睡眠及唤醒使用指南

概述

本手册适用范围：

适用范围
SPC1169，SPD1179，SPD1176

文中有关 Active，Stop，Sleep 的描述以及状态切换因 SPD1179/SPD1176 与 SPC1169 设计不同而具有本质差别，详见正文。

目录

图片列表

修订历史

版本	日期	作者	状态	变更
A/0	2023-11-30	Hang Su	已过期	1. 首次发布。
C/0	2025-06-04	Hang Su	已发布	1. 更新章节 1.3.2 示例。

术语或缩写

术语或缩写	描述
LIN	Local Interconnect network，低成本串行通信网络
MON	指 SPD1179 中的 MON 管脚

1 SPD1179/SPD1176 电源管理模块（PMU）

电源管理模块为 MCU 部分（DVDD5，DVDD33，VCAP12）和 VDD5EXT 供电，通过状态机控制电源模式转换，并确保 SPD1179/SPD1176 的安全运行。该模块支持三种工作模式：

Active（正常工作模式）

DVDD5，DVDD33 和 VCAP12 由片内 LDO 以满载能力供电，所有时钟根据用户配置开或关。

Stop（停止模式）

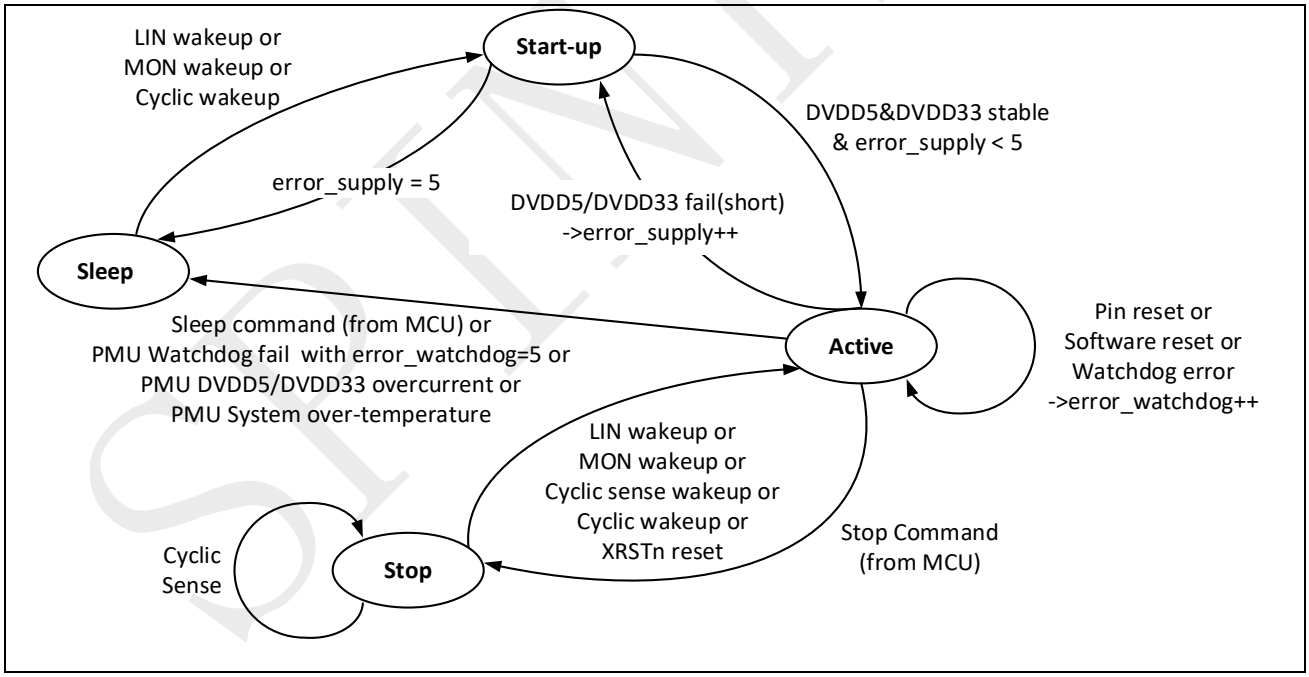
除了用于电源模式状态机的 100 kHz 时钟外，其余所有时钟关闭。DVDD5、DVDD33 和 VCAP12 由低功耗、低负载能力的 LDO 供电，VCAP12 可根据用户配置降至 1.2V 以下，以进一步降低静态电流。所有寄存器和存储器保持内容，系统可由不同唤醒源唤醒，例如：XRSTn、LIN、MON、cyclic-wakeup 和 cyclic-sense-wakeup（可配置为任意 GPIO 引脚上的任意电平）。CPU 将直接继续执行停止时的下一条指令（XRSTn 唤醒将导致系统复位）。

