32 位 RSIC 架构的微控制器

SDK 关于 TK 说明

V1.1

修订历史记录

变更类型：A - 增加　M - 修订　D - 删除

变更版本号	日期	变更类型	修改人	审核	摘要

版权声明

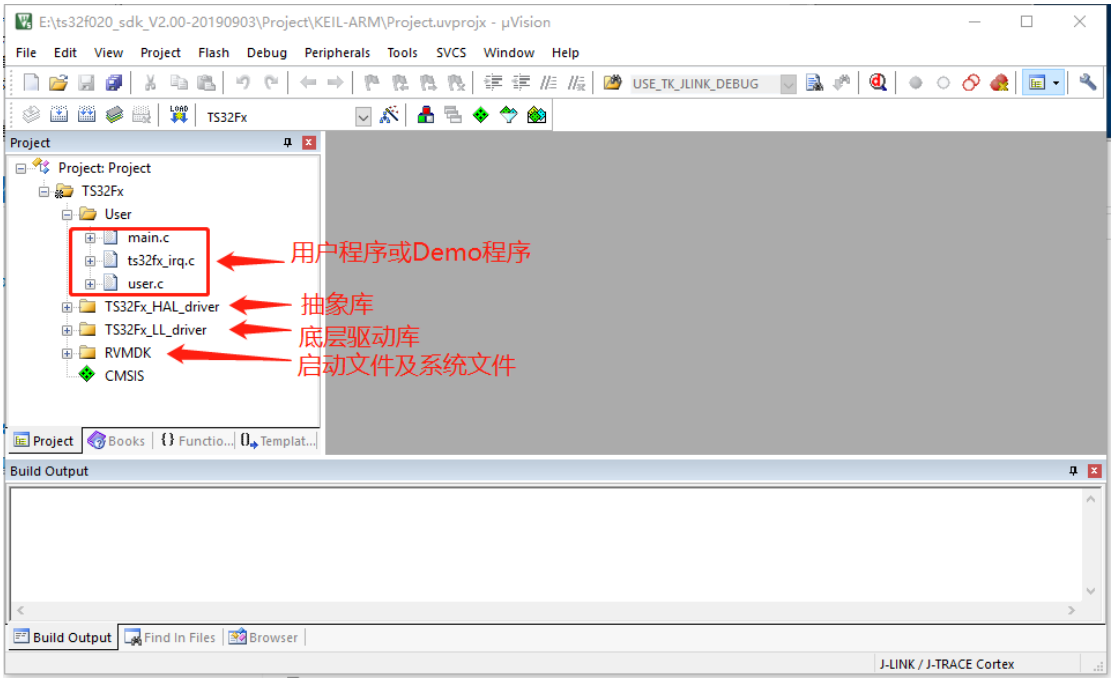
目录

1. SDK 文档结构及项目结构介绍

下面为 SDK 的文档结构:

Examples (例程文件夹): 将芯片各个模块制作的 demo 放置于此文件夹中, 方便用户查看。
Hal (抽象层文件夹): 目前只把 TK 的接口抽象后放置于此文件夹里面。
Libraries (库文件夹): 里面包含芯片的启动文件, 及各个模块的底层库等重要文件。
Project (项目工程文件夹): 包含项目启动文件及烧录生成的加密文件等。
User (用户文件夹): 放置用户自定义的文件或者可以将 Demo 例程文件拷贝至该文件夹使用。

双击打开工程, 工程路径为……\Project\KEIL-ARM\Project.uvprojx,打开的工程如下:



