

1 总体描述

本文档是赛元 SC95F 系列 Touchkey MCU 低功耗触控库的应用指南。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：

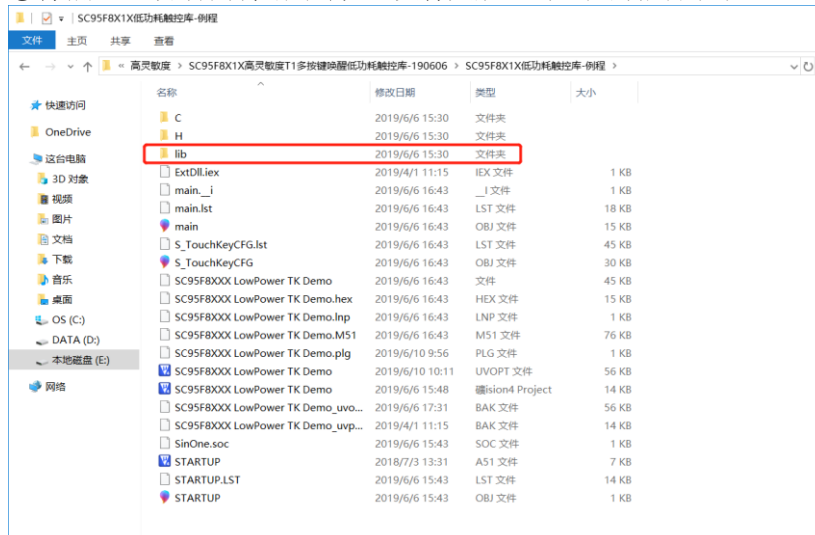
- 1、高灵敏度模式可适应隔空按键触控、接近感应等对灵敏度要求较高的触控应用
- 2、高可靠模式具有很强的抗干扰能力。
- 3、最多可实现 31 路触控按键及衍生功能
- 4、高灵活度开发软件库支持，低开发难度
- 5、自动化调试软件支持，智能化开发
- 6、部分型号可以在 MCU STOP 模式下进入低功耗模式工作，单个触控按键唤醒时芯片整体功耗可低至 8uA

低功耗触控库是基于通用的触控库增加了低功耗的实现方法，按键的调试过程和通用库的方法一致，具体参照《赛元 SC95F 系列 TouchKey MCU 应用指南》，在函数调用接口上，**低功耗触控库较通用库增加了 7 个函数接口和低功耗下外部中断唤醒处理接口**，调用过程和通用库稍有差异。。同时，针对 95F8X1X 提供了 STOP LIB 库文件，包含进入 STOP 模式和退出 STOP 模式两个标准库函数，以供调用。其他系列无须调用 STOP LIB 库。

