

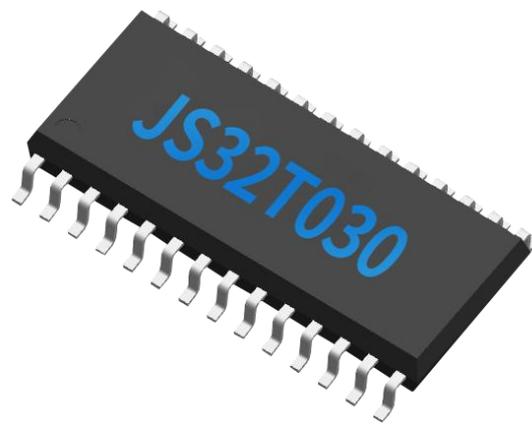


支持 LED/LCD 驱动+触控的 32 位微控制器

JS32T030 触控系列

软件设计注意事项

V1.0



珠海巨晟科技股份有限公司

地 址: 广东省珠海市高新区金唐路 1 号港湾 1 号湾 8 栋 4 楼

电 话: 0756-3335384 传 真: 0756-3335384

网 站: www.honor-ic.com 邮 编: 519080



版本历史

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
V1.0	2022-6-21	M			正式版本

版权声明

1、本资料是为了让用户根据自身需求选择合适的产品而提供的参考资料，相关的知识产权属于珠海巨晟科技股份有限公司或来自第三方的合法授权；提供上述资料不构成对相关知识产权的许可或转让，未经珠海巨晟科技股份有限公司的许可，任何人不得翻印或者复制本资料的全部或部分内容。

2、在使用本资料所记载的信息并对有关产品是否适用做出最终判断前，请您务必将所有信息作为一个整体来评价。对于本资料所记载的信息使用不当而引起的任何损失，珠海巨晟科技股份有限公司概不负责。

3、本资料所记载的产品会持续更新迭代并发布，在购买本资料所记载的产品时，请预先向珠海巨晟科技股份有限公司确认最新信息，并请您通过公司网站、微信公众号等各种方式关注珠海巨晟科技股份有限公司公布的信息，相关更新恕不另行通知。

4、如果您需要进一步了解有关本资料所记载的信息或产品的详情，请与珠海巨晟科技股份有限公司的技术服务部门联系，我们会为您提供全方位的技术支持。

目录

1. 用户配置文件说明	1
1.1. 用户配置文件的配置说明	1
1.2. 用户配置文件的说明	2
2. SWD 的使用说明	2
2.1. SWD 作为通用 IO 的配置	2
2.2. SWD_CLK 引脚电平的使用说明	2
2.3. SWD 作为 LCD 功能	2
3. TK 使用说明	2
3.1. 移植说明	2
3.2. 低功耗唤醒说明	3
3.3. 调试说明	5
3.4. 其他功能说明	5