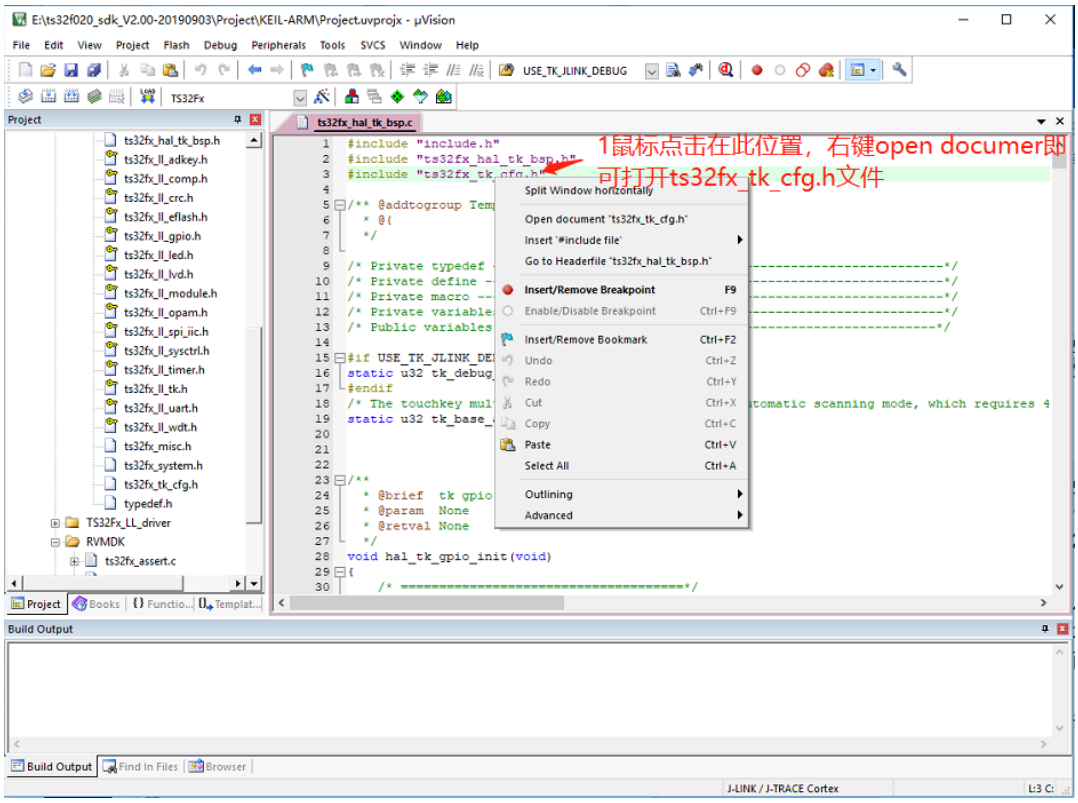
2. SDK 中 TK 的 Demo 的分析

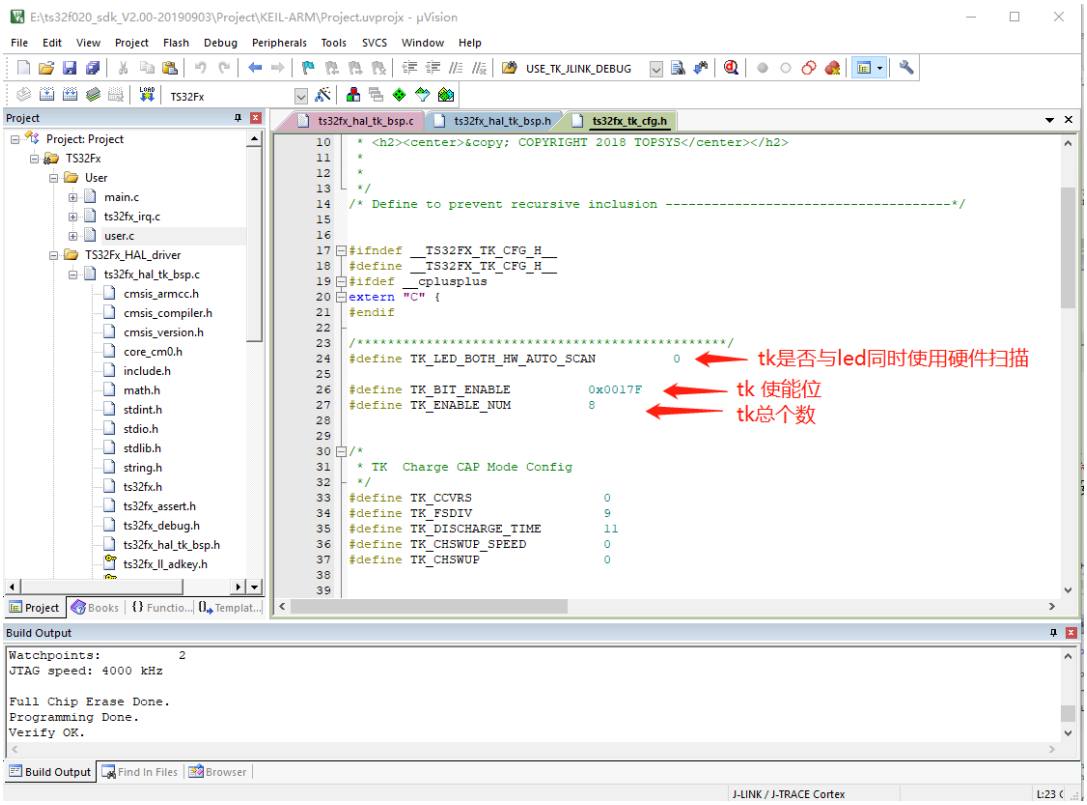
2.1. 只有 TK 为硬件自动扫描的情况下：

客户可以参考 Demo 在该路径下……\Examples\TK\TK_ONLY，将此目录下的所有文件复制拷贝到 User 目录（默认的 SDK 已经完成拷贝），之后打开工程……\Project\KEIL_ARM\Project.uvprojx。

下面介绍该 demo 的主要参数及调用方法

a）先看下使能的 TK 的按键及个数和是否开启 led 与 tk 的硬件扫描宏定义，通过查看 ts32fx_tk_cfg.h (在项目的 TS32Fx_HAL_driver 下有 ts32fx_hal_tk_bsp.c 双击打开后，查看代码中的#include “ts32fx_tk_cfg.h”的位置，并鼠标点击此位置右击再点击 open documer “ts32fx_tk_cfg.h”文件即可打开该文件)即可查看到使能的 tk 的位为：0x0017f，个数为 8，根据使能位可以知道使能 tk0，tk1，tk2，tk3，tk4，tk5，tk6，tk8，共 8 个 tk，且 TK_LED_BOTH_HW_AUTO_SCAN 宏定义为 0，表示只使用 tk 进行硬件扫描。如下图：



