文档编号：AN_137

上海东软载波微电子有限公司

应用笔记

ES7P2131

修订历史

版本	修订日期	修改概要
V1.0	2021-12-09	初版发布
V1.1	2022-03-21	1、增加 ADC 使用注意事项 2、增加基线更新时间计算说明
V1.2	2022-07-07	1、增加软件抖频说明 2、增加 ADC 模块 CHOP 使用注意事项 3、纠正内部基准描述 1.024V 为 1.0V
V1.3	2023-07-04	1、增加 IO 脉冲、端口负压、端口调制注意事项 2、增加 UART 回环功能的注意事项 3、增加 ADC 工作电压及参考电压稳定时间说明 4、删除部分 TK 库已固化配置的应用说明 5、补充修改 ADC 等应用例程说明 6、增加 WDT 应用说明
V1.4	2024-05-23	1、修订 TK 模块注意事项，针对 V1.4 版本的 SDK 增加上电速率要求说明 2、增加 TK 低功耗例程使用说明
V1.5	2024-08-14	1、增加关于 MRST 脚复用为 GPIO 的注意事项 2、增加 ADC 例程中校准函数的操作说明 3、补充完善 TK 低功耗例程说明
V1.6	2024-10-11	1、增加烧录调试时 ADC 注意事项

地　址：中国上海市徐汇区古美路 1515 号凤凰园 12 号楼 3 楼
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

目 录

内容目录

第1章 ES7P2131 应用注意

1.1 烧录调试

1.MRSTN 引脚复用为 GPIO 时，建议仍引出，做成 5 线制烧录调试口。用 4 线制烧录调试时需将上电延时（PWRTEB）功能使能。

2.仿真时调试口不能设定为高阻态，否则程序无法正常执行。

3.在调试 ADC 模块时，不支持动态设定、修改断点，调试 ADC 时需要预设好断点，然后进行调试。

1.2 IO 模块

1.不支持端口负压，应用时注意避免出现端口负压的情况。

2.在上下电过程中端口可能存在脉冲，对地挂接 10K 电阻可有效减弱脉冲。

3.端口调制输出在 IO 被配置为输出时才有对应波形，当复用其他功能时优先使能其他功能。

1.3 WDT 模块

窗口看门狗可配置 25%及 75%的窗口，禁止对 WDTCS<1:0>使用其他配置。

1.4 UART 模块

使用回环功能时 TX 需接上拉电阻。

1.5 ADC 模块

1.使用 ADC 时 ADVREF_EN 必须使能，使能后需等待约 1ms 稳定时间。

2.模拟输入在选择内部基准 1.2V 或 1.0V 时，需将 ADC 配置为低速模式（转换速度小于等于 250KHz）。

3.使用此模块时芯片供电要求在 2.8V 以上。

4.芯片的默认校准信息为 CHOP 禁止时的值，应用时注意保持 ADCTL2[1](保留位) = 0。

1.6 TK 模块

1.6.1 概述

ES7P2131 芯片最多支持 20 个触摸按键，1 路按键补偿通道。TK 模块工作频率支持 4MHz、2MHz、1MHz、0.5MHz，并且最高可支持 8MHz 工作(占空比须为 1/2)。模块抗干扰性增强，支持按键降耦功能、抖频功能，抖频范围为(±3%,±25%)。可支持 6 种工作模式：单通道单次扫描、单通道多次扫描、多通道单次轮询扫描、多通道单次合并扫描、多通道多次轮询扫描、多通道多次合并扫描。TK 模块采样数据可选择充电次数或者充电时间作为数据单位，数据结果可为原始计数值输出或者经放大系数运算后比值。TK 模块多次采样模式时，可对各通道采样数据进行累加或者平均输出。

1.6.2 上电速率

针对触控 SDK 例程（V1.4），从上电到基线确定时间 T_{base} 约 400ms（16M），因此对上电速率 $V_{power\ on}$ 有以下要求：

$$V_{power\ on} > \frac{V_{DD} - V_{BOR}}{T_{base}}$$

注：V_{DD} 为芯片供电电压，V_{BOR} 为设置的掉电复位电压。

1.6.3 合并扫描结果（TKSDA）

TK 模块若为单通道工作模式，则合并扫描结果寄存器（TKSDAL、TKSDAH）的采样结果即等于所选通道的采样值。若工作于多通道模式时，则等于所选择的 TK 通道中最小序号的采样值。多通道合并模式充放电频率 4M 下若合并通道超过 8 个以上，受负载影响波形会发生充不满的情况。

