

## 目录

<b>1 赛元 TOUCHKEY MCU 应用指南总体描述 .....</b>	<b>2</b>
<b>2 赛元触控库介绍.....</b>	<b>3</b>
2.1 触控库应用类型.....	3
2.2 触控项目开发简要步骤.....	3
2.3 赛元触控库文件介绍 .....	3
<b>3 触控开发流程 .....</b>	<b>4</b>
3.1 安装开发工具 .....	4
3.2 调试触控参数 .....	5
3.2.1 高灵敏度调试触控参数 .....	5
3.3 实现赛元软件库的功能测试 .....	12
3.3.1 高灵敏库触控软件库移植.....	12
3.4 完成用户程序和赛元触控软件库的融合 .....	16
3.4.1 高灵敏库触控软件 and 用户程序 .....	16
3.4.2 注意事项 .....	18
3.5 附加功能-动态调试功能.....	18
3.5.1 高灵敏度动态调试步骤 .....	18
附录.....	22
<b>4 规格更改记录 .....</b>	<b>26</b>
<b>声明.....</b>	<b>26</b>

## 1 赛元 TOUCHKEY MCU 应用指南总体描述

本文档是赛元 Touchkey MCU 触控的应用指南，主要介绍如何使用赛元提供的触摸按键库文件以及触控上位机如何调试参数。其特点如下：

- 高灵敏度模式可适应普通触摸按键、隔空触摸按键、滑轮滑条、接近感应等对灵敏度要求较高的触控应用
- 高灵敏度具有很强的抗干扰能力
- 最多可实现 31 路触摸按键及衍生功能
- 高灵活度开发软件库支持，低开发难度
- 自动化调试软件支持，智能化开发
- 部分型号可以在 MCU STOP 模式下进入低功耗模式工作

不同系列芯片低功耗功耗有所差异，其中，12 个触摸按键 500mS 唤醒时芯片功耗最低可至 9uA@5V。

用户通过使用赛元提供的触控按键库文件，可选择触控模式并快速简单实现所需的触摸功能。用户可以通过下表的信息选择最适合当前应用的触控模式：

说明	高灵敏度模式
特点	<ul style="list-style-type: none"><li>●超强抗干扰能力，可通过 10V 动态 CS</li><li>●超高灵敏度</li></ul>
适用的应用	<ul style="list-style-type: none"><li>●弹簧触摸按键应用</li><li>●隔空触摸按键应用</li><li>●接近感应应用</li><li>●滑轮滑条应用</li><li>●对灵敏度要求较高的触控应用</li></ul>
如何进入模式	通过项目工程载入赛元所提供的触控库来选择高灵敏度模式
库体说明	<a href="#">3.3.1 高灵敏库触控软件库移植</a>
对应的库文件	“SC9XF8XXX_HighSensitive_Lib_Tn_Vx.x.x.LIB”
注意事项	<ul style="list-style-type: none"><li>●T1 库应用于覆盖面板层紧贴触摸按键/弹簧类型的应用，支持组合按键</li><li>●T2 库应用于隔空类型的应用，且按键个数至少 3 个以上，不支持组合按键</li></ul>
选择说明	通常状况下建议使用此高灵敏度模式，将会获得更佳的使用体验。

### 赛元系列 Touchkey MCU 触摸芯片供电电源注意事项：

- 触摸芯片供电电源范围：参考不同芯片规格书要求范围使用
- 触摸芯片供电电源纹波：推荐触摸芯片工作电源电压纹波应控制小于 100mv，最大不超过 200mv。

## 2 赛元触控库介绍

### 2.1 触控库应用类型

赛元 Touchkey MCU 提供一个可以供用户调用的库文件，以降低用户触摸按键部分的开发难度。大体分为以下几类库体类型：

- 普通触摸按键库
  - 弹簧触控库（简称 T1 库）
  - 隔空触控库（简称 T2 库）
- 滑轮滑条触控库
- 接近感应触控库
- 低功耗触控库（包含普通低功耗触控库，滑轮滑条低功耗触控库）

本文将初步介绍普通触摸按键库体使用以及触控调试上位机 SOC TouchKey Tool Menu 软件使用，滑轮滑条库以及接近感应库体将由特殊应用指南详细介绍使用，详细请查看：《赛元 TK 触控特殊应用说明》

用户仅仅需要经过以下几个步骤，便可实现触摸按键的功能，并将赛元的触控软件库跟用户的软件完美结合，实现最终的产品功能。

### 2.2 触控项目开发简要步骤

完整触控项目开发分以下几个步骤：

#### 1. 安装开发工具并配置参数、导出配置参数

赛元提供了专门的触控调试上位机软件 SOC TouchKey Tool Menu，方便用户能通过一系列的人机交互完成调试工作，用户需要安装此软件，并配合 DPT52/SC-LINK/SC-LINK PRO 在线烧录器使用。用户可通过软件界面配置参数来找到用户 PCB 最合适的触摸按键关键参数，并将最终的相关参数导出生成头文件加入到用户工程中使用。

#### 2. 实现赛元软件库的功能测试

将步骤 1 生成的配置文件加入到赛元触控软件库中，将整个库相关文件加入到用户项目工程进行编译。赛元提供简单的测试程序，可供用户完成按键部分功能的测试。

#### 3. 完成用户程序和赛元触控软件库的融合

用户自行写好除触摸按键以外的其它部分软件，并将赛元的软件库嵌套进用户程序中，从而完成整个产品的整体功能。

详细开发操作流程请至相关章节查看：[3 触控开发流程](#)

### 2.3 赛元触控库文件介绍

赛元触控库包括以下几个文件：

**SensorMethod.h**：该文件是触控库对外的接口函数声明。用户需要在主程序引用该头文件。

**SC9XF8XXX\_X\_X\_Vx.x.x.LIB**：该文件是触控库算法部分，用户需要将该文件加入工程进行编译

**S\_TOUCHKEYCFG.H**：该文件是触控相关参数的配置文件。（用户通过 SOC TouchKey Tool Menu 软件调试后生成）

**S\_TouchKeyCFG.C**：该文件包含触控参数头文件与触控库交互的相关接口，用户需要将文件加入工程编译。普通触控库该文件无需修改，请知悉。

而滑轮滑条触控库及接近感应触控库需要进行参数配置，详细修改项请移至特殊应用指南进行查阅：

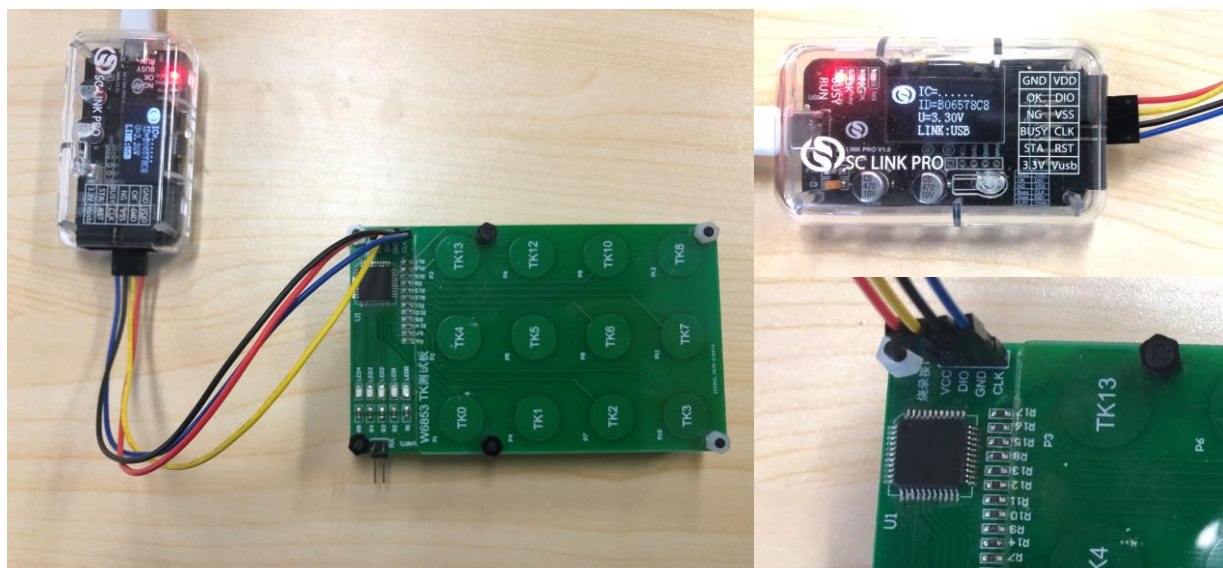
《赛元 TK 触控特殊应用说明》。

### 3 触控开发流程

本章节主要介绍的是如何进行完整的触控项目开发。需要注意的是在项目开发之前，完整的触控板 PCBA 也是项目调试中不可或缺的部分，关于触控 MCU Layout 请参考《赛元触摸按键 MCU PCB 设计要点》。保证触控项目硬件符合要求再进行开发，将减少开发过程中所遇到的问题。