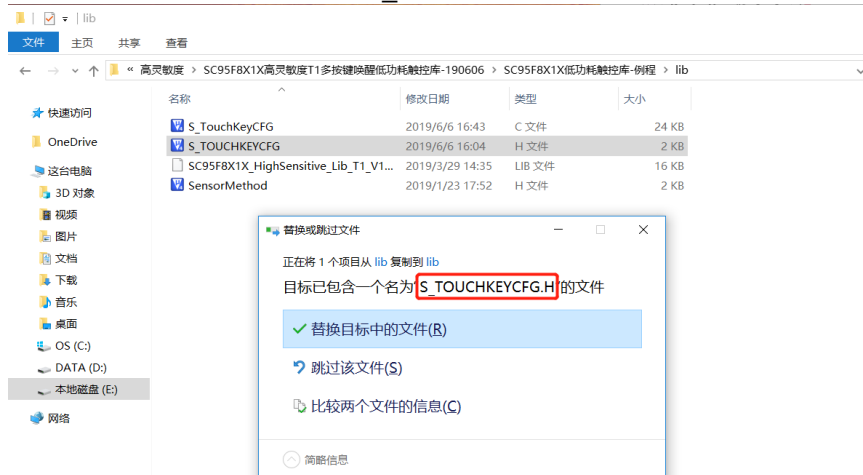
2 低功耗触控库调用方法

2.1 在用户工程中加入低功耗触控库的方法

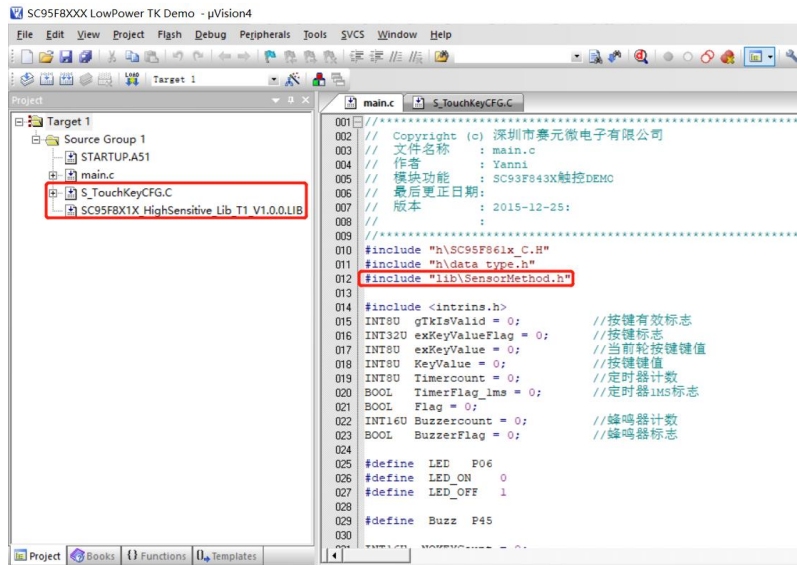
①将相应型号的低功耗触控库 lib 文件夹放置到工程的根目录下



②将通过触控调试软件生成的“S_TOUCHKEYCFG.H”替换到 lib 文件夹中



③将“S_TouchKeyCFG.C”和“SC95F8XXX_HighSensitive_Lib_Tn_Vn.n.n.LIB”添加到用户工程中，引用头文件 `#include "lib\SensorMethod.h"`

