零死角玩转STM32

电源管理—实现低功耗

淘宝：firestm32.taobao.com

论坛： www.firebbs.cn

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_cm3.h中把这些指令封装成了函数：

代码清单 41-1 WFI 与 WFE 的指令定义(core_cm3.h 文件)

```
1
2  /** brief   等待中断
3
4      等待中断 是一个暂停执行指令
5      暂停至任意中断产生后被唤醒
6   */
7  #define __WFI                                   __wfi
8
9
10 /** brief   等待事件
11
12       等待事件 是一个暂停执行指令
13       暂停至任意事件产生后被唤醒
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：

代码清单 41-2 进入停止模式

```
1
2  /**
3  * @brief 进入停止模式
4  *
5  * @note    在停止模式下所有 I/O 的会保持在停止前的状态
6  * @note    从停止模式唤醒后，会使用 HSI 作为时钟源
7  * @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
8  * @param   PWR_Regulator: 设置停止模式时调压器的工作模式
9  *             @arg PWR_MainRegulator_ON: 调压器正常运行
10 *             @arg PWR_Regulator_LowPower: 调压器低功耗运行
11 * @param   PWR_STOPEntry: 设置使用 WFI 还是 WFE 进入停止模式
12 *             @arg PWR_STOPEntry_WFI: WFI 进入停止模式
13 *             @arg PWR_STOPEntry_WFE: WFE 进入停止模式
14 * @retval None
15 */
16 void PWR_EnterSTOPMode(uint32_t PWR_Regulator, uint8_t PWR_STOPEntry)
17 {
18     uint32_t tmpreg = 0;
19     /* 检查参数 */
20     assert_param(IS_PWR_REGULATOR(PWR_Regulator));
21     assert_param(IS_PWR_STOP_ENTRY(PWR_STOPEntry));
22
23     /* 设置调压器的模式 ------------*/
24     tmpreg = PWR->CR;
25     /* 清除 PDDS 及 LPDS 位 */
26     tmpreg &= CR_DS_MASK;
27     /* 根据 PWR_Regulator 的值(调压器工作模式)配置 LPDS,MRLVDS 及 LPLVDS 位*/
28     tmpreg |= PWR_Regulator;
29     /* 写入参数值到寄存器 */
30     PWR->CR = tmpreg;
31     /* 设置内核寄存器的 SLEEPDEEP 位 */
32     SCB->SCR |= SCB_SCR_SLEEPDEEP;
33
34     /* 设置进入停止模式的方式------------------*/
35     if (PWR_STOPEntry == PWR_STOPEntry_WFI) {
36         /* 需要中断唤醒 */
37         __WFI();
38     } else {
39         /* 需要事件唤醒 */
40         __WFE();
41     }
42
43     /* 以下的程序是当重新唤醒时才执行的，清除 SLEEPDEEP 位的状态 */
44     SCB->SCR &= (uint32_t)~((uint32_t)SCB_SCR_SLEEPDEEP);
45 }
46
```

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(8MHz)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，把系统时钟设置回原来的状态。

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：

代码清单 41-3 进入待机模式

```
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     /* 清除 Wake-up 标志 */
15     PWR->CR |= PWR CR CWUF;
16     /* 选择待机模式 */
17     PWR->CR |= PWR_CR_PDDS;
18     /* 设置内核寄存器的 SLEEPDEEP 位  */
19     SCB->SCR |= SCB_SCR_SLEEPDEEP;
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 #if defined ( __CC_ARM   )
22     __force_stores();
23 #endif
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。

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.firebbs.cn

淘宝：firestm32.taobao.com

扫描进入淘宝店铺