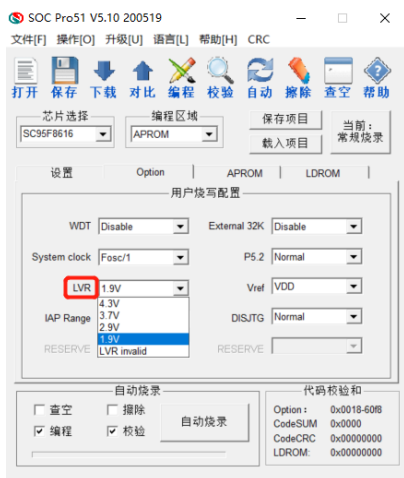
#### 3.1 安装开发工具

- 1) Setup SOC Pro51/SOC Programming Tool  
安装赛元软件 SOC Pro51 Vx.xx.exe/ SOC Programming Tool (请从赛元网站找最新版本)。
- 2) Setup SOC TouchKey Tool Menu  
安装赛元触控调试软件 SOC TouchKey Tool Menu (请从赛元网站找最新版本)。
- 3) 升级 DPT52/SC-LINK/SC-LINK PRO 固件,更新 MCU 库  
在线烧写器 DPT52/SC-LINK/SC-LINK PRO 的固件和 SOC Pro51/ SOC Programming Tool 的 MCU 库文件需升级到赛元官网最新版本。
- 4) 安装 SOC\_KEIL 插件;  
请将赛元 MCU 的插件安装文件更新到官网最新版本。安装方法及注意事项如下:  
a.安装 SOC\_KEIL 插件,此插件可自动查找系统中安装的 KEIL (C51 版本) 的安装目录,并将所有文件安装到 KEIL C 安装目录下 C51 目录内 SinOne\_Chip/SinOne\_Chip\_SCLinkPRO 目录。  
b.SinOne\_Chip /SinOne\_Chip\_SCLinkPRO 目录内所有文件如下:  
CDB: 赛元 MCU 开发库文件  
DEMO: 赛元 MCU 示例程序  
INC: 赛元 MCU 头文件  
PDF: 赛元 SOC 烧录仿真工具 SC-LINK PRO 使用说明  
SOC\_Debug\_Driver/SCLINK\_PRO\_Debug\_Driver: 赛元仿真插件  
c.赛元 SOC\_KEIL 插件会新建一个赛元 MCU 专用列表,不会覆盖掉 KEIL C 原有的 MCU 列表。  
d.如果无法安装 SOC\_KEIL 插件,请检查您的 KEIL 是否是 C51 版本。
- 5) 硬件连接顺序: 电脑 USB-->DPT52/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 SCLINK PRO 的接线。

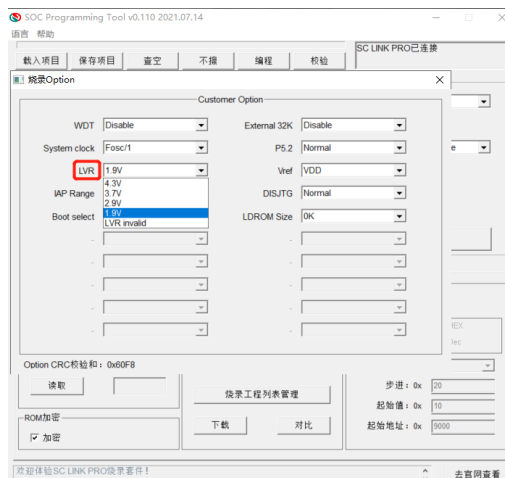


- 6) 烧录静态调试代码 hex 文件到用户 PCB 上的 SC9XF8XXX IC 中  
打开 SOC Pro51/SOC Programming Tool 软件,选择项目使用的 MCU 型号,载入静态调试代码 hex 文

件, 点击“编程”, 完成后关闭 SOC Pro51/ SOC Programming Tool 软件, 重新拔插 USB 上电。(注意:LVR 设置必须低于供电电压,如供电为 3.3V,则 Option 中 LVR 必须选择 3.3V 以下的档位)  
见以下 SOC Pro51/SOC Programming Tool 软件所示图:



SOC Pro51 软件



SOC Programming Tool 软件

高灵敏调试见以下操作步骤: [3.2.1 高灵敏度调试触控参数](#)

## 3.2 调试触控参数


### 3.2.1 高灵敏度调试触控参数

需提前烧录好对应芯片的静态调试码, 具体操作请看 [3.1 安装开发工具](#) 第 6 项

#### 1) 打开 Touch Key Tool Menu, 选择高灵敏度触控

常规: 普通触摸按键调试, 滑轮滑条按键调试, 接近感应按键调试。

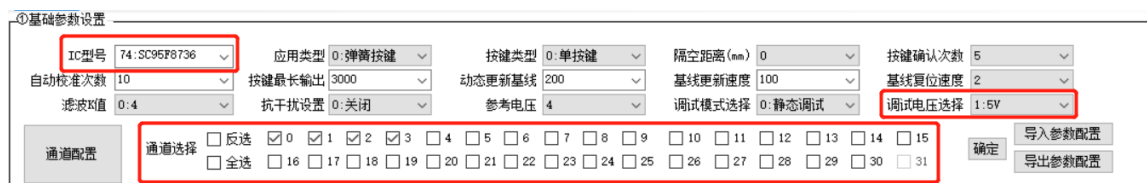
含低功耗: 普通触控低功耗按键调试, 滑轮滑条低功耗按键调试。

 SOC TouchKey Tool V2.00



#### 2) 参数配置, 进入触控参数调试

① 选择项目使用的 MCU 型号以及勾选使用的 TK 通道, 如图所示:



**隔空距离:** 针对隔空按键应用需要设置选择 0~3mm。其他应用无需设置。

(更远距离请联系赛元工程师协助)

**调试电压选择:** 与项目中赛元 MCU 芯片 VDD 供电电压有关。

5V 项目选择 5V 调试, 3.3V 项目选择 3.3V 调试。

② 配置触控算法运算的相关参数 (保持默认参数不改动, 以下为各个参数相关介绍)

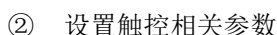
**按键确认次数:** 建议保持默认。该参数决定触控算法运行的出键速度, 出键速度与一轮按键扫描时间有关, 若扫描一轮按键需要 12MS, 按键确认次数为 5 次, 则按键需要的响应时间为  $5 \times 12\text{MS} = 60\text{MS}$ 。

**自动校准次数:** 建议保持默认。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



注意：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。

用户点击“确定”按键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。





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。

**时钟：**保持默认，不进行改动

**分辨率：**保持默认，不进行改动 **增益：**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。

### ③ 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：



触摸过程：



普通按键调试阶段将手垂直紧贴于按键感应面上方



滑轮滑条调试阶段将手垂直紧贴于锯齿中心感应面上方，如左图所示白色区域

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！

单通道调试结束: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

3

数据值

3794

分辨率

50

电容值PF

6

增益

4

信噪比

102

扫描周期

8

变化率

405

阈值

5

变化量

1537

数据修正

22

当前调试通道

TK0



当前通道测试完成.

**限定条件**

当前参数满足度值: 8000

CP电容要求: <48PF

信噪比要求: >5

变化量要求: > 75

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

3

数据值

3795

分辨率

50

电容值PF

6

增益

4

信噪比

0

扫描周期

8

变化率

0

阈值

5

变化量

1

数据修正

22

当前调试通道

TK0



当前通道测试完成.

**限定条件**

当前参数满足度值: 8000

CP电容要求: <48PF

信噪比要求: >5

变化量要求: > 75

数据修正值: 0 ≤ N ≤ 128

图表显示

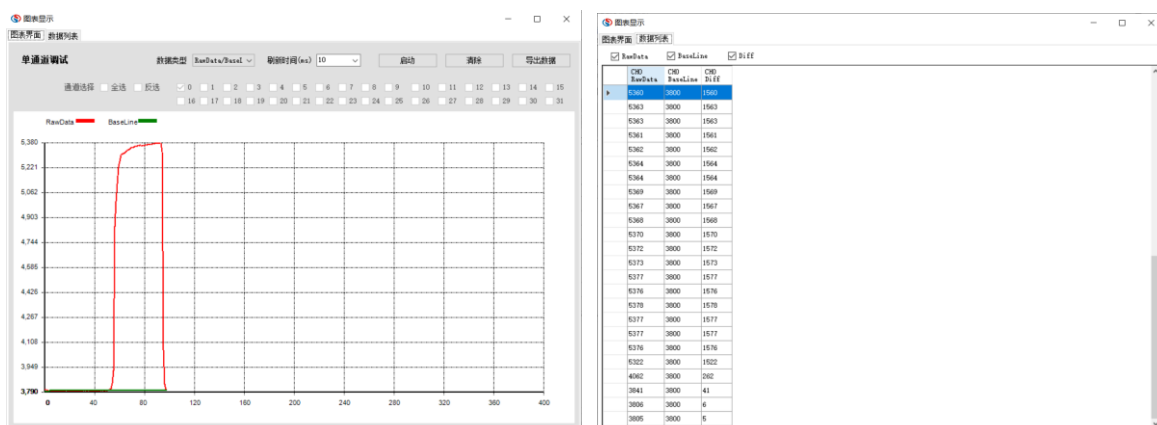
启动调试

不通过的项目相应会红色字体标出。

依次调试每个按键，调至按键均通过。

注：附加观察项（调试非必要流程）

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



④ 进行按键诊断（只针对普通触摸按键进行诊断，滑轮滑条不需要诊断，过程中遇到滑轮滑条按键诊



断无需操作，静待切换至普通按键在进行诊断指示操作)