Sleep（睡眠模式）

DVDD5，DVDD33 和 VCAP12 完全断电，系统可以通过 LIN/MON/cyclic-wakeup 唤醒，并开始冷启动过程。

三种电源模式转换如图 1-1 所示。

图 1-1：电源模式转换



1.1 睡眠模式

从图 1-1 可以看出，导致进入睡眠模式的事件有：睡眠命令、**PMU WDT 错误、PMU DVDD5/DVDD33 过流、PMU 系统过温**，其中 **PMU WDT 错误、PMU DVDD5/DVDD33 过流、PMU 系统过温发生时，硬件将会自动迫使电源管理模块进入睡眠模式**，不需要软件做任何设置。

注意： 当唤醒条件有效时，即使发生如上描述的四种睡眠事件，电源管理模块也不会进入睡眠模式。

1.1.1 睡眠命令

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，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 "spd1179.h"
#include <stdio.h>

ErrorStatus              eErrorState;

int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /* Enable sleep/stop command */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_PMU_CMD);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    pHWLIB->SYSTEM_Sleep();

    while (1)
    {

    }
}
```

1.1.2 系统过温

SPD1179 内置有一个温度传感器，当芯片温度超过某一温度（记为 OT_fatal）时，电源管理模块将会进入睡眠模式，这一动作由硬件自动完成，软件唯一需要做的是设置一个发出该动作的温度阈值。温度阈值 OT_fatal 有 7 个档位可以选择（详细请查阅 SPD1179 技术参考手册中 EVTTHCTL0 寄存器），且通过如下所示的代码进行设置。

Example Code

```
#include "spd1179.h"
#include <stdio.h>

ErrorStatus           eErrorState;            /* Function State */

int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* HV init */
    eErrorState = HV_Init(&u16PREDRIID);
    if (eErrorState == ERROR)
    {
        printf("Init HV mode FAIL\n");
        return 0;
    }else{
        printf("Init HV mode SUCCESS[ID:%d]\n", u16PREDRIID);
    }

    /* HV parameter write enable */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_USER_REG);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    /* Set the PMU over-temperature threshold */
    eErrorState = EPWR_WriteRegisterField(HV_REG_EVTTHCTL0,
EVTTHCTL0_PMUOTWARN_Msk, EVTTHCTL0_PMUOTWARN_129C);
    if (eErrorState == ERROR)
    {
        printf("EPWR_WriteRegisterField FAIL\n");
        return 0;
    }
    while (1)
    {
    }
}
```

1.2 停止模式

如前所述，进入停止模式之后，DVDD5，DVDD33 和 VCAP12 由低功耗低载能力的 LDO 供电，此时对芯片的供电维持正常，RAM 的数据不会丢失。但需要注意的是，此时 MCU 没有时钟，外设功能将停止（例如：PWM 将不能产生波形），且 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，但 MCU 无法在停止模式下捕捉该事件，无法在被唤醒后进入对应的 GPIO 边沿中断。

从图 1-1 中可以看出，通过软件方式向高压部分发送停止指令是进入停止模式的唯一途径，如下代码演示了如何使用代码发送停止命令进入停止模式。

Example Code

```
#include "spd1179.h"
#include <stdio.h>

ErrorStatus                eErrorState;

int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    /* Enable sleep/stop command */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_PMU_CMD);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    pHWLIB->SYSTEM_Stop();

