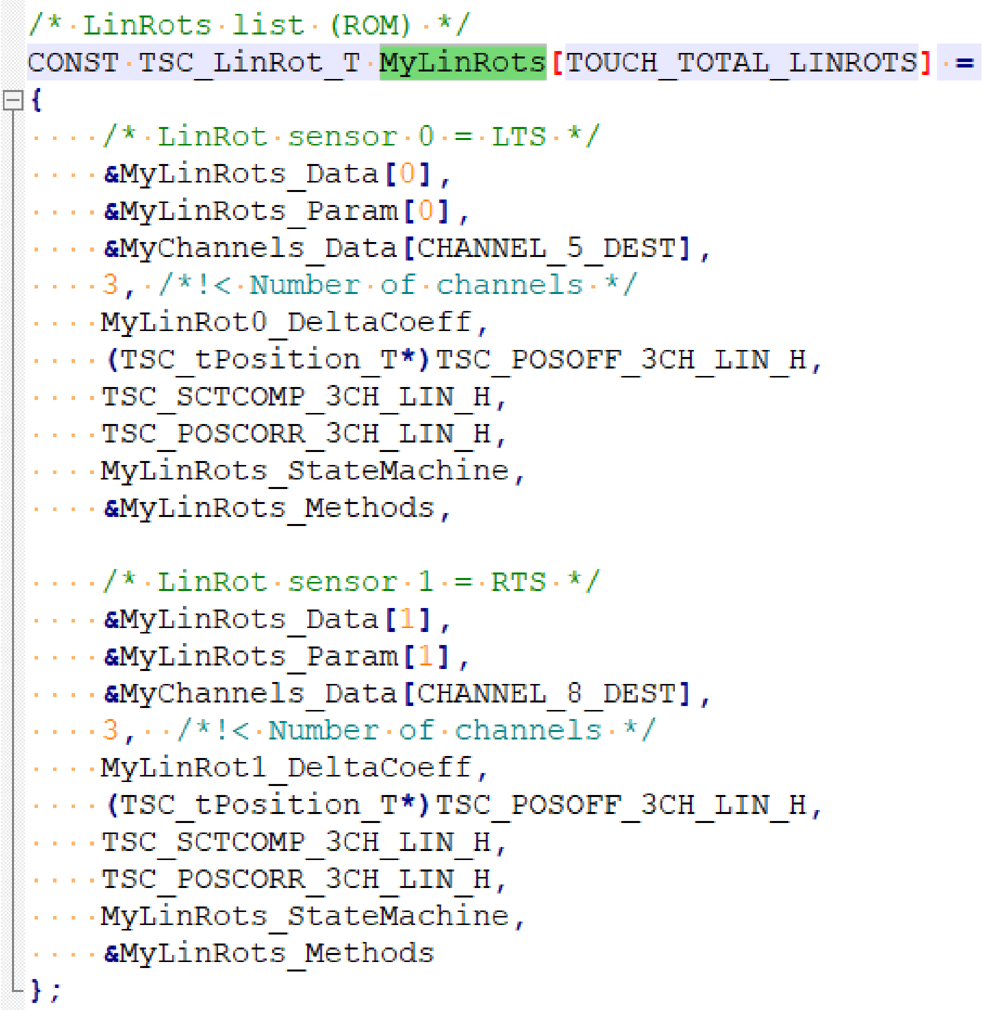
Function Name	Purpose
TSC_Linrot_Config	Configure linear rotation touch.
TSC_Linrot_Process	Process linear rotation state machine.

TSC_Linrot_CalcPos	Calculate the position of linear rotation.

4.5.7. Usage



```
/* Methods for "extended" type (ROM) */
CONST TSC_LinRotMethods_T MyLinRots_Methods =
{
    TSC_Linrot_Config,
    TSC_Linrot_Process,
    TSC_Linrot_CalcPos
};
```



```
/* LinRots list (ROM) */
CONST TSC_LinRot_T MyLinRots[TOUCH_TOTAL_LINROTS] =
{
    /* LinRot sensor 0 = LTS */
    &MyLinRots_Data[0],
    &MyLinRots_Param[0],
    &MyChannels_Data[CHANNEL_5_DEST],
    3, /*!< Number of channels */
    MyLinRot0_DeltaCoeff,
    (TSC_tPosition_T*)TSC_POSOFF_3CH_LIN_H,
    TSC_SCTCOMP_3CH_LIN_H,
    TSC_POSCORR_3CH_LIN_H,
    MyLinRots_StateMachine,
    &MyLinRots_Methods,

    /* LinRot sensor 1 = RTS */
    &MyLinRots_Data[1],
    &MyLinRots_Param[1],
    &MyChannels_Data[CHANNEL_8_DEST],
    3,  /*!< Number of channels */
    MyLinRot1_DeltaCoeff,
    (TSC_tPosition_T*)TSC_POSOFF_3CH_LIN_H,
    TSC_SCTCOMP_3CH_LIN_H,
    TSC_POSCORR_3CH_LIN_H,
    MyLinRots_StateMachine,
    &MyLinRots_Methods
};
```

4.6. Detection Time Out (DTO)

This mechanism provides a maximum duration for the detection status of all sensors to avoid accidental contact with liquids or other obstacles. When exceeding this time, the sensor will automatically recalibrate, even if liquids and obstacles still exist.

