

1 总体描述	3
1.1 赛元触控 MCU 描述	3
1.2 EasyCodeCube(赛元代码生成器) TK 触控描述	3
2 EasyCode TK 调试流程	4
2.1 安装 EasyCodeCube, 打开调试工具及连接硬件	4
2.2 调试触控参数	5
① 配置基本设置	5
② 点击“配置”按钮	5
③ 参数配置, 进入触控调试	6
④ 触摸按钮参数自适应	7
⑤ 进行单通道调试	8
⑥ 进行按钮诊断	12
⑦ 生成配置文件	13
⑧ 滑轮和滑条的参数配置	13
⑨ 低功耗参数设置	14
⑩ 生成工程	14
2.3 按钮滑轮滑条测试功能调试	16
2.4 用户程序和赛元触控软件库的融合思路	18
2.4.1 主程序和库文件的整体结构关系	18
2.4.2 库文件的调用流程	18
2.4.3 主程序和库文件的时序关系	20
2.4.4 软件融合的注意事项	20
2.4.5 整体 Code 的测试	20
2.5 注意事项	21
3 代码例程	22
以下黄色区域是客户可修改函数实现	22
3.1 按钮使用	22
3.1.1 图形化编程	22
3.1.2 C 语言代码	22
3.2 按钮, 滑轮和滑条三者合用	23
3.2.1 图形化编程	23
3.2.2 C 语言代码	23
3.3 低功耗按钮使用	24
3.3.1 图形化编程	24
3.3.2 C 语言代码	24
3.3.3 回调函数的编辑	25

1 总体描述

本文档是赛元 Touchkey MCU 系列触控库在 EasyCode 上的配置手册。用户可以通过 EasyCode 开始开发赛元触控 MCU，快速配置 TK 资源。

1.1 赛元触控 MCU 描述

赛元触控 MCU 的触控架构具有高灵敏度触控模式，其特点如下：

1. 高灵敏度模式可适应隔空按键触控、接近感应等对灵敏度要求较高的触控应用；
2. 最多可实现 31 路触控按键及衍生功能；
3. 高灵活度开发软件库支持，低开发难度；
4. 自动化调试软件支持，智能化开发；
5. 部分型号可以在 MCU STOP 模式下进入低功耗模式工作，单个触控按键唤醒时芯片整体功耗可低至 8uA@3.3V。

用户通过使用赛元提供的 EasyCodeCube，快速简单实现所需的触控功能。用户可以通过下表的信息了解 EasyCodeCube 的触控模式：

说明	高灵敏度模式
特点	① 高抗干扰能力，可通过 10V 动态 CS ② 超高灵敏度
适用的应用	① 普通触控按键应用 ② 隔空触控按键应用 ③ 接近感应应用 ④ 对灵敏度要求较高的触控应用
如何进入模式	通过项目工程载入高灵敏度的触控库来选择高灵敏度模式
对应的库文件	“TK.LIB”

EasyCodeCube 目前高灵敏度触控模式应用具体分为“弹簧触控”和“隔空触控”，下表是两个模式的对比：

说明	弹簧触控	隔空触控
应用场景	触控按键需要与接触面（用户产品外壳）直接接触	触控按键可以与接触面（用户产品外壳）保持一定距离，常规情况下支持 3mm
按键适用范围	常规任意按键	三个以上（包含三个）

用户可以通过需要自行选择应用类型。

1.2 EasyCodeCube(赛元代码生成器) TK 触控描述

EasyCodeCube 为赛元微电子有限公司开发的代码工具，其中整合 TK 调试工具和触控库，结合流程图编程功能，可以协助用户简单、快捷以及高效的使用赛元 TK 的功能，降低入门门槛，减少配置时间，提高产品研发效率。用户可以通过 EasyCodeCube 完成 TK 配置的使用步骤，包括调试以及用户程序的开发。

在赛元的官方网站上可以下载工具的安装程序，网址为：

<http://www.socmcu.com/uploadfile/download/TOOLS/EasyCodeCube.rar>，工具还在持续更新，以支持更多的 MCU 和更丰富的功能，因此定期更新工具可以获得更好的体验

2 EasyCode TK 调试流程

2.1 安装 EasyCodeCube，打开调试工具及连接硬件

① Setup EasyCodeCube;

从赛元官网中获取安装包，根据提示安装赛元[EasyCodeCube.exe](#)。

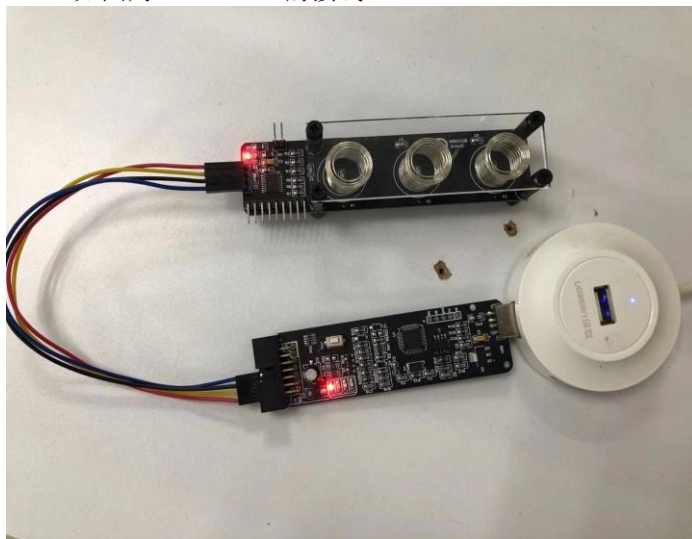
② 升级 SC-LINK / SC-LINK PRO 固件,更新MCU 库;

在线烧写器SC-LINK / SC-LINK PRO 的固件和SOC Pro51 的MCU 库文件需升级到赛元官网最新版本;

③ 硬件连接: 电脑 USB--> SC-LINK / SC-LINK PRO (VCC/GND/CLK/DIO)-->用户

PCB(VCC/GND/tCK/tDIO); 并测试连接正常。调试过程需要用到硬件 UART 资源, 请 PCB 预留接线。

如图为 SC-LINK 的接线:

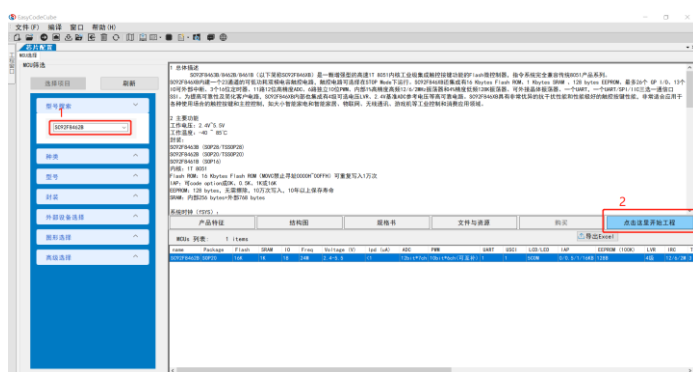


④ 新建 EasyCodeCube 工程, 详情请参考《赛元代码生成器用户手册》;

注: 当使用到小 Rom 资源(SC92F827X)的芯片时, 如果需要在板调试, 建议先建立同系列的大 ROM 资源芯片的工程, 避免重复原因。具体原因可参考“[调试触控参数](#)“的步骤②的注意事项。

⑤ 选择所用芯片型号后, 点击“开始工程”, 在模板选择界面提供了赛元所支持的 8 种 TK 应用情景,

用户可对应选择应用种类进行开发。



以下以 TKDEMO-弹簧触控为例进行说明:

