目录

目录.....1

1 赛元 SC92F_93F 系列 TOUCHKEY MCU 应用指南总体描述.....3

2 SC92F8XXX_SC93F8XXX_HIGHSENSITIVE_LIB_T1 库说明.....4

2.1 前言.....4
2.1.1 赛元高灵敏度触控库介绍.....4
2.1.2 赛元高灵敏度触控库文件介绍.....4

2.2 调试流程.....5
2.2.1 安装开发工具并配置参数、导出数据.....5
2.2.2 实现赛元高灵敏度触控软件库的功能测试.....21
2.2.3 完成用户程序和赛元触控软件库的融合.....27
2.2.4 库使用注意事项.....29

附录.....30
一、应用参考原理图（以 SC93F8433 为例）.....30
二、程序调用例程.....31

3 SC92F8XXX_SC93F8XXX_HIGHSENSITIVE_LIB_T2 库说明.....33

3.1 前言.....33
3.1.1 赛元高灵敏度触控库介绍.....33
3.1.2 赛元高灵敏度触控库文件介绍.....33

3.2 调试流程.....34
3.2.1 安装开发工具并配置参数、导出数据.....34
3.2.2 实现赛元高灵敏度触控软件库的功能测试.....50
3.2.3 完成用户程序和赛元触控软件库的融合.....56
3.2.4 库使用注意事项.....59

附录.....60
一、应用参考原理图.....60
二、程序调用例程.....61

4 SC92F8XXX_HIGHRELIABILITY_LIB_T1 库说明.....64

4.1 前言.....64
4.1.1 赛元触控库文件介绍.....64
4.1.2 赛元触控原理介绍.....64

4.2 调试流程概括提纲.....65
4.2.1 安装开发工具并配置参数、导出数据.....65
4.2.2 实现赛元软件库的功能测试.....66
4.2.3 完成用户程序和赛元触控软件库的融合.....66

4.3 赛元触控软件库的详细使用说明.....67

4.3.1 安装开发工具并配置参数、导出数据 67
4.3.2 实现赛元软件库的功能测试 72
4.3.3 完成用户程序和赛元触控软件库的融合 80

附录 85
一、应用参考原理图 85
二、库使用注意事项 86

5 更改记录 90

1 赛元 SC92F_93F 系列 TOUCHKEY MCU 应用指南总体描述

本文档是赛元 SC92F_93F 系列 Touchkey MCU 触控的应用指南。赛元触控 MCU 的触控架构分为高灵敏度触控模式和高可靠触控模式，部分型号内建双模触控(具体参见规格书描述)，可通过选择不同的触控库文件来使用高灵敏度模式或高可靠模式，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、高可靠模式具有很强的抗干扰能力，可通过 10V 动态 CS 测试
3、最多可实现 31 路触控按键及衍生功能
4、高灵活度开发软件库支持，低开发难度
5、自动化调试软件支持，智能化开发
6、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 11uA

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控功能。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	高可靠模式
特点	① 高抗干扰能力，可通过 3V 动态 CS ② 超高灵敏度	① 超强抗干扰能力，可通过 10V 动态 CS ② 功耗更低
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	① 要求具有超强抗干扰性的应用 ② 有 10V 动态 CS 要求的应用
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	通过项目工程载入高可靠的触控库来选择高可靠模式
对应说明	2 SC92F8XXX_SC93F8XXX_HighSensitive_Lib_T1 库说明 3 SC92F8XXX_SC93F8XXX_HighSensitive_Lib_T2 库说明	4 SC92F8XXX_HighReliability_LIB_T1 库说明
对应的库文件	“SC92F8XXX_HighSensitive_Lib_Tn_Vx.x.x.LIB”	“SC92F8XXX_HighReliability_Lib_Tn_Vx.x.x.LIB
注意事项	① T1 库应用于弹簧类型的应用 ② T2 库应用于隔空类型的应用，且按键个数至少 3 个以上	只可应用于弹簧类型的应用
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。	只有两种情况下建议使用高可靠模式： ① 需要通过 10V 动态 CS ② 需要更低的低功耗电流，且高灵敏度模式下电流无法充满

2 SC92F8XXX_SC93F8XXX_HIGHSENSITIVE_LIB_T1 库说明

2.1 前言

2.1.1 赛元高灵敏度触控库介绍

赛元 Ftouch MCU SC92F8XXX/SC93F8XXX 提供一个可以供用户调用的库文件，以降低用户触控按键部分的开发难度。本触控库是 SC92F8XXX/SC93F8XXX 弹簧触摸按键库，支持单按键和组合按键功能。**应用范围：弹簧按键**。

用户仅仅需要经过以下几个步骤，便可实现触控按键的功能，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。

SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T1_Vx.x.x 的使用分以下几个步骤：

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按键电脑界面软件 SOC HighSensitive TouchKey Tool，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 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 最合适的触控按键关键参数，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。

用户自行写好除触控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。

2.1.2 赛元高灵敏度触控库文件介绍

赛元高灵敏度触控按键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。

SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T1_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 HighSensitive TouchKey Tool 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无需修改。

2.2 调试流程

2.2.1 安装开发工具并配置参数、导出数据

1、安装调试工具及连接硬件；

① Setup Pro51；

安装赛元 Pro51 软件 SOC Pro51 Vx.xx.exe(请从赛元网站找最新版本)。

② Setup SOC TouchKey Tool；

安装赛元触控调试软件 SOC TouchKey Tool (请从赛元网站找最新版本)。

③ 升级 DPT52/SC-LINK 固件,更新 MCU 库；

在线烧写器 DPT52/SC-LINK 的固件和 SOC Pro51 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 的插件安装文件版本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 目录内的 SinOne_Chip 目录中。

b. SinOne_Chip 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

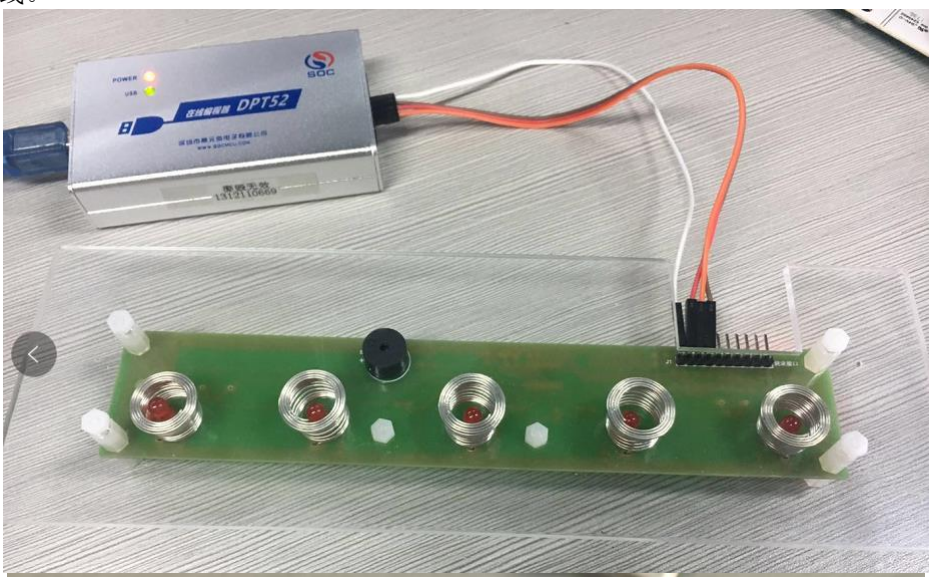
PDF：赛元在线开发工具 DPT52 使用说明

SOC_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：电脑 USB-->DPT52/SC-LINK(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 DPT52 的接线。





图为 DPT52 的接口

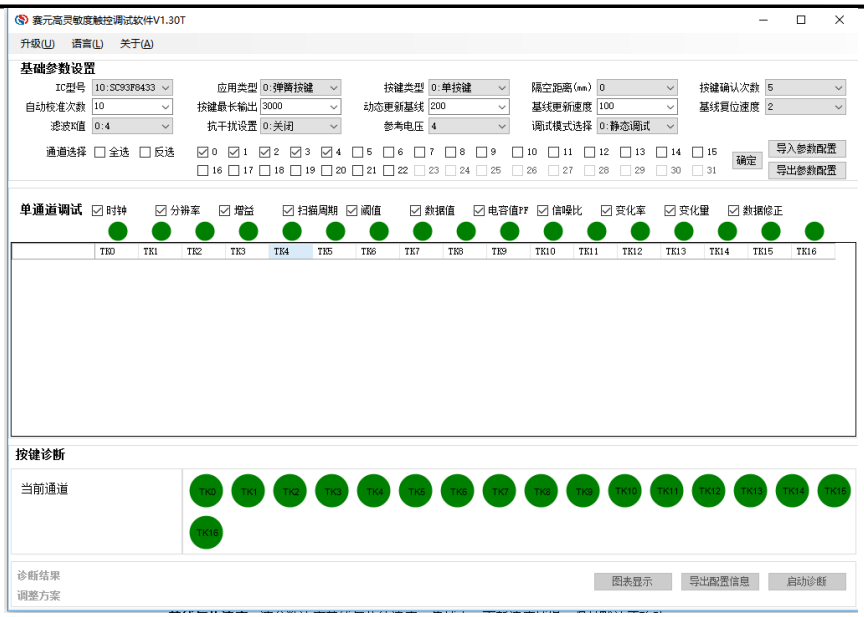
⑥ 烧录 SC92F8XXX/SC93F8XXX_HighSensitiveTKStaticDebug_Vx.x.x.hex 到用户 PCB 上的 SC92F8XXX/SC93F8XXX IC 中；打开 SOC Pro51 软件，选择项目使用的 MCU 型号，载入“SC92F8XXX/SC93F8XXX_HighSensitiveTKStaticDebug_Vx.xx.hex”文件，点击“编程”，完成后关闭 SOC Pro51 软件，重新拔插 USB 上电。(**注意:LVR 设置必须比供电电压低,如供电为 3.3V,则 Option 中 LVR 必须选择 3.3V 以下的档位**)见以下图：



⑦ 将接线接为触摸调试模式，硬件的连接方法为：电脑 USB-->DPT52/SC-LINK(VCC/GND/CLK/DIO) -->用户 PCB(VCC/GND/tCK/tDIO)；并测试连接正常；

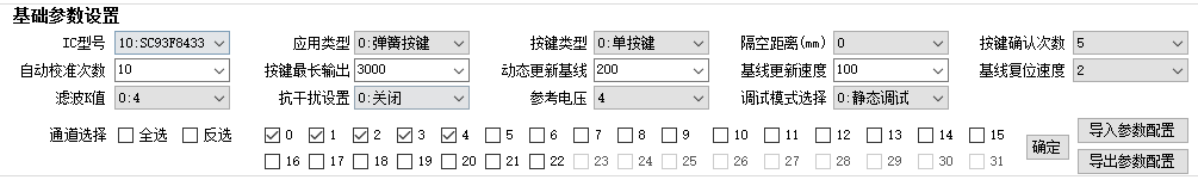
2、调试触控参数

① 打开 Touch Key Tool Menu,选择高灵敏度触控



② 参数配置，进入触控调试

a. 选择项目使用的 Ftouch MCU 以及勾选使用的 TK 通道，如图所示：



b. 设置应用的基本信息如下：

应用类型：选择弹簧按键

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：选择 0

c. 配置触控算法运算的相关参数

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*12MS=60MS。

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，保持默认不改动

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。

参考电压：保持默认不改动

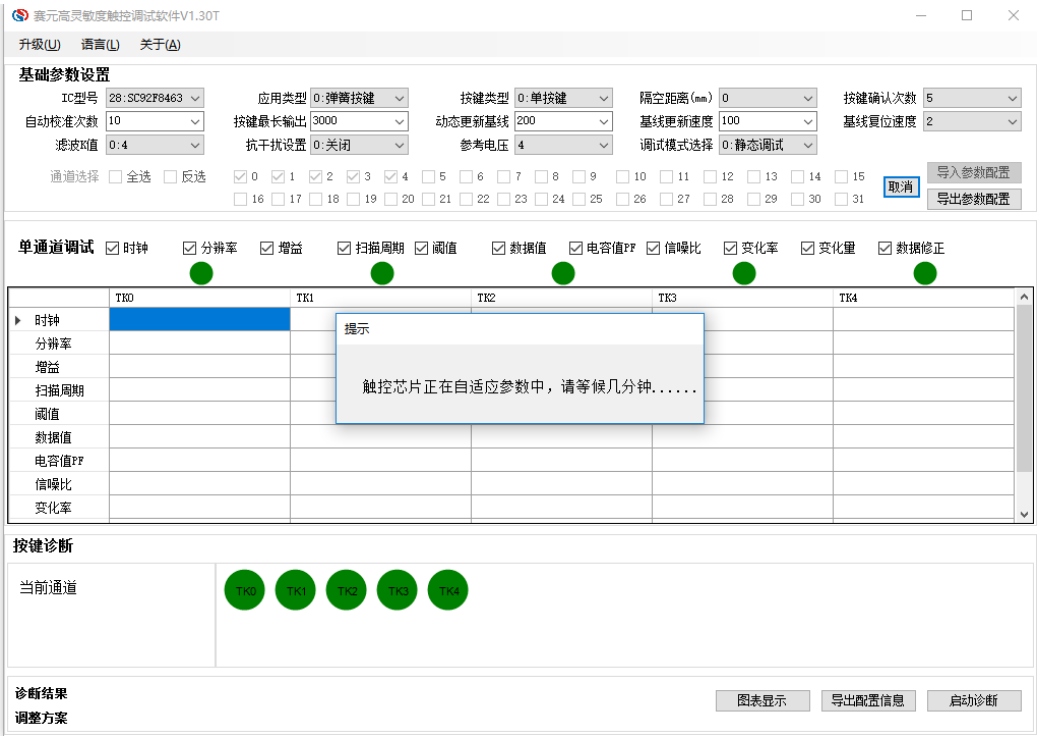
调试模式选择：静态调试为确定触控参数，动态调试为应用中采集数据，这里选择静态调试，后续章节会介绍动态调试。

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。

③ 触摸按键参数自适应

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。**



④ 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：



b. 设置触控相关参数



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。

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：

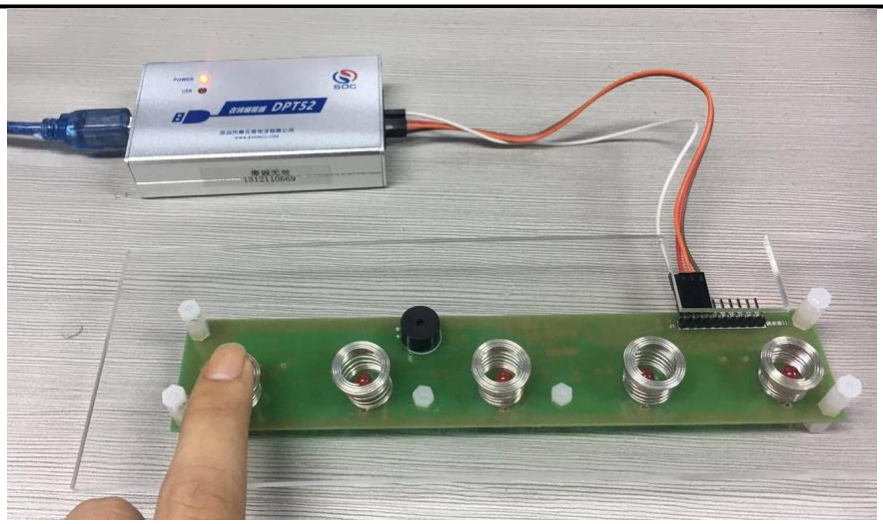
调试分两个过程：无触摸过程以及触摸过程。请按照界面的提示相应进行操作。该过程大约需要 15 秒。

不触摸过程：



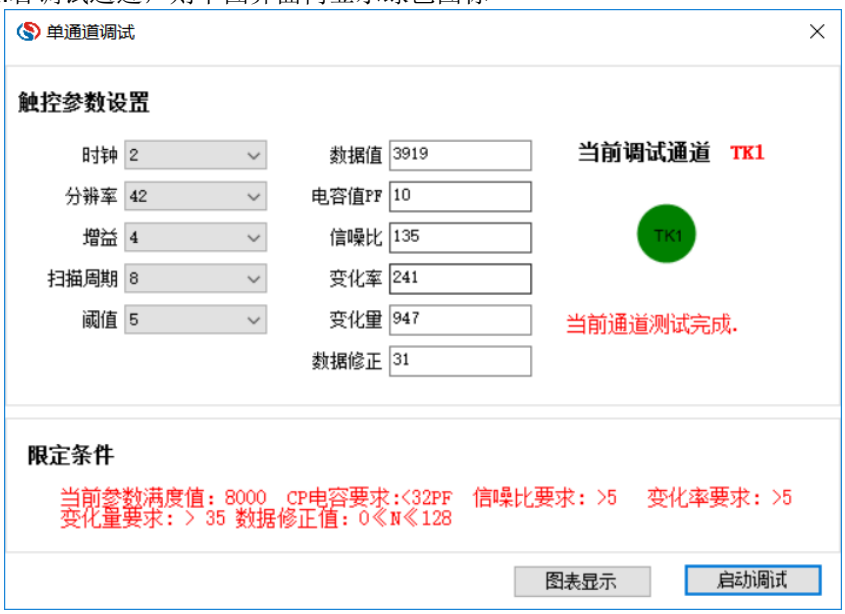
触摸过程：





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

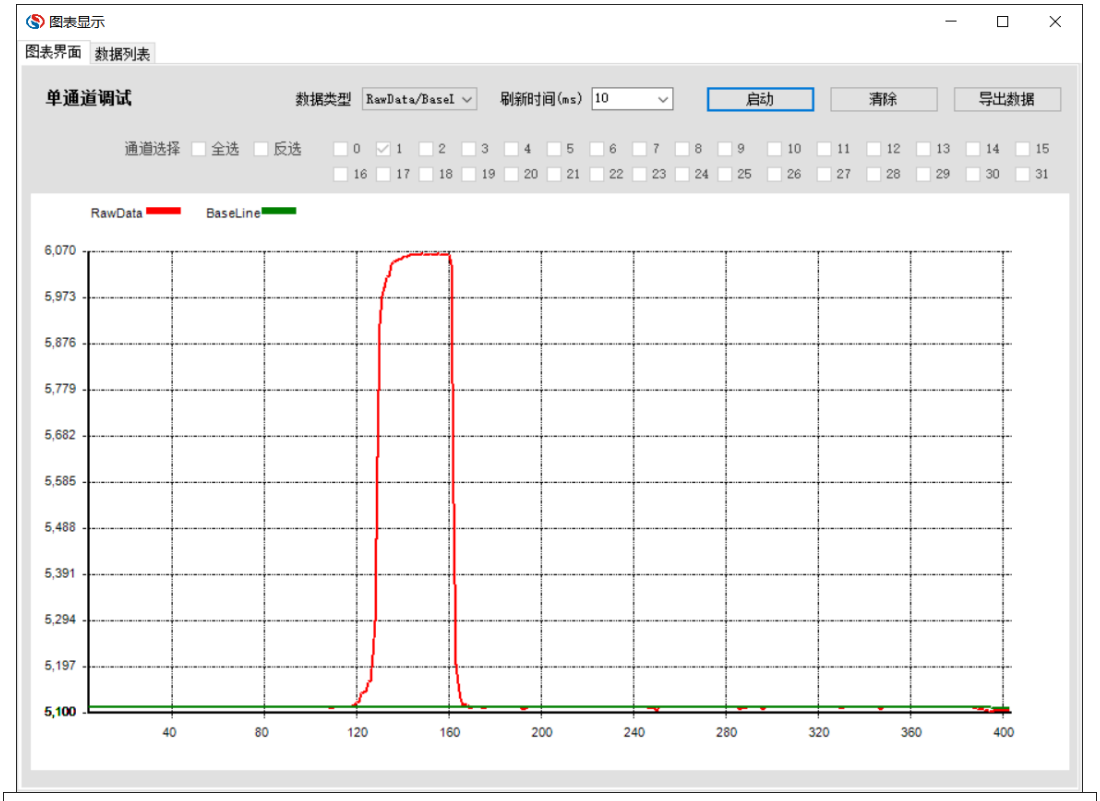


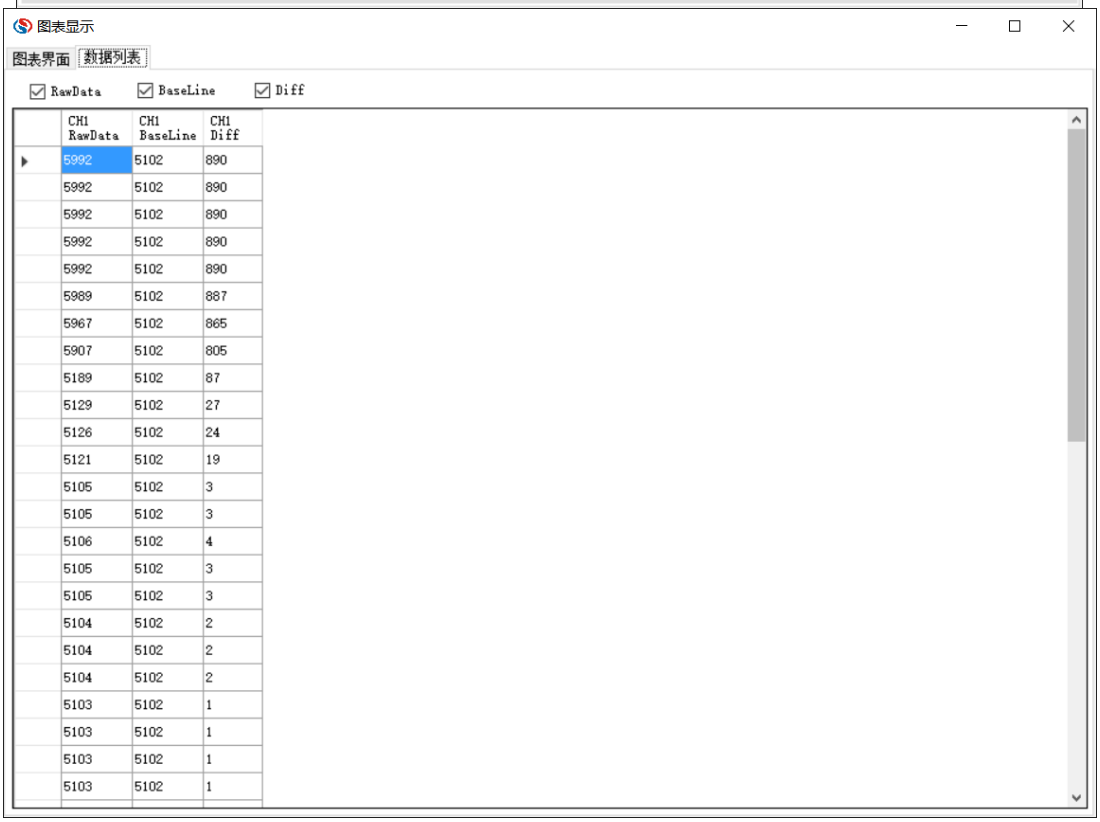
若调试不通过，则显示红色图标。



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

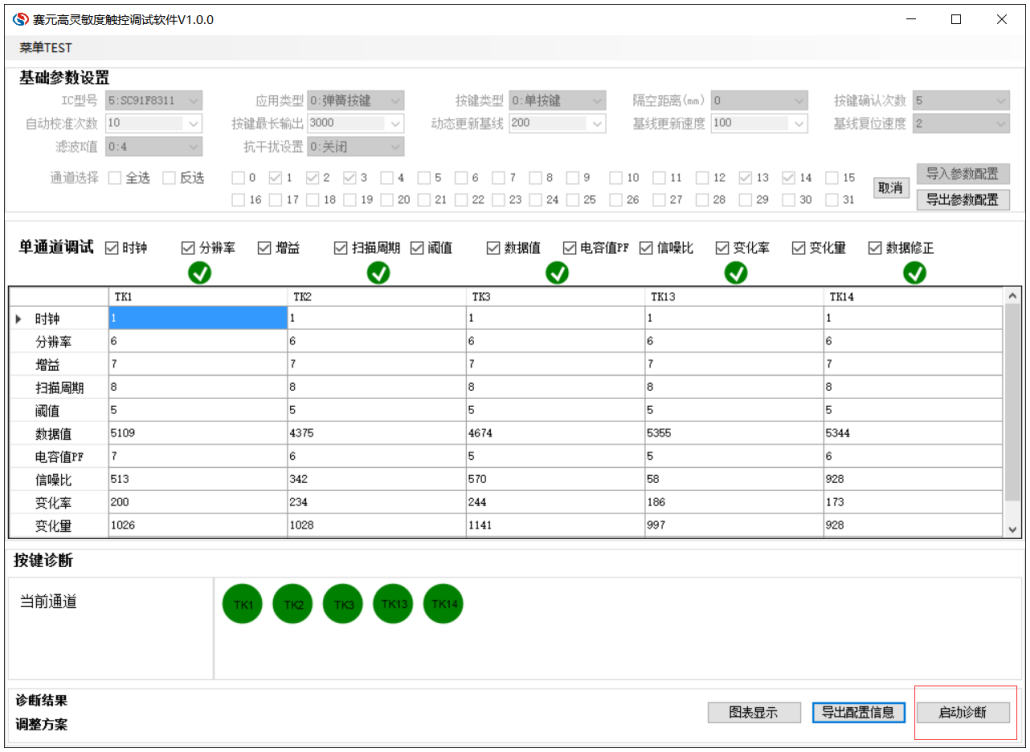




CH1 RawData	CH1 BaseLine	CH1 Diff
5992	5102	890
5992	5102	890
5992	5102	890
5992	5102	890
5992	5102	890
5989	5102	887
5967	5102	865
5907	5102	805
5189	5102	87
5129	5102	27
5126	5102	24
5121	5102	19
5105	5102	3
5105	5102	3
5106	5102	4
5105	5102	3
5105	5102	3
5104	5102	2
5104	5102	2
5104	5102	2
5103	5102	1
5103	5102	1
5103	5102	1
5103	5102	1

依次调试每个按键，调至按键均通过。

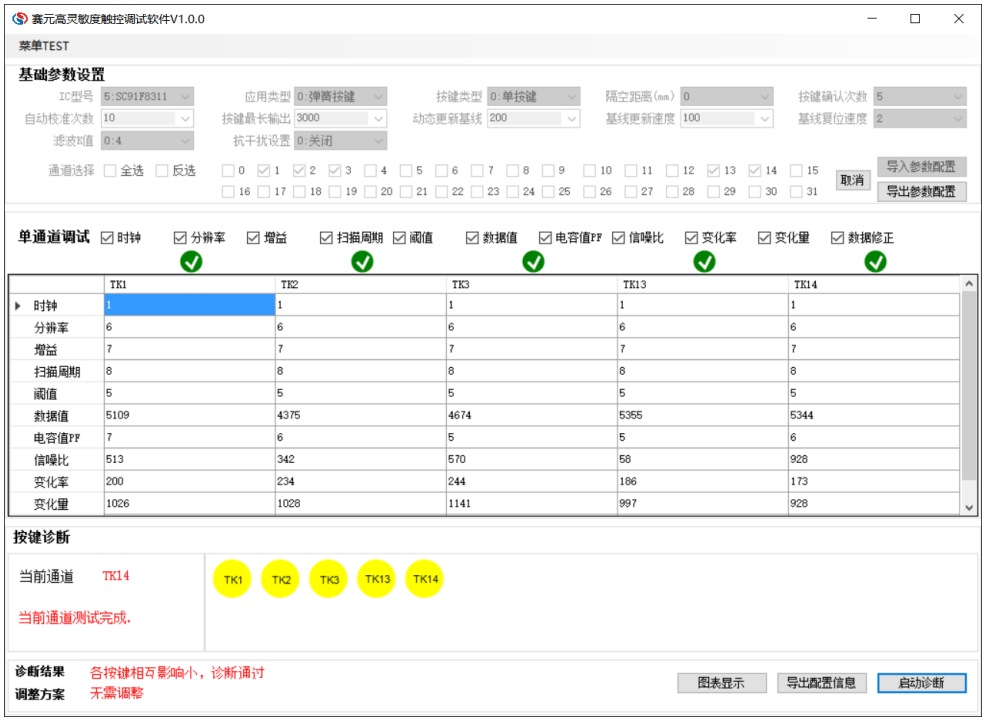
⑤ 进行按键诊断。按键诊断是分析按键间的相互影响的过程。若按键间的相互影响比较大，会影响到按键的性能。点击“启动诊断“按钮。