    while (1)
    {

    }
}
```

1.3 唤醒

如概述部分图 1-1 的描述，在睡眠模式和停止模式下，两者可供选择的唤醒源不同。

睡眠模式：可以通过 LIN/MON/Cyclic-WakeUp 唤醒，并开始冷启动过程，冷启动意味着将重新开始执行代码。

停止模式：电源管理模块可以被不同的唤醒源唤醒，比如，XRSTn、LIN、MON、cyclic-wakeup 和 cyclic-sense-wakeup（可以配置为任意 GPIO 引脚上的任意电平），其中，用 LIN、MON、cyclic-wakeup 和 cyclic-sense-wakeup 方式唤醒之后，将继续执行停止时的下一条指令，而用 XRSTn 唤醒之后，Core 将重启，这将导致代码重新开始执行。

1.3.1 睡眠模式唤醒

本小节将介绍如何使用三种唤醒睡眠模式的代码，代码中已使用宏区分开三种不同的唤醒方式，用户可按需选择。

LIN 唤醒

若用户选择使用 LIN 作为唤醒源，将 LIN 管脚接地，并维持低电平大于 80us，将会唤醒电源管理模块，在验证示例代码时，若测试成功，将会看到“Init PRE-DRIVER mode SUCCESS”的串口打印字样。

注意： LIN 协议要求从机要能识别>150us 的唤醒信号，按照 80us 规格设计的 SPD1179/SPD1176 完全满足 LIN 协议的要求。

MON 唤醒

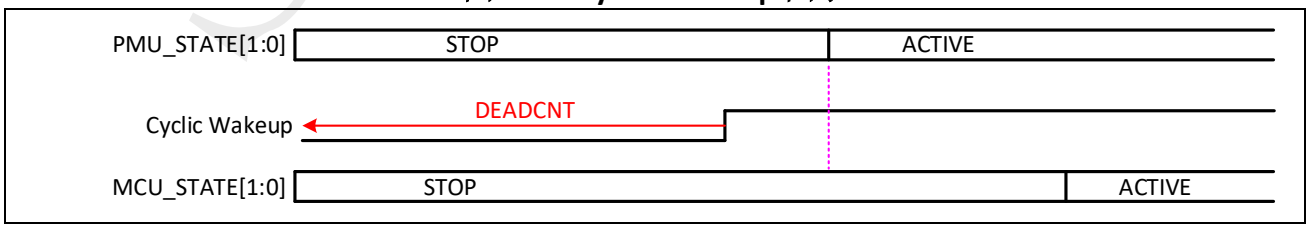
若用户选择使用 MON 作为唤醒源，由于在示例代码中已经将 MON 唤醒极性设置为高，所以在验证示例代码时，需要将 MON 管脚接到高电平，并维持超过 40us，将会唤醒电源管理模块，若测试成功，将会看到“Init PRE-DRIVER mode SUCCESS”的串口打印字样。

Cyclic-WakeUp 唤醒

若用户使用 Cyclic-WakeUp 方式唤醒，在示例代码中已设置 DEADCNT 为 12ms，这意味着，电源管理模块将以 12ms 为周期，循环发出唤醒信号，使自己从停止模式转入 Active 模式。在验证示例代码时，若测试成功，将会看到“Init PRE-DRIVER mode SUCCESS”的串口打印字样。

Cyclic-WakeUp 图示如图 1-2 所示。

图 1-2：Cyclic-WakeUp 图示



注意： 睡眠模式唤醒后的启动流程和冷启动一致，所以测试时需要注意保持 BOOT 脚为低电平，此时才会执行正常的启动流程，否则，芯片将一直卡死在 ROM 代码段。

Example Code

```
#include "spd1179.h"
#include <stdio.h>


uint32_t                u32PWMPeriod;           /* PWM Period*/
uint16_t                u16PREDRIID;           /* PRE-DRIVER mode ID */
ErrorStatus             eErrorState;
#define                   Wake_Up_Mode                          3


int main(void)
{
    CLOCK_InitWithRCO(CLOCK_CPU_100MHZ);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* HV init */
    eErrorState = HV_Init(&u16PREDRIID);
    if (eErrorState == ERROR)
    {
        printf("Init PRE-DRIVER mode FAIL\n");
        return 0;
    }
    else
    {
        printf("Init PRE-DRIVER mode SUCCESS[ID:%d]\n", u16PREDRIID);
    }

    /* HV parameter write enable */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_USER_REG);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    #if(Wake_Up_Mode == 0) //LIN wake up setting
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_LINWKUPEN_Msk |
PMUCTL_CYCWKUPEN_Msk, PMUCTL_LINWKUPEN_ENABLE | PMUCTL_CYCWKUPEN_DISABLE);
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }
    #elif(Wake_Up_Mode == 1) //MON asynchronous wake up setting
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_MONWKUPEN_Msk,
PMUCTL_MONWKUPEN_ENABLE);
    if (eErrorState == ERROR)
```

Example Code

```
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }
    eErrorState = EPWR_WriteRegisterField(HV_REG_MONCTL, MONCTL_WKUPPOL_Msk | MONCTL_EN_Msk,
MONCTL_WKUPPOL_ACTIVE_HIGH | MONCTL_EN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("Write MONCTL register FAIL\n");
        return 0;
    }