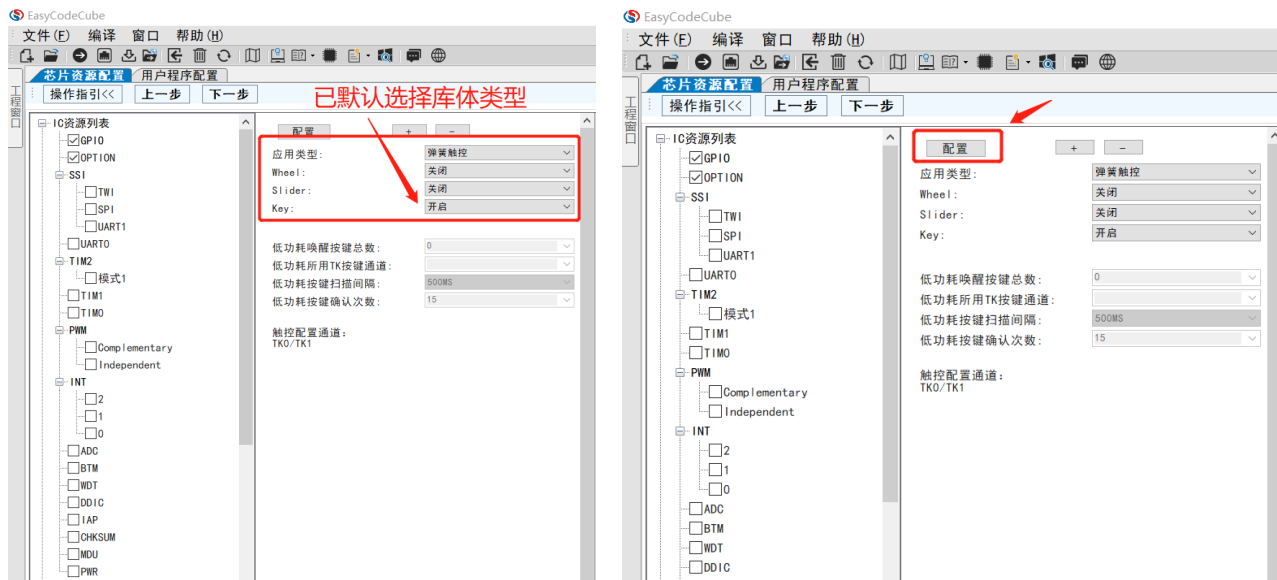
双击进入对应 TK 触控模板

2.2 调试触控参数

① 配置基本设置

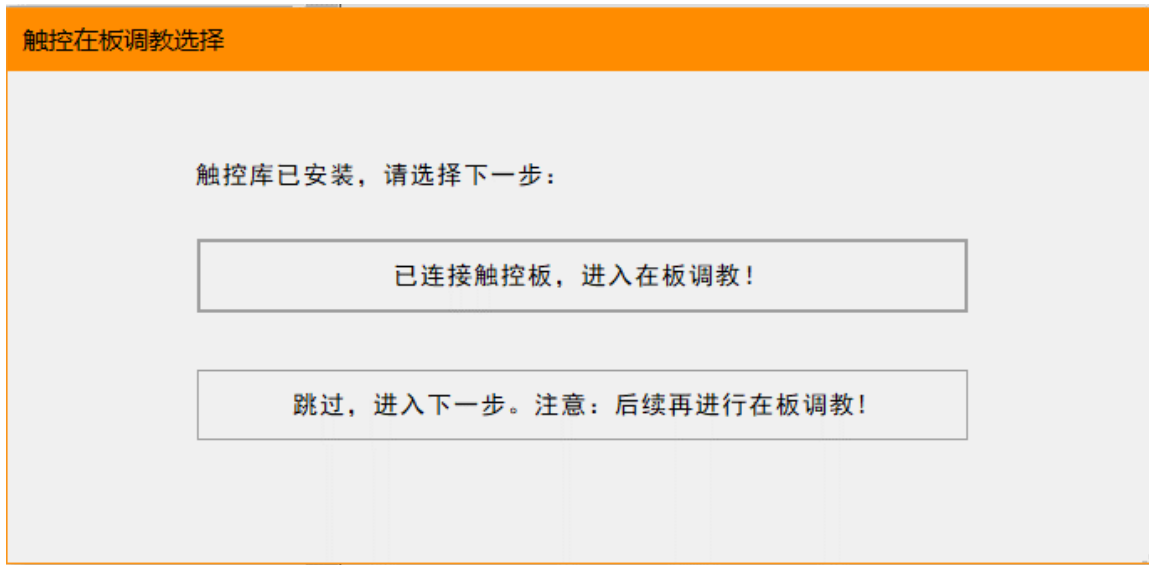
根据选择不同模板，魔盒将自动选择应用类型。例：选择为“弹簧触控”时，将开启按键(KEY)，滑轮(Wheel)、滑条(Slider)将关闭。

切勿手动修改该类型，若需要做其他类型应用，只需切换模板进行开发即可。



② 点击“配置”按钮

点击“配置”按钮，弹出以下界面：



注：若点击“跳过，进入下一步”选项，则不进行板调教，直接跳转到[步骤⑦](#)；

点击“已连接触控板，进入在板调教”，进入 Touch Key Tool Menu 软件界面：



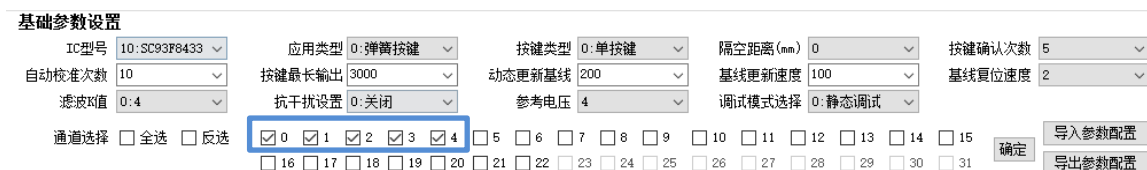
注：

- 1) 当芯片使用到小 ROM 资源时，无法进行在板调试，如果点击在板调试，会提示如果需要获取 PCB 触控，需要执行以下步骤：（只针对 SC92F827X 系列型号）
- 2) 新建与小 ROM 资源同系列的大 ROM 资源芯片的魔盒工程（如 SC92F8271 芯片需要建立 SC92F8371 芯片的工程）。
- 3) 通过大 ROM 资源芯片工程完成“调试触控参数”的步骤①到⑦，从而获得 PCB 板的触控参数配置文件。
- 4) 此时可以在“用户工程驱动列表”中把“TK 触控”导入到备份区。
- 5) 切换回或者重新新建小 ROM 资源芯片工程，在备份区中再导入刚才备份的“TK 触控”，此时可以跳过在板调试流程，触控库会直接调用大 ROM 资源魔盒工程生成的配置文件，而且相关的配置信息也会保留（注意要点击“配置”按钮）。



③ 参数配置，进入触控调试

- a. 该界面只需关注 IC 型号是否正确，应用类型是弹簧还是隔空，其他参数保持不动即可。
- b. 设勾选项项目使用的TK通道，如图所示：



- c. 设置应用的基本信息如下：

应用类型：上位机自动配置

按键类型：选择单按键或者组合按键（双键）。根据实际项目需要选择。

隔空距离: 弹簧按键选择0, 隔空按键按照实际项目设置隔空距离。

按键确认次数: 该参数决定触控算法运行的出键速度, 出键速度与一轮按键扫描时间有关, 若扫描一轮按键需要12MS, 按键确认次数为5次, 则按键需要的响应时间为 $5 \times 12\text{MS} = 60\text{MS}$ 。

自动校准次数: 该参数决定了初始化基线的速度, 次数越多基线越稳定, 同时时间也更长。建议保持默认。

按键最长输出: 该参数决定了按键持续响应的的时间, 单位为轮数。按键时间到达指定次数, 则该按键的标志会被清除。

动态更新基线时间: 该参数用于处理按键浮起的更新速度, 保持默认不改动

基线更新速度: 该参数用于更新基线。保持默认不改动

基线复位速度: 该参数决定基线复位的速度。值越大, 更新速度越慢, 保持默认不动

滤波K值: 保持默认不改动

抗干扰设置: 用于扫描时钟变频, 有助于通过 EMI 测试, 当项目有 EMI 测试要求, 需要选择打开1:12bit 低功耗应用不支持该参数抗干扰设置。

参考电压: 保持默认不改动

调试模式选择: 静态调试为确定触控参数, 动态调试为应用中采集数据, 这里选择静态调试, 后续章节会介绍动态调试。

- d. 选择通道, 配置参数完成后点击“确定”按钮, 此时通道选择上锁, 不能进行设置。若需要更改通道, 需要点击“取消”按钮。

注: 由于调试触摸需要用到烧录口上的 UART 资源, 部分型号烧录口也具有 TK 功能, 因此在进行触摸调试时无法调试这两路的参数。若用户需要用到这两个 TK 口, 请联系赛元的工程师协助。