珠海巨晟科技股份有限公司

1. 用户配置文件说明

应用工程编译成功后，在 Project->KEIL-ARM->Eflash_Proj 中生成烧录文件。

同目录有以下几个文件：

app_project.bin: 应用工程最终生成的烧录文件。

makecode.ini : 用户配置文件。

makecode.exe : 用户配置文件转换工具，应用工程默认每次编译后自动调用此程序，用户注意不要随意更改该配置。

js_link_burn.exe: 用户烧录文件生成工具，应用工程默认每次编译后自动调用此程序，用户注意不要随意更改该配置。

1.1. 用户配置文件的配置说明

用户可通过修改 makecode.ini，配置芯片的相关功能，功能如下表所示

注：所有配置为 16 进制，前面不加 0x，配置内容中 '=' 前后不要加空格等额外符号。

Name	Default	Description
CODE_PROTECT_DIS	1	代码是否需要保护 0: 保护 1: 不保护 (默认)
NVR_WRITE_PER	FF	Nvr0~1 使能擦写标识，总共 2bit，分别对应 Nvr0~1，每一 bit 中 0: 禁止擦写 1: 使能擦写 (默认)
MAIN_WRITE_PER	FFFFFFF	用户程序 main 区使能擦写标识，总共 32bit，每一 bit 对应 1k(2 个 sector) 内容，32bit 分别对应 32k 内容，每一 bit 中 0: 禁止擦写 1: 使能擦写 (默认)
MAIN_CODE_CHECK_EN	0	用户 main 程序区代码校验是否使能 0: 不使能 (默认) 1: 使能；若设置为校验使能且校验不通过，用户程序不能启动
MAIN_CODE_LENGTH	8000	用户程序 main 区代码大小，代码大小的值为一个 sector(512byte)的倍数。用户 main 区程序的 4 字节校验放在最大代码长度处。
SWD_EN	0	SWD 是否使能 0: SWD 失能，需要和 SYS_CON1 的【SWD_EN】同时作用，才能关闭 SWD 接口。 1: SWD 使能
MCLR_EN	0	MCLR(PA12)复位引脚使能控制 0: MCLR 功能无效 1: MCLR 功能有效
SWD_CLK_PULLUP	1	SWD_CLK 引脚上下拉配置 0: 下拉 1: 上拉 (默认)

1.2. 用户配置文件的功能说明

1.2.1. 关于校验功能

使能用户 main 区代码校验功能，需要配置 makecode.ini 的【MAIN_CODE_CHECK_EN】和【MAIN_CODE_LENGTH】。

若设置【MAIN_CODE_CHECK_EN】=1，则开启 main 区代码校验功能，校验代码区的大小由【MAIN_CODE_LENGTH】决定，校验码存放在校验代码区的最后 4 个字节，注意要正确配置，否则校验不通过，用户程序无法启动。

1.2.2. 关于 nvr 区的使用说明

NVR0~1 区域可以用于保存用户的数据信息。

出于对 NVR 的保护，请不要频繁保存参数。如果需要频繁地保存参数，建议避免重复操作某个 NVR 区域。

2. SWD 的使用说明

2.1. SWD 作为通用 IO 的配置

SWD 的 IO 口默认是特殊功能，要使得 SWD_CLK 和 SWD_DATA 两个 IO 作为通用 IO，需要关闭 SWD，具体做法如下，

- 配置 makecode.ini 的【SWD_EN】，即 SWD_EN=0（默认配置）。
- 用户程序中配置寄存器 SYS_CON1 的 SWD_EN 为 0。

为了方便可以在 SWD 失能前，使用 SWD 接口烧录/升级程序。建议在 SWD 失能前，**延时 150ms 以上**。一旦 SWD 失能，需要进入超级模式，才能恢复 SWD 接口的默认功能。

2.2. SWD_CLK 引脚电平的使用说明

用户可通过配置 makecode.ini 的【SWD_CLK_PULLUP】决定 SWD_CLK 引脚开机后是否有上下拉功能。

SWD_CLK, SWD_DAT, PA13 三个引脚需要在芯片开机时满足特殊电平才能保证芯片正常开机。为了方便用户设计硬件电路，若 SWD_CLK 配置是上拉/下拉功能，开机时 SWD_CLK 电平要保证一直是高/低电平（SWD_CLK 配置上拉，开机时此引脚电平要为高；SWD_CLK 配置下拉，开机时此引脚电平要为低），这样，芯片便会正常开机。其次引脚不能接开机时影响电平的外设，例如：ADC，LED 等。

2.3. SWD 作为 LCD 功能

当 SWD 的 IO 被设置为 LCD 功能时，SWD 功能自动关闭，建议不要用 SWD 的 IO 口做 LCD 功能，方便用户使用 SWD 功能。

3. TK 使用说明

3.1. 移植说明

3.1.1. 文件移植

将原厂提供的按键相关文件移植到目标工程，文件包括 tk.lib, tk_cfg.c, tk_cfg.h, tk_define.h, tk_uart.h

3.1.2. 文件修改

在目标工程中，根据开发板实际环境的触摸按键总个数及触摸按键 IO 口，按照格式修改 tk_cfg.h 文件中的宏定义，如下图，**黑色框标注的是需要修改的内容，红色框标注的是注释；**

