文档版本	V1.0
发布日期	20210608

APT32F102 TOUCH 应用指南

目录

1 概述

本文介绍了在APT32F102中使用TOUCH的应用范例。

2. 适用的硬件

该例程使用于 APT32F102x 系列学习板

3. 应用方案代码说明

3.1 TOUCH 配置

基于 APT32F102 完整的库文件系统，对 TOUCH KEY 进行配置。

- **硬件配置：**

支持 17 个扫描通道的电容式触摸按键，基于电荷转移的检测技术，实现按键或者触摸滑条等应用，通过提供的触摸算法库使用该功能。

使用触摸功能后,会占用 102x 中 Coret 定时模块,用户不能在程序中配置 Coret。

Touch 模块在低功耗下的扫描启动信号由 LPT 提供,所以开启低功耗模式后,将占用 LPT 模块,用户不能再配置 LPT 模块

- **触摸模块框图：**

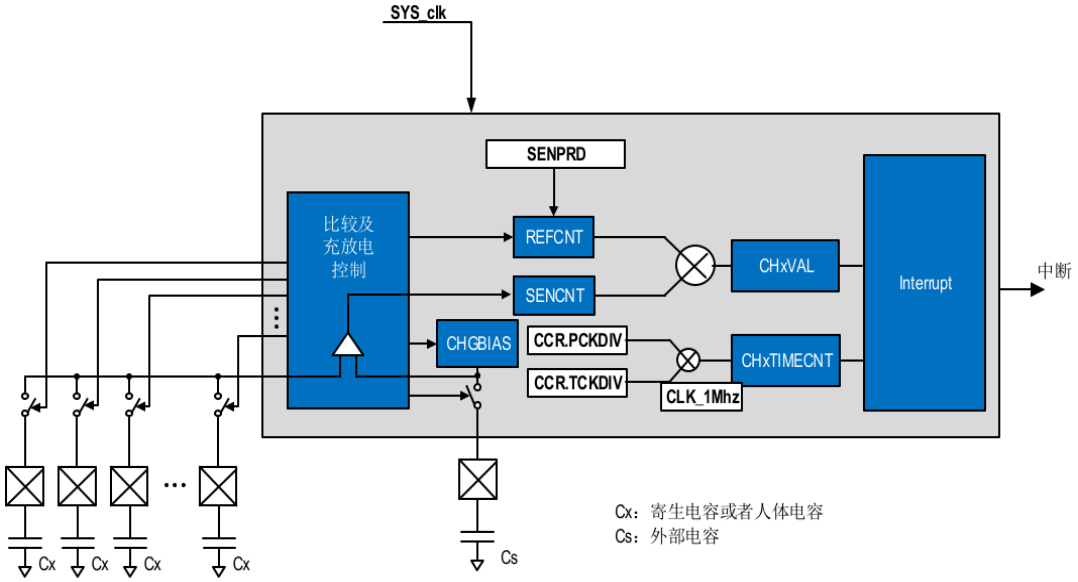

图 3.1.1 功能框图

- **触摸硬件原理图：**

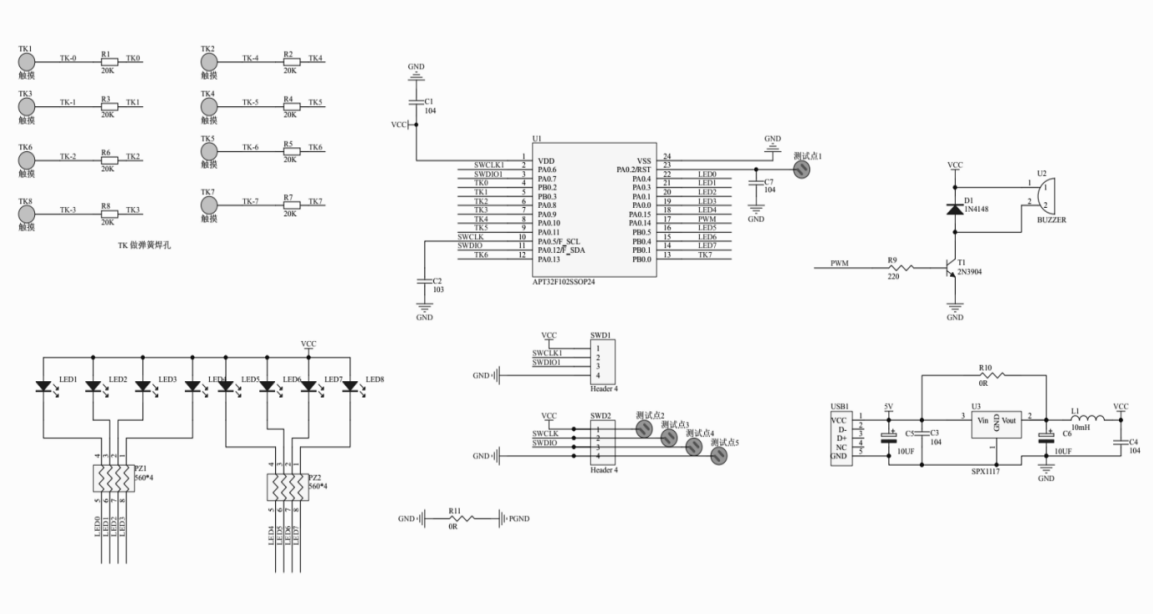

图 3.1.2 硬件原理图

- **注意：**

使用触摸功能 C0(PA0.5)必须接一个 103 电容到地。

要求给 MCU 供电的电源要稳定，建议加稳压电路。

TouchKey 参考电压选择：

1. 若选择 FVR 做参考 PIN23(PA0.2)需要预留出来接 104 电容到地。

2. 若选择 VDD 做参考 PIN23 可以做其他功能。

- **软件配置：**

可在 apt32f102_initial.c 文件中 TK_CONFIG 进行初始化的配置

```
/*************************************************************/
//TK config
//EntryParameter:NONE
//ReturnValue:NONE
/*************************************************************/
void TK_CONFIG(void)
{
    tk_init();
}
```