④ 触摸按键参数自适应

用户点击“确定”按钮后会进入按键参数自适应阶段, 此时需要等待几十秒到几分钟的时间, 具体时间和按键的个数有关, 直到弹出的提示窗口关闭, 自适应完成。在此过程, 需要用户安装好整机, 请勿对面板以及面板周围进行任何操作。



⑤ 进行单通道调试

a. 在通道调试区点击对应通道绿色的按钮，进行单通道调试界面：

⑤单通道调试

☒ 时钟
 ☒ 分辨率
 ☒ 增益
 ☒ 扫描周期
 ☒ 阈值
 ☒ 数据值
 ☒ 电容值PF
 ☒ 信噪比
 ☒ 变化率
 ☒ 变化量
 ☒ 数据修正

	TK14	TK15	TK18
▶ 时钟			
分辨率			
增益			
扫描周期			
阈值			
数据值			
电容值PF			
信噪比			

b. 设置触控相关参数


 单通道调试
 ×

触控参数设置

时钟
 分辨率
 增益
 扫描周期
 阈值

数据值
 电容值PF
 信噪比
 变化率
 变化量
 数据修正

当前调试通道 **TK1**

限定条件

图表显示

启动调试

时钟：保持默认，不进行改动。

分辨率：保持默认，不进行改动

扫描周期：设置范围 1-32，单位为 128us。数值越大，该键扫描时间越长，变化量越大。

阈值设置：设置范围1-8，数值越大，灵敏度越低，反之亦然。如设置值为5，即阈值设置为变化量的50%，当数据变化超过阈值认为有键。建议设置为5。

一般情况下，按键经过自适应过程，用户无需修改以上参数，直接点击启动调试。

c. 点击“启动调试”按钮进行调试：调试分两个过程：无触摸过程以及触摸过程。请按照界面的提示相应进行操作。该过程大约需要 15 秒。

不触摸过程:


 单通道调试
 ×

触控参数设置

时钟	2	数据值		当前调试通道 TK1  请将手移开面板,在下一个提示前不要放置任何物体在面板之上.
分辨率	42	电容值PF		
增益	4	信噪比		
扫描周期	8	变化率		
阈值	5	变化量		
		数据修正	31	

限定条件

图表显示

停止调试

触摸过程:


 单通道调试
 ×

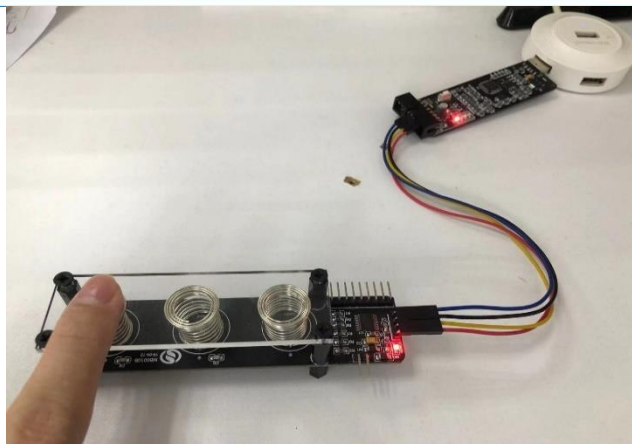
触控参数设置

时钟	2	数据值		当前调试通道 TK1  请将手指或者手持铜柱放置 在对应按键感应面的垂直方 向上.数据采集中.....
分辨率	42	电容值PF		
增益	4	信噪比		
扫描周期	8	变化率		
阈值	5	变化量		
		数据修正	31	

限定条件

图表显示

停止调试



注：软件显示的TK通道与MCU规格书一致，请根据实际PCB的layout布局，操作对应的按键，否则得到的结果将会错误！

调试结束:若调试通过，则下图界面内显示绿色图标

 单通道调试

触控参数设置

时钟	2	数据值	3919	当前调试通道 TK1  当前通道测试完成.
分辨率	42	电容值PF	10	
增益	4	信噪比	135	
扫描周期	8	变化率	241	
阈值	5	变化量	947	
		数据修正	31	

限定条件

当前参数满意度: 8000 CP电容要求: <32PF 信噪比要求: >5 变化率要求: >5
变化量要求: > 35 数据修正值: 0 ≤ N ≤ 128

图表显示

启动调试

若调试不通过，则显示红色图标。

 单通道调试

触控参数设置

时钟	2	数据值	3904	当前调试通道 TK1  当前通道测试完成.
分辨率	42	电容值PF	10	
增益	4	信噪比	0	
扫描周期	8	变化率	0	
阈值	5	变化量	0	
		数据修正	31	

限定条件

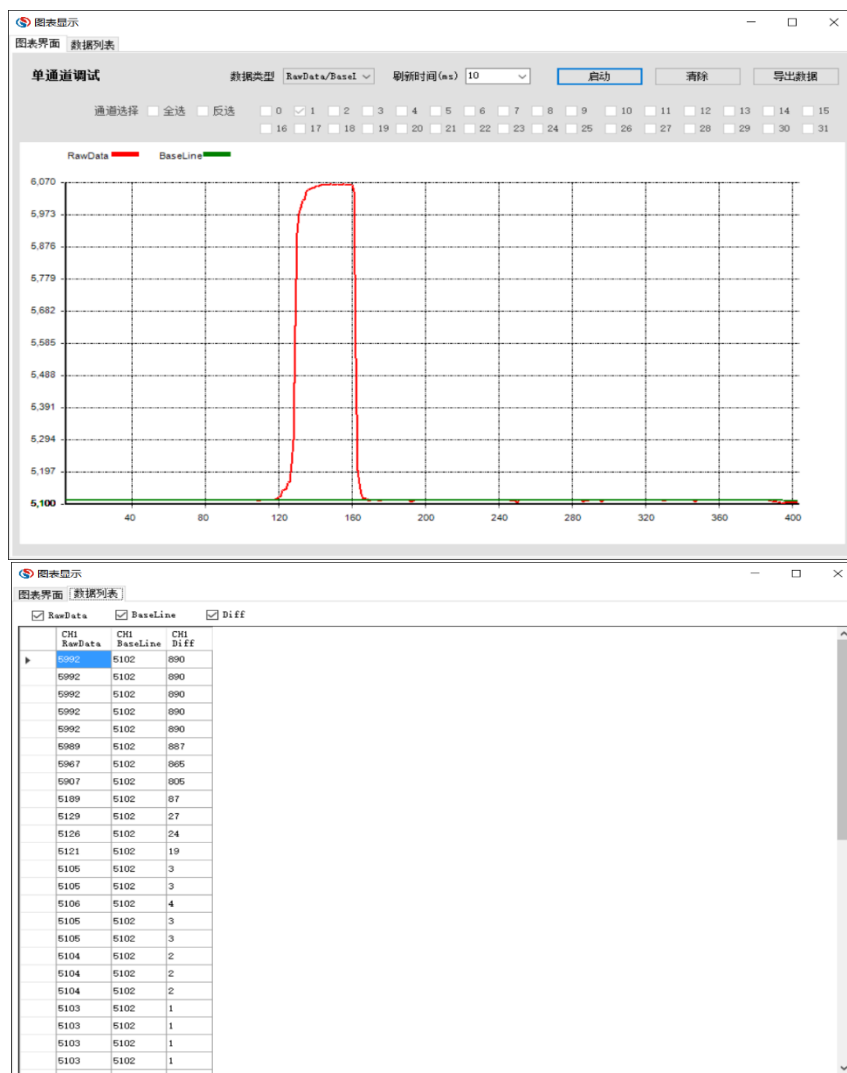
当前参数满意度: 8000 CP电容要求: <32PF 信噪比要求: >5 变化率要求: >5
变化量要求: > 35 数据修正值: 0 ≤ N ≤ 128