1.6.4 低功耗应用 TK 库

TK 库中的低功耗应用使用了外部触发功能，由 WDT 触发 TK 工作及唤醒。芯片低功耗进入 Idle 状态将自动清零 TKVREF_EN，由 WDT 唤醒后需通过软件使能。TKVREF 稳定至少需要 200us 以上，若按键个数少于 8 个，处理所需的时间就较少，唤醒处理完按键数据后就需要软件做延时处理。

1.6.5 基线更新时间计算

针对触控 AN138_ESTKLIB_ES7P213x V1.4 之前的版本，可依照以下计算方法估算。

按键充放电扫描时间由硬件和环境决定，实际时间需测量按键充放电波形计算时间后决定。若单个按键的充放电扫描时间为 150us，则基线更新时间为：

150*TK_NUM*TK_Samples_perscan*TK_BaseSamples_perscan us

针对触控 AN138_ESTKLIB_ES7P213x V1.4 版本，由参数 TK_BASE_UP_TIME 与计时的时基相乘的结果决定。

第2章ES7P2131 模块例程

2.1 系统时钟设置

操作说明：

7P2131 芯片提供 2 种振荡器，内部高速 RC 振荡器（16MHz）和内部低速 RC 振荡器（32KHz）。本例程中初始化时钟均初始化为选择高速时钟，不分频，所以系统时钟初始为 16MHz。本例程中有 4 个测试 case。分别实现从 HRC 16M 切换到 8M / 4M / 2M 和 32KHz LRC 时钟源。使能时钟输出功能，在 CLKO（PB0）引脚固定 128 分频输出系统时钟。

实现步骤：

a) 初始化 RAM；

b) 选择其中一个测试 case，进行时钟切换，从 CLKO 输出系统时钟。

2.2 窗口看门狗

操作说明：

本例程实现窗口看门狗中断的触发及喂狗操作，在正确喂狗后翻转 IO 电平。

实现步骤：

a) 初始化 RAM；

b) 设置 IO 及 WDT；

c) 使能总中断；

d) 在中断处理函数中喂狗并翻转电平。

2.3 GPIO 模块

操作说明：

本芯片最多支持 26 个 I/O 端口，共分为 PA，PB，PC，PE 共 4 组。所有 I/O 端口都是 TTL/SMT 输入和 CMOS 输出驱动。端口驱动能力可配置。

本例程实现 PB7 口输出，PB6 口检测 PB7 口电平状态。

实现步骤：

a) 初始化 RAM 和端口；

b) 改变 PB7 端口输出状态，检查 PB6 口数值是否符合预期。

2.4 IO 调制输出

操作说明：

本例程实现 PA0 口调制输出方波。

实现步骤：

a) 初始化 RAM 和端口；

b) PA0 配置为数字输出；

c) 设定频率与占空比并将调制输出使能。

2.5 低电压检测（LVD）

操作说明：

使用 ES7P2131 芯片的 LVD 模块，检测 VDD 电源。

本例程实现电压检测，当发生 VDD 电压从 2.8V 以上掉到 2.8V 以下时，PB1 口电平翻转，同时 VDD 高于 2.8V 时，PB2 输出低电平，VDD 低于 2.8V 时，PB2 输出高电平。

