低功耗模式说明

1、简介

11xx 是一颗双核 MCU，所以 M0 和 bt 核都必须进入低功耗状态，才能成功进入低功耗模式；低功耗模式分为两种，一种是 lpm 模式（芯片周期性进入休眠）；一种是 Hibernate 模式（待机模式，该模式下只能被 GPIO 唤醒）。BT 的低功耗模式是已经配置好的，下面主要讲 M0 的低功耗模式配置。

2、低功耗模式使能寄存器-mem_lpm_mode

Address	byte	功能	说明	R/W	复位值
0x4131	1	1：开始低功耗模式 0：关闭低功耗模式	低功耗模式使能	R/W	---

3、休眠时间寄存器 0 - mem_lpm_interval

Address	byte	功能	说明	R/W	复位值
0x41f3	2	设置未连接状态下的休眠的时间	单位:0.625ms	R/W	---

4、休眠时间寄存器 1- mem_time_wake_gap

功能：根据蓝牙的连接参数的 interval 来设置 休眠时间

备注：蓝牙连接状态下，休眠时间不能超过 1s，否则容易断开蓝牙。

Address	byte	功能	说明	R/W	复位值
0x4e4a	2	设置连接状态下的休眠的时间	单位：1.25ms	R/W	---

实际休眠时间 = mem_time_wake_gap*1.25ms-interval*1.25ms 。

5、低电平唤醒寄存器-mem_gpio_wakeup_low

Address	bit	功能	说明	R/W	复位值
0x00ab	0-31	每一个 bit 表示一个 GPIO 的低电平唤醒功能	1：使能 0：关闭	R/W	---

6、高电平唤醒寄存器-mem_gpio_wakeup_high

Address	bit	功能	说明	R/W	复位值
0x00af	0-31	每一个 bit 表示一个 GPIO 的高电平唤醒功能	1：使能 0：关闭	R/W	---

7、休眠前设置 GPIO 状态和唤醒- Bt_ActionBeforeLpm()

该函数的目的是讲 GPIO 的状态设置为一个确定态（上拉或者下拉）；同时设置根据设置的 GPIO 状态，来决定是否设置该 GPIO 的唤醒功能。

对于接有负载（例如，电阻，灯）的 GPIO 需要配置下拉，避免上拉状态下负载两段有电压造成漏电；

同理，对于外部有高电平输入的 GPIO 需要配置成上拉。

8、LPM 模式

低功耗模式下，芯片默认进入 LPM 模式，该模式下，底层会根据当前芯片的运行状态自动发起休眠行为，M0 无法强制进入休眠，M0 收到休眠指令后，然后根据自身的一个运行状态判断是否进入休眠。因此低功耗模式下，M0 需要不断去获取当前芯片运行状态，来判断是否进入休眠。因此，M0 的 main 函数需要按照下面的框架去编写代码；

```
int main( )
{
    代码段 1     ;
    while(1)
    {
        switch (HREAD(IPC_MCU_STATE))
        {
            case IPC_MCU_STATE_RUNNING:
                代码段 2     ;
                break;
            case IPC_MCU_STATE_LMP:
                if (IPC_IsTxBuffEmpty())
                {
                    OS_ENTER_CRITICAL();
                    Bt_ActionBeforeLpm();
                    HWRITE(IPC_MCU_STATE,IPC_MCU_STATE_STOP);
                }
                else{
                HWRITE(IPC_MCU_STATE,IPC_MCU_STATE_RUNNING);
                }
                break;
            case IPC_MCU_STATE_LMP:
                OS_ENTER_CRITICAL();
                Bt_ActionBeforeHibernate();
                HWRITE(IPC_MCU_STATE,IPC_MCU_STATE_STOP);
                break;
        }
    }
}
```

解析:

A、代码段 1:

M0 从唤醒中醒来是需要重新运行 main 函数的，所以代码段 1 的代码是每次上电或者唤醒都会运行，所以这段代码运行时间的长短会影响芯片唤醒工作的时间的长短，从而影响整体的休眠平均电流。

为了减少代码段 1 的运行，LPM 模式下有如下操作:

1、 修改 startip.s 文件

运行 main 函数前，先判断 M0 是否是从休眠中醒来，如果是，则跳过对内存的初始化过程，直接运行 mian 函数，保留休眠前的内存数据。

startip.s 修改代码如下:

```
; Reset Handler

Reset_Handler    PROC
                 EXPORT  Reset_Handler             [WEAK]
                 IMPORT  __main
                 LDR     R0,= LOADRAMFLAG
                 LDR     R0,[R0]
                 LDRB    R0,[R0]
                 CMP     R0,#0
                 BEQ     RUN_TO_MAIN
                 LDR     R0, =__main
                 BX      R0
                 ENDP

RUN_TO_MAIN      PROC
                 IMPORT  main
                 LDR     R0, = main
                 BX      R0
                 ENDP

                 align 4
; mem_wake_flag
LOADRAMFLAG      DCD     0x10004704
Default_Handler PROC
; ToDo:  Add here the export definition for the device specific external interrupts handler
                 EXPORT      SYStick_handle      [WEAK]
                 EXPORT      USB_IRQHandler      [WEAK]
                 EXPORT      IIC_IRQHandler      [WEAK]
                 EXPORT      QSPI_IRQHandler     [WEAK]
```

2、 在代码段 1 进行　上电/唤醒行为 判断

通过读取 mem_wake_flag 的值去判断上电/唤醒行为。

如果是上电，则执行初始化流程。

如果是唤醒，则跳过初始化流程，

如果用到定时器中断，需要在唤醒的时候重新设置定时器中断。

```
if(HREAD(mem_wake_flag)==POWERON_WAKE)
{
        IPC_init(&gTIPCHandleCb);          //蓝牙回调函数初始化
        SYS_TimerInit();
}
else
{
    SysTick_Config(SYSTEM_CLOCK/1000);
}
```

B、 当前 CPU 运行状态寄存器-IPC_MCU_STATE:

Address	byte	功能	说明	R/W	复位值
0x4334	1	0: running 1：待机模式 2：LPM 模式 3：MCU 停止运行		R/W	---

底层会自动更新该寄存器的值，M0 通过不断去读取该寄存器的值来判断 MCU 当前运行状态，并执行相应操作。

C、代码段 2

代码段 2 为芯片正在运行时所要执行的操作。

D、IPC_IsTxBuffEmpty()　　//检测 BT 核是否有数据发送过来

M0 检测到芯片要进入休眠的状态后，先判断下 BT 核有没有数据发送过来，如果有数据，则 M0 发出继续运行的指令HWRITE(IPC_MCU_STATE,IPC_MCU_STATE_RUNNING);。

反之，

E、Bt_ActionBeforeLpm();

M0 在进入休眠前的一些动作，设置所有的 GPIO 唤醒功能，配置 GPIO 为确定态（所有 GPIO 配置成上拉或者下拉）。

GPIO 的配置需要根据硬件来配置，否则容易造成漏电，最近直接的现象就是，测得休眠电流偏大。如果 PCBA 该 GPIO 有接负载（电阻，（高有效的）灯等，该 GPIO 需要配置成下拉，反之，配置成上拉。

9、待机模式

函数名：Bt_enter_Hibernate（）

功能：进入待机模式，只能通过 GPIO 唤醒。