    #elif(Wake_Up_Mode == 2) //Cyclic wake up setting
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_CYCWKUPEN_Msk,
PMUCTL_CYCWKUPEN_ENABLE);
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    /* Dead time is (DEADCNT+1) * 2ms */
    eErrorState = EPWR_WriteRegisterField(HV_REG_CYCWKUPCTL, CYCWKUPCTL_DEADCNT_Msk,
CYCWKUPCTL_DEADCNT_(5));
    if (eErrorState == ERROR)
    {
        printf("Write CYCWKUPCTL register FAIL\n");
        return 0;
    }
    #endif

    /* Enable sleep/stop command */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_PMU_CMD);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    pHWLIB->SYSTEM_Sleep();

    while (1)
    {

    }
}
```

1.3.2 停止模式唤醒

停止模式下可选的唤醒源有五种，其中 XRSTn 属于硬件直接唤醒，不需要软件进行设置，不再赘述。在以下示例代码中，已使用宏区分开四种不同的唤醒方式，用户可按需选择。

LIN 唤醒

若用户选择使用 LIN 作为唤醒源，将 LIN 管脚接地，并维持低电平大于 80us，将会唤醒电源管理模块，在验证示例代码时，若测试成功，将会看到“total test passed”的串口打印字样。

注意：
1. LIN 协议要求从机要能识别>150us 的唤醒信号，按照 80us 规格设计的 SPD1179/SPD1176 完全满足 LIN 协议要求；
2. 默认状态下 CYCWKUPEN 和 LINWKUPEN 作为唤醒源处于使能状态，在测试 LIN 唤醒功能时，请关闭 CYCWKUPEN 功能，避免出现不必要的现象。

MON 唤醒

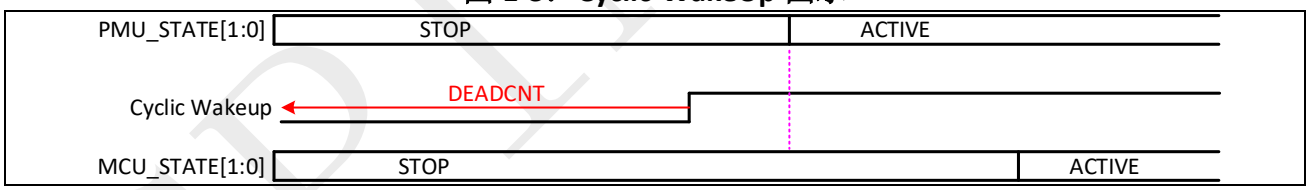
若用户选择使用 MON 作为唤醒源，由于在示例代码中已经将 MON 唤醒极性设置为高，所以在验证示例代码时，需要将 MON 管脚接到高电平，并维持超过 40us，将会唤醒电源管理模块，若测试成功，将会看到“total test passed”的串口打印字样。

Cyclic-WakeUp 唤醒

若用户使用 Cyclic-WakeUp 方式唤醒，在示例代码中已设置 DEADCNT 为 12ms，这意味着，电源管理模块将以 12ms 为周期，循环发出唤醒信号，使自己从停止模式转入 Active 模式。在验证示例代码时，若测试成功，将会看到“total test passed”的串口打印字样。

Cyclic-WakeUp 图示如图 1-3 所示。

图 1-3：Cyclic-WakeUp 图示

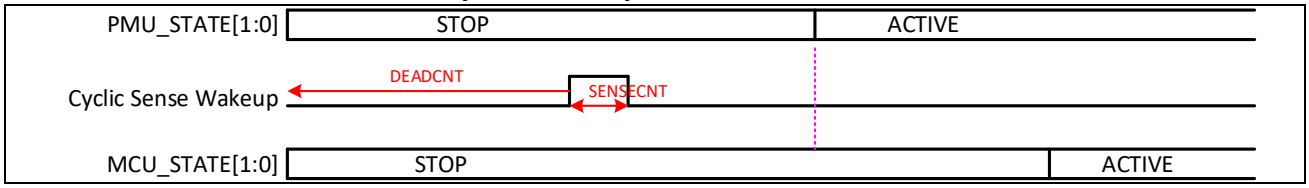


Cyclic-Sense-WakeUp 唤醒

Cyclic-Sense-WakeUp 唤醒在实际的使用场景中，有三个参数可以设置，分别是：DEADCNT，SENSECNT，RAMPCNT。以下用两个场景对三个参数的功能加以说明。

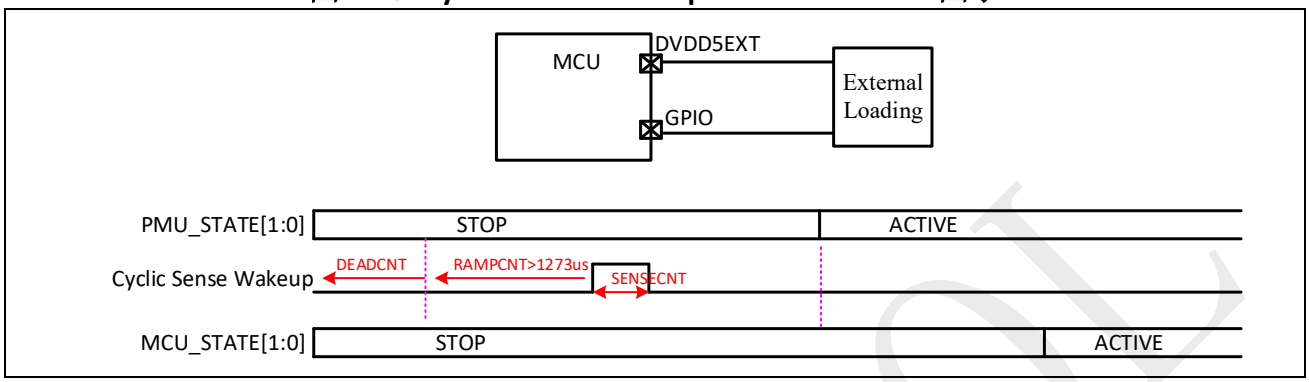
场景一：电源管理模块以 DEADCNT 为周期，周期性开启以 SENSECNT 指定窗口时间的唤醒侦测窗口，在此窗口期内，检查在指定的 GPIO 上是否存在唤醒电平，若存在唤醒电平，则进入 Active 模式（若唤醒电平不在侦测窗口期内，则无法唤醒电源管理模块）；若不存在，则继续保持停止模式。此时 Cyclic-Sense-WakeUp 的图示如图 1-4 所示。

图 1-4：Cyclic-WakeUp Without DVDD5EXT 图示



场景二：外部负载需要通过本产品的 SVDD5EXT 供电，与此同时，GPIO 唤醒电平需要通过外部负载给出，在此应用场景中 Cyclic-Sense-WakeUp 的图示如图 1-5 所示。与场景一不同的是，在开启侦测窗口之前，需要设定 DVDD5EXT 的启动时间，此时间由 RAMPCNT 指定。需要注意的是，如图 1-5 所示，RAMPCNT 的时间最小需要设置为 1273us。

图 1-5：Cyclic-Sense-Wakeup With DVDD5EXT 图示



Example Code