实现步骤：

a) 初始化 RAM 和端口；

b) 使能 LVDIE 和 GIE；

c) 配置 LVD 模式及触发电压等；

d) 在主函数中获取 LVD 输出状态并通过 IO 输出；

e) 在中断服务函数中检测 LVDIF，若是 LVD 引起的中断则清除中断标志，IO 翻转。

2.6 8 位定时/计数器（T10）

2.6.1 定时模式

操作说明：

使用芯片的 T10 定时器模块，在 PB1 端口输出一个周期为 4ms，占空比 50%的方波。

芯片使用 16MHz 系统时钟，则对应的 T10 定时器时钟源周期为 0.0625us。将预分频器分配给 T10 定时器，分频比为 1：64。

实现步骤：

a) 初始化 RAM 和端口；

b) 初始化 T10 定时器；

c) 使能 T10IE，GIE 中断；

d) 使能定时器；

e) 中断服务程序中判断中断标志，确定是 T10 中断后则清除 T10IF 标志位；

f) 先得到当前计数值，然后再设置定时初值为 0（由于没有重载功能，所以定时初值需要重新设置），翻转 IO。

2.6.2 计数模式

操作说明：

使用芯片的 T10 异步计数模式，PA4（T10CKI）与 PA5 短接，PA5 模拟外部时钟输入，当计数到发生溢出中断时，PC1 电平翻转。

实现步骤：

a) 初始化 RAM，端口；

b) 初始化 T10 定时器为同步计数模式；

c) 使能 T10IE，GIE 中断；

d) 使能定时器；

e) PA5 端口电平翻转，模拟一次时钟上升沿（PA5 与 PA4 T10CKI 短接）；

f) 读取 T10 计数值，查询是否进行了一次计数；

2.7 16 位多功能定时器（T20/T21）

2.7.1 定时模式

操作说明：

使用 ES7P2131 芯片的 T20 timer 模块，实现定时器功能。

本例程实现 T20 的定时功能，在 PB1 端口输出一个周期为 4ms，占空比 50%的方波。

芯片使用 16MHz 系统时钟，则对应的 T20 定时器时钟源周期为 0.0625us。设置预分频比为 1：8。T20P 寄存器初始值的计算公式应为：

2ms/0.0625us = T20×8（预分频比），计算得到 T20P = 4000(0x0FA0)。

实现步骤：

a) 初始化 RAM 和端口；

b) 配置 T20 模块；

c) 使能 T20VIE(溢出中断)，GIE 中断；

d) 使能定时器，等待计数器溢出中断；

e) 进入中断服务程序后判断并清中断标志，重载初值，翻转 IO。

2.7.2 双精度 PWM 模式

操作说明：

使用芯片的双精度 PWM 模式，在 PA5 端口及 PA6 端口实现周期为 100us~50us，占空比分别为 25%及 50%的方波输出（其中周期分四次递减至 50us，再将周期重新装载，占空比不变，以次反复）。

芯片使用 16MHz 系统时钟，则对应的 T20 定时器时钟源周期为 1/16MHz。T20 的预分频采用：1:1，T20 周期寄存器 T20P 初始值计算公式应为：

PWM 周期 100us = T20P ×（1/16MHz）× 1（预分频比），

计算得 T20P 初值 PERIOD = 1600（0x0640）。

PWM 脉宽 50us = T20PR ×（1/16MHz）× 1（预分频比），

计算得 T20R 初值 DUTY21 = 800（0x0320）。

这里以 PA5 端口的输出方波为例。PWM 占空比 50% = T20R / T20P。

使用时注意周期时间若设置过短（测试为 20us 内）会发生占空比更新延迟的情况。

实现步骤：

a) 初始化 RAM，端口；

b) 初始化 T20 为双精度 PWM 模式，输出极性低有效，使能 PWM200 和 PWM201，时钟不分频，分别设置周期寄存器和双精度寄存器；

c) 使能 T20 定时器，周期中断及全局中断；

d) 在中断服务程序中判断并清标志，重新设置周期寄存器和双精度寄存器。

2.8 ADC 程序设置

2.8.1 外部通道输入

操作说明：

使用 ES7P2131 芯片的 ADC 模块，采用查询方式实现对模拟输入电压的数字量转换。

ES7P2131 最多支持 10 个输入通道，最大转换分辨率为 12bit。

本例程实现对 AIN0 通道进行模数转换，调用 get_gain_offset();和 cal_adc();函数（可能产生盲区，用户根据应用情况选择是否使用），使用 Gain 和 Offset 对转换结果进行校准，测量的数字量保存在 ADC_AIN0[]数组中。