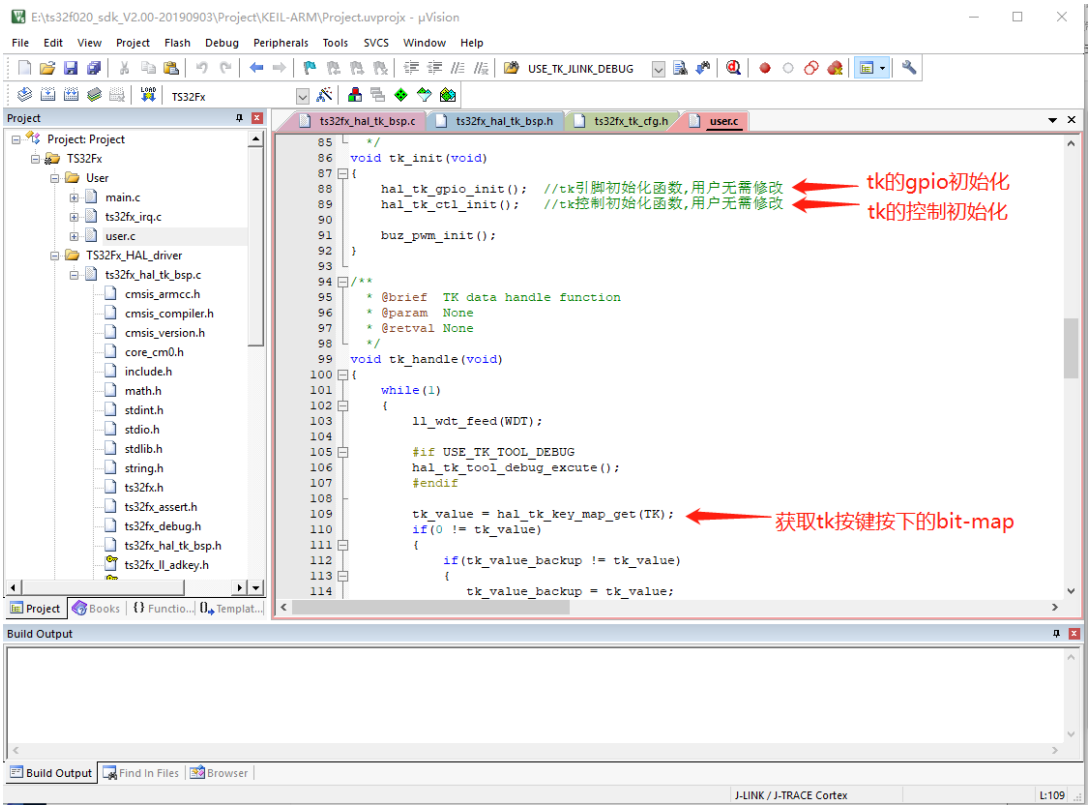


b）demo 中是如何使用 tk 的：

1、初始化 TK 的 IO: hal_tk_gpio_init()　（该函数为抽象层自带）

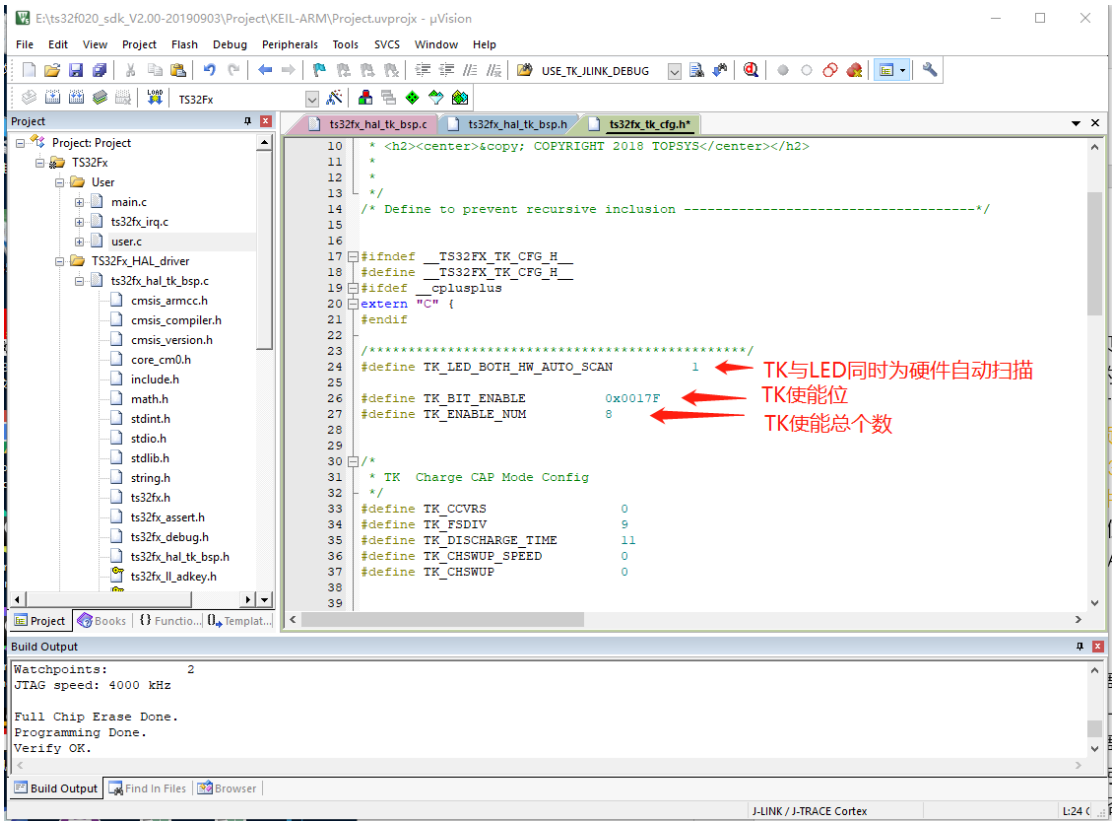
2、初始化 TK 的控制: hal_tk_ctl_init()　（该函数为抽象层自带）

3、在需要获取按键的地方调用 hal_tk_key_map_get(TK)　获取 key 被按下的 bit-map（该函数为抽象层自带）。如下图：



2.2. TK 与 LED 同时为硬件自动扫描的情况下：

客户可以参考 Demo 在该路径下……\Examples\TK\ TK&LED_HW_AUTO_SCAN，将此目录下的所有文件复制拷贝到 User 目录，之后打开工程……\Project\KEIL_ARM\Project.uvprojx。
下面介绍该 demo 的主要参数及调用方法
a）先看下使能的 TK 的按键及个数和是否开启 led 与 tk 的硬件扫描宏定义，通过查看 ts32fx_tk_cfg.h (在项目的 TS32Fx_HAL_driver 下有 ts32fx_hal_tk_bsp.c 双击打开后，查看代码中的#include “ts32fx_tk_cfg.h”的位置，并鼠标点击此位置右击再点击 open documer “ts32fx_tk_cfg.h”文件即可打开该文件)即可查看到使能的 tk 的位为：0x0017f，个数为 8，根据使能位可以知道使能 tk0，tk1，tk2，tk3，tk4，tk5，tk6，tk8，共 8 个 tk，需要将 TK_LED_BOTH_HW_AUTO_SCAN 宏定义置为 1，因为这样表示只使用 tk 与 led 都为硬件自动扫描。如下图：