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！



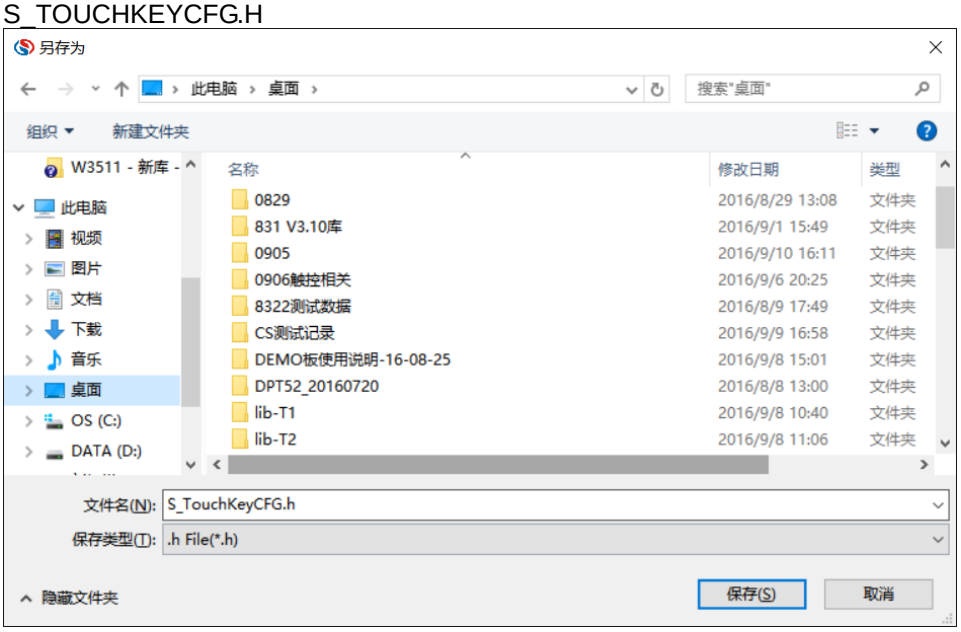


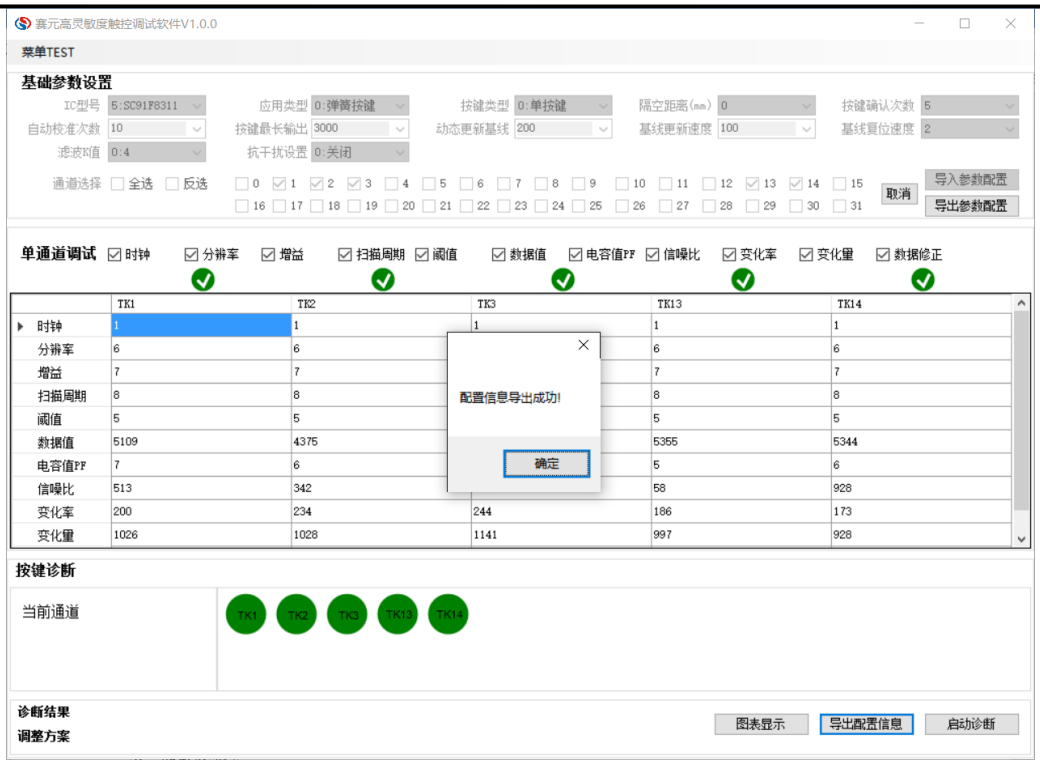


若诊断不通过，请根据诊断结果和调整方案，调整硬件 Layout。如下图是诊断不通过的提示语：

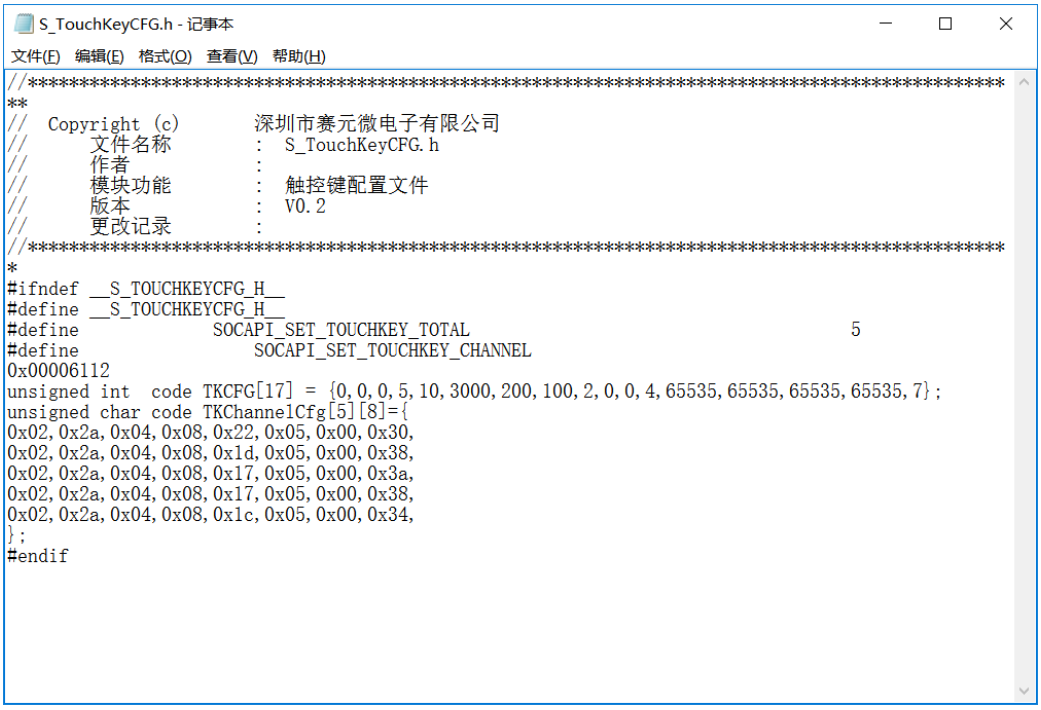


⑥ 完成按键诊断并且测试通过后，点击“导出配置信息”按钮生成配置文件 S_TOUCHKEYCFG.H





S_TOUCHKEYCFG.H 内容如下：



```
S_TouchKeyCFG.h - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
//*********************************************************************************************************
**
//  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                                   5
#define                 SOCAPI_SET_TOUCHKEY_CHANNEL
0x00006112
unsigned int  code TKCFG[17] = {0,0,0,5,10,3000,200,100,2,0,0,4,65535,65535,65535,65535,7};
unsigned char code TKChannelCfg[5][8]={
0x02,0x2a,0x04,0x08,0x22,0x05,0x00,0x30,
0x02,0x2a,0x04,0x08,0x1d,0x05,0x00,0x38,
0x02,0x2a,0x04,0x08,0x17,0x05,0x00,0x3a,
0x02,0x2a,0x04,0x08,0x17,0x05,0x00,0x38,
0x02,0x2a,0x04,0x08,0x1c,0x05,0x00,0x34,
};
#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 65535 不改动
TKCFG[12]		保持默认 65535 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

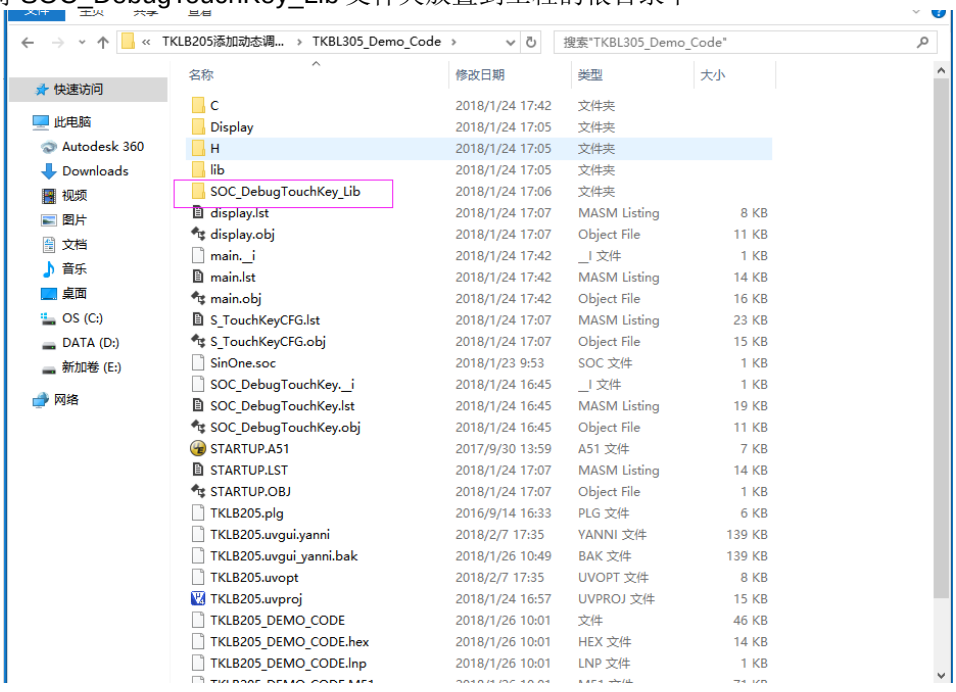
至此触控按键的调试过程结束。

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。

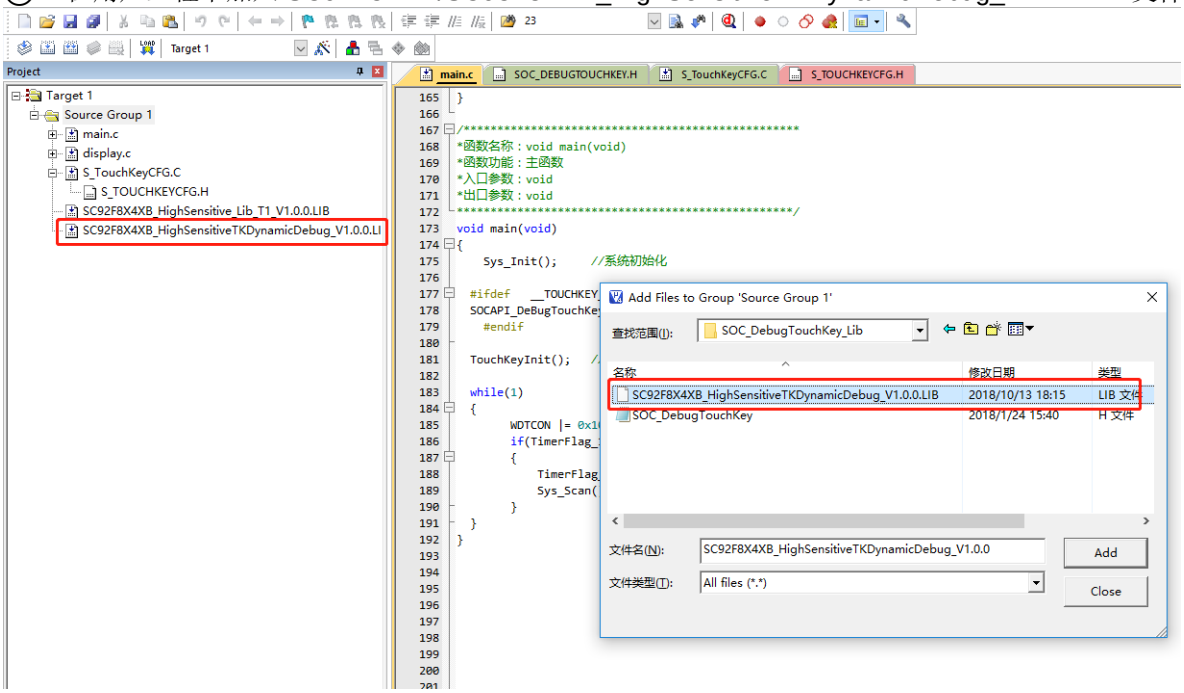
3、加载动态调试库进行动态调试数据

赛元触控动态调试库的功能是可以在用户程序中加入调试代码，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。

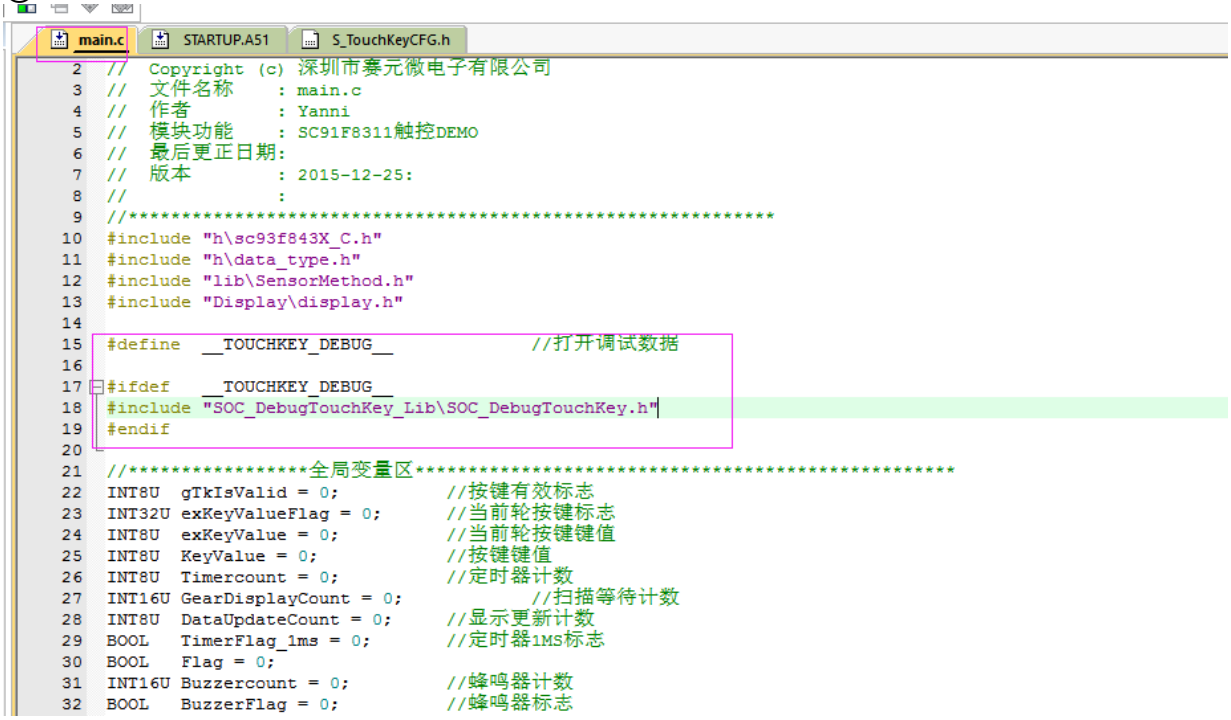
① 将 SOC_DebugTouchKey_Lib 文件夹放置到工程的根目录下



② 在用户工程中加入 SC92F8XXX/SC93F8XXX_HighSensitiveTKDynamicDebug_Vx.x.x.LIB 文件



③ 在 main.c 中 include 头文件



④ 在 main 函数中调用 SOCAPI_DeBugTouchKey_Init 进行初始化

```
    }
}
/*************************************************
*函数名称：void main(void)
*函数功能：主函数
*入口参数：void
*出口参数：void
*************************************************/
void main(void)
{
    Sys_Init();

    #ifdef   __TOUCHKEY_DEBUG__
    SOCAPI_DeBugTouchKey_Init();
    #endif

    //触控按键初始化
    TouchKeyInit();

    while(1)
    {
       WDTCON  = 0x10;
       if(TimerFlag_1ms==1)
       {
           TimerFlag_1ms=0;
           Sys_Scan();
           BuzzerWork();
       }