- **代码说明：**

  tk_init(); ----用于初始化 TK，具体函数实现封装在触摸库中。

3.2 TOUCH 常用参数配置

此函数是配置 TOUCH 触摸内部参数，已经在 tk_init()中调用。

```
void tk_parameter_init(void)
{
/*******************************************************
//TK parameter define
*******************************************************/
  TK_IO_ENABLE=TCH_EN(4)|TCH_EN(5)|TCH_EN(6)|TCH_EN(7);
                                                    //TK IO ENABLE Bit16-->Bit0;0=DISABLE 1=ENABLE
  TK_senprd[0]=50;                                  //TCH0 scan period = TCH0 sens
  TK_senprd[1]=50;                                  //TCH1 scan period = TCH1 sens
  TK_senprd[2]=50;                                  //TCH2 scan period = TCH2 sens
  TK_senprd[3]=50;                                  //TCH3 scan period = TCH3 sens
  TK_senprd[4]=50;                                  //TCH4 scan period = TCH4 sens
  TK_senprd[5]=50;                                  //TCH5 scan period = TCH5 sens
  TK_senprd[6]=50;                                  //TCH6 scan period = TCH6 sens
  TK_senprd[7]=50;                                  //TCH7 scan period = TCH7 sens
  TK_senprd[8]=50;                                  //TCH8 scan period = TCH8 sens
  TK_senprd[9]=50;                                  //TCH9 scan period = TCH9 sens
  TK_senprd[10]=50;                                 //TCH10 scan period = TCH10 sens
  TK_senprd[11]=50;                                 //TCH11 scan period = TCH11 sens
  TK_senprd[12]=50;                                 //TCH12 scan period = TCH12 sens
  TK_senprd[13]=50;                                 //TCH13 scan period = TCH13 sens
  TK_senprd[14]=50;                                 //TCH14 scan period = TCH14 sens
  TK_senprd[15]=50;                                 //TCH15 scan period = TCH15 sens
  TK_senprd[16]=50;                                 //TCH16 scan period = TCH16 sens
  TK_Triggerlevel[0]=60;                            //TCH0 TK_Trigger level
  TK_Triggerlevel[1]=60;                            //TCH1 TK_Trigger level
  TK_Triggerlevel[2]=60;                            //TCH2 TK_Trigger level
  TK_Triggerlevel[3]=60;                            //TCH3 TK_Trigger level
  TK_Triggerlevel[4]=60;                            //TCH4 TK_Trigger level
  TK_Triggerlevel[5]=60;                            //TCH5 TK_Trigger level
  TK_Triggerlevel[6]=60;                            //TCH6 TK_Trigger level
  TK_Triggerlevel[7]=60;                            //TCH7 TK_Trigger level
  TK_Triggerlevel[8]=60;                            //TCH8 TK_Trigger level
  TK_Triggerlevel[9]=60;                            //TCH9 TK_Trigger level
  TK_Triggerlevel[10]=60;                           //TCH10 TK_Trigger level
```

```
    TK_Triggerlevel[11]=60;                    //TCH11 TK_Trigger level
    TK_Triggerlevel[12]=60;                    //TCH12 TK_Trigger level
    TK_Triggerlevel[13]=60;                    //TCH13 TK_Trigger level
    TK_Triggerlevel[14]=60;                    //TCH14 TK_Trigger level
    TK_Triggerlevel[15]=60;                    //TCH15 TK_Trigger level
    TK_Triggerlevel[16]=60;                    //TCH16 TK_Trigger level
