Table of Contents

1. Overall Description of SinOne SC95F Series Touchkey MCU Application Guidelines

As the application guideline of SinOne SC95F Series Touchkey MCU, this document aims to introduce how to use SinOne TouchKey library file and how to debug the parameters of Touchkey upper computer. SinOne MCU touch architecture is divided into high-sensitivity touch mode and high-reliability touch mode, some models have built-in dual-module touch (refer to specifications for details). High-sensitivity or high-reliability mode can be used by selecting different touch library files and its characteristics are as follows:

- High-sensitivity mode is applicable for Touchkey, spaced touch key, wheel/slider, proximity sensor and other touch application with high-sensitivity requirements
- Both high-sensitivity/high-reliability require strong anti-interference capability
- Support up to 31 touchkeys and derivative functions
- Support flexible development software library and easy for development
- Support automatic debugging software and intelligent development
- Partial models can run with low power in MCU STOP mode, and the overall power of the chip upon waking up by using 12 touchkeys (500mS) can be as low as 22uA@3.3V / 25uA@5V

The user can use SinOne Touchkey library files to select touch mode and fast realize required touch functions easily. For the most suitable touch mode, see the following table in detail:

Description	High-sensitivity Mode	High-reliability Mode
Features	●High anti-interference capability, passing 10V dynamic CS ● Super-high Sensitivity	●High anti-interference capability, passing 10V dynamic CS ● Lower power consumption
Applications	●Spring touch key application ●Spaced touch key application ●Proximity sensing application ●Wheel/slider application ●Touch application with high-sensitivity requirements	● Require low power current
Applicable Mode	Select high-sensitivity mode by loading SinOne high-sensitivity touch library through the project	Select high-reliability mode by loading SinOne high-reliability touch library through the project
Library Description	3.3.1 High-sensitivity Library Touch Software Library Porting	3.3.2 High-reliability Library Touch Software Library Porting
Corresponding Library File	"SC95F8XXX_HighSensitive_Lib_Tn_Vx.x.x.LIB"	"SC95F8XXX_HighReliability_Lib_Tn_Vx.x.x.LIB
Notes	●T1 library is applicable to spring type applications ● T2 library is applicable to spaced touch type applications with its key at least 3 or more	●Only applicable to spring type applications
Selection Description	In general, it is recommended to use this high-sensitivity mode, which may obtain a better experience	Only in the following situations, high-reliability mode is recommended: A low power current is required and the current can not be filled in the high-sensitivity mode

Notes for Power Supply of SinOne Series Touchkey MCU Touchkey Chip:

- Power supply range of touch key chip: refer to the specifications of corresponding chip
- Power supply ripple of touch key chip: Recommended working supply voltage ripple magnitude of touch chip ≦ 3%~4% with a maximum of no more than 200mv.

2. Introduction to SinOne Touch Library

2.1 APPLICATION TYPES OF TOUCH LIBRARY

SinOne SC95F Series Touchkey MCU provides a library file that can be called by the user to reduce the difficulty in developing user touchkeys.
The library can be divided into the following types:

- Ordinary Touchkey Library
  - Spring touch library (T1 library for short)
  - Spaced touch library (T2 for short)
  - Highly reliable library
- Wheel/slider touch key library
- Proximity sensing touch library
- Low-power touch library (including ordinary low-power touch library, wheel/slider low-power touch library)

The document is to introduce the use of touchkey library and touch debugging upper computer SOC TouchKey Tool Menu software; the use of wheel/slider library and proximity sensing library will be introduced in the special application guidelines. For details, see Descriptions for SinOne TK Special Applications

The user shall follow the steps below to realize the functions of touch keys, perfectly combine SinOne touch software library with the user's software and finally achieve the product's functions.

2.2 GENERAL STEPS FOR TOUCH PROJECT DEVELOPMENT

A complete touch project is developed in the following steps:

1. Install the development tool, configure and export the parameters
A specialized touch debugging upper computer software SOC TouchKey Tool Menu provided by SinOne aims to complete the debugging work through a series of human-machine interaction for the user. The user needs to install this software and use it with the online programmer DPT52/SC-LINK/SC-LINK PRO. The user can search for the most suitable touch key parameters of the user's PCB through the configuration parameters of the software interface, and export the final relevant parameters to generate the head file to be used in the user project.

2. Realize function testing of SinOne software library
Add the configuration file generated in Step 1 to SinOne touch key library and add the whole library-related files to the user project for compilation. The simple testing program for the user provided by SinOne can be used to complete the testing of key functions.

3. Complete the integration of user program and SinOne touch software library
The user can write the other part of the software except the touch key and nest SinOne software library into the user program to complete the overall functions of the product.
For detailed development operations, see 3 Touchkey Development Process

2.3 INTRODUCTION TO SINONE TOUCH LIBRARY FILES

SinOne touch library is composed of the following files:
SensorMethod.h: This file is the declaration of the program function of touch library. The user needs to refer to this head file in the main program.
SC95F8XXX_X_X_Vx.x.x.LIB: This file is related to the algorithm of the touch library. The user needs to add this file to the project for compilation
S_TOUCHKEYCFG.H: This file is the configuration file of touch-related parameters, which is generated by debugging with SOC TouchKey Tool Menu by the user.
S_TouchKeyCFG.C: This file is composed of the head file of touch parameters and interfaces related to touch library interaction, the user needs to add this file to the project for compilation. For the ordinary touch library, there is no need to modify this file.
For wheel/slider touch library and proximity sensing touch library, the parameters need to be configured. Please refer to the

special application guidelines for details:
Descriptions for SinOne TK Special Applications

3. Touch Development Process

This section is to introduce how to develop the touch project. **It should be noted that, for the project development, the complete touchpad PCBA is also an indispensable part of the project debugging; for touch MCU layout, please refer to Design Points for SinOne Touch Key MCU PCB. Before the development, make sure that the hardware for the touch project complies with relevant requirements, which may eliminate the problems maybe encountered during the development process.**

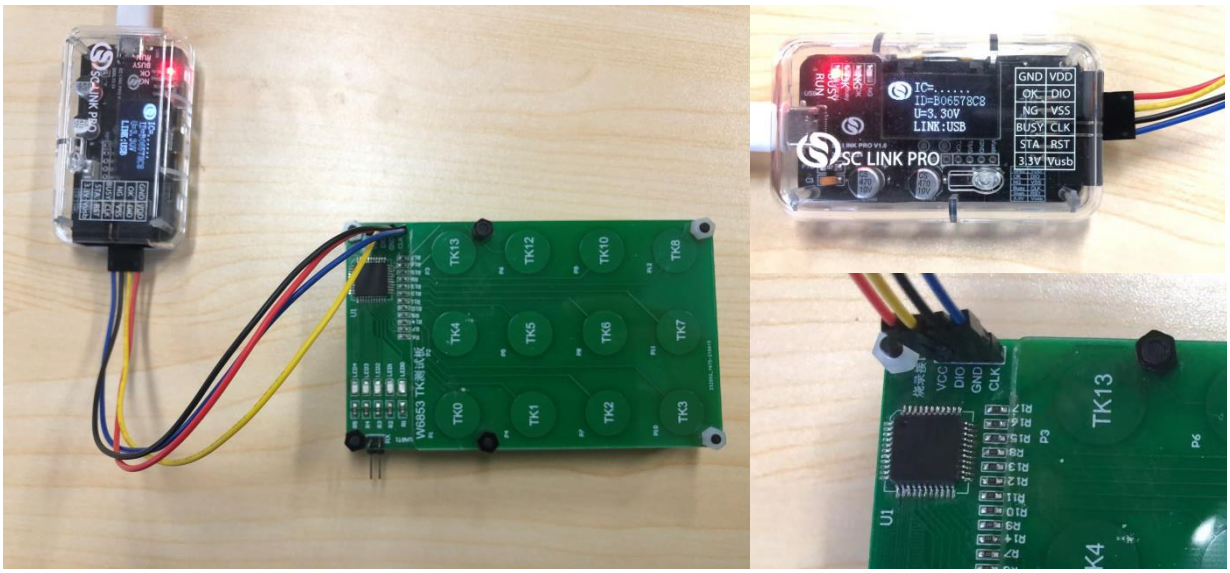
3.1 Install the Development Tool

1) Setup SOC Pro51/SOC Programming Tool
Install SOC Pro51 Vx.xx.exe/ SOC Programming Tool (Visit SinOne website for the latest version).

2) Setup SOC TouchKey Tool Menu
Install SOC TouchKey Tool Menu (Visit SinOne website for the latest version).

3) Upgrade the firmware DPT52/SC-LINK/SC-LINK PRO and update MCU library
The firmware of the online programmer DPT52/SC-LINK/SC-LINK PRO and MCU library files of SOC Pro51/ SOC Programming Tool shall be updated to the latest version from SinOne website).

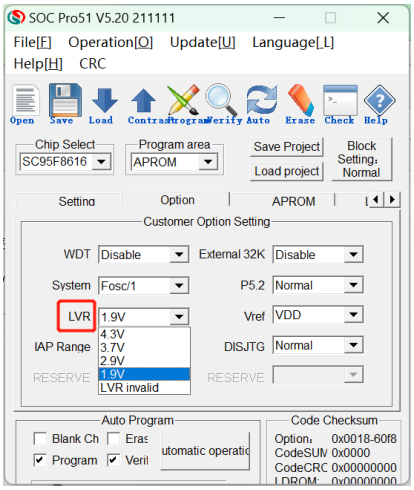
4) Install the plug-in SOC_KEIL;
Please update the plug-in of SinOne MCU to the latest version from the website. Specific installation method and notes are as follows:
a. Install the plug-in SOC_KEIL, which can be used to automatically search for the installation directory of KEIL (C51 version) and install all files in SinOne_Chip/SinOne_Chip_SClinkPRO in C51 directory of KEIL C installation directory.
b. All files in the directory of SinOne_Chip /SinOne_Chip_SClinkPRO are as follows:
CDB: SinOne MCU development library file
DEMO: SinOne MCU demonstration program
INC: SinOne MCU head file
PDF: Instructions for SinOne SOC programming simulation tool SC-LINK PRO
SOC_Debug_Driver/SCLINK_PRO_Debug_Driver: SinOne simulation plug-in
c. SinOne SOC_KEIL plug-in is to create a specified list for SinOne MCU, which will not override the original MCU list of KEIL C.
d. If it is unable to install SOC_KEIL plug-in ,please confirm if your KEIL is C51 version.
5) Hardware connection sequence: PC USB-->DPT52/SC-LINK/SC-LINK PRO(VCC/GND/CLK/DIO)-->User PCB(VCC/GND/tCK/tDIO); test and confirm if the connection is normal. In the debugging process, the hardware UART resources are required, so reserve PCB routing interface for UART, as shown in following figure for SCLINK PRO wiring.



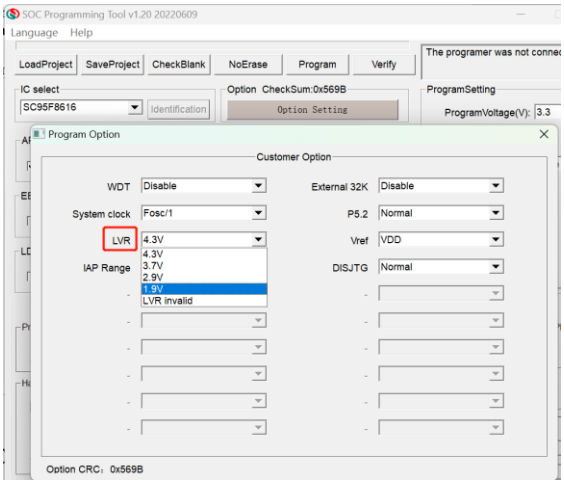
6) Program static debugging code hex file to SC95F8XXX IC of user PCB

Open SOC Pro51/SOC Programming Tool, select MCU model used for the project, load the hex file of static debugging code, click "Program" and close SOC Pro51/SOC Programming Tool after completion, then plug in the USB and power it on. (**Note: LVR shall be set to be lower than the supply voltage, for example, if the supply voltage is 3.3V, LVR in the Option must choose a gear lower than 3.3V**)

See the diagram of SOC Pro51/SOC Programming Tool below:



SOC Pro51 Software



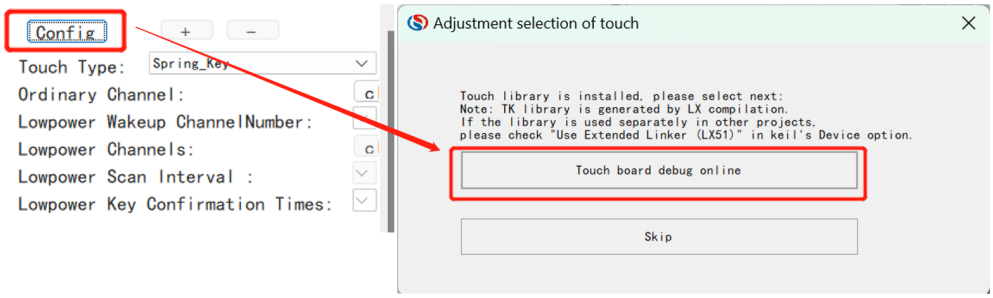
SOC Programming Tool Software

For the operating steps of high-sensitivity debugging, see: 3.2.1 High-sensitivity Debugging Touch Parameters
For the operating steps of high-reliability debugging, see: 3.2.2 High-reliability Debugging Touch Parameters

3.2 Debugging Touch Parameters

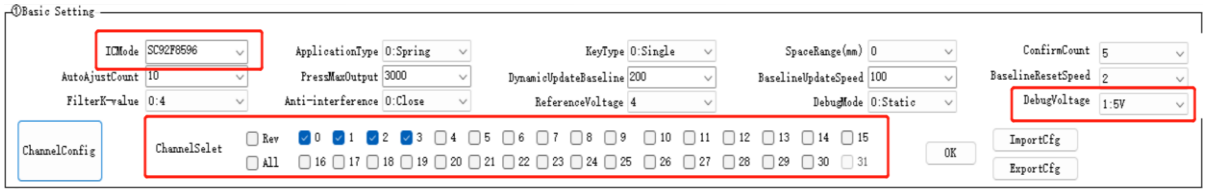
3.2.1 High-sensitivity Debugging Touch Parameters

1) Open **Adjustment selection of Touch**



2) Configure the parameters to go to touch parameter debugging

① Select MCU model used for the project and check TK channel used, as shown in the figure below:



② Set up the basic information of the application, as shown below:

Application Type: Select Spring Key/Spaced Key/Proximity Sensing based on the project needs (it is temporarily not supported for matrix key application).

Key Type: It is required to be set for spring key application. No settings are required for other applications.

Select single key or combined (double) key according to the actual project.