实现步骤：

a) 初始化 RAM，端口；

b) 初始化配置 ADC 模块，选择通道 0 进行 A/D 转换，使能 ADC 模块；

c) 启动两次 A/D 转换，忽略前两次获得的数据；

d) 获得 A/D 转换值，将结果保存在 ADC_AIN0[]数组中。

2.8.2 内部基准输入

操作说明：

使用 ES7P2131 芯片的 ADC 模块，采用查询方式实现对模拟输入电压的数字量转换。

ADC 处理包括采样和转换两个过程，本例程实现对内部 1.2V 基准通道进行模数转换，测量的数字量保存在 ADC_Data 变量中。

实现步骤：

a) 初始化 RAM；

b) 初始化配置 ADC 模块，选择基准 1.2V 进行 A/D 转换，使能 ADC 模块；

c) 启动多次 A/D 转换，并进行数据滤波处理；

d) 获得 A/D 转换值，将结果保存在 ADC_Data 变量中。

2.9 外部中断

操作说明：

使用 ES7P2131 的外部中断功能，对 PINT0（PB7 端口）外部电平变化进行判别。PINT0（PB7 端口）的双边沿或上升沿会产生外部中断标志，若使能外部中断和全局中断，则会进入中断函数。

例程实现对 PB6 口进行电平进行电平检测，将 PB6 与 PB7 端口短接，PB7 作为输出模拟电平变化，PBO 口作为中断标志位，发生一次中断 PB0 口电平进行一次翻转。

实现步骤：

a) 初始化 RAM，端口；

b) 配置为双边沿触发；

c) 使能外部中断，使能全局中断；

d) 改变 PB6 输出状态使与之相连接的 PB7（PINT0）检测到电平变化；

e) 有电平变化则进入中断服务程序，清除中断标志并翻转 IO。

2.10 内部 FLASH 读写

操作说明：

ES7P2131 内部有 16K 字 FLASH 程序存储器，程序存储器中可配置 1K 字作为 FLASH 数据存储区。

查表读操作通过查表读指令将 FRA 所指向的地址单元中的两个字读入 ROMD 和 ROMD1 中。

数据 FLASH 的写以双字为单位，写操作前必须先擦除所写单元所在的页，故写数据 FLASH 时包含三个基本操作：数据备份，页擦除和字编程。

芯片页擦除时间至少为 2ms，单个地址编程时间至少为 30us，此期间芯片处于暂停状态。

对 DataFLASH 存储器读/写操作前，芯片配置字中需使能 IAPEN 位。

建议用户在执行页擦除操作时关闭 WDT(在配置字选项中关闭)，避免执行擦除操作时因看门狗复位导致芯片误操作。如果使能 WDT，须设置合理的 WDT 溢出时间，并合理清狗。建议用户在执行擦写操作前关闭总中断使能位 GIE，避免执行擦写操作时因中断导致芯片误操作。

关于 Flash 存储器的可靠性操作方法详见《AN062_应用笔记_MCU 片内非易失性存储器操作》。

例程中 Flash 操作对 IAP 数据区操作，向 4000H 地址（DataFLASH 起始地址）写入 16 个字节数据，所以应该先对该页进行擦除，然后向其中写 16 个字节数据，接着从此地址开始读出 16 个字节数据进行校验看是否成功写入 DataFLASH。

实现步骤：

a) 初始化 RAM，WDT；

b) 初始化地址并读取当前数据；

c) 重新初始化地址并执行页擦除；

d) 准备填入的数据 buffer 并执行 flash 写操作；

e) 重新初始化地址并执行 flash 读操作。

2.11 UART 通讯

2.11.1 UART 收发

操作说明：

异步收发器支持 8/9 位数据格式，支持高速/低速模式。

波特率计算公式：

低速模式：BRGHn=0, Baudrate=Fosc/(64×(BRR<7:0>+1))

高速模式：BRGHn=1, Baudrate=Fosc/(16×(BRR<7:0>+1))

UART 通讯时，发送/接收中断标志通过写/读对应的 Buffer 硬件自动清零，通过 USB 转串口模块，和 PC 机相连进行数据通信。

例程实现 UART 接收和发送功能，初始化先发送一个字符串<UART>，接着在中断中接收并将接收的值再发出。9600bps，8 位数据格式。

实现步骤：

a) 初始化 RAM，端口；

b) 初始化 uart，设置波特率位 9600，默认为 8 位模式；

c) 执行发送；

d) 使能 UART 接收及 GIE 中断；

e) 在中断服务程序中处理接收数据并将接收的值再发出。

2.11.2 单线模式

操作说明：

1.使用 ES7P2131 芯片的 UART 模块 TX 脚，需要外部上拉。

2.本例程实现 UART 发送与接收功能，主机先发送 40123，接着接收从机发送的 45678 可更改 TX_num 值实现任意长度的数据发送与接收。9600bps，8 位数据格式。