图表显示

启动调试

不通过的项目相应会红色字体标出

d. 点击“图表显示”按钮，再按“启动”按钮可以实时的观察数据变化



依次调试每个按键，调至按键均通过

⑥ 进行按键诊断



赛元高灵敏度触控调试软件

升级(U) 语言(L) 关于(A)

①基础参数设置

IC型号: 52:SC95F8522 应用类型: 0:弹簧按键 按键类型: 0:单按键 隔空距离(mm): 0 按键确认次数: 5

自动校准次数: 10 按键最长输出: 3000 动态更新基线: 200 基线更新速度: 100 基线复位速度: 2

滤波系数: 0:4 抗干扰设置: 0:关闭 参考电压: 4 调试模式选择: 0:静态调试

通道配置

通道选择: ☐ 反选 ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐ 13 ☒ 14 ☒ 15 ☐ 16 ☐ 17 ☒ 18 ☐ 19 ☐ 20 ☐ 21 ☐ 22 ☐ 23 ☐ 24 ☐ 25 ☐ 26 ☐ 27 ☐ 28 ☐ 29 ☐ 30 ☐ 31

②单通道调试

☒ 时钟 ☒ 分辨率 ☒ 增益 ☒ 扫描周期 ☒ 阈值 ☒ 数据值 ☒ 电容值PF ☒ 信噪比 ☒ 变化率 ☒ 变化量 ☒ 数据修正

	TK14	TK15	TK18
时钟	3	3	3
分辨率	50	50	50
增益	4	4	4
扫描周期	8	8	8
阈值	5	5	5
数据值	3791	3840	3902
电容值PF	8	8	8
信噪比	52	44	55

③按键诊断

当前通道: TK14 TK15 TK18

诊断结果: 调整方案

按键诊断是分析按键间的相互影响的过程。若按键间的相互影响比较大，会影响到按键的性能。点击“启动诊断”按钮。



④按键诊断

当前通道: TK14

请将手移开面板, 在下一个提示前不要放置任何物体在面板之上。

TK14 TK15 TK18

诊断结果: 调整方案

注：软件显示的 TK 通道与 MCU 规格书一致，请根据实际 PCB 的 layout 布局，操作对应的按键，否则得到的结果将会错。

若诊断不通过，请根据诊断结果和调整方案，调整硬件 Layout。如下图是诊断不通过的



按键诊断

当前通道: TK14

TK1 TK2 TK3 TK13 TK14

当前通道测试完成。

诊断结果: TK1和2, TK2和1, TK13和3, 相互影响大

调整方案: 调整布线, 修改TK1和2, TK2和1, TK13和3, 感应器件的距离

⑦ 生成配置文件

完成按键诊断并且测试通过后，点击“导出配置信息”按钮。EasyCodeCube 生成配置文件 S_TOUCHKEYCFG.H，并且把该文件放置在用户工程的“工程路径\工程名\Keil_C\Drivers\TKDriver\H”路径中。在 EasyCodeCube 的 TK 配置页面中，可以看到目前配置完成的 TK 通道：



注：配置信息导出完成后，如果选择模板应用类型中没有选择到滑轮滑条或者低功耗，可以直接进行用户程序和 TK 触控库的融合，省略以下参数设置操作，直接跳至步骤⑩生成工程即可。

⑧ 滑轮和滑条的参数配置

选择弹簧/隔空滑轮滑条应用，点击模板进入后，需设置以下相关参数后，再点击“配置”按钮调试参数（步骤与以上调试一致）

- 确保是滑条(Wheel)/滑轮(Slider)的模板，点击 EasyCodeCube 的“TK_触控”界面中的“+”号，界面中就增加一项滑动模块配置界面（有 x 条滑轮/滑条，就需要“+” x 组参数），目前 EasyCodeCube 最多支持四组滑动模块（92 系列只支持两组滑动模块）。以下举例一条滑轮：



- 配置选择滑动模块模型（根据客户项目而定），Slider 为滑条，Wheel 为滑轮；
- 通道数量需填入单个滑动模块所用的 TK 通道数；
- Max 档位代表触控模块输出的最大值；
- 顺序通道代表滑动模块中每个 TK 通道的位置，需要填入代表 TK 通道的序号，每个数字用逗号隔开，写在前面的通道代表其在滑动模块的越前端；
- 门限值完成调试触控后会由 EasyCodeCube 自动写入，无需手动写入；
- 重复步骤 c~f，直到配置完所有的滑动模块。

注：当发现多次测试工程中自动配置的门限值小于 20 时，属于错误情况，详情请联系赛元工程师。
如果项目需要进行 CS、EFT 等抗干扰测试时，需要人工配置门限值，详情请联系赛元工程师。

⑨ 低功耗参数设置

选择低功耗模板应用，点击模板进入后，需设置以下相关参数，再点击“配置”按钮调试参数

- 设置唤醒低功耗的按键数量，数值应小于配置通道数，按键个数越多，功耗越大。
注：低功耗唤醒按键数量推荐小于 12 个，否则会使芯片功耗较大。
- 设置低功耗唤醒按键的详细通道，按照顺序 TK 所用通道，以数字格式依次填写，中间用英文逗号隔开，结尾无需逗号。其实切记 TK 通道个数与设置的个数一致，且各不相同。
- 写入低功耗按键扫描间隔，进入低功耗模式后每次进行 TK 扫描间的间隔时间。
- 低功耗按键确认次数，低功耗扫描模式下扫描到有 TK 按键按下的次数大于此值，TK 按键按下为真实信号，芯片进入正常运行模式。

低功耗唤醒按键总数：	3
低功耗所用TK按键通道：	1, 2, 3
低功耗按键扫描间隔：	500MS
低功耗按键确认次数：	15

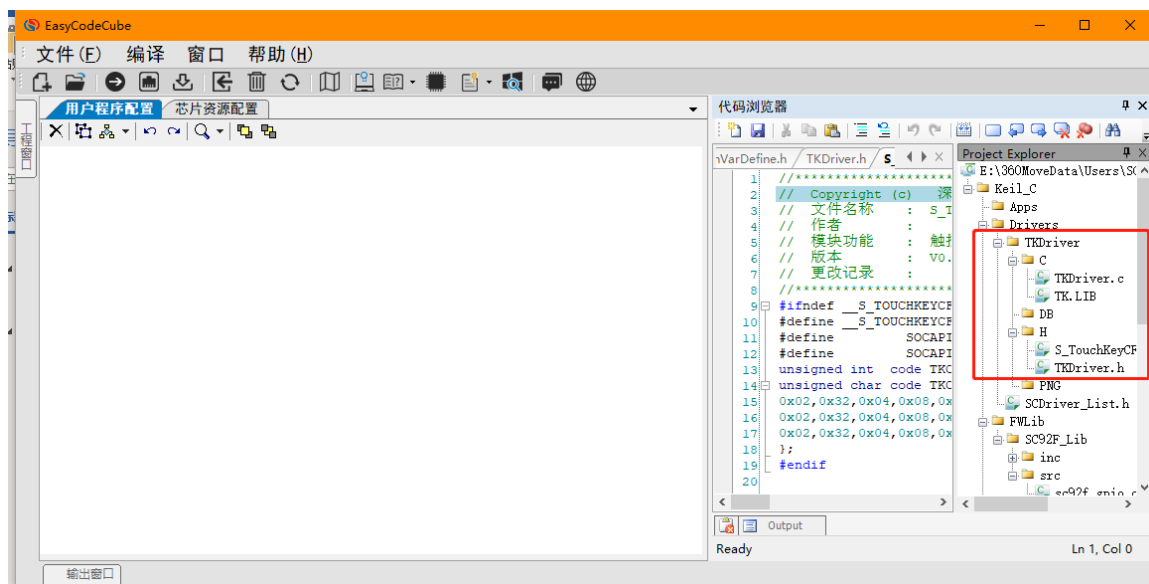
注：当使用低功耗时，请把没有使用到的引脚设置为强推挽模式高/低，以消除引脚上的电流对芯片低功耗时工作电流的影响。