    Press_debounce_data=5;                     //Press debounce 1~10
    Release_debounce_data=5;                   //Release debounce 1~10
    Key_mode=1;                                //Key mode 0=single key 1=multi key
    MultiTimes_Filter=0;                       //MultiTimes Filter,>4 ENABLE <4 DISABLE
    Valid_Key_Num=4;                           //Valid Key number when touched
    Base_Speed=10;                             //baseline update speed
    TK_longpress_time=16;                      //longpress rebuild time = _TK_longpress_time1*1s   0=disable
    TK_BaseCnt=59999;                          //10ms   TK_BaseCnt=10ms*48M/8-1,this register need to modify when mcu's
Freq changed
  /*********************************************************
  //TK low power function define
  *********************************************************/
    TK_Lowpower_mode=DISABLE;                  //touch key can goto sleep when TK lowpower mode enable
    TK_Lowpower_level=2;                       //0=20ms 1=50ms 2=100ms 3=150ms 4=200ms,Scan interval when sleep
    TK_Wakeup_level=50;                        //touch key Trigger level in sleep
  /*********************************************************
  //TK special parameter define
  *********************************************************/
    //tk power sel:TK_PSEL_FVR/TK_PSEL_AVDD when select TK_PSEL_FVR PA0.2(TCH3) need a 104 cap
    TK_PSEL_MODE=TK_PSEL_AVDD;
    TK_FVR_LEVEL=TK_FVR_2048V;                 //FVR level:TK_FVR_2048V/TK_FVR_4096V
    TK_EC_LEVEL=TK_EC_3V;                      //C0 voltage sel:TK_EC_1V/TK_EC_2V/TK_EC_3V/TK_EC_3_6V
    TK_icon[0]=4;                              //TCH0 TK Scan icon range 0~7
    TK_icon[1]=4;                              //TCH1 TK Scan icon range 0~7
    TK_icon[2]=4;                              //TCH2 TK Scan icon range 0~7
    TK_icon[3]=4;                              //TCH3 TK Scan icon range 0~7
    TK_icon[4]=4;                              //TCH4 TK Scan icon range 0~7
    TK_icon[5]=4;                              //TCH5 TK Scan icon range 0~7
    TK_icon[6]=4;                              //TCH6 TK Scan icon range 0~7
    TK_icon[7]=4;                              //TCH7 TK Scan icon range 0~7
    TK_icon[8]=4;                              //TCH8 TK Scan icon range 0~7
    TK_icon[9]=4;                              //TCH9 TK Scan icon range 0~7
    TK_icon[10]=4;                             //TCH10 TK Scan icon range 0~7
    TK_icon[11]=4;                             //TCH11 TK Scan icon range 0~7
    TK_icon[12]=4;                             //TCH12 TK Scan icon range 0~7
    TK_icon[13]=4;                             //TCH13 TK Scan icon range 0~7
    TK_icon[14]=4;                             //TCH14 TK Scan icon range 0~7
    TK_icon[15]=4;                             //TCH15 TK Scan icon range 0~7
```

```
    TK_icon[16]=4;                                  //TCH16 TK Scan icon range 0~7
}
```

- **参数说明：**
  - **Tkey IO 使能**

    bit=1 表示使能对应的 TCHx 做 touch key 功能，低位至高位的顺序对应 TCH0~TCH16

    TK_IO_ENABLE=0B00011111111000000;