    }
}
```

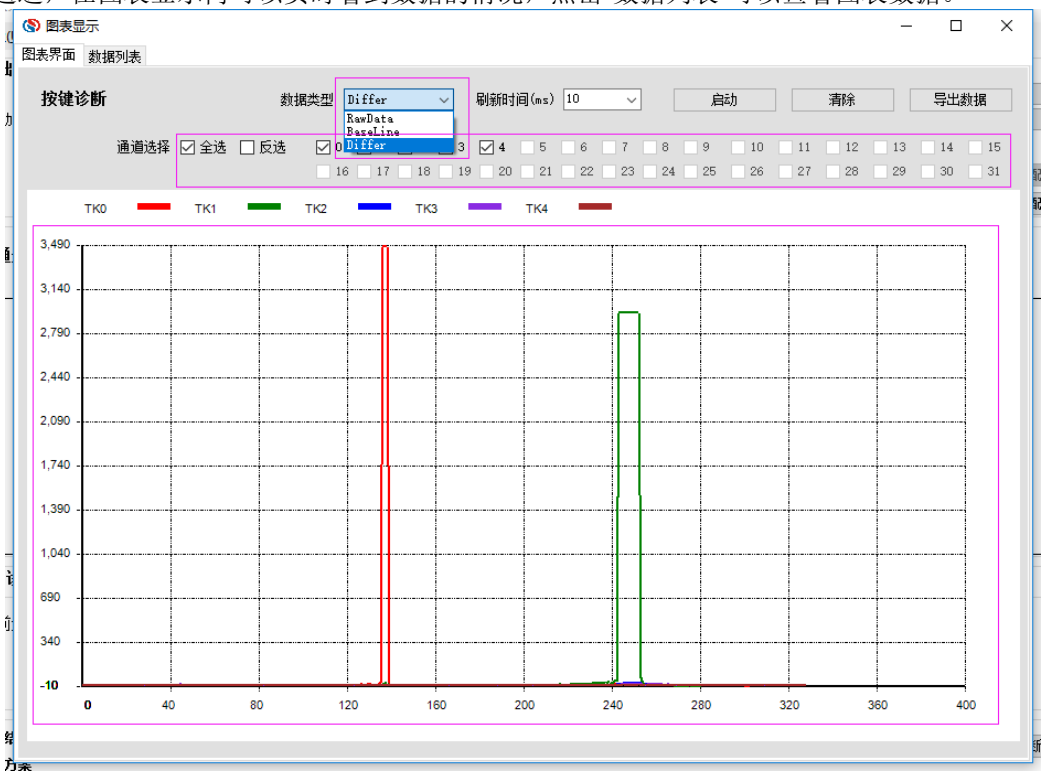
⑤ 打开 Touch Key Tool Menu,选择高灵敏度触控，芯片型号选择与实际芯片对应的型号，调试模式选择动态调试，勾选 TK 通道（必须与项目实际使用的通道一致）。

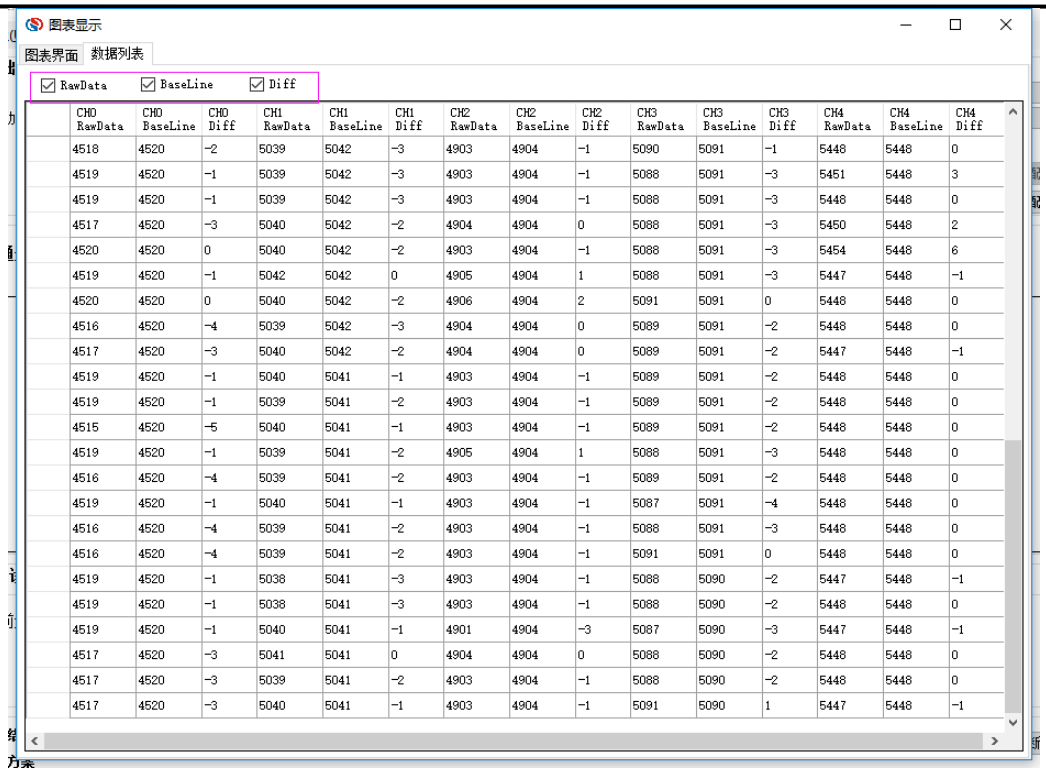


⑥ 点击确定按钮，然后点击下方“动态调试”按钮。



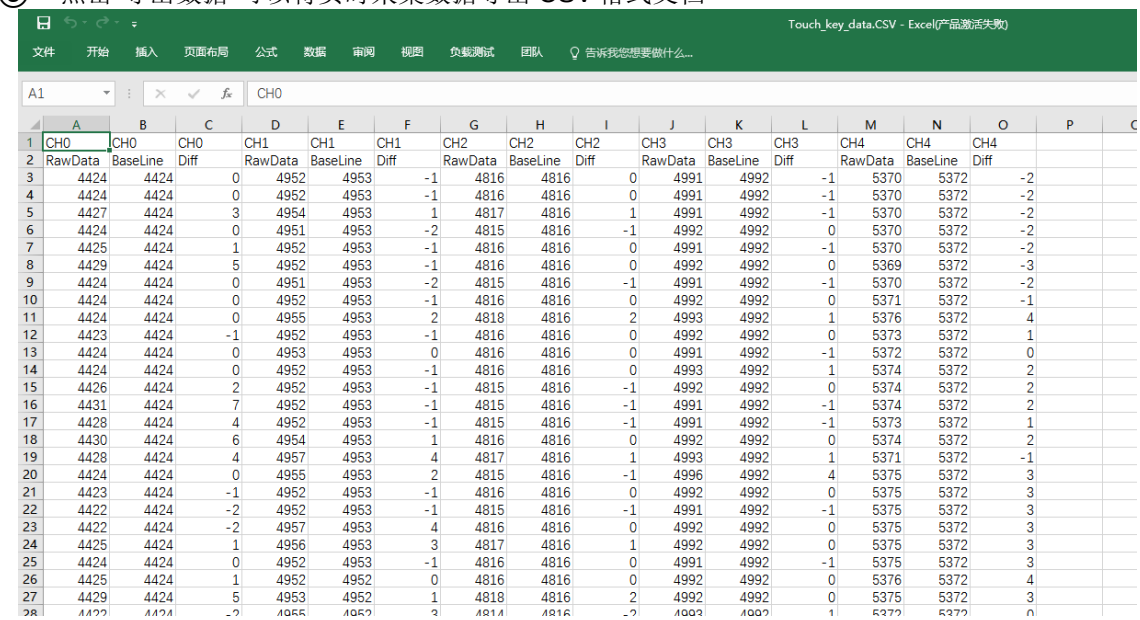
⑦ 在动态调试界面内，可以通过选择“数据类型”查看想要查看的数据，通过“通道选择”勾选要查看的通道，在图表显示内可以实时看到数据的情况，点击“数据列表”可以查看图表数据。





CH0 RawData	CH0 BaseLine	CH0 Diff	CH1 RawData	CH1 BaseLine	CH1 Diff	CH2 RawData	CH2 BaseLine	CH2 Diff	CH3 RawData	CH3 BaseLine	CH3 Diff	CH4 RawData	CH4 BaseLine	CH4 Diff
4518	4520	-2	5039	5042	-3	4903	4904	-1	5090	5091	-1	5448	5448	0
4519	4520	-1	5039	5042	-3	4903	4904	-1	5088	5091	-3	5451	5448	3
4519	4520	-1	5039	5042	-3	4903	4904	-1	5088	5091	-3	5448	5448	0
4517	4520	-3	5040	5042	-2	4904	4904	0	5088	5091	-3	5450	5448	2
4520	4520	0	5040	5042	-2	4903	4904	-1	5088	5091	-3	5454	5448	6
4519	4520	-1	5042	5042	0	4905	4904	1	5088	5091	-3	5447	5448	-1
4520	4520	0	5040	5042	-2	4906	4904	2	5091	5091	0	5448	5448	0
4516	4520	-4	5039	5042	-3	4904	4904	0	5089	5091	-2	5448	5448	0
4517	4520	-3	5040	5042	-2	4904	4904	0	5089	5091	-2	5447	5448	-1
4519	4520	-1	5040	5041	-1	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5039	5041	-2	4903	4904	-1	5089	5091	-2	5448	5448	0
4515	4520	-5	5040	5041	-1	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5039	5041	-2	4905	4904	1	5088	5091	-3	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5040	5041	-1	4903	4904	-1	5087	5091	-4	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5088	5091	-3	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5091	5091	0	5448	5448	0
4519	4520	-1	5038	5041	-3	4903	4904	-1	5088	5090	-2	5447	5448	-1
4519	4520	-1	5038	5041	-3	4903	4904	-1	5088	5090	-2	5448	5448	0
4519	4520	-1	5040	5041	-1	4901	4904	-3	5087	5090	-3	5447	5448	-1
4517	4520	-3	5041	5041	0	4904	4904	0	5088	5090	-2	5448	5448	0
4517	4520	-3	5039	5041	-2	4903	4904	-1	5088	5090	-2	5448	5448	0
4517	4520	-3	5040	5041	-1	4903	4904	-1	5091	5090	1	5447	5448	-1

⑧ 点击“导出数据”可以将实时采集数据导出 CSV 格式文档



CH0 RawData	CH0 BaseLine	CH0 Diff	CH1 RawData	CH1 BaseLine	CH1 Diff	CH2 RawData	CH2 BaseLine	CH2 Diff	CH3 RawData	CH3 BaseLine	CH3 Diff	CH4 RawData	CH4 BaseLine	CH4 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

⑨ 动态调试注意事项

1) 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。
2) 调试主界面勾选的通道必须与实际工程使用的通道一致.
3) 动态调试库占用了 43byte idata 和 504byte ROM 资源，请预留足够资源，保证动态调试程序运行正常。

2.2.2 实现赛元高灵敏度触控软件库的功能测试

1、SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T1_Vx.x.x.LIB 库文件介绍

文件	用途	说明
SC92F8XXX/SC93F8XXX_HighSensitive_Lib	库文件，实现触控按键检测算法	
Sensormethod.h	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
S_TouchKeyCFG.C	C 文件， 实现触控参数与库交互	
S_TOUCHKEYCFG.H	头文件，提供宏供用户修改参数	

2、Lib 用到的资源：

芯片系列	RAM 占用	ROM 占用
SC93F8X3X_HighSensitive_Lib	data 区：28.5 个 byte；Xdata 区：18 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 15 个 Byte； 如有 5 个按键： Data 区 28.5byte, xdata 区 18+5*15=93 byte	库使用 ROM 大小约 3.25K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X4X_HighSensitive_Lib	data 区：25 个 byte；Xdata 区：18 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 15 个 Byte； 如有 5 个按键： Data 区 25byte, xdata 区 18+5*15=93 byte	库使用 ROM 大小约 3.16K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X4XB_HighSensitive_Lib	data 区：25 个 byte；Xdata 区：18 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 15 个 Byte； 如有 5 个按键： Data 区 25byte, xdata 区 18+5*15=93 byte	库使用 ROM 大小约 3.16K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X6XB_HighSensitive_Lib	data 区：25.5 个 byte；Xdata 区：18 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 15 个 Byte； 如有 5 个按键： Data 区 25.5byte, xdata 区 18+5*15=93 byte	库使用 ROM 大小约 3.16K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X7X_HighSensitive_Lib	data 区：25.3 个 byte；Xdata 区：7 个 Byte；与按键的个数无关，均要使用； Xdata 区：每增加一个按键 需要 15 个字节，分别是 5Byte data + 10byte xdata； 如有 5 个按键： Data 区 25.3+5*5 = 50.3 byte, xdata 区 10*5 = 50 byte	库使用 ROM 大小约 2.79K，且增加或减少若干按键，该大小基本不变，差异不到 200byte

3、Lib API 接口函数的调用说明；

函数	用途	说明
TouchKeyInit(void)	**触摸按键初始化**	**1,用户在上电复位后调用一次； 2,本函数通过 S_TOUCHKEYCFG.H 参数配置用户选定的按键通道、按键参数并初始化 Baseline 基线； 3，执行本函数用时约：200~500mS，与按键个数，按键扫描时间，自动校准次数相关;**
TouchKeyRestart(void)	**使能一轮触控按键扫描**	**1，用户主程序来控制何时启动按键扫描； 2，启动按键扫描后，在一轮触控按键扫描完成之前，不能对触控按键通道进行操作：如改触控按键通道为 IO。否则，触控按键功能将无法实现。**
Unsigned long int TouchKeyScan(void)/ Unsigned int TouchKeyScan(void)	**触摸按键算法处理**	**1，用户需要在触控按键一轮扫描完成后调用； 2，如用户未调用该函数，一定不能重新调用 TouchKeyRestart(void)使能一轮触控按键扫描，否则上一轮数据将被当前数据覆盖； 3，执行该函数用时约为：(235*N 个按键)uS； 4，SC92F8X7X 该函数的返回类型是 unsigned int，其他均为 unsigned long int**

4、全局变量 SOCAPI_TouchKeyStatus 的说明

① 全局变量在 S_TouchKeyCFG.c 头文件中声明：

unsigned char xdata SOCAPI_TouchKeyStatus;

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 使能触控按键扫描之前，一定要清掉标志。

清除一轮扫描标志 SOCAPI_TouchKeyStatus &=0x7f;

5、LIB API 接口函数的返回值说明

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。

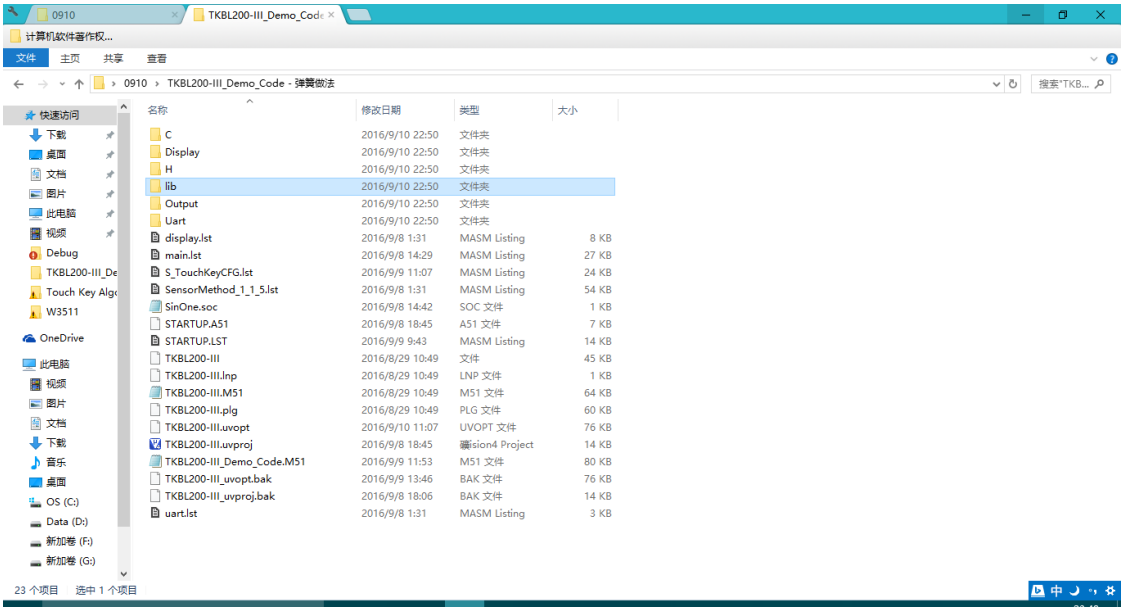
数据位		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
含义		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	触控按键状态（1：有效；0：无效）							

数据位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
含义	TK23	TK22	TK21	TK20	TK19	TK18	TK17	TK16
	触控按键状态（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
	触控按键状态（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
	触控按键状态（1：有效；0：无效）							

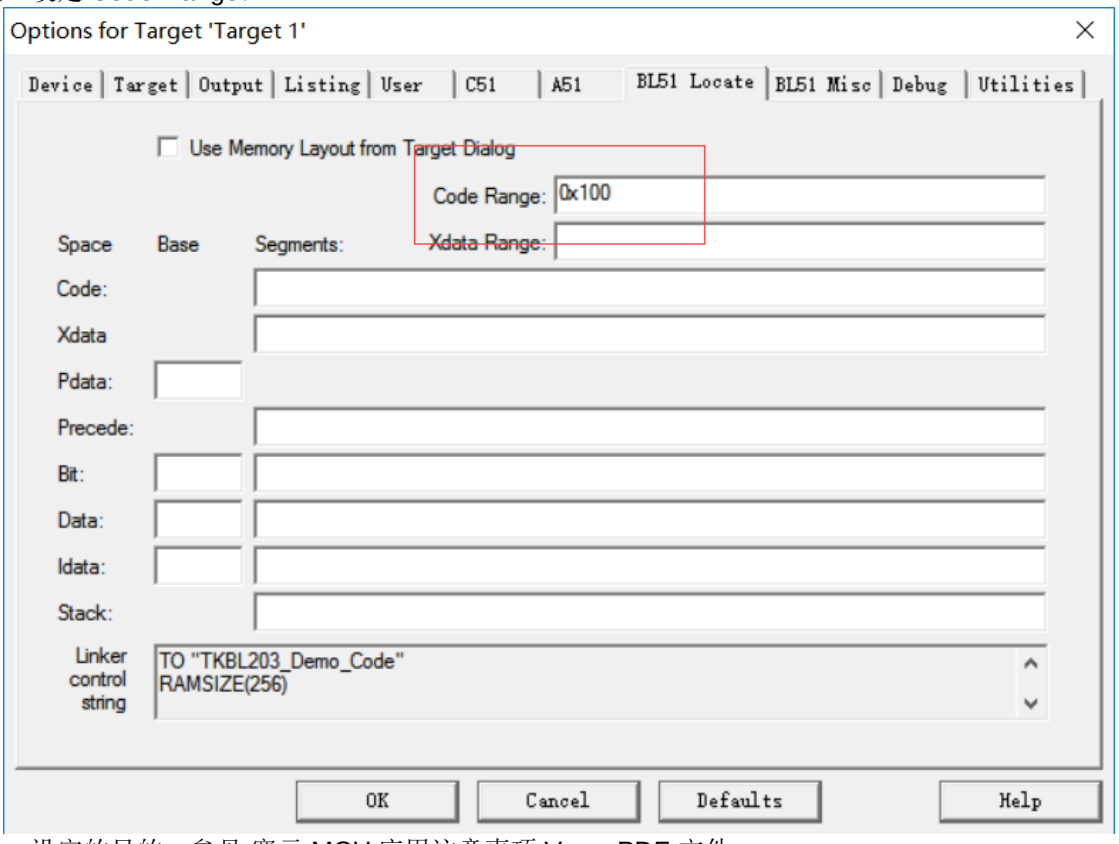
注：SC92F8X7X 函数的返回类型是 unsigned int，其他均为 unsigned long int ；TKn 为触控通道，具体请参照对应规格书。

6、打开工程项目文件复制“lib”文件夹至工程文件夹内。



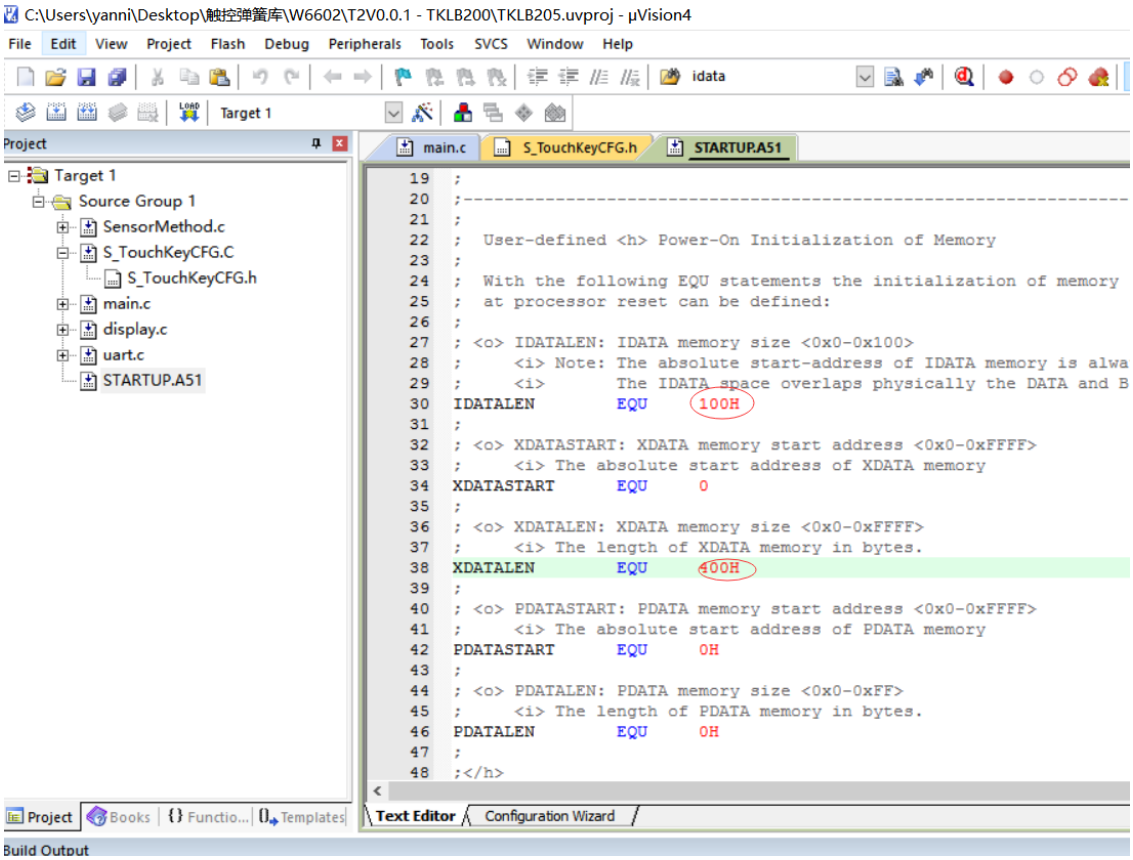
7、在 Keil 中打开工程项目文件；注意设定（Code range）、设定（XDATALEN EQU xxxH）；

① 设定 Code Range：



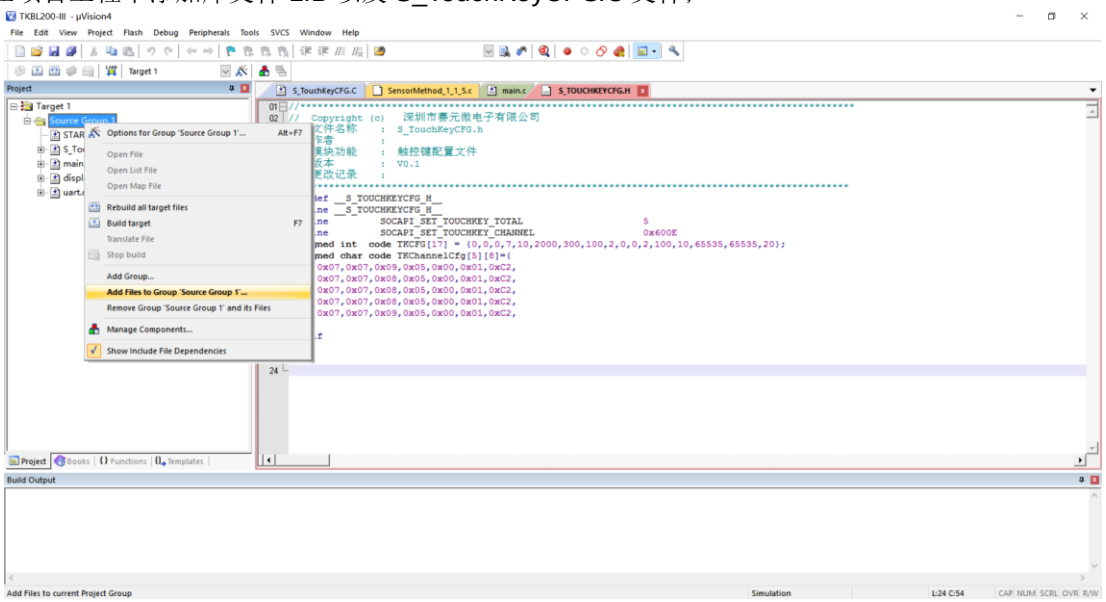
设定的目的，参见 赛元 MCU 应用注意事项 Vx.xx.PDF 文件。

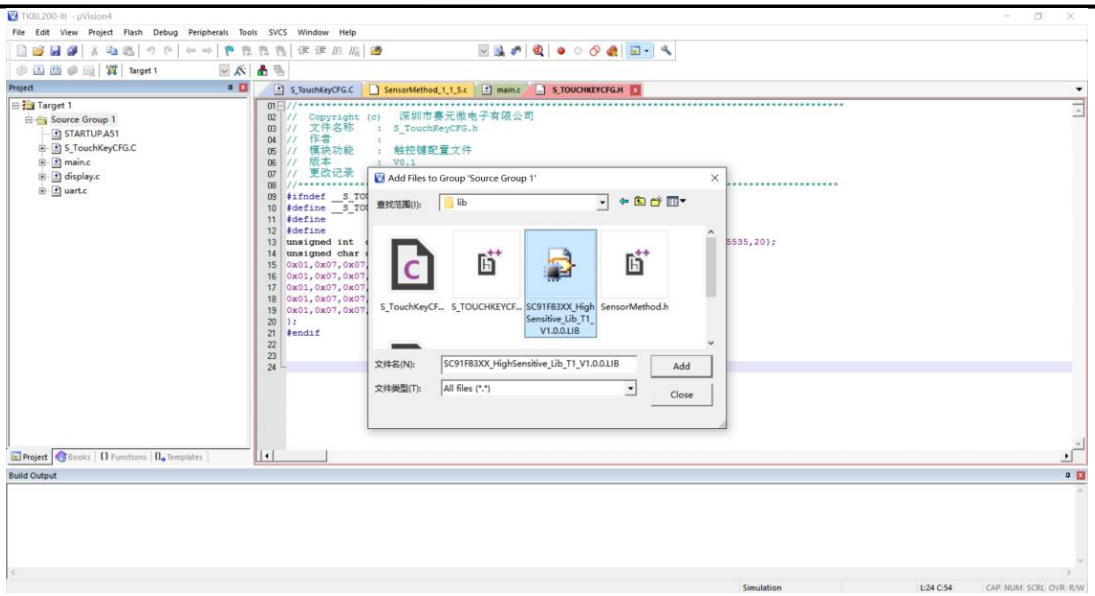
② 设定 XDATALEN：

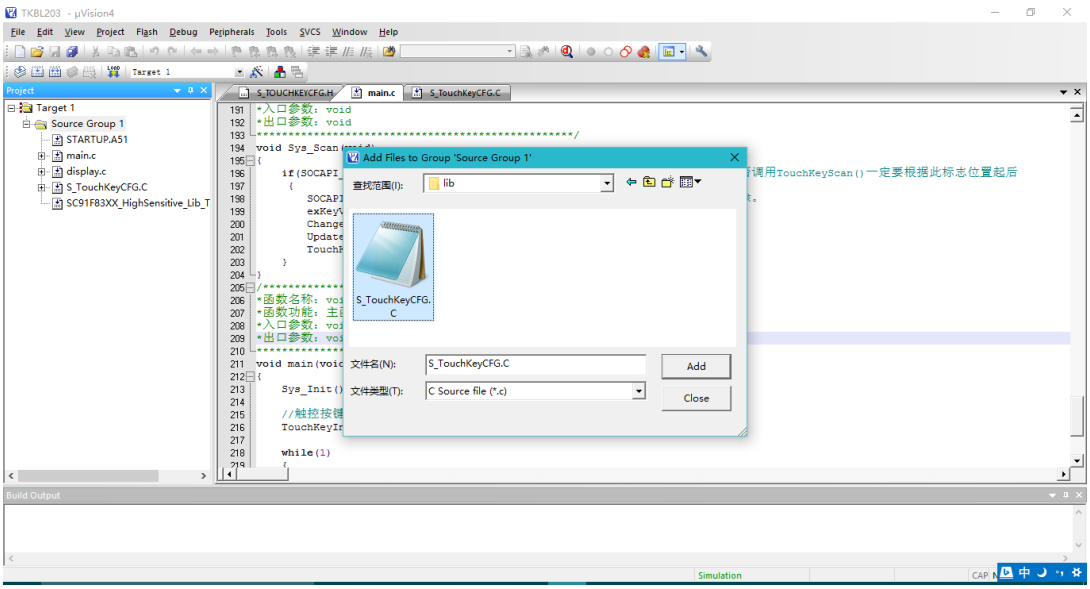


用于 STARTUP.A51 中，清掉外部 XData，具体型号的 XData 大小见规格书。

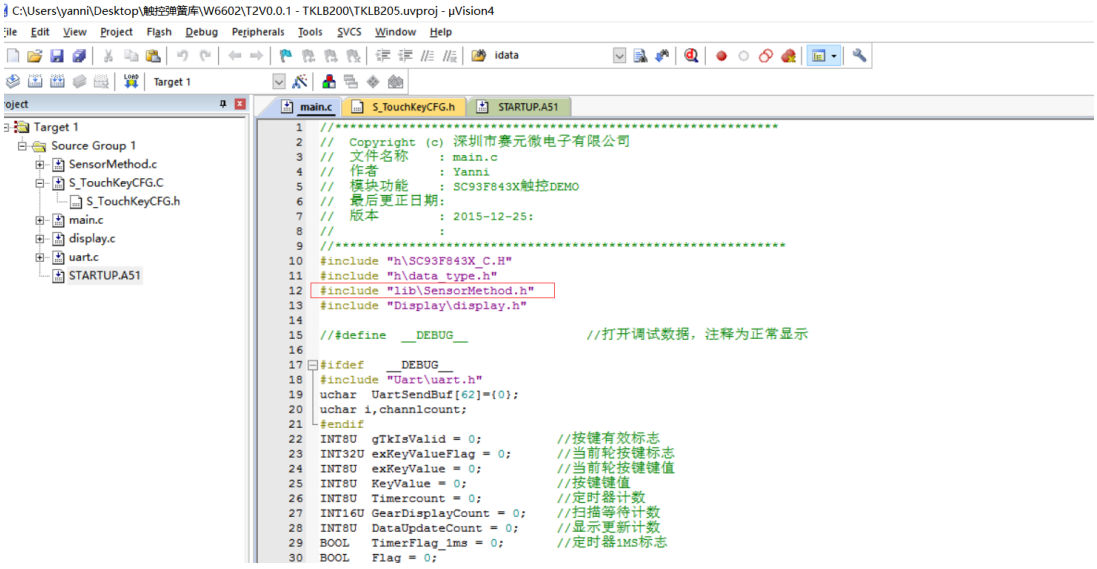
8、在项目工程中添加库文件 LIB 以及 S_TouchKeyCFG.C 文件；



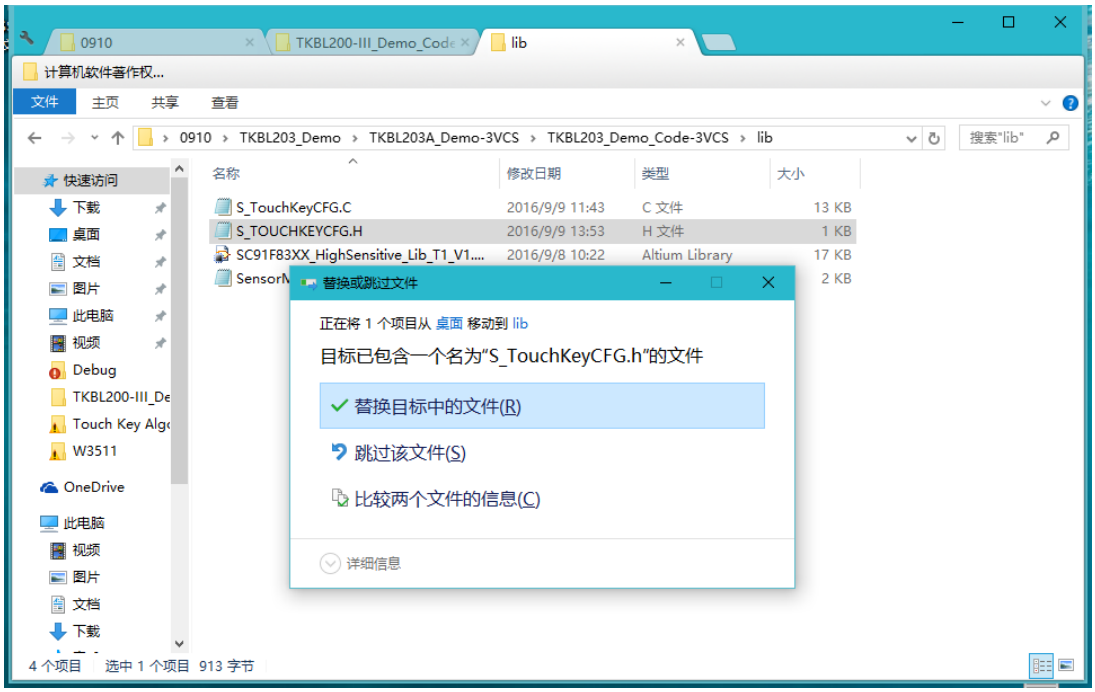




9、在主程序文件中添加头文件引用；



10、将生成的配置文件 S_TOUCHKEYCFG.H 替换到 LIB 文件夹内。



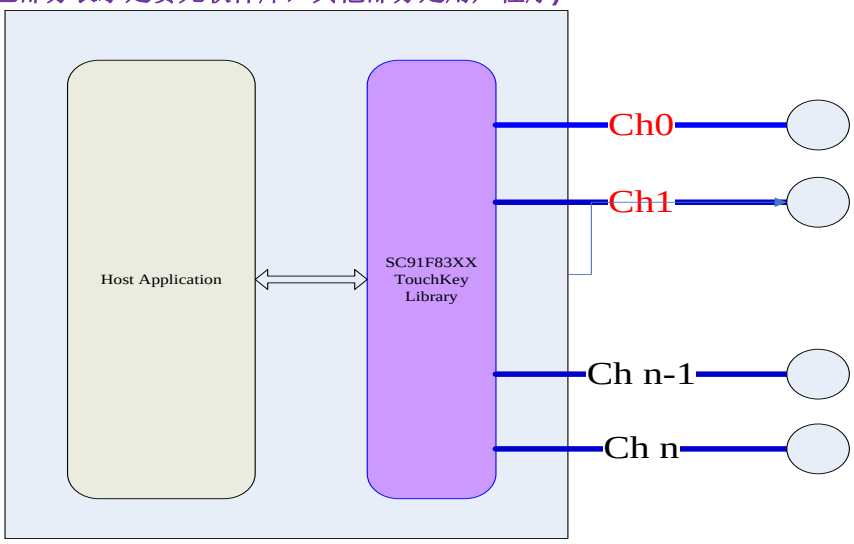
至此，就将触控库添加进项目工程中。

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。

2.2.3 完成用户程序和赛元触控软件库的融合

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：

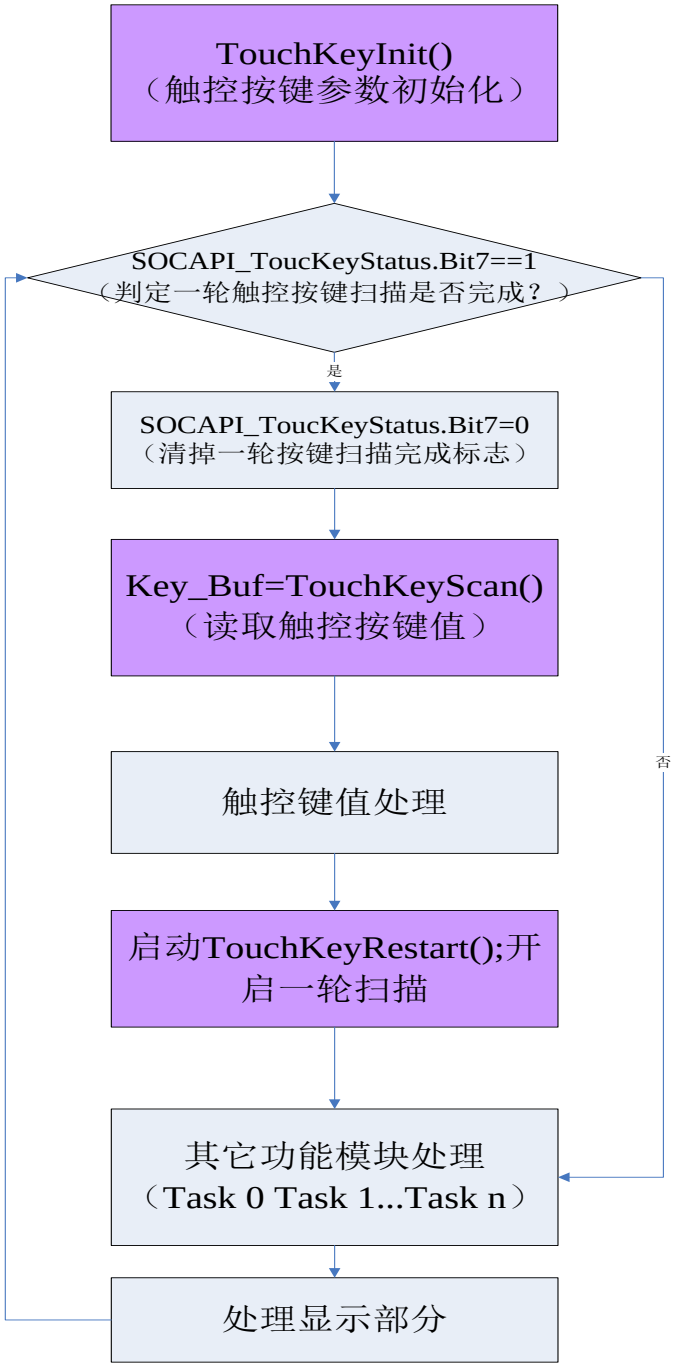
(下面附图中紫色部分表示是赛元软件库，其他部分是用户程序)



2、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_TouckKeyStatus&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程

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；如改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；

4、软件融合的注意事项

① **运行时间：**

接口函数；

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**

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。

2.2.4 库使用注意事项

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。

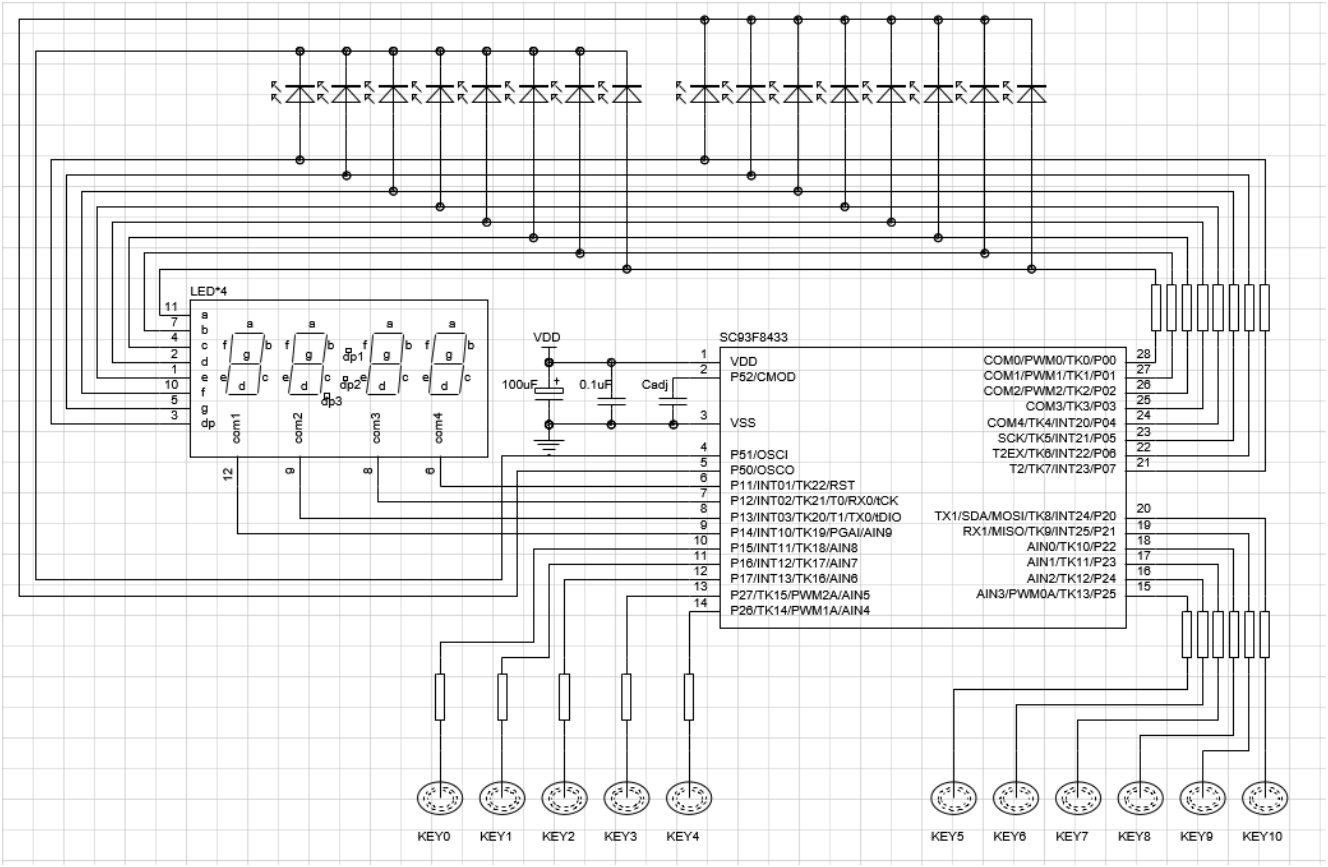
4、感应盘与感应盘之间应该尽量留一定的间距，以保证手指头触控时不会覆盖到 2 个感应盘，同时也能防止感应盘寄生电容过大。

5、基准电容 C 电容建议使用温度系数小精度高的电容，以免造成灵敏度不一致或随温度变化而变化。

6、TK 的 IO 口设置为强推挽输出高。

附录

一、应用参考原理图（以 SC93F8433 为例）



二、程序调用例程

Main.c

```
/*****************************************************
*函数名称：void  Sys_Scan(void)
*函数功能：扫描 TK
*入口参数：void
*出口参数：void
*****************************************************/
void Sys_Scan(void)
{
        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();//按键数据处理函数
                                ChangeTouchKeyvalue();                  //转换键值
                        UpdateLcdBufFunc();             //更新显示数据
                        TouchKeyRestart();                              //启动下一轮转换

        }
}S
/*****************************************************
*函数名称：void main(void)
*函数功能：主函数
*入口参数：void
*出口参数：void
*****************************************************/
void main(void)
{
                Sys_Init();
      TouchKeyInit(); //触控按键初始化
       while(1)
                {
                   WDTCR |= 0x10;
                   if(TimerFlag_1ms==1)
                  {
                                TimerFlag_1ms=0;

                                Sys_Scan();

                                BuzzerWork();
                                DisplayData();      //显示 LED 函数

                   }
                }
}
Display.c
/*****************************************************
*函数名称：void  DisplayData(void)
*函数功能：数码管显示函数
*入口参数：void
*出口参数：void
*****************************************************/
void  DisplayData(void)
{

        ComAllClose;
        if(isLcdComflag == 0)                                           //显示 COM0  数据
        {
                LedSetSegData(gIsLedDataBuf[0]);
                      COM0 = 0;
                isLcdComflag = 1;
        }
        else if(isLcdComflag ==1)                               //显示 COM1 数据
        {
                LedSetSegData(gIsLedDataBuf[1]);
                COM1 = 0;
                isLcdComflag = 2;
        }
        else if(isLcdComflag ==2)
        {
```