注：按键总个数要和按键 IO 口个数一致，否则程序会返回错误

```

#define TK_NUM          6          //按键总个数
/*格式
#define TK_0           TK_PA5 | 50 | TK_WKP_EN
                       IO口  | 阈值| 是否是tk唤醒IO
*/
#define TK_0           TK_PA5 | 50 | TK_WKP_EN
#define TK_1           TK_PA6 | 60
#define TK_2           TK_PA7 | 60
#define TK_3           TK_PA8 | 50
#define TK_4           TK_PA9 | 50
#define TK_5           TK_PA10 | 80
#define TK_6           TK_NONE | 0
#define TK_7           TK_NONE | 0
#define TK_8           TK_NONE | 0
#define TK_9           TK_NONE | 0
#define TK_10          TK_NONE | 0
#define TK_11          TK_NONE | 0
#define TK_12          TK_NONE | 0
#define TK_13          TK_NONE | 0
#define TK_14          TK_NONE | 0
#define TK_15          TK_NONE | 0
#define TK_16          TK_NONE | 0
#define TK_17          TK_NONE | 0
#define TK_18          TK_NONE | 0
#define TK_19          TK_NONE | 0
#define TK_20          TK_NONE | 0
#define TK_21          TK_NONE | 0
#define TK_22          TK_NONE | 0
#define TK_23          TK_NONE | 0
#define TK_24          TK_NONE | 0
#define TK_25          TK_NONE | 0

```

3.1.3. 阈值修改

经过调试之后获得合适的阈值，需要将阈值回填到 tk_cfg.h 文件中，如下图，注意按键的阈值与 IO 口是一一对应的关系，如图 TK_0 的 IO 口是 TK_PA5，对应的阈值是 50。

```

#define TK_NUM          6          //按键总个数
/*格式
#define TK_0           TK_PA5 | 50 | TK_WKP_EN
                       IO口  | 阈值| 是否是tk唤醒IO
*/
#define TK_0           TK_PA5 | 50 | TK_WKP_EN
#define TK_1           TK_PA6 | 60
#define TK_2           TK_PA7 | 60
#define TK_3           TK_PA8 | 50
#define TK_4           TK_PA9 | 50
#define TK_5           TK_PA10 | 80
#define TK_6           TK_NONE | 0
#define TK_7           TK_NONE | 0
#define TK_8           TK_NONE | 0
#define TK_9           TK_NONE | 0
#define TK_10          TK_NONE | 0
#define TK_11          TK_NONE | 0
#define TK_12          TK_NONE | 0
#define TK_13          TK_NONE | 0
#define TK_14          TK_NONE | 0
#define TK_15          TK_NONE | 0
#define TK_16          TK_NONE | 0
#define TK_17          TK_NONE | 0
#define TK_18          TK_NONE | 0
#define TK_19          TK_NONE | 0
#define TK_20          TK_NONE | 0
#define TK_21          TK_NONE | 0
#define TK_22          TK_NONE | 0
#define TK_23          TK_NONE | 0
#define TK_24          TK_NONE | 0
#define TK_25          TK_NONE | 0

```

3.2. 低功耗唤醒说明

3.2.1. 唤醒 IO 设置

tk_cfg.h 文件中，在触摸按键 IO 宏定义后加上 TK_WKP_EN，即将该 IO 设置为触摸唤醒 IO，如图中，TK_0 的 IO 口是 TK_PA5，对应的阈值 50，并且设置为唤醒 IO。

用户可以根据实际需求设置唤醒 IO 个数，可以单选也可以多选，任意触摸 IO 都可以被设置为触摸唤醒 IO。

```

#define TK_NUM           6           //按键总个数

/*格式
#define TK_0            TK_PA5 | 50 | TK_WKP_EN
                        IO口 | 阈值 | 是否是tk唤醒IO
*/
#define TK_0            TK_PA5 | 50 | TK_WKP_EN
#define TK_1            TK_PA6 | 60
#define TK_2            TK_PA7 | 60
#define TK_3            TK_PA8 | 50
#define TK_4            TK_PA9 | 50
#define TK_5            TK_PA10 | 80
#define TK_6            TK_NONE | 0
#define TK_7            TK_NONE | 0
#define TK_8            TK_NONE | 0
#define TK_9            TK_NONE | 0
#define TK_10           TK_NONE | 0
#define TK_11           TK_NONE | 0
#define TK_12           TK_NONE | 0
#define TK_13           TK_NONE | 0
#define TK_14           TK_NONE | 0
#define TK_15           TK_NONE | 0
#define TK_16           TK_NONE | 0
#define TK_17           TK_NONE | 0
#define TK_18           TK_NONE | 0
#define TK_19           TK_NONE | 0
#define TK_20           TK_NONE | 0
#define TK_21           TK_NONE | 0
#define TK_22           TK_NONE | 0
#define TK_23           TK_NONE | 0
#define TK_24           TK_NONE | 0
#define TK_25           TK_NONE | 0