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



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！




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提示语：



- ⑤ 完成按键诊断并且测试通过后，点击“导出配置信息”按钮生成配置文件 S\_TOUCHKEYCFG.H 将生成的配置文件保存（后续触控软件库移植融合步骤会用到，请保存好）。



S\_TOUCHKEYCFG.H 内容如下：

```

1 //*****
2 // Copyright (c) 深圳市赛元微电子有限公司
3 // 文件名称 : S_TouchKeyCFG.h
4 // 作者 :
5 // 模块功能 : 触控键配置文件
6 // 版本 : V0.2
7 // 更改记录 :
8 //*****
9 #ifndef S_TOUCHKEYCFG_H
10 #define S_TOUCHKEYCFG_H
11 #define SOCAPI_SET_TOUCHKEY_TOTAL 4
12 #define SOCAPI_SET_TOUCHKEY_CHANNEL 0x0000000F
13 unsigned int code TKCFG[17] = {0,0,0,5,10,3000,200,100,2,0,0,4,0,1,65535,65535,20};
14 unsigned char code TKChannelCfg[4][8]={
15 0x03,0x32,0x04,0x08,0x16,0x05,0x02,0xcf,
16 0x03,0x32,0x04,0x08,0x19,0x05,0x02,0x97,
17 0x03,0x32,0x04,0x08,0x1b,0x05,0x02,0x66,
18 0x03,0x32,0x04,0x08,0x1c,0x05,0x02,0xb0,
19 };
20 #endif
21

```

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	0-3 0为弹簧 1为隔空 3为接近感应
TKCFG[1]	按键类型	0-1 0为单键 1为双键
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]		保持默认不改动
TKCFG[12]		保持默认不改动
TKCFG[13]		保持默认不改动
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

至此触摸按键的调试过程结束。

若用户调试完成后，需要微调灵敏度，可以改变 TKChannelCfg[][6] 和 TKChannelCfg[][7] 的数值，TKChannelCfg[][6]是阈值的高 8 位，TKChannelCfg[][7]是阈值的低 8 位，值越小，灵敏度越高，反之亦然。建议多调试机台整机，以便取到折中效果的参数来去除材料对一致性的影响。

## 5) 附加功能-按键/滑轮/滑条手感模拟功能测试:

主要功能：在“启动诊断”及“导出配置信息”操作后，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。

模拟功能测试按键类型包含：按键，滑条，滑轮。

以下为测试流程：

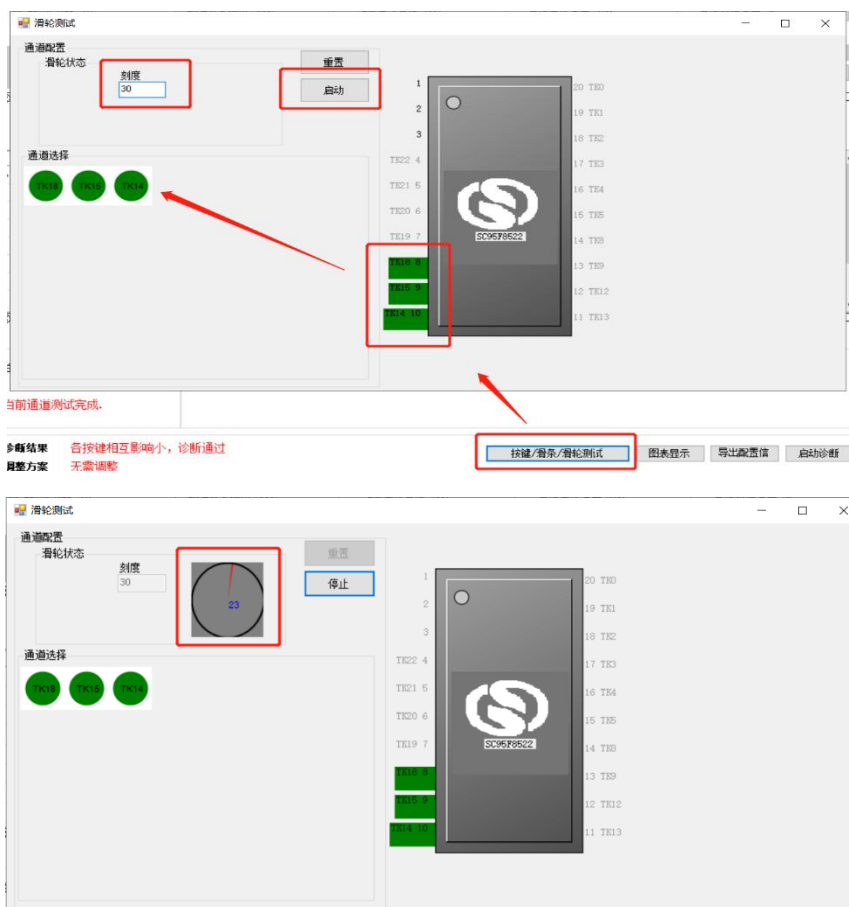
- 1) 在“启动诊断”及“导出配置信息”操作后，选择某一种按键类型：以下以滑轮为例。



- 2) 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”，无法测试单个按键功能。
- 要体验滑条/滑轮/按键测试功能请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。

### 3.3 实现赛元软件库的功能测试

#### 3.3.1 高灵敏库触控软件库移植

##### 1. 库文件介绍

1) 弹簧库文件（简称 T1 库，SC9XF8XXX\_HighSensitive\_Lib\_T1\_Vx.x.x.LIB）

2) 隔空库文件（简称 T2 库，SC9XF8XXX\_HighSensitive\_Lib\_T2\_Vx.x.x.LIB）

以下是 T1/T2 库文件简单介绍：（库体含有 4 个文件）

文件	用途	说明
<b>SC9XF8XXX_HighSensitive_Lib</b>	库文件，实现触摸按键检测算法	
<b>Sensormethod.h</b>	头文件，提供接口函数供用户调用	声明的函数可供外部调用
<b>S_TouchKeyCFG.C</b>	C 文件，实现触控参数与库交互	
<b>S_TOUCHKEYCFG.H</b>	头文件，提供宏供用户修改参数	

##### 2. Lib API 接口函数的调用说明

函数	用途	说明
<b>TouchKeyInit(void)</b>	触摸按键初始化	<p>1 用户在上电复位后调用一次；</p> <p>2 本函数通过 <b>S_TOUCHKEYCFG.H</b> 参数配置用户选定的按键通道、按键参数并初始化 <b>Baseline</b> 基线；</p> <p>3 执行本函数用时约：<b>200~500mS</b> 与按键个数，按键扫描时间，自动校准次数相关；按键每多 <b>N</b> 个，大约时间 <b>24M</b> 主频：<b>54 uS *N</b> 按键；<b>16M</b> 主频：<b>48 uS *N</b> 按键</p>
<b>TouchKeyRestart(void)</b>	使能一轮触摸按键扫描	<p>1 用户主程序控制何时启动按键扫描；</p> <p>2 启动按键扫描后，在一轮触摸按键扫描完成之前，不能对触摸按键通道进行操作：如操作触摸按键通道的 IO，否则触摸按键功能将无法实现。</p>
<b>Unsigned long int TouchKeyScan(void)</b>	触摸按键算法处理	<p>1 用户需要在触摸按键一轮扫描完成后调用；</p> <p>2 如用户未调用该函数之前，一定不能重新调用 <b>TouchKeyRestart()</b>，否则上一轮数据将被当前数据覆盖；</p> <p>3 执行该函数用时约为：<b>(50uS*N 个按键 @32M，340 uS*N 个按键 @24M)</b>；</p>