⑩ 生成工程



点击“生成工程源代码”按钮，选择更新工程文件，等待生成步骤完成。

此时打开用户程序配置界面，可以在代码浏览器的 Project Explorer 窗口中发现 TK 相关文件，代表着 TK 触控库已经成功加入工程中。



可以点击相关文件可用查看相关内容，点击 S_TOUCHKEYCFG.H 可以发现。

弹簧触控调试结果 S_TOUCHKEYCFG.H 内容如下：

```

//*****
// Copyright (c) 深圳市赛元微电子有限公司
// 文件名称 : S_TouchKeyCFG.h
// 作者 :
// 模块功能 : 触控键配置文件
// 版本 : V0.2
// 更改记录 :
//*****
#ifndef S_TOUCHKEYCFG_H
#define S_TOUCHKEYCFG_H
#define SOCAPI_SET_TOUCHKEY_TOTAL 3 // 隔空按键为1
#define SOCAPI_SET_TOUCHKEY_CHANNEL 0x0000001C
unsigned int code TKCFG[17] = {0,0,0,5,10,3000,200,100,2,0,0,4,0,65535,65535,65535,22};
unsigned char code TKChannelCfg[3][8]={
0x04,0x52,0x04,0x08,0x30,0x05,0x01,0x93,
0x04,0x52,0x04,0x08,0x37,0x05,0x01,0xf3,
0x04,0x52,0x04,0x08,0x27,0x05,0x02,0x60,
};
#endif

```

生成的配置文件用于加入到项目工程中。配置文件的定义如下：

数据类型	说明	范围
SOCAPI_SET_TOUCHKEY_TOTAL	通道个数	1-31
SOCAPI_SET_TOUCHKEY_CHANNEL	通道对应数据位	0x00000001-0xffffffff
TKCFG[0]	应用类型	0-1 0为弹簧，1为隔空
TKCFG[1]	按键类型	0-1, 0为单按键 1为双
TKCFG[2]		保持默认 0 不改动
TKCFG[3]	按键确认次数	3-50
TKCFG[4]		保持默认 10 不改动
TKCFG[5]	按键最长输出	0-5000
TKCFG[6]		保持默认 200 不改动
TKCFG[7]		保持默认 100 不改动
TKCFG[8]		保持默认 2 不改动
TKCFG[9]		保持默认 0 不改动
TKCFG[10]		保持默认 不改动
TKCFG[11]		保持默认 4 不改动
TKCFG[12]		保持默认 0 不改动
TKCFG[13]		保持默认 65535 不改动
TKCFG[14]		保持默认 65535 不改动
TKCFG[15]		保持默认 65535 不改动
TKCFG[16]	噪声值	3-50
TKChannelCfg[][0]		保持默认 不改动
TKChannelCfg[][1]		保持默认 不改动
TKChannelCfg[][2]		保持默认 不改动
TKChannelCfg[][3]	扫描周期	0x01-0x20
TKChannelCfg[][4]		保持默认 不改动
TKChannelCfg[][5]		保持默认 不改动
TKChannelCfg[][6]	阈值高 8 位	0x00-0xff
TKChannelCfg[][7]	阈值低 8 位	0x01-0xff

2.3 按键滑轮滑条测试功能调试

主要功能：可直接在上位机测试按键参数手感，观察参数是否适应整机。按键类型包含：按键，滑条，滑轮。



以下为测试流程：

(1) 在测试按键/滑轮/滑条功能前，应保证手指按在单个 pad 中心位置时，获取滑轮/滑条 TK 单个通道参数，如图所示：红色箭头指示点

(2) 在“启动诊断”及“导出配置信息”操作后，选择某一种按键类型：以下以滑轮为例。

赛元高灵敏度触控调试软件

升级(U) 语言(L) 关于(A)

①基础参数设置

IC型号: 52: SC95F8522 应用类型: 0: 弹簧按键 按键类型: 0: 单按键 隔空距离(mm): 0 按键确认次数: 5
 自动校准次数: 10 按键最长输出: 3000 动态更新基线: 200 基线更新速度: 100 基线复位速度: 2
 滤波系数: 0.4 抗干扰设置: 0: 关闭 参考电压: 4 调试模式选择: 0: 静态调试

通道配置 通道选择 ☐ 反选 ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐ 13 ☒ 14 ☒ 15 ☐ 16 ☐ 17 ☒ 18 ☐ 19 ☐ 20 ☐ 21 ☐ 22 ☐ 23 ☐ 24 ☐ 25 ☐ 26 ☐ 27 ☐ 28 ☐ 29 ☐ 30 ☐ 31

②单通道调试

☒ 时钟 ☒ 分辨率 ☒ 增益 ☒ 扫描周期 ☒ 阈值 ☒ 数据值 ☒ 电容值PF ☒ 信噪比 ☒ 变化率 ☒ 变化量 ☒ 数据修正

	TK14	TK15	TK18
时钟	3	3	3
分辨率	50	50	50
增益	4	4	4
扫描周期	8	8	8
阈值	5	5	5
数据值	3815	3857	3933
电容值PF	8	8	8
信噪比	49	44	49

③按键诊断

当前通道 TK18 TK14 TK15 TK18

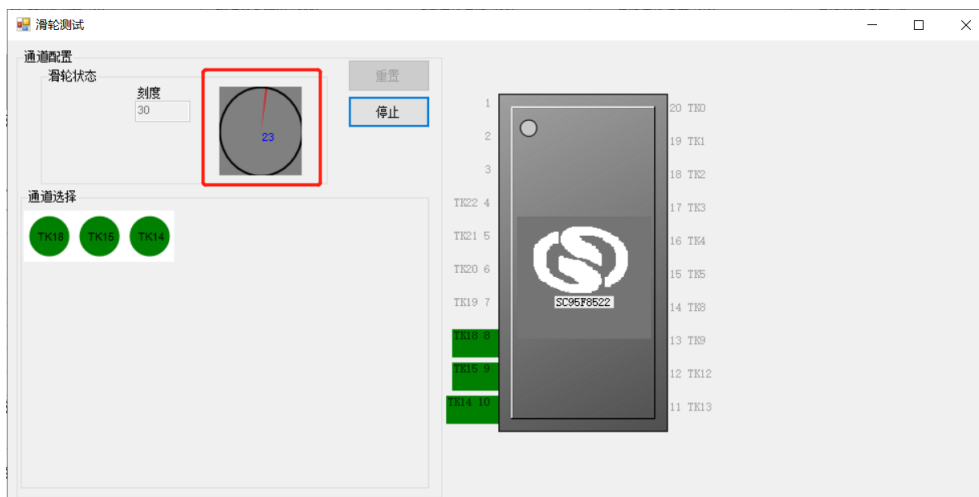
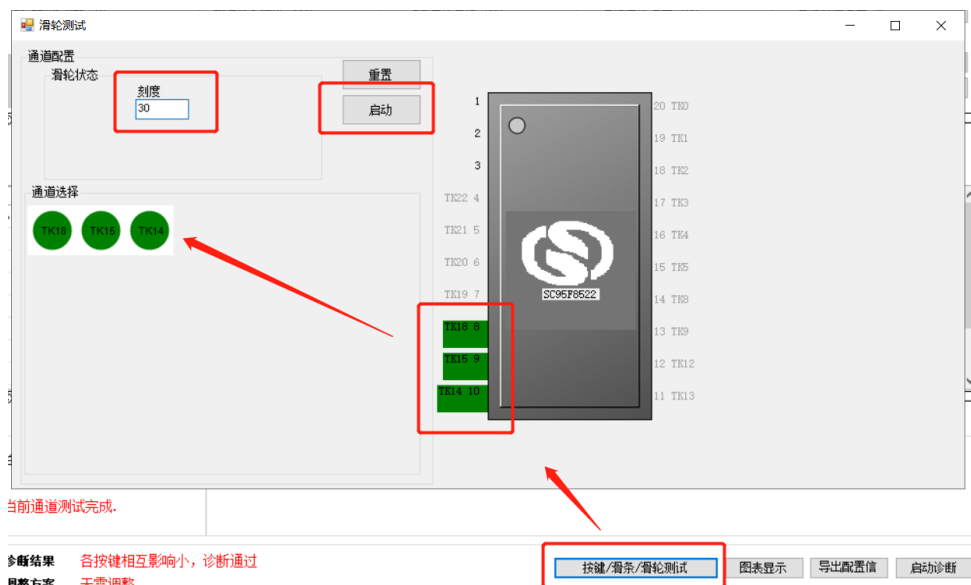
当前通道测试完成.