  - **Tkey 通道扫描周期配置。**

    值越大灵敏度越高，但不能超过理论值否则按键无法扫描通过，常用值不大于 150*

    TK_senprd [i]=50； //i=0~16

  - **Tkey 通道触发门槛值配置。**

    值越大门槛值越高，取值范围为按键差值的 50%~60%，未使用的通道设置成 0xFF

    TK_Triggerlevel[i]=60; //i=0~16

  - **Tkey 触发去抖配置**

    Press_debounce_data=5;//按下去抖 1~10，默认配置为 5

  - **Tkey 释放去抖配置**

    Release_debounce_data=5; //释放去抖 1~10，默认配置为 5

  - **Tkey 按键模式**

    0 表示单键模式，1 表示多键模式

    Key_mode=1; //0=single key 1=multi key

  - **OFFSET 滤波倍数**

    大于等于 4 时，表示开启相应的倍数滤波；小于 4 时表示倍数滤波关闭；默认配置关闭

    MultiTimes_Filter=0; //倍数滤波>=4

  - **最多有效按键个数**

    此配置表示允许同时按下按键时最多有效个数。默认为 4

    Valid_Key_Num=4; //最大按下有效个数

  - **Baseline 更新速度**

    数值越小，baseline 更新速度越快；数值越大，baseline 更新速度越慢；默认为 10 约 100ms

    Base_Speed=10;

  - **按键长按强制更新时间设置**

    长按键强制更新配置。时间= *TK_longpress_time*1s*；默认 16 秒

TK_longpress_time=16;

- **按键扫描基准时间配置**

若系统时钟修改时需要修改此参数，保证基准时间为 10ms；计算公式 TK_BaseCnt=10ms*PCLK/8-1，默认 59999 数值基于 48MHz

TK_BaseCnt=59999;

：某些特殊应用时，此数值可能会大于此值

- **低功耗配置参数：**

- **低功耗模式使能**

当低功耗模式配置为 ENABLE 时，系统在进入睡眠后，Touch 模块才能实现唤醒

TK_Lowpower_mode=ENABLE;

- **低功耗模式扫描间隔**

此处为配置在低功耗模式下，Touch 的扫描间隔，间隔越小，则扫描的频次越大，唤醒的速度也越快；

0=20ms/1=50ms /2=100ms /3=150ms /4=200ms,

TK_Lowpower_level=2;

- **低功耗模式唤醒阈值**

在低功耗模式下，所有的 Touch 通道拥有相同的唤醒阈值；此值可设置为 TK_Triggerlevel[17]数组中最小值的 80%左右

TK_Wakeup_level=50;

注意：Touch 模块在低功耗下的扫描启动信号由 LPT 提供，所以开启低功耗模式后，将占用 LPT 模块，用户不能再配置 LPT 模块。库文件中以 LP 为后缀的*.a 文件才包含低功耗唤醒功能，如 lib_102TKey_1_05LP.a

- **特殊配置参数:**

- **触摸参考电压源配置**

TK_PSEL_AVDD 表示选择 VDD；

TK_PSEL_FVR 表示选择 FVR(PIN23 需要接 104 电容)

默认选择 VDD 做参考；选择 FVR 时拥有更好的抗干扰能力

TK_PSEL_MODE=TK_PSEL_AVDD;

- **FVR 参考电压**

选择 FVR 做参考时，需根据实际 VDD 选择 4.096V 或 2048V 两个参考电压点，默认选择为 4.096V。

当选择 TK_PSEL_AVDD 时，此位无效。选择的 FVR 参考电压必须比 VDD 电压小 500mV 以上

TK_FVR_LEVEL=TK_FVR_4096V;

- **C0 充放电电压选择**

C0 充放电电压选择，默认配置 3.6V，可配置 1V/2v/3/v/3.6v，当 FVR 为 2.048V 时强制选择 1V。

选择的 C0 充放电电压必须比 VDD 电压小 500mV 以上

TK_EC_LEVEL=TK_EC_3_6V;

- **充电电流配置参数。**

若上电 PAD 寄生电容太大，采样值超出上限，那么触摸扫描出现异常，此时应调高对应通道 ICON 值。

TK_icon [i]=4; *//i=0~16*

3.3 TOUCH 低功耗配置

1. 设置 TK_Lowpower_mode 为 ENABLE。使用 Touch Key 睡眠唤醒功能必须使能此参数
2. 配置 TK_Lowpower_level 和 TK_Wakeup_level.