Space Distance: Set between 0 and 3mm for spaced key application. No settings are required for other applications.

(For farther distance, please contact SinOne engineer for assistance)

Debugging voltage selection: Related to VDD supply voltage of SinOne MCU chip in the project.

Select 5V debugging for 5V project and 3.3V debugging for 3.3V project.

③ Configure the parameters related to touch algorithm operation (keep the default parameters unchanged, the following is relevant contents of each parameter)

Key Confirmation Counts: The default value is recommended. This parameter determines the key response speed of touch algorithm running, which is related to one round of key scanning time; if such time is 12MS and key confirmation counts are 5, the response time for the key is 5*1 2MS=60MS.

Auto Calibration Counts: The default value is recommended. This parameter determines the speed of initialized baseline, the more times the auto calibration is done, the stabler the baseline will be and the longer the time will be.

Maximum Response Time of Key: The default value is recommended. This parameter determines the continuous response time of key in runs. After the key time reaches specific times, the key's sign will be cleared.

Dynamic Update Baseline Time: The default value is recommended. This parameter is used to deal with the update speed of key up and remains the default value unchanged

Baseline Update Rate: This parameter is used to update the baseline.

Baseline Reset Rate: The default value is recommended. This parameter determines the speed of baseline reset. The larger the value is, the slower the update rate will be.

Filter K Value: The default value is recommended.

Anti-interference Setup: Used to scan variable frequency of clock and conducive to passing EMI test; when EMI test is required for the project, select to open 1:12bit. Note: For low power applications, it is not allowed to enable anti-interference setup.

Reference Voltage: The default value is recommended.

Debugging Mode Select: The default value is recommended. Static debugging is to confirm TouchKey parameters and dynamic debugging is to collection data in the application; here we select static debugging, and dynamic debugging will be introduced in the subsequent chapter.

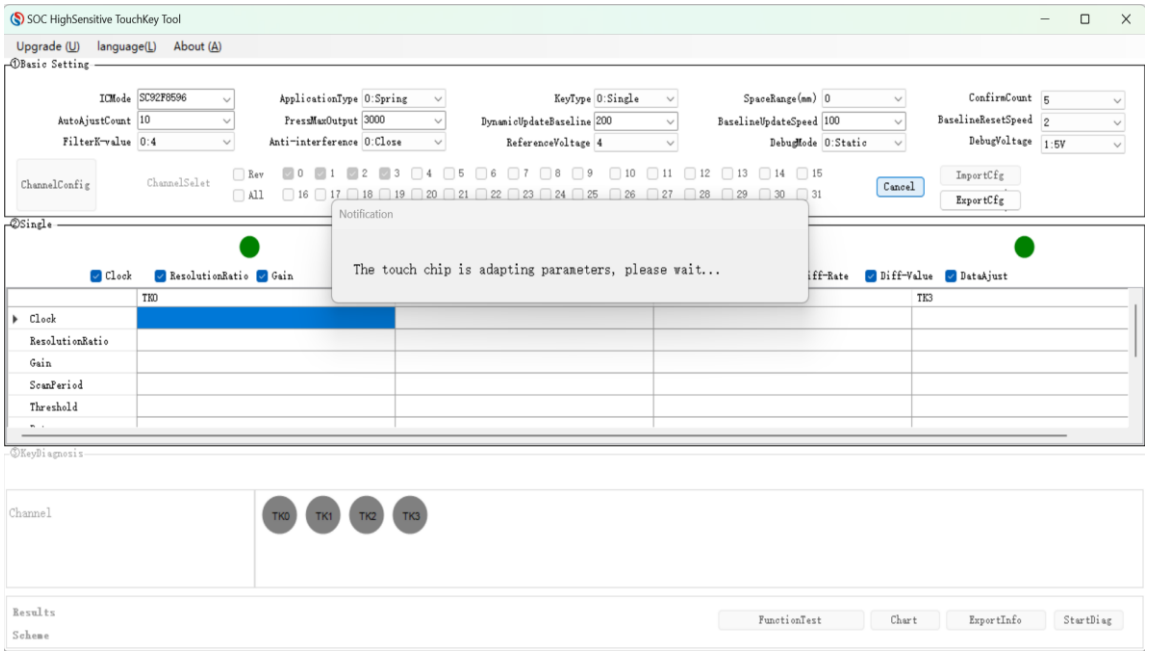
④ Check the complete TK channel interfaces required for the project, after the steps mentioned above are completed, click "OK"; At this time, the channel will be locked and can not be set up. To change the channel, click "Cancel" button.

Note: Since UART resources on the programming interface are needed for touch debugging and some models of

programming interface features TK, the parameters of these two channels can not be debugged during the process of touch debugging. To use these two TK interfaces, please contact SinOne engineers for assistance.

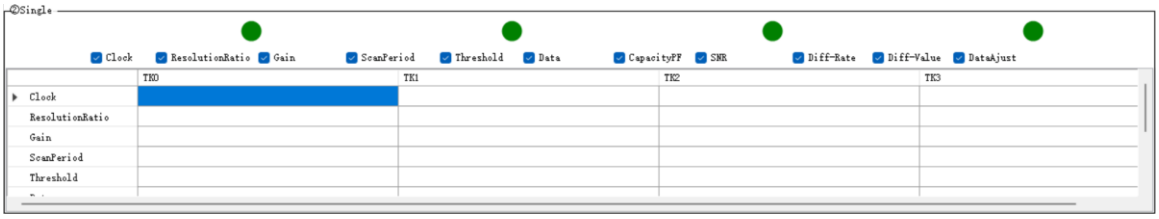
3) Touch Key Parameter Self-adaption

Click "OK" to enter touch key parameter self-adaption phase, then wait for tens of seconds to several minutes, which are related to the number of keys, until the popped-up prompt window is closed and the self-adaption is completed. **During this process, the user needs to install the machine and do not perform any operation on or around the panel.**

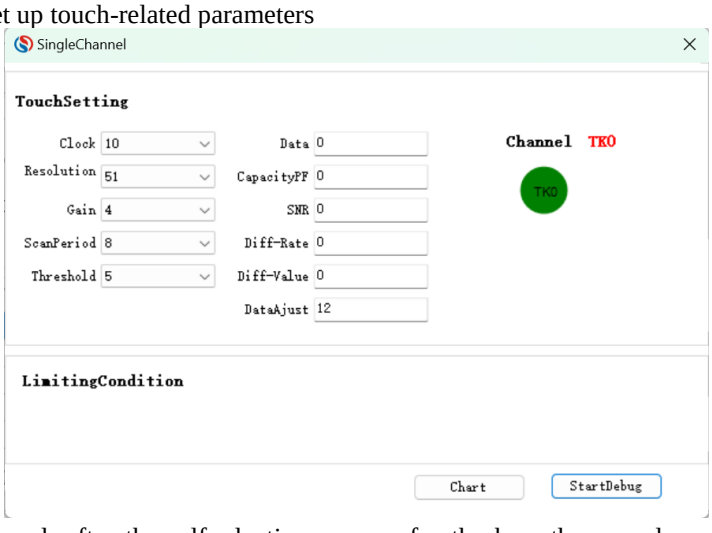


4) Conduct Single-channel Debugging

① Click green button of corresponding channel in Channel Debugging Area to enter Single-channel Debugging Interface



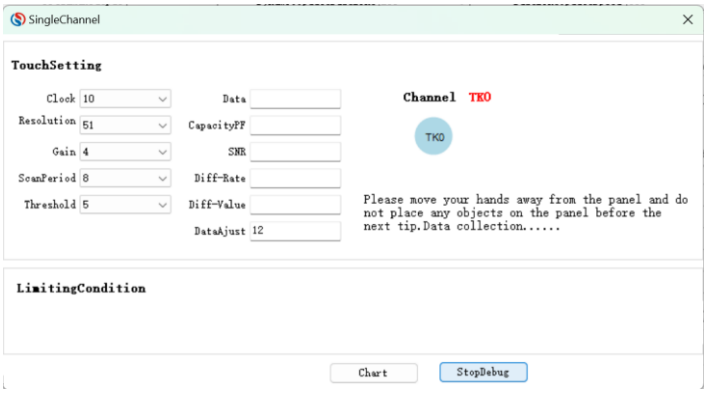
② Set up touch-related parameters



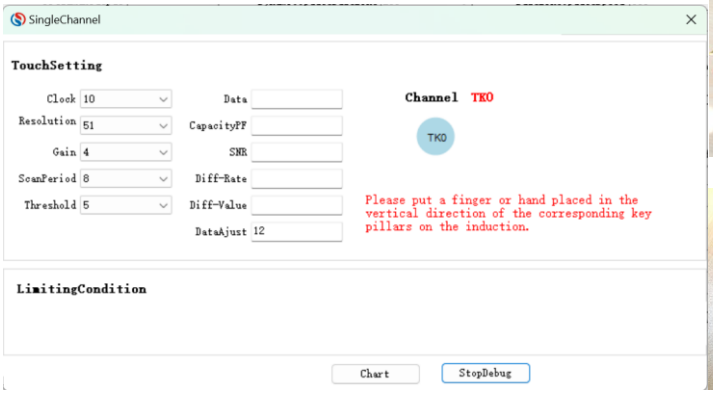
In general, after the self-adaption process for the key, the user does not need to modify the above-mentioned parameters and just click to start the debugging.

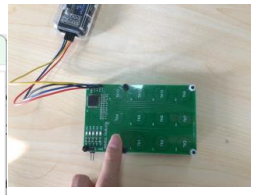
Clock: Remain the default value unchanged
Resolution: Remain the default value unchanged
Gains: Remain the default value unchanged
Scanning Cycle: Setup range 1-32 in 128us. The larger the value is, the longer the scanning time of this key will be, and the larger the variation will be.
Threshold Setup: Setup range: 1-8. The larger the value is, the lower the sensitivity will be. If the set value is 5, the threshold is set as 50% of variation; when the data variation exceeds the threshold, it is considered that there is the key. The recommended value is 5.

③ Click "Start Debugging" button to debug
It is divided into touchless process and touch process.
Please operate as prompted. The process will take about 15 seconds.
Touchless Process:



Touch Process:



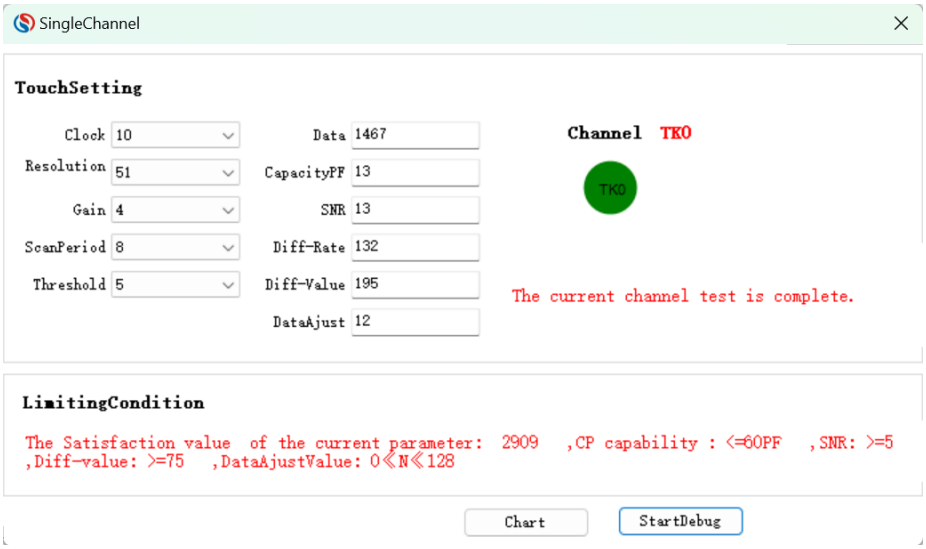


In the debugging phase of common keys, vertically place the finger closely on the sensing surface of the

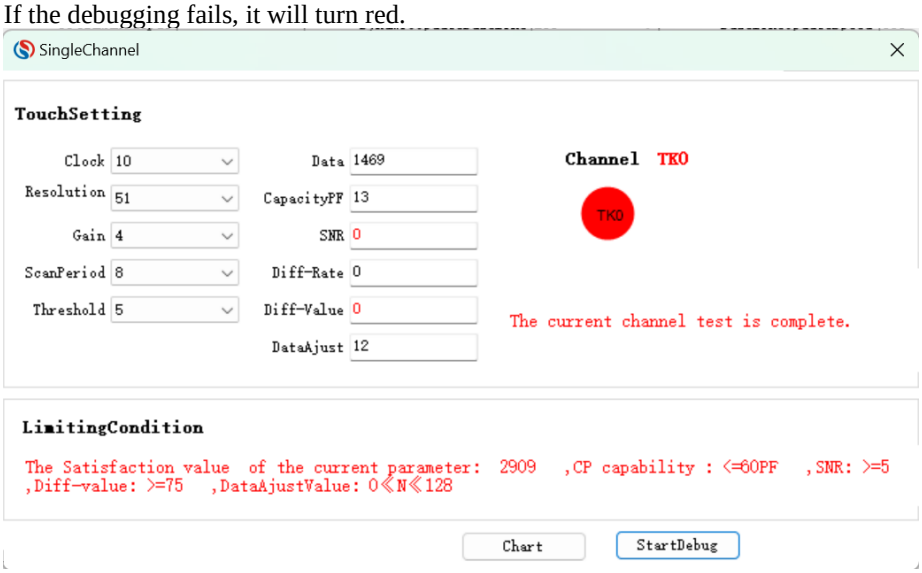
In the wheel/slider debugging phase, vertically place the finger closely over the sensing surface of sawtooth center, as shown in the white area on the left

Note: TK channel displayed in the software is consistent with MCU specifications. Please follow the actual PCB layout to operate corresponding keys, or else, the result will go wrong.

Single-channel Debugging End: If the debugging is passed, the icon in the interface below will turn green:



If the debugging fails, it will turn red.