诊断结果 各按键相互影响小, 诊断通过
 调整方案 无需调整

按键/滑条/滑轮测试

(3) 按键/滑条/滑轮测试：选择滑轮测试

1. 设置刻度值：对应滑轮最大刻度
 2. 设置滑轮通道顺序：注意选择的顺序，需按着硬件上滑轮 TK 通道顺序设置，例如以下图片滑轮按着 TK18->TK15->TK14 的顺序排布
 3. 点击启动，滑动硬件上滑轮按键，可在上位机上观察到滑轮效果和手感。
 4. 测试完点击停止按钮即可，随后关闭该界面。
- 补充：重置按钮是在启动前，需重新设置，可点击重置按钮。



注意点：

“滑条/滑轮测试”和“按键测试”设置的按键不能重复，且不能共用。

先设置完滑条/滑轮的按键测试后，再次选择该 TK 通道“按键测试”，上位机此时无法测试单个按键功能。

要体验滑条/滑轮/按键测试功能需更新最新的高灵敏静态调试文件。

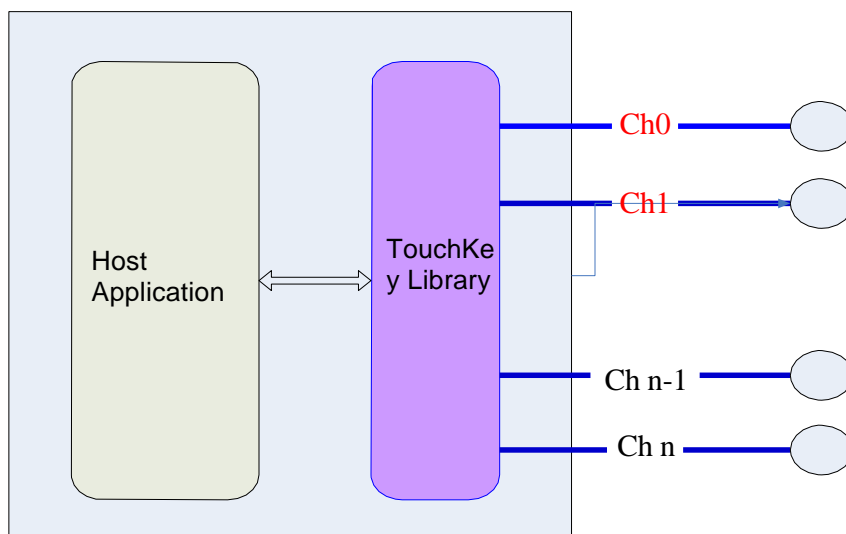
按键测试项，不需要设置刻度参数，直接点击启动即可，可在点击对应 TK 按键在上位机观察按键情况。

2.4 用户程序和赛元触控软件库的融合思路

2.4.1 主程序和库文件的整体结构关系

通过连接库文件，并在用户程序中包含指定的头文件，调用库内的接口函数即可以增加触控按键的功能。库函数仅在主程序调用时才会运行。库文件会占用一些 ROM、RAM、寄存器、中断等资源，但不占用定时器资源。库函数只管触控按键功能，用户必须自己处理其他的控制部分，如：输入输出、LED 数码管显示、通讯等功能。库函数和用户主程序的结构如下：

(下面附图中紫色部分表示是赛元软件库，其他部分是用户程序)

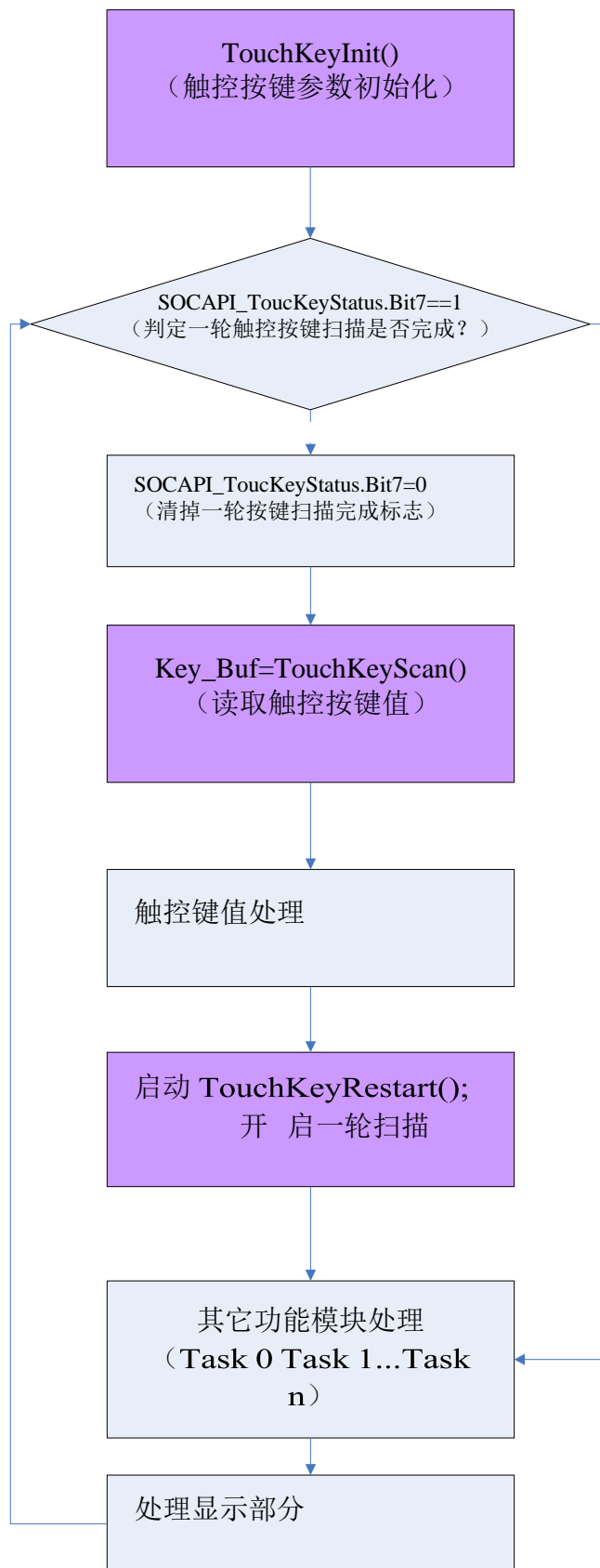


2.4.2 库文件的调用流程。

用户通过一定的流程调用库文件的接口函数，便可得到触控按键的键值。

- ① 将TK对应的IO 设置为强推挽输出高。
- ② 主程序调用接口函数“TouchKeyInit()”用于配置触控按键通道的参数，并初始化 Baseline 基线；
- ③ 主程序通过查看全局变量 SOCAPI_TouchKeyStatus&0x80 来判定一轮触控按键扫描是否完成；
- ④ 主程序调用接口函数“TouchKeyScan()”用于读取触控按键值；
- ⑤ 主程序调用“TouchKeyRestart()”启动下一轮扫描。

(下图中紫色的部分是库文件，其它部分是用户程序)



