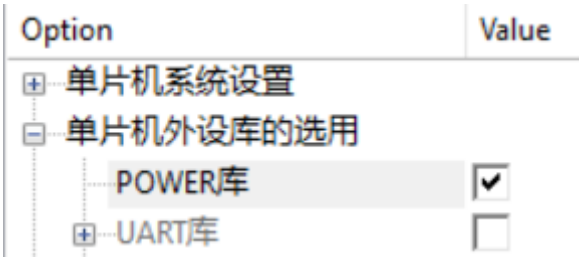
POWER库

POWER库是有关于单片机的电源操作库。在使用本库之前先到ecbm_core.h里使能。双击打开ecbm_core.h文件，然后进入图形化配置界面，使能POWER库。



API

power_reset_code

函数原型：void power_reset_code(void);

描述

单片机复位函数，复位后从main函数开始运行。

输入

无

输出

无

返回值

无

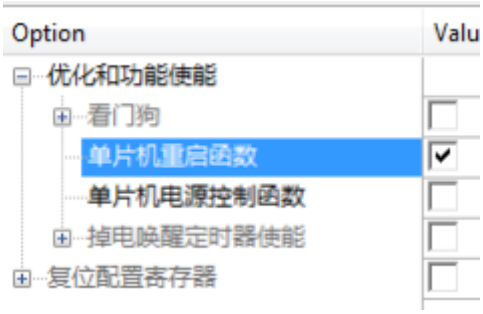
调用例程

```
if(SBUF=='*'){//串口接收到*号的时候，
    power_reset_code();//重启单片机。
}
```

注意事项

1. 如果是想做软重启自动下载功能，那么这个函数不能满足要求，需要power_reset_isp函数。

2. 为了优化空间，本函数默认不使能。请在power.h文件的图形化配置界面里使能【单片机重启函数】。



power_reset_isp

函数原型：void power_reset_isp(void);

描述

单片机复位函数，复位后从单片机BootLoader开始运行。

输入

无

输出

无

返回值

无

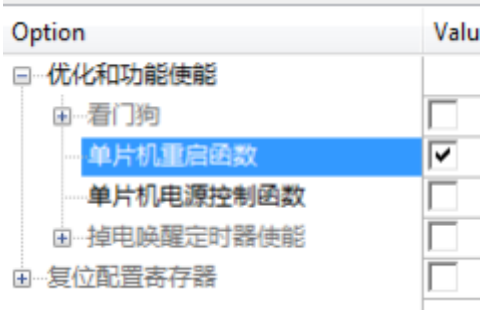
调用例程

自动下载功能的核心：

```
if(SBUF==0x7F){//STC-ISP在下载前会发送一连串的0x7F。
    count++;//每收到一次就让计数值+1。
    if(count==200){//收到200个0x7F的时候，认为是合法的下载流。
        power_reset_isp();//重启单片机，下载代码。
    }
}else{//如果收到的不是0x7F，说明这可能是正常的数据。
    count=0;//清零计数值。
}
```

注意事项

1. **为了优化空间，本函数默认不使能。**请在power.h文件的图形化配置界面里使能【单片机重启函数】。



power_powerdown

函数原型：void power_powerdown(void);

描述

掉电函数，单片机进入掉电模式，CPU和外设的电源都会被关掉。

输入

无

输出

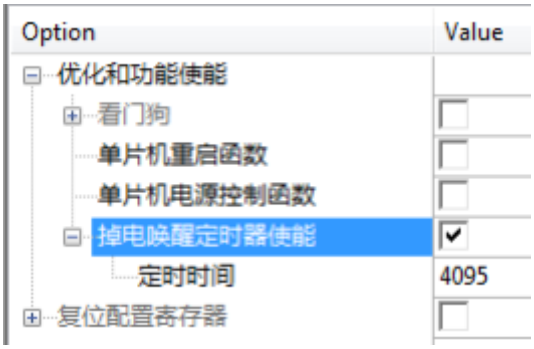
无

返回值

无

参数配置

在掉电模式下，单片机可以由内部的一个掉电定时器唤醒。如果要使用定时唤醒功能，首先在图形化配置界面下使能【掉电唤醒定时器使能】，并且在【定时时间】选项处填写掉电定时器的定时时间。