实现步骤：

a) 初始化变量，初始化 uart，设置波特率位 9600，默认为 8 位模式；

b) 配置为发送或者接收模式；

c) 使能 GIE 中断；

d) 在中断服务程序中处理发送接收数据。

2.12 I2C 从动模式

操作说明：

当 I2CSRIF 标志位置 1，且 I2CRW 位为 0 时表示主机写从机，即从机接收数据。从机接收数据缓冲器满时，中断标志位 I2CRBIF 置位，读取一次接收数据缓冲器 I2CRB，I2CRBIF 标志位自动清零，其他使用注意事项请参考芯片数据手册。

当 I2CSRIF 标志位置位，且 I2CRW 位为 1 时表示主机读从机，即从机发送数据。I2CTB 发送数据缓冲器不满时，中断标志位 I2CTBIF 置位，写发送数据缓冲器 I2CTB，若数据缓冲器写满，I2CTBIF 标志位自动清零，若发出一个字节，数据缓冲器未满，则中断标志位 I2CTBIF 置位。

从机接收到 STOP 信号时，收发数据缓冲器不会自动清零，需置位 I2CRST 位来软件复位 I2C 模块。复位后，需重新初始化 I2C 模块。

例程中主机写一个数据给从机，再读回两个数据。从机将接收到的数据以原码和反码的形式发回给主机。

实现步骤：

a) 初始化 RAM，端口；

b) 复位 I2C 模块；

c) I2C 端口使能，设为上拉及开漏输出，发送应答 ACK；

d) 设置从机的地址，中断功能使能；

e) 使能 I2C 模块及 I2C 通讯中断；

f) 使能 GIE；

g) 在中断服务函数中对 I2C 的中断状态标志和 I2C 状态标志做判断并处理。

2.13 TK 低功耗

操作说明：

低功耗过程中，由 WDT 定时唤醒进行 TK 扫描及按键识别过程。

由函数 tk_reg_config();对触控模块做初始化配置。根据应用控制逻辑，当符合进入低功耗条件时将 tk_sleep_flag 置位。

用户还需要根据应用在函数 prepareforsleepmode_init()以及 wakeupfromsleepmode_init()中调整相关 IO 及外设配置。

例程中预留了端口及按键中断，通过宏定义 PINT_WAKE_FUNC 及 KINT_WAKE_FUNC 使能，在函数 prepareforsleepmode_init()、wakeupfromsleepmode_init()、tk_standby()调整端口配置。