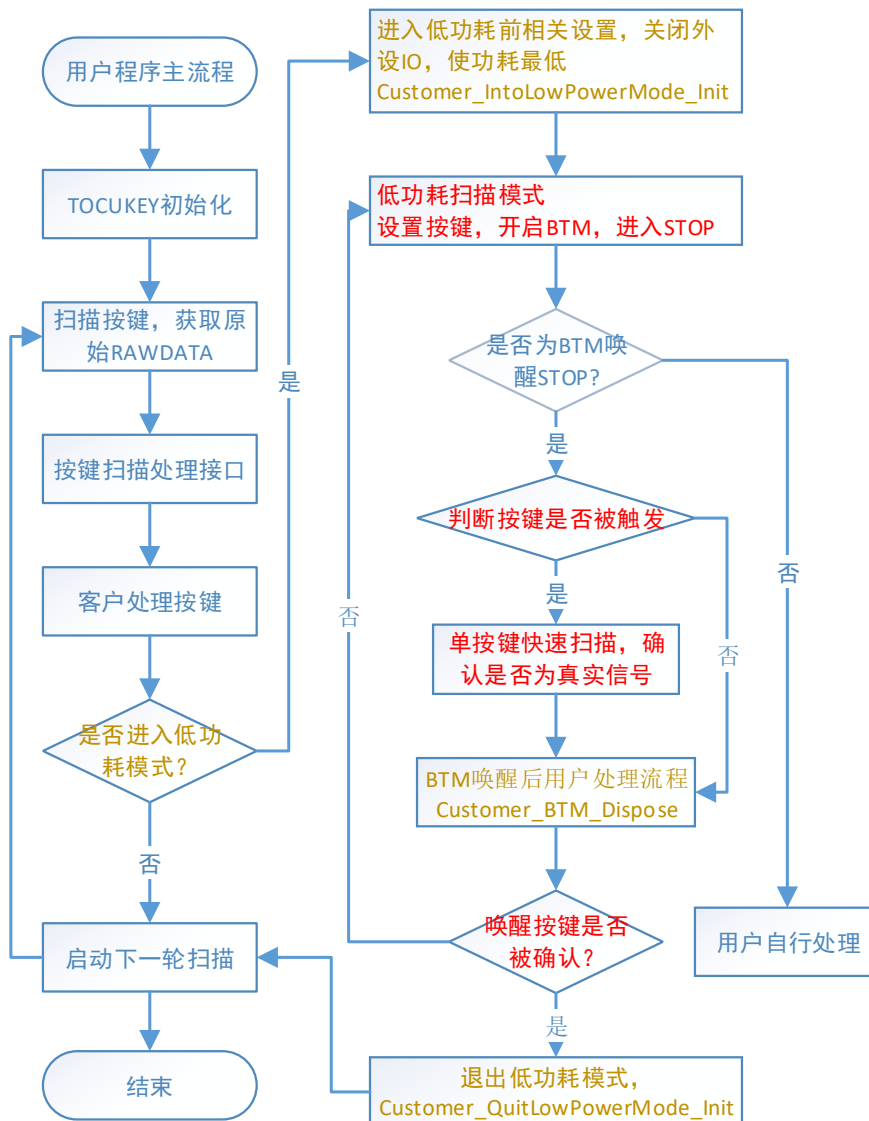


2.2 用户程序调用过程

①函数接口说明：

接口名	接口定义	调用说明
GetLowPowerScanFlag	获取低功耗模式标志：0 为正常运行模式，1 为低功耗模式	在主循环内区分两种模式，执行不同的程序分支
TouchKey_IntoLowPowerMode	进入低功耗模式	调用该函数进入低功耗模式。 用户根据实际应用决定调用 ，例如，一段时间未有按键按下进入低功耗，调用该函数。但需注意， 进入低功耗要在无按键的情况下进入（即返回的键值为0）
Customer_IntoLowPowerMode_Init	用户进入低功耗模式前的初始化	该函数用户需要自行添加在主程序中，并对其进行实现，无需调用。如果程序中有使用中断资源，也请在此函数中关闭对应中断使能，以防应用程序逻辑上不严谨导致功能异常。
LowPower_Touchkey_Scan	低功耗 TK 扫描处理	在低功耗模式下调用
TouchKey_QuitLowPowerMode	退出低功耗模式	调用该函数系统进入正常模式。 当TK按键被唤醒，自动被调用，退出低功耗。 用户也可以根据外部信号退出低功耗
Customer_QuitLowPowerMode_Init	用户退出低功耗模式前的设置	该函数用户需要自行添加在主程序中，并对其进行实现，无需调用 同时也需在此函数中使能之前所关闭的对应中断资源。
Customer_BTM_Dispose	用户 BTM 函数处理	该函数用户需要自行添加在主程序中，并对其进行实现，无需调用。由于TK低功耗的实现占用了 BTM 资源，这里开放了用户接口，用户在此函数内添加 BTM 唤醒的执行内容，例如读取外部的 IO 状态