4.6.1. Parameter

#define TOUCH_DTO

4.6.2. Function Description

Function Name	Purpose
TSC_TouchKey_ReadTimeForDTO	Read the timeout value of the key.
TSC_Linrot_ReadTimeForDTO	Read the linear/rotary timeout value.

4.6.3. Usage

DTO is automatically executed in the driver, and the switch can be selected by defining parameters through macros, not requiring calling this module function.

4.7. Timing Management

Increase the time in a timing way by defining global variables and compare them to determine whether the delay effect is achieved.
At the same time, it can be used as the time benchmark for ECS and DTO mechanisms, and can also be used at the application layer.

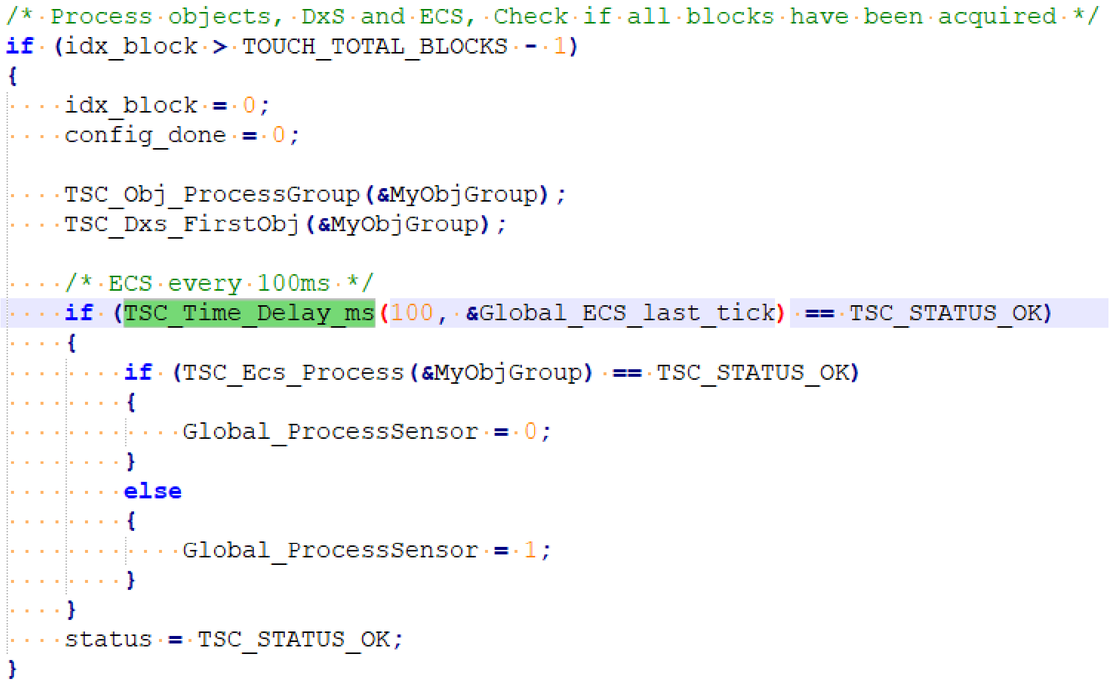
4.7.1. Parameter

#define TOUCH_TICK_FREQ

4.7.2. Function Description

Function Name	Purpose
TSC_Time_Config	Initialize start timing
TSC_Time_Delay_ms	Delay ms
TSC_Time_Delay_sec	Delay sec
TSC_Time_ProcessInterrupt	Timing interrupt function

4.7.3. Usage



```c
/* Process objects, DxS and ECS, Check if all blocks have been acquired */
if (idx_block > TOUCH_TOTAL_BLOCKS - 1)
{
    idx_block = 0;
    config_done = 0;

    TSC_Obj_ProcessGroup(&MyObjGroup);
    TSC_Dxs_FirstObj(&MyObjGroup);

    /* ECS every 100ms */
    if (TSC_Time_Delay_ms(100, &Global_ECS_last_tick) == TSC_STATUS_OK)
    {
        if (TSC_Ecs_Process(&MyObjGroup) == TSC_STATUS_OK)
        {
            Global_ProcessSensor = 0;
        }
        else
        {
            Global_ProcessSensor = 1;
        }
    }
    status = TSC_STATUS_OK;
}
```