```
                isLcdComflag = 0;
                ComAllClose;
            }
}
```

3 SC92F8XXX_SC93F8XXX_HIGHSENSITIVE_LIB_T2 库说明

3.1 前言

3.1.1 赛元高灵敏度触控库介绍

赛元 Ftouch MCU SC92F8XXX/SC93F8XXX 提供一个可以供用户调用的库文件，以降低用户触控按键部分的开发难度。本触控库是 SC92F8XXX/SC93F8XXX 与 LED 不共用，按键与显示需分时扫描的隔空触摸按键库，目前版本仅支持单按键功能。**应用范围：面板上隔空操作，隔空 3MM 或者填充其他介质的应用。按键个数不少于 3 个。**

用户仅仅需要经过以下几个步骤，便可实现触控按键的功能，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。

SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T2_Vx.x.x 的使用分以下几个步骤：

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按键电脑界面软件 SOC HighSensitive TouchKey Tool，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 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 最合适的触控按键关键参数，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。

用户自行写好除触控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。

3.1.2 赛元高灵敏度触控库文件介绍

赛元高灵敏度触控按键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。

SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T2_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 HighSensitive TouchKey Tool 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无需修改。

3.2 调试流程

3.2.1 安装开发工具并配置参数、导出数据

1、安装调试工具及连接硬件；

① Setup Pro51；

安装赛元 Pro51 软件 SOC Pro51 Vx.xx.exe(请从赛元网站找最新版本)。

② Setup SOC TouchKey Tool

安装赛元触控调试软件 SOC TouchKey Tool (请从赛元网站找最新版本)。

③ 升级 DPT52 /SC-LINK 固件,更新 MCU 库；

在线烧写器 DPT52/SC-LINK 的固件和 SOC Pro51 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 的插件安装文件版本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 目录内的 SinOne_Chip 目录中。

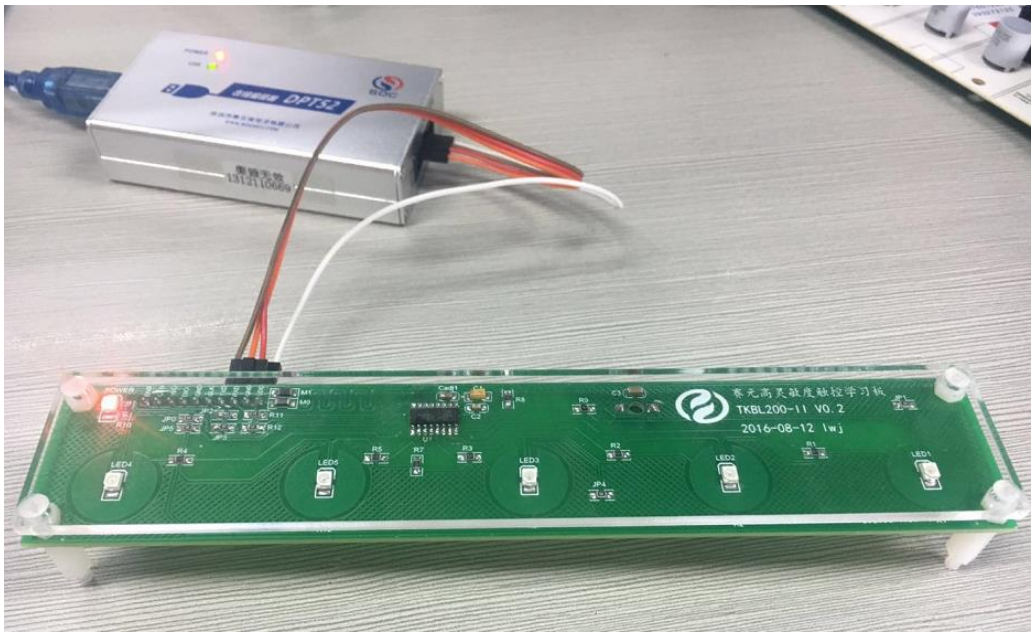
b.SinOne_Chip 目录内所有文件如下：

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：赛元在线开发工具 DPT52 使用说明
SOC_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。

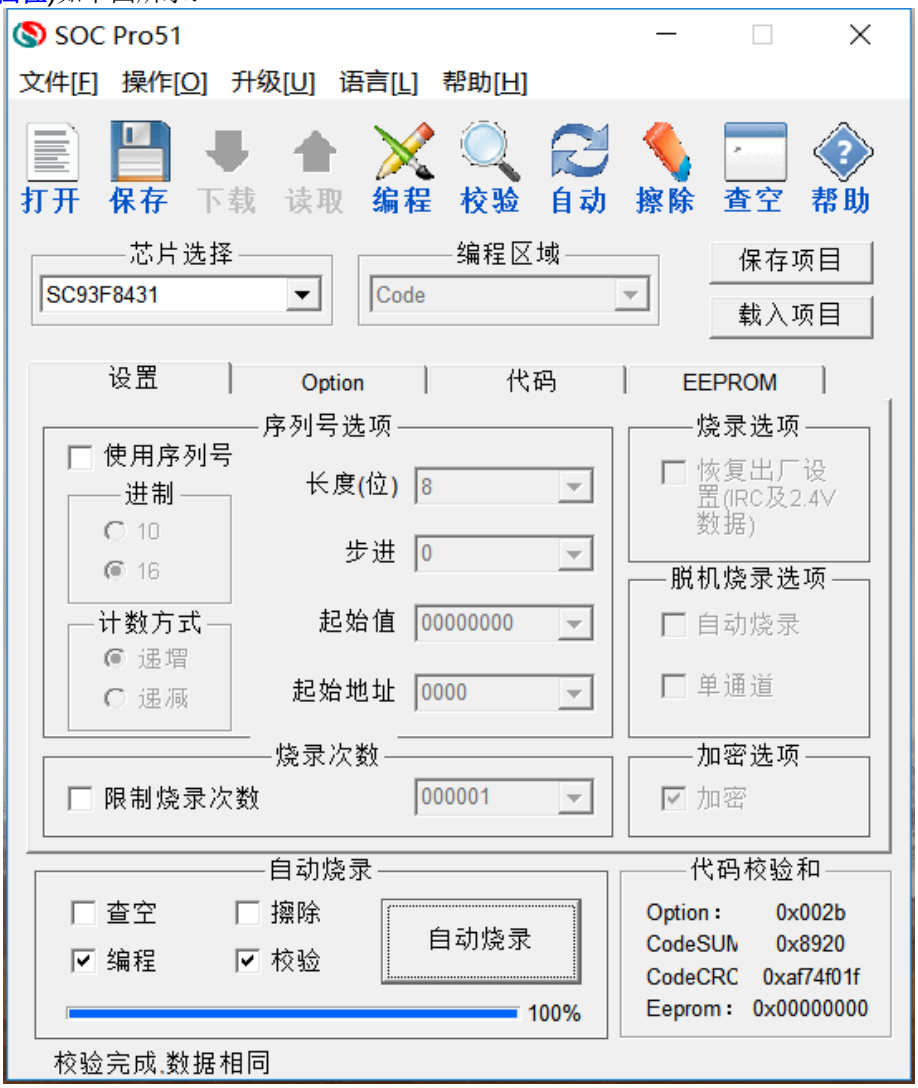
⑤ 硬件连接顺序：电脑 USB-->DPT52/SC-LINK (VCC/GND/CLK/DIO)-->用户 PCB(VCC/GND/tCK/tDIO)；并测试连接正常。调试过程需要用到硬件 UART 资源，请 PCB 预留接线。如图为 DPT52 的接线。





图为 DPT52 的接口

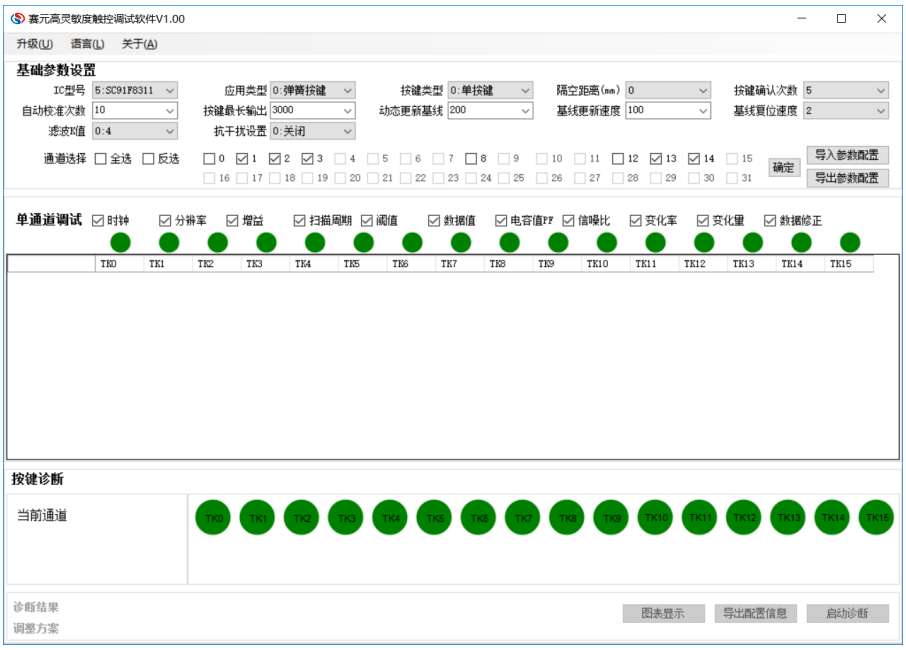
⑦ 烧录 SC92F8XXX/SC93F8XXX_HighSensitiveTKStaticDebug_ Vx.xx.hex 到用户 PCB 上的 SC92F8XXX/SC93F8XXX IC 中； 打开 SOC Pro51 软件，选择项目使用的 MCU 型号，载入“SC92F8XXX/SC93F8XXX _HighSensitiveTKStaticDebug_ Vx.xx.hex”文件，点击“编程”，完成后关闭 SOC Pro51 软件。(**注意:LVR 设置必须比供电电压低,如供电为 3.3V,则 Option 中 LVR 必须选择 3.3V 以下的档位**)如下图所示：



⑧ 将接线接为触摸调试模式，硬件的连接方法为：电脑 USB-->DPT52/SC-LINK (VCC/GND/CLK/DIO)-->用户 PCB(VCC/GND/tCK/tDIO)；并测试连接正常；

2、调试触控参数

① 打开 Touch Key Tool Menu,选择高灵敏度触控



② 参数配置，进入触控调试

a.选择项目使用的 Ftouch MCU 型号以及勾选使用的 TK 通道，如图所示：



b.设置应用的基本信息如下：

应用类型：选择隔空按键

按键类型：选择单按键

隔空距离：按照实际项目设置隔空距离

c.配置触控算法运算的相关参数

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*12MS=60MS。

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时间，单位为轮数。按键时间到达指定次数，则该按键会被释放掉，按键标志会被清除。

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，保持默认不改动。

滤波 K 值：保持默认不改动

抗干扰设置：用于扫描时钟变频，当项目有 EMI 测试要求，需要选择打开 1:12bit。

参考电压：保持默认不改动

d.通道选择，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。

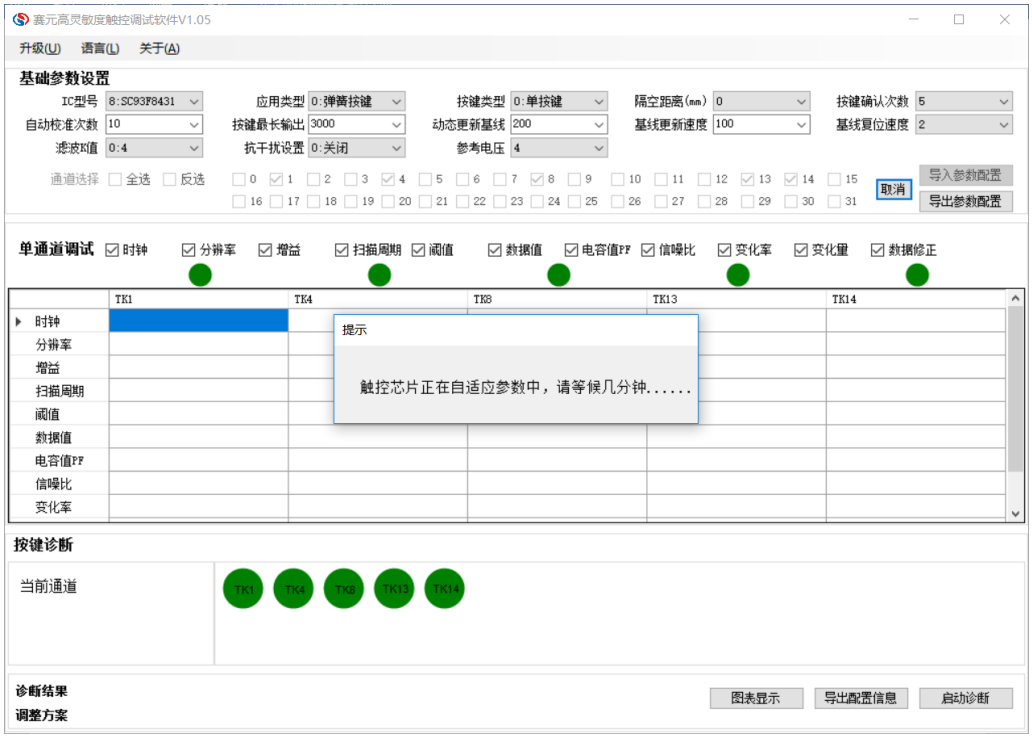
若需要更改通道，需要点击“取消”按钮

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。**

③ 按键参数自适应

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。



④ 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，进入单通道调试界面：



b.设置触控相关参数



时钟：保持默认，不进行改动

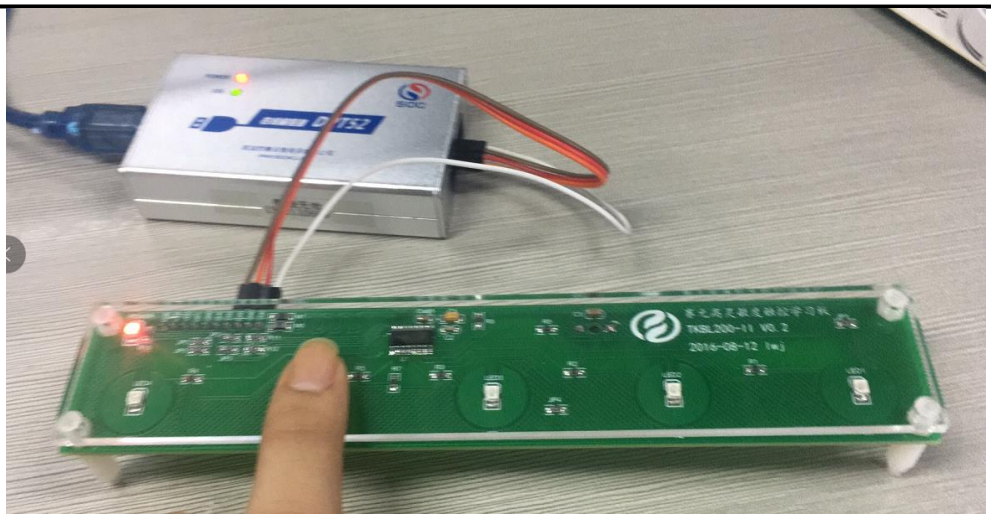
分辨率：保持默认，不进行改动
增益：保持默认，不进行改动
扫描周期：设置范围 1-32，单位为 128us。数值越大，该键扫描时间越长，变化量越大
阈值设置：设置范围 1-8，数值越大，灵敏度越低，反之亦然。如设置值为 5，即阈值设置为变化量的 50%，当数据变化超过阈值认为有键。建议设置为 5。
一般情况下，按键经过自适应过程，用户无需修改以上参数，直接点击启动调试。

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：



触摸过程：





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

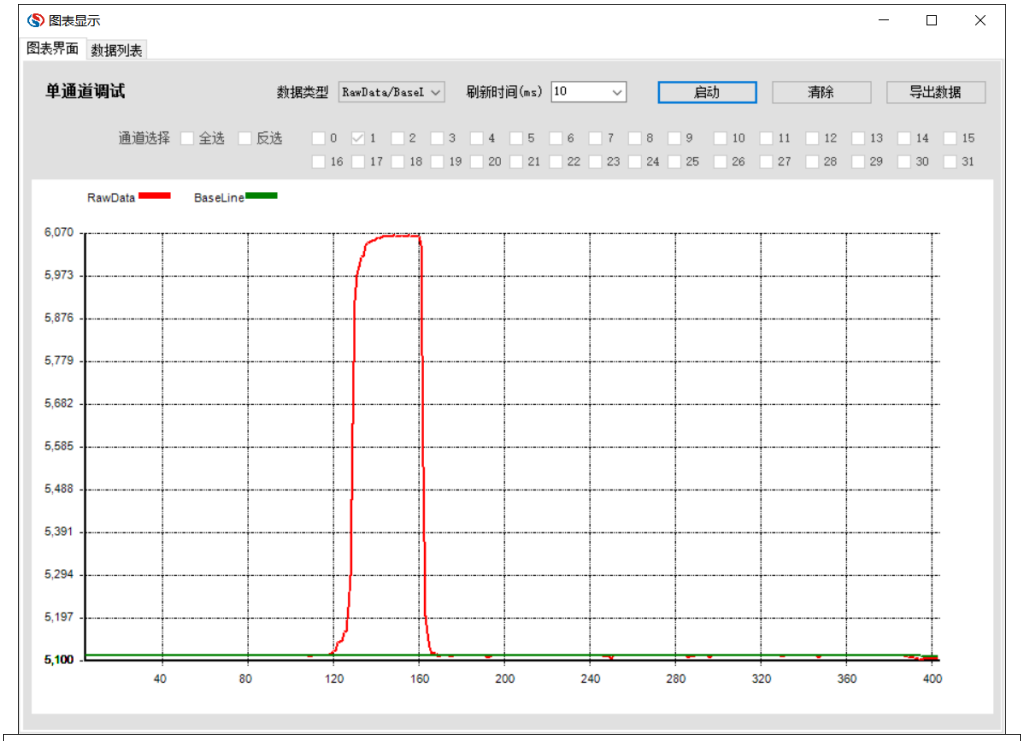


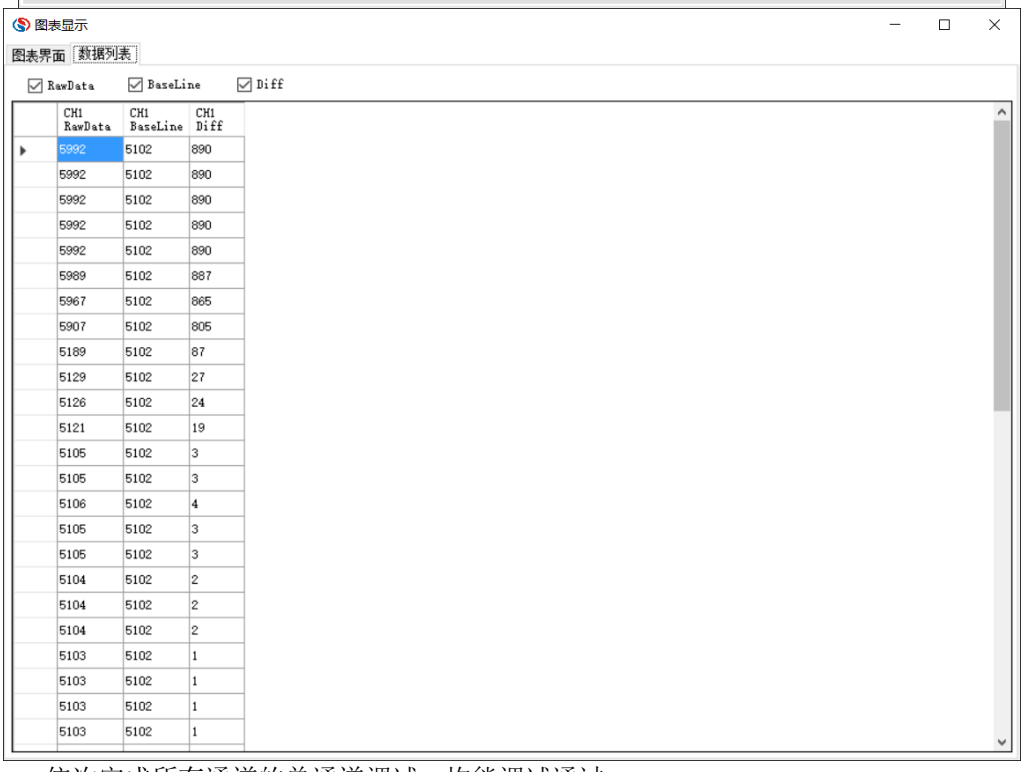
若调试不通过，则显示红色图标。



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

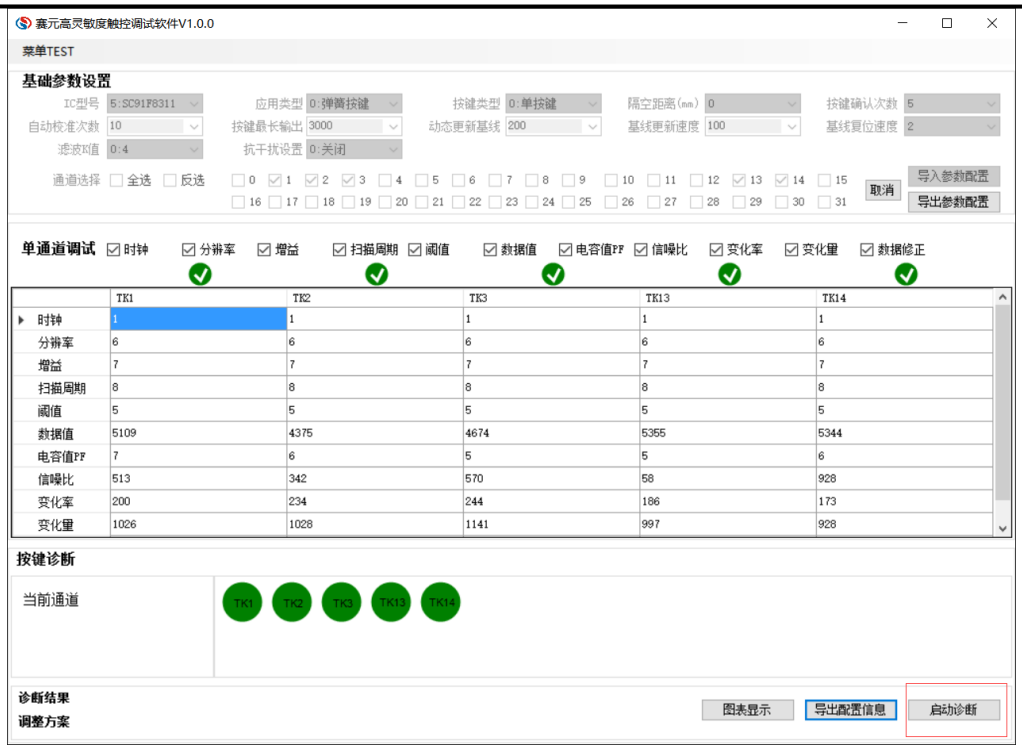




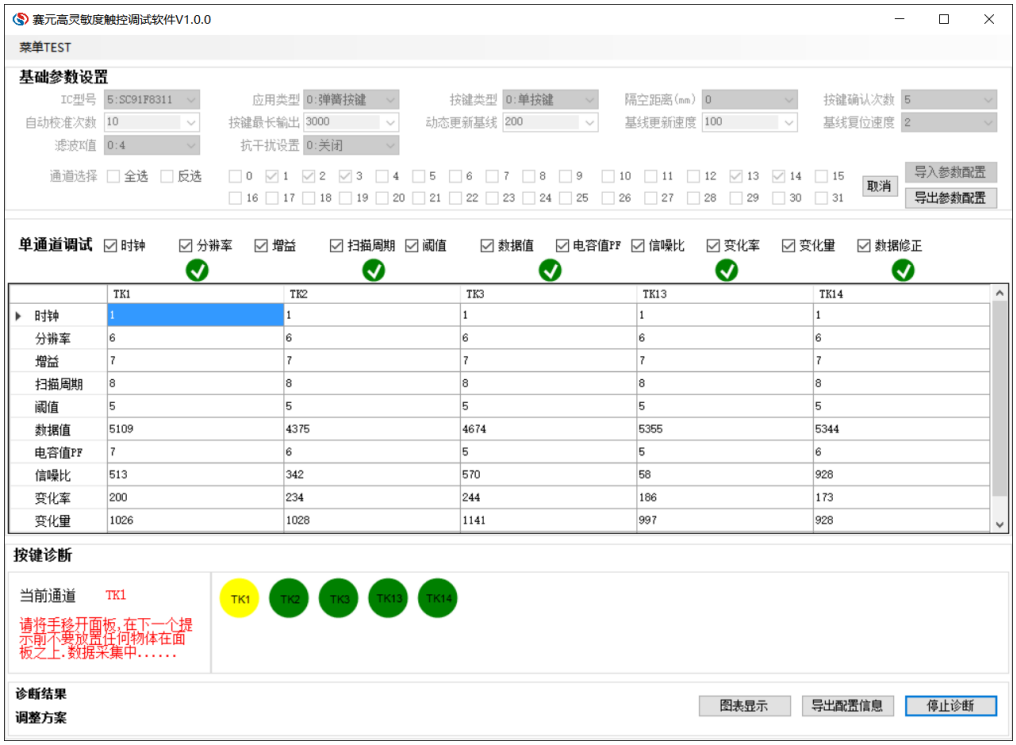
e. 依次完成所有通道的单通道调试，均能调试通过。

⑤ 进行按键诊断

按键诊断是分析按键之间相互影响的过程。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！

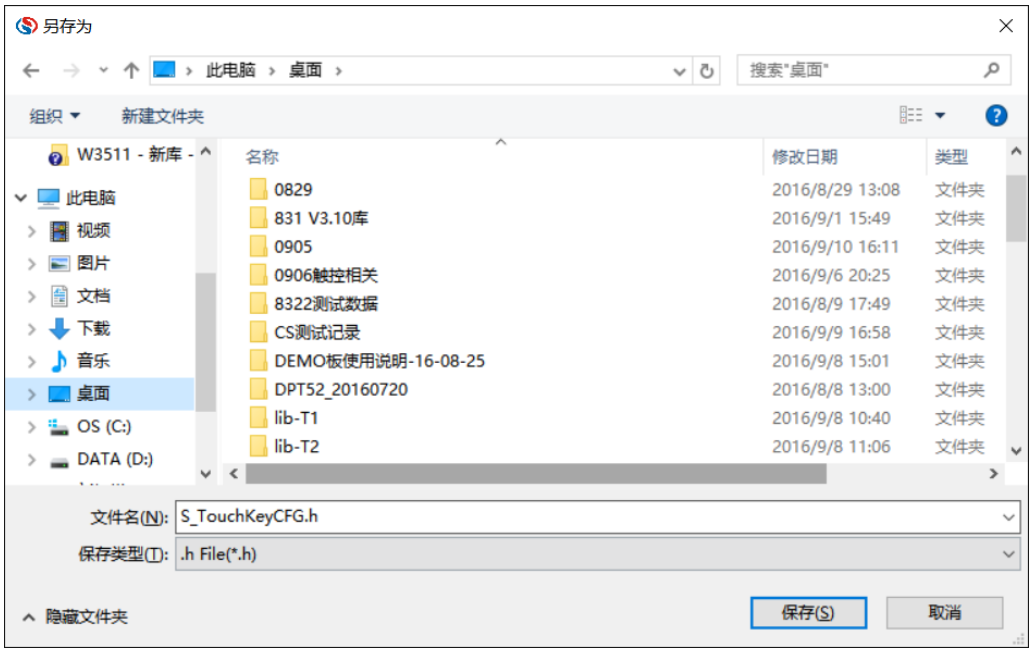


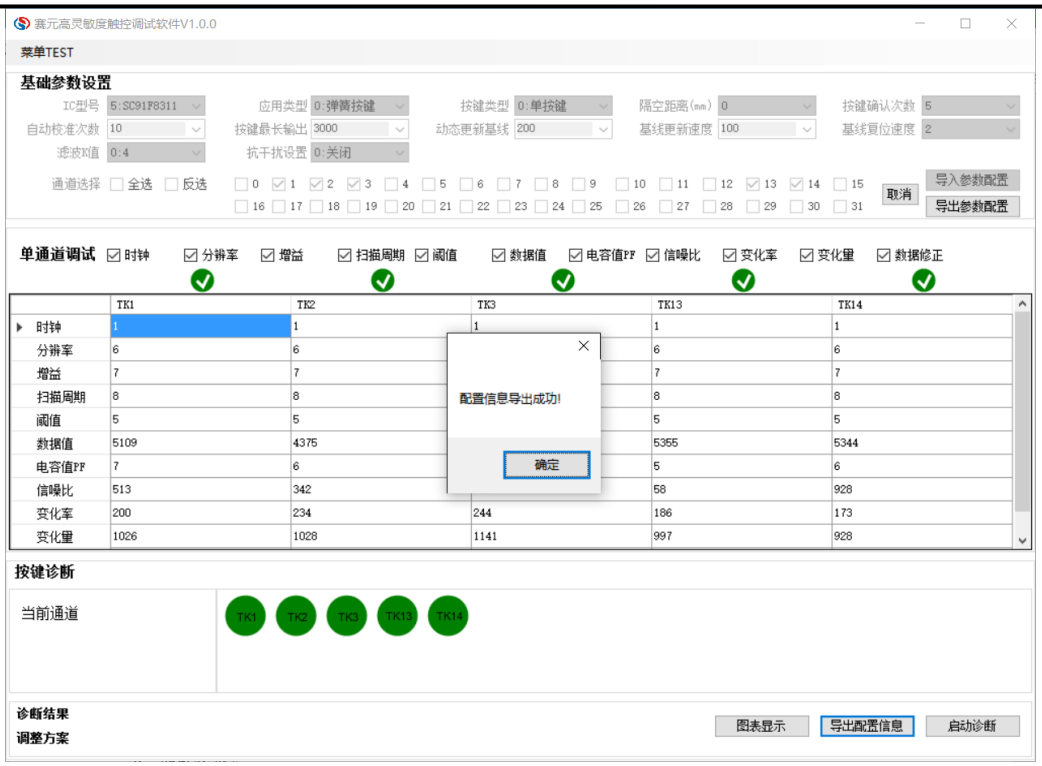


若诊断不通过，请根据诊断结果和调整方案，调整硬件 Layout，如下图是诊断不通过的提示语：

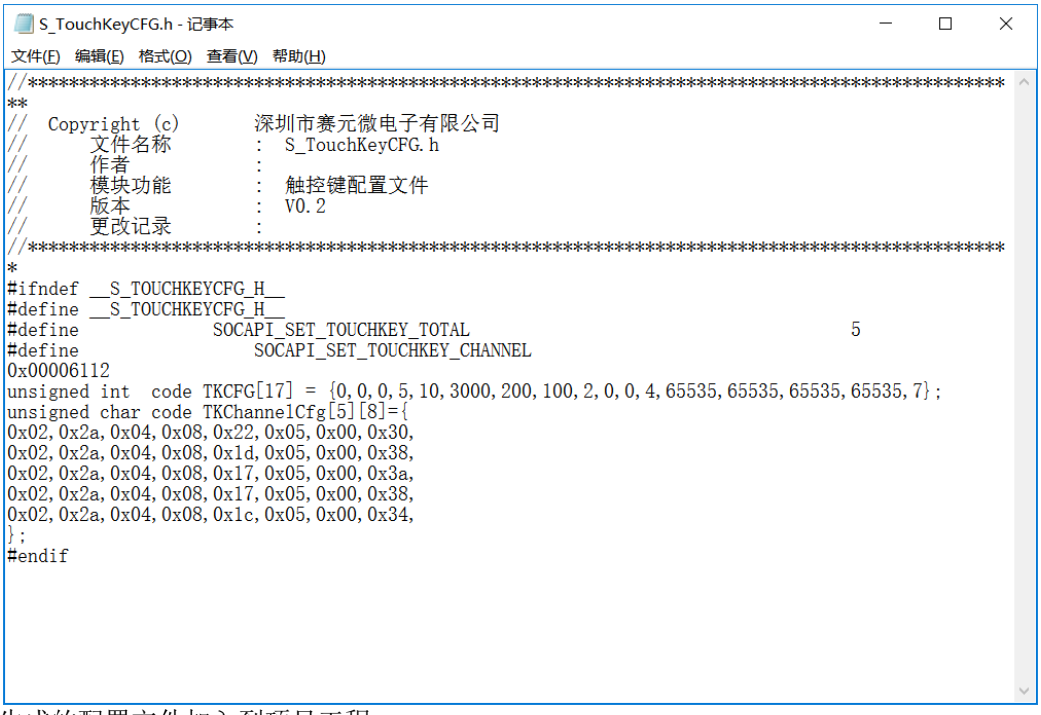


⑥ 完成按键诊断并且测试通过后，点击“导出配置信息”按钮生成配置文件 S_TOUCHKEYCFG.H。





S_TOUCHKEYCFG.H 内容如下：



S_TouchKeyCFG.h - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//*****************************************************************************************************
**
//  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                                   5
#define                 SOCAPI_SET_TOUCHKEY_CHANNEL
0x00006112
unsigned int  code TKCFG[17] = {0,0,0,5,10,3000,200,100,2,0,0,4,65535,65535,65535,65535,7};
unsigned char code TKChannelCfg[5][8]={
0x02,0x2a,0x04,0x08,0x22,0x05,0x00,0x30,
0x02,0x2a,0x04,0x08,0x1d,0x05,0x00,0x38,
0x02,0x2a,0x04,0x08,0x17,0x05,0x00,0x3a,
0x02,0x2a,0x04,0x08,0x17,0x05,0x00,0x38,
0x02,0x2a,0x04,0x08,0x1c,0x05,0x00,0x34,
};
#endif
```

将生成的配置文件加入到项目工程。

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	保持默认 0 不改动
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 65535 不改动
TKCFG[12]		保持默认 65535 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

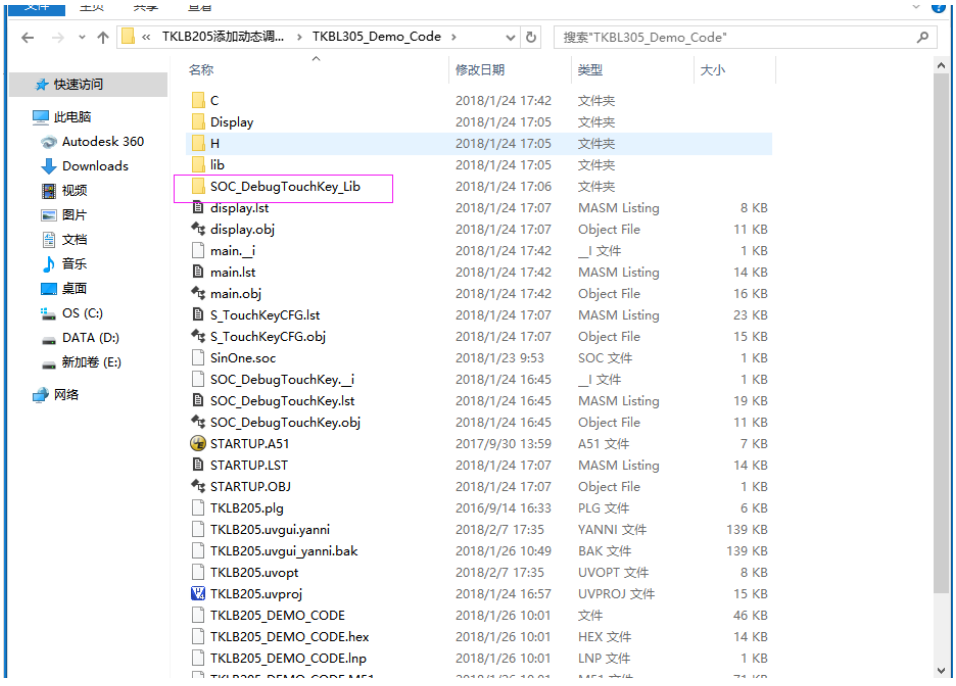
至此触控按键的调试过程结束。

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。

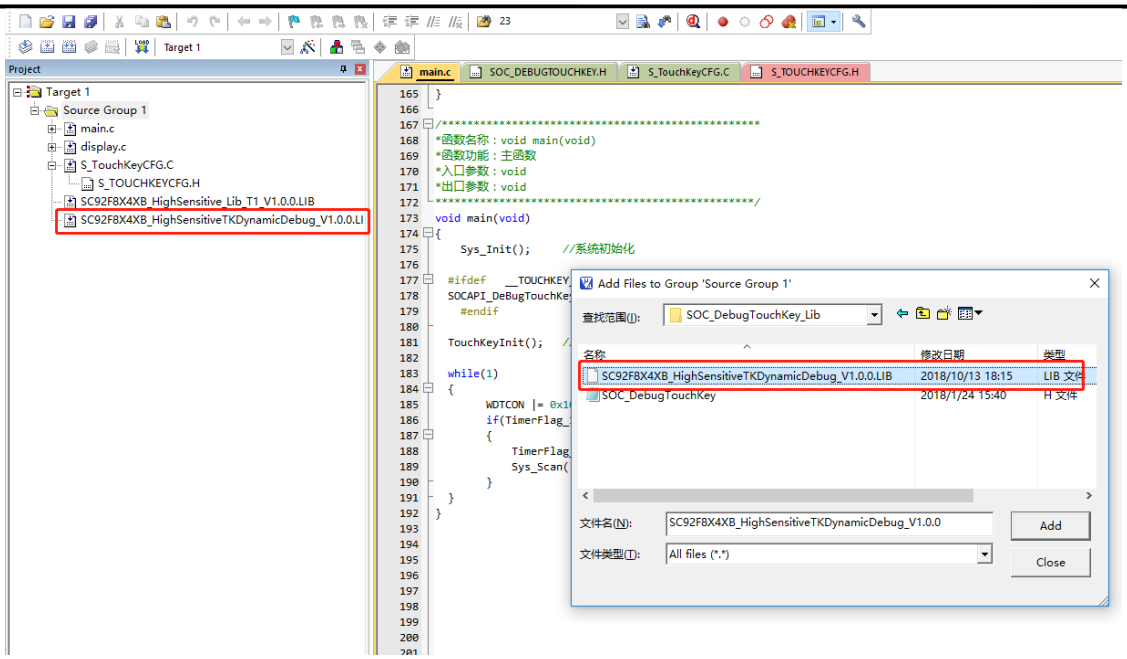
3、加载动态调试库进行动态调试数据

赛元触控动态调试库的功能是可以在用户程序中加入调试代码，利用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。

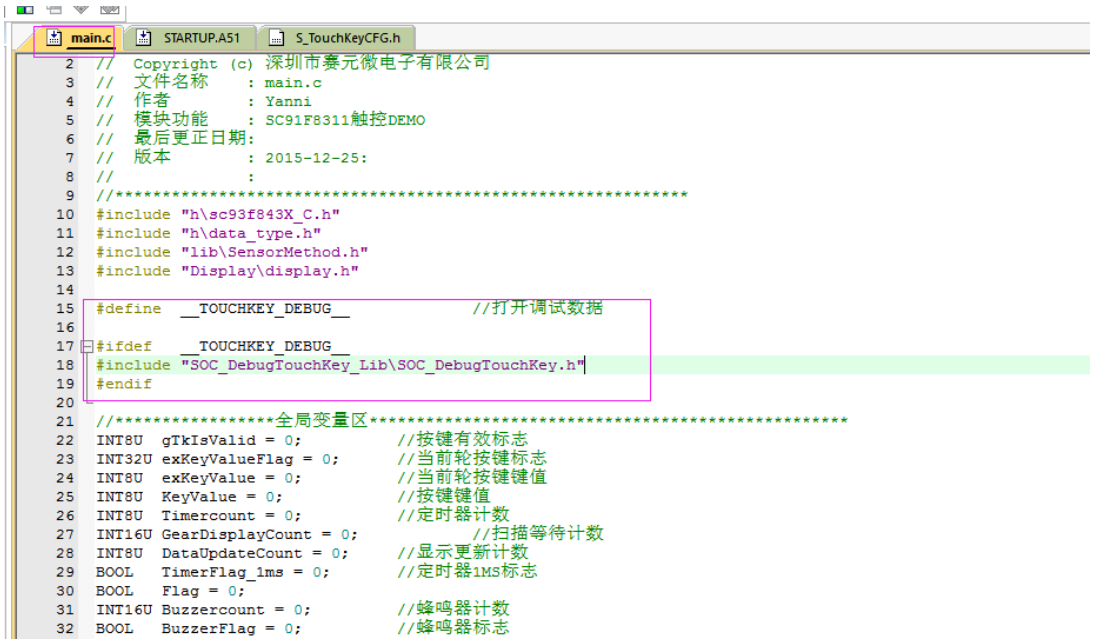
① 将 SOC_DebugTouchKey_Lib 文件夹放置到工程的根目录下



① ② 在用户工程中加入 SC92F8XXX/SC93F8XXX_HighSensitiveTKDynamicDebug_Vx.x.x.LIB 文件



③ 在 main.c 中 include 头文件



③ 在 main 函数中调用 SOCAPI_DeBugTouchKey_Init 进行初始化

```
    }
}
/*************************************************
*函数名称：void main(void)
*函数功能：主函数
*入口参数：void
*出口参数：void
*************************************************/
void main(void)
{
    Sys_Init();

    #ifdef   __TOUCHKEY_DEBUG__
    SOCAPI_DeBugTouchKey_Init();
    #endif

    //触控按键初始化
    TouchKeyInit();

    while(1)
    {
       WDTCON  = 0x10;
       if(TimerFlag_1ms==1)
       {
           TimerFlag_1ms=0;
           Sys_Scan();
           BuzzerWork();
       }

    }
}
```