②程序流程图如下：



③在 main 文件中加入下面三个函数，并根据实际应用加入所需的实现代码

```
/*
*****
*函数名称: void Customer_IntoLowPowerMode_Init(void)
*函数功能: 用户进入低功耗模式前的初始化
*入口参数: void
*出口参数: void
*备注说明: 在进入低功耗前, 用户需要关闭外围耗电的电路, 使电流最低
*          该函数已在S_TouchKeyCFG.C文件TouchKey_IntoLowPowerMode()函数中调用, 用户无需调用
*****
void Customer_IntoLowPowerMode_Init(void)
{
    /*进入低功耗前设置*/

    //该函数已在TouchKey_IntoLowPowerMode()里调用

    //用户无需调用该函数!!!

    //用户需要自己编写该函数实体!!!

    //关闭耗电的外设, 保持最低功耗 如果程序中有使用中断资源, 也请及时关闭对应中断使能, 以防应用程序逻辑上不严谨导致功能异常。
    LED = LED_OFF;
}

/*
*****
*函数名称: void Customer_QuitLowPowerMode_Init(void)
*函数功能: 用户退出低功耗模式后的初始化
*入口参数: void
*出口参数: void
*备注说明: 在退出低功耗前, 用户进行的必要操作
*          该函数已在S_TouchKeyCFG.c文件TouchKey_QuitLowPowerMode()函数中调用, 用户无需调用
*****
void Customer_QuitLowPowerMode_Init(void)
{
    /*退出低功耗前设置*/

    //该函数已在低功耗下满足TK唤醒退出低功耗时调用, 即在TouchKey_QuitLowPowerMode()中调用

    //用户无需调用该函数!!!

    //用户需要自己编写该函数实体!!!

    //恢复必要的外设 同时也需使能之前所关闭的对应中断。
}

/*
*****
*函数名称: void Customer_BTMDispose(void)
*函数功能: 用户BTM唤醒后自己的处理函数
*入口参数: void
*出口参数: void
*备注说明: 低功耗实现利用了BTM资源, 用户需要在BTM中实现的内容在此函数实现
*          该函数已在S_TouchKeyCFG.c文件LowPower_Touchkey_Scan()函数中调用
*****
void Customer_BTMDispose(void)
{
    //该函数已在低功耗模式扫描函数中调用

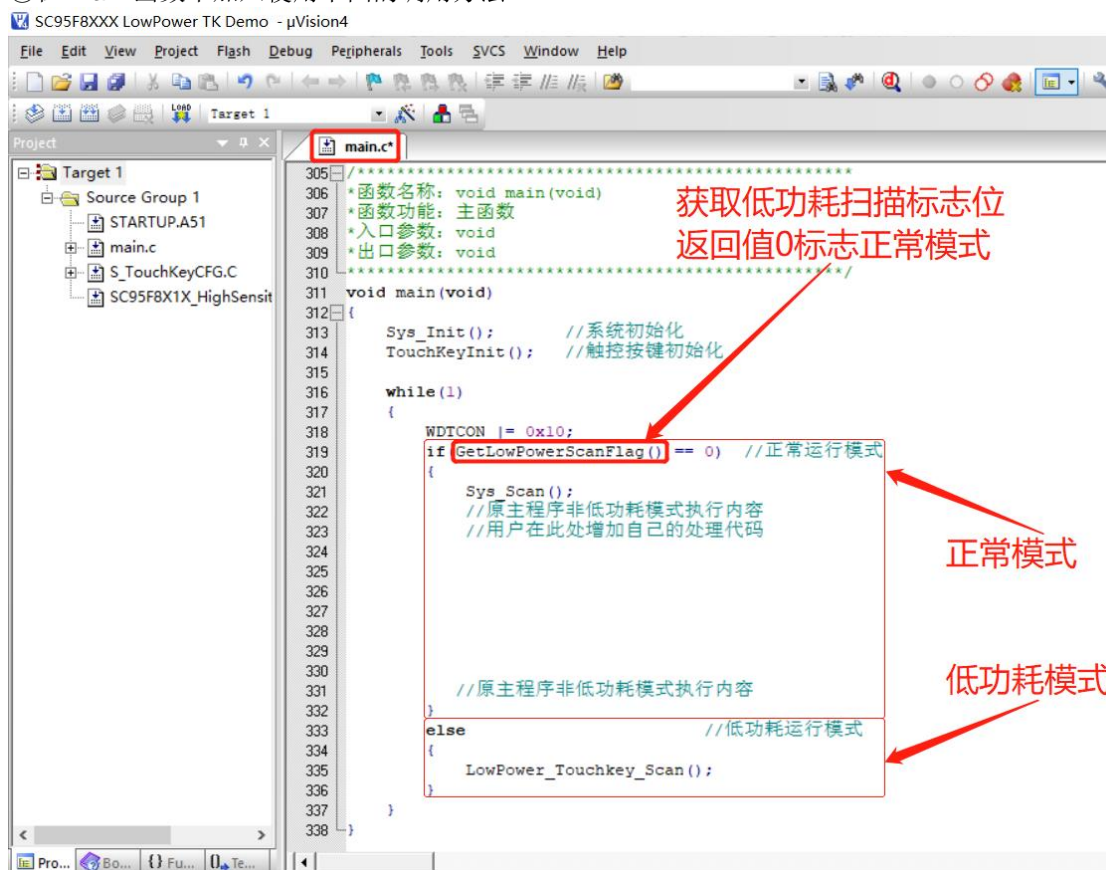
    //此函数为BTM定时唤醒后, 用户的处理函数

    //低功耗模式下, 用户需在BTM中实现的功能可在此函数实现

    //比如查询某个IO, 电平发生变化等条件成立, (TK唤醒除外)需退出低功耗模式可在此函数中调用TouchKey_QuitLowPowerMode()

    /*
    if(TEST_IO == 0)
    {
        //do something
        TouchKey_QuitLowPowerMode();
    }
    */
}
```

④在 main 函数中加入使用下图的调用方法



⑤用户根据实际应用需求在正常运行模式下添加是否进入低功耗的判断，需注意的是，进入低功耗要在无按键的情况下进入(即返回的键值为 0)，满足条件调用函数 `TouchKey_IntoLowPowerMode()`即可进入低功耗模式。