掉电定时器使用内部的32KHz的频率工作，其定时时间计算公式和预估时间如下所示：

$$掉电唤醒定时器定时时间 = \frac{10^6 \times 16 \times 计数次数}{F_{wt}} (微秒)$$

调用例程

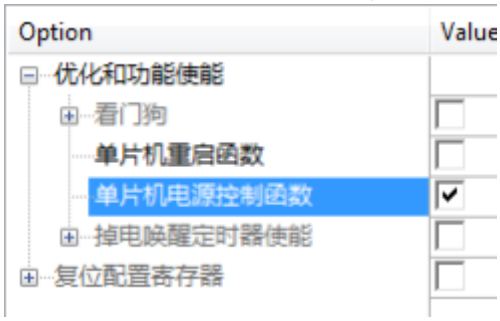
关机键的核心：

```
if(KEY_OFF==0){//当按键OFF按下时。
    power_powerdown();//进入掉电模式，此时功耗很低。
}
```

注意事项

1. 掉电模式可以通过外部触发的中断唤醒。不仅仅是外部中断脚，串口接收中断或者IIC起始帧中断这种也行，只要这个信号是从外部进来的就可以。
2. 内部能唤醒单片机的只有掉电计时器，如有需要请在图形化配置界面里提前设置好。
3. 如果希望单片机只能用某一个中断唤醒，那么在进入掉电模式前，请把其他的中断使能关掉，只留下那个中断。

4. 掉电状态下被唤醒了之后，CPU将会从掉电指令后继续执行，不会重启到main函数。
5. **为了优化空间，本函数默认不使能。**请在power.h文件的图形化配置界面里使能【单片机电源控制函数】。



power_cpu_idle

函数原型：void power_cpu_idle(void);

描述

CPU空闲函数，单片机进入空闲模式，CPU的电源都会被关掉，但是外设的电源依然存在。

输入

无

输出

无

返回值

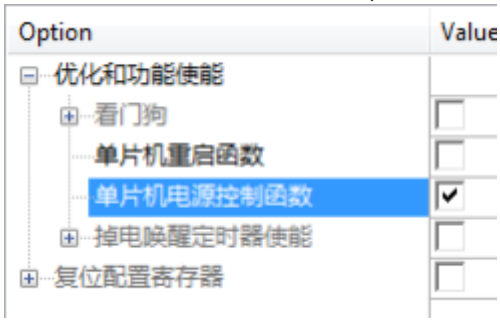
无

调用例程

```
while(1){//主循环中，
    power_cpu_idle();//进入空闲模式，等待某种条件唤醒。
    do_something();//唤醒后执行某些动作。
}
```

注意事项

1. 空闲模式由于外设的电源没有关，所以外设依然会工作，单片机整体的功耗基本不会下降太多。
2. 似乎这个功能太鸡肋，因为掉电模式也可以从掉地语句后面开始执行代码。所以在一些最新的STC型号里，取消了空闲模式。因此我推荐用掉电模式就可以了，空闲模式最好不用。
3. **为了优化空间，本函数默认不使能。**请在power.h文件的图形化配置界面里使能【单片机电源控制函数】。



wdt_start

函数原型：void wdt_start(void);

描述

看门狗开启函数，看门狗打开后直到重启前都不能关掉。

输入

无

输出

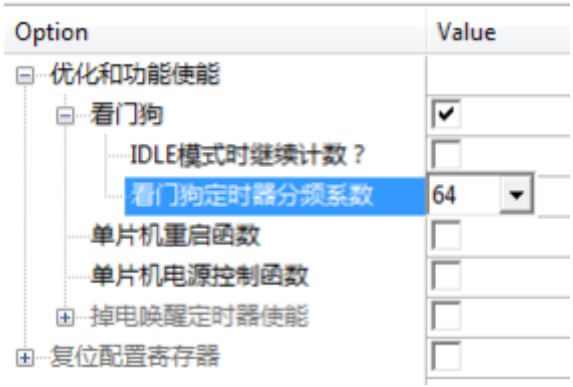
无

返回值

无

参数配置

看门狗就是一个不断在跑的定时器，如果看门狗定时器溢出，看门狗就会复位单片机。而这个时间在power.h文件的图形化配置界面里可以看到：



看门狗的定时时间和分频系数有关，公式和大致的预估时间如下图所示：

WDT_PS[2:0]：看门狗定时器时钟分频系数

WDT_PS[2:0]	分频系数	12M 主频时的溢出时间	20M 主频时的溢出时间
000	2	≈ 65.5 毫秒	≈ 39.3 毫秒
001	4	≈ 131 毫秒	≈ 78.6 毫秒
010	8	≈ 262 毫秒	≈ 157 毫秒
011	16	≈ 524 毫秒	≈ 315 毫秒
100	32	≈ 1.05 秒	≈ 629 毫秒
101	64	≈ 2.10 秒	≈ 1.26 秒
110	128	≈ 4.20 秒	≈ 2.52 秒
111	256	≈ 8.39 秒	≈ 5.03 秒