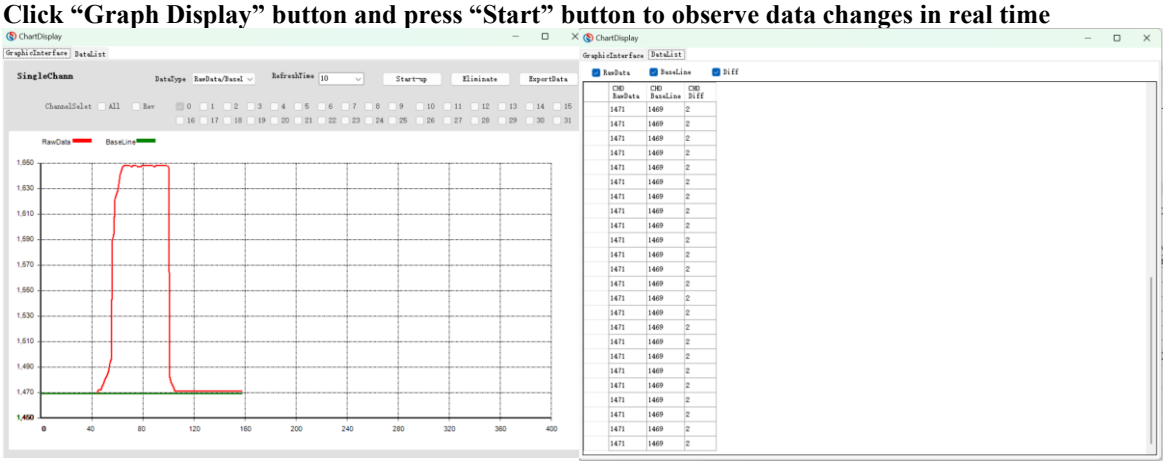


Failed items will be marked in red accordingly.

Debug each key in turn until all keys are passed.

Note: Observation items are attached (non-essential process for debugging)

Click "Graph Display" button and press "Start" button to observe data changes in real time



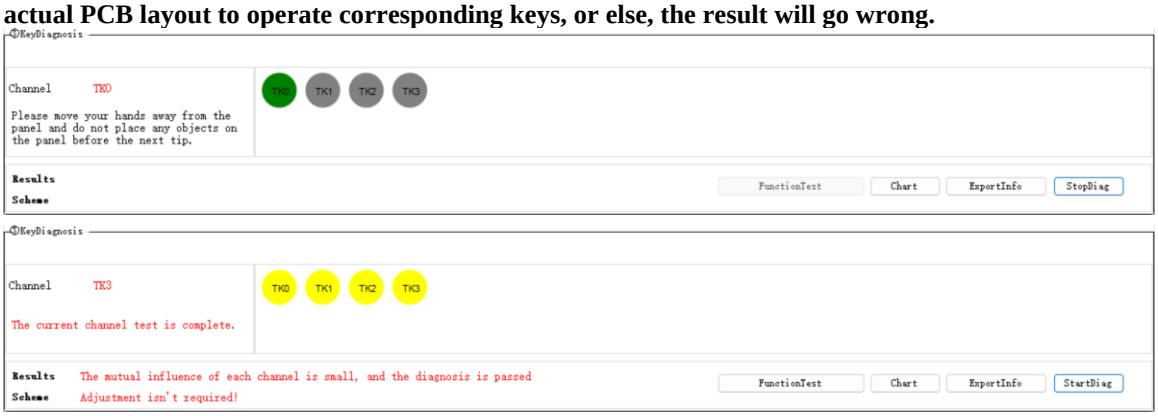
④ Conduct Key Diagnosis (Diagnosis is performed only for ordinary touch keys other than wheel/slider; there is no need to operate for wheel/slider keys during the process of diagnosis, and wait until it is switched to ordinary keys for diagnosis)

Key Diagnosis is the process to analyze the interaction between keys. If such interaction is large, it may

influence the performance of keys.

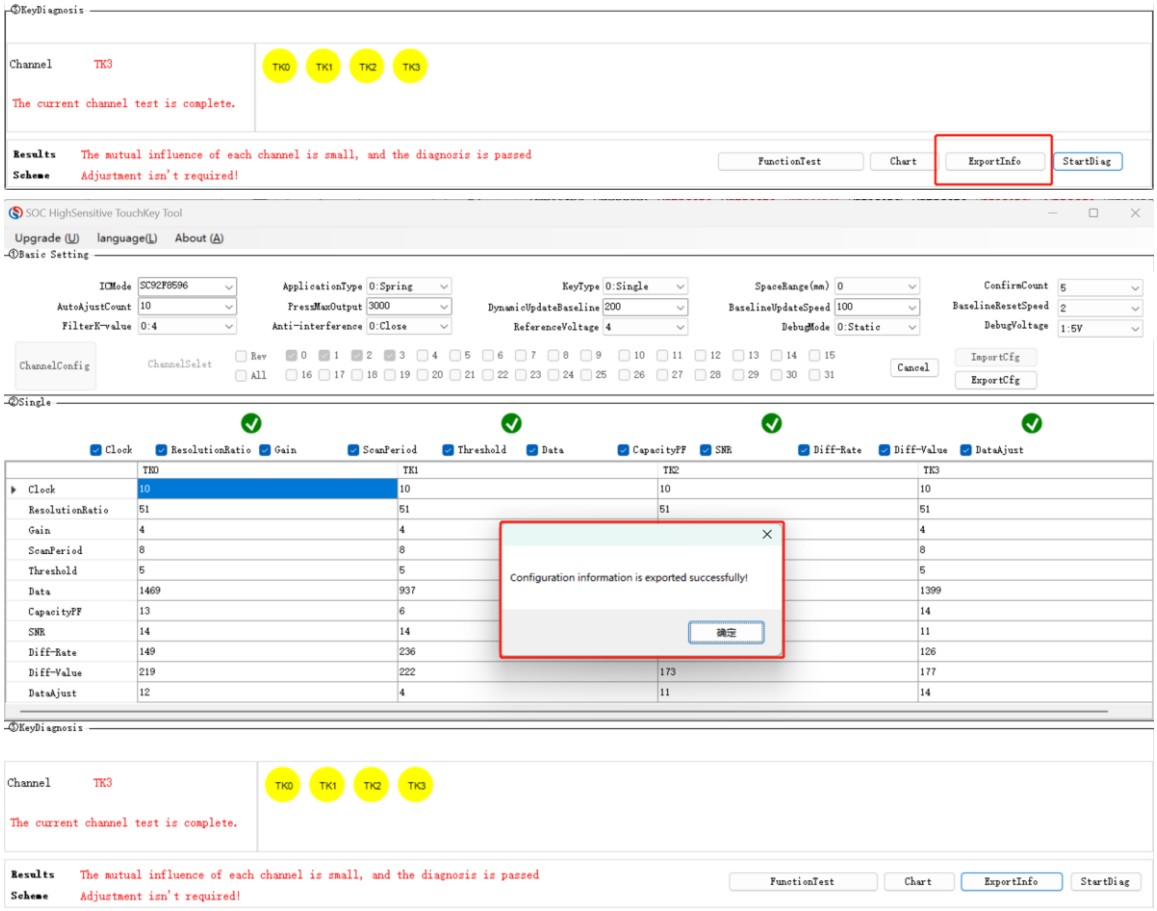
Click “Start Diagnosis” button

Note: TK channel displayed in the software is consistent with MCU specifications. Please follow the actual PCB layout to operate corresponding keys, or else, the result will go wrong.



If the diagnosis is failed, please adjust the hardware Layout according to the diagnosis results and adjustment scheme. Below is the prompt of failed diagnosis:

⑤ After completing key diagnosis and passing the test, click “Export Configuration Information” button to generate the configuration file S_TOUCHKEYCFG.H, and then save the configuration file generated (please properly keep it properly for subsequent reference to porting and merging of touch software library).



The contents of S_TOUCHKEYCFG.H are as follows:

S_TouchKeyCFG.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                   4
#define         SOCAPI_SET_TOUCHKEY_CHANNEL                 0x0000000F
unsigned int  code TKCFG[17] = {0,0,0,5,10,3000,200,100,2,0,0,4,0,1,65535,65535,20};
unsigned char code TKChannelCfg[4][8]={
0x03,0x32,0x04,0x08,0x16,0x05,0x02,0xcf,
0x03,0x32,0x04,0x08,0x19,0x05,0x02,0x97,
0x03,0x32,0x04,0x08,0x1b,0x05,0x02,0x66,
0x03,0x32,0x04,0x08,0x1c,0x05,0x02,0xb0,
};
#endif
```

The definitions of configuration files are as follows:

Data Type	Description	Range
SOCAPI_SET_TOUCHKEY_TOTAL	Number of Channel	1-31
SOCAPI_SET_TOUCHKEY_CHANNEL	Corresponding Data Bit of Channel	0x00000001-0xffffffff
TKCFG[0]	Application Type	1-3 0 for spring 1 for spaced 3 for proximity sensing
TKCFG[1]	Key Type	0-10 for single key, 1 for double keys

TKCFG[2]		Remain the default value of 0 unchanged
TKCFG[3]	Key Confirmation Times	3-50
TKCFG[4]		Remain the default value of 10 unchanged
TKCFG[5]	Maximum Output of Key	0-5000
TKCFG[6]		Remain the default value of 200 unchanged
TKCFG[7]		Remain the default value of 100 unchanged
TKCFG[8]		Remain the default value of 2 unchanged
TKCFG[9]		Remain the default value of 0 unchanged
TKCFG[10]		Remain the default value unchanged
TKCFG[11]		Remain the default value unchanged
TKCFG[12]		Remain the default value unchanged
TKCFG[13]		Remain the default value unchanged
TKCFG[14]		Remain the default value of 65535 unchanged
TKCFG[15]		Remain the default value of 65535 unchanged
TKCFG[16]	Noise Value	3-50
TKChannelCfg[][0]		Remain the default value unchanged
TKChannelCfg[][1]		Remain the default value unchanged
TKChannelCfg[][2]		Remain the default value unchanged
TKChannelCfg[][3]	Scan Cycle	0x01-0x20
TKChannelCfg[][4]		Remain the default value unchanged
TKChannelCfg[][5]		Remain the default value unchanged
TKChannelCfg[][6]	Threshold high 8-bit	0x00-0xff
TKChannelCfg[][7]	Threshold low 8-bit	0x01-0xff

The debugging process of touch keys is completed.

If the user needs to fine tune the sensitivity after debugging, change the value of TKChannelCfg[][6] and TKChannelCfg[][7] with the former of the higher 8 bits of the threshold and the latter of the lower 8 bits of the threshold, the lower the value is, the higher the sensitivity will be, vice versa. It is recommended to debug several machines so as to get the compromised effect of parameters and remove the influence of materials on consistency.

5) Additional Functions – Key/Wheel/Slider Hand Feel Simulation Function Testing:

Main Functions: After performing "Start Diagnosis" and "Export Configuration Information", test the hand feel of the key parameters on the upper computer directly and observe if the parameters adapt to the whole machine.

The key type for simulation function testing includes: key, slider and wheel.

The test procedures are shown as follows:

1) After clicking "Start Diagnosis" and "Export Configuration Information", select a certain type of key, take the wheel as an example.

2) Key/Slider/Wheel Test: Select Wheel Test

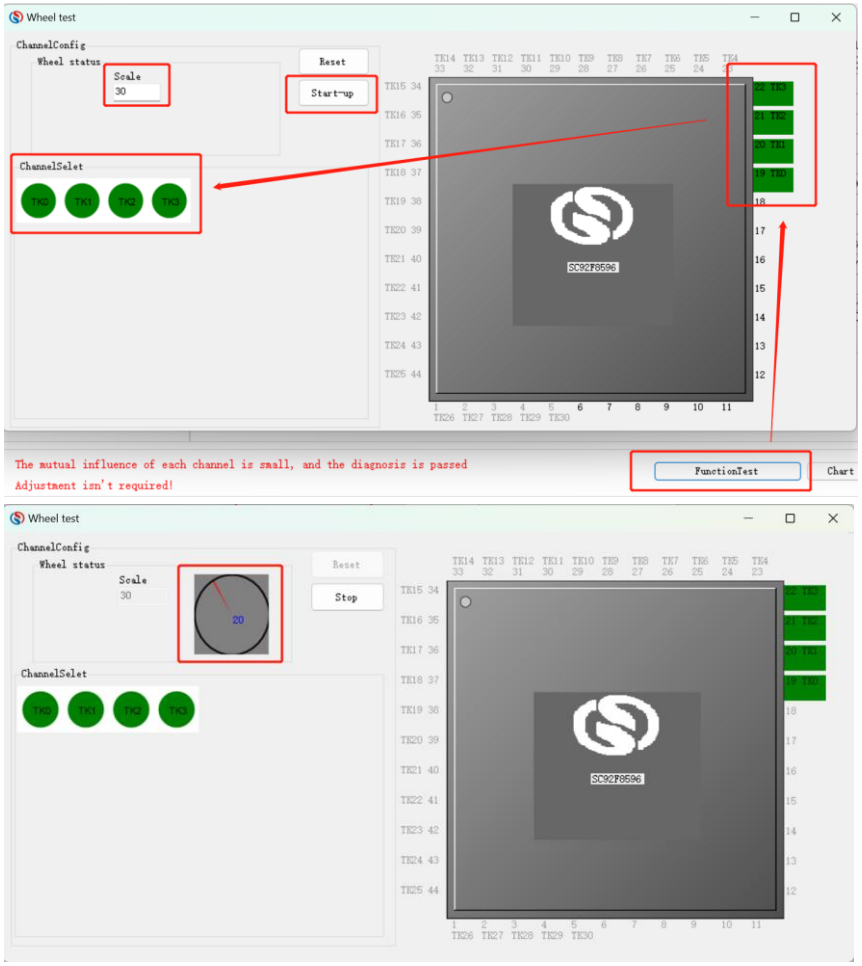
① Set up scale value: the maximum scale of wheel

② Set up wheel channel order: Select the order based on the wheel TK channel order on the hardware, for example, the wheel below is arranged in the order of TK0->TK1->TK2->TK3

③ Click Start and slide the wheel key on the hardware to observe the wheel effects and hand feel on the upper computer.

④ After testing, click Stop button to close the interface.

Note: Reset button is to reset before start.



Notes:

- **"Slider/Wheel Test" and "Key Test" can not be set up repeated or be shared.**
- **After setting up slider/wheel test, select "Key Test" for this TK channel again, it is unable to test single key functions.**
- **To experience slider/wheel/key test function, it is required to update the latest high-sensitivity debugging file.**
- **For key test items, directly click Start with no need to set up scale parameters. Click corresponding TK to observe the key situations on the upper computer.**

3.3 Realizing Function Test of SinOne Software Library

3.3.1 Porting of High-sensitivity Touch Software

1. Introduction of Library File

1) Spring Library File (T1 library for short, SC95F8XXX_HighSensitive_Lib_T1_Vx.x.x.LIB)

2) Spaced Library File (T2 library for short, SC95F8XXX_HighSensitive_Lib_T2_Vx.x.x.LIB)

The following is the brief introduction to T1/T2 library: (4 files contained)

File	Application	Description
SC95F8XXX_HighSensitive_Lib	Library file, to realize the detection algorithm of touch keys	
Sensormethod.h	Head file, to provide the interface function for user to call	The declared functions are available for external call
S_TouchKeyCFG.C	C file, to realize the interaction between the touch parameters and the library	
S_TOUCHKEYCFG.H	Head file, to provide macro for the user to modify the parameters	

2. Resources Used for Lib

Library Series	RAM Occupied Memory	ROM Occupied Memory
(T1 Library) Size occupied of SC95F8XXX_HighSensitive_Lib and S_TouchKeyCFG.C	**data area: 49.3 bytes; Xdata area: 18 bytes; unrelated to the number of keys; all required to use; Xdata area: 15 Bytes for each key; for 3 keys: Data area 49.3 bytes, xdata area 18+3*15=63 byte**	**The size of ROM used by the library is about 3.6K, and adding or reducing several keys may not influence the size basically, and the difference is no more than 200byte**
(T2 Library) Size occupied of SC95F8XXX_HighSensitive_Lib and S_TouchKeyCFG.C	**data area: 59.3 bytes; Xdata area: 10 Bytes; unrelated to the number of keys; all required to use; Xdata area: 15 Bytes for each key; for 3 keys: Data area 59.3 bytes, xdata area 18+3*15=55 byte**	**The size of ROM used by the library is about 3.6K, and adding or reducing several keys may not influence the size basically, and the difference is no more than 200byte**

Note: The memory size of each chip library has little difference, and the specific memory size is subject to SinOne data.

3. Descriptions for Calling Lib API Functions

Function	Application	Description
TouchKeyInit(void)	**Initialize the touch keys**	**1. Call once after power-on and reset; 2. This function configures the user-selected key channel and key parameters by using S_TOUCHKEYCFG.H parameters and initialize the Baseline; 3. The time to execute this function is about 200-500mS, depending on the number of the key, key scan time and auto calibration times; approximately time for every N more keys: 54 uS *N keys for 24M basic frequency; 48 uS *N keys for 16M basic frequency; 45 uS *N keys for 32M basic frequency**
TouchKeyRestart(void)	**Enable the scan of the touch keys**	**1. The user's main program controls when the key scanning is initiated; 2. After the key scanning is initiated and before the scanning of the touch keys is**

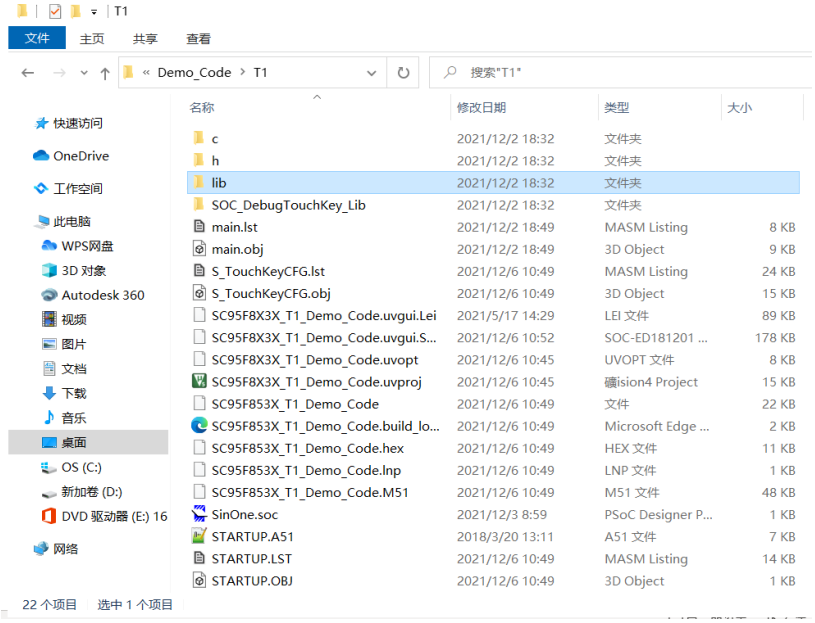
		completed, do not operate the touch key channel, such as IO of touch key channel, or else, the touch key function can not be realized.
Unsigned long int TouchKeyScan(void)	**Process the algorithm of the touch keys**	**1. Call the algorithm after one round scan of the touch key is completed;** **2. TouchKeyRestart() can not be recalled before this function is called by the user; otherwise, the last round of data will be overwritten by current data;** **3 The time to execute this function is about 50uS*N keys @32M and 340 uS*N keys @24M;**

4. Description for Global Variable SOCAPI_ToucKeyStatus
   1) Global variables are declared in the head file S_TouchKeyCFG.c
      ① Unsigned char xdata SOCAPI_ToucKeyStatus;
      ② SOCAPI_ToucKeyStatus Bit7 of 1 indicates that current round of key scanning is completed;
   2) This variable is called in the user's main program
      For if(SOCAPI_ToucKeyStatus&0x80), call TouchKeyScan(void) for algorithm data processing and give the key value;
   3) Be sure to clear the mark before enabling the scanning of the touch keys.
      Clear one round of scanning mark SOCAPI_ToucKeyStatus &=0x7f;
5. Description for Returned Value of LIB API Functions
   1) Returned Value of TouchKeyScan(void) function:
      Bit 1 of the returned value indicates that there is the key in this channel and 0 indicates that there is no key. If double keys are enabled and triggered, two bit positions will be initiated.

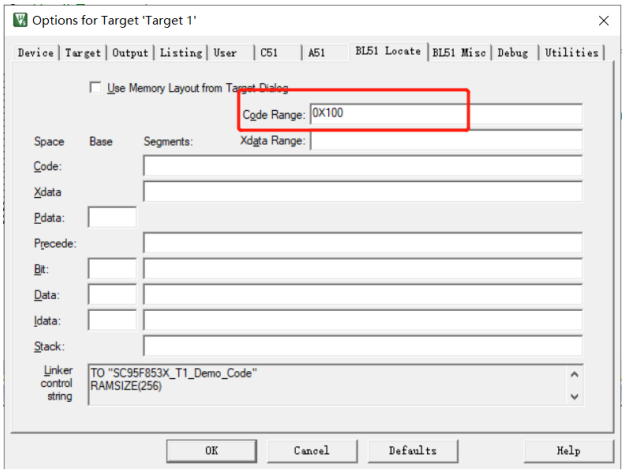
Data Bit		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
Meaning		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	Touch Key State (1: Valid; 0: Invalid)							
Data Bit	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Meaning	TK23	TK22	TK21	TK20	TK19	TK18	TK17	TK16
	Touch Key State (1: Valid; 0: Invalid)							
Data Bit	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Meaning	TK15	TK14	TK13	TK12	TK11	TK10	TK9	TK8
	Touch Key State (1: Valid; 0: Invalid)							
Data Bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Meaning	TK7	TK6	TK5	TK4	TK3	TK2	TK1	TK0
	Touch Key State (1: Valid; 0: Invalid)							

Note: The return type of the function is unsigned long int; TKn is for touch channel, see corresponding specifications for details.

6. Open the project file and copy "lib" folder in the project folder



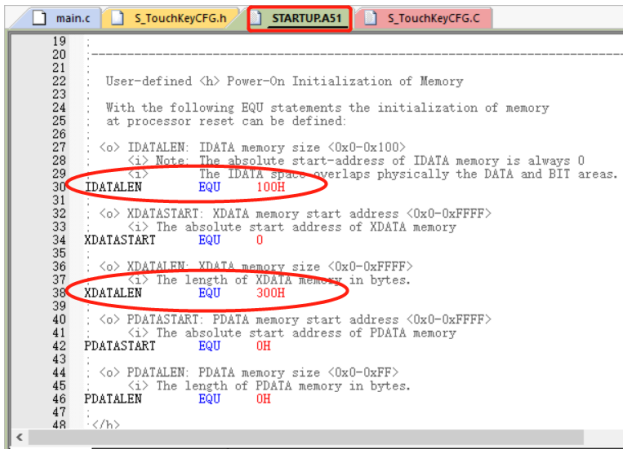
7. Open the project file in Keil, and set up Code range and XDATALEN EQU xxxH



For the objectives of the settings, see SinOne MCU notes Vx.xx.PDF file.

(There is no need to set up this parameter for partial 92/95F series chips, please read corresponding specifications carefully)

8. Set up XDATALEN



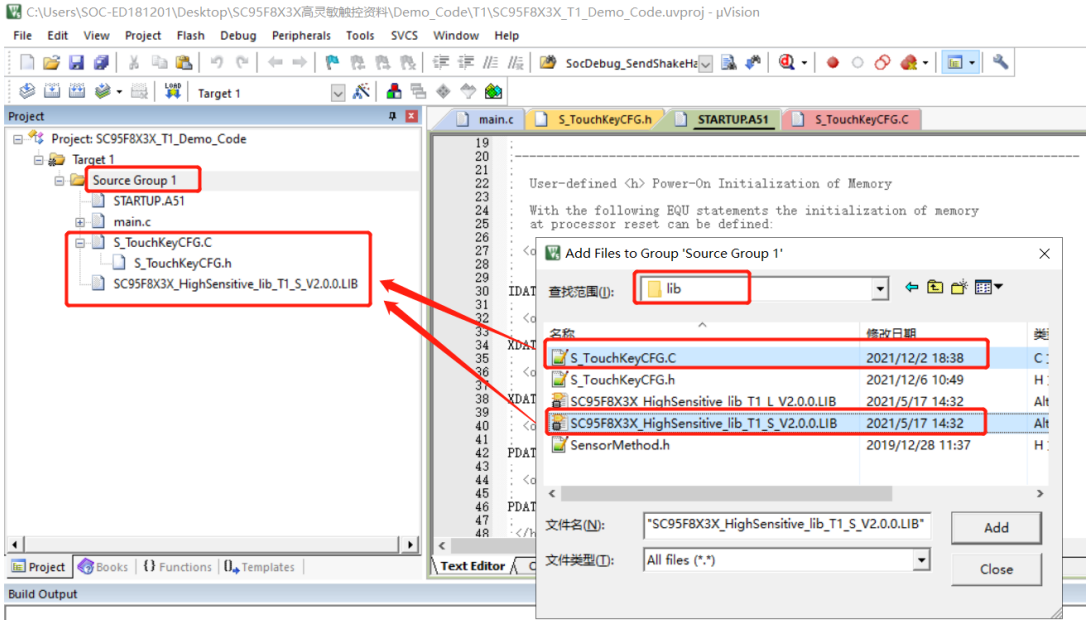
Note: It is used in STARTUP.A51 to clear the external XData; for XData size of specific model, see

corresponding specifications.

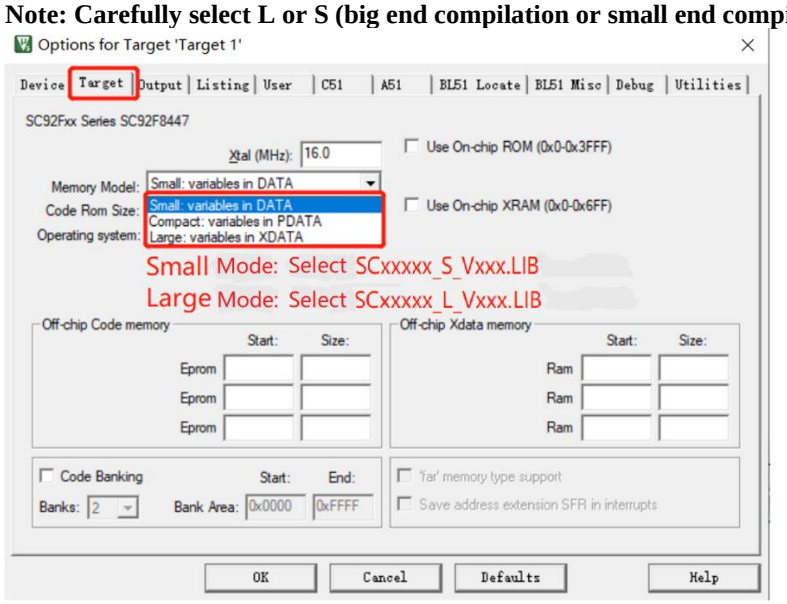
9. Add the library file LIB and S_TouchKeyCFG.C file to the project

1) Add the library file LIB and S_TouchKeyCFG.C to the project from LIB folder of SinOne library data.

The following figure takes T1 library as an example: (Operations of T2 library is the same)

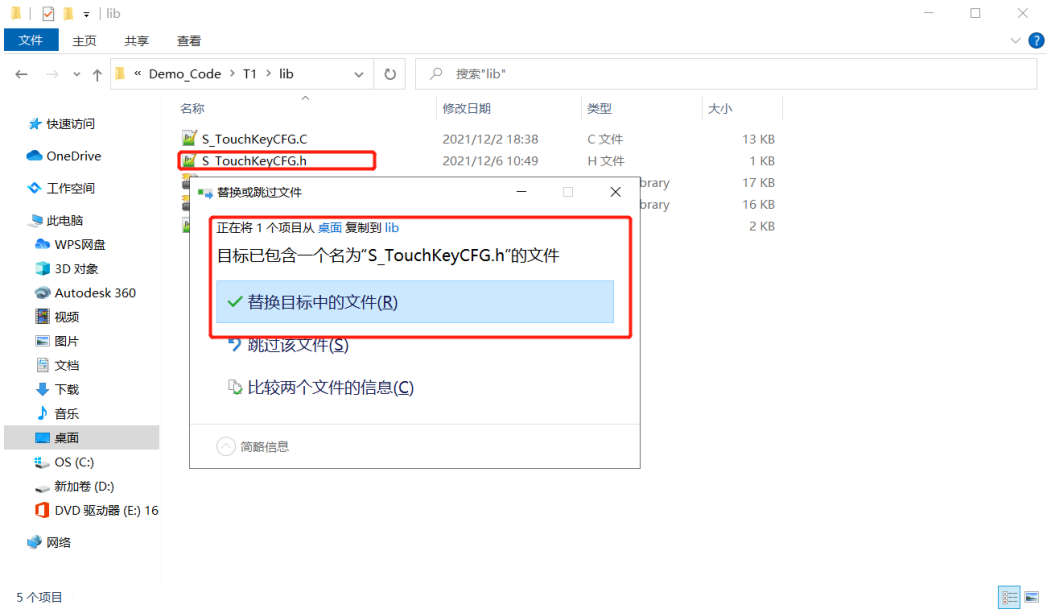


Note: Carefully select L or S (big end compilation or small end compilation), as shown in the figure below:



10. Add the head file reference to the main program file

11. Replace the configuration file S_TOUCHKEYCFG.H generated by TK touch debugging upper computer in LIB folder



A complete SinOne touch high-sensitivity software library has been added to the project.
Note: It is required to set IO interface of TK as strong push-pull output high in the application program.

3.3.2 Porting of High-reliability Touch Software

1. Introduction to Library File
1) High-reliability File (SC95F8XXX_HighReliability_Lib_T1_Vx.x.x.LIB)

File	Application	Description
SC95F8XXX_HighReliability_Lib_T1_Vx.x.x.lib	Library file, to realize the detection algorithm of touch keys	
Sensormethod.h	Head file, to provide the interface function for user to call	The declared functions are available for external call
S_TouchKeyCFG.C	C file, to realize the interaction between the touch parameters and the library	No need to modify
S_TOUCHKEYCFG.H	Head file, to provide the macro definition of TouchKey function	The user can modify partial macro definitions to change the settings of TouchKey register

2. Resources used for Lib
1) The size of resources occupied by LIB library (RAM and ROM)

Library Series	RAM Occupied Memory	ROM Occupied Memory
Size occupied of SC95F8XXX_Reliability_Lib and S_TouchKeyCFG.C	data area: 40.2 bytes; Xdata area: 23 Bytes; unrelated to the number of keys; all required to use; Xdata area: 13 Bytes for each key; for 5 keys: Data area 40.2 bytes, xdata area 23+5*13=88 byte	The size of ROM used by the library is about 3.2K, and adding or reducing several keys may not influence the size basically, and the difference is no more than 200byte

Note: The memory size of each chip library has little difference, and the specific memory size is subject to SinOne data.

2) Interrupt: Only use TK for interruption and the priority by default

Interrupt Source	Interrupt Priority	Interrupt Vector	Inquire Priority	Interrupt Number (C51)	Mark Clear Mode
TK	低	005BH	12	11	H/W Auto

Note: The library uses touch interruption with low priority and there is no nested function in the interrupt service program with the execution time of 6.8us.

3) Meanings of the Parameters in the S_TouchKeyCFG.C file

Parameter	Type	Value	Description
SOCAPI _SET_TKCFG1	**Unsigned 8-bit integer constant**	**The default value is recommended**	**Description for the control register TKCFG1**
SOCAPI _SET_TKCFG2	**Unsigned 8-bit integer constant**	**CTIME = 0x03 to 0x0f is recommended**	**Description for the control register TKCFG2**
SOCAPI _SET_TKCFG3	**Unsigned 8-bit integer constant**	**The default value is recommended**	**Description for the control register TKCFG3**
SOCAPI _SET_TouchKey_Total	**Unsigned 32-bit integer constant**	**1~23**	**Set the number of touch keys**
SOCAPI_SET_TouchKey_Channel	**Unsigned 32-bit integer constant**	**Depending on the touch key channel selected by the user**	**Select the touch key channel; Bit0~Bit23 corresponds to TK0~TK23; 1 is for TK channel; 0 is for IO; 0000 0000 0000 0101: TK0 and TK2 are for TK, others for IO; The number of the channel selected by SOCAPI _SET_TouchKey_Channel shall be the same as that of SOCAPI_SET_TouchKey_Total**
SOCAPI _SET_TouchKeyCONFIRM_CNT	**Unsigned 8-bit integer constant**	**Recommendation: 5-40, 10 times is preferred**	**Key confirmation time. This key that is scanned for successive SOCAPI_SET_TouchKey CONFIRM_CNT rounds can be considered as being pressed by the key; the larger value may result in the slower key response;**
SOCAPI _SET_NOISE_Threshold	**Unsigned 8-bit integer constant**	**Recommendation: 20-40**	**Set the noise value**
SOCAPI _SET_FINGER_Threshold	**Unsigned 8-bit integer constant**	**Set according to the data collected**	**Set the finger threshold. Use SOC Touch KeyTool to collect data and take *60% of diff value after pressing with 10mm-diameter copper pillar, and keep the ration of the finger threshold to the noise value be more than 5. Match the number of group elements with that of**

			the touch key configured

3 Description for Calling Lib API S_TouchKeyCFG.c

Function	Application	Description
TouchKeyInit(void)	**Initialize the touch keys**	**1. Call once after power-on and reset;** **2. This function configures the user-selected key channel and key parameters by using S_TOUCHKEYCFG.H parameters and initialize the Baseline;** **3. The time to execute this function depends on the number of the key, key scan time and auto calibration times; approximately time for every N more keys:** **12M basic frequency: 54*N keys** **16M basic frequency: 48*N keys** **32M basic frequency: 45*N keys**
TouchKeyRestart(void)	**Enable the scan of the touch keys**	**1. The user's main program controls when the key scanning is initiated;** **2. After the key scanning is initiated and before the scanning of the touch keys is completed, do not operate the touch key channel, such as IO of touch key channel, or else, the touch key function can not be realized.**
Unsigned long int TouchKeyScan(void)	**Process the algorithm of the touch keys**	**1. Call the algorithm after one round scan of the touch key is completed;** **2. TouchKeyRestart() can not be recalled before this function is called by the user; otherwise, the last round of data will be overwritten by current data;** **3 The time to execute this function: The time for algorithm execution is positively related to the number of the keys.。** **N Keys about (240*N) us@12M.**

4. Descriptions for Global Variable SOCAPI_ToucKeyStatus

1) Global variables are declared in the head file S_TouchKeyCFG.c

Unsigned char xdata SOCAPI_ToucKeyStatus;

SOCAPI_ToucKeyStatus Bit7 of 1 indicates that current round of key scanning is completed;

2) This variable is called in the user's main program

For if(SOCAPI_ToucKeyStatus&0x80), call TouchKeyScan(void) for algorithm data processing and give the key value;

3) Be sure to clear the mark before enabling the scanning of the touch keys.

Clear one round of scanning mark SOCAPI_ToucKeyStatus &=0x7f;

5. Description for Returned Value of LIB API Functions

1) Returned Value of TouchKeyScan(void) function:

Bit 1 of the returned value indicates that there is the key in this channel and 0 indicates that there is no key. The details are as follows:

Data Bit		Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
Meaning		TK30	TK29	TK28	TK27	TK26	TK25	TK24
	Touch Key State (1: Valid; 0: Invalid)							

Data Bit	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
Meaning	TK23	TK22	TK21	TK20	TK19	TK18	TK17	TK16
	Touch Key State (1: Valid; 0: Invalid)							
Data Bit	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Meaning	TK15	TK14	TK13	TK12	TK11	TK10	TK9	TK8
	Touch Key State (1: Valid; 0: Invalid)							
Data Bit	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Meaning	TK7	TK6	TK5	TK4	TK3	TK2	TK1	TK0
	Touch Key State (1: Valid; 0: Invalid)							

Note: The return type of the function is unsigned long int; TKn is for touch channel, see corresponding specifications for details.

6. Maximum Output Time of a Key that Remains Valid