##### 3. 全局变量 SOCAPI\_TouchKeyStatus 的说明

1) 全局变量在 **S\_TouchKeyCFG.c** 头文件中声明

① **Unsigned char xdata SOCAPI\_TouchKeyStatus;**

② **SOCAPI\_TouchKeyStatus Bit7** 为 1 时表示当前一轮扫描按键完成；

2) 该变量在用户主程序中调用

**if(SOCAPI\_TouchKeyStatus&0x80)**时，调用 **TouchKeyScan(void)** 进行算法数据处理，给出键值；

3) 使能触摸按键扫描之前，一定要清掉标志。

清除一轮扫描标志 **SOCAPI\_TouchKeyStatus &=0x7f;**

##### 4. LIB API 接口函数的返回值说明

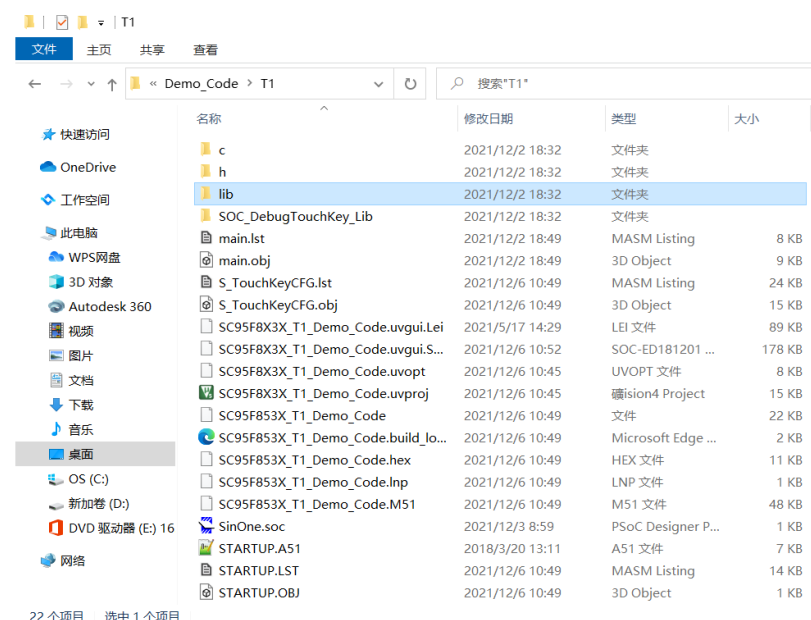
1) **TouchKeyScan(void)**函数返回值：

返回值对应 bit 为 1 即该通道有按键，0 为无按键。  
若使能双键，且有两键触发，则会有两个 bit 位置起。

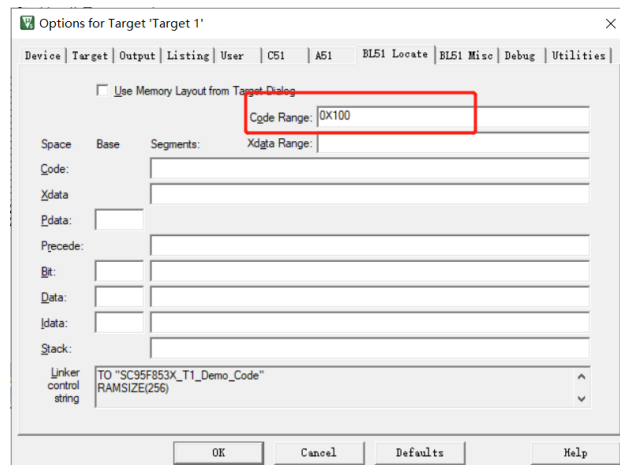
数据位		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
含义		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	触摸按键状态（1：有效；0：无效）							
数据位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
含义	TK23	TK22	TK21	TK20	TK19	TK18	TK17	TK16
	触摸按键状态（1：有效；0：无效）							
数据位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
含义	TK15	TK14	TK13	TK12	TK11	TK10	TK9	TK8
	触摸按键状态（1：有效；0：无效）							
数据位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
含义	TK7	TK6	TK5	TK4	TK3	TK2	TK1	TK0
	触摸按键状态（1：有效；0：无效）							

注：函数的返回类型是 **unsigned long int**；TKn 为触控通道，具体请参照对应规格书。

5. 打开工程项目文件复制“lib”文件夹至工程文件夹内



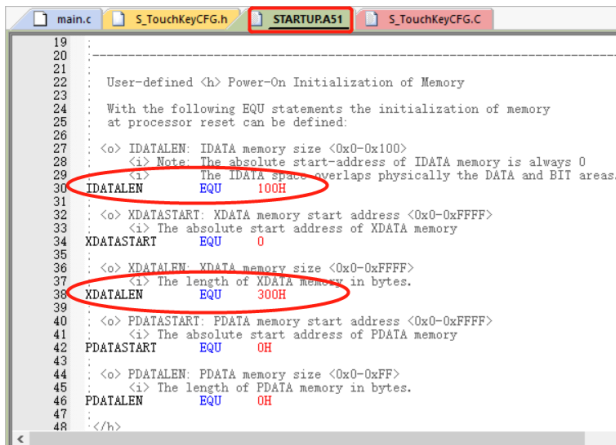
6. 在 Keil 中打开工程项目文件；注意设定（Code range）、设定（XDATALEN EQU xxxH）



设定的目的，参见赛元 MCU 应用注意事项 Vx.xx.PDF 文件。  
（部分芯片不用设置该项，请仔细阅读）



## 7. 设定 XDATALEN



```

19
20
21
22 User-defined <h> Power-On Initialization of Memory
23
24 With the following EQU statements the initialization of memory
25 at processor reset can be defined:
26
27 <o> IDATALEN: IDATA memory size <0x0-0x100>
28 <i> Note: The absolute start-address of IDATA memory is always 0
29 <i> The IDATA space overlaps physically the DATA and BIT areas.
30 IDATALEN EQU 100H
31
32 <o> XDATASTART: XDATA memory start address <0x0-0xFFFF>
33 <i> The absolute start address of XDATA memory
34 XDATASTART EQU 0
35
36 <o> XDATALEN: XDATA memory size <0x0-0xFFFF>
37 <i> The length of XDATA memory in bytes.
38 XDATALEN EQU 300H
39
40 <o> PDATASTART: PDATA memory start address <0x0-0xFFFF>
41 <i> The absolute start address of PDATA memory
42 PDATASTART EQU 0H
43
44 <o> PDATALEN: PDATA memory size <0x0-0xFF>
45 <i> The length of PDATA memory in bytes.
46 PDATALEN EQU 0H
47
48 </h>

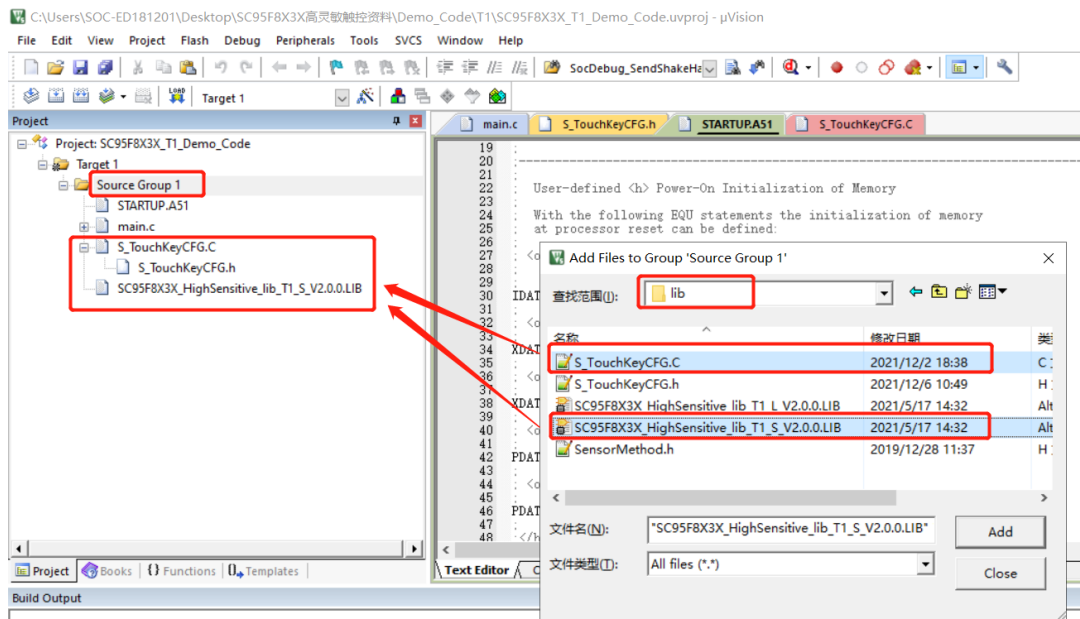
```

注意：用于 STARTUP.A51 中，清掉外部 XData，具体型号的 XData 大小见规格书。

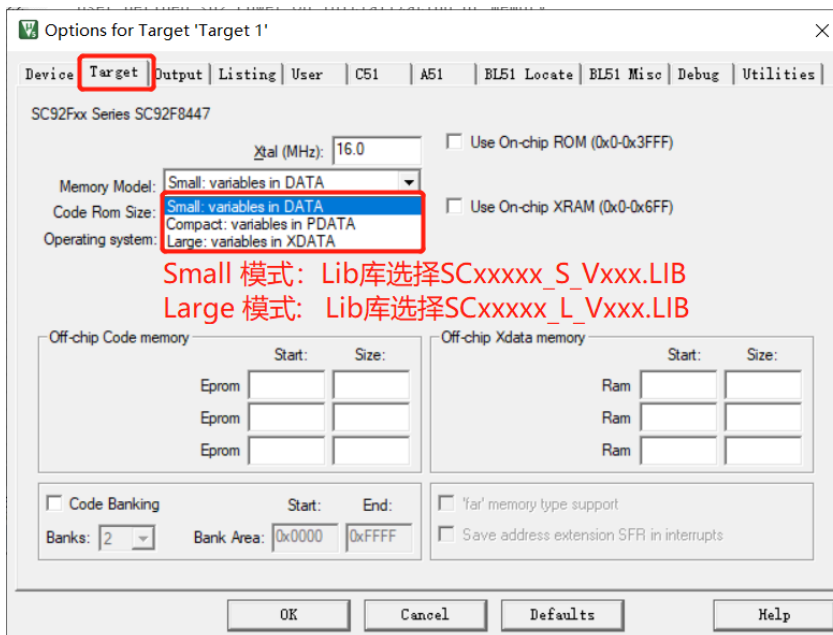
## 8. 在项目工程中添加库文件 LIB 以及 S\_TouchKeyCFG.C 文件