看门狗溢出时间计算公式如下：

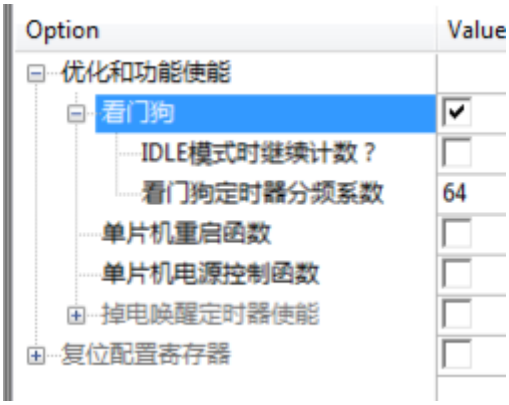
$$\text{看门狗溢出时间} = \frac{12 \times 32768 \times 2^{(WDT\_PS+1)}}{SYSclk}$$

调用例程

```
wdt_start();//打开看门狗。
```

注意事项

1. 看门狗一但开启就不能关闭，除非单片机重启或断电。
2. 使用看门狗之前在图形化配置界面勾选【看门狗】的使能。



3. 如果在空闲模式下使用看门狗，要勾选【IDLE模式时继续计数】。但勾选之后必须在看门狗溢出前喂狗，否则单片机就会重启。

wdt_feed

函数原型：void wdt_feed(void);

描述

看门狗喂狗函数，清零看门狗定时器。

输入

无

输出

无

返回值

无

调用例程

```
wdt_start();//先打开看门狗。
while(1){//主循环里
    do_something();//做某事。
    wdt_feed();//做完之后喂狗。
}
```

注意事项

1. 先执行wdt_start函数打开看门狗，然后喂狗函数才有意义。
2. 喂狗函数的执行位置不限，执行时间也不限但要在看门狗溢出之前至少执行一次。
3. 如果某个函数的执行时间大于看门狗的溢出时间，那么最好在该函数内部添加多句喂狗函数。**严禁为了偷懒而在定时器中断里喂狗**，因为有时候在执行一些函数的时候会陷入死循环中，但此时不影

响中断的跳转。如果在中断喂狗，意味这个异常不会退出。

power_rstcfg_init

函数原型：void power_rstcfg_init(void);

描述

复位寄存器初始化函数。

输入

无

输出

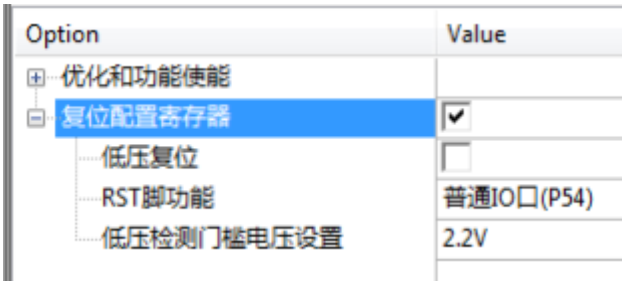
无

返回值

无

参数配置

在使用这个函数之前先使能【复位配置寄存器】：



如果需要低压复位，那么就勾选【低压复位】使能。低压复位的功能就是当单片机的VCC低于某个电压的时候，复位单片机。在下面的【电压检测门槛电压设置】那里设置触发低压复位的电压阈值。**需要注意的是，检测到低压之后，不一定非得复位。**设置好阈值电压之后，不勾选【低压复位】使能，单片机就会触发低电压中断。可以利用该中断，做一些断电保存参数的事情。

RST脚在STC8里是可以复用成IO口的，对应着P5.4。

在这里可以设置成IO口或者RST脚，由于本函数是main函数里执行，那么STC-ISP上的配置就会在执行本函数的时候失效。比如在STC-ISP上设置RST/P5.4脚为复位脚，而在ECBM库中设置为P5.4脚。那么在单片机BootLoader阶段，该脚是复位脚，在执行main之后，该脚就变成了P5.4脚。

调用例程

无，该函数会在system_init函数中自动调用。

注意事项

1. 本函数所执行的操作和设置，都可以在STC-ISP上设置。所以从优化角度来看可以不用这部分。
2. 之所以写成库，是为了一种情况：开源固件的时候，有些设置比较重要（比如P5.4脚一定得是RST脚），而使用开源固件的人不一定会去STC-ISP上设置这个选项，那么这个时候该函数就能发挥作用了。

优化建议

基本每个系列的函数（比如重启系列有两个，电源控制有两个）都有使能选项，不用到的函数不使能即可。