```
#include "spd1179.h"
#include <stdio.h>


uint32_t            u32PWMPeriod;       /* PWM Period*/
uint16_t            u16PREDRIID;        /* PRE-DRIVER mode ID */
ErrorStatus          eErrorState;
#define             Wake_Up_Mode        0

int main(void)
{
    int32_t * pi32TrimValue;
    int32_t i32Temperature;

    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* HV reset */
    eErrorState = HV_Reset();
    if (eErrorState == ERROR)
    {
        printf("HV_Reset FAIL\n");
        return 0;
    }
    else
    {
        printf("HV_Reset SUCCESS\n");
    }

    /* HV init */
```

Example Code

```
    eErrorState = HV_Init(&u16PREDRIID);
    if (eErrorState == ERROR)
    {
        printf("Init PRE-DRIVER mode FAIL\n");
        return 0;
    }
    else
    {
        printf("Init PRE-DRIVER mode SUCCESS[ID:%d]\n", u16PREDRIID);
    }

    /* HV parameter write enable */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_USER_REG);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    /* LIN wake up setting */
    #if(Wake_Up_Mode == 0)
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_LINWKUPEN_Msk |
      PMUCTL_CYCWKUPEN_Msk, PMUCTL_LINWKUPEN_ENABLE | PMUCTL_CYCWKUPEN_DISABLE);
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    /* MON asynchronous wake up setting */
    #elif(Wake_Up_Mode == 1)
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_MONWKUPEN_Msk |
                                PMUCTL_LINWKUPEN_Msk | PMUCTL_CYCWKUPEN_Msk,
                                PMUCTL_MONWKUPEN_ENABLE |
PMUCTL_LINWKUPEN_DISABLE |
                                PMUCTL_CYCWKUPEN_DISABLE);
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    /* Wakeup level is high, MON pin need pull down */
    eErrorState = EPWR_WriteRegisterField(HV_REG_MONCTL, MONCTL_WKUPPOL_Msk |
                            MONCTL_EN_Msk | MONCTL_PULLMODE_Msk,
                            MONCTL_WKUPPOL_ACTIVE_HIGH | MONCTL_EN_ENABLE |
                            MONCTL_PULLMODE_PULL_DOWN);
    if (eErrorState == ERROR)
    {
        printf("Write MONCTL register FAIL\n");
        return 0;
    }

