SPC2188 睡眠及唤醒使用指南

版本 A/0 – 2023 年 9 月

概述

SPC2188 的电源分为三部分（DVDD(3,3V),LDO(1.2V),AVDD(3.3V)），它支持以下三种电源模式：

- Active（正常使用模式）

  DVDD，AVDD 和 LDO 都打开，所有时钟根据用户配置开或关。

- Stop（停机模式）

  除了处理唤醒请求的低频 128KHz 时钟外，其余所有时钟关闭。寄存器和 RAM 均维持其内容，可以超时自动唤醒和 I/O 引脚唤醒（可以配置为任意 GPIO 引脚上的任意电平），唤醒后软件从进入停机模式的位置继续执行。

- Sleep（睡眠模式）

  关闭片内电源、时钟和所有外设，所有引脚处于输入高阻状态，可以配置 GPIO10 或者 GPIO11 的任意电平唤醒，唤醒后开始冷启动过程。

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-09-01	X.He	Released	首次发布。

术语或缩写

术语或缩写	描述

1 睡眠模式

此种模式是通过软件发送睡眠指令，主动迫使电源管理模块进入睡眠模式。通过 WAKEUP_GPIO 可以选择唤醒源是 PIN_GPIO10 或者 PIN_GPIO11，WAKEUP_LEVEL 选择低电平唤醒还是高电平唤醒，如下代码演示了如何使用代码发送睡眠命令主动进入睡眠模式。

Example Code

```
#include "spc2188.h"

#include <stdio.h>

ErrorStatus          status;


/*
 *     WAKEUP_GPIO:
 *                 1. PIN_GPIO10
 *                 2. PIN_GPIO11
 *
 *     WAKEUP_LEVEL:
 *                 1. HIGH
 *                 2. LOW
 */
#define              WAKEUP_GPIO     PIN_GPIO11
#define              WAKEUP_LEVEL    HIGH



ErrorStatus SYSTEM_Sleep_Cmd(void)
{
    uint32_t i;
    if (SYSTEM->GPIOWKUPCTL >> 17)  /* Wakeup by GPIO11 */
    {
        PIN_EnableInputChannel(PIN_GPIO11, PIN_CH0);
        __ASM("NOP");
        __ASM("NOP");
        i = (GPIO->GPLR[0] >> 11) & 1U;
    }
    else                            /* Wakeup by GPIO10 */
    {
        PIN_EnableInputChannel(PIN_GPIO10, PIN_CH0);
        __ASM("NOP");
        __ASM("NOP");
        i = (GPIO->GPLR[0] >> 10) & 1U;
    }

    if ((SYSTEM->GPIOWKUPCTL >> 16) & 1U)   /* Wakeup if GPIO = 1 */
    {
        i = 1 - i;
    }
    if (i == 1)
    {
        POWER->PWRREGKEY    = 0xacce55cdU;
        POWER->PWRMODECMD   = 0xa5a51ee9U;
    }

    return SUCCESS;
}
```

```
int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_MAX);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Just a Sample....\n");


    status = SYSTEM_SetSleepWakeUpByGPIO(WAKEUP_GPIO, WAKEUP_LEVEL);
    if (status != SUCCESS)
    {
        printf("Set sleep wakeup by gpio fail\n");
    }

    SYSTEM_Sleep_Cmd();

    printf("total test passed\n");

    while (1)
    {
    }
}
```

2 停机模式

如文档概述部分所述，进入停止模式之后，电源依然会对芯片的供电，所以 RAM 的数据不会丢失。但需要注意的是，此时 MCU 没有任何时钟，外设的一切功能将停止（例如：PWM 将不能产生波形等），且此时的 MCU 也不能捕捉 GPIO 等外设的瞬时信号（例如，在停止模式时，某个 GPIO 有上升沿产生，此时 MCU 并不知道有这个事件发生，也就不能在唤醒之后进入对应的 GPIO 边沿中断）。

此模式通过软件发送停机指令，主动迫使芯片进入停机模式，可以通过 Wake_Up_Mode 宏来选择超时唤醒，还是 GPIO 引脚唤醒，若选择 GPIO 引脚唤醒可以配置 WAKEUP_GPIO 和 WAKEUP_LEVEL 来选择唤醒引脚和唤醒电平，如下代码演示了如何使用代码发送停机命令进入停机模式。

Example Code

```
#include "spc2188.h"

#include <stdio.h>

ErrorStatus          status;


/*
 *     Wake_Up_Mode = 0; Cycle wakeup
 *     Wake_Up_Mode = 1; Gpio  wakeup
 *
 *     WAKEUP_LEVEL:
 *                 1. HIGH
 *                 2. LOW
 */
#define              WAKEUP_GPIO     PIN_GPIO11
#define              WAKEUP_LEVEL    HIGH
#define              Wake_Up_Mode    1


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_MAX);
    Delay_Init();

    PIN_SetChannel(PIN_GPIO62, PIN_GPIO62_UART0_TXD);
    PIN_SetChannel(PIN_GPIO63, PIN_GPIO63_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Just a Sample....\n");

    /* Wait for debugging BUFF to empty, otherwise it will print fail */
    Delay_Us(200);

    if (Wake_Up_Mode == 1)
    {
        /* High Level wakeup */
        status = SYSTEM_SetStopWakeUpByGPIO(WAKEUP_GPIO, WAKEUP_LEVEL,
DEGLITCH_WINDOW_2MS);
        if (status != SUCCESS)
        {
            printf("Set stop wakeup by gpio fail\n");
        }

        pHWLIB->SYSTEM_Stop(0);
    }
    else
```

```
    {
        /* 1000ms automatically wakeup */
        pHWLIB->SYSTEM_Stop(1000 * 128);
    }

    printf("total test passed\n");

    while (1)
    {
    }
}
```