零死角玩转STM32

电源管理—实现低功耗

淘宝：firestm32.taobao.com

论坛： www.chuxue123.com

扫描进入淘宝店铺

主讲内容

01 STM32的电源管理简介

02 低功耗模式

03 电源管理相关的库函数及命令

04 电源管理实验

参考资料:《零死角玩转STM32》

“电源管理—实现低功耗”章节

电源管理—实现低功耗

电源管理相关的库函数及命令

STM32标准库对电源管理提供了完善的函数及命令，使用它们可以方便地进行控制。

配置PVD监控功能

PVD可监控VDD的电压，当它低于阈值时可产生PVD中断以让系统进行紧急处理，这个阈值可以直接使用库函数PWR_PVDLevelConfig配置成前面阈值表中说明的阈值等级。

电源管理—实现低功耗

WFI与WFE命令

在前面可了解到进入各种低功耗模式时都需要调用WFI或WFE命令，它们实质上都是内核指令，在库文件core_cmInstr.h中把这些指令封装成了函数：

代码清单 42-1 WFI 与 WFE 的指令定义(core_cmInstr.h 文件)

```
1
 2 /** \brief  Wait For Interrupt
 3
 4     Wait For Interrupt is a hint instruction that suspends execution
 5     until one of a number of events occurs.
 6  */
 7 #define __WFI                                   __wfi
 8
 9
10 /** \brief  Wait For Event
11
12     Wait For Event is a hint instruction that permits the processor to
enter
13     a low-power state until one of a number of events occurs.
14  */
15 #define __WFE                                   __wfe
```

电源管理—实现低功耗

WFI与WFE命令

对于这两个指令，应用时只需要知道，调用它们都能进入低功耗模式，需要使用函数的格式“__WFI();”和“__WFE();”来调用(因为__wfi及__wfe是编译器内置的函数，函数内部使用调用了相应的汇编指令)。

其中WFI指令决定了它需要用中断唤醒，而WFE则决定了它可用事件来唤醒，关于它们更详细的区别可查阅《cortex-CM3/CM4权威指南》了解。

电源管理—实现低功耗

进入停止模式

直接调用WFI和WFE指令可以进入睡眠模式，而进入停止模式则还需要在调用指令前设置一些寄存器位，STM32标准库把这部分的操作封装到PWR_EnterSTOPMode函数中了，它的定义如下：

```c
/**
  * @brief 进入停止模式
  *
  * @note   在停止模式下所有 I/O 的会保持在停止前的状态
  * @note   从停止模式唤醒后，会使用 HSI 作为时钟源
  * @note   调压器若工作在低功耗模式，可减少功耗，但唤醒时会增加延迟
  * @param  PWR_Regulator: 设置停止模式时调压器的工作模式
  *           @arg PWR_MainRegulator_ON: 调压器正常运行
  *           @arg PWR_LowPowerRegulator_ON: 调压器低功耗运行
  * @param  PWR_STOPEntry: 设置使用 WFI 还是 WFE 进入停止模式
  *           @arg PWR_STOPEntry_WFI: WFI 进入停止模式
  *           @arg PWR_STOPEntry_WFE: WFE 进入停止模式
  * @retval None
  */
void PWR_EnterSTOPMode(uint32_t PWR_Regulator, uint8_t PWR_STOPEntry)
{
    uint32_t tmpreg = 0;

    /* 设置调压器的模式 ------------*/
    tmpreg = PWR->CR;
    /* 清除 PDDS 及 LPDS 位 */
    tmpreg &= CR_DS_MASK;

    /* 根据 PWR_Regulator 的值(调压器工作模式)配置 LPDS,MRLVDS 及 LPLVDS 位 */
    tmpreg |= PWR_Regulator;

    /* 写入参数值到寄存器 */
    PWR->CR = tmpreg;

    /* 设置内核寄存器的 SLEEPDEEP 位 */
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* 设置进入停止模式的方式------------------*/
    if (PWR_STOPEntry == PWR_STOPEntry_WFI) {
        /* 需要中断唤醒*/
        __WFI();
    } else {
        /* 需要事件唤醒 */
        __WFE();
    }
    /* 以下的程序是当重新唤醒时才执行的，清除 SLEEPDEEP 位的状态*/
    SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP_Msk);
}
```