```
#define SOCAPI_SET_KEY_CONTI_TIME 1000      // Maximum output time of a key that remains valid ranging from 0 to 5000 with the default value of 1000, Output Time = 1000* Unit Scanning Time per Round (such as 10ms) =10S
```

Notes:

1) TK0-TK31 refer to the touch key channels; see SC95F8XXX specifications for details.

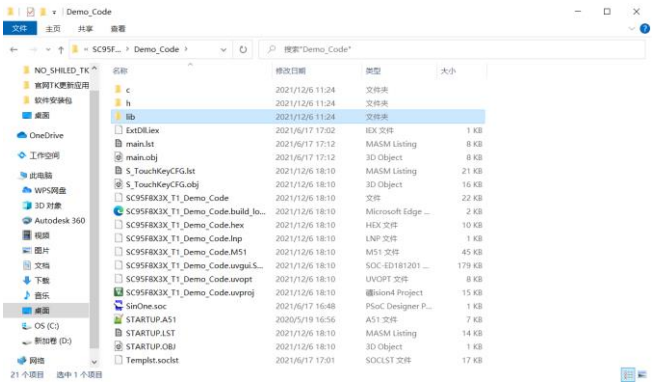
2) In the actual application, the user can eliminate the dithering repeatedly to enhance the reliability; the key value is valid only when it is read for two to five times consecutively.

3) It can also be judged by reading the key value:

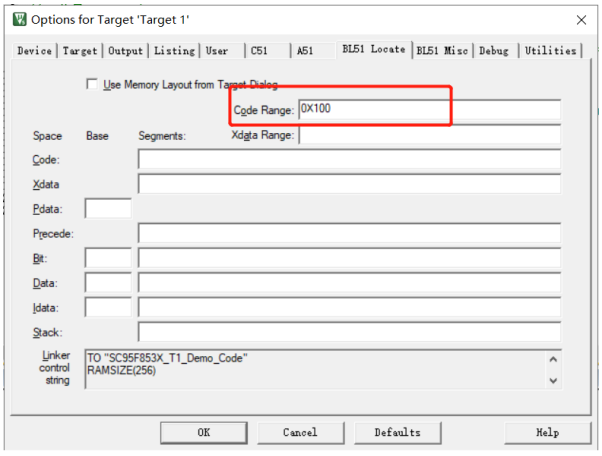
① Double click: Press the key twice in 1S;

② Long press: Press for 2S;

7. Copy “lib” folder in the project folder



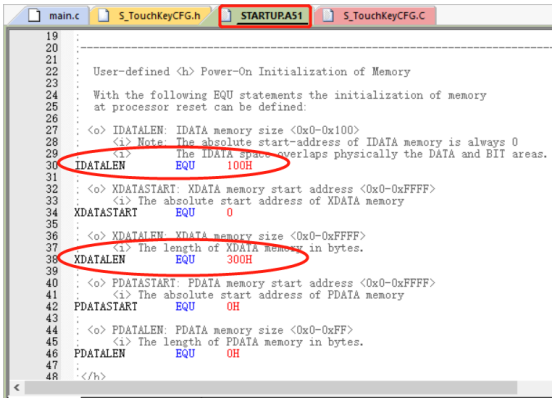
8. Open the project file in Keil, and set up Code range and XDATALEN EQU xxxH



For the objectives of the settings, see SinOne MCU notes Vx.xx.PDF file.

(There is no need to set up this parameter for partial 92/95F series chips, please read corresponding specifications carefully)

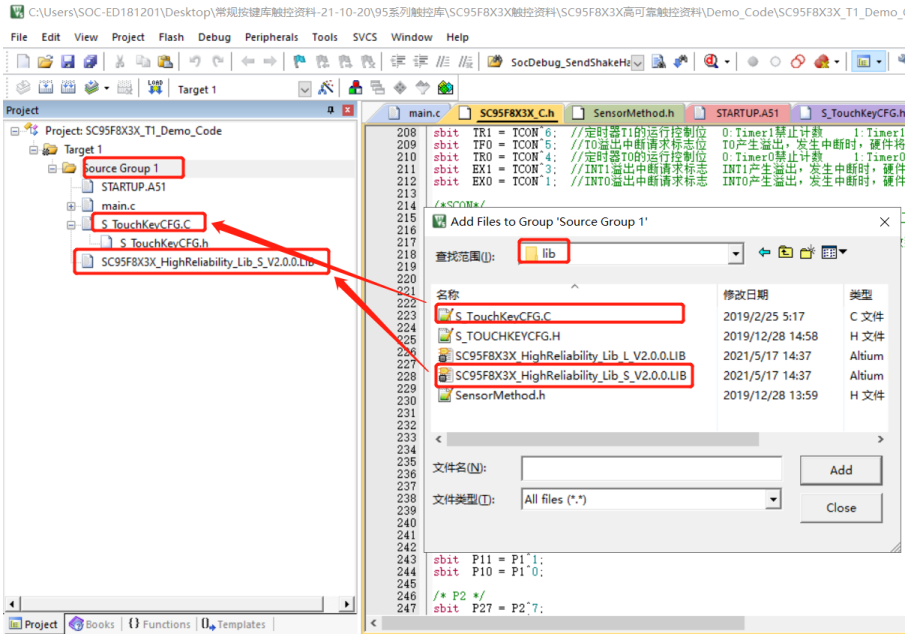
9. Set up XDATALEN



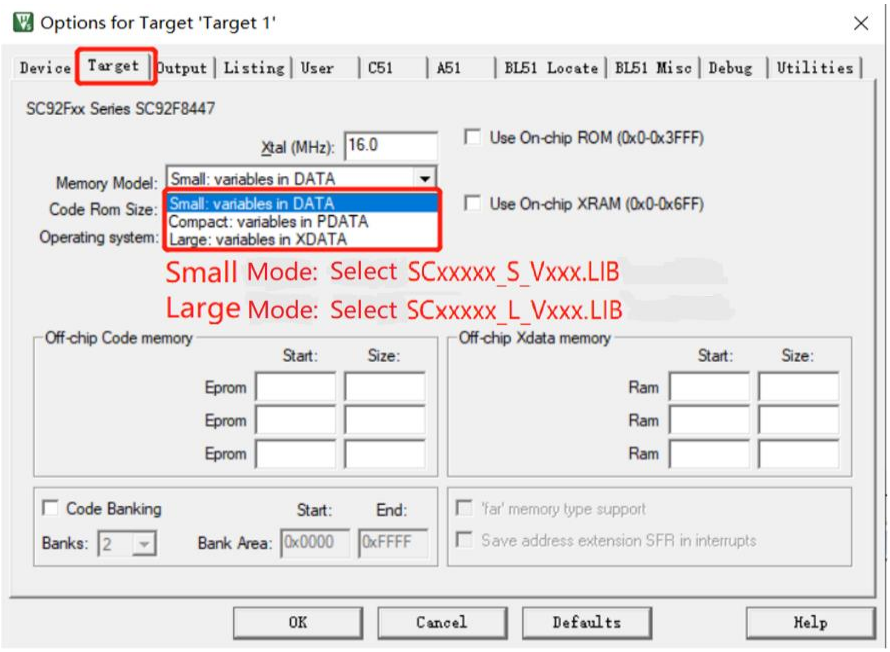
Note: It is used in STARTUP.A51 to clear the external XData; for XData size of specific model, see corresponding specifications.

10. Add the library file LIB and S_TouchKeyCFG.C file to the project

1) Add the library file LIB and S_TouchKeyCFG.C to the project from LIB folder of SinOne library data.

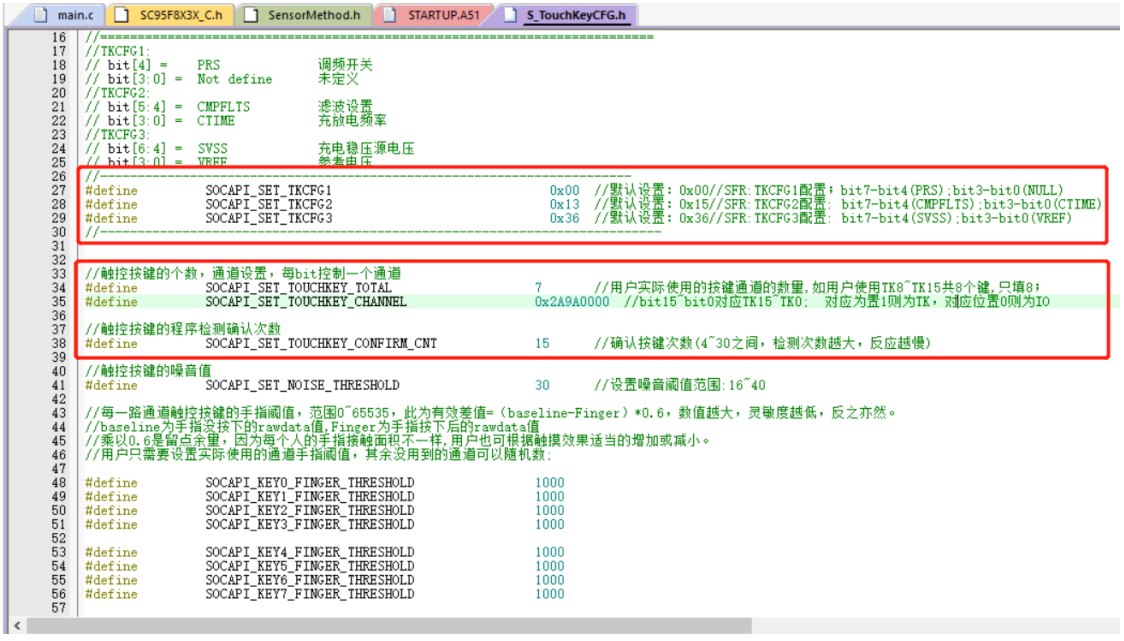


Note: Carefully select L or S (big end compilation or small end compilation), as shown in the figure below:



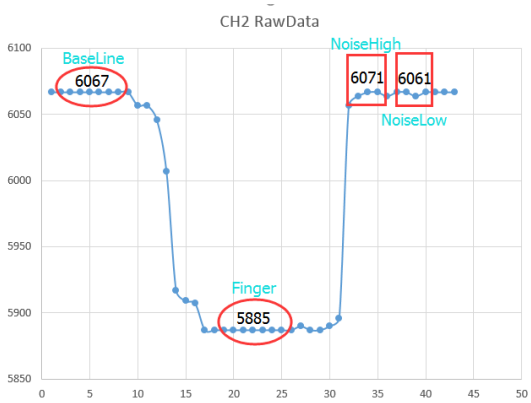
11. Add the head file reference to the main program file

12. Modify the parameters (TKCFG1, TKCFG2 and TKCFG3 recorded in the steps of debugging high-reliability touch parameters) and the channel and number of the touch key in S_TouchKeyCFG.h and set up the times that a touch key is valid



13. Calculate the noise threshold and finger threshold according to RAW DATA in the debugging steps and modify them in S_TouchKeyCFG.h

1) Calculate the finger threshold and noise threshold



Calculate the finger threshold:

① Average Baseline is 6067 when there is no key;
Average finger threshold is 5885 when there is a key;

② The data change: Baseline-Finger=6067-5885=182；

③ SOCAPI_KEY3_FINGER_THRESHOLD: Baseline-Finger; So the theoretical value of SOCAPI_KEY3_FINGER_THRESHOLD: 182；

Considering the contact surface of the finger, the theoretical value*0.6 is recommended, so the valid finger threshold is 182*0.6=109;

Calculate the noise threshold:

① Average Baseline is 6067 when there is no key;
Average finger threshold is 5885 when there is a key;

② Peak value when there is no key NoiseHigh=6071,NoiseLow =6061

③ Data change: 6071-6061=10；SOCAPI_SET_NOISE_THRESHOLD: 10;

Collect the noise threshold of all touch channels, and take the largest value as the noise threshold of all channels, ranging from 20 to 40.

2) Modify the noise threshold and finger threshold in S_TouchKeyCFG.h

A complete SinOne touch high-reliability software library has been added to the project.
Note: It is required to set IO interface of TK as strong push-pull output high in the application program.

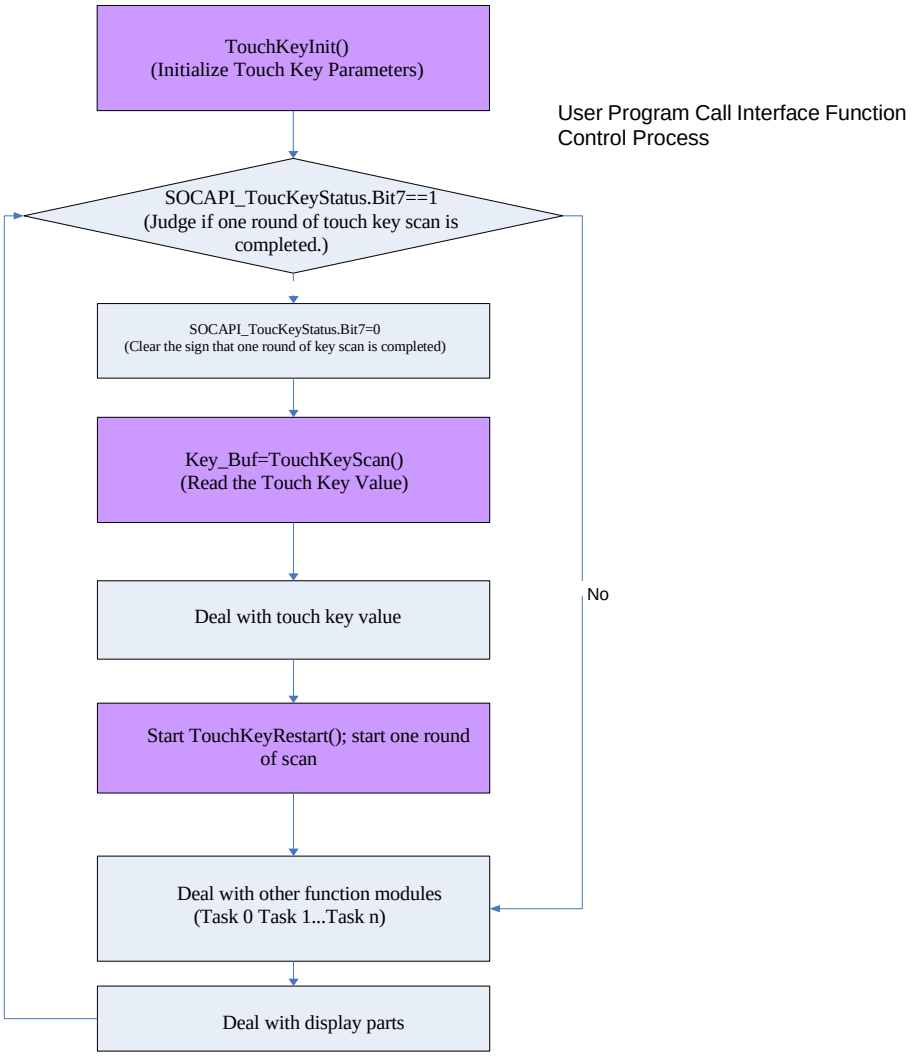
3.4 Complete the Integration of User Program and SinOne Touch Software Library

3.4.1 High-sensitivity Touch Software and User Program

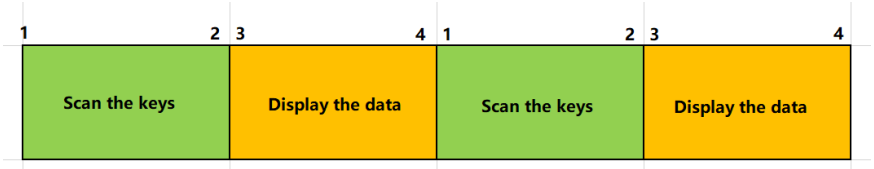
1) Overall Structure Relation between Main Program and Library File
① Add the library file to the project, include specific head file in the user program and call the interface functions in the library to add the touch key functions.
② Library functions run only when the main program is called. The library file will occupy ROM, RAM, register, interrupter and other resources without occupying timer.
③ The library functions are only for touch key functions, and other control functions have to be dealt with by the user, such as input/output, LED, digital display, communication, etc.
2) Call Process of Library Files **(The call process of spring and spaced library is different, please read it carefully)**
The user can call the interface functions of library files via a certain process to obtain the key value of the touch key.

Call Process of Spring Library File (T1 library for short)
① Set corresponding IO of TK as strong push-pull output high.
② The main program calls the interface function “TouchKeyInit()” to configure the parameters of touch key channel and initialize the Baseline;

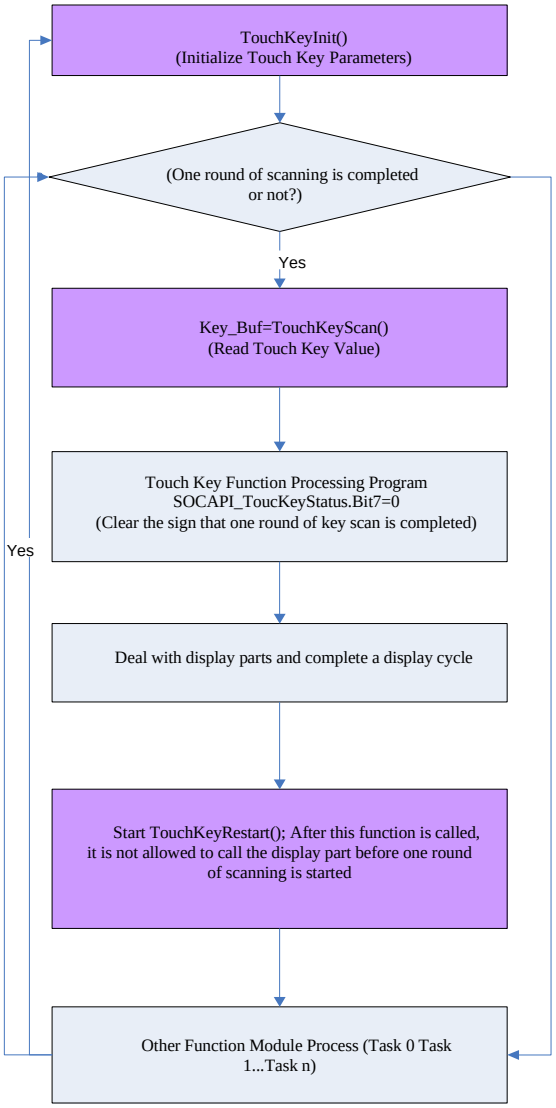
③ The main program views the global variable SOCAPI_TouchKeyStatus&0x80 to judge if one round of touch key scan is completed;
④ The main program calls the interface function “TouchKeyScan()” to read the touch key value;
⑤ The main program calls “TouchKeyRestart()” to start new round of scan.
(The purple part in the figure below refers to the library file, and others represent user’s programs)



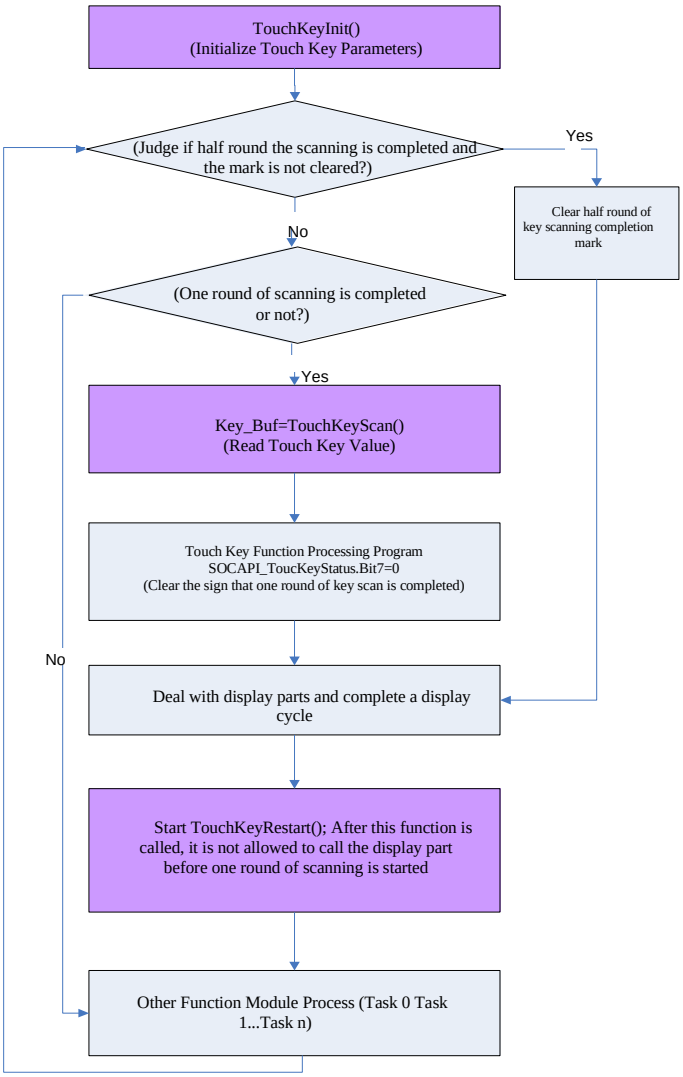
Call Process of Spaced Library File (T2 library for short)
① Set corresponding IO of TK as strong push-pull output high.
② The main program calls the interface function “TouchKeyInit()” to configure the parameters of TouchKey channel and initialize the Baseline;
③ If the number of keys is more than 8, the main program will judge if the half round of touch key scanning has been completed by checking the global variable SOCAPI_TouchKeyStatus&0x40; if it is completed, go to complete the display of next cycle and the scanning of the second half round of touch key.
④ The main program views the global variable SOCAPI_TouchKeyStatus&0x80 to judge if one round of TouchKey scan is completed;
