# TOUCH

## 概述

为了方便客户使用，加快客户上手速度，ATP CSI接口TOUCH的设计中，提供简单方便的配置及其操作。配置方面包括TOUCH常用参数的配置接口。操作方面只需调用简单的函数，然后读取相应的变量即可实现TOUCH功能。

TOUCH的实现有两种方式：①定时中断里调用TOUCH处理函数；②在主循环里调用TOUCH处理函数

|  |  |  |
| --- | --- | --- |
| **方式 特点** | **定时器中断里调用TOUCH处理函数** | **主循环里调用TOUCH处理函数** |
| 优点 | 每一轮的按键扫描时间可控，触摸体验良好 | 不会占用中断资源，对别的中断不会有影响 |
| 缺点 | 会占用中断资源，如果有高时序要求的中断，没有配置中断好中断优先级的话会影响高时序要求的中断 | 每一轮的按键扫描时间不可控，如果主循环一次循环里有函数占用大量时间，会影响按键的触摸体验 |

注意：

① 在定时器里调用TOUCH处理函数，如果有其他对时序要求高的中断处理函数，请调整中断优先级，以确保高时序要求的中断处理函数不被影响。

②如果有使用到TOUCH低功耗唤醒，则需要注意LPT定时器的使用，因为在有TOUCH的低功耗模式下会占用LPT定时器

## API列表

Table 25‑1 TOUCH CSI接口函数

|  |  |  |
| --- | --- | --- |
| API | 说明 | 函数位置 |
| csi\_tkey\_timer\_handler | 定时器中断里TOUCH的处理函数 | TOUCH库 |
| csi\_tkey\_prgm | 主函数里TOUCH的处理函数 |
| csi\_tkey\_setup\_sleep | TOUCH进低功耗前时需要调用的函数 |
| csi\_tkey\_quit\_sleep | TOUCH从低功耗模式唤醒后需要调用的函数 |
| csi\_tkey\_init | TOUCH初始化函数 |
| csi\_tkey\_handler | TOUCH中断处理函数 |
| csi\_tkey\_parameter\_init | TOUCH参数配置函数 | Tkey\_parameter.c |

## API详细说明

### csi\_tkey\_timer\_handler

void csi\_tkey\_timer\_handler(void)

#### 功能描述

定时器中断里需要调用的TOUCH处理函数（方式①使用的函数）。

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值。

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_handler

void csi\_tkey\_handler(void)

#### 功能描述

TOUCH中断处理函数。

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值。

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_setup\_sleep

void csi\_tkey\_setup\_sleep(void)

#### 功能描述

TOUCH进低功耗前时需要调用的函数。

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值。

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_quit\_sleep

void csi\_tkey\_quit\_sleep(void)

#### 功能描述

TOUCH从低功耗模式唤醒后需要调用的函数

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值。

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_init

void csi\_tkey\_init(void)

#### 功能描述

TOUCH初始化函数

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_prgm

void csi\_tkey\_prgm(void)

#### 功能描述

主函数里TOUCH的处理函数（方式②使用的函数）

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值

1. 结构体/枚举说明表

无结构体/枚举

### csi\_tkey\_parameter\_init

void csi\_tkey\_parameter\_init(void)

#### 功能描述

TOUCH参数配置函数

#### 参数/返回值说明

1. 参数

无参数。

1. 返回值

无返回值。

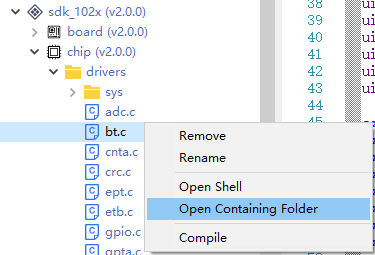
1. 结构体/枚举说明表

无结构体/枚举

## 添加TOUCH库文件

如果工程里没有库文件需要重新添加库文件或者升级库文件的方法如下：

1. 打开工程，依次展开SDK包 -> chip -> drivers,鼠标右键单击一个.c文件，然后点击Open Containing Folder选项，打开该.c文件所在的文件夹。如图：