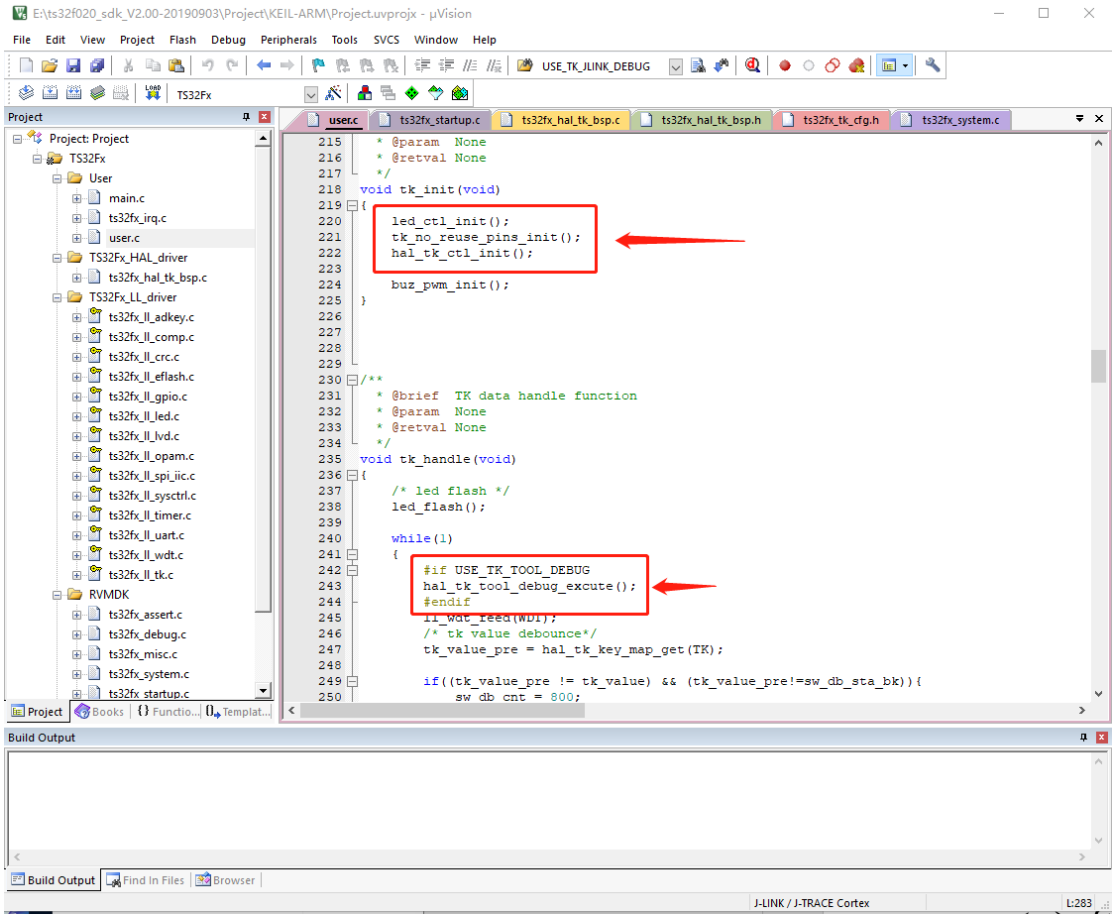
b）demo 中是如何初始化 tk 及 led 的，及使用 tk:

1、初始化 led:　led_ctl_init()　（该函数需要用户根据自己定义修改）

2、初始化没有被 led 复用的 tk 引脚，将这些引脚配置为输出低: tk_no_reuse_pins_init()　（该函数需要用户根据自己定义修改）

3、初始化 tk 控制:　hal_tk_ctl_init()　（该函数为抽象层自带）

4、在需要获取按键的地方调用 hal_tk_key_map_get(TK)　获取 key 被按下的 bit-map（该函数为抽象层自带），如下图:

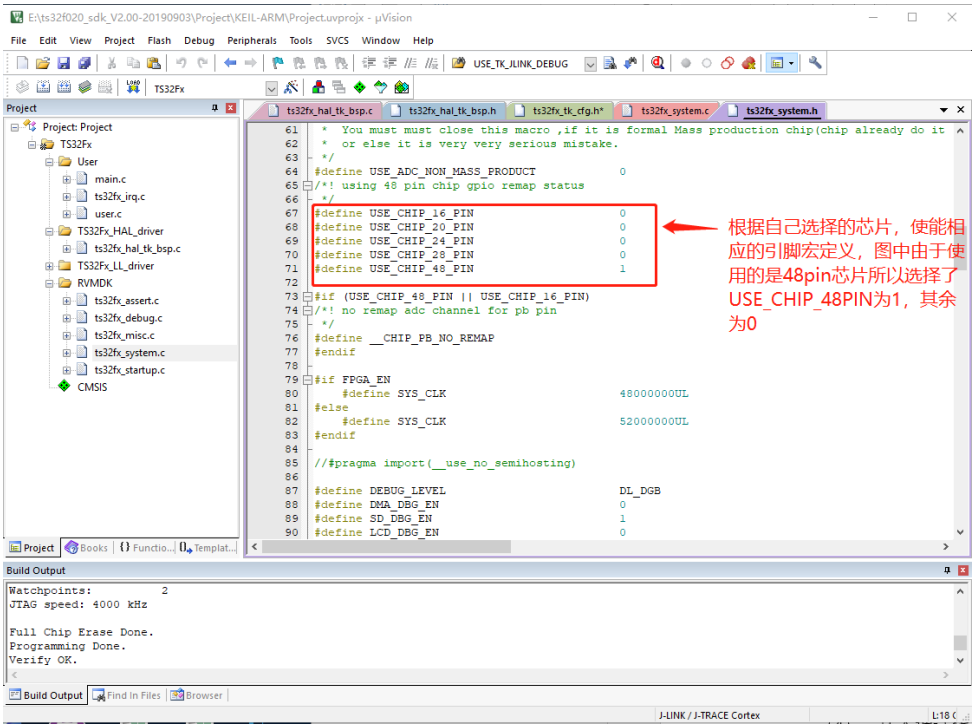


3. 在客户的方案板中调试 tk 按键方式：

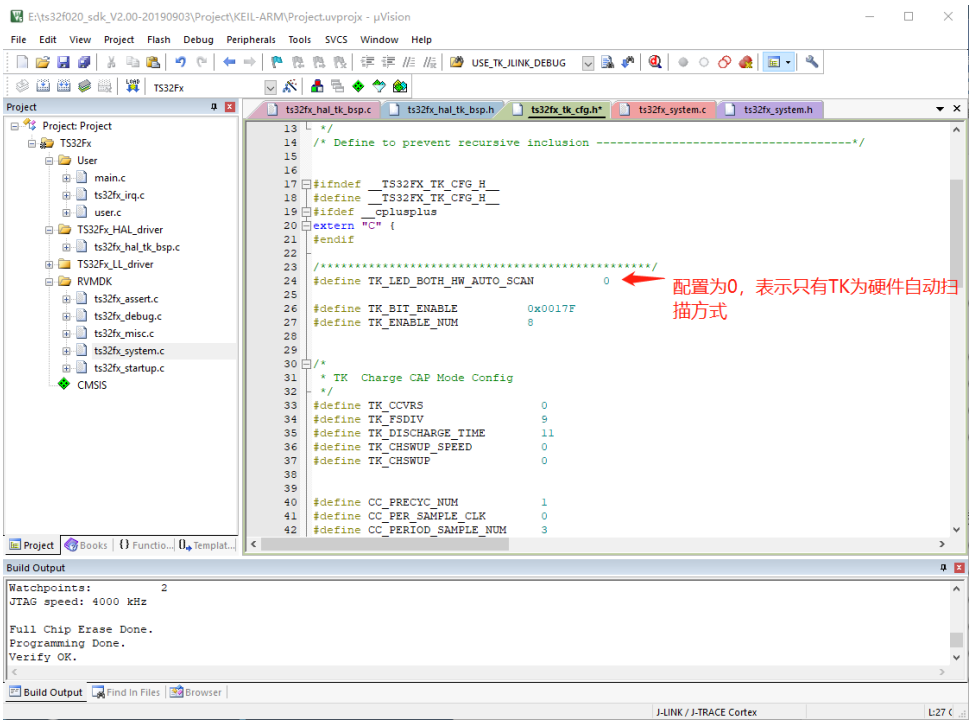
调试过程分两种，一种为使用调试工具上位机进行调试（推荐，但需要占用芯片一路 uart 资源），另一种为使用 jlink 配合 keil 进行在线调试。

3.1. 使用调试工具进行调试：

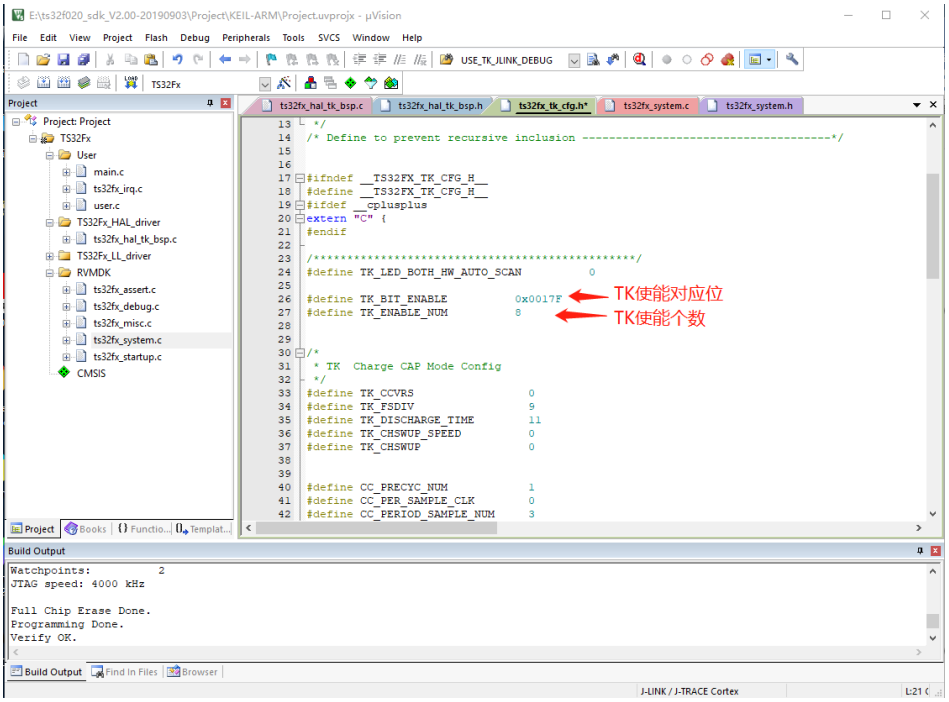
（1） 确认芯片的引脚数，并在 sdk 中设置相应的宏定义；
打开 ts32fx_system.h 文件将相应的引脚宏定义置 1