电源管理—实现低功耗

这个函数有两个输入参数，分别用于控制调压器的模式及选择使用WFI或WFE停止，代码中先是根据调压器的模式配置PWR_CR寄存器，再把内核寄存器的SLEEPDEEP位置1，这样再调用WFI或WFE命令时，STM32就不是睡眠，而是进入停止模式了。函数结尾处的语句用于复位SLEEPDEEP位的状态，由于它是在WFI及WFE指令之后的，所以这部分代码是在STM32被唤醒的时候才会执行。

要注意的是进入停止模式后，STM32的所有I/O都保持在停止前的状态，而当它被唤醒时，STM32使用HSI作为系统时钟(16MHz)运行，由于系统时钟会影响很多外设的工作状态，所以一般我们在唤醒后会重新开启HSE，把系统时钟设置会原来的状态。

电源管理—实现低功耗

前面提到在停止模式中还可以控制内部FLASH的供电，控制FLASH是进入掉电状态还是正常供电状态，这可以使用库函数

PWR_FlashPowerDownCmd配置，它其实只是封装了一个对FPDS寄存器位操作的语句，这个函数需要在进入停止模式前被调用，即应用时需要把它放在上面的PWR_EnterSTOPMode之前。

代码清单 42-3 控制 FLASH 的供电状态

```
1  /**
2    * @brief  设置内部 FLASH 在停止模式时是否工作在掉电状态
3    *         掉电状态可使功耗更低，但唤醒时会增加延迟
4    * @param  NewState:
5    *             ENABLE:FLASH 掉电
6    *             DISABLE:FLASH 正常运行
7    * @retval None
8    */
9  void PWR_FlashPowerDownCmd(FunctionalState NewState)
10 {
11     /*配置 FPDS 寄存器位*/
12     *(__IO uint32_t *) CR_FPDS_BB = (uint32_t)NewState;
13 }
```

电源管理—实现低功耗

进入待机模式

类似地，STM32标准库也提供了控制进入待机模式的函数，其定义如下：

```
1  /**
2    * @brief 进入待机模式
3    * @note   待机模式时，除以下引脚，其余引脚都在高阻态：
4    *           -复位引脚
5    *           - RTC_AF1 引脚 (PC13) (需要使能侵入检测、时间戳事件或 RTC 闹钟事件)
6    *           - RTC_AF2 引脚 (PI8) (需要使能侵入检测或时间戳事件)
7    *           - WKUP 引脚 (PA0) (需要使能 WKUP 唤醒功能)
8    * @note  在调用本函数前还需要清除 WUF 寄存器位
9    * @param  None
10   * @retval None
11   */
12 void PWR_EnterSTANDBYMode(void)
13 {
14     /* 选择待机模式 */
15     PWR->CR |= PWR_CR_PDDS;
16
17     /* 设置内核寄存器的 SLEEPDEEP 位 */
18     SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
19
20     /* 存储操作完毕时才能进入待机模式，使用以下语句确保存储操作执行完毕 */
21
22     __force_stores();
23
24     /* 等待中断唤醒 */
25     __WFI();
26 }
```

电源管理—实现低功耗

该函数中先配置了PDDS寄存器位及SLEEPDEEP寄存器位，接着调用__force_stores函数确保存储操作完毕后再调用WFI指令，从而进入待机模式。这里值得注意的是，待机模式也可以使用WFE指令进入的，如果您有需要可以自行修改；另外，由于这个函数没有操作WUF寄存器位，所以在实际应用中，调用本函数前，还需要清空WUF寄存器位才能进入待机模式。

在进入待机模式后，除了被使能了的用于唤醒的I/O，其余I/O都进入高阻态，而从待机模式唤醒后，相当于复位STM32芯片，程序重新从头开始执行。

THANKS

论坛： www.chuxue123.com

淘宝：firestm32.taobao.com

扫描进入淘宝店铺