```

3.2.2. 关闭其他模块

关闭其他低功耗模式下不需要运行的模块。

注: timer4 如果被正常程序使用到, 低功耗唤醒后需要重新初始化;

3.2.3. 调用低功耗唤醒函数

调用 tk_cfg.c 文件中的函数 tk_lp_init();

函数简单的解析如下:

```

void tk_lp_init()
{
    //IO, 除触摸按键 IO 外的 IO 设置为模拟状态, 注意将悬空的引脚设置为输入上拉
    ...

    //clk, 系统时钟切换到 lirc_256k, 关闭除 lirc_256k 之外其他的时钟
    ...

    tk_lp_ctrl_init(); //设置 TK 模块为低功耗模式

    //pmu, 关闭 pmu 模块
    ...

    tk_wakeup(); //进入睡眠, 等待唤醒

    //pmu, 恢复 pmu 模块
    ...

    //clk, 恢复时钟
    ...
}

```

3.2.4. 恢复现场

当触摸按键唤醒系统之后, 需要把之前关闭的其他模块打开, IO 恢复到初始状态。

3.3. 调试说明

3.3.1. JLINK 调试

将 tk_cfg.c 文件中宏定义 **TK_DEBUG_EN** 设置为 1，即

```
#define TK_SWD_DEBUG_EN 1
```

同时在 keil 的调试窗口中添加变量 tk_swd_buf，便可在该数组中观察到触摸按键的变化值。

3.3.2. 串口调试

基于已移植好触摸库的用户开发工程，用户需要在工程中配置三个地方

- 根据开发板实际环境，初始化 uart 接口，在 tk_cfg.c 中配置宏定义 TK_UART_SEL

```
#define TK_UART_SEL UART1_TX_PA1_RX_PA0
```

注：配置的选项在 tk_define.c 的枚举类型 TYPE_ENUM_TK_UART_SEL 中，必须使用该类型。

- 主程序调用函数 void tk_uart_init(); 初始化串口
- 调用函数 void tk_debug_process(void); 替换 void tk_process(void);

根据以上步骤正确配置后，编译下载到用户的开发芯片中，连接串口工具到 PC 端，同时在 PC 端打开触摸上位机，并设置好上位机参数，便可以通过图形界面看到触摸按键的采样情况，如下图所示。



3.4. 其他功能说明

3.4.1. 组合按键

在 tk_cfg.c 文件中，设置 tk_struct 的 two_keys_proc 和 multi_keys_proc，可以实现简单的组合按键控制。当设置为 0 的时候，功能失效；当设置为 1 的时候，功能如下：

```
tk_proc.two_keys_proc = 1; //当两个按键同时按下取变化比例大的按键值
```

```
tk_proc.multi_keys_proc = 1; //当三个或三个以上按键同时按下时，按键值为 0
```

3.4.2. 长按失键

长按失键，即长按按键，达到设置的时间自动释放按键值。长按失键的计时方式是 SysTick，所以需

要先配置 SysTick, 配置步骤如下:

- SystemTickInit(); //初始化

- 中断设置

```
void SysTick_Handler(void)
```

```
{
```

```
    SysTickCnt++; //在中断中, 用全局变量 SysTickCnt 计数,其他变量也可以
```

```
}
```

- 在 tk_cfg.c 中,将全局变量 SysTickCnt 配置到 tk_struct.systick 中

```
...
```

```
tk_struct.systick = (u32)&SysTickCnt;
```

```
...
```

- tk_proc.hold_downm_release_time 可以设置长按释放的时间, 该值小于 8, 0 等于 10s, 1 等于 20s, 以此类推

```
tk_proc.hold_downm_release_time = 0; //<8 0: 10s, 1: 20s,...,7:80s
```

3.4.3. 滑条

原厂提供一个滑条的参考算法,源码在 tk_slip.c 文件中,使用该文件首先需要将 TK_SWD_DEBUG_EN 宏定义设置为 1, 用户可以根据实际需求对算法进行修改。