⑥低功耗模式下 BTM 中断及外部中断处理

```

-*****/
void LowPower_Touchkey_Scan(void)
{
    #ifdef TK_LowPowerMode
    SleepMode(); //进入STOP模式
    //进行按键处理
    if(BTM_WakeUpFlag)
    {
        TouchKey_LowPower_Dispose(); //检测按键

        if( SingleKeyFastScan_Flag == 1)
        {
            SingleKeyFastScan(); //若有按键信息进入单按键快速多次扫描确定按键是否真实信号。
        }

        // 用户BTM唤醒后的处理函数
        Customer_BTMDispose();

        //其他唤醒请在以下分支进行判断并调用退出低功耗函数TouchKey_QuitLowPowerMode()
        //例如: if (外部中断唤醒标志==1)
        // {
        //     TouchKey_QuitLowPowerMode();
        //     //业务层唤醒后实现逻辑
        // }
        //
        //
        //
        //
        //非BTM唤醒，用户根据需要自行增添处理程序
    }
    #endif
}
    
```

BTM唤醒后进行低功耗TK扫描

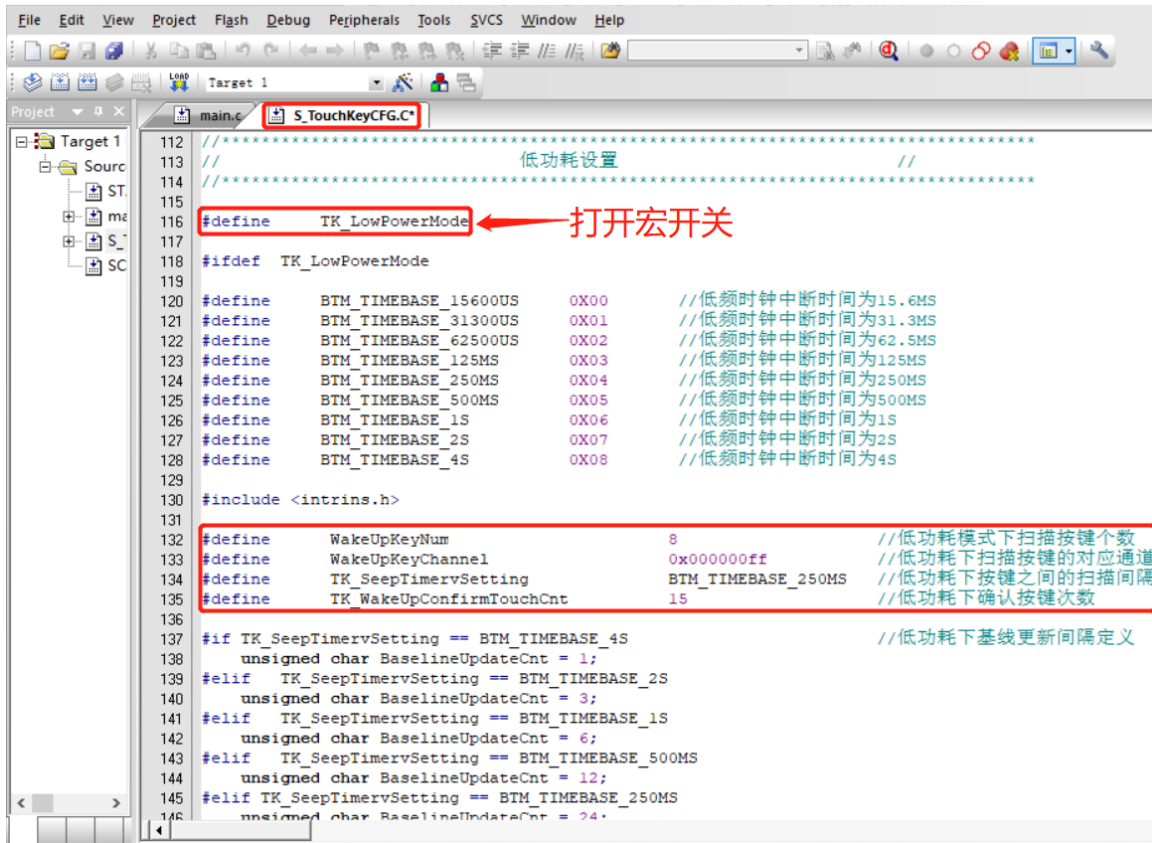
低功耗占用BTM资源，若用户需要在BTM中处理其他功能，可在此函数中添加相关处理程序

3 低功耗参数设置与功耗

3.1 低功耗参数设置

①打开 S_TouchKeyCFG.c 文件，将低功耗功能的宏开关打开（当用户无须低功耗功能可以将其注释，以达到节省代码空间），设置低功耗唤醒的个数，低功耗下要扫描的按键，以及 BTM 定时唤醒的时间。