    /* Cyclic wake up setting */
    #elif(Wake_Up_Mode == 2)
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL, PMUCTL_CYCWKUPEN_Msk |
                                    PMUCTL_LINWKUPEN_Msk,
PMUCTL_CYCWKUPEN_ENABLE |
                                    PMUCTL_LINWKUPEN_DISABLE);
    if (eErrorState == ERROR)
    {
```

Example Code

```
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    /* Dead time is (DEADCNT+1) * 2ms */
    eErrorState = EPWR_WriteRegisterField(HV_REG_CYCWKUPCTL,
CYCWKUPCTL_DEADCNT_Msk, CYCWKUPCTL_DEADCNT_(5));
    if (eErrorState == ERROR)
    {
        printf("Write CYCWKUPCTL register FAIL\n");
        return 0;
    }

    /* Cyclic sense wake up setting */
    #elif(Wake_Up_Mode == 3)
    /* Use GPIO9 as the wake up source */
    /* Config pin as GPIO input */
    PIN_SetPinAsGPIO((PIN_NameEnum)PIN_GPIO9);
    GPIO_SetPinDir((PIN_NameEnum)PIN_GPIO9, GPIO_INPUT);

    /* Set GPIO number and active level */
    SYSTEM->GPIOWKUPCTL = (LOW << GPIOWKUPCTL_STOPWKUPPOL_Pos) |
                          (PIN_GPIO9  << GPIOWKUPCTL_STOPWKUPIO_Pos) |
                          GPIOWKUPCTL_STOPWKUPWE_ENABLE;

    printf("LIN or MON must be enabled, otherwise cyclic wake up can not be
disabled\n");
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL,
                                          PMUCTL_LINWKUPEN_Msk |
PMUCTL_MONWKUPEN_Msk ,
                                          PMUCTL_LINWKUPEN_DISABLE |
PMUCTL_MONWKUPEN_ENABLE );
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    printf("Enable cyclic sense wake up and disable cyclic wake up\n");
    eErrorState = EPWR_WriteRegisterField(HV_REG_PMUCTL,
                                          PMUCTL_CYCSENWKUPEN_Msk |
PMUCTL_CYCWKUPEN_Msk,
                                          PMUCTL_CYCSENWKUPEN_ENABLE |
PMUCTL_CYCWKUPEN_DISABLE);
    if (eErrorState == ERROR)
    {
        printf("Write PMUCTL register FAIL\n");
        return 0;
    }