TK_Lowpower_level 数值越大，睡眠状态下触摸扫描间隔越大，唤醒速度也越慢；

TK_Wakeup_level 数值越小，唤醒门槛越低，越容易唤醒

3. 在睡眠前和唤醒后需对 Touch 进行配置

```
TK_setup_sleep();                  //touch key 准备进入深度睡眠
PCLK_goto_deepsleep_mode();  //进入深度睡眠
TK_quit_sleep();                   //touch key 退出睡眠
```

3.4 TOUCH 触摸数据读取

- **Tkey 按键值**

有按键触发时，在 single key 模式 Key_Map 寄存器对应的 bit 位置 1。当使能 Multi key 模式时同时按下按键对应的 bit 位会同时置 1.

序号	Key_Map	对应按键
1	00000001b	TCH0 触发
2	00000010b	TCH1 触发
3	00000100b	TCH2 触发

4	00001000b	TCH3 触发
5	00010000b	TCH4 触发
6	00100000b	TCH5 触发
7	01000000b	TCH6 触发
8	10000000b	TCH7 触发
9	100000000b	TCH8 触发
10	1000000000b	TCH9 触发
11	10000000000b	TCH10 触发
12	100000000000b	TCH11 触发
13	1000000000000b	TCH12 触发
14	10000000000000b	TCH13 触发
15	100000000000000b	TCH14 触发
16	1000000000000000b	TCH15 触发
17	10000000000000000b	TCH16 触发

- **Tkey 按键差值**

offset_data0_abs[17];数组对应 TCH0~TCH16 的按键差值

- **Tkey 按键实时采样值**

sampling_data0 [17];数组对应 TCH0~TCH16 的按键实时采样值

- **Tkey 按键基准值**

baseline_data0 [17];数组对应 TCH0~TCH16 的按键基准值

3.5 TOUCH 触摸灵敏度

默认 C0 电容是 103，有些特殊情况下，需要调整灵敏度的情况，可以使用发送 Touchkey 按键差值和 Touchkey 按键采样基准值的调试函数，透过 UART TX 发送实时的触摸数据，再利