实现步骤：

a) 触控模块初始化；

b) 根据应用条件置位低功耗工作标志 tk_sleep_flag（用户自行添加程序）；

c) 为低功耗模式调整外设配置；

d) 低功耗模式，无按键时循环休眠唤醒过程，识别到按键操作时执行返回正常模式过程；

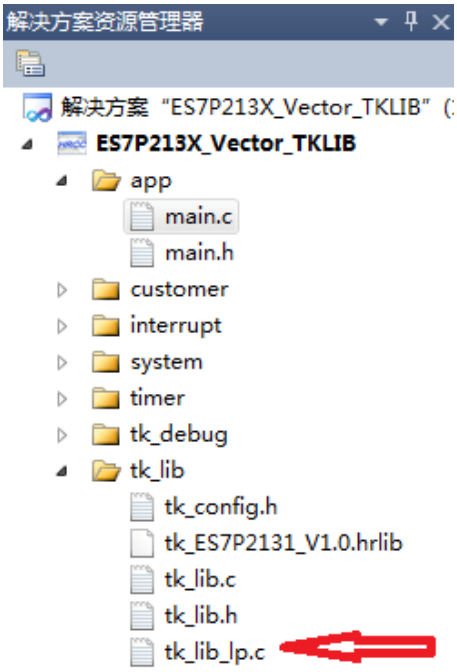
e) 为正常模式调整外设配置；

f) 返回正常模式，重新等待低功耗工作标志置位；

移植示例：

以 ESTK_ES7P213x_Button_Vector_V1.4 为例做移植步骤说明。

1) 拷贝 tk_config.h 及 tk_lib_lp.c 到对应的工程文件夹 tk_lib 下，并在工程中添加文件



2) 拷贝低功耗处理过程（ESTK_ES7P213x_Button_Vector_V1.4 中已有触控模块的初始化配置）。

```
void main()
{
    tk_reg_config();
    while(1)
    {
        clr_wdt();
        if(tk_sleep_flag)
        {
            prepareforsleepmode_init();
            while(tk_sleep_flag)
            {
                tk_standby();
            }
        }
    }
}
```

```
void main()
{
    delay100ms(1);          //上电延时，避开_
    mcu_init();             //系统配置封装函
    gpio_set();             //根据应用电路配
    tk_pin();               //TK通道管脚的通
    tk_init();              //TK初始化必要函
    timer_t10_init();       //2ms定时器用于进
    #if TK_UART_FUNC == ON
    uart_init();            //用于与上位机通
    #endif
    customer_key_init();
    GIE = 1;
    while(1)
    {
        clr_wdt();
        timer_check();
        tk_service();
        customer_key_handler();
        if(tk_sleep_flag)
        {
            prepareforsleepmode_init();
            while(tk_sleep_flag)
            {
                tk_standby();
            }
        }
    }
}
```

3) 添加进入低功耗的应用判断逻辑，比如20S无按键操作时进入休眠。可在timer_check()函数中增加如下程序。

```
uint no_key_time = 0;
void timer_check(void)
{
    if (timer_10ms < 5)      //2ms
        return;

    timer_10ms = 0;

    tk_base_up_lock_cnt();  //基
#if...
#endif

#if...
#endif

    if(tk_state == 0)
    {
        no_key_time++;
        if(no_key_time > 2000)
        {
            no_key_time = 0;
            tk_sleep_flag = 1;
        }
    }
    else
    {
        no_key_time = 0;
    }
}
```

4) 执行IDLE指令前，通过关闭各功能模块使能位，可使芯片在执行IDLE指令后进入更深程度的低功耗状态。比如在ESTK_ES7P213x_Button_Vector_V1.4中的定时器在低功耗过程中未使用，可在prepareforsleepmode_init()及wakeupfromsleepmode_init()修改配置以降低功耗，注意模块时钟（CLKG）使能时配置对应模块的寄存器才能生效。同时应避免数字输入的I/O管脚处于浮空状态，需将这些管脚接固定电平，或在芯片外部进行上拉或下拉处理，否则会引起I/O端口漏电。

```
void prepareforsleepmode_init(void)
{
    ......

    T10IE = 0;
    T10EN = 0;      //关闭模块使能
    CLKG &= 0xFE;   //关闭模块时钟

    ......
}
```

```
void wakeupfromsleepmode_init(void)
{
    ......

    CLKG |= 0x01;   //使能模块时钟
    T10EN = 1;      //模块使能
    T10IE = 1;
    T10IF = 0;

    ......
}
```

若正常模式及低功耗模式下所用触控引脚相同，则不必再重新配置触控引脚（如下图）。若不同，则在wakeupfromsleepmode_init()函数中应调用tk_pin();还原正常模式下的配置。

```
void prepareforsleepmode_init(void)
{
    ......

    for(i = 0;i < TK_NUM_LP;i++) {
        j = (ch_table_lp[i] & 0x1F);
        j = reg_port_lp[j];
        k = (ch_table_lp[i] & 0xE0);
        if(k == PA_PORT) {              // PA
            //PAT |= j;//输入
            PAS |= j;//模拟
        }
        else if(k == PB_PORT) {         // PB
            //PBT |= j;
            PBS |= j;
        }
        else if(k == PE_PORT) {         // PE
            //PET |= j;
            PES |= j;
        }
    }

    //Cx
    PCT0 = 1;
    PCS0 = 1;
    ......
}
```

5) 根据应用要求修改低功耗下的按键个数、通道及阈值设定。

```
#define  TK_NUM_LP                  4

#define  TK_CHANNEL_SEL_LP          0x000001e0

#define  TK_CHANNEL_LP_0            TK5
#define  TK_CHANNEL_LP_1            TK6
#define  TK_CHANNEL_LP_2            TK7
#define  TK_CHANNEL_LP_3            TK8

#define  TK_THD_CHANNEL_LP_0        80
#define  TK_THD_CHANNEL_LP_1        80
#define  TK_THD_CHANNEL_LP_2        80
#define  TK_THD_CHANNEL_LP_3        80
```