    /* Dead time is (DEADCNT+1) * 2ms, sense window is (SENSECNT+1) * 10us */
    eErrorState = EPWR_WriteRegisterField(HV_REG_CYCSENSECTL,
CYCSENSECTL_DEADCNT_Msk | \
          CYCSENSECTL_SENSECNT_Msk, CYCSENSECTL_DEADCNT_(10) |
CYCSENSECTL_SENSECNT_(15));
    if (eErrorState == ERROR)
    {
        printf("Write CYCSENSECTL register FAIL\n");
        return 0;
    }
    #endif
```

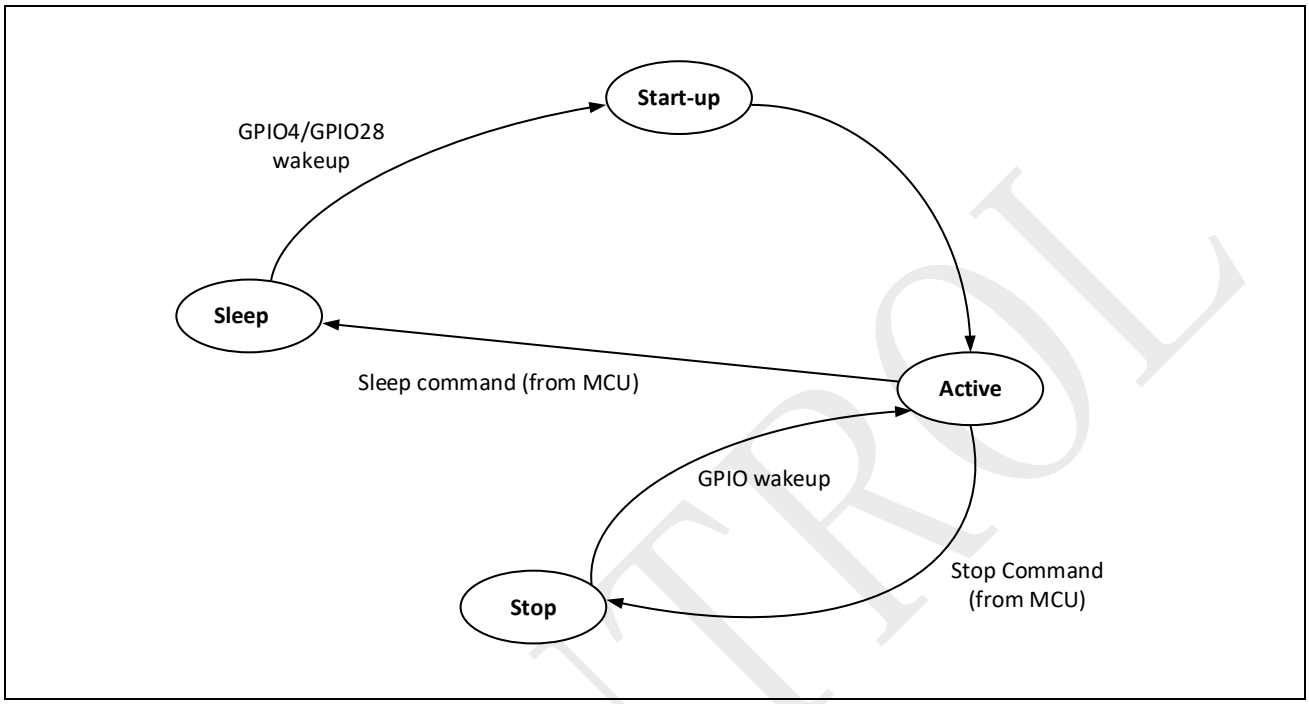
Example Code

```
    /* Enable sleep/stop command */
    eErrorState = EPWR_WriteRegister(HV_REG_CTLKEY, KEY_PMU_CMD);
    if (eErrorState == ERROR)
    {
        printf("Write CTLKEY register FAIL\n");
        return 0;
    }

    /* Enable monitor MCU */
    SYSTEM_EnableMonitorItem(MONITOR_MCU_TEMPERATURE);
    ADC_EnableTSensor(ADC);
    /* Wait for sampling to end */
    Delay_Ms(500);

    /* TPMUOFFSET
     * TPMUGAIN
     * TLINOFFSET
     * TLINGAIN
     * TMCUOFFSET
     * TMCUGAIN
     */
    pi32TrimValue = (int32_t *)0x11001168;

    /* The if branch is equivalent to the else branch, and the if branch is
more accurate */
    if (pi32TrimValue[4] != 0xffffffff)
    {
        i32Temperature = ((pi32TrimValue[4] + pi32TrimValue[5] * SYSTEM-
>TMCUCODE )/65536);
    }
    else
    {
        i32Temperature = ((4096 * SYSTEM->TMCUCODE - 305657)/1081);
    }

    /* If MCU over temperature 105 degrees Celsius, can not enter SYSTEM_Stop()
*/
    if (i32Temperature > 105)
    {
        printf("the MCU temperature is %d\n", i32Temperature);
        return 0;
    }

    pHWLIB->SYSTEM_Stop();

    printf("total test passed\n");