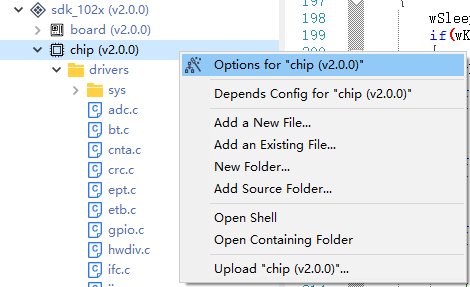
**Figure**  **25‑1 添加Touch库文件1**

1. 把对应的.a文件拷贝到该文件夹，如果是升级库文件，请删除原来的.a文件



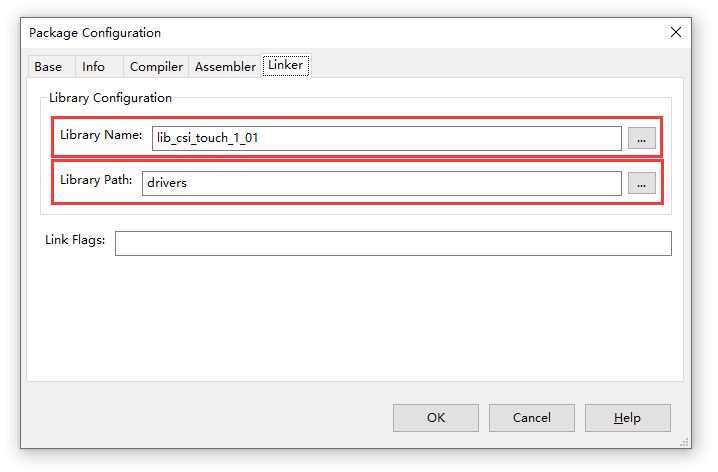
**Figure**  **25‑2 添加Touch库文件2**

1. 鼠标右键点击chip组件，选择Options for “chip(vx.x.x)” ，如图



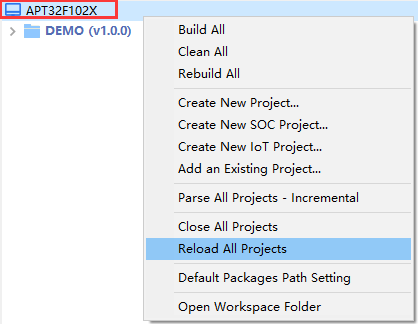
**Figure**  **25‑3 添加Touch库文件3**

1. 然后在linker选项卡里的library name选项里输入拷贝过去的.a文件的名称，library path选项里输入drivers，如图所示：



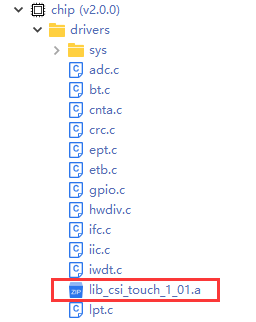
**Figure**  **25‑4 添加Touch库文件4**

1. 鼠标右键点击工程节点，选择reload all projects 选项如图。



**Figure**  **25‑5 添加Touch库文件5**

1. 然后即可在chip组件下的drivers文件下就包含了所添加的.a文件，至此库文件添加完毕



**Figure**  **25‑6 添加Touch库文件5**

## 使用定时器中断的TOUCH使用示例

以CORET定时器为例，配置好定时器，并打开定时器中断（定时器的配置请参考相关章节的介绍），在中断函数里调用csi\_tkey\_timer\_handler()如下所示：

在TKEY中断函数里调用csi\_tkey\_handler()，如下所示：

void CORETHandler**(**void**)** **{**

// ISR content ...

//CK801->CORET\_CVR = 0x0; // Clear counter and flag

tick\_irq\_handler**(NULL);** //tick自己的处理函数

csi\_tkey\_timer\_handler**();** //TOUCH的处理函数

**}**

最后主循环里查询wKeyMap变量即可，如下所示：

void TKEYIntHandler**(**void**)** **{**

// ISR content ...

csi\_tkey\_handler**();**

**}**

void touch\_timer\_demo **(**void**){**

csi\_tkey\_init**();**//TOUCH初始化函数调用

**while(**1**){**

//wKeyMap 的bit位代表TOUCH的通道是否有按键按下，如：bit0对应TOUCH通道0 bit16对应TOUCH通道16

**if(**wKeyMap**!=**0**) {** //按键键处理

**if((**wKeyMap **&** 0x20**)==**0x20**)** **{** //判断TOUCH通道5是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x40**)==**0x40**) {** //判断TOUCH通道6是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x80**)==**0x80**) {** //判断TOUCH通道7是否按下

nop**;**

**}**

//......

**}**

wKeyMap**=**0**;**

mdelay**(**10**);**

**}**

**}**

注意：在定时器里的TOUCH长按键有效时间 = 定时器时间\*100\* wTkeyLongPressTime（单位与时器时间单位一致）

## 不使用定时器中断的TOUCH使用示例

在TKEY中断函数里调用csi\_tkey\_handler()，如下所示：

在主循环里调用csi\_tkey\_prgm()函数，然后查询wKeyMap变量即可，如下所示：

void TKEYIntHandler**(**void**)** **{**

// ISR content ...

csi\_tkey\_handler**();**

**}**

void touch\_main\_demo **(**void**){**

csi\_tkey\_init**();**//TOUCH初始化函数调用

**while(**1**){**

csi\_tkey\_prgm**();** //TOUCH处理函数

//wKeyMap 的bit位代表TOUCH的通道是否有按键按下，如：bit0对应TOUCH通道0 bit16对应TOUCH通道16

**if(**wKeyMap**!=**0**)** //按键键处理

**{**

**if((**wKeyMap **&** 0x20**)==**0x20**) {** //判断TOUCH通道5是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x40**)==**0x40**) {** //判断TOUCH通道6是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x80**)==**0x80**) {** //判断TOUCH通道7是否按下

nop**;**

**}**

//......