⑤ The main program calls the interface function “TouchKeyScan()” to read the TouchKey value;
⑥ What needs to be emphasized in particular is that, after calling TouchKeyRestart() to start scanning the keys, do not display the data before one round or half round of the scanning is completed.



(The purple part in the figure below refers to the library file, and others represent user's programs)



User Program Call Interface Function Control Process (Number of Keys less than 8)



User Program Call Interface Function Control Process (Number of Keys more than 8)

3) Timing Relationship between Main Program and Library File

Running the touch key library consumes partial IC resources and time, to perfectly integrate the user's program and library program, the main program shall comply with the following requirements:

① Provide ROM, RAM, time and other resources for library running;

② After starting the key scanning and before completing one round of scan, do not perform any operations to the touch key channel;

If the touch key channel is output IO; or else, the touch key function will be disabled;

③ Provide sufficient stack depth for main program and library functions;

④ Data conversion from TouchKey scanning is realized during the process of TK interruption, but the data algorithm is completed in the main program. The user needs to call the library function in a reasonable frequency to avoid missing the key actions;

Notes for Software Integration:

① Running Time:

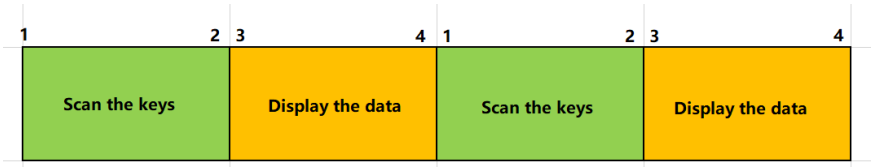
TouchKeyInit(void): The algorithm execution time will be increased/decreased with the number of keys selected, 200~500ms@12M;

TouchKeyScan(void): The time to execute this function is related to the basic frequency of different chips, please refer to the data in the table of 3.3

② Overall Code Testing
After the user completes the program call, please test the performance of related functions in detail to avoid software conflict. If any exception occurs, look for causes in the program flow, call timing, time allocation, stack, ROM/RAM/INT and other resources.
③ Suggestions for machine debugging: Due to the difference of component performance, it is recommended to test more PCBs after one piece of PCB has been debugged, so as to get the compromised effect of parameters and remove the influence of materials on consistency.

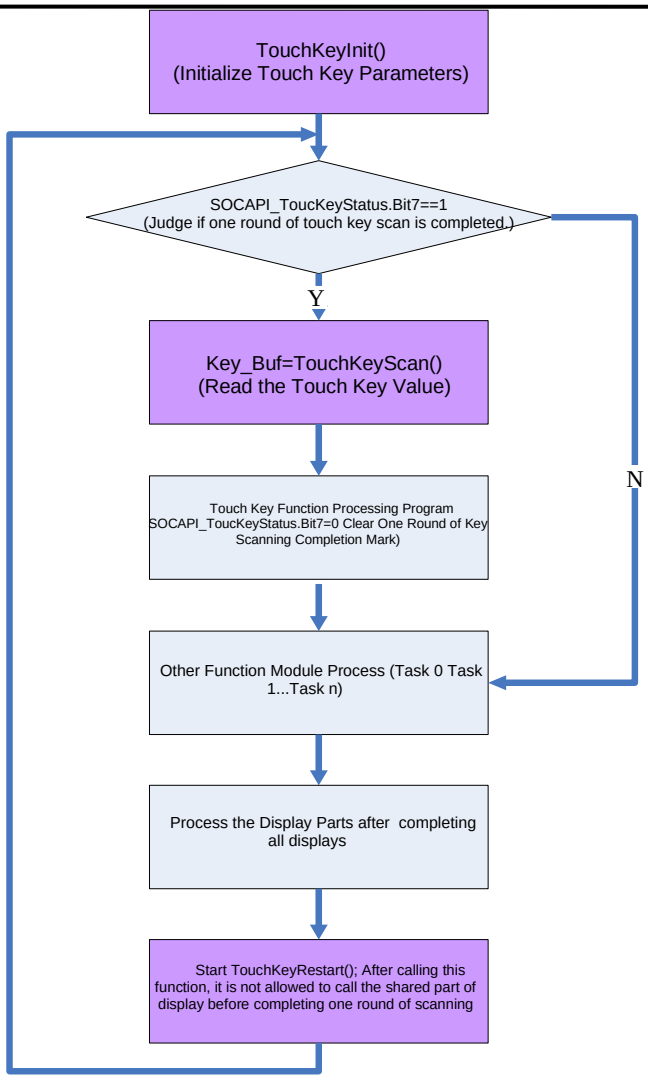
3.4.2 High-reliability Touch Software and User Program

1) Overall Structure Relation between Main Program and Library File
① Add the library file to the project, include specific head file in the user program and call the interface functions in the library to add the touch key functions.
② Library functions run only when the main program is called. The library file will occupy ROM, RAM, register, interrupter and other resources without occupying timer.
③ The library functions are only for touch key functions, and other control functions have to be dealt with by the user, such as input/output, LED, digital display, communication, etc.
2) Call process of Library Files
The user can call the interface functions of library files via a certain process to obtain the key value of the touch key.
① Set corresponding IO of TK as strong push-pull output high.
② The main program calls the interface function "TouchKeyInit()" to configure the parameters of TouchKey channel and initialize the Baseline;
③ The main program views the global variable SOCAPI_TouchKeyStatus&0x80 to judge if one round of touch key scan is completed;
④ The main program calls the interface function "TouchKeyScan()" to read the touch key value;
⑤ What needs to be emphasized in particular is that, if the user shares Touchkey and LED, when calling TouchKeyRestart() to start scanning the keys, do not display the data before the mark of SOCAPI_TouchKeyStatus & 0X80 appears.



1. Start TouchKeyRestart() to scan the keys.
2. If the bit7 of SOCAPI_TouchKeyStatus is set, the round of keyboard scanning is over.
3. Start up display.
4. All shows are done.
1. Start TouchKeyRestart() again and repeat.

(The purple part in the figure below refers to the library file, and others represent user's programs)



User Program Call Interface Function Control Process

3) Timing Relationship between Main Program and Library File

Running the touch key library consumes partial IC resources and time, to perfectly integrate the user's program and library program, the main program shall comply with the following requirements:

① Provide ROM, RAM, time and other resources for library running;

② After starting the key scanning and before completing one run of scan, do not perform any operations to the touch key channel;

If the touch key channel is output IO; or else, the touch key function will be disabled;

③ Provide sufficient stack depth for main program and library functions

④ Data conversion from TouchKey scanning is realized during the process of TK interruption, but the data algorithm is completed in the main program. The user needs to call the library function in a reasonable frequency to avoid missing the key actions;

Notes for Software Integration:

① Running Time:

TouchKeyInit(void): The algorithm execution time will be increased/decreased with the number of keys selected, 200~500ms@12M;

TouchKeyScan(void): The time to execute this function is related to the basic frequency of different chips, please refer to the data in the table of 3.3

② Overall Code Testing:

After the user completes the program call, please test the performance of related functions in detail to avoid

software conflict. If any exception occurs, look for causes in the program flow, call timing, time allocation, stack, ROM/RAM/INT and other resources.