用户程序调用接口函数控制流

2.4.3 主程序和库文件的时序关系。

因为运行触控按键库消耗了部分 IC 资源和时间，为了让用户程序和库程序能完美融合，主程序需要遵循以下要求：

- ① 提供给库运行的资源 ROM、RAM 和时间；
 - ② 启动按键扫描后，在一轮扫描未完成之前，不能对触控按键通道进行操作；如触控按键通道为输出 IO；否则触控按键功能将无法实现；
 - ③ 保证有足够的堆栈深度提供给主程序和库函数；
 - ④ 触控按键扫描取计数值数据的动作，是在中断内实现的，但数据的算法处理是在主程序中完成的。
- 用户需要按照一个合理的频度来调用库函数检测按键，以免错过按键动作。

2.4.4 软件融合的注意事项

运行时间：

- i TouchKeyInit(void)：算法执行时间会因按键选择个数的增减而增减，200~500mS；
- ii TouchKeyScan(void)：执行该函数用时约为(235*N 个按键)uS。

2.4.5 整体 Code 的测试

用户完成程序调用后，请详细测试相关功能的性能，以防止软件的冲突。如发生异常情况，请在程序流程、调用时序、时间分配、堆栈、ROM/RAM/INT 资源等部分查找原因。关于整机调试的建议：因为元器件的性能差异，建议用户可在一块 PCB 完成调试的情况下，多测试一些 PCB 的效果，以便取到折中效果的参数来去除材料对一致性的影响。

2.5 注意事项

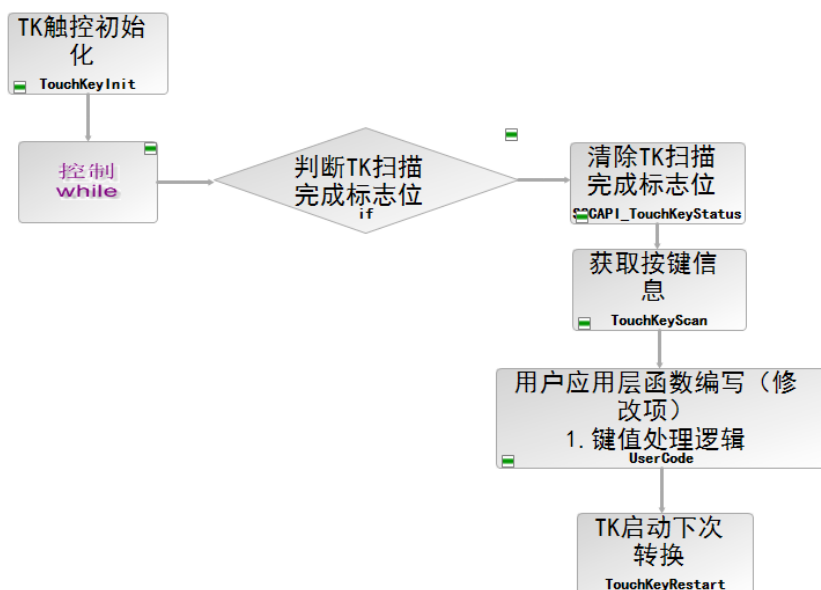
- 1、使用单面 PCB 板，一般用弹簧片来做触控按键。因为其侧面也能同手指形成电场，使用弹簧片比使用 PCB 上覆铜做触控按键能获得更高的灵敏度。
- 2、从感应盘到 IC 管脚的连线长度尽量不绕太远，尽量避免连线之间的耦合电容，也要避免与其他高频信号线有耦合电容。
- 3、灵敏度与感应盘面积成正比，与外壳厚度成反比。根据外壳厚度和尺寸选择合适的触控面积。一般玻璃外壳比塑料具有更高的穿透力。
- 4、感应盘与感应盘之间应该尽量留一定的间距，以保证手指头触控时不会覆盖到 2 个感应盘，同时也能防止感应盘寄生电容过大。
- 5、基准电容是赛元触控感应电路的充放电电容，是实现触控功能的重要器件，它保障了触控电路的正常工作，其容值范围为472-104，推荐使用103电容，材质无特殊要求。
。
- 6、应用程序中需要将 TK 对应的 IO 口设置为强推挽输出高

3 代码例程

以下黄色区域是客户可修改函数实现

3.1 按键使用

3.1.1 图形化编程



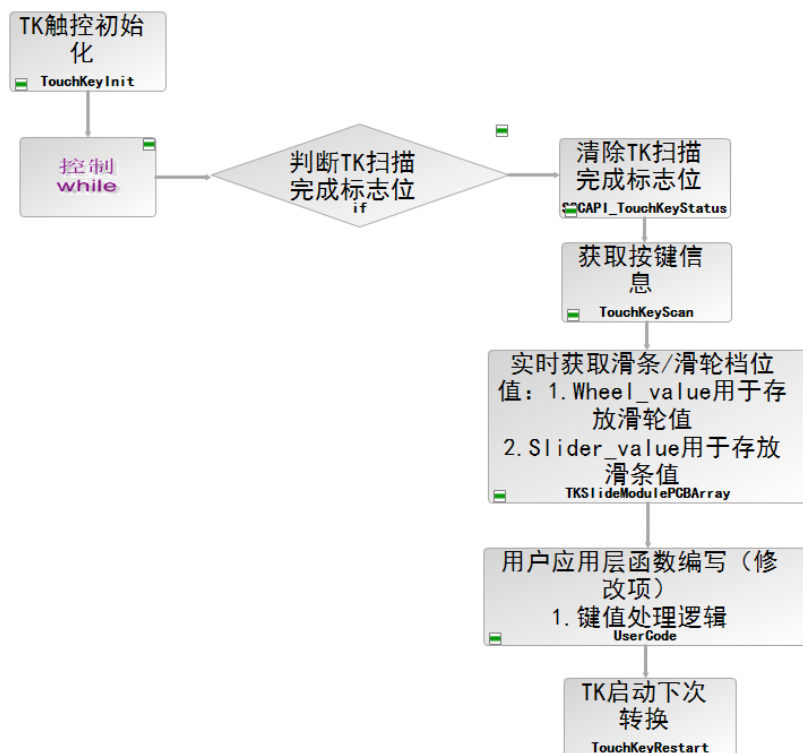
3.1.2 C 语言代码

```

/** MCU 初始化函数 */
SC_Init();
/*<UserCodeStart>/**<SinOne-Tag><9>*/
TouchKeyInit();
/*<UserCodeEnd>/**<SinOne-Tag><9>*/
/*<UserCodeStart>/**<SinOne-Tag><10>*/
while(1)
{
    /*<UserCodeStart>/**<SinOne-Tag><14>*/
    if(SOCAPI_TouchKeyStatus & 0x80)
    {
        /*<UserCodeStart>/**<SinOne-Tag><22>*/
        SOCAPI_TouchKeyStatus &= 0x7f;
        /*<UserCodeEnd>/**<SinOne-Tag><22>*/
        /*<UserCodeStart>/**<SinOne-Tag><18>*/
        exKeyValueFlag = TouchKeyScan();
        /*<UserCodeEnd>/**<SinOne-Tag><18>*/
        /*<UserCodeStart>/**<SinOne-Tag><35>*/
        UserCode();
        /*<UserCodeEnd>/**<SinOne-Tag><35>*/
        /*<UserCodeStart>/**<SinOne-Tag><36>*/
        TouchKeyRestart();
        /*<UserCodeEnd>/**<SinOne-Tag><36>*/
    }
    /*<UserCodeEnd>/**<SinOne-Tag><14>*/
}
/*<UserCodeEnd>/**<SinOne-Tag><10>*/
  