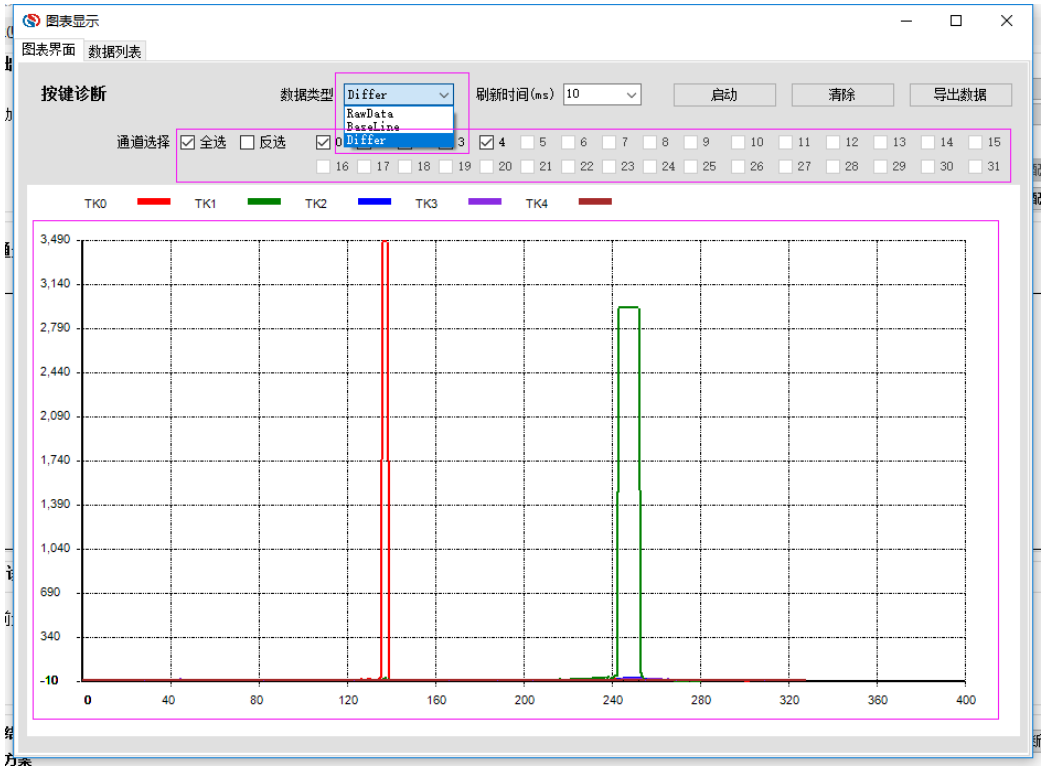
④ 打开 Touch Key Tool Menu,选择高灵敏度触控，芯片型号选择与实际芯片对应的型号，调试模式选择动态调试，勾选 TK 通道（必须与项目实际使用的通道一致）。

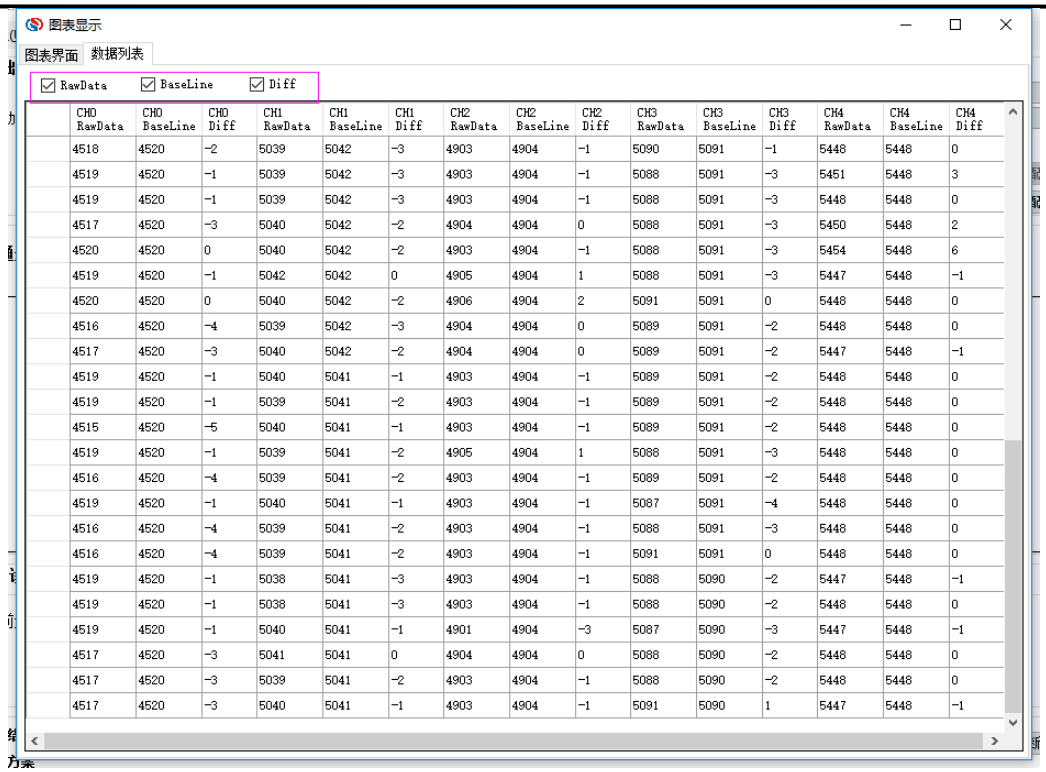


⑥ 点击确定按钮，然后点击下方“动态调试”按钮。



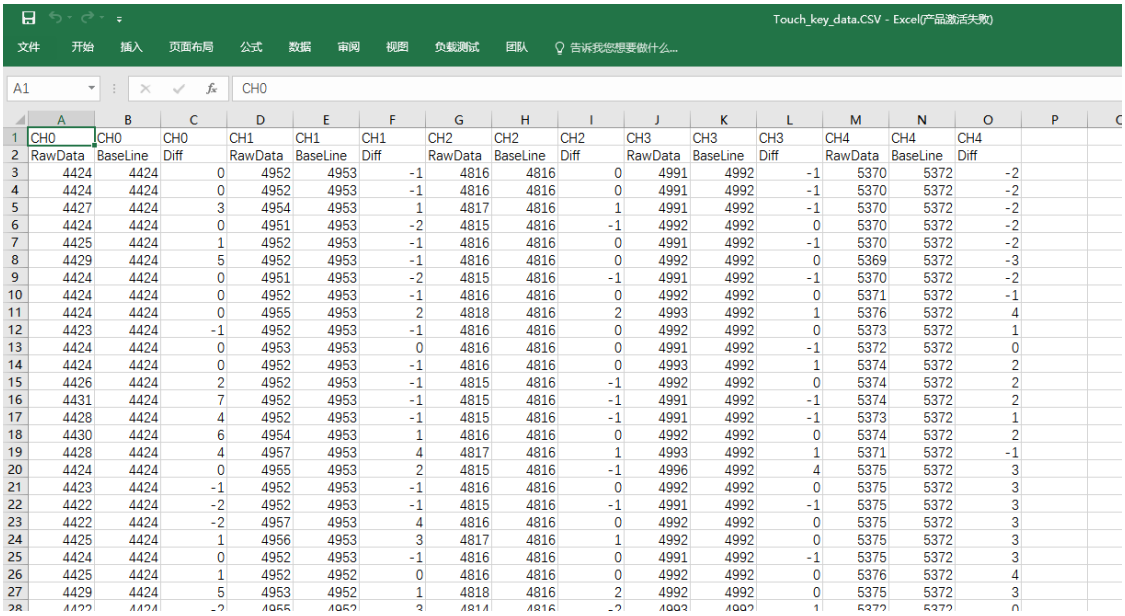
⑦ 在动态调试界面内，可以通过选择“数据类型”查看想要查看的数据，通过“通道选择”勾选要查看的通道，在图表显示内可以实时看到数据的情况，点击“数据列表”可以查看图表数据。





CH0 RawData	CH0 BaseLine	CH0 Diff	CH1 RawData	CH1 BaseLine	CH1 Diff	CH2 RawData	CH2 BaseLine	CH2 Diff	CH3 RawData	CH3 BaseLine	CH3 Diff	CH4 RawData	CH4 BaseLine	CH4 Diff
4518	4520	-2	5039	5042	-3	4903	4904	-1	5090	5091	-1	5448	5448	0
4519	4520	-1	5039	5042	-3	4903	4904	-1	5088	5091	-3	5451	5448	3
4519	4520	-1	5039	5042	-3	4903	4904	-1	5088	5091	-3	5448	5448	0
4517	4520	-3	5040	5042	-2	4904	4904	0	5088	5091	-3	5450	5448	2
4520	4520	0	5040	5042	-2	4903	4904	-1	5088	5091	-3	5454	5448	6
4519	4520	-1	5042	5042	0	4905	4904	1	5088	5091	-3	5447	5448	-1
4520	4520	0	5040	5042	-2	4906	4904	2	5091	5091	0	5448	5448	0
4516	4520	-4	5039	5042	-3	4904	4904	0	5089	5091	-2	5448	5448	0
4517	4520	-3	5040	5042	-2	4904	4904	0	5089	5091	-2	5447	5448	-1
4519	4520	-1	5040	5041	-1	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5039	5041	-2	4903	4904	-1	5089	5091	-2	5448	5448	0
4515	4520	-5	5040	5041	-1	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5039	5041	-2	4905	4904	1	5088	5091	-3	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5089	5091	-2	5448	5448	0
4519	4520	-1	5040	5041	-1	4903	4904	-1	5087	5091	-4	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5088	5091	-3	5448	5448	0
4516	4520	-4	5039	5041	-2	4903	4904	-1	5091	5091	0	5448	5448	0
4519	4520	-1	5038	5041	-3	4903	4904	-1	5088	5090	-2	5447	5448	-1
4519	4520	-1	5038	5041	-3	4903	4904	-1	5088	5090	-2	5448	5448	0
4519	4520	-1	5040	5041	-1	4901	4904	-3	5087	5090	-3	5447	5448	-1
4517	4520	-3	5041	5041	0	4904	4904	0	5088	5090	-2	5448	5448	0
4517	4520	-3	5039	5041	-2	4903	4904	-1	5088	5090	-2	5448	5448	0
4517	4520	-3	5040	5041	-1	4903	4904	-1	5091	5090	1	5447	5448	-1

⑨ 点击“导出数据”可以将实时采集数据导出 CSV 格式文档



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	CH0	CH0	CH0	CH1	CH1	CH1	CH2	CH2	CH2	CH3	CH3	CH3	CH4	CH4	CH4	
2	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff	
3	4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2	
4	4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2	
5	4427	4424	3	4954	4953	1	4817	4816	1	4991	4992	-1	5370	5372	-2	
6	4424	4424	0	4951	4953	-2	4815	4816	-1	4992	4992	0	5370	5372	-2	
7	4425	4424	1	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2	
8	4429	4424	5	4952	4953	-1	4816	4816	0	4992	4992	0	5369	5372	-3	
9	4424	4424	0	4951	4953	-2	4815	4816	-1	4991	4992	-1	5370	5372	-2	
10	4424	4424	0	4952	4953	-1	4816	4816	0	4992	4992	0	5371	5372	-1	
11	4424	4424	0	4955	4953	2	4818	4816	2	4993	4992	1	5376	5372	4	
12	4423	4424	-1	4952	4953	-1	4816	4816	0	4992	4992	0	5373	5372	1	
13	4424	4424	0	4953	4953	0	4816	4816	0	4991	4992	-1	5372	5372	0	
14	4424	4424	0	4952	4953	-1	4816	4816	0	4993	4992	1	5374	5372	2	
15	4426	4424	2	4952	4953	-1	4815	4816	-1	4992	4992	0	5374	5372	2	
16	4431	4424	7	4952	4953	-1	4815	4816	-1	4991	4992	-1	5374	5372	2	
17	4428	4424	4	4952	4953	-1	4815	4816	-1	4991	4992	-1	5373	5372	1	
18	4430	4424	6	4954	4953	1	4816	4816	0	4992	4992	0	5374	5372	2	
19	4428	4424	4	4957	4953	4	4817	4816	1	4993	4992	1	5371	5372	-1	
20	4424	4424	0	4955	4953	2	4815	4816	-1	4996	4992	4	5375	5372	3	
21	4423	4424	-1	4952	4953	-1	4816	4816	0	4992	4992	0	5375	5372	3	
22	4422	4424	-2	4952	4953	-1	4815	4816	-1	4991	4992	-1	5375	5372	3	
23	4422	4424	-2	4957	4953	4	4816	4816	0	4992	4992	0	5375	5372	3	
24	4425	4424	1	4956	4953	3	4817	4816	1	4992	4992	0	5375	5372	3	
25	4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5375	5372	3	
26	4425	4424	1	4952	4952	0	4816	4816	0	4992	4992	0	5376	5372	4	
27	4429	4424	5	4953	4952	1	4818	4816	2	4992	4992	0	5375	5372	3	

② 动态调试注意事项

4) 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。

5) 调试主界面勾选的通道必须与实际工程使用的通道一致.

6) 动态调试库占用了 43byte idata 和 504byte ROM 资源，请预留足够资源，保证动态调试程序运行正常。

3.2.2 实现赛元高灵敏度触控软件库的功能测试

1、SC92F8XXX/SC93F8XXX_HighSensitive_Lib_T2_Vx.x.x.LIB 库文件介绍

文件	用途	说明
SC92F8XXX/SC93F8XXX_HighSensitive_Lib	库文件，实现触控按键检测算法	
Sensormethod.h	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
S_TouchKeyCFG.C	C 文件， 实现触控参数与库交互	
S_TOUCHKEYCFG.H	头文件，提供宏供用户修改参数	

2、Lib 用到的资源：

芯片系列	RAM 占用	ROM 占用
SC93F8X3X_HighSensitive_Lib	data 区：38.5 个 byte；Xdata 区：14 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 14 个 Byte； 如有 5 个按键： Data 区 38.5byte, xdata 区 14+5*14=84 byte	库使用 ROM 大小约 3.19K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X4X_HighSensitive_Lib	data 区：38.5 个 byte；Xdata 区：14 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 14 个 Byte； 如有 5 个按键： Data 区 38.5byte, xdata 区 14+5*14=84 byte	库使用 ROM 大小约 3.19K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X4XB_HighSensitive_Lib	data 区：38.5 个 byte；Xdata 区：14 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 14 个 Byte； 如有 5 个按键： Data 区 38.5byte, xdata 区 14+5*14=84 byte	库使用 ROM 大小约 3.19K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X6XB_HighSensitive_Lib	data 区：35.5 个 byte；Xdata 区：14 个 Byte；与按键的个数无关，均要使用； Xdata 区：每一个按键 14 个 Byte； 如有 5 个按键： Data 区 35.5byte, xdata 区 14+5*14=84 byte	库使用 ROM 大小约 3.10K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X7X_HighSensitive_Lib	data 区：34.3 个 byte；Xdata 区：7 个 Byte；与按键的个数无关，均要使用； Xdata 区：每增加一个按键 需要 14 个字节，分别是 4Byte data + 10byte xdata； 如有 5 个按键： Data 区 34.3+4*5 = 54.3 byte, xdata 区 7+10*5 = 57 byte	库使用 ROM 大小约 2.88K，且增加或减少若干按键，该大小基本不变，差异不到 200byte

3、Lib API 接口函数的调用说明；

函数	用途	说明
TouchKeyInit(void)	触摸按键初始化	1,本函数通过 S_TOUCHKEYCFG.H 参数配置用户选定的按键通道、按键参数并初始化 Baseline 基线； 2，执行本函数用时约：200~500mS，与按键个数，按键扫描时间，自动校准次数相关; 3,在该函数执行期间不要做显示动作
TouchKeyRestart(void)	使能一轮触控按键扫描	1，用户主程序来控制何时启动按键扫描； 2，启动按键扫描后，在一轮或者半轮触控按键扫描完成之前，不能对触控按键通道进行操作：如改触控按键通道为 IO。否则，触控按键功能将无法实现。
Unsigned long int TouchKeyScan(void)/ Unsigned int TouchKeyScan(void)	触摸按键算法处理	1，用户需要在触控按键一轮扫描完成后调用； 2，如用户未调用该函数，一定不能重新调用 TouchKeyRestart(void)使能一轮触控按键扫描，否则上一轮数据将被当前数据覆盖； 3，执行该函数用时约：(240*N 个按键)us； 4，SC92F8X7X 该函数的返回类型是 unsigned int，其他均为 unsigned long int

4、全局变量 SOCAPI_TouchKeyStatus 的说明

① 全局变量在 S_TouchKeyCFG.c 头文件中声明：

unsigned char xdata SOCAPI_TouchKeyStatus;

SOCAPI_TouchKeyStatus Bit7 为 1 时 表示当前一轮按键扫描完成;

SOCAPI_TouchKeyStatus Bit6 为 1 时 表示当前半轮按键扫描完成;

注：由于按键和显示需要分时扫描，当用户使用的 KEY 个数大于 8 个时，扫描时间会较长，如果显示部分的扫描频率低于 60HZ，那么显示会抖动，所以触控库会将按键分成两部分进行扫描，此时半轮扫描完成标志 Bit6 位有效。若用户使用的 KEY 个数在 8 个或者 8 个以下，则无需处理半轮扫描完成标志。触控库不会置起 Bit6 位，此时 Bit6 无效。

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；

if(SOCAPI_TouchKeyStatus&0x40)时，完成半轮扫描，跳转去执行显示相关；在显示完成一个周期后再调用 TouchKeyRestart(void)函数继续后半轮扫描。

③ 使能触控按键扫描之前，一定要清掉标志。

清除一轮扫描标志 SOCAPI_TouchKeyStatus &=0x7f;

清除半轮扫描标志 SOCAPI_TouchKeyStatus &=0xbf;

5、LIB API 接口函数的返回值说明；

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。

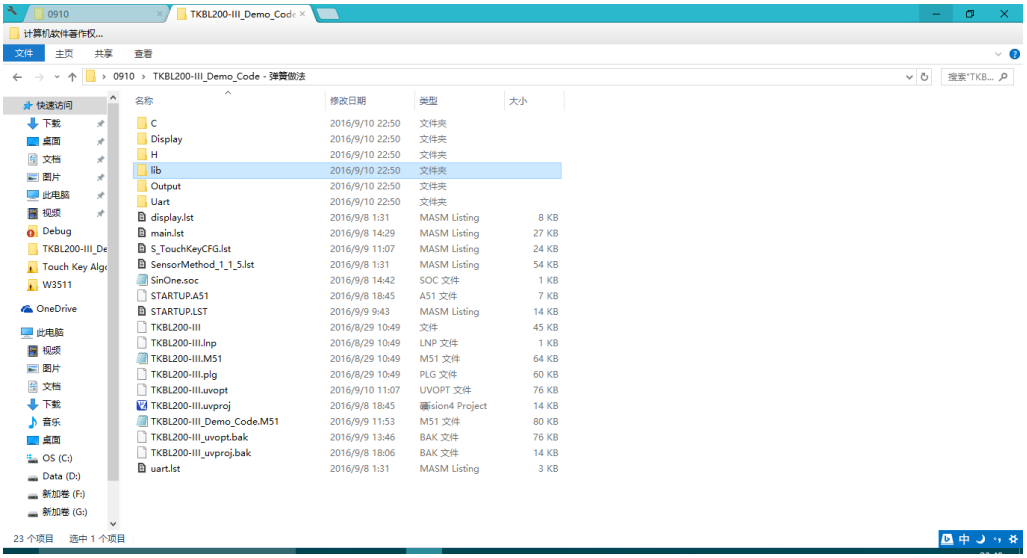
数据位		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
含义		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	触控按键状态（1：有效；0：无效）							

数据位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
含义	TK23	TK22	TK21	TK20	TK19	TK18	TK17	TK16
	触控按键状态（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
	触控按键状态（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
	触控按键状态（1：有效；0：无效）							

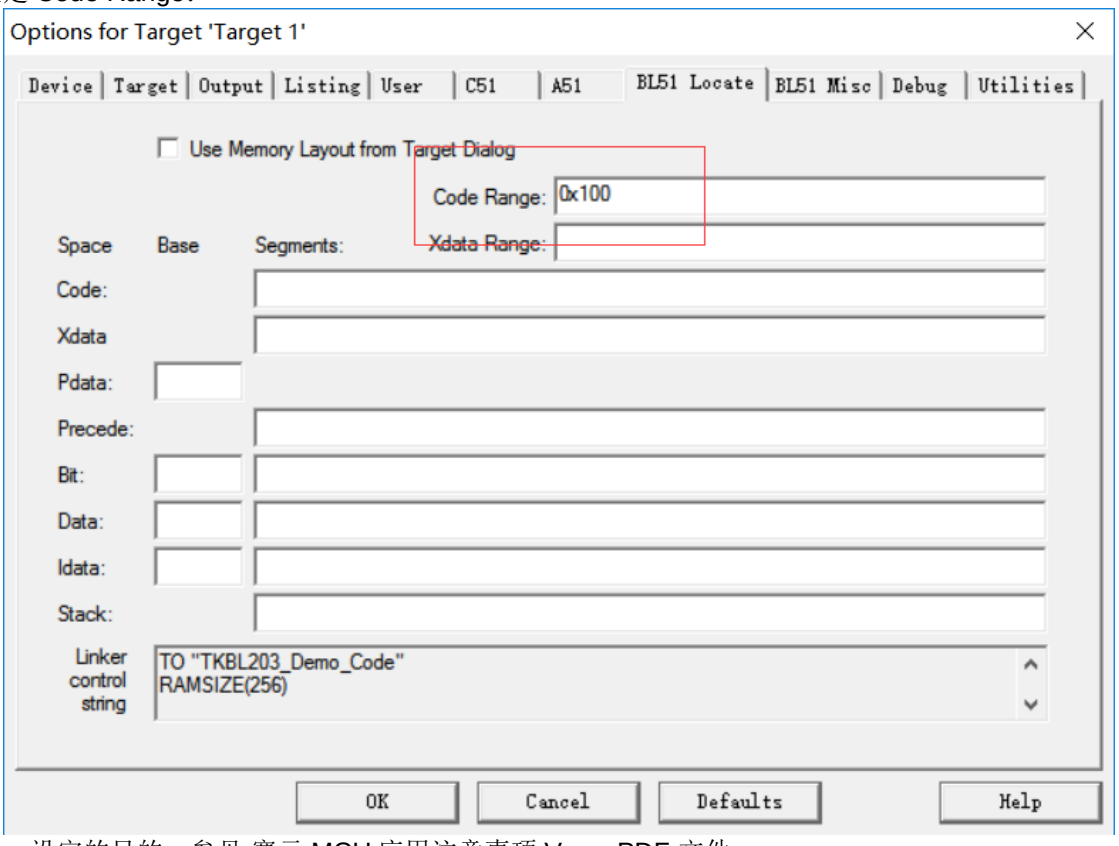
注：SC92F8X7X 函数的返回类型是 unsigned int，其他均为 unsigned long int ;TKn 为触控通道，具体请参照对应规格书。

6、打开工程项目文件复制“lib”文件夹至工程文件夹内。



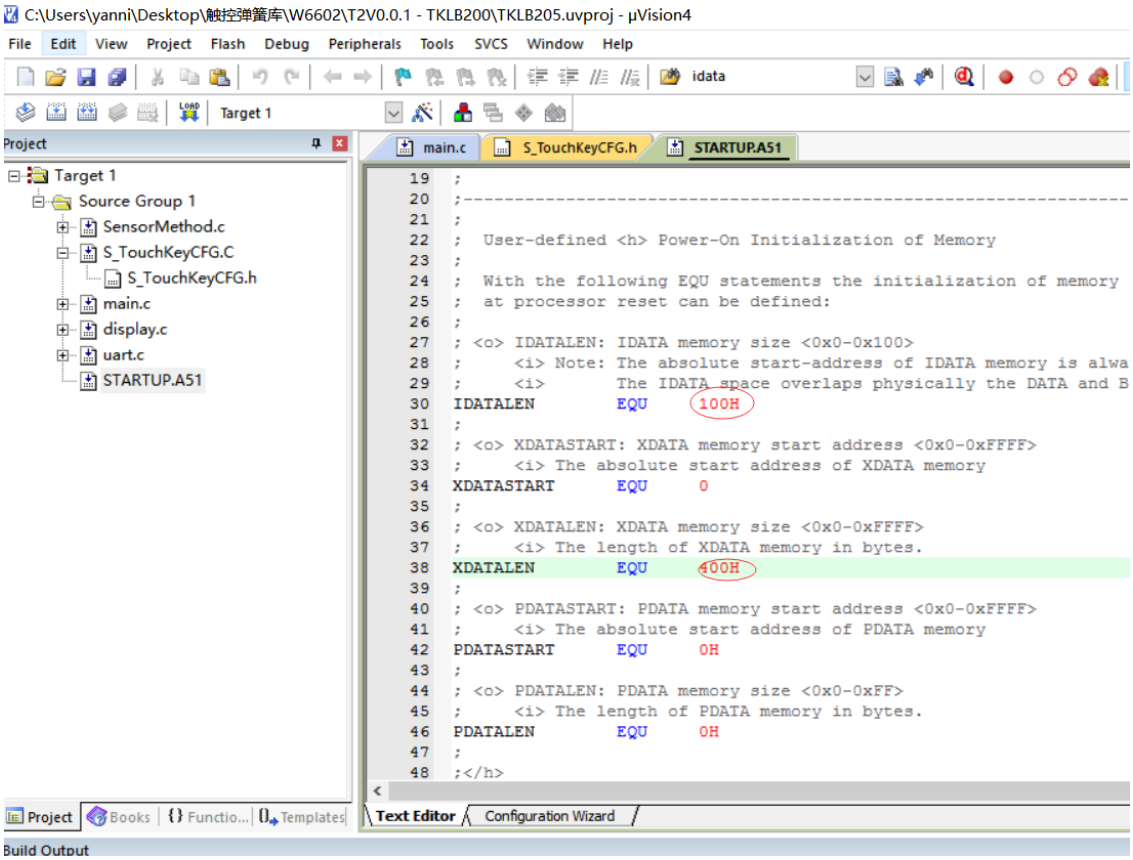
7、在 Keil 中打开工程项目文件；注意设定（Code range）、设定（XDATALEN EQU xxxH）；

设定 Code Range：



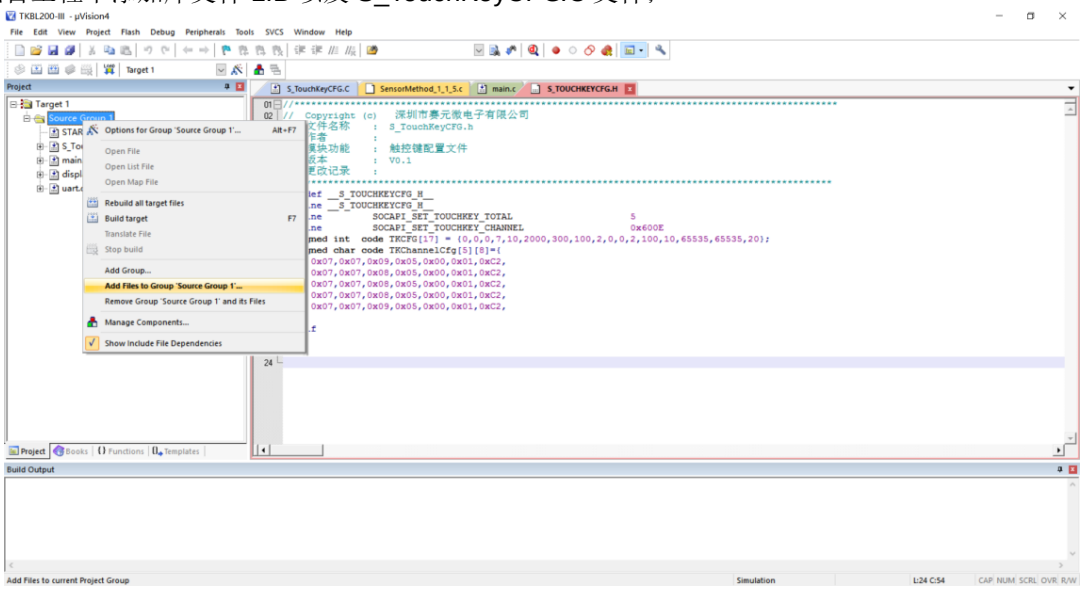
设定的目的，参见 赛元 MCU 应用注意事项 Vx.xx.PDF 文件。

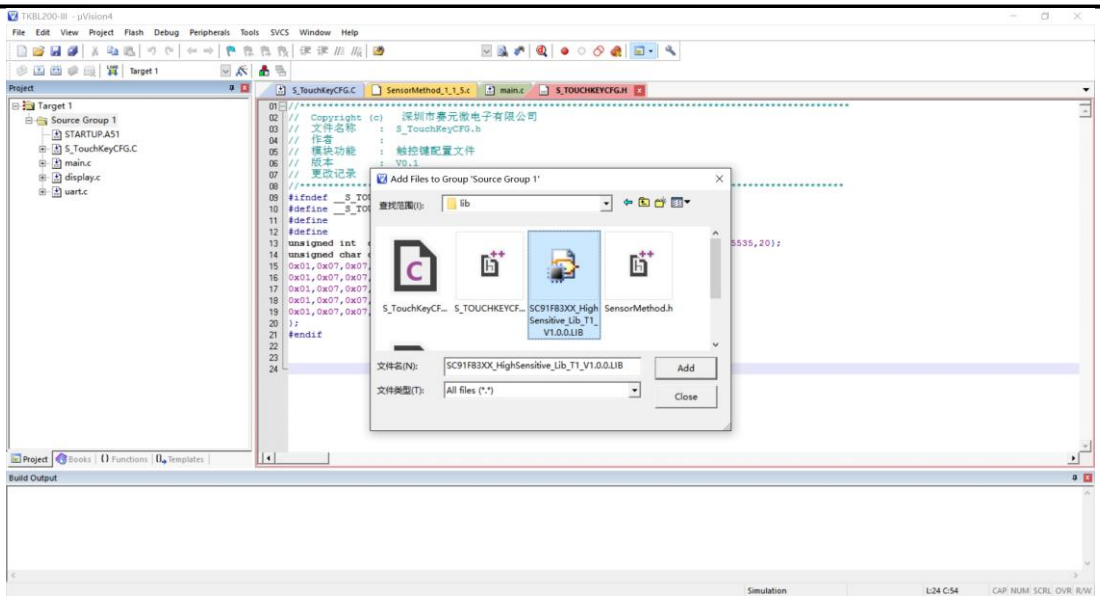
8、设定 XDATALEN：

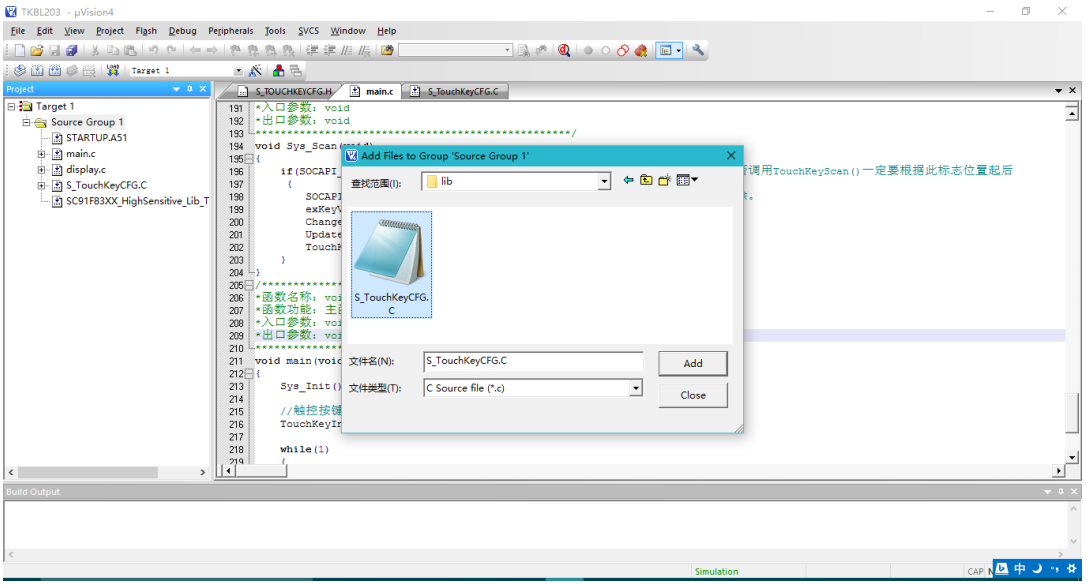


用于 STARTUP.A51 中，清掉外部 XData，具体型号的 XData 大小见规格书。

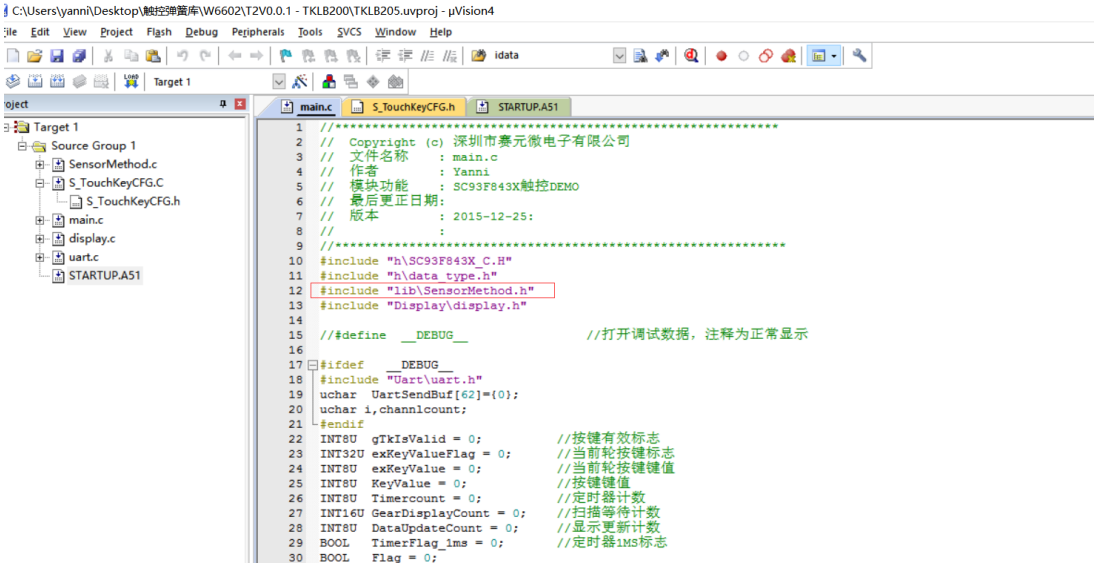
9、在项目工程中添加库文件 LIB 以及 S_TouchKeyCFG.C 文件；