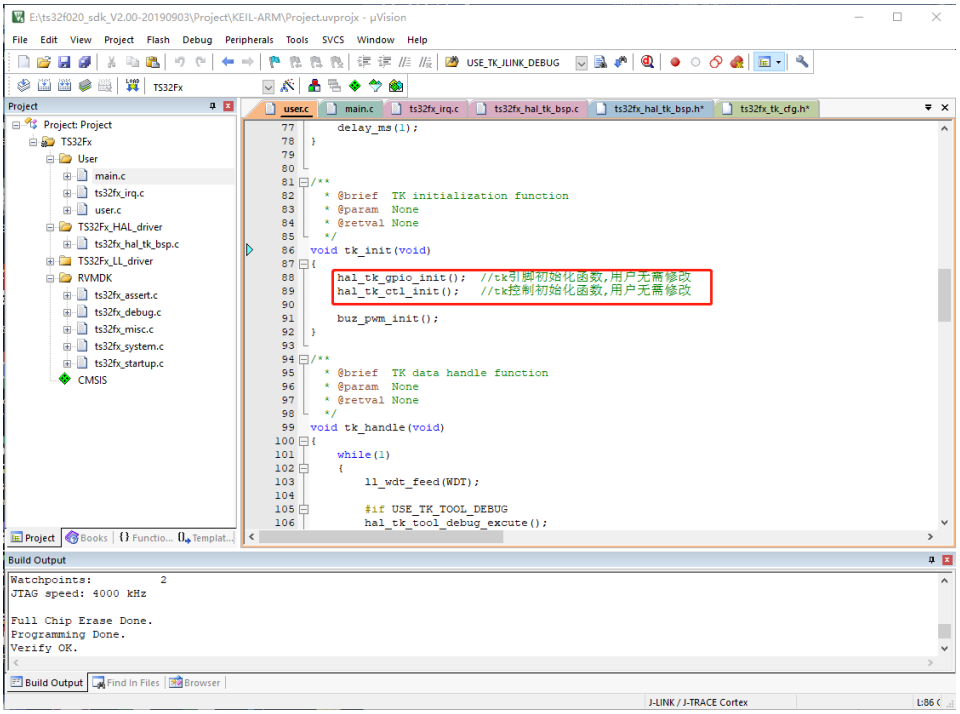
(2) 确认芯片是否同时使用 tk 与 led 内部硬件扫描，并设置相应的宏定义；
打开 ts32fx_tk_cfg.h



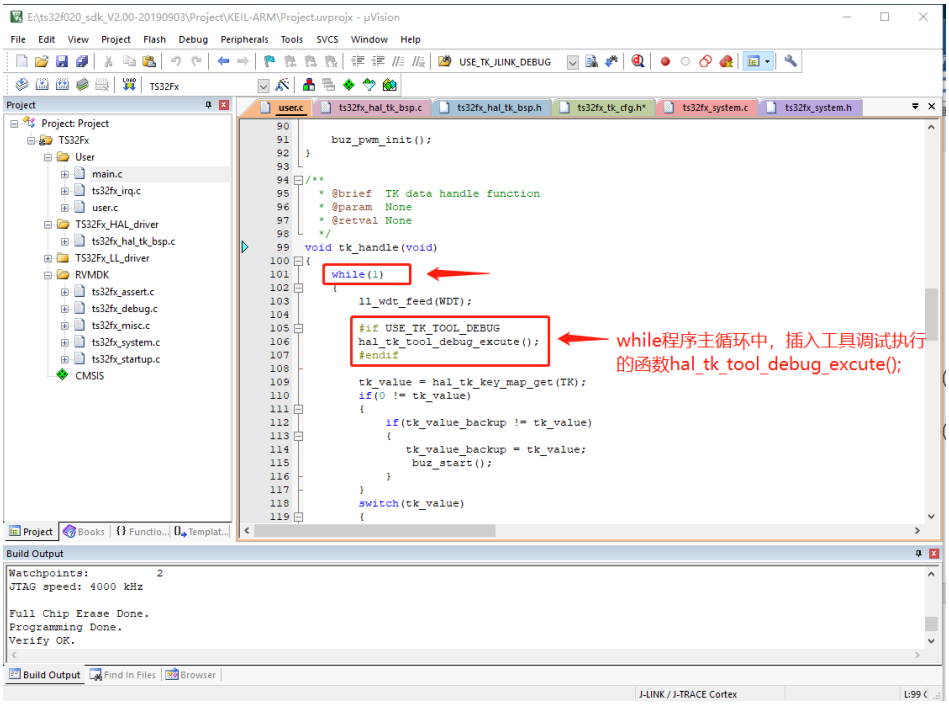
(3) 确认硬件使用 tk 的索引，配置使能的 tk map 位及数量；
打开 ts32fx_tk_cfg.h