- 1) 从赛元提供的库体资料 Lib 文件夹内，添加库文件 LIB 以及 S\_TouchKeyCFG.C 至工程内。

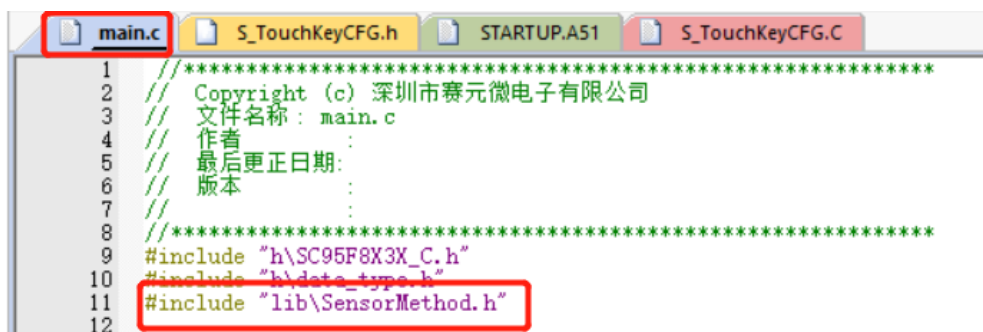
下图以 T1 库体为例：（T2 库操作一致）



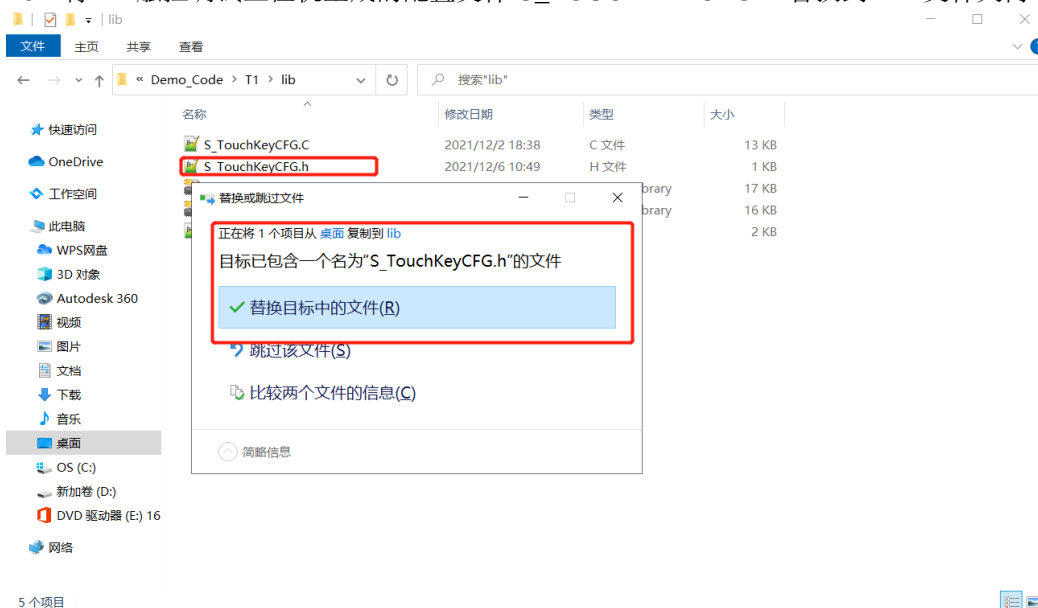
注意：库体选择 L 或者 S（大端编译或小端编译）请仔细区分，如下图所示：



#### 9. 在主程序文件中添加头文件引用



#### 10. 将 TK 触控调试上位机生成的配置文件 S\_TOUCHKEYCFG.H 替换到 LIB 文件夹内



到此，完整的赛元触控高灵敏软件库已添加至项目工程内。

**注意：**应用程序中需要将 TK 对应的 IO 口设置为强推挽输出高。

### 3.4 完成用户程序和赛元触控软件库的融合

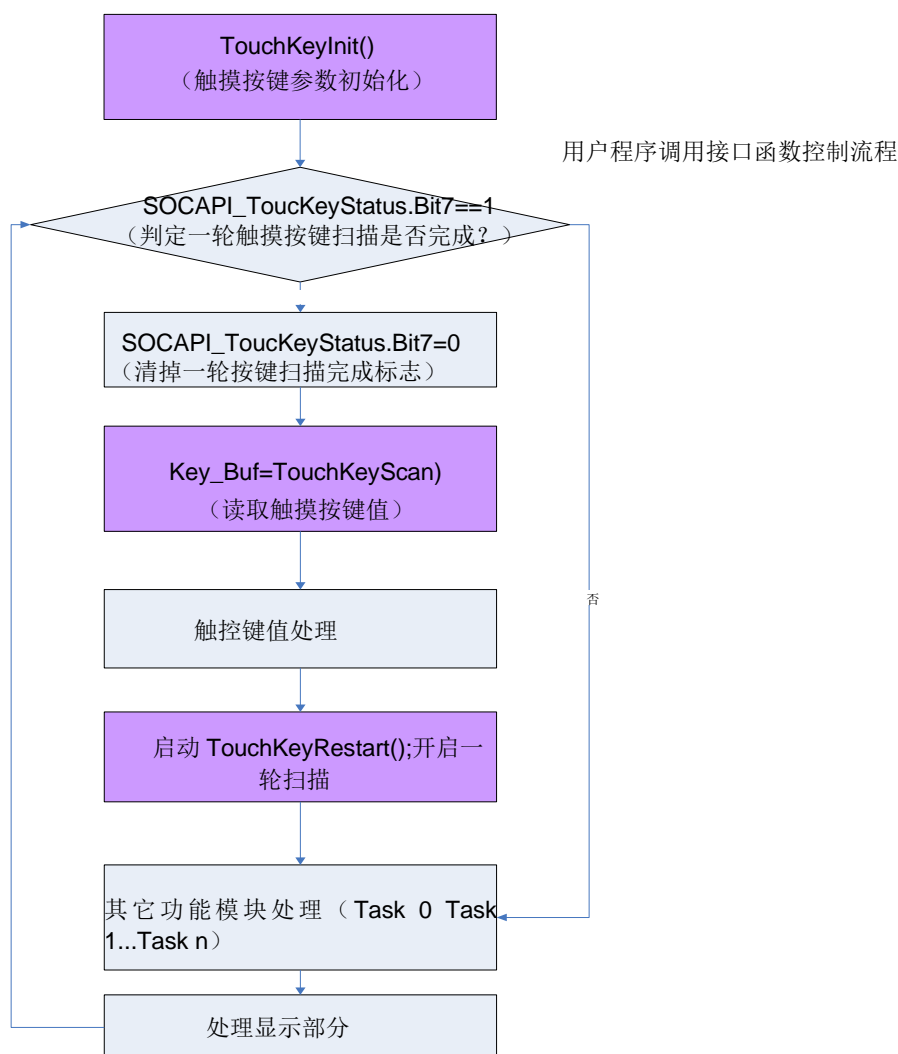
#### 3.4.1 高灵敏库触控软件 and 用户程序

- 1) 主程序和库文件的整体结构关系
  - ① 通过添加库文件至工程项目内，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摸按键的功能。
  - ② 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。
  - ③ 库函数只管触摸按键功能，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。
- 2) 库文件的调用流程（**弹簧和隔空库体调用流程有差异，请仔细阅读**）  
 用户通过一定的流程调用库文件的接口函数，便可得到触摸按键的键值。

##### 弹簧库文件调用流程（简称 T1 库）

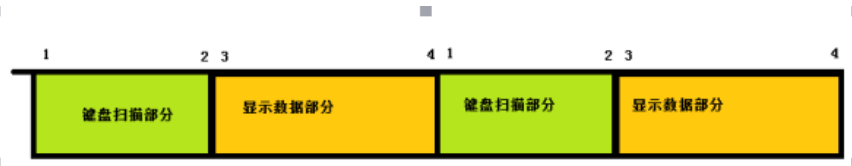
- ① 将 TK 对应的 IO 设置为强推挽输出高。
- ② 主程序调用接口函数 “TouchKeyInit()” 用于配置触摸按键通道的参数，并初始化 Baseline 基线；
- ③ 主程序通过查看全局变量 SOCAPI\_TouchKeyStatus&0x80 来判定一轮触摸按键扫描是否完成；
- ④ 主程序调用接口函数 “TouchKeyScan()” 用于读取触摸按键值；
- ⑤ 主程序调用 “TouchKeyRestart()” 启动下一轮扫描。