③ Suggestions for machine debugging: Due to the difference of component performance, it is recommended to test more PCBs after one piece of PCB has been debugged, so as to get the compromised effect of parameters and remove the influence of materials on consistency.

3.4.3 Notes

1) For single-side PCB, use spring Touchkey. Because its side can also form the electric field with fingers, and using spring TouchKey can obtain higher flexibility than using copper clad TouchKey on PCB.
2) The wire length from TouchKey pad to IC pin should not be wound too far; avoid the coupling capacitance between wires and between wires and other high-frequency signal line.
3) The sensitivity is proportional to the area of TouchKey pad and inversely proportional to the thickness of the enclosure. Select the appropriate touch area based on the enclosure thickness and size. Generally, glass enclosure has higher penetration than the plastics.
4) A certain distance shall be reserved between the TouchKey pads to guarantee that finger touch will not cover 2 TouchKey pads and prevent too large parasitic capacitance of TouchKey pad.
5) The reference capacitance is the charging/discharging capacitance of SinOne TouchKey induced circuit and the important component to realize TouchKey function. It can guarantee normal work of touch circuit with the capacitance range of 472-104, and 103 capacitance is recommended. There is no special requirements on materials.
6) Set IO interface of TK as strong push-pull output high.

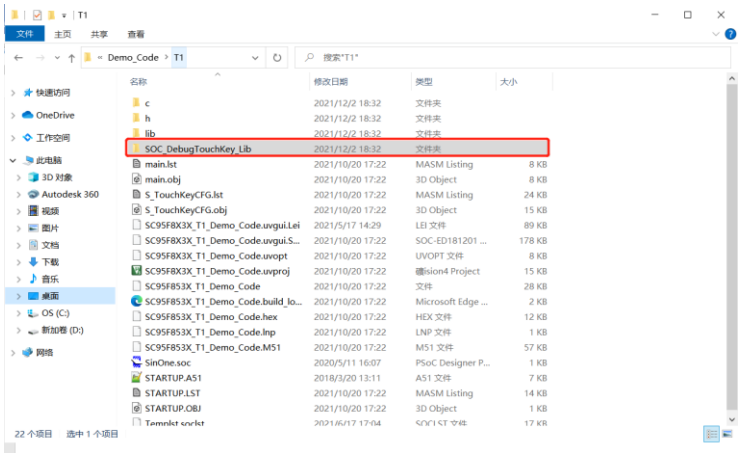
For more Layout notes, please refer to: Design Points for SinOne Touch Key MCU PCB.

3.5 Additional Functions – Dynamic Debugging Functions

Main Functions: Use SinOne touch debugging upper computer software to view the real-time data, so as to help the user to conduct the overall evaluation of the system, understand the actual operation situations and analyze any anomalies, etc.

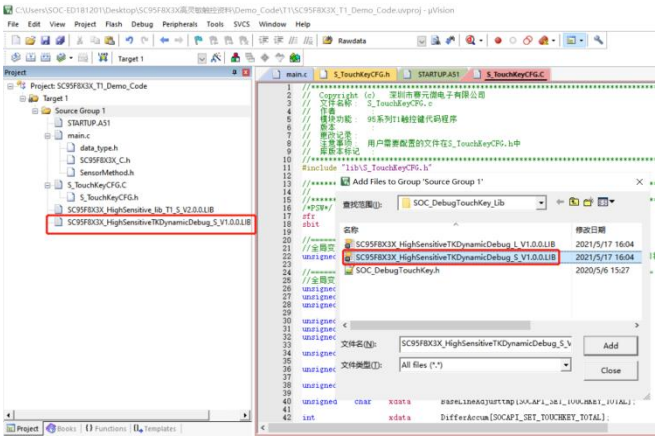
3.5.1 High-sensitivity Dynamic Debugging Steps

1) Place SOC_DebugTouchKey_Lib folder in the root directory of the project

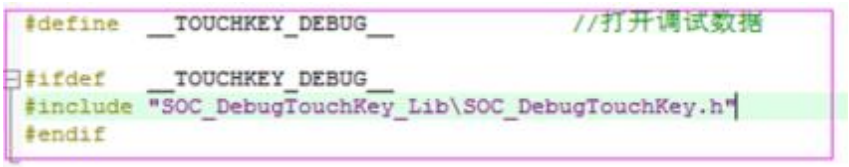


2) Add SC95F8XXX_HighSensitiveTKDynamicDebug_S/L_Vx.x.x.LIB in the user project

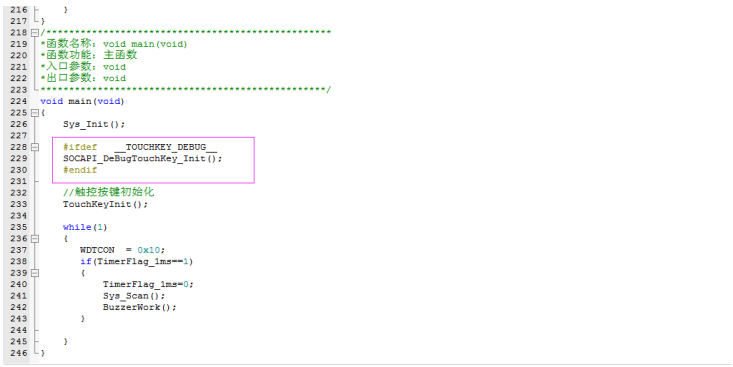
S/L –> refers to compile dynamic debugging library lib with small end/large end compilation, which shall be consistent with that of the touch library S/L, as shown in the figure below.



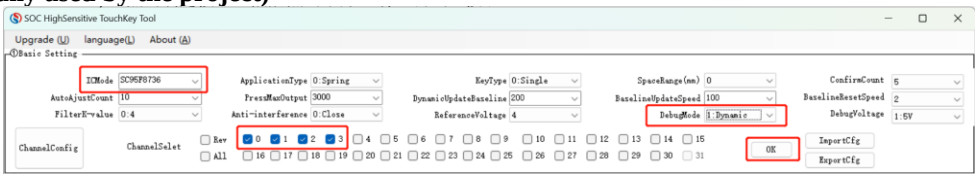
3) Include the head file in the main.c



4) Call SOCAPI_DeBugTouchKey_Init in the main function for initialization and program to the chip after compilation

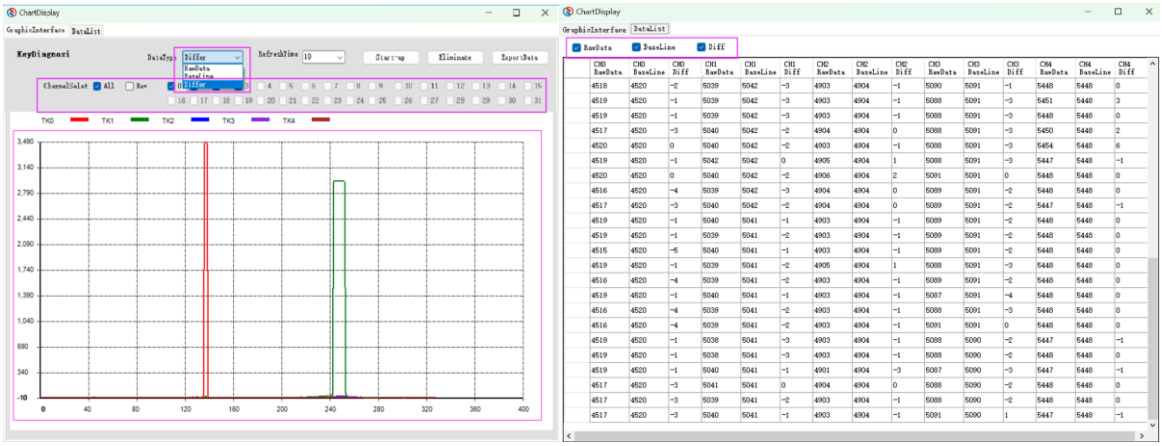


5) Open Touch Key Tool Menu and select high-sensitivity touch key and the chip type corresponding to the actual chip, then select dynamic debugging for debugging mode and check TK channel **(consistent with the channel actually used by the project)**



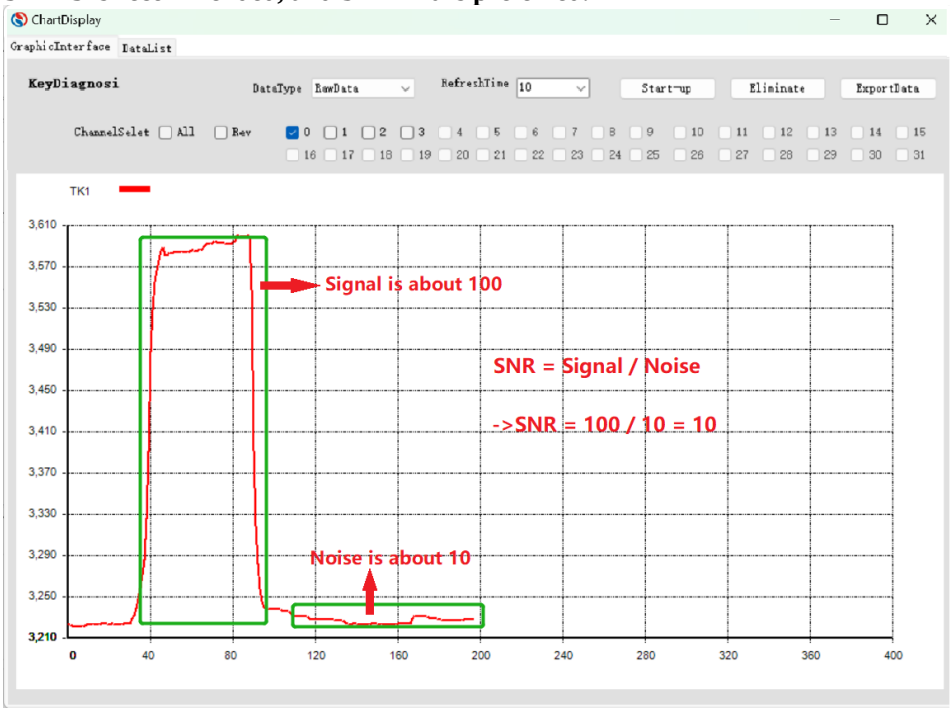
6) Click OK button, then click “Dynamic Debugging” button at the bottom

7) In the dynamic debugging interface, you can select “Data Type” to view the data to be viewed, “Channel Select” to check the channel to be viewed, view the real-time data in the figure and click “Data List” to view the figure data



Note: When observing data through dynamic debugging, it is necessary to confirm if the SNR meets the operating conditions.

SNR>5 is recommended, and SNR >10 is preferred.



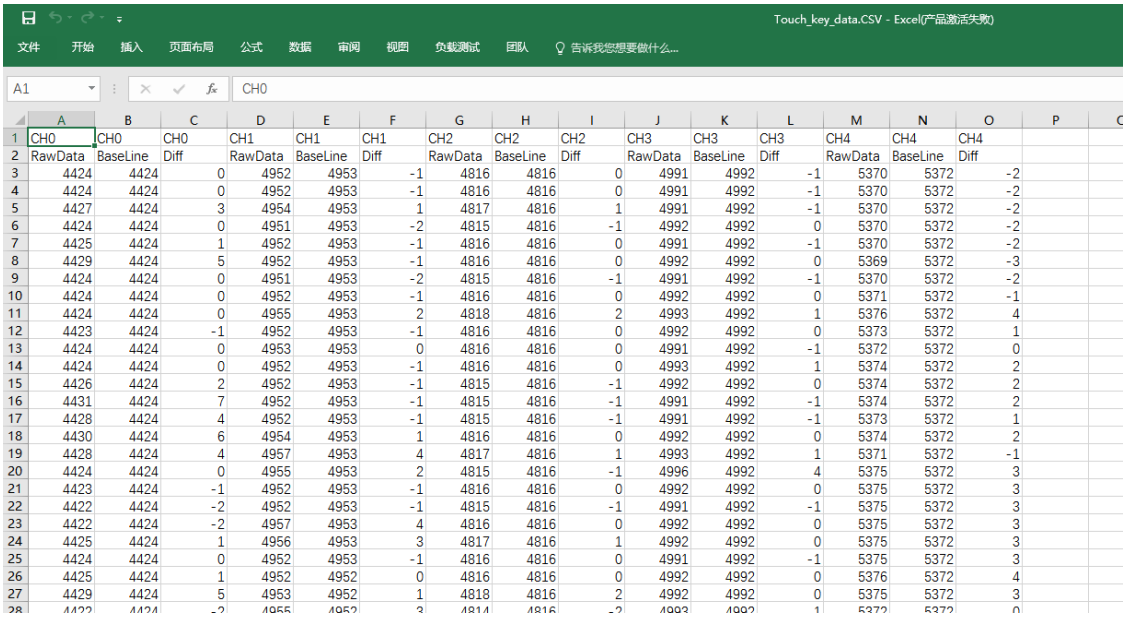
SNR Calculation Mode: Dynamic debugging observation in process

Noise amplitude: The difference between the maximum and minimum value of RawData when no finger is pressed in the standing state

Signal amplitude: The average value of RawData when the finger is pressed

SNR Calculation Mode: SNR = signal/noise, as shown in the figure above, SNR=10

8) Click “Export Date” to real-time collect data in CSV format



Touch_key_data.CSV - Excel(产品激活失败)

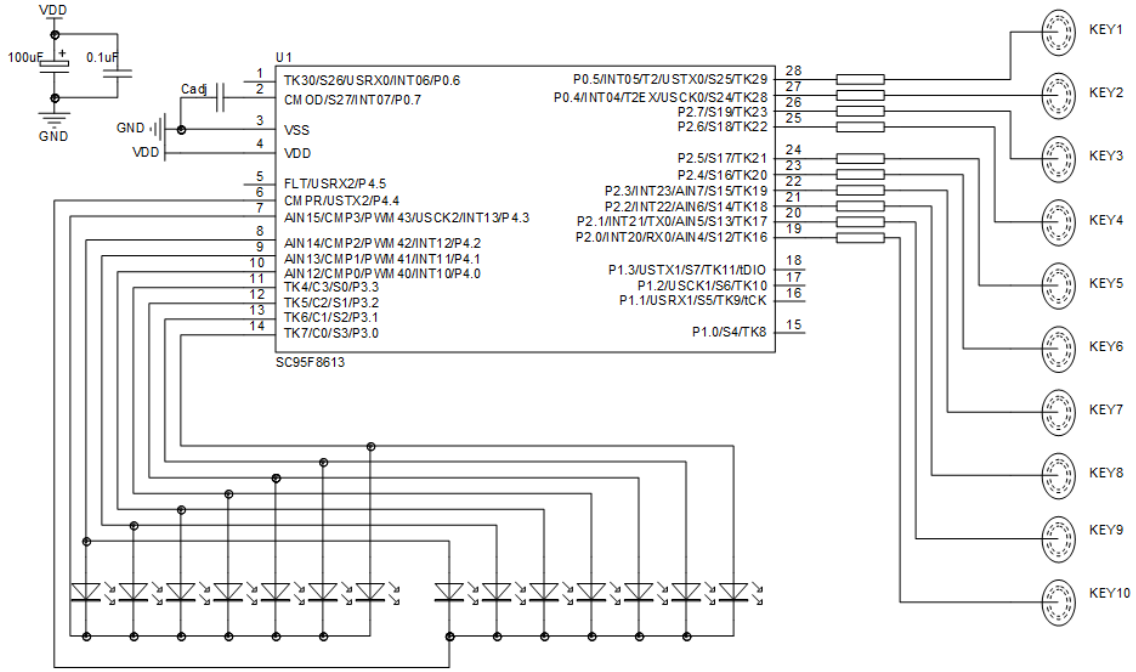
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	CH0	CH0	CH0	CH1	CH1	CH1	CH2	CH2	CH2	CH3	CH3	CH3	CH4	CH4	CH4
2	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff	RawData	BaseLine	Diff
3	4424	4424	0	4952	4953	-1	4816	4816	0	4991	4992	-1	5370	5372	-2
4	4424	4424	0	4952	4953	-1	4816	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

9) Notes for Dynamic Debugging

- **For UART (UART0 or SSI) resources on the programming interface are used in the dynamic debugging library, the user program must first mask partial UART (UART0 or SSI) programs, including initialization, interrupt service functions. No UART (UART0 or SSI)-related register and operating-related pins can be operated, among them, the model with UART0 on the programming port also uses Timer2 as the baud rate generator , so Timer2 can not be used either.**
- **The channel checked for debugging the main interface shall be consistent with that in the practical use.**
- **43byte idata and 501byte ROM resources are occupied for the dynamic debugging library, please reserve sufficient resources to guarantee the debugging program work normally.**

4. Appendix

I. Schematic Diagram for Application Reference (Taking SC95F8613 as an example)



II. Software Reference Example

1) For the items that share TouchKey and LED, when calling TouchKeyRestart() to start scanning the keys, do not display the data before the mark of SOCAPI_TouchKeyStatus & 0X80 appears;
2) For the items that do not share TouchKey and LED, it is unnecessary to display and scan the keys separately.

Procedure instructions are attached below

For items that share TouchKey and LED:

Main.c