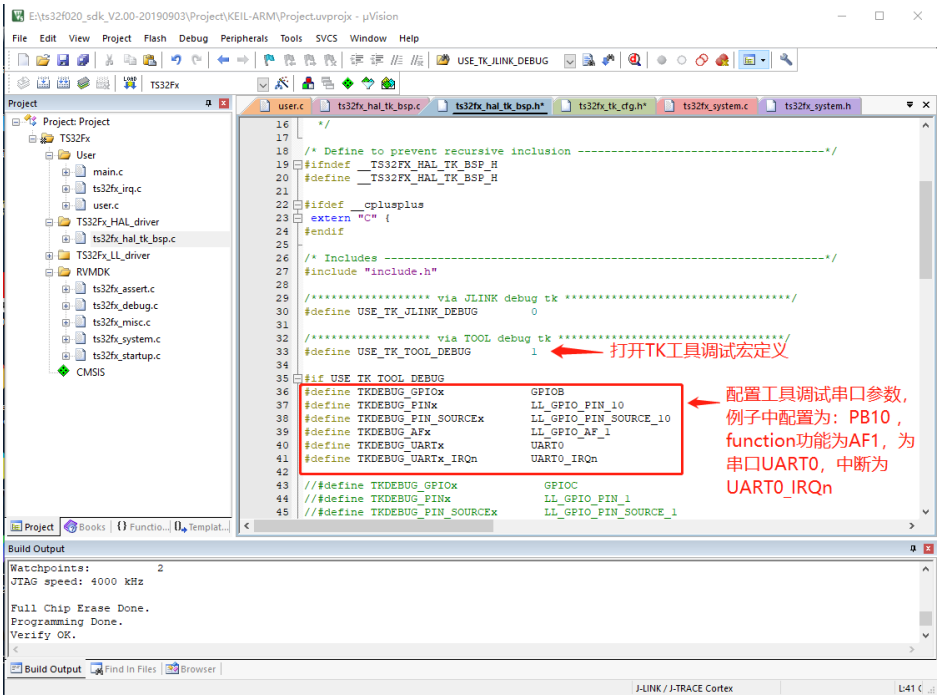
(4) 调用 tk 抽象层的初始化函数



(5) 在程序循环主体中，插入 tk 工具调试的接口

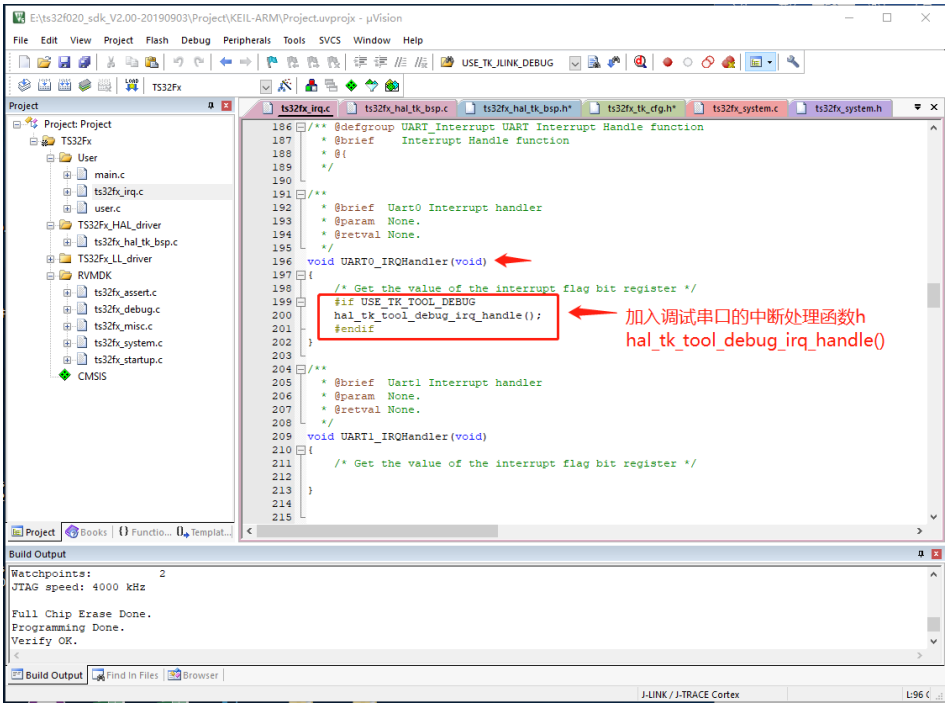


（6） 开启 tk 工具调试的宏定义，并配置相应的单 pin 调试串口参数；



（7） 插入串口中断调试函数

如上图使用的串口调试为 UART0，即在 ts32fx_irq.c 中的 UART0 中断中加入函数



(8) 使用串口单 pin 模块连接方案板及上位机，根据上位机提示进行调试，生成“ts32fx_tk_cfg.h”替换工程下的同名文件即可。

3.2. 使用 jlink 配合 keil 进行调试

(1) 确认芯片的引脚数，并在 sdk 中设置相应的宏定义，同上面方法 1；

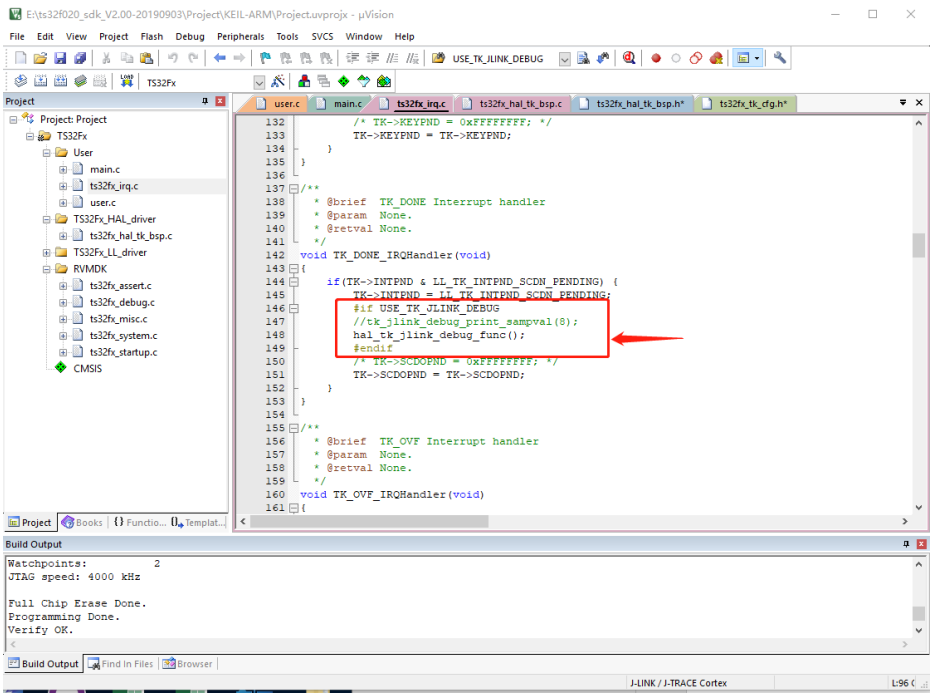
(2) 确认芯片是否同时使用 tk 与 led 内部硬件扫描，并设置相应的宏定义，同上面方法 1；