4.8. Detection Exclusion System (DXS)

This mechanism is mainly used to prevent false triggering between multiple sensors. When the distance between sensors is too close or the sensitivity of the sensors is too high, it will cause false triggering. Therefore, the first sensor in the sensor group has the highest priority and enters the DETECT state, other sensors will be “blocked” and enter the “TOUCH” state. The so-called first sensor depends on the position of the sensor in the DXS group and the processing sequence of the DXS group, as shown in the following figure:

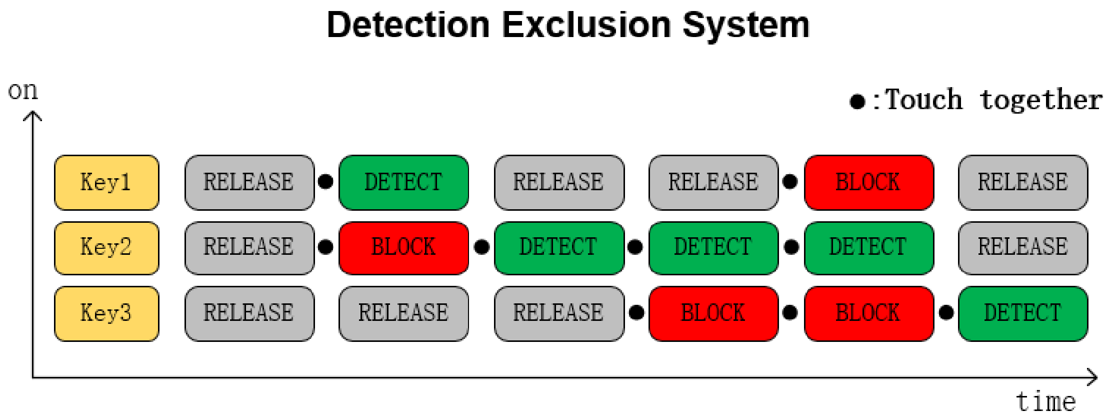

Detection Exclusion System
on
●:Touch together
Key1 RELEASE ● DETECT RELEASE RELEASE ● BLOCK RELEASE
Key2 RELEASE ● BLOCK ● DETECT ● DETECT ● DETECT RELEASE
Key3 RELEASE RELEASE RELEASE ● BLOCK ● BLOCK ● DETECT
time