（下图中紫色的部分是库文件，其它部分是用户程序）

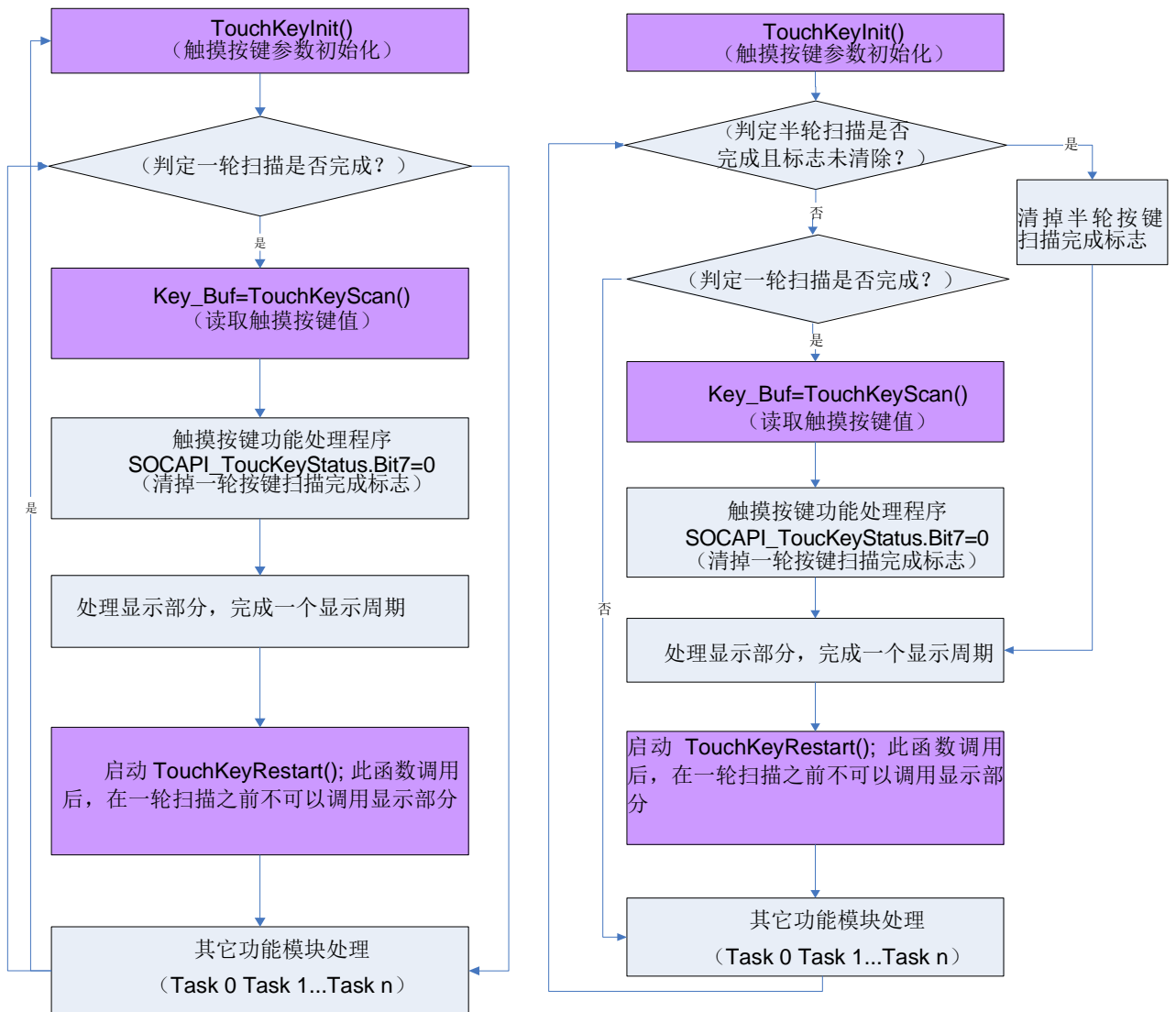


### 隔空库文件调用流程（简称 T2 库）

- 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()” 用于配置触摸按键通道的参数，并初始化 Baseline 基线；
- ③ 若按键个数大于 8 个，主程序通过查看全局变量 SOCAPI\_TouchKeyStatus&0x40 来判定半轮触控按键扫描是否完成；半轮按键扫描完成后跳转完成一个周期的显示后再完成后半轮按键的扫描。
- ④ 主程序通过查看全局变量 SOCAPI\_TouchKeyStatus&0x80 来判定一轮触摸按键扫描是否完成；
- ⑤ 主程序调用接口函数 “TouchKeyScan()” 用于读取触摸按键值；
- ⑥ 特别需要强调一点的是：调用 TouchKeyRestart() 开始扫描按键时后，一轮扫描或者半轮扫描的标志 还没有出现时，一定不要去 做显示数据的部分。



（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程（按键个数8个以下）

用户程序调用接口函数控制流程（按键个数大于 8 个）

### 3) 主程序和库文件的时序关系

因为运行触摸按键库消耗了部分 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，不能对触摸按键通道进行操作；  
如触摸按键通道为输出 IO；否则触摸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 触摸按键扫描计数值数据的动作，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率来调用库函数检测按键，以免错过按键动作；

#### 软件融合的注意事项：

##### ① 运行时间：

**TouchKeyInit(void):** 算法执行时间会因按键选择个数的增减而增减，200~500ms@12M；

**TouchKeyScan(void):** 执行该函数用时与不同芯片主频有关联,请参考 3.3 内容表格中数据

##### ② 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。

- ③ 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。

### 3.4.2 注意事项

- 1) 使用单面 PCB 板，一般用弹簧片来做触摸按键。因为其侧面也能同手指形成电场，使用弹簧片比使用 PCB 上覆铜做触摸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，推荐使用 103 电容，精度 10%。
- 6) TK 对应的 IO 口设置为强推挽输出高。

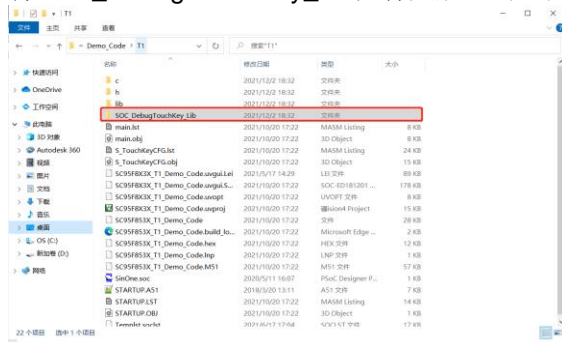
更多 Layout 注意事项请参考指南：《赛元触摸按键 MCU PCB 设计要点》。

## 3.5 附加功能-动态调试功能

主要功能：利用赛元触控调试上位机软件查看实时的数据情况，帮助用户对系统进行整体的评估，了解系统实际运行的情况，分析异常等。

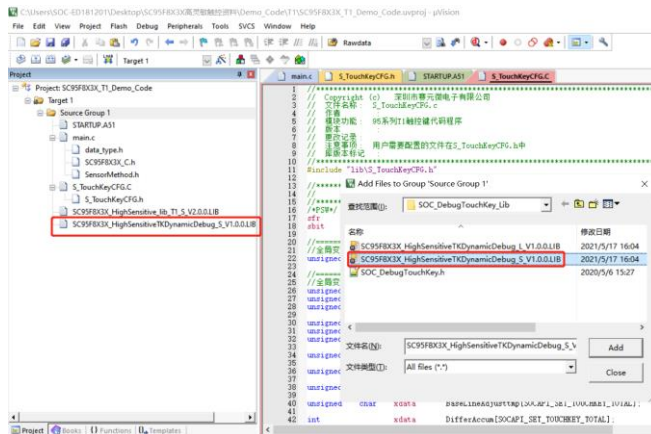
### 3.5.1 高灵敏度动态调试步骤

- 1) 将SOC\_DebugTouchKey\_Lib 文件夹放置到工程的根目录下





- 2) 在用户工程中加入 SC9XF8XXX\_HighSensitiveTKDynamicDebug\_S/L\_Vx.x.x.LIB  
S/L ->指的是编译动态调试库体 lib 是用小端/大端编译，需要和触控库体 S/L 保持一致,如图所示。



- 3) 在 main.c 中 include 头文件

```
#define __TOUCHKEY_DEBUG__ //打开调试数据，动态调试

#ifdef __TOUCHKEY_DEBUG__
#include "SOC_DebugTouchKey_Lib\SOC_DebugTouchKey.h"
#endif
```