用串口转 USB 工具在 PC 上打印出实时的波形数据。详情参考《AN1511 Touch Key 使用串口工具波形分析使用指南》。

3.6 TOUCH 触摸具体使用

选择内部主频 48MHz 作为系统时钟，TOUCH 0 \TOUCH15 \TOUCH16 有触摸输出对应 GPIO 为低，无触摸 2S 进入休眠。使用 COUNTA 定时 1ms。

```
/*****************************************************************/
//GPIO Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void GPIO_CONFIG(void)
{
    GPIO_Write_High(GPIOA0,10);
    GPIO_Write_High(GPIOA0,11);
    GPIO_Write_High(GPIOA0,12);
    GPIO_Init(GPIOA0,10,0);
    GPIO_Init(GPIOA0,11,0);
    GPIO_Init(GPIOA0,12,0);
}
/*****************************************************************/
//COUNTA Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void COUNTA_CONFIG(void)
{
    COUNT_DeInit();                                                          //clear all countA Register
    COUNTA_Init(48000,0,Period_H,DIV1,REPEAT_MODE,CARRIER_ON,OSP_LOW);       //Data_H=Time/(1/sysclock)
    COUNTA_Config(SW_STROBE,PENDREM_OFF,MATCHREM_OFF,REMSTAT_0,ENVELOPE_0);  //countA mode set
    COUNTA_Start();                                                          //countA start
    COUNTA_Int_Enable();                                                     //countA INT enable
}
void APT32F102_init(void)
{
//-------------------------------------------------------------/
//Peripheral clock enable and disable
//EntryParameter:NONE
//ReturnValue:NONE
//-------------------------------------------------------------/
    SYSCON->PCER0=0xFFFFFFF;                                                 //PCLK Enable   0x00410071
```

```
    SYSCON->PCER1=0xFFFFFFF;                                    //PCLK Enable
    while(!(SYSCON->PCSR0&0x1));                                //Wait PCLK enabled
//-----------------------------------------------------------------/
//ISOSC/IMOSC/EMOSC/SYSCLK/IWDT/LVD/EM_CMFAIL/EM_CMRCV/CMD_ERR OSC stable interrupt
//EntryParameter:NONE
//ReturnValue:NONE
//-----------------------------------------------------------------/
    SYSCON_CONFIG();                                            //syscon   initial
   CK_CPU_EnAllNormalIrq();                                     //enable all IRQ
//-----------------------------------------------------------------/
//Other IP config
//-----------------------------------------------------------------/
   GPIO_CONFIG();                                               //GPIO initial
   COUNTA_CONFIG();                                             //CountA initial
   TK_CONFIG();
}

volatile u16 Sleeptimercounter;
volatile _Bool BitTIME10ms;
volatile u8 Tim1msadd10ms;
/*****************************************************************/
//CONTA Interrupt
//EntryParameter:NONE
//ReturnValue:NONE
/*****************************************************************/
void CNTAIntHandler(void)
{
    // ISR content ...
        Tim1msadd10ms++;
        //
        if(Tim1msadd10ms>9)
        {
            Tim1msadd10ms=0;
            //
            BitTIME10ms = 1;                                    //
        }
}
/*****************************************************/
//key data get
/*****************************************************/
void Keymap_Porg(void)
{
   if(Key_Map!=0)
   {
```

```
            if(Key_Map_bk!=Key_Map)
            {
                Key_Map_bk=Key_Map;
                if(Key_Map==0x01)
                {
                  GPIO_Write_Low(GPIOA0,10);
                }
                if(Key_Map==0x8000)
                {
                  GPIO_Write_Low(GPIOA0,11);
                }
                if(Key_Map==0x10000)
                {
                  GPIO_Write_Low(GPIOA0,12);
                }
            }
    }
    else
    {
             Key_Map_bk=0;
            GPIO_Write_High(GPIOA0,10);
            GPIO_Write_High(GPIOA0,11);
            GPIO_Write_High(GPIOA0,12);
    }
}
void SLEEP_CTRL(void)
{
    if(Key_Map_bk == 0)                                          //无触摸
    {
            if(++Sleeptimercounter > 200)                          // 2S
            {
                Sleeptimercounter--; //
                SYSCON_WDT_CMD(DISABLE);                           //关闭
                SYSCON_IWDCNT_Reload();
                TK_setup_sleep();                                  // touch key 准备进入深度睡眠
                PCLK_goto_deepsleep_mode();                        //进入深度睡眠
                TK_quit_sleep();                                   // touch key 退出睡眠
                SYSCON_WDT_CMD(ENABLE);                            //enable WDT
                SYSCON_IWDCNT_Reload();
                Sleeptimercounter = 0;
                return;
            }
    }
    else
```

```
    {
        Sleeptimercounter = 0;
    }
}
/*****************************************************/
//main
/*****************************************************/
int main(void)
{
    APT32F102_init();
    //
    while(1)
    {
        SYSCON_IWDCNT_Reload();
        Keymap_Porg();
        if(BitTIME10ms)
        {
            BitTIME10ms = 0;
            //
            SLEEP_CTRL();                                   //休眠
        }
    }
}
```

- **触摸测试：**

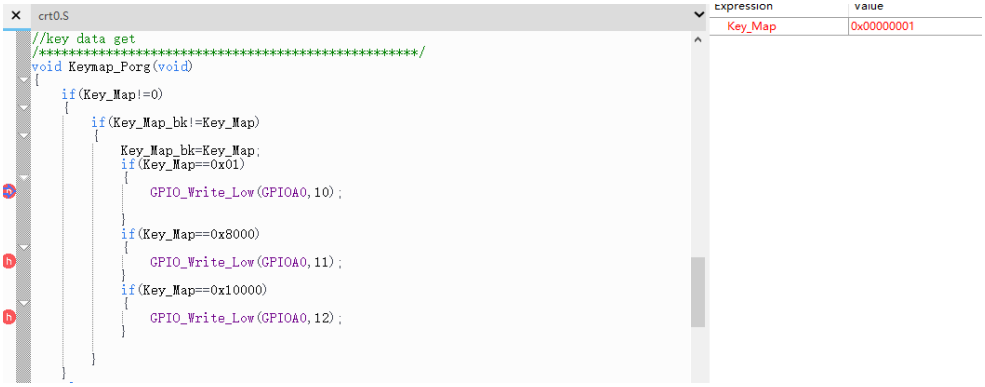

图 3.6.1 触摸 KEY_MAP 值

4. 程序下载和运行

1．将目标板与仿真器连接，分别为 VDD SCLK SWIO GND

2. 程序编译后仿真运行

3. 触摸按键观察 KEY_MAP 的值，确认触摸按键是否有效