**}**

wKeyMap**=**0**;**

mdelay**(**10**);**

**}**

**}**

注意：在主循环里的TOUCH长按键有效时间 = 主循环调用csi\_tkey\_prgm()的时间间隔\*100\* wTkeyLongPressTime（单位与主循环调用csi\_tkey\_prgm()的时间间隔单位一致）

## TOUCH低功耗唤醒使用示例

在TKEY中断函数里调用csi\_tkey\_handler()，如下所示：

然后参考以下示例程序即可：

void TKEYIntHandler**(**void**)** **{**

// ISR content ...

csi\_tkey\_handler**();**

**}**

void touch\_lowpower\_demo **(**void**){**

uint32\_t wSleepCnt **=** 0**;**

csi\_tkey\_init**();**//TOUCH初始化函数调用

**while(**1**){**

csi\_tkey\_prgm**();** //TOUCH处理函数

//wKeyMap 的bit位代表TOUCH的通道是否有按键按下，如：bit0对应TOUCH通道0 bit16对应TOUCH通道16

**if(**wKeyMap**!=**0**)** //按键键处理

**{**

wSleepCnt **=** 0**;** //清除低功耗进低功耗时间计数

**if((**wKeyMap **&** 0x20**)==**0x20**) {** //判断TOUCH通道5是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x40**)==**0x40**) {** //判断TOUCH通道6是否按下

nop**;**

**}**

**if((**wKeyMap **&** 0x80**)==**0x80**)** **{** //判断TOUCH通道7是否按下

nop**;**

**}**

//......

**}**

wSleepCnt**++;**

**if(**wSleepCnt **>** 500**){**

csi\_tkey\_setup\_sleep**();** //设置TOUCH唤醒，及相关的配置

soc\_pm\_enter\_sleep**(**PM\_MODE\_DEEPSLEEP**);** //进入低功耗模式（deepsleep）

csi\_tkey\_quit\_sleep**();** //唤醒后清除一些操作

wSleepCnt**=**0**;**

**}**

wKeyMap**=**0**;**

mdelay**(**10**);**

**}**

**}**

注意：TOUCH的低功耗唤醒需要占用LPT定时器，如果有使用LPT的请在调用csi\_tkey\_setup\_sleep()，停止使用LPT，并在唤醒后重新初始化LPT。

## 参数说明

void csi\_tkey\_parameter\_init**(**void**)**

**{**

wTkeyIoEnable **=** 0xf**<<**10**;** //TOUCH通道使能设置，Bit0->通道0，bit16->通道16, 1使能 0关闭通道

hwTkeySenprd**[**x**]** **=** 60**;** //扫描目标值，值越大灵敏度越高，通道扫描时间增大

hwTkeyTriggerLevel**[**x**]** **=** 60**;** //触发阈值设置值，阈值越大灵敏度越低

byPressDebounce **=** 5**;** //按键按下滤波系数设置 1~10

byReleaseDebounce **=** 5**;** //按键释放滤波系数 1~10

byKeyMode **=** 1**;** //按键扫描模式 0 = 单按键模式 1 = 多按键模式

byMultiTimesFilter **=** 0**;** //阈值超上限滤波设置, >=4 ENABLE <4 DISABLE

byValidKeyNum **=** 4**;** //最大有效按键数设置，如果同时按下的按键数超过该值则认为按键无效

byBaseSpeed **=** 10**;** //基线更新速度设置

wTkeyLongPressTime **=** 16**;** //长按键有效时间设置 = wTkeyLongPressTime\*1s 0=disable ，长按键有效时间与调用按键处理函数的速度有关

byTkeyLowpowerMode **=** ENABLE**;** //按键低功耗唤醒模式设置 ENABLE使能低功耗唤醒，DISABLE关闭

byTkeypowerLevel **=** 2**;** //0=20ms 1=50ms 2=100ms 3=150ms 4=200ms,低功耗模式按键扫描触发时间设置

byTkeyWakeupLevel **=** 60**;** //低功耗模式下的按键触发阈值设置值

hwTkeyPselMode **=** TK\_PSEL\_AVDD**;** //TOUCH参考电源选择:TK\_PSEL\_FVR/TK\_PSEL\_AVDD 当选择TK\_PSEL\_FVR时PA0.2(TCH3)需要外接104电容，且TCH3通道不能使用

hwTkeyFVRLevel **=** TK\_FVR\_4096V**;** //FVR电压等级设置:TK\_FVR\_2048V/TK\_FVR\_4096V

hwTkeyECLevel **=** TK\_EC\_1V**;** //C0电压设置:TK\_EC\_1V/TK\_EC\_2V/TK\_EC\_3V/TK\_EC\_3\_6V

hwTkeyIcon**[**x**]** **=** 4**;** // C0补偿电流设置，值越大补偿电流越大

byTkeyScanTime**[**x**]** **=** 0**;** //扫描时间设置0不设置扫描时间 1->1ms 2->1.5ms 3->2ms 4->3ms 5->5ms 6->10ms 7->100ms

byTkeyTckdiv **=** 0**;** //低功耗模式下的TKEY时钟分频设置，FIMOSC/(byTkeyTckdiv + 1) = 1MHz

byTkeyPckdiv **=** 0**;** //运行模式下的TKEY时钟分频设置，FSYSCLK/(byTkeyPckdiv + 1) = 1MHz

**}**