- 4) 在 main 函数中调用 SOCAPI\_DeBugTouchKey\_Init 进行初始化。编译成功后烧录到芯片内

```
216 }
217 }
218 //函数名称: void main(void)
219 //函数功能: 主函数
220 //入口参数: void
221 //出口参数: void
222 //.....
223 void main(void)
224 {
225     Sys_Init();
226
227     #ifdef __TOUCHKEY_DEBUG__
228     SOCAPI_DeBugTouchKey_Init();
229     #endif
230
231     //触控按键初始化
232     TouchKeyInit();
233
234     while(1)
235     {
236         NOTCON = 0x10;
237         if(TimerFlag_1ms==1)
238         {
239             TimerFlag_1ms=0;
240             Sys_Scan();
241             BuzzerWork();
242         }
243     }
244 }
245 }
246 }
```

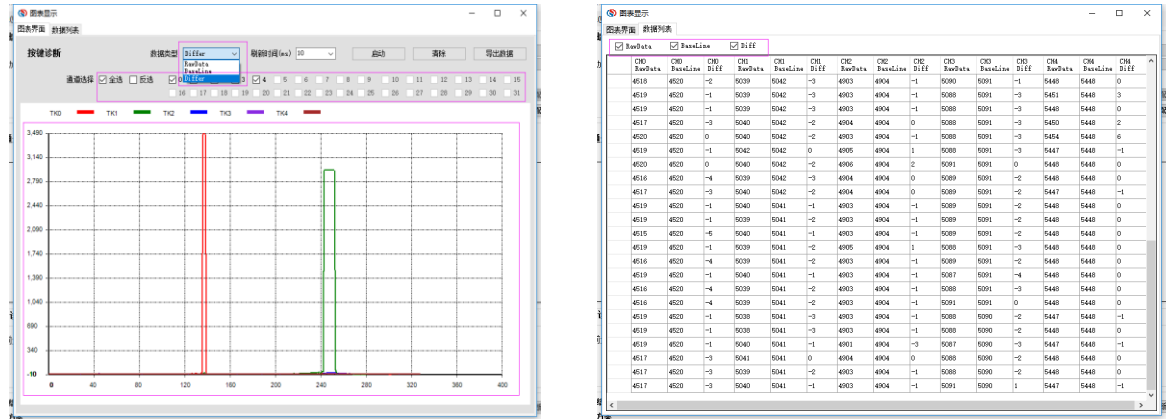
- 5) 打开 Touch Key Tool Menu,选择高灵敏度触控，芯片型号选择与实际芯片对应的型号，调试模式选择动态调试，勾选 TK 通道（必须与项目实际使用的通道一致）



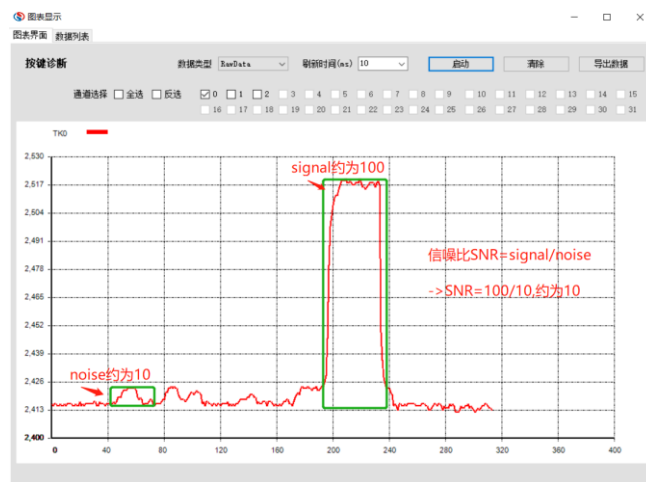
- 6) 点击确定按钮，然后点击下方“动态调试”按钮



- 7) 在动态调试界面内，可以通过选择“数据类型”查看想要查看的数据，通过“通道选择”勾选要查看的通道，在图表显示内可以实时看到数据的情况，点击“数据列表”可以查看图表数据



注意：动态调试观测数据时，需要注意信噪比 **SNR** 是否符合使用条件。  
建议是 **SNR > 5** 可用，推荐是 **SNR > 10** 效果较好。



**SNR 计算方式：动态调试观测中**

**Baseline:** 没有手按下时的 RawData 的平均值

**Noise** 噪声幅度为：静置状态下，没有手按下时 RawData 的最大值减去最小值的差值

**Finger:** 手指按下时 RawData 的平均值(图示绿色框内所示)

**Signal** 信号幅度为：Finger 值减去 Baseline 值

**SNR 计算方式：**  $SNR = \text{Signal}/\text{Noise} = (\text{Finger} - \text{Baseline})/\text{Noise}$ , 如上图计算  $SNR=10$  左右

- 8) 点击“导出数据”可以将实时采集数据导出CSV格式文档

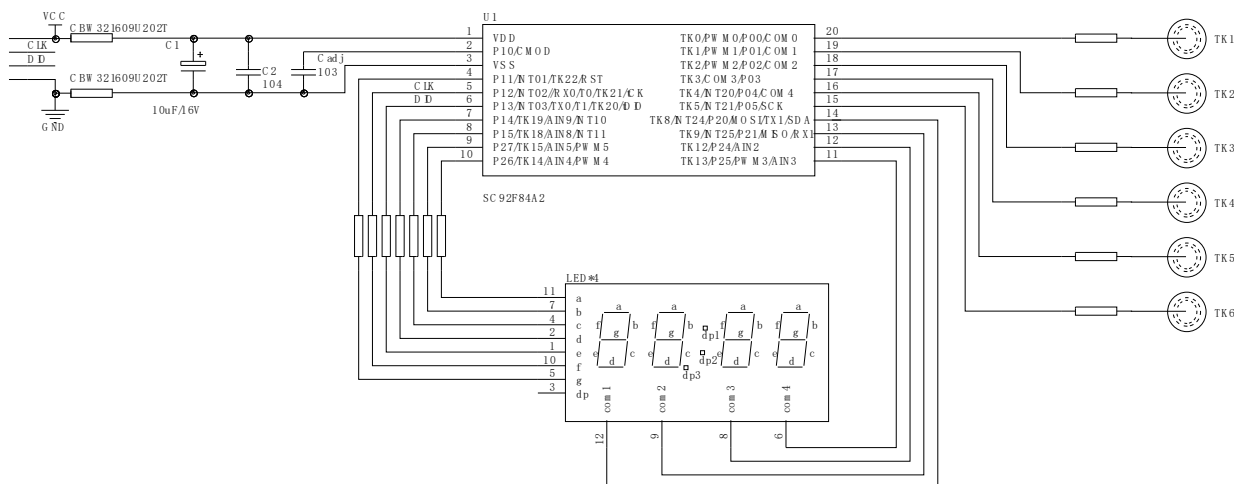
Touch_key_data.CSV - Excel(产品激活失败)																		
文件	开始	插入	页面布局	公式	数据	审阅	视图	加载测试	团队	告诉我想要做什么...								
A1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
CH0	CH0	CH0	CH1	CH1	CH1	CH2	CH2	CH2	CH3	CH3	CH3	CH4	CH4	CH4				
RawData	Baseline	Diff	RawData	Baseline	Diff	RawData	Baseline	Diff	RawData	Baseline	Diff	RawData	Baseline	Diff				
4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2				
4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2				
4427	4424	3	4954	4953	1	4817	4816	1	4991	4992	-1	5370	5372	-2				
4424	4424	0	4951	4953	-2	4815	4816	-1	4992	4992	0	5370	5372	-2				
4425	4424	1	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2				
4429	4424	5	4952	4953	-1	4816	4816	0	4992	4992	0	5369	5372	-3				
4424	4424	0	4951	4953	-2	4815	4816	-1	4991	4992	-1	5370	5372	-2				
4424	4424	0	4952	4953	-1	4816	4816	0	4992	4992	0	5371	5372	-1				
4424	4424	0	4955	4953	2	4818	4816	2	4993	4992	1	5376	5372	4				
4423	4424	-1	4952	4953	-1	4816	4816	0	4992	4992	0	5373	5372	1				
4424	4424	0	4953	4953	0	4816	4816	0	4991	4992	-1	5372	5372	0				
4424	4424	0	4952	4953	-1	4816	4816	0	4993	4992	1	5374	5372	2				
4426	4424	2	4952	4953	-1	4815	4816	-1	4992	4992	0	5374	5372	2				
4431	4424	7	4952	4953	-1	4815	4816	-1	4991	4992	-1	5374	5372	2				
4428	4424	4	4952	4953	-1	4815	4816	-1	4991	4992	-1	5373	5372	1				
4430	4424	6	4954	4953	1	4816	4816	0	4992	4992	0	5374	5372	2				
4428	4424	4	4957	4953	4	4817	4816	1	4993	4992	1	5371	5372	-1				
4424	4424	0	4955	4953	2	4815	4816	-1	4996	4992	4	5375	5372	3				
4423	4424	-1	4952	4953	-1	4816	4816	0	4992	4992	0	5375	5372	3				
4422	4424	-2	4952	4953	-1	4815	4816	-1	4991	4992	-1	5375	5372	3				
4422	4424	-2	4957	4953	4	4816	4816	0	4992	4992	0	5375	5372	3				
4425	4424	1	4956	4953	3	4817	4816	1	4992	4992	0	5375	5372	3				
4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5375	5372	3				
4425	4424	1	4952	4952	0	4816	4816	0	4992	4992	0	5376	5372	4				
4429	4424	5	4953	4952	1	4818	4816	2	4992	4992	0	5375	5372	3				
4422	4424	-2	4955	4952	3	4814	4816	-2	4993	4992	1	5377	5372	5				