    while (1)
    {

    }
}
```

2 SPC1169 电源管理模块（PMU）

电源管理单元（PMU）负责为嵌入式 MCU（VCAP12）供电。PMU 具有明确定义的状态机，用于控制电源模式的转换，如图 2-1 所示。

图 2-1：电源模式转换



SPC1169 的启动过程和 3 种电源模式如下：

Active 模式：

稳压器（VCAP12）由芯片内部具有全电流驱动能力的 LDO 供电；所有时钟根据用户在寄存器控制位中的设置启用。

Stop 模式：

除了用于电源模式状态机的 128 kHz 时钟外，所有时钟都停止，VCAP12 由具有有限电流驱动能力的低功耗 LDO 供电，VCAP12 可配置为低于 1.2V，以进一步降低静态电流，所有寄存器和内存保持内容不变。系统可以通过不同的 GPIO 进行唤醒，当退出停止模式时，CPU 将继续执行下一条指令。

Sleep 模式：

稳压器（VCAP12）完全关闭，可以通过 GPIO4/GPIO28 被唤醒，整个系统将进行冷启动。

注意：
1. 在发送停止命令之前，用户应安全禁用未使用的模拟模块。
2. 在停止模式下，泄漏电流在高温下显著增加，可能超过低功耗 LDO 的负载能力。在发出停止命令之前，用户需要确保结温低于 100°C。

2.1 睡眠模式

从图 2-1 可以看出，导致进入睡眠模式的事件只有睡眠命令。

需要特别强调的是，当唤醒条件有效时（见章节 2.3），即使运行睡眠指令，电源管理模块也不会进入睡眠模式。

以下代码演示了使用代码发送睡眠命令进入睡眠模式。

Example Code

```
pHWLIB->MCU_Sleep();
```

2.2 停止模式

进入停止模式之后，VCAP12 由低功耗低载能力的 LDO 供电，此时对芯片的供电维持正常，RAM 的数据不会丢失。但需要注意的是，此时 MCU 没有时钟，外设功能将停止（例如：PWM 将不能产生波形），且 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，但 MCU 无法在停止模式下捕捉该事件，无法在被唤醒后进入对应的 GPIO 边沿中断。

从图 2-1 中可以看出，通过软件方式向高压部分发送停止指令是进入停止模式的唯一途径，以下代码演示了使用代码发送停止命令进入停止模式。

Example Code

```
pHWLIB->MCU_Stop();
```

2.3 唤醒

如图 2-1 描述，在睡眠模式和停止模式下，两者可供选择的唤醒源不同。

- 睡眠模式

  通过 GPIO4/GPIO28 唤醒，并开始冷启动过程，冷启动意味着将重新开始执行代码。

- 停止模式

  电源管理模块可以被不同的 GPIO 唤醒，唤醒之后，将继续执行停止时的下一条指令。

2.3.1 睡眠模式唤醒

示例代码演示了如何通过 GPIO4 低电平唤醒，并开始冷启动过程。

注意： 睡眠模式唤醒后的启动流程和冷启动一致，所以测试时需注意保持 BOOT 脚为低电平，此时才会执行正常的冷启动流程，否则，芯片将一直卡死在 ROM 代码段。

Example Code

```
#include <stdio.h>

#if defined(SPD1179)
    #include "spd1179.h"
#else
    #include "spc1169.h"
#endif


ErrorStatus                eErrorState;



int main(void)
{
    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* Set the wakeup pin and level */
    eErrorState = SYSTEM_SetSleepWakeUpByGPIO(PIN_GPIO4, LOW);
    if (eErrorState == SUCCESS)
    {
        printf("SetSleepWakeUp passed\n");
    }
    else
    {
        printf("SetSleepWakeUp failed\n");
    }

    /* MCU sleep */
    pHWLIB->MCU_Sleep();

    while (1)
    {

    }
}
```

2.3.2 停止模式唤醒

示例演示了如何通过 GPIO0 低电平唤醒，唤醒后将继续执行停止时的下一条指令。

Example Code

```
#include <stdio.h>

#if defined(SPD1179)
    #include "spd1179.h"
#else
    #include "spc1169.h"
#endif


ErrorStatus                 error_flag;


/*******************************************************************************
***
 *
 * @brief      In this case, we use the PIN_GPIO0 to wakeup the MCU from STOP
state.
 *

*******************************************************************************
*/


int main(void)
{
    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

    printf("Enter the test\n");

    /* Enable the PIN_GPIO0 LOW to wakeup the MCU */
    error_flag = SYSTEM_SetStopWakeUpByGPIO(PIN_GPIO0, LOW);
    if (error_flag == SUCCESS)
    {
        printf("SetStopWakeUp passed\n");
    }
    else
    {
        printf("SetStopWakeUp failed\n");
    }

    /* Wait until uart printf finished */
    Delay_Ms(10);

    while (1)
    {
        pHWLIB->MCU_Stop();

        /* the printf will be execute if we connect the PIN_GPIO0 low
```

```
        to wakeup the MCU from STOP state */
        printf("total test passed\n");

        /* Wait until uart printf finished */
        Delay_Ms(10);
    }
}
```