```
void main()
{
        unsigned char result =0;
        SegOutState;        // Initialize IO interface, displayed SEG, COM pins
        ComOutState; ComAllClose; SegAllClose;
        TestIOPortOut;      //P17 as testing IO interface

        EA = 1;   // Enable global interrupts

        TouchKeyInit();     //Important Step 1: Initialization functions of scanning the touch key
        InitialLcd();       //Initialize the displayed section

        while(1)
        {
                WDTCON |= 0x10; //Clear watchdog

//Important Step  2: Scan one round of touch key mark, whether to call TouchKeyScan() subject to the flag position
                if(SOCAPI_TouchKeyStatus & 0X80)
                {
                        //Important Step 3: Clear the //flag position, external clear is required.
                        SOCAPI_TouchKeyStatus &= 0X7F;
                        exKeyValue = TouchKeyScan();          //Important Step  4: Analyze key data and return the results

                        //// If there is any key, update and display the cache data
                        {
                        UpdateLcdBufFunc();  //Update and display the data

                        ////If there is no display, directly operate IO interface and view the results with the oscilloscope
                        TESTIO=~TESTIO;
                        }

//Important Step 4: After bSensorCycleDone flag position is raised, the internal detection keys will be stopped, and time slice will be set aside to display the data
                        {
                                DisplayData();    //After scanning the touch keys, start to display immediately
                                OpenPwm();        //Enable PWM used for display
                        }

                }

        }
}
```

Display.c

```
void  DisplayData(void)
{
    ComAllClose; SegAllClose;

    if(isLcdComflag == 0)       //Display COM0 data
    {
            SetSegData(gIsLcdDataBuf[0]);
            seg8 = gIsLcdDataBuf[4] & 0x01;
```

```
            seg9 = gIsLcdDataBuf[4] & 0x02; COM0 = 0;

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

            isLcdComflag = 2;
    }
    else if(isLcdComflag ==2) //Display COM2 data
    {
            SetSegData(gIsLcdDataBuf[2]);
            seg8 = gIsLcdDataBuf[6] & 0x01;
            seg9 = gIsLcdDataBuf[6] & 0x02;
            COM2 = 0;
            isLcdComflag = 3;
    }
    else if(isLcdComflag ==3) //Display COM3 data
    {
            SetSegData(gIsLcdDataBuf[3]);
            seg8 = gIsLcdDataBuf[7] & 0x01;
            seg9 = gIsLcdDataBuf[7] & 0x02;
            COM3 = 0;
            isLcdComflag = 4;
    }
    else
    {



            if(isLcdComflag == 4)       //COM display is completed, start to scan the key
            {

                    ClosePwm();       //Disable PWM used for display.
                    isLcdComflag = 0;

                    //Important Step 5: After all displays are completed, it is required to recall TouchKeyRestart();
                    start to scan the key, otherwise, //it will not scan the key, at the same time, some IO interfaces
                    hared with TK need to be closed to maintain the consistency of key detection.
                    ComAllClose;
                    SegAllClose;
                    TouchKeyRestart();
            }

        }
}
```

Items that do not share with TouchKey and LED

Main.c

```
void main()
{
    unsigned char result =0;
    SegOutState;       // Initialize IO interface, displayed SEG, COM pins
    ComOutState; ComAllClose; SegAllClose;
```

```
        TestIOPortOut;    //P17 as testing IO interface

        EA = 1;  // Enable global interrupts

        TouchKeyInit();   //Important Step 1: Initialization functions of scanning the touch key
        InitialLcd();     // Initialize the displayed section

        while(1)
        {
                WDTCON |= 0x10;          //Clear watchdog if(TimerFlag_1ms==1)
                {
                        TimerFlag_1ms=0;
                        if(SOCAPI_TouchKeyStatus&0x80)          //Important Step 2: Scan one round of touch key
        mark, whether to call
                                                                TouchKeyScan() depends on if the flag position is
        raised
                        {
                                SOCAPI_TouchKeyStatus &=0x7f;  //Important Step 3: Clear the flag position, external
   clear is required.
                                exKeyValueFlag = TouchKeyScan();
                                ChangeTouchKeyvalue();
                                UpdateLcdBufFunc();  // Update and display the data
                                TouchKeyRestart();          //Start new round of switching TimerFlag_1ms=0;

                        }
                        BuzzerWork();
                        //********************Buzzer Drive Functions**********************
                        if(++Timercount>=10)
                        {
                                Timercount = 0;
                                DataUpdateCount++;
                        }
                        UpdateDisplay(); //********************Processing Display
Contents**********************
                }
            }
      }

      void  DisplayData(void)
      {
          ComAllClose;
          if(isLcdComflag == 0)       //Display COM0 data
          {
                  LedSetSegData(LcdDisplayBuf[gIsLedDataBuf[0]]); COM0 = 0;
                  isLcdComflag = 1;
          }
          else if(isLcdComflag ==1) //Display COM1 data
          {
                  LedSetSegData(LcdDisplayBuf[gIsLedDataBuf[5]]); COM1 = 0;
                  isLcdComflag = 2;
          }
          else if(isLcdComflag ==2) //Display COM2 data
          {
                  LedSetSegData(LcdDisplayBuf[gIsLedDataBuf[1]]); COM2 = 0;
                  isLcdComflag = 3;
          }
          else if(isLcdComflag ==3) //Display COM3 data
          {
                  LedSetSegData(gIsLedDataBuf[3]); COM3 = 0;
```

```
            isLcdComflag = 4;
    }
    else if(isLcdComflag ==4) //Display COM4 data
    {
            LedSetSegData(gIsLedDataBuf[4]); COM4 = 0;
            isLcdComflag = 5;
    }
    else if(isLcdComflag ==5) //Display COM5 data
    {
            LedSetSegData(LcdDisplayBuf[gIsLedDataBuf[2]]); COM5 = 0;
            isLcdComflag = 6;
    }
    else if(isLcdComflag ==6)
    {
            isLcdComflag = 0; ComAllClose;
    }
}
```

5. Version Change History

Version	Change History	Date
V1.0	Document formatting	Dec. 2022

Statement

Shenzhen SinOne Microelectronics Co., Ltd. (hereinafter referred to as SinOne) reserves the right to change, correct, enhance, modify and improve SinOne products, documents or services at any time without prior notice. SinOne believes that the information provided is both accurate and reliable. The information in this document becomes available since November 2021. In the actual production design, please refer to the latest data manual of each product and other relevant materials.