## 9) 动态调试注意事项

- 由于动态调试库使用了烧录口上的 **UART** (**UART0** 或者 **SSI**) 资源, 用户程序必须先屏蔽掉 **UART** (**UART0** 或者 **SSI**) 部分程序, 包括初始化、中断服务函数等, 用户程序不能操作和 **UART** (**UART0** 或者 **SSI**) 相关的寄存器以及操作对应的管脚, 其中烧录口上为 **UART0** 的型号 还使用了 **Timer2** 作为波特率发生器, 所以 **Timer2** 也不能使用。
- 调试主界面勾选的通道必须与实际工程使用的通道一致。
- 动态调试库占用了 **43byte idata** 和 **501byte ROM** 资源, 请预留足够资源, 保证动态调试程序运行正常。

## 附录

### 一、应用参考原理图



### 二、软件参考示例

- 1) 对于 TouchKey 与 LED 共用的项目，调用 TouchKeyRestart() 开始扫描按键 时，SOCAPI\_TouchKeyStatus & 0X80 的标志还没有出现时，一定不要去做显示数据的事情；
- 2) 对于 TouchKey 与 LED 不共用的项目，则显示和扫描按键没有必要分开去做。

下面附程序说明

TouchKey 与 LED 共用的项目：

Main.c

void main()

{

```
    unsigned char result =0;
    SegOutState;    // 初始化 IO 口， 显示的 SEG,COM 脚
    ComOutState; ComAllClose; SegAllClose;
    TestIOPortOut; //P17 作为测试 IO 口
```

```
    EA = 1; //开总中断
```

```
    TouchKeyInit(); //重要步骤 1: 扫键的初始化函数
    InitialLcd();   //初始化显示部分
```

```
    while(1)
```

```
    {
```

```
        WDTCON |= 0x10;    //清 watchdog
```

```
        //重要步骤 2: 触摸键扫描一轮标志，是否调用 TouchKeyScan()一定要根据此标志位置起后
        if(SOCAPI_TouchKeyStatus & 0X80)
```

```
        {
```

```
            //重要步骤 3: 清除标//志位， 需要外部清除。
```

```
            SOCAPI_TouchKeyStatus &= 0X7F;
```

```
            exKeyValue = TouchKeyScan(); //重要步骤 4: 分析按键数据， 并返回结果出来
```

```
            /// 如果有按键，则更新显示缓冲区数据
```

```
            {
```

```
                UpdateLcdBufFunc(); //更新显示数据
```

```
            ///如果没有显示，直接对 IO 口操作作用示波器看结果
```

```
            TESTIO=~TESTIO;
```

```
            }
```

//重要步骤 4: bSensorCycleDone 标志位置起后,内部会停止检测按键,此时留出时间片显示数据

```
{  
    DisplayData(); //扫键完成后,立即启动显示  
    OpenPwm();    //启动显示用的 PWM  
}
```

```
}
```

```
}
```

```
}
```

Display.c

void DisplayData(void)

```
{
```

```
    ComAllClose; SegAllClose;
```

```
    if(isLcdComflag == 0) //显示 COM0 数据
```

```
    {
```

```
        SetSegData(glsLcdDataBuf[0]);  
        seg8 = glsLcdDataBuf[4] & 0x01;  
        seg9 = glsLcdDataBuf[4] & 0x02; COM0 = 0;
```

```
        isLcdComflag = 1;
```

```
    }
```

```
    else if(isLcdComflag ==1) //显示 COM1 数据
```

```
    {
```

```
        SetSegData(glsLcdDataBuf[1]);  
        seg8 = glsLcdDataBuf[5] & 0x01;  
        seg9 = glsLcdDataBuf[5] & 0x02;  
        COM1 = 0;
```

```
        isLcdComflag = 2;
```

```
    }
```

```
    else if(isLcdComflag ==2) //显示 COM2 数据
```

```
    {
```

```
        SetSegData(glsLcdDataBuf[2]);  
        seg8 = glsLcdDataBuf[6] & 0x01;  
        seg9 = glsLcdDataBuf[6] & 0x02;  
        COM2 = 0;  
        isLcdComflag = 3;
```

```
    }
```

```
    else if(isLcdComflag ==3) //显示 COM3 数据
```

```
    {
```

```
        SetSegData(glsLcdDataBuf[3]);  
        seg8 = glsLcdDataBuf[7] & 0x01;  
        seg9 = glsLcdDataBuf[7] & 0x02;  
        COM3 = 0;  
        isLcdComflag = 4;
```

```
    }
```

```
    else
```

```
    {
```

```
        if(isLcdComflag == 4) //COM 都显示完毕,准备启动按键扫描
```

```
        {
```

```
            ClosePwm(); //关闭显示用到的 PWM。
```



```
isLcdComflag = 0;

//重要步骤 5: 待所有的显示完毕后, 需要重新调用 TouchKeyRestart(); 启动按键扫描,
//否则//不会扫描按键, 同时需要关闭与 TK 共用的一些 IO 口, 保持检测按键的一致性。
ComAllClose;
SegAllClose;
TouchKeyRestart();
    }
}
}
```

### TouchKey 与 LED 不共用的项目

#### Main.c

```
void main()
{
    unsigned char result =0;
    SegOutState; // 初始化 IO 口, 显示的 SEG,COM 脚
    ComOutState; ComAllClose; SegAllClose;
    TestIOPortOut; //P17 作为测试 IO 口

    EA = 1; //开总中断

    TouchKeyInit(); //重要步骤 1: 扫描的初始化函数
    InitialLcd(); //初始化显示部分

    while(1)
    {
        WDTCN |= 0x10; //清 watchdog if(TimerFlag_1ms==1)
        {
            TimerFlag_1ms=0;
            if(SOCAPI_TouchKeyStatus&0x80) //重要步骤 2: 触摸键扫描一轮标志, 是否调用
                TouchKeyScan()一定要 根据此标志位置起后
            {
                SOCAPI_TouchKeyStatus &=0x7f; //重要步骤 3: 清除标志位, 需要外部清除。
                exKeyValueFlag = TouchKeyScan();
                ChangeTouchKeyvalue();
                UpdateLcdBufFunc(); //更新显示数据
                TouchKeyRestart(); //启动下一轮转换 TimerFlag_1ms=0;
            }
            BuzzerWork();
            //*****蜂鸣器驱动函数*****
            if(++Timercount>=10)
            {
                Timercount = 0;
                DataUpdateCount++;
            }
            UpdateDisplay(); //*****处理显示内容*****
        }
    }
}

void DisplayData(void)
{
    ComAllClose;
    if(isLcdComflag == 0) //显示 COM0 数据
```

```
{
    LedSetSegData(LcdDisplayBuf[glsLedDataBuf[0]]); COM0 = 0;
    isLcdComflag = 1;
}
else if(isLcdComflag ==1)      //显示 COM1 数据
{
    LedSetSegData(LcdDisplayBuf[glsLedDataBuf[5]]); COM1 = 0;
    isLcdComflag = 2;
}
else if(isLcdComflag ==2)      //显示 COM2 数据
{
    LedSetSegData(LcdDisplayBuf[glsLedDataBuf[1]]); COM2 = 0;
    isLcdComflag = 3;
}
else if(isLcdComflag ==3)      //显示 COM3 数据
{
    LedSetSegData(glsLedDataBuf[3]); COM3 = 0;
    isLcdComflag = 4;
}
else if(isLcdComflag ==4)      //显示 COM4 数据
{
    LedSetSegData(glsLedDataBuf[4]); COM4 = 0;
    isLcdComflag = 5;
}
else if(isLcdComflag ==5)      //显示 COM5 数据
{
    LedSetSegData(LcdDisplayBuf[glsLedDataBuf[2]]); COM5 = 0;
    isLcdComflag = 6;
}
else if(isLcdComflag ==6)
{
    isLcdComflag = 0; ComAllClose;
}
}
```

## 4 规格更改记录

版本	记录	日期
V1.6	1. 删除系列区分，统一版本 2. 完善高灵敏度触控描述，去除高可靠模式描述	2022 年 10 月
V1.5	增加打开触控调试上位机调试参数前，需烧录静态调试码的提示	2022 年 7 月
V1.4	文档格式规范化 修改高灵敏度信噪比描述	2021 年 11 月

## 声明

深圳市赛元微电子股份有限公司（以下简称赛元）保留随时对赛元产品、文档或服务进行变更、更正、增强、修改和改进的权利，恕不另行通知。赛元认为提供的信息是准确可信的。本文档信息于 2021 年 11 月开始使用。在实际进行生产设计时，请参阅各产品最新的数据手册等相关资料。