(3) 确认硬件使用 tk 的索引，配置使能的 tk map 位及数量，同上面方法 1；

(4) 调用 tk 抽象层的初始化函数，同上面方法 1；

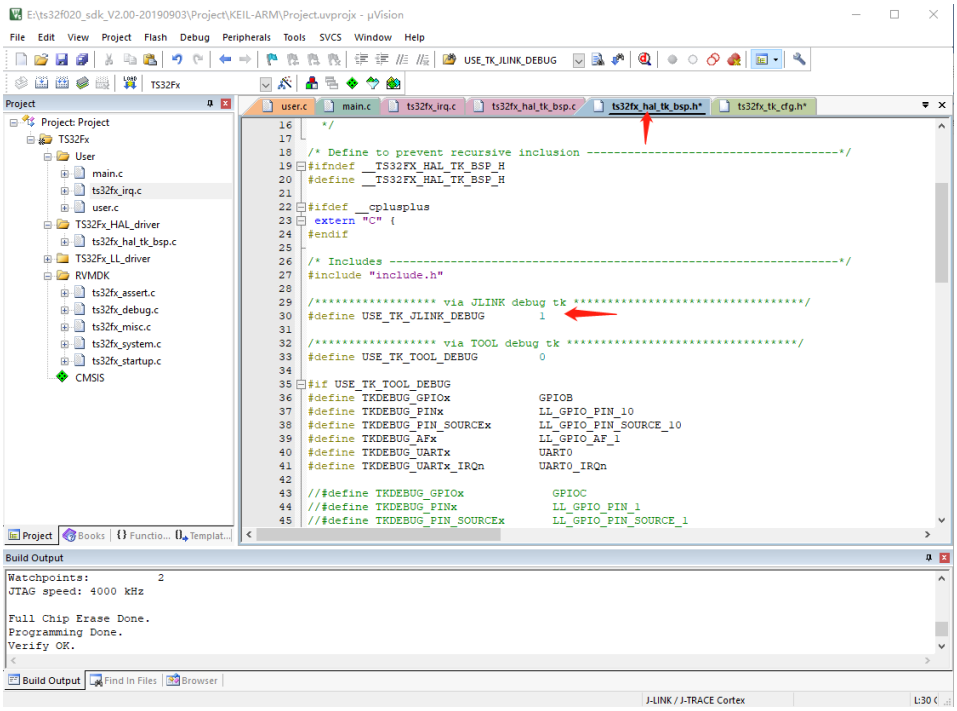
(5) 插入调试代码；

打开 ts32fx_irq.c 文件在 TK_DONE_IRQHandler 中断函数中插入调试代码

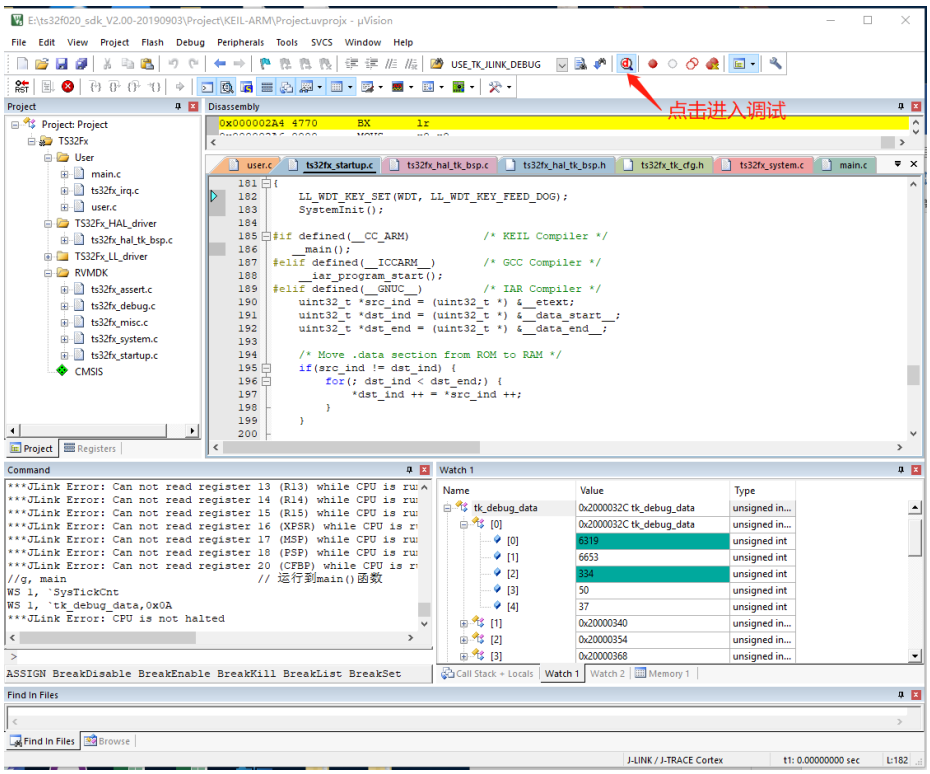


(6) 开启 tk 调试的宏定义；

打开文件 ts32fx_hal_tk_bsp.h 中 USE_TK_JLINK_DEBUG 置为 1



(7) 点击编译后再点击 keil 中的在线调试，添加 tk_debug_data 到调试窗口，按下按键进行 tk 调试。



如下图为该数组的含义，可根据这些参数设置相应的软件阈值及硬件阈值