SC95F8XXX LowPower TK Demo - µVision4



②参数说明：

参数名	参数定义	设置说明
WakeUpKeyNum	低功耗模式下扫描的按键个数	不可以超过正常扫描的按键个数，按键个数越多，功耗越大
WakeUpKeyChannel	低功耗下扫描的按键通道	只能设置正常扫描下已开启的按键，将对应的按键 BIT 位置起，置起的位数必须与设置的按键个数相同
TK_SleepTimervSetting	低功耗下按键间的扫描周期	BTM_TIMEBASE_15600US BTM_TIMEBASE_31300US BTM_TIMEBASE_62500US BTM_TIMEBASE_125MS BTM_TIMEBASE_250MS BTM_TIMEBASE_500MS BTM_TIMEBASE_1S BTM_TIMEBASE_2S BTM_TIMEBASE_4S 可设置以上的选择，设置的时间越长功耗越低，按键响应速度越慢。用户可根据实际需要进行设置。
TK_WakeUpConfirmTouchCnt	低功耗下按键的确认次数	可以设置 10-20

3.2 高灵敏度触控功耗

高灵敏度低功耗采用一次 BTM 唤醒扫描所有唤醒 TK 按键的方法实现低功耗模式 TK 按键扫描，设置的 BTM 定时时间越短，唤醒按键个数越多，工作电流越大。下表为灵敏度触控低功耗下不同唤醒按键数及不同 BTM 定时时间与工作电流的对应关系（@ $V_{DD}=3.3V$ & $F_{SYS}=32MHz$ ），用户根据实际应用需要自行设置低功耗参数。

唤醒按键个数 BTM 时间 (ms)	低功耗电流(uA)											
	1	2	3	4	5	6	7	8	9	10	11	12
500	4.9	5.1	5.3	5.7	6.1	6.2	6.5	6.8	7.4	7.7	7.9	8.3
250	8.8	10.0	11.3	12.1	13.5	14.9	16.0	16.6	17.8	18.8	20.1	20.9

*由于芯片个体差异，测试数据会有 $\pm 2\mu A$ 左右的误差

建议用户设置 BTM 定时时间为 250ms。

3.3 高可靠触控功耗

高可靠低功耗采用一次 BTM 唤醒扫描一个唤醒 TK 按键的方法实现低功耗模式 TK 按键扫描，设置的 BTM 定时时间越短，工作电流越大；相同 BTM 定时时间，唤醒按键个数越多，按键响应速度越慢。下表为高可靠触控低功耗不同 BTM 定时时间与工作电流的对应关系（@ $V_{DD}=3.3V$ ）。

BTM 时间(ms)	250	125	62.5	31.25	15.625
低功耗电流(uA)	8.9	10.9	13.8	20.9	33.7

①WakeUpKeyNum=1，建议 BTM 设置 250ms 或 125ms；

②WakeUpKeyNum=2，建议 BTM 设置 125ms；

③WakeUpKeyNum=3,4,5，建议 BTM 设置 62.5ms；

④WakeUpKeyNum=6,7,8,9,10,11,12，建议 BTM 设置 31.25ms；用户根据实际应用自行设置低功耗参数。

4 注意事项

1. 赛元低功耗库支持需大于 12 个按键唤醒的低功耗应用。
2. 赛元低功耗库在低功耗模式下仅支持单键。
3. Customer_IntoLowPowerMode_Init():用户进入低功耗模式前的初始化
Customer_QuitLowPowerMode_Init():用户退出低功耗模式前的设置
涉及到该两个函数接口说明，请详细查看 2.2 章节提及的调用说明。
4. 进入 STOP 模式的注意事项：
 - a. CMOD 口初始化设置强推挽输出模式，输出 0
 - b. 关闭模拟外设和 WDT
 - c. 悬空以及未封装出的 IO 设置为强推挽输出模式
 - d. 根据实际应用电路设置 IO 口电平
 - e. 建议电路匹配 CMOD 电容为 102

5 更改记录

版本	记录	日期
V1.3	更新 CMOD 初始化 IO 状态及匹配电容值	2023 年 4 月
V1.2	更新进出 stop 初始化函数函数内程序注意事项	2023 年 2 月
V1.1	更新高灵敏低功耗数据	2022 年 7 月
V1.0	增加了 STOP LIB 库文件，提供进入和退出 STOP 模式的函数	2020 年 4 月
V0.2	修改了功耗部分描述	2020 年 3 月
V0.1	初版	2019 年 6 月