```

3.2 按键，滑轮和滑条三者合用

3.2.1 图形化编程



3.2.2 C 语言代码

```

/** MCU 初始化函数 */
SC_Init();
/*<UserCodeStart>/**<SinOne-Tag><9>*/
TouchKeyInit();
/*<UserCodeEnd>/**<SinOne-Tag><9>*/
/*<UserCodeStart>/**<SinOne-Tag><10>*/
while(1)
{
    /*<UserCodeStart>/**<SinOne-Tag><14>*/
    if(SOCAPI_TouchKeyStatus & 0x80)
    {
        /*<UserCodeStart>/**<SinOne-Tag><22>*/
        SOCAPI_TouchKeyStatus &= 0x7f;
        /*<UserCodeEnd>/**<SinOne-Tag><22>*/
        /*<UserCodeStart>/**<SinOne-Tag><18>*/
        exKeyValueFlag = TouchKeyScan();
        /*<UserCodeEnd>/**<SinOne-Tag><18>*/
        /*<UserCodeStart>/**<SinOne-Tag><63>*/
        Wheel_value = TKSlideModulePCBArray[0].OutValue;
        Slider_value = TKSlideModulePCBArray[1].OutValue;
        /*<UserCodeEnd>/**<SinOne-Tag><63>*/
        /*<UserCodeStart>/**<SinOne-Tag><35>*/
        UserCode();
        /*<UserCodeEnd>/**<SinOne-Tag><35>*/
        /*<UserCodeStart>/**<SinOne-Tag><36>*/
        TouchKeyRestart();
        /*<UserCodeEnd>/**<SinOne-Tag><36>*/
    }
}
  
```

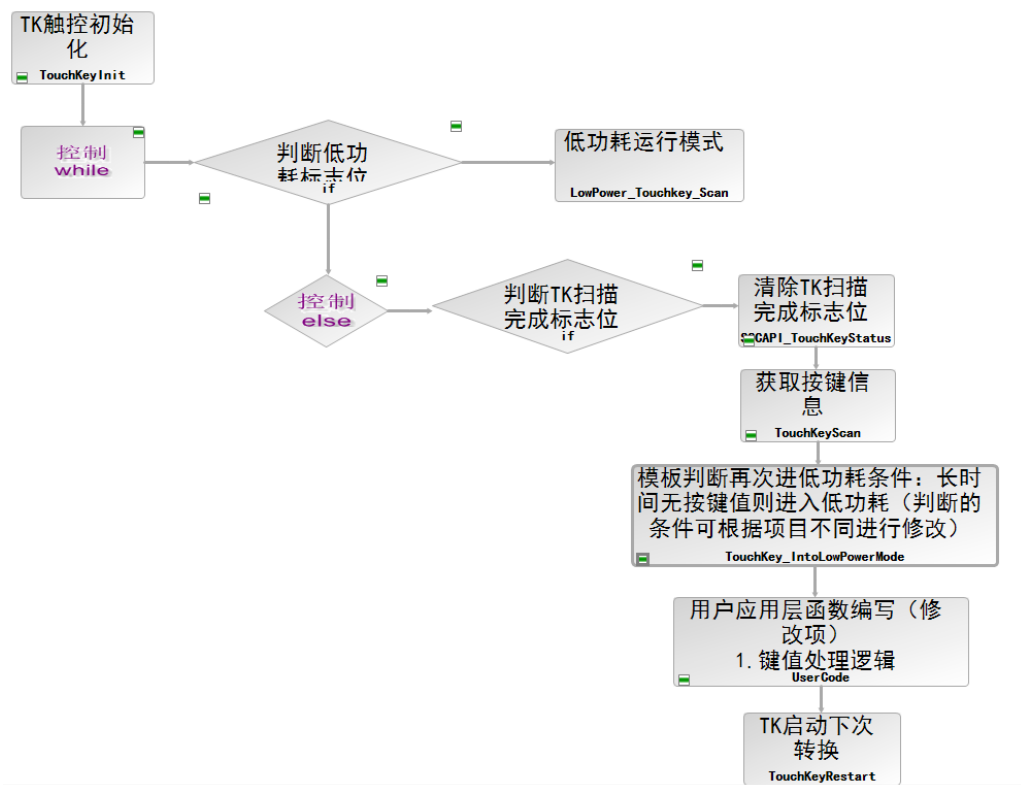
```
/*<UserCodeEnd>*//*<SinOne-Tag><14>*/
```

```
}
```

```
/*<UserCodeEnd>*//*<SinOne-Tag><10>*/ /*<UserCodeEnd>*//*<SinOne-
```

3.3 低功耗按键使用

3.3.1 图形化编程



3.3.2C 语言代码

```

/** MCU 初始化函数 */
SC_Init();
/*<UserCodeStart>*//*<SinOne-Tag><9>*/
TouchKeyInit();
/*<UserCodeEnd>*//*<SinOne-Tag><9>*/
/*<UserCodeStart>*//*<SinOne-Tag><10>*/
while(1)
{
    /*<UserCodeStart>*//*<SinOne-Tag><55>*/
    if(GetLowPowerScanFlag())
    {
        /*<UserCodeStart>*//*<SinOne-Tag><56>*/
        LowPower_Touchkey_Scan();
        /*<UserCodeEnd>*//*<SinOne-Tag><56>*/
    }
    /*<UserCodeEnd>*//*<SinOne-Tag><55>*/
    /*<UserCodeStart>*//*<SinOne-Tag><57>*/
    else
    {
        /*<UserCodeStart>*//*<SinOne-Tag><14>*/
        if(SOCAPI_TouchKeyStatus & 0x80)
        {
            /*<UserCodeStart>*//*<SinOne-Tag><22>*/
            SOCAPI_TouchKeyStatus &= 0x7f;
            /*<UserCodeEnd>*//*<SinOne-Tag><22>*/
        }
    }
}

```

```

/*<UserCodeStart>*/*<SinOne-Tag><18>*/
exKeyValueFlag = TouchKeyScan();
/*<UserCodeEnd>*/*<SinOne-Tag><18>*/
/*<UserCodeStart>*/*<SinOne-Tag><65>*/
if(exKeyValueFlag == 0 )
{
    TK_NoKeyCount ++;
    if(TK_NoKeyCount > 100)
    {
        TK_NoKeyCount = 0;
        TouchKey_IntoLowPowerMode();
    }
}
else
{
    TK_NoKeyCount = 0;
}
/*<UserCodeEnd>*/*<SinOne-Tag><65>*/
/*<UserCodeStart>*/*<SinOne-Tag><35>*/
UserCode();
/*<UserCodeEnd>*/*<SinOne-Tag><35>*/
/*<UserCodeStart>*/*<SinOne-Tag><36>*/
TouchKeyRestart();
/*<UserCodeEnd>*/*<SinOne-Tag><36>*/
}
/*<UserCodeEnd>*/*<SinOne-Tag><14>*/
}
/*<UserCodeEnd>*/*<SinOne-Tag><57>*/
}
/*<UserCodeEnd>*/*<SinOne-Tag><10>*/

```

注：以上框架是低功耗实现逻辑，只需在 **UserCode()** 和回调函数内修改即可，建议不要自行添加除现有 **if**、**else** 之外其他分支判断，只能在现有逻辑框架内添加函数实现。

3.3.3 回调函数的编辑

回调函数的编辑的步骤如下：

- ① EasyCode 中“芯片资源配置”启用 TK 低功耗资源；
- ② 打开“用户程序配置”界面；
- ③ 在右侧的“工具窗口”的“工具栏”中，找到“用户工程列表”，可以看到 TK 触控的相关函数；

右击需要用到回调函数，即可进入相应的回调函数编辑界面，写入相应的执行代码。

注：在右侧工程 **SysFunVarDefine.c** 中可找到该回调函数实现。

回调函数编辑界面



右击相应的回调函数



更改记录

版本	记录	日期
V0.05	1. 更新增添 TK 模板开发说明	2021 年 7 月 15
V0.05	2. 更新增添 TK 函数实现说明	2021 年 7 月 6
V0.04	3. 更新修改 TK 小资源芯片使用方法	2021 年 6 月 28
V0.03	4. 增加小 ROM 资源的说明 5. 增加低功耗按键说明 6. 修改图片	2021 年 6 月 18
V0.02	1.优化低功耗的说明	2021 年 6 月 07
V0.01	1.完成初版	2021 年 6 月 04