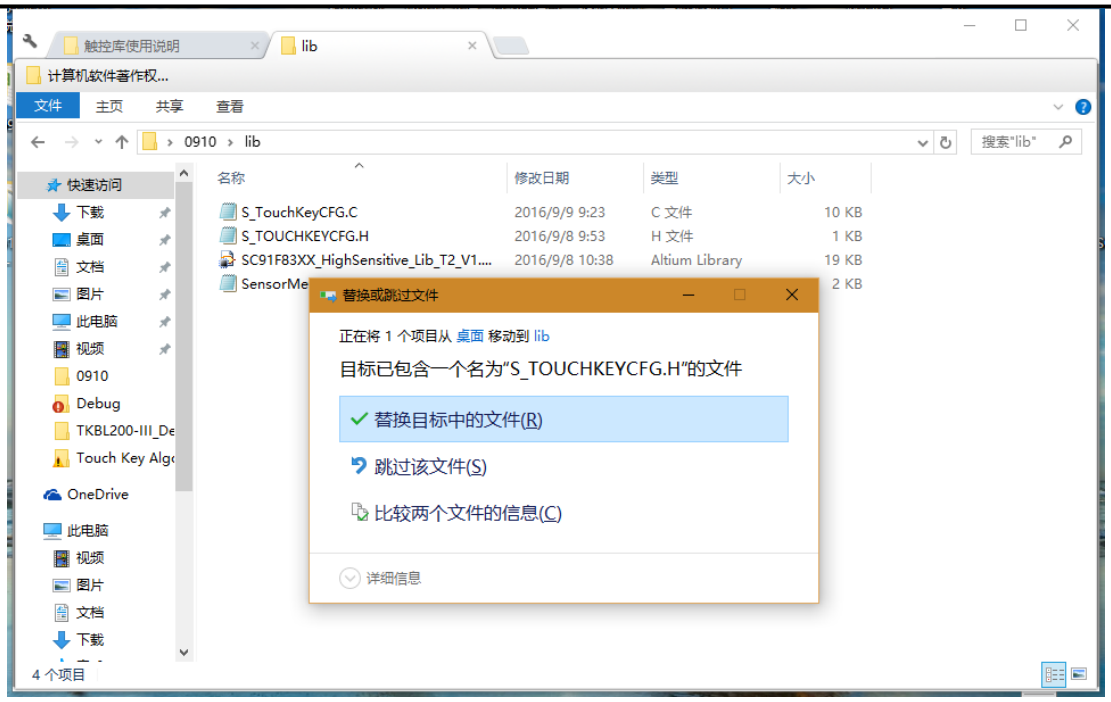




10、在主程序文件中添加头文件引用；



11、将生成的配置文件 S_TOUCHKEYCFG.H 替换到 LIB 文件夹内。



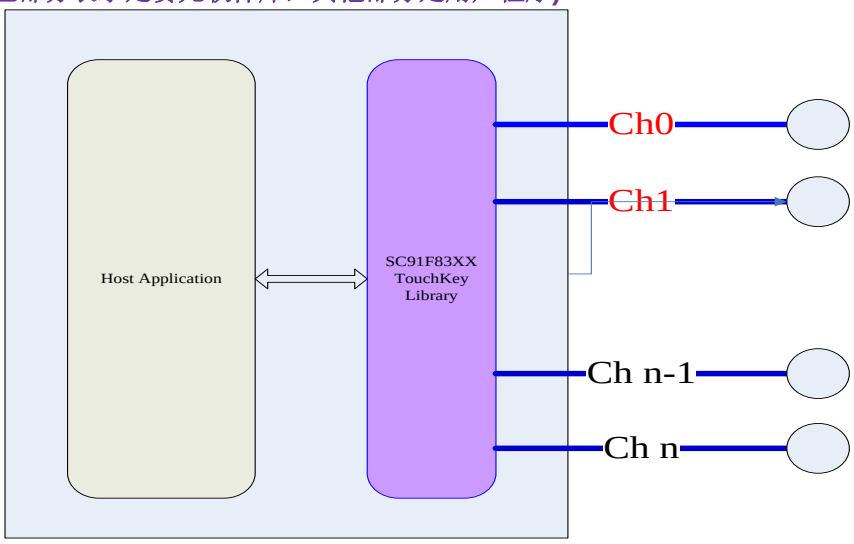
至此，就将触控库添加进项目工程中。

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。

3.2.3 完成用户程序和赛元触控软件库的融合

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、**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

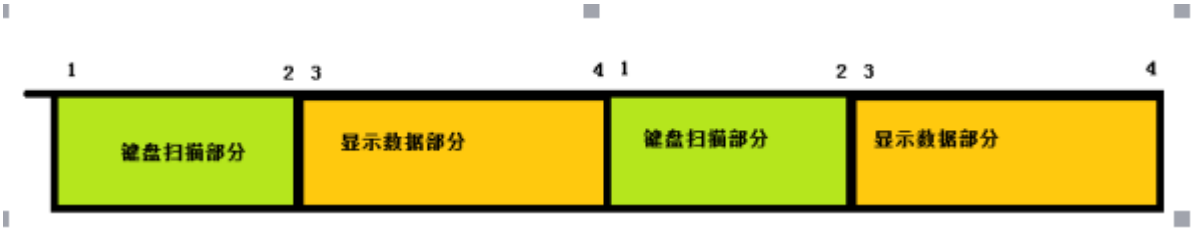
② 主程序调用接口函数"TouchKeyInit()"用于配置触控按键通道的参数，并初始化 Baseline 基线；

③ 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。

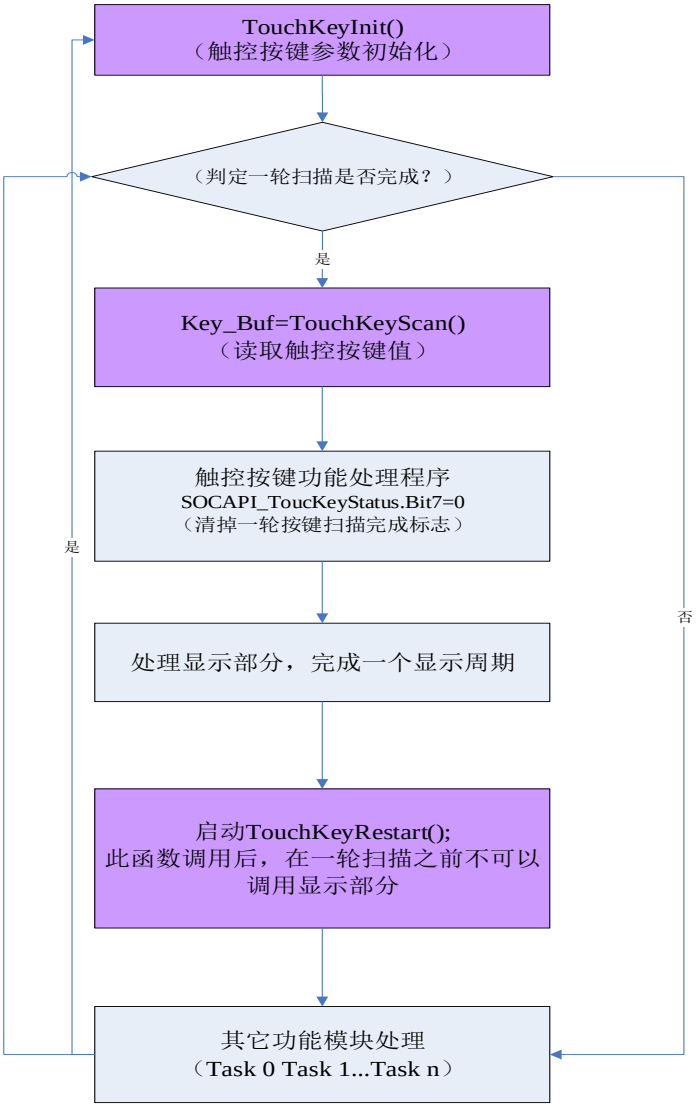
④ 主程序通过查看全局变量 SOCAPI_TouchKeyStatus&0x80 来判定一轮触控按键扫描是否完成；

⑤ 主程序调用接口函数"TouchKeyScan()"用于读取触控按键值；

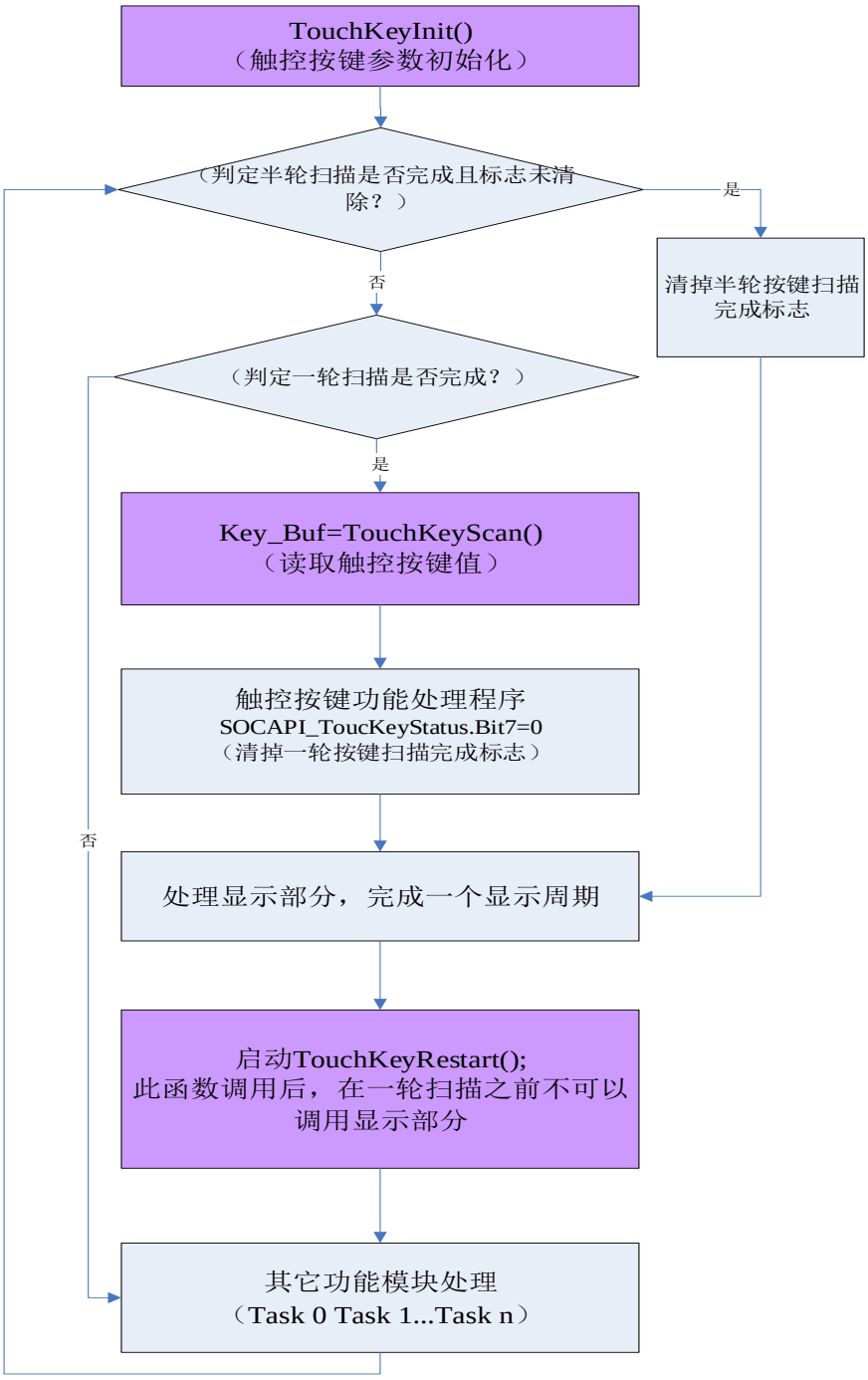
⑥ 特别需要强调一点的是：调用 TouchKeyRestart()开始扫描按键时后，一轮扫描或者半轮扫描的标志还没有出现时， 一定不要去做显示数据的部分。



（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，在一轮或者半轮按键扫描未完成之前，不能对触控按键通道进行操作；如改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；

4、软件融合的注意事项

① 运行时间：

接口函数；

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：(240*N 个按键)us

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。

3.2.4 库使用注意事项

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。

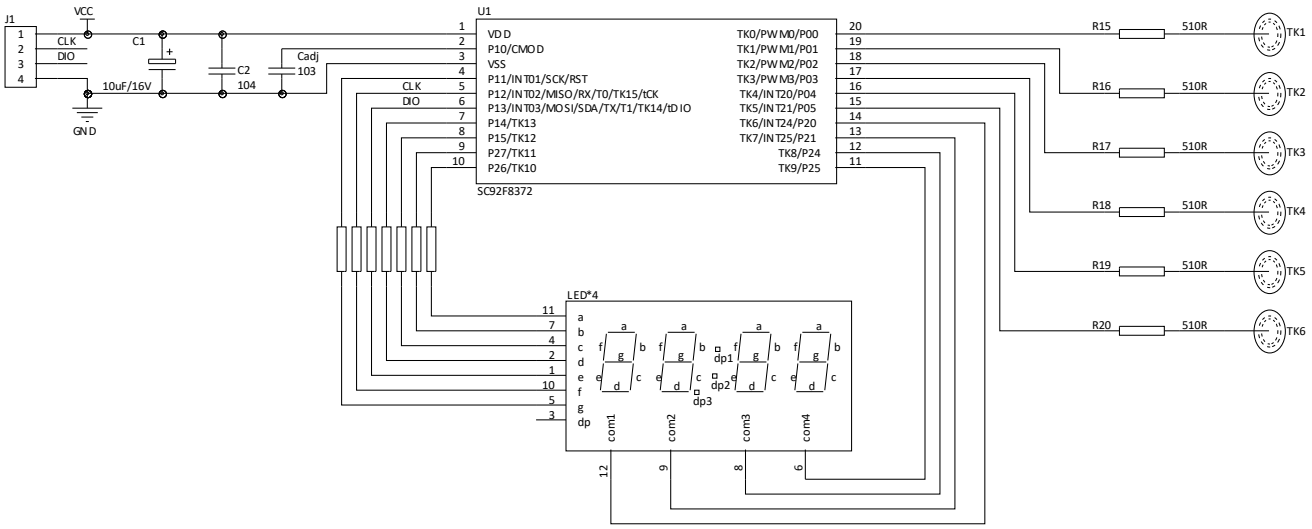
4、感应盘与感应盘之间应该尽量留一定的间距，以保证手指头触控时不会覆盖到 2 个感应盘，同时也能防止感应盘寄生电容过大。

5、基准电容 C 电容建议使用温度系数小精度高的电容，以免造成灵敏度不一致或随温度变化而变化。

6、TK 对应的 IO 口设置为强推挽输出高。

附录

一、应用参考原理图（以 SC92F8372 为例）



二、程序调用例程

8 个以上按键处理例程：

Main.c

```
/*****************************************************
*函数名称：void  Sys_Scan(void)
*函数功能：扫描 TK 和显示
*入口参数：void
*出口参数：void
*****************************************************/
void Sys_Scan(void)
{

                                    if(TKScanflag==1)
        {
                                  if(SOCAPI_TouchKeyStatus&0x40)
                                 {
                                              SOCAPI_TouchKeyStatus&=0xbf;
                                              ComOutState;
                                             TKScanflag=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
    TKScanflag=0;
                                        ComOutState;
                                            TimerFlag_1ms=0;
                                        return;


                               }
        }


 if(TKScanflag==0)
        {
           DisplayData();
            if(isLcdComflag==0)
           {
                        SOCAPI_TouchKeyStatus &=0x7f; //重要步骤 3: 清除标志位， 需要外部清除。

                        ComAllClose;
                       TouchKeyRestart();                          //启动下一轮转换
                       TKScanflag=1;
           }
        }
}

/*****************************************************
*函数名称：void main(void)
*函数功能：主函数
*入口参数：void
*出口参数：void
*****************************************************/
void main(void)
{
          Sys_Init();
          while(1)
        {
              WDTCR |= 0x10;
                        TimerFlag_1ms=0;
                        if(TimerFlag_1ms==1)
                              {
          TimerFlag_1ms=0;

          Sys_Scan();
```

```
            BuzzerWork();
        }
}
```

8 个按键即以下处理例程：

Main.c

```
/*****************************************************
*函数名称：void  Sys_Scan(void)
*函数功能：扫描 TK 和显示
*入口参数：void
*出口参数：void
*****************************************************/
void Sys_Scan(void)
{

                if(TKScanflag==1)
                {
                        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
                                               TKScanflag=0;
                                               ComOutState;
                                              TimerFlag_1ms=0;
                                              return;


                        }
                }


    if(TKScanflag==0)
                {
                     DisplayData();
                       if(isLcdComflag==0)
                       {
                                        SOCAPI_TouchKeyStatus &=0x7f;        //重要步骤 3: 清除标志位， 需要外部清除。

                                        ComAllClose;
                                       TouchKeyRestart();                    //启动下一轮转换
                                       TKScanflag=1;
                         }
                }
}

/*****************************************************
*函数名称：void main(void)
*函数功能：主函数
*入口参数：void
*出口参数：void
*****************************************************/
void main(void)
{
                    Sys_Init();
                while(1)
                {
                        WDTCR |= 0x10;
                                if(TimerFlag_1ms==1)
        {
          TimerFlag_1ms=0;

          Sys_Scan();

          BuzzerWork();
        }

                    }
}
```

Display.c

```
/*************************************************
*函数名称：void  DisplayData(void)
*函数功能：数码管显示函数
*入口参数：void
*出口参数：void
*************************************************/
void  DisplayData(void)
{

    ComAllClose;
        if(isLcdComflag == 0)                                   //显示 COM0  数据
        {
                    LedSetSegData(gIsLedDataBuf[0]);
                COM0 = 0;
                    isLcdComflag = 1;
        }
        else if(isLcdComflag ==1)                               //显示 COM1 数据
        {
                    LedSetSegData(gIsLedDataBuf[1]);
                    COM1 = 0;
                isLcdComflag = 2;
            }
            else if(isLcdComflag ==2)
        {
             isLcdComflag = 0;
            ComAllClose;
        }
}
```

4 SC92F8XXX_HIGHRELIABILITY_LIB_T1 库说明

4.1 前言

4.1.1 赛元触控库文件介绍

赛元 Stouch MCU SC92F8XXX 提供一个可以供用户调用的高可靠库文件，以降低用户触控按键部分的开发难度。用户仅仅需要经过以下几个步骤，便可实现触控按键的功能，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。

SC92F8XXX_HighReliability_Lib_T1_Vx.x.x 的使用分以下几个步骤：

1、 **安装开发工具并配置参数、导出数据。**

赛元提供了专门的触控按键电脑界面软件 SOC Touch Key Tool Vx.xx，方便用户能直观地看到触控按键的数据和图形，用户需要安装此软件，并配合 DPT52 或者 SC LINK 使用。用户可通过软件界面配置参数（TKCFG1、TKCFG2 和 TKCFG3）来找到用户 PCB 最合适的触控按键关键参数，并将最终的相关数据导出 EXCEL 表格并保存。

2、 **实现赛元软件库的功能测试。**

用步骤 1 中提取的参数来修改赛元软件库中 S_TouchKeyCFG.h 头文件中的参数

3、 **完成用户程序和赛元触控软件库的融合。**

用户自行写好除触控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。

4.1.2 赛元触控原理介绍

赛元触控按键 Stouch MCU 系列 IC 内部集成了一个电容触控按键模块。此模块利用电荷转移原理，具体的工作原理为：模块外接一个基准的高精度大电容 Cadj（102~473 不等），系统内通过一个模块通过外部触控按键的等效电容 Cx 给 Cadj 充电，系统记录充放电到某一个具体数值的时间来判断 Cx 是否有变化。如果有触控按键时 Cx 变大，则此充放电时间会变小，用户可通过算法来准确地判断是否有按键被按下。

4.2 调试流程概括提纲

为使用户能概略了解用赛元触控软件库来完成触控按键部分的调试，特给提纲如下，用户可在第三部分看到详细的操作截图和流程说明。

4.2.1 安装开发工具并配置参数、导出数据

1、 安装调试工具及连接硬件；
   ① Setup SOC Pro51；
   ② Setup SOC TouchKey Tool Vx.xx;
   ③ 升级 DPT52 或者 SC LINK 固件,更新 MCU 库；
   ④ 安装 SOC_KEIL Vx.xx 插件；
   ⑤ 硬件连接。
2、 准备 Lib 文件
   ⑩ 一个库文件：SC92F8XXX_HighReliability_Lib_T1_Vx.x.x.lib；
   ⑪ 二个头文件：S_TouchKeyCFG.h,SensorMethod.h；
   ⑫ 一个 C 文件：S_TouchKeyCFG.c。
3、 提取参数
   ① 理论数据的采集方式（用电脑的 USB 电源供电），此时数据的 Noise 较实际使用时要小，可供参考使用：
      a. 配置调试软件 SOC Touch Key Tool 界面中的寄存器：通道选择（将用作 TK 的通道打勾）；配置 TKCFG1 中的 PRS；配置 TKCFG2 中的 CTIME、CMPFLTS；配置 TKCFG3 中的 SVSS、CMPVREFS；点击界面的“保存调试项目”按钮来保存此配置；
      b. 开始调试并读取导出数据；
      c. 配置 TKCFG1、TKCFG2 和 TKCFG3 让所有通道的 RAW DATA 值达到目标值, 不共用项目目标值一般定 3000~9000，对于共用项目，目标值配置在 9000~13000。
      d. 配置 TKCFG1、TKCFG2 和 TKCFG3 让所有通道信噪比都符合要求值（建议 SNR>5）；
      e. 导出 EXCEL 格式的原始数据 RAW DATA（每一通道的有触控按键值和无触摸按键值）；
      f. 人工记录 TKCFG1、TKCFG2 和 TKCFG3 的设置值，供后续修改库文件设置使用；
   ② 实际数据采集方式（用实际产品电源供电，实际产品电源与电脑电源，两者必须有一端接隔离变压器），此数据接近于实际的数据，特别是信噪比更加准确；
      a. 配置调试软件 SOC Touch Key Tool 界面中的寄存器：通道选择（将用作 TK 的通道打勾）；配置 TKCFG1 中的 PRS；配置 TKCFG2 中的 CTIME、CMPFLTS；配置 TKCFG3 中的 SVSS、CMPVREFS；点击界面中的“保存调试项目”按钮来保存此配置；
      b. 开始调试并读取导出数据；
      c. 配置 TKCFG1、TKCFG2 和 TKCFG3 让所有通道的 RAW DATA 值达到目标值, 不共用项目目标值一般定 3000~9000，对于共用项目，目标值配置在 9000~13000。
      d. 配置 TKCFG1、TKCFG2 和 TKCFG3 让所有通道信噪比都符合要求值（建议 SNR>5）；
      e. 导出 EXCEL 格式的原始数据 RAW DATA（每一通道的有触控按键值和无触摸按键值）；
      f. 记录 TKCFG1、TKCFG2 和 TKCFG3 的设置值，供后续修改库文件设置使用；

4.2.2 实现赛元软件库的功能测试

1、 SC92F8XXX_HighReliability_Lib_T1_Vx.x.x 库文件介绍
2、 Lib 用到的资源：
   ① 数据类型；
   ② RAM；
   ③ ROM；
   ④ 中断；
3、 Lib API 接口函数的调用说明；
4、 LIB API 接口函数的返回值说明及代码例程；
5、 在 Keil 中打开工程项目文件 STARTUP.A51；注意设定（Code range）、设定（XDATALEN EQU 80H）；
6、 添加库文件 LIB；
7、 在 S_TouchKeyCFG.h 中修改参数（上面步骤中记录的 TKCFG1、TKCFG2 和 TKCFG3 的参数）；
8、 在 S_TouchKeyCFG.h 中设定触控按键的通道和个数；设定触控按键有效的次数；
9、 根据调试步骤中的 RAW DATA 初步设定噪声阈值和手指阈值，在 S_TouchKeyCFG.h 中修改。

4.2.3 完成用户程序和赛元触控软件库的融合

1、 主程序和库文件的整体结构关系
2、 库文件的调用流程
3、 主程序和库文件的时序关系
4、 软件融合的注意事项
5、 整体 Code 的测试

4.3 赛元触控软件库的详细使用说明

4.3.1 安装开发工具并配置参数、导出数据

1、 安装调试工具及连接硬件

① **Setup SOC Pro51**；
安装赛元 Pro51 软件 SOC Pro51 Vx.xx.exe(请从赛元网站找最新版本)。

② **Setup SOC TouchKey Tool**；
安装赛元触控按键调试软件 SOC TouchKey Tool (请从赛元网站找最新版本)。

③ **升级 DPT52 固件,更新 MCU 库**；在线烧写器 DPT52 或者 SC LINK 的固件需升级最新版本固件；SOC Pro51 的 MCU 库文件升级到最新版本。

DPT52 固件升级方法：

a. 到赛元网站（ http://www.socmcu.com ）下载最新的 DPT52 固件文件。
b. 拨下 DPT52 的固件更新开关，通过 USB 连接到电脑，此时 DPT52 上的 USB 指示灯（绿色）闪烁，表明已经进入固件升级模式。
c.打开 SOC Pro51 软件，点击“升级”菜单下的“升级固件”。
d.在“打开文件”对话框中找到固件文件（.iap 文件），并点击打开。
e.弹出对话框显示当前版本，和要更新的版本，点击“确定”按钮进行更新。
f.更新完成后，请断开 USB 连接，重新连接 USB 后才能正常使用。

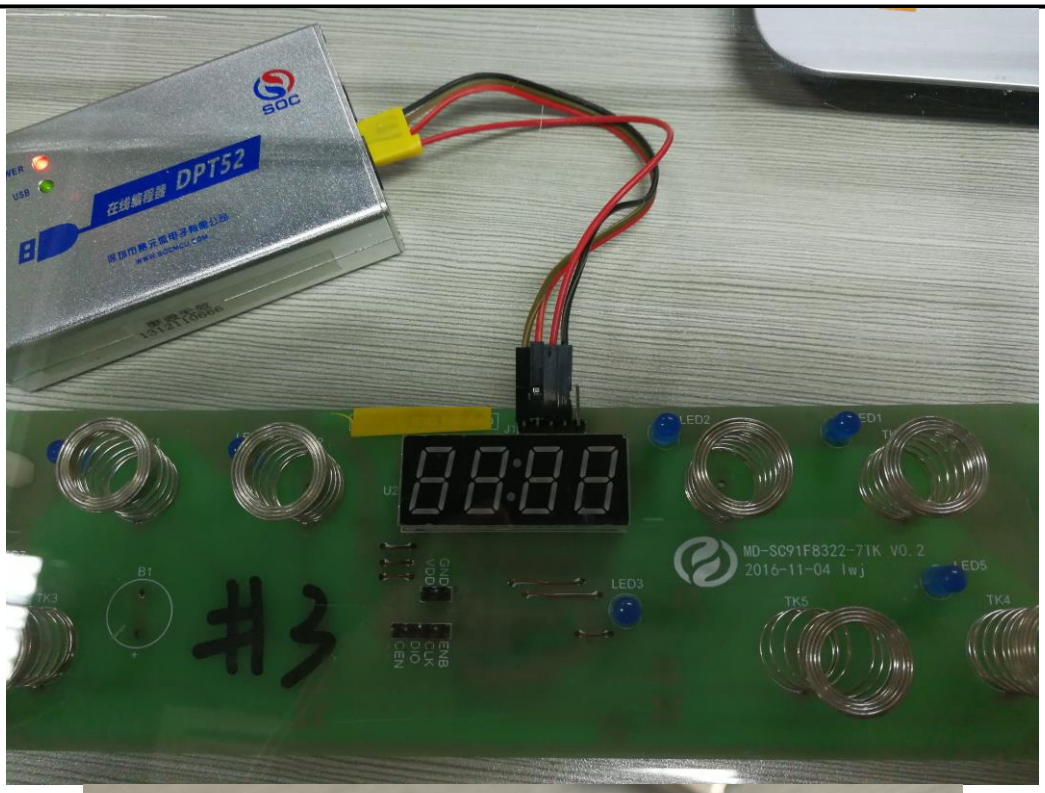
更新 MCU 库文件的方法：

a. 到赛元网站（http://www.socmcu.com）下载最新的 MCU 库文件。
b. 打开 SOC Pro51 软件，点击“升级”菜单下的“更新 MCU 库文件”。
c. 在“打开文件”对话框中找到 MCU 库文件（.mcu 文件），并点击打开。
d. 更新完成，重新启动 SOC Pro51 软件。

④ **安装 SOC_KEIL Vx.xx 插件**；请将赛元 MCU 的插件安装文件版本更新到最新版本。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 安装目录下的 SinOne_Chip 目录中。
b.SinOne_Chip 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
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(VCC/GND/CLK/DIO)-->用户 PCB(VCC/GND/CLK/DIO)；由于调试过程中需用到 IC 的硬件 SSI 的 UART 资源，请客户在 PCB 板上预留接线，并且在程序中不要使用 SSI 功能或者将 SSI 口用于其他功能，待调试完成后用户便可正常使用 SSI。**注意:LVR 设置必须比供电电压低,如供电为 3.3V,则 Option 中 LVR 必须选择 3.3V 以下的档位**。硬件连接见下图