Figure 4-7

4.8.1. Parameter

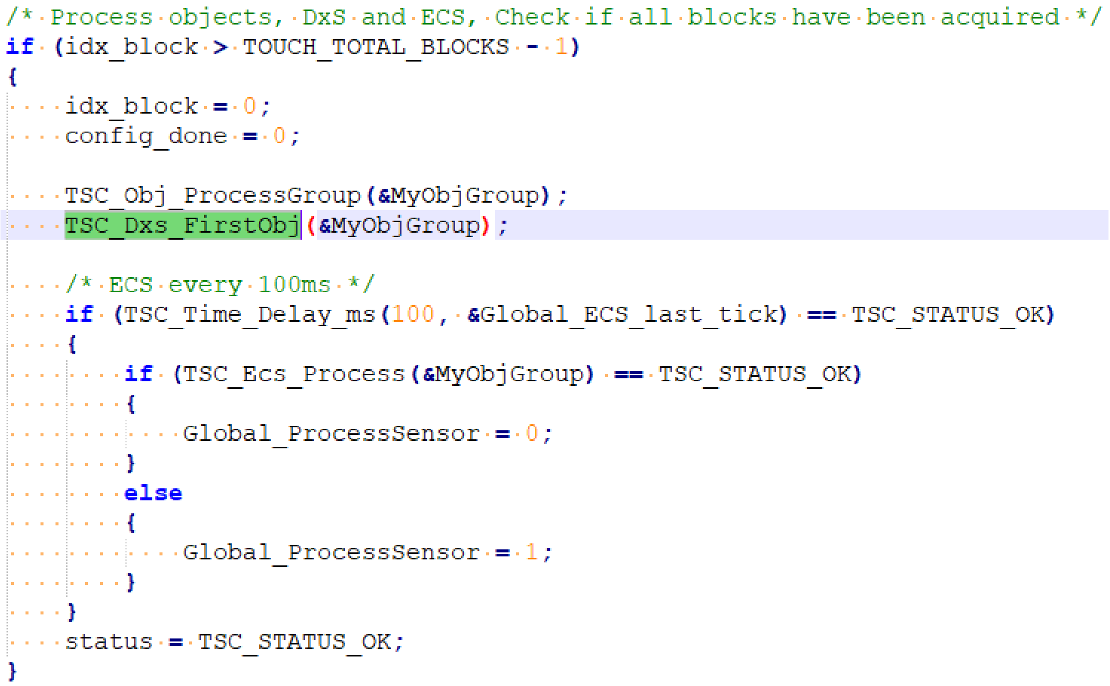
#define TOUCH_USE_DXS

4.8.2. Function Description

Function Name	Purpose
TSC_Dxs_FirstObj	Detect the first object

4.8.3. Usage

DXS can be executed through user definition or executed in interrupt programs, but before execution, it is necessary to determine whether the sensor state machine has completed processing, namely execute TSC_Obj_ProcessGroup function.



```c
/* Process objects, DxS and ECS, Check if all blocks have been acquired */
if (idx_block > TOUCH_TOTAL_BLOCKS - 1)
{
    idx_block = 0;
    config_done = 0;

    TSC_Obj_ProcessGroup(&MyObjGroup);
    TSC_Dxs_FirstObj(&MyObjGroup);

    /* ECS every 100ms */
    if (TSC_Time_Delay_ms(100, &Global_ECS_last_tick) == TSC_STATUS_OK)
    {
        if (TSC_Ecs_Process(&MyObjGroup) == TSC_STATUS_OK)
        {
            Global_ProcessSensor = 0;
        }
        else
        {
            Global_ProcessSensor = 1;
        }
    }
    status = TSC_STATUS_OK;
}
```

4.9. Environment Change System (ECS)

The environment includes external factors such as power supply voltage, temperature, and air humidity. Change in any factor will affect the measurement signal. In order to improve the impact of the environment, ECS processing is adopted. ECS is based on the first-order low-pass filtering principle:

$$Y(n) = KX(n) + (1-K)Y(n-1)$$

Y = Filtered output value (reference value).
X = Current sampling value (last measured value).
K = Filter coefficient.
The larger the K value is, the faster the response is. The response speed can be configured through parameters.
When it is in timeout detection mode or is touched for a long time, ECS will be disabled, and then Yn=Y (n-1).

4.9.1. Parameter

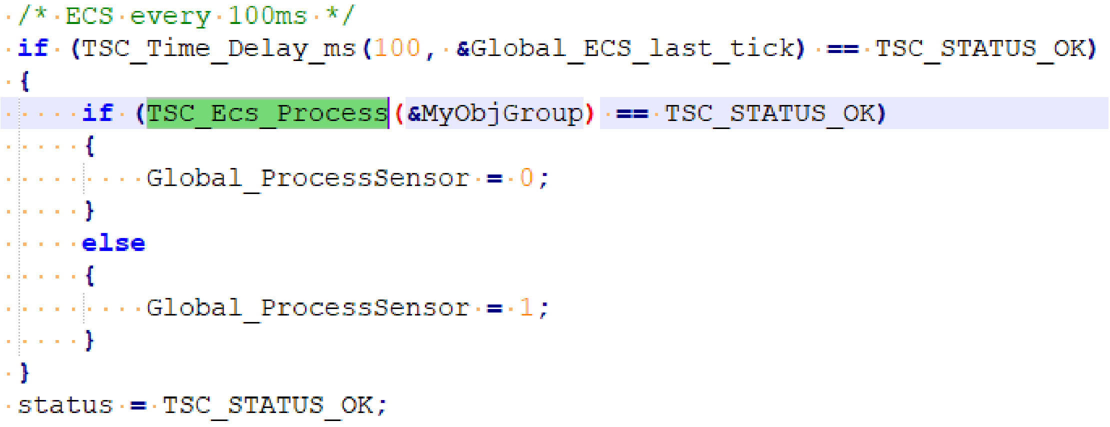
```
#define   TSLPRM_ECS_K_SLOW
#define   TSLPRM_ECS_K_FAST
#define   TSLPRM_ECS_DELAY
```

4.9.2. Function Description

Function Name	Purpose
TSC_Ecs_Process	ECS processing function (user)
TSC_Ecs_CalculateK	Calculate the coefficient K.
TSC_Ecs_ProcessK	Process the coefficient K.

4.9.3. Usage

ECS can be executed at user-defined time interval or executed in interrupt programs, but before execution, it is necessary to determine whether the sensor state machine has completed processing.



```
 /* ECS every 100ms */
 if (TSC_Time_Delay_ms(100, &Global_ECS_last_tick) == TSC_STATUS_OK)
 {
     if (TSC_Ecs_Process(&MyObjGroup) == TSC_STATUS_OK)
     {
         Global_ProcessSensor = 0;
     }
     else
     {
         Global_ProcessSensor = 1;
     }
 }
 status = TSC_STATUS_OK;
```

5. Version History

Table 1 Document Revision History

Date	Revision	Change
2023.02.28	V1.0	Initial version