2、准备 Lib 文件

① 一个库文件：SC92F8XXX_HighReliability_Lib_T1_VX.X.X.lib；

② 二个头文件：S_TouchKeyCFG.h,SensorMethod.h；

③ 一个 C 文件：S_TouchKeyCFG.c。

3、提取参数

理论数据的采集方式（用电脑的 USB 电源供电），此时数据的 Noise 较实际使用时要小，可供参考使用。本部分省略实际数据采集内容，所有步骤跟理论数据采集方式相同。

① 配置调试软件 SOC Touch Key Tool 界面中的寄存器：

先选择 IC 型号，然后点击“开启触摸通道选择”按钮，勾选实际使用的通道，点击“确定”； 配置 TKCFG1 中的 PRS；配置 TKCFG2 中的 CTIME、CMPFLTS；配置 TKCFG3 中的 SVSS、CMPVREFS；点击界面的“保存调试项目”按钮来保存此配置。

赛元建议优先先按缺省的配置来测试：

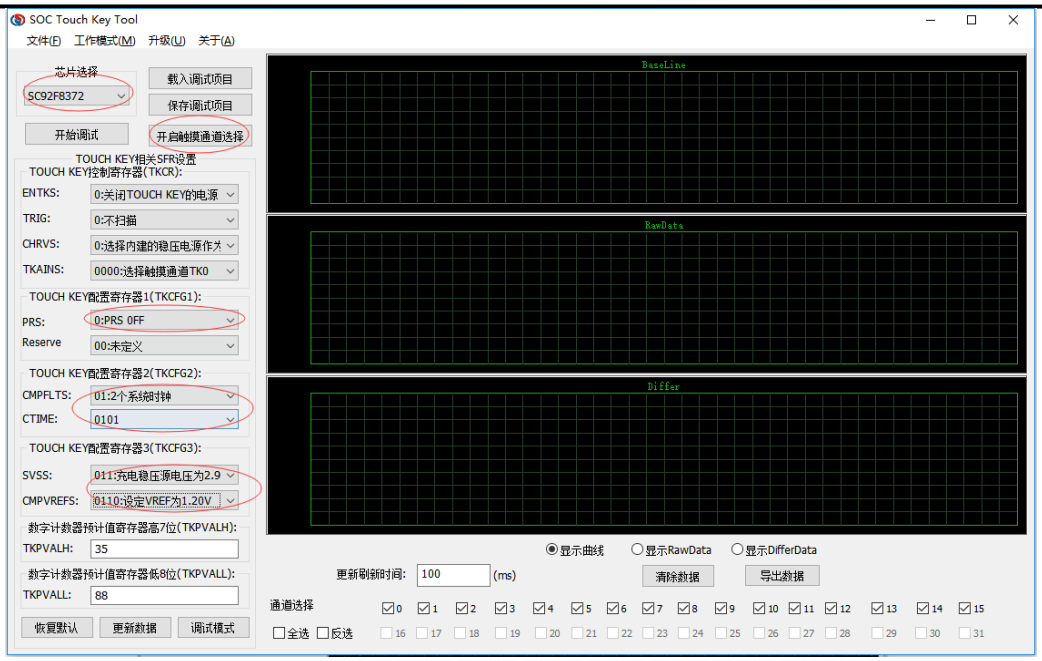
a. PRS：设为 0；PRS OFF

b. CTIME：0101b=5；建议范围：1~F；(SC92F8X4XB 系列建议设置为 3)

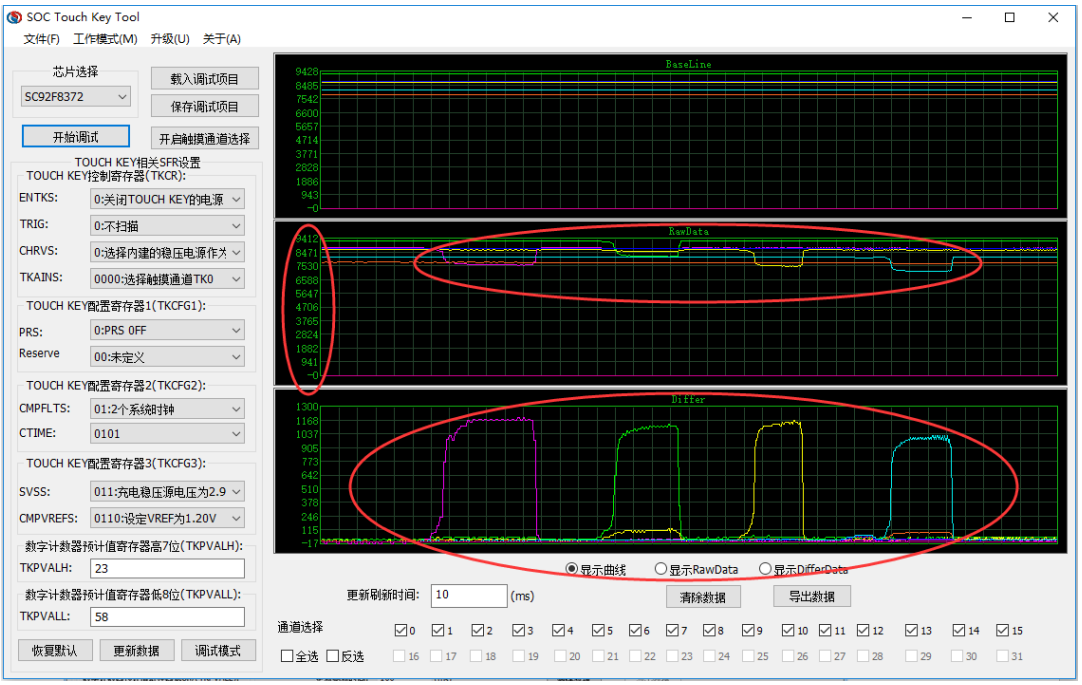
c. CMPFLTS：设为 01,两个时钟滤波；建议范围：1~3；

d. SVSS：设为 2.90V；建议范围：2V~4.2V

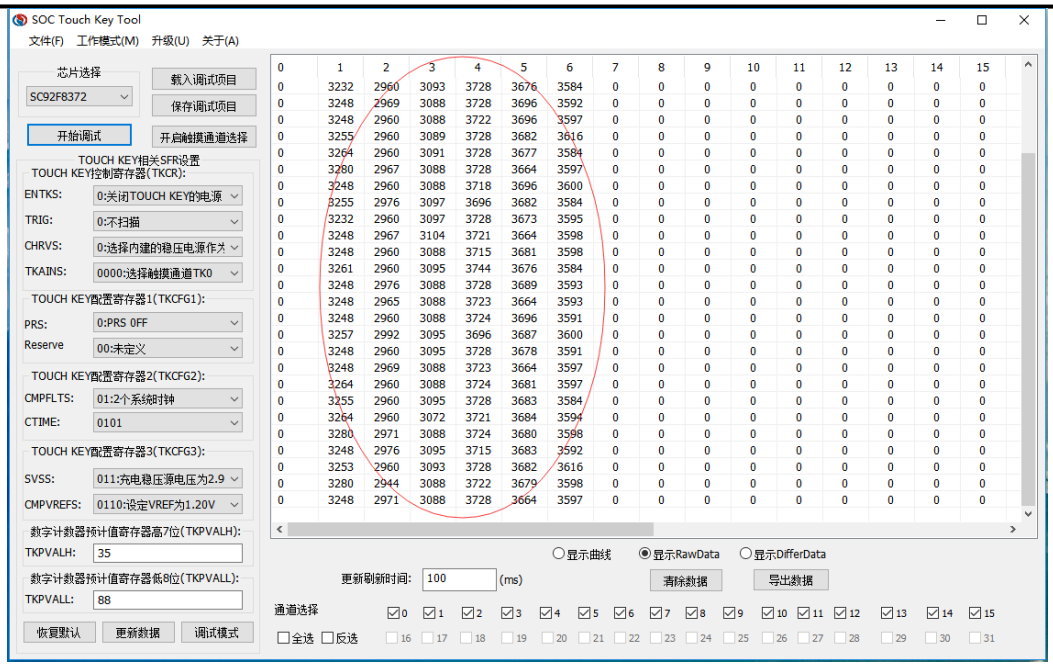
e. CMPVREFS：设为 1.20V；建议范围：0.6V~2.1V（SVSS-CMPVREFS 要大于 0.7V）



② 开始调试并读取导出数据：单击“开始调试”按钮，可在图形模式下看到所选中通道出现不同颜色的波形，如果对应通道有按键被按下，则对应通道的波形幅度会有相应的变化，如下图：



用户可以点击选择“显示 RawData”切换到数据模式，便可看到每个通道对应的原始数据值，此数据就是 IC 的 TKCNT 的真实值。如下图：



图中所圈部分即是选中的 TK1~TK7 的 7 个通道的 TKCNT 值，每一列代表一个通道，没有选中的通道 TKCNT 值为 0。如果出现特殊情况，有可能是数据太大，超出了 TKCNT 的范围，请检查 Cadj 的电容是否接错，或者减小此电容。

③ 配置 TKCFG2 和 TKCFG3 让所有通道的 RAW DATA 值达到目标值, 不共用项目目标值一般定 3000~9000，对于共用项目，目标值配置在 9000~13000。

建议用户先按照原始配置的建议参数值来看此时的 RAW DATA 是否符合要求，如果不符合要求则按下述方式进行调整（注意，下述调节中如果完成第一步就 OK，请停止修改其它参数；不行再修改下一步；依次类推）：

a. 如果所有通道 RAWDATA 数据过小

i. 可增加 SOCAPI_SET_TKCFG2 中的 CTIME (建议 CTIME： 1~F)；
ii. 可减小 SOCAPI_SET_SET_TKCFG3 中的 SVSS（建议 SVSS：2.0~4.2V）；
iii. 可增大 SOCAPI_SET_SET_TKCFG3 中的 CMPVREFS（建议 CMPVREFS：0.6~2.1V）；
iv. SVSS 与 CMPVREFS 之间的关系：SVSS-CMPVREFS>0.7V；
v. 可加大电容（建议 223 以下）；

b. 如果所有通道 RAWDATA 数据过大

i. 可减小 SOCAPI_SET_TKCFG2 中的 CTIME (建议 CTIME：1~F)；
ii. 可增大 SOCAPI_SET_TKCFG3 中的 SVSS（建议 SVSS：2.0~4.2V）；
iii. 可减小 SOCAPI_SET_TKCFG3 中的 CMPVREFS（建议 CMPVREFS：0.6~2.1V）；
iv. SVSS 与 CMPVREFS 之间的关系：SVSS-CMPVREFS>0.7V；
v. 可减小电容（建议 222 以上）；

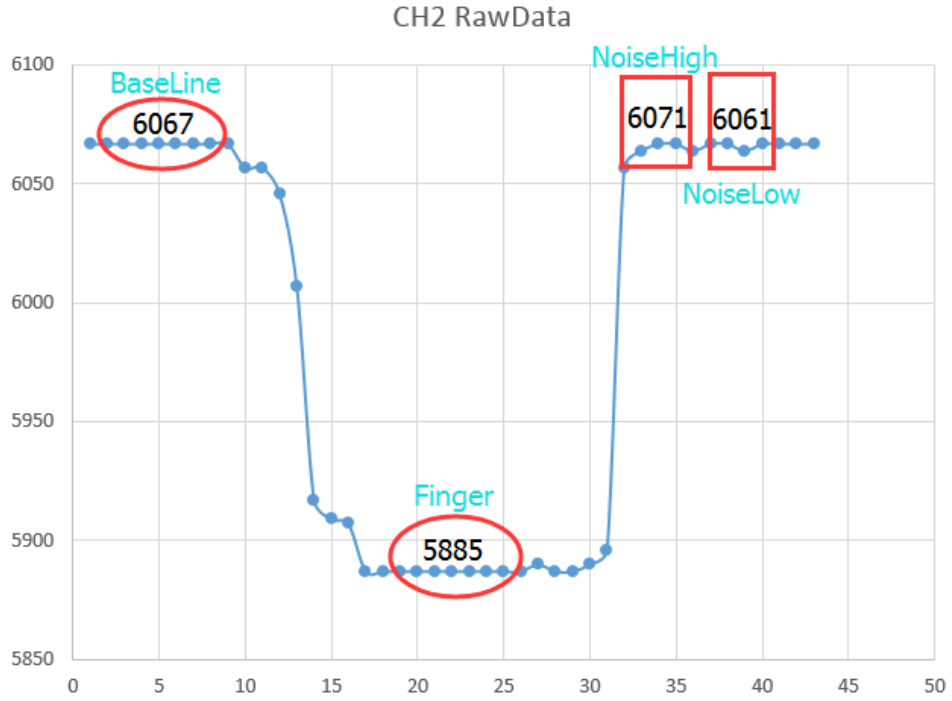
c. 如不同触控按键通道的 RAW DATA 值差异太大

i. 可能的原因是该 PCB 走线长度及外围耦合电容有较大差异；要设法排除；
ii. 再通过上述方式调整参数；
iii. 若差异太大无法调整，则考虑修改 PCB；

d. 其它对触控按键灵敏度的影响：

i. 覆盖物介质厚度：①介质越厚 RAWDATA 值越小；②介质越厚触控按键灵敏度越低；
ii. 弹簧直径与覆盖物介质接触面①弹簧直径越大 RAWDATA 值越小；②弹簧直径越大触控按键灵敏度越高；

④ 配置 TKCFG2 和 TKCFG3 让所有通道信噪比到符合要求值（建议 SNR>5）；下图为将原始数据 RAW DATA 导出成为 EXCEL 表格后（参考下一步导出数据），选择一个通道的数据，在 EXCEL 中做出的图形。



如图所示为一个通道手指按下的波形图，从中我们能取到几个有用的值：

Noise High：没有按下时 RAW DATA 的最大值；

Noise Low：没有按下时 RAW DATA 的最小值；

Baseline：没有按下时的 RAW DATA 的平均值；

Finger：手指按下时 RAW DATA 的平均值

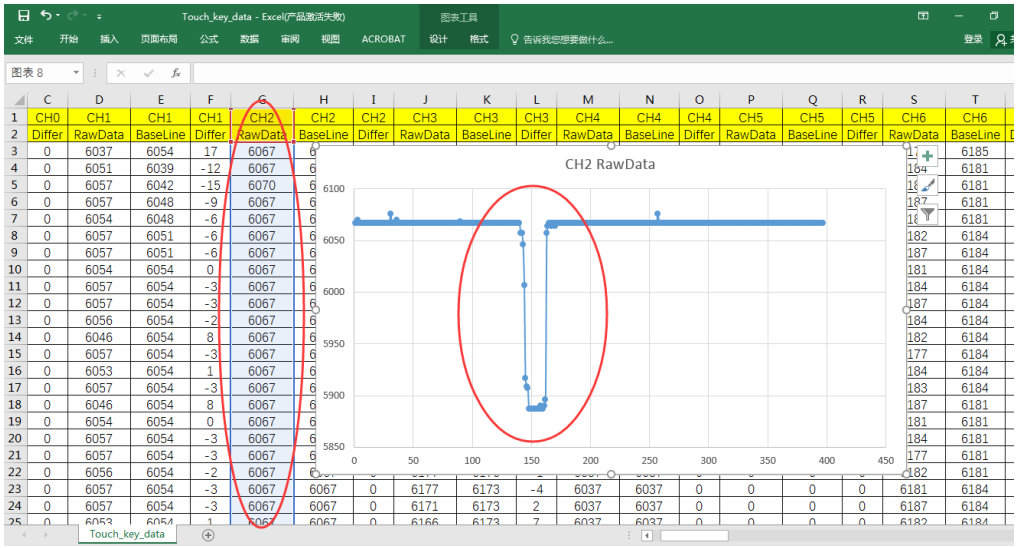
信噪比 SNR 计算方法：

SNR=（Baseline-Finger)÷（NoiseHigh-NoiseLow）

如上图中：SNR=（6067-5885）÷（6071-6061）=18；

调试方式参考上一步，配置 TKCFG2 和 TKCFG3 的参数，让所有通道的 RAWDATA 值达到目标值，并且信噪比 SNR 符合要求（SNR 大于 5）。

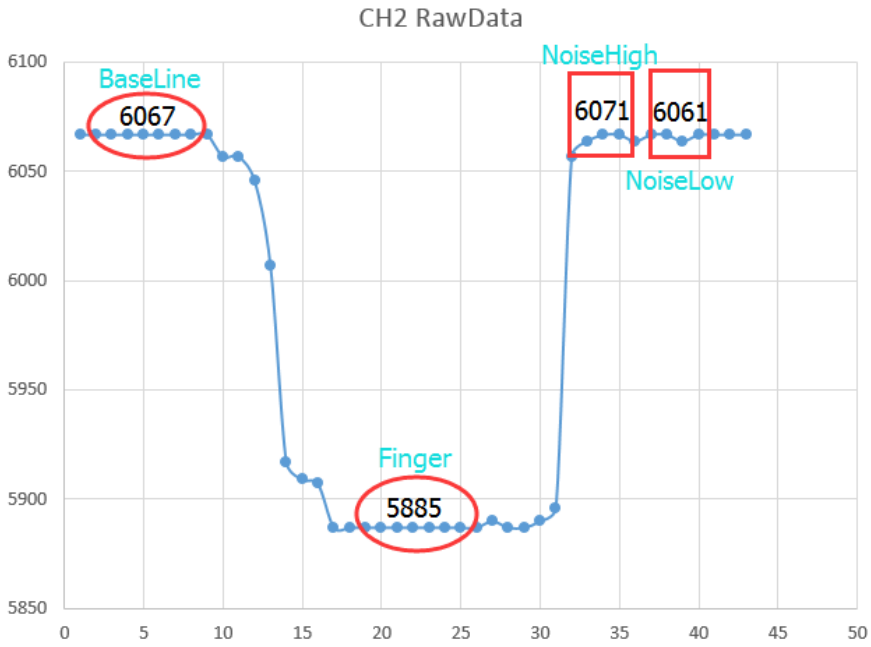
⑤ 导出 EXCEL 表格的原始数据 RAW DATA（每一通道的有触控按键值和无触摸按键值）；



用户此时需要记录下参数配置和重要数据：

①每个通道不同的数据：包括 Baseline、Finger、NoiseHigh、NoiseLow；

②所有通道共同的参数为：寄存器 TKCFG2 中 CMPFLTS 和 CTIME 的值,寄存器 TKCFG3 中 SVSS 和 CMPVREFS 的值；



（请记录每个通道的典型值）

4.3.2 实现赛元软件库的功能测试

SC92F8XXX_HighReliability_Lib_T1_Vx.x.x 文件介绍：

1、 库文件介绍

文件	用途	说明
SC92F8XXX_HighReliability_Lib_T1_Vx.x.x.lib	库文件，实现触控按键检测算法	
SensorMethod.h	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
S_TouchKeyCFG.c	c 文件，提供按键检测等函数	用户不需要修改
S_TouchKeyCFG.h	头文件，提供 TouchKey 功能的宏定义	用户可改变部分宏定义，来改变 TouchKey 寄存器的设置

2、Lib 用到的资源：

① 数据类型；

参数	类型	数值	说明
SOCAPI _SET_TKCFG1	无符号 8 位整型常量	建议使用默认值	控制寄存器 TKCFG1 描述
SOCAPI _SET_TKCFG2	无符号 8 位整形常量	建议 CTIME = 0x03 到 0x0f	控制寄存器 TKCFG2 描述
SOCAPI _SET_TKCFG3	无符号 8 位整形常量	建议使用默认值	控制寄存器 TKCFG3 描述
SOCAPI _SET_TouchKey_Total	无符号 32 位整形常量	1~23	设定触控按键的个数；
SOCAPI _SET_TouchKey_Channel	无符号 32 位整形常量	用户选定触控按键通道而定	选定触控按键通道； Bit0~Bit23 对应 TK0~TK23；1 为 TK 通道；0 为 IO； 0000 0000 0000 0101：TK0 和 TK2 是 TK，其它是 IO 0000 1111 0000 0000：TK8~TK11 是 TK，其它是 IO 以此类推 SOCAPI _SET_TouchKey_Channel 选中了的通道个数需和 SOCAPI _SET_TouchKey_Total 值相等
SOCAPI _SET_TouchKey CONFIRM_CNT	无符号 8 位整型常量	建议：5~40，一般选择 10 次	按键的确认时间。 连续 SOCAPI _SET_TouchKey CONFIRM_CNT 轮均扫描到该按键，才认为是有键按下； 数值大，会导致按键反应变慢；
SOCAPI _SET_NOISE_Threshold	无符号 8 位整型常量	建议:20-40	设置噪音值，
SOCAPI _SET_FINGER_Threshold	无符号 8 位整型常量	根据采集数据设定	设置手指阈值。用 SOC Touch Key Tool 工具采集，用 10mm 直径的铜柱按下后的 diff 值*60%作为手指值， 同时保持设置的手指阈值与设置的噪音值比要大于 5。 数组元素的个数与设定触控按键的个数相匹配

② RAM：

芯片系列	RAM 占用	ROM 占用
SC92F8X6XB_HighReliability_Lib_T1	data 区：18.3 个 byte；Xdata 区：23 个 Byte；与按键的个数无关，均要使用； Xdata 区：每增加一个按键 需要 14 个字节，分别是 N*13byte xdata； 如有 5 个按键： Data 区 18.3 byte, xdata 区 23+13*5 = 88 byte	库使用 ROM 大小约 2.73K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X4XB_HighReliability_Lib_T1	data 区：13.2 个 byte；Xdata 区：23 个 Byte；与按键的个数无关，均要使用； Xdata 区：每增加一个按键 需要 13 个字节，分别是 N*13byte xdata； 如有 5 个按键： Data 区 13.2 byte, xdata 区 23+13*5 = 88 byte	库使用 ROM 大小约 2.73K，且增加或减少若干按键，该大小基本不变，差异不到 200byte
SC92F8X7X_HighReliability_Lib_T1	data 区：15.2 个 byte；Xdata 区：18 个 Byte；与按键的个数无关，均要使用； Xdata 区：每增加一个按键 需要 13 个字节，分别是 4Byte data + 9byte xdata； 如有 5 个按键： Data 区 15.2+4*5 = 35.2 byte, xdata 区 7+9*5 = 52 byte	库使用 ROM 大小约 2.46K，且增加或减少若干按键，该大小基本不变，差异不到 200byte

③ 中断；仅使用 TK 中断，默认优先级。

中断源	中断优先级	中断向量	查询优先级	中断号（C51）	标志清除方式
TK	低	005BH	12	11	H/W Auto

库仅使用了触摸中断，优先级为低，且中断服务程序内无函数嵌套，执行时间约 6.8us

3、 Lib API 接口函数 S_TouchKeyCFG.c 的调用说明；

函数	用途	说明
TouchKeyInit(void)	触摸按键初始化	1,用户在上电复位后调用一次； 2,本函数通过 S_TouchKeyCFG.c 宏参数配置用户选定的按键通道、按键参数并初始化 Baseline 基线； 3，执行本函数用时约：200~500mS;由于该函数执行时间较长，建议用户先完成整个系统初始化后，再来调用该函数。
TouchKeyRestart(void)	使能一轮触控按键	1，用户主程序来控制何时启动按键扫描；

	扫描	2，启动按键扫描后，在一轮触控按键扫描完成之前，不能对触控按键通道进行操作：如改触控按键通道为 IO。否则，触控按键功能将无法实现。
Unsigned long int TouchKeyScan(void)/ Unsigned int TouchKeyScan(void)	触摸按键算法处理	1，用户需要在触控按键一轮扫描完成后调用； 2，如用户未调用该函数，一定不能重新调用 TouchKeyRestart(void) 使能一轮触控按键扫描，否则上一轮数据将被当前数据覆盖； 3，执行该函数用时：算法执行时间会因按键选择个数的增减而增减。 N 个 Key 约(240*N) us@12M。 4, SC92F8X7X 该函数的返回类型是 unsigned int，其他型号该函数的返回类型是 unsigned long int

4、 全局变量 SOCAPI_ToucKeyStatus 的说明

① 全局变量在 S_TouchKeyCFG.c 头文件中声明：

unsigned char xdata SOCAPI_ToucKeyStatus;

SOCAPI_ToucKeyStatus 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_ToucKeyStatus&0x80)时，调用 TouchKeyScan(void) 进行算法数据处理，给出键值；

③ 使能下一轮触控按键扫描之前，一定要清掉标志，SOCAPI_ToucKeyStatus &=0x7f;。

5、 LIB API 接口函数 TouchKeyScan()的返回值说明

TouchKeyScan() 返回值：返回值对应 bit 为 1 即该通道有按键，0 为无按键。

具体如下所示：

数据位		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
含义		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	触控按键状态（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
	触控按键状态（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
	触控按键状态（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
	触控按键状态（1：有效；0：无效）							

注：SC92F8X7X 函数的返回类型是 unsigned int，其他型号的返回类型是为 unsigned long int；TKn 为触控通道，具体请参照对应规格书。

6、 按键有效的最长输出时间

#define SOCAPI_SET_KEY_CONTI_TIME 1000 // 按键有效的最长输出时间，设置范围 0~5000，默认 1000，输出时间= 1000*单位每轮扫描时间（如 10ms）=10S

注意事项：

① TK0~TK31 为触控按键通道；具体细节可参见 SC92F8XXX 规格书。

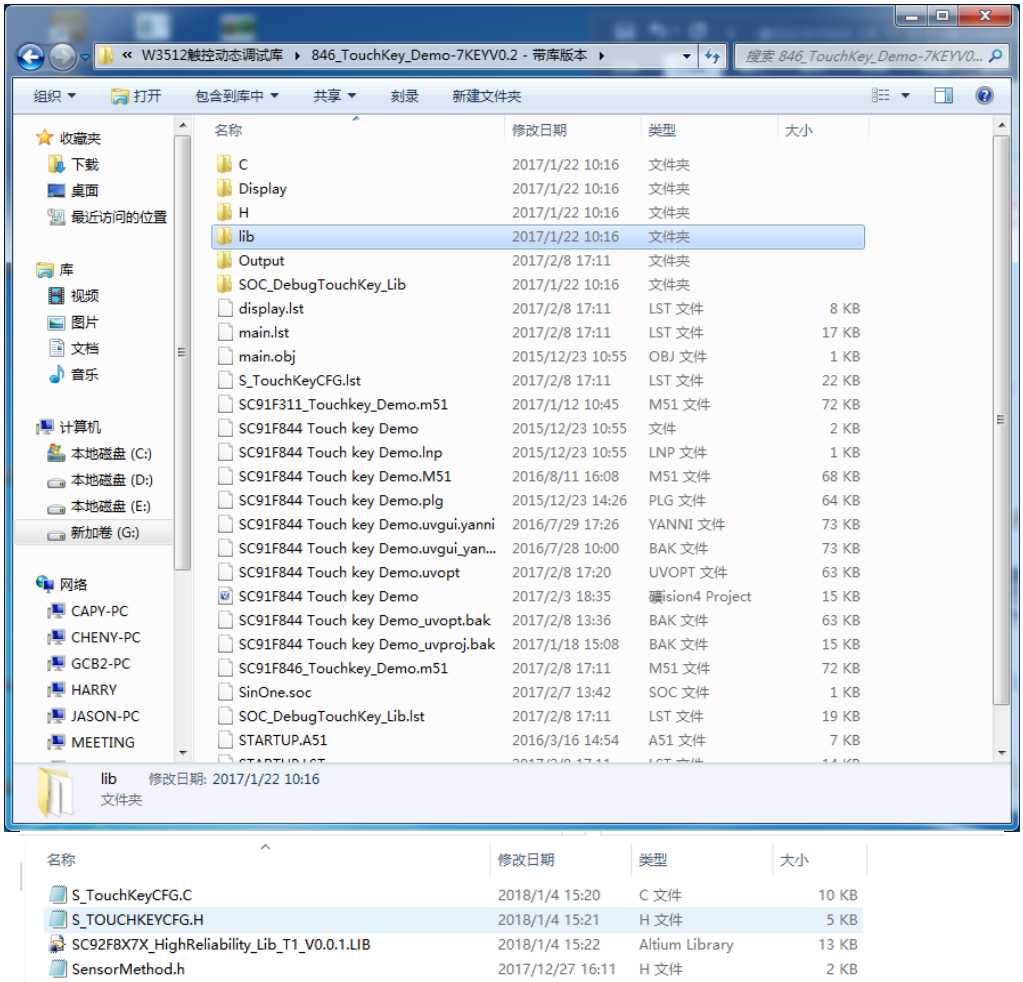
② 用户在实际应用过程中，可再次消抖，以增强可靠性；如连续 2~5 次读到该键值，才有效。

③ 也可通过读取键值的方式来判断：

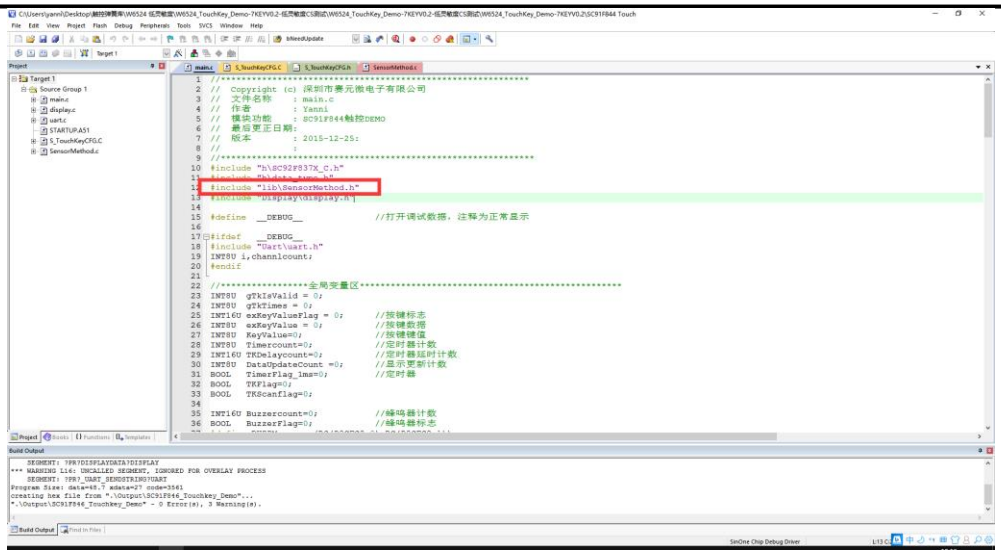
a) 双击；如 1S 内有 2 次按键；

b) 长按键；如持续按键 2S；

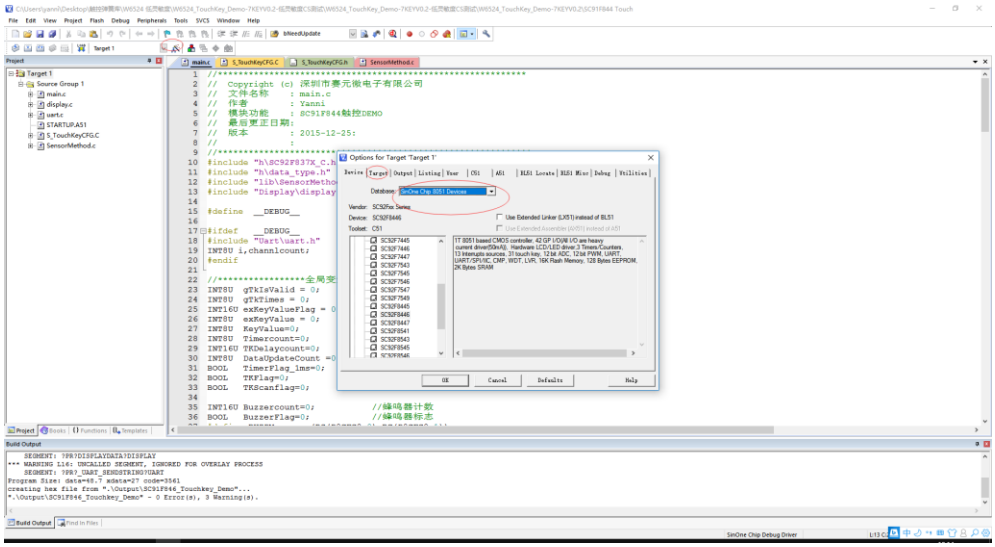
7、 复制“lib”文件夹至工程文件夹内



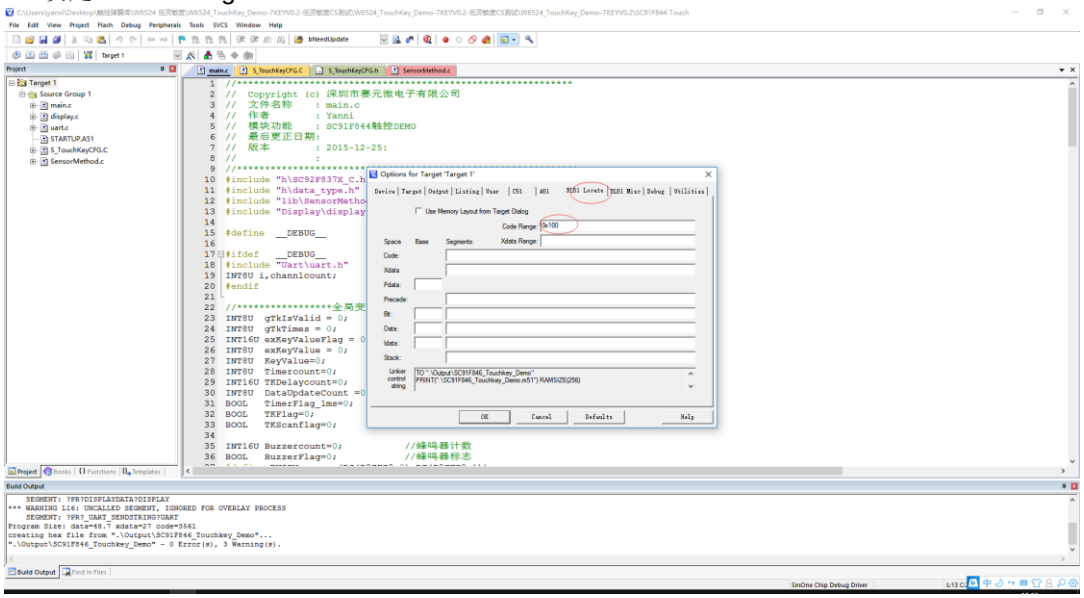
打开工程文件：



设置为赛元 MCU：

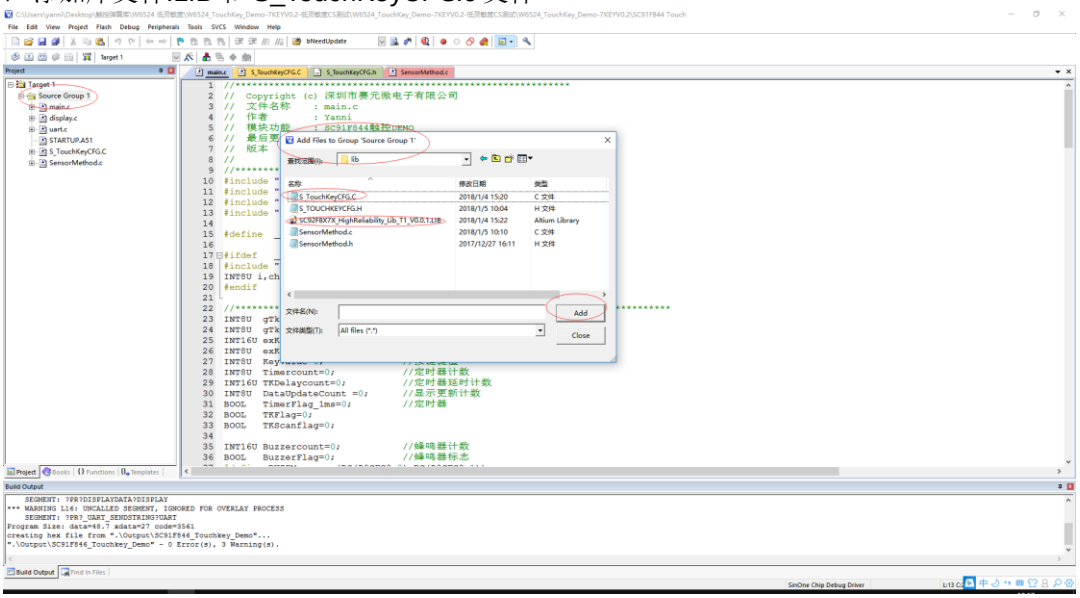


设定 Code Range：

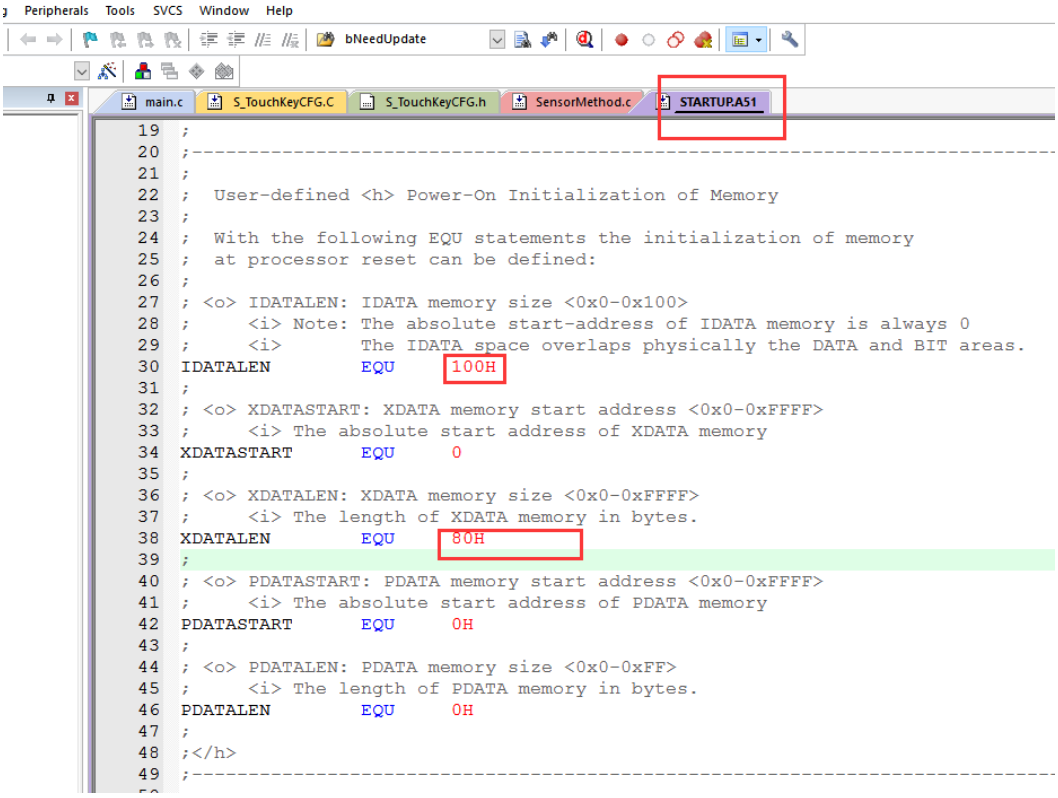


设定的目的，参见 赛元 SC92F 系列 MCU 应用指南 PDF 文件。

8、 添加库文件.LIB 和 S_TouchKeyCFG.c 文件



9、 设定 XDATALEN：



用于 STARTUP.A51 中，清掉外部 128Byte XData。具体型号的 XData 大小见规格书。

10、 在 S_TouchKeyCFG.h 中修改参数（步骤一中记录的 TKCFG1、TKCFG2 和 TKCFG3 的参数）以及触控按键的通道和个数；设定触控按键有效的次数

```
S_TouchKeyCFG.C | S_TOUCHKEYCFG.H | main.c
//  更改记录    :
//  注意事项    :  用户配置文件，用户可修改
//*****************************************************************************************
#ifndef __S_TOUCHKEYCFG_H__
#define __S_TOUCHKEYCFG_H__

//=========================================================================
//触控按键基本配置部分，用户可修改
//触控按键寄存器设置
//=========================================================================
//TKCFG1:
// bit[4] =    PRS             调频开关
// bit[3:0] =  Not define      未定义
//TKCFG2:
// bit[5:4] =  CMPFLTS         滤波设置
// bit[3:0] =  CTIME           充放电频率
//TKCFG3:
// bit[6:4] =  SVSS            充电稳压源电压
// bit[3:0] =  VREF            参考电压
//-------------------------------------------------------------------------
#define         SOCAPI_SET_TKCFG1                   0x00    //默认设置：0x00//SFR:TKCFG1配置：bit7-bit4(PRS);bit3-bit0(NULL)
#define         SOCAPI_SET_TKCFG2                   0x15    //默认设置：0x15//SFR:TKCFG2配置：bit7-bit4(CMPFLTS);bit3-bit0(CTIME)
#define         SOCAPI_SET_TKCFG3                   0x36    //默认设置：0x36//SFR:TKCFG3配置：bit7-bit4(SVSS);bit3-bit0(VREF)
//-------------------------------------------------------------------------

//触控按键的个数，通道设置，每bit控制一个通道
#define         SOCAPI_SET_TOUCHKEY_TOTAL           6       //用户实际使用的按键通道的数量,如用户使用TK8~TK15共8个键,只填8;
#define         SOCAPI_SET_TOUCHKEY_CHANNEL         0x007e  //bit15~bit0对应TK15~TK0;  对应为置1则为TK，对应位置0则为IO

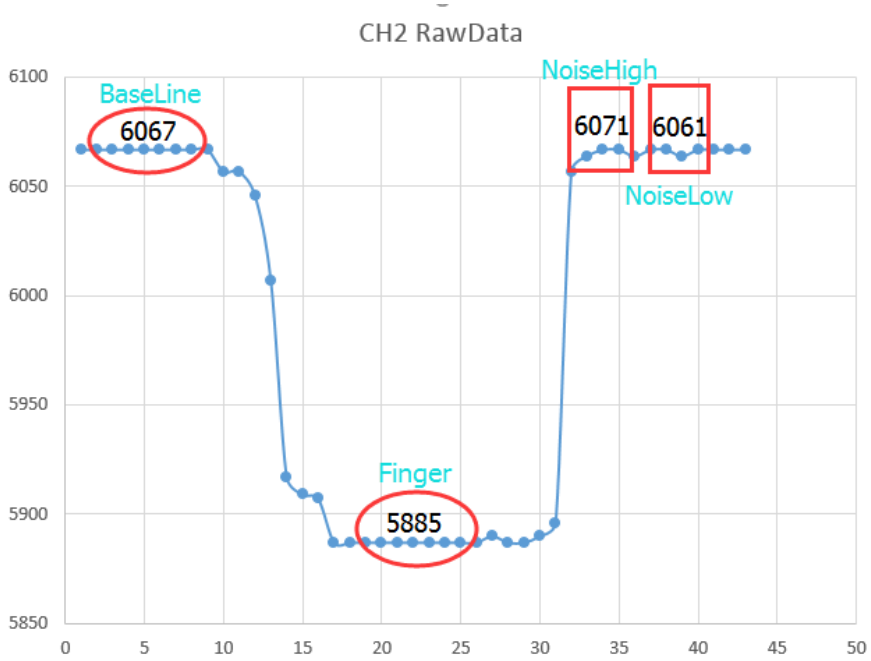
//触控按键的程序检测确认次数
#define         SOCAPI_SET_TOUCHKEY_CONFIRM_CNT     8       //确认按键次数(4~30之间，检测次数越大，反应越慢)

//触控按键的噪音值
#define         SOCAPI_SET_NOISE_THRESHOLD          30      //设置噪音阈值范围:16~40

//每一路通道触控按键的手指阈值，范围0~65535，此为有效差值=（baseline-Finger）*0.6，数值越大，灵敏度越低，反之亦然。
//baseline为手指没按下的rawdata值,Finger为手指按下后的rawdata值
//乘以0.6是留点余量，因为每个人的手指接触面积不一样,用户也可根据触摸效果适当的增加或减小。
//用户只需要设置实际使用的通道手指阈值，其余没用到的通道可以随机数;
```

11、 根据调试步骤中的 RAW DATA 数据，计算出噪声阈值和手指阈值，在 S_TouchKeyCFG.h 中修改；

① 计算手指阈值



a. 无按键时 Baseline 基线平均为 6067；

b. 有按键时 Finger 平均为 5885；

c. 数据变化为 Baseline-Finger=6067-5885=182；

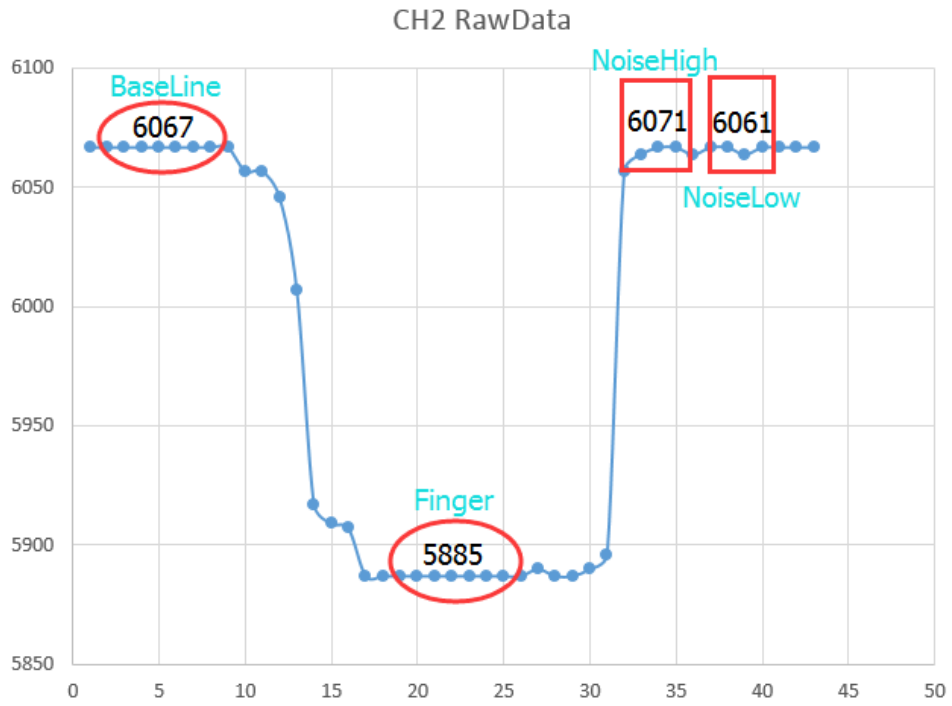
d. SOCAPI_KEY3_FINGER_THRESHOLD：Baseline-Finger；

e. 故这个按键通道的 SOCAPI_KEY3_FINGER_THRESHOLD：

理论值：182；

考虑到手指接触面问题，建议，理论值*0.6，即有效手指阈值为 182*0.6=109；

② 计算噪声阈值



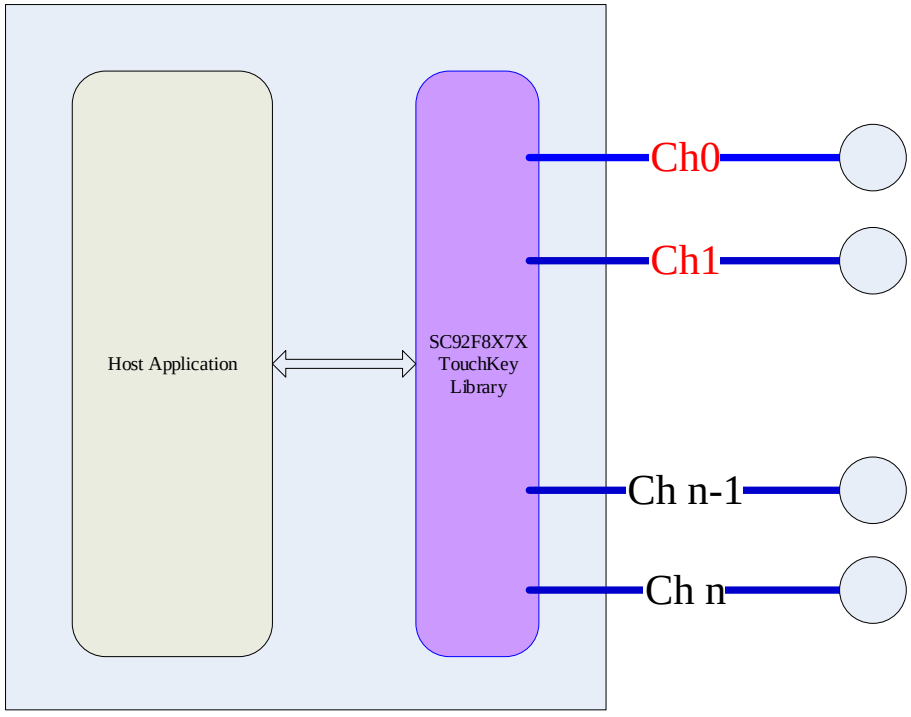
a. 无按键时 Baseline 基线平均为 6067；
b. 无按键时峰值 NoiseHigh=6071,NoiseLow =6061；
c. 数据变化为 6071-6061=10；
d. SOCAPI_SET_NOISE_THRESHOLD：10；
e. 采集所有触控通道的噪声阈值,取最大值作为所有通道的噪声阈值，一般设置为 20~40 之间。

③ 在 S_TouchKeyCFG.h 中修改噪声阈值和手指阈值

4.3.3 完成用户程序和赛元触控软件库的融合

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)



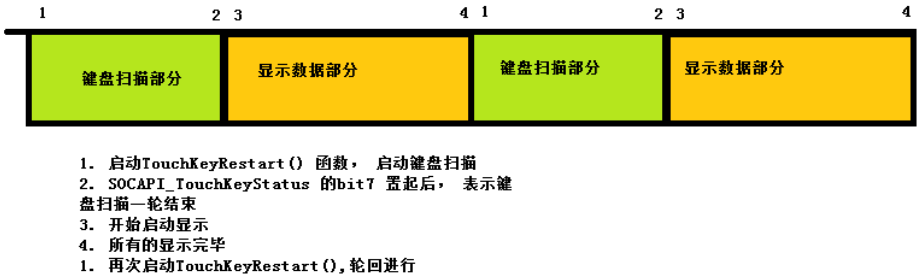
2、库文件的调用流程。用户通过一定的流程调用库文件的接口函数，便可得到触控按键的键值。

① 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；

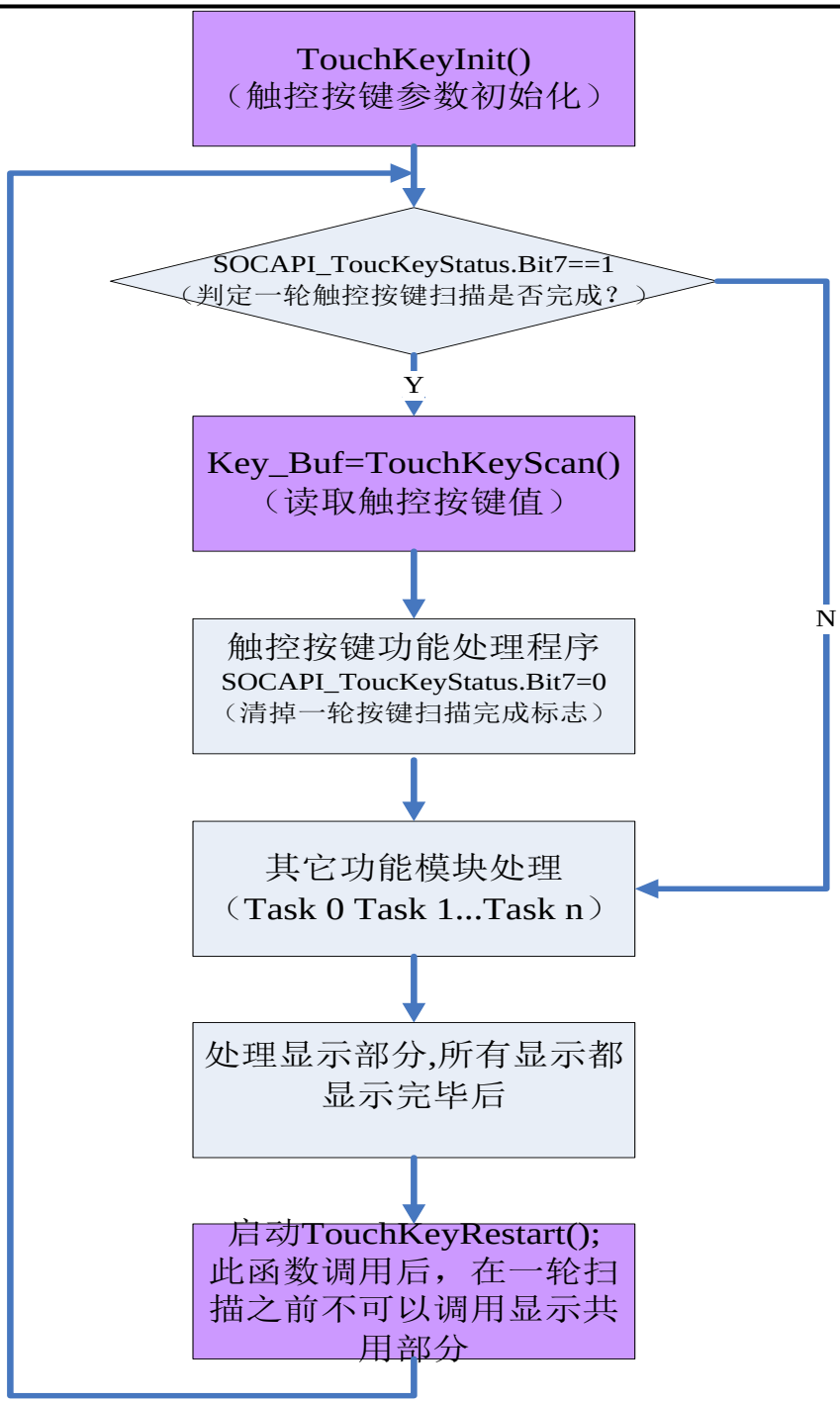
② 主程序通过查看全局变量 SOCAPI_TouchKeyStatus&0x80 来判定一轮触控按键扫描是否完成；

③ 主程序调用接口函数“Key_Buf=TouchKeyScan()”用于读取触控按键值；

④ **特别需要强调一点的是：如果用户是 TouchKey 与 LED 共用项目，那么在调用 TouchKeyRestart()开始扫描按键时，在 SOCAPI_TouchKeyStatus & 0X80 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

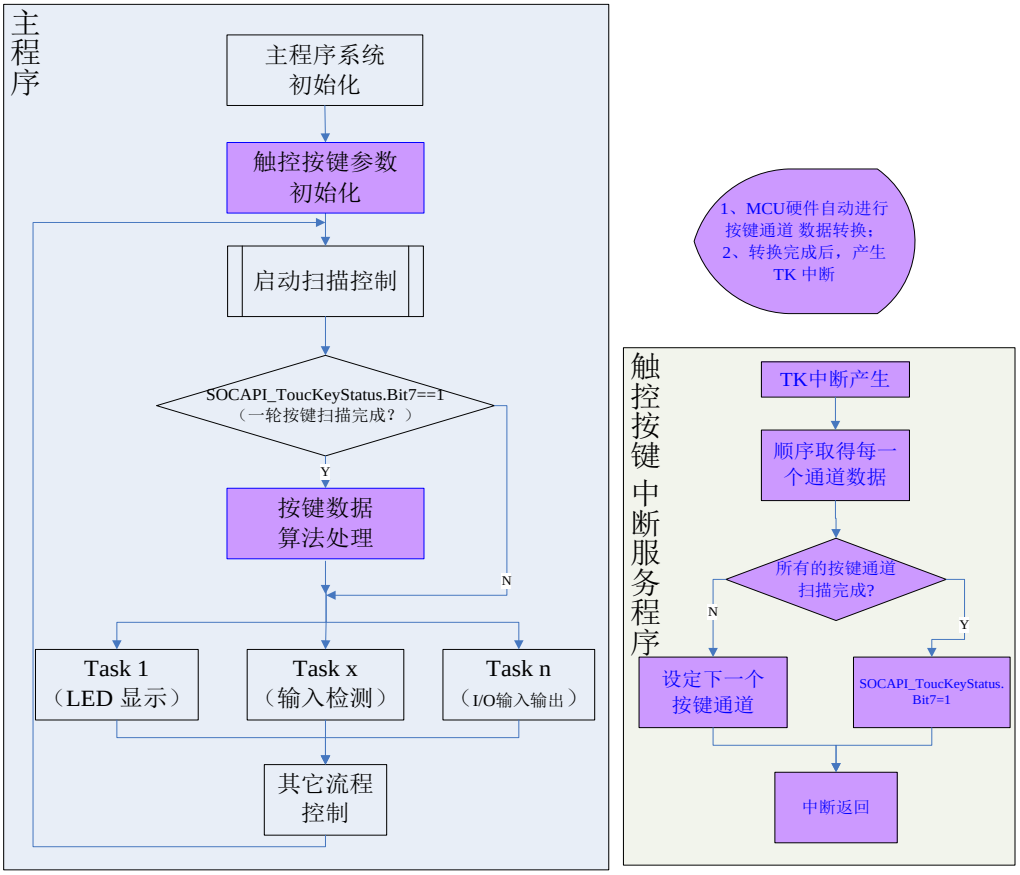
3、主程序和库文件的时序关系。因为运行触控按键库消耗了部分 IC 资源和时间，为了让用户程序和库程序能完美融合，主程序需要遵循以下要求：

① 提供给库运行的资源 ROM、RAM、INT 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；如改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；（触控按键中断服务程序内无函数嵌套）

④ 触控按键扫描取 Counter 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；



4、软件融合的注意事项

② 运行时间：

TK 按键中断内：6.8uS 左右

执行一轮按键扫描过程中，TK 每次中断的时间间隔，约等于 Rawdata/16（us）

接口函数；

TouchKeyInit(void)： 200~500Ms@12M；

TouchKeyScan(void)：算法执行时间会因按键选择个数的增减而增减。N 个 Key 约(240*N) us@12M 。

③ 中断优先级：

库文件使用 TK 中断，优先级为低；

用户可以在主程序控制 Timming 的需求：

a) 设置其它中断优先级为高；会拉长 TK 按键确认时间；

b) 设置 TK 中断优先级为高；会打断其它中断服务程序；

c) 如均保持所有中断优先级为默认值，中断服务顺序执行，TK 排在向量入口最前面，会先进中断。

① 用户在实际应用过程中，可再次消抖，以增强可靠性；如连续 2~5 次读到该键值，才有效。

② 也可通过读取键值的方式，判断：

a) 双击；如 1S 内有 2 次按键；

b) 长按键；如持续按键 2S；

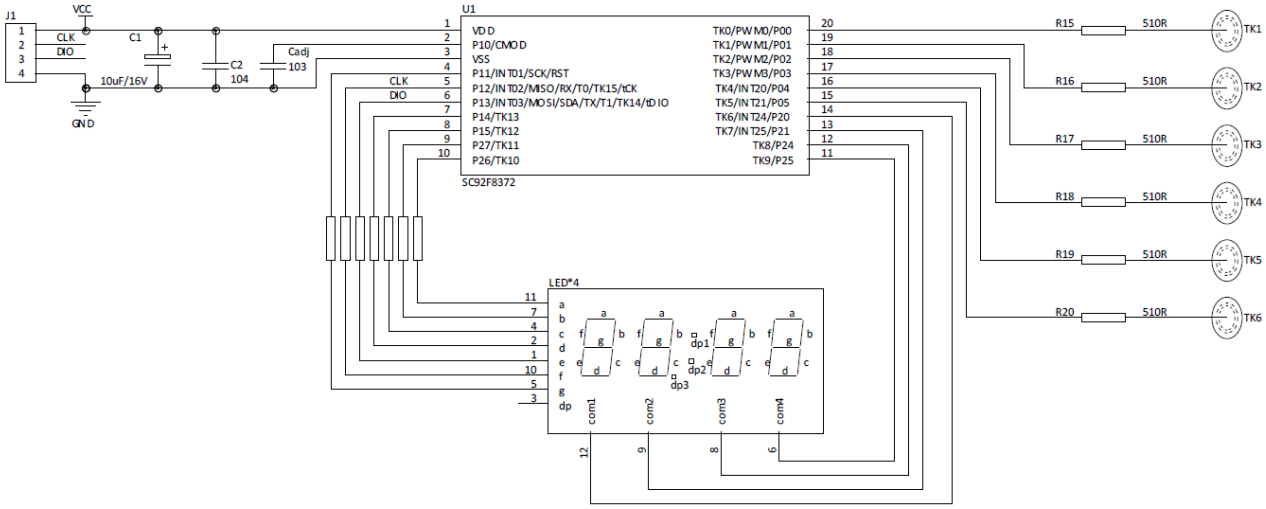
5、整体 Code 的测试

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。

附录

一、应用参考原理图（以 SC92F8372 为例）



TouchKey 与 LED 不共用原理图

二、库使用注意事项

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头形成电场，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、 感应盘与地的寄生电容越大，则需要越大的 Cadj 电容来匹配，从而影响 Cadj 电容的取值范围以及灵敏度的调节范围。感应盘与感应盘之间距离足够保持 2mm 以上，尽量避免不同感应盘间平行引线距离过近，这些都能降低感应盘对地的寄生电容，有利于产品灵敏度的提高。
3、 有些产品为了获得更好的 EMC 测试效果，需要在按键板上铺地，铺地时地线需要尽量远离感应盘以及感应盘引线距离 2mm 以上。
4、 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。
5、 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更高穿透力。
6、 感应盘与感应盘之间应该尽量留一定的间距，以保证手指触控时不会覆盖到 2 个感应盘，同时也能防止感应盘寄生电容过大。
7、 因为空气介电常数太小，并且受湿度影响，所以介质中最好不要有空气。感应盘与绝缘外壳应压合紧密，保持平整，以免有气隙产生。外壳与感应盘之间可以采用非导电胶进行粘和，例如压克力胶 3M HBM 系列。
8、 基准电容 C 电容建议使用温度系数小精度高的电容，以免造成灵敏度不一致或随温度变化而变化。一般插件电容建议 5%精度涤纶电容，如需贴片电容则建议使用 10%或更高精度的 NPO 材质电容或 X7R 材质电容。
9、 芯片供电电源需要采用三端稳压 IC、RC 滤波、LC 滤波等电路来防止交流纹波干扰，以保证系统的稳定性能，使用 大电流 LED、交流蜂鸣器时 建议一定要给触控芯片加 RC 滤波。
10、 TK 的 IO 口设置为强推挽输出 1
11、 软件方面，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；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(gIsLcdDataBuf[0]);
                seg8 = gIsLcdDataBuf[4] & 0x01;
                seg9 = gIsLcdDataBuf[4] & 0x02;
                COM0 = 0;

                isLcdComflag = 1;


        }
        else if(isLcdComflag ==1)                       //显示 COM1 数据
        {
                 SetSegData(gIsLcdDataBuf[1]);
                 seg8 = gIsLcdDataBuf[5] & 0x01;
                 seg9 = gIsLcdDataBuf[5] & 0x02;
                 COM1 = 0;

                 isLcdComflag = 2;
        }
        else if(isLcdComflag ==2)                       //显示 COM2 数据
        {
                 SetSegData(gIsLcdDataBuf[2]);
                 seg8 = gIsLcdDataBuf[6] & 0x01;
                 seg9 = gIsLcdDataBuf[6] & 0x02;
                 COM2 = 0;
                 isLcdComflag = 3;
        }
        else if(isLcdComflag ==3)                       //显示 COM3 数据
        {
                 SetSegData(gIsLcdDataBuf[3]);
                 seg8 = gIsLcdDataBuf[7] & 0x01;
                 seg9 = gIsLcdDataBuf[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: 扫键的初始化函数
        InitialLcd();           //初始化显示部分

        while(1)
        {
          WDTCON |= 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**********************
          }
        }
}
//定时中断 1ms 显示一次数码管
void timer0()interrupt 1
{
        TH0=(8192-4000)/32;                     //2000*1/4us=500us
        TL0=(8192-4000)%32;
        TimerFlag_1ms = 1;
        DisplayData();
}
void  DisplayData(void)
{
        ComAllClose;
        if(isLcdComflag == 0)                                   //显示 COM0 数据
        {
                LedSetSegData(LcdDisplayBuf[gIsLedDataBuf[0]]);
                COM0 = 0;
                isLcdComflag = 1;
```

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

5 更改记录

版本	记录	日期
V1.0	初版	2018 年 05 月
V1.1	1、增加烧录静态调试代码时的 LVR 选择设置说明 2、增加的 TK 对应的 IO 口设置说明 3、更改了 8x6x 系列部分说明	2018 年 06 月
V1.2	1、更改了 T2 库的部分说明 2、增加动态调试注意事项 3、更改了 8x6xB 系列部分说明 4、增加了 8X4XB 的说明 5、更改调试 code 以及动态调试库名称 6、修正软件库函数耗时的说明	2018 年 10 月