GigaDevice Semiconductor Inc.

GD32L233 从深度睡眠模式 1 唤醒的多种方法

应用笔记

AN094

1.0 版本

（2023 年 5 月）

目录

表索引

1. 前言

在嵌入式系统应用开发中，常会遇到低功耗应用场景。GD32L233微控制器提供了10种省电模式，分别为运行模式，运行模式1，运行模式2，睡眠模式，睡眠模式1，睡眠模式2，深度睡眠模式，深度睡眠模式1，深度睡眠模式2和待机模式。在GD32L233低功耗系统开发中常用到深度睡眠模式1。系统进入到深度睡眠模式1后，将关闭1.1V域时钟、IRC16M、IRC48M、HXTAL及PLLs时钟，同时NPLDO关闭，LPLDO开启，SARAM和寄存器的内容被保留。任何来自EXTI线的中断或事件都可以将系统从深度睡眠模式唤醒。针对特定的低功耗系统，我们需要在满足系统设计的需求上，提供不同的唤醒方法。

该应用笔记讲述如何使用EXTI、RTC、USART、LPUART、LPTIMER、I2C和LVD等方法将系统从深度睡眠模式1唤醒。

该应用笔记基于GD32L233R-EVAL V1.0硬件板卡资源开发。

2. 深度睡眠模式 1 唤醒方法

2.1. EXTI 唤醒

使用两个按键，其中一个按键用于触发系统进入深度睡眠模式 1，另一个按键用于将系统从深度睡眠模式 1 唤醒。系统上电后，LED1 闪烁。当按下 Tamper 按键后，系统进入深度睡眠模式 1，LED1 停止闪烁；按下 Wakeup 按键后，系统从深度睡眠模式 1 唤醒，LED 继续闪烁。由于从深度睡眠唤醒后，系统将切换 IRC16M 作为系统时钟源，因此需要重新配置系统时钟才能保证 LED1 按照之前的频率闪烁。

EXTI 唤醒相关配置如下：

1. 按键配置

```
static void wakeup_key_init(void)
{
    rcu_periph_clock_enable(RCU_GPIOA);
    rcu_periph_clock_enable(RCU_SYSCFG);
    /* wakeup key init */
    gpio_mode_set(GPIOA, GPIO_MODE_INPUT, GPIO_PUPD_NONE, GPIO_PIN_0);

    /* EXTI line 0 configuration */
    nvic_irq_enable(EXTI0_IRQn, 2);
    syscfg_exti_line_config(EXTI_SOURCE_GPIOA, EXTI_SOURCE_PIN0);
    exti_init(EXTI_0, EXTI_INTERRUPT, EXTI_TRIG_FALLING);
    exti_interrupt_flag_clear(EXTI_0);
}

static void tamper_key_init(void)
{
    rcu_periph_clock_enable(RCU_GPIOC);
    rcu_periph_clock_enable(RCU_SYSCFG);
    /* tamper key init */
    gpio_mode_set(GPIOC, GPIO_MODE_INPUT, GPIO_PUPD_NONE, GPIO_PIN_13);

    /* EXTI line 13 configuration */
    nvic_irq_enable(EXTI10_15_IRQn, 2);
    syscfg_exti_line_config(EXTI_SOURCE_GPIOC, EXTI_SOURCE_PIN13);
    exti_init(EXTI_13, EXTI_INTERRUPT, EXTI_TRIG_FALLING);
    exti_interrupt_flag_clear(EXTI_13);
}
```

2. 按键中断处理

```
void EXTI0_IRQHandler(void)
```

```
{
    if(SET == exti_interrupt_flag_get(EXTI_0)) {
        /* clear EXTI line 0 pending flag */
        exti_interrupt_flag_clear(EXTI_0);
    }
}

void EXTI10_15_IRQHandler(void)
{
    if(SET == exti_interrupt_flag_get(EXTI_13)) {
        /* clear EXTI line 13 pending flag */
        exti_interrupt_flag_clear(EXTI_13);
        enter_deepsleep_flag = 1;
    }
}
```

2.2. RTC 唤醒

2.2.1. RTC 自动唤醒

系统上电后，LED1 闪烁。当 Tamper 按键按下后，产生 EXTI 中断，在此中断中开启 RTC 自动唤醒功能；退出 EXTI 中断后，MCU 进入进入深度睡眠模式 1 且 LED1 停止闪烁。当自动唤醒时间到达后，产生 RTC 自动唤醒事件并将系统从深度睡眠模式 1 唤醒，LED 继续闪烁。由于事先开启了 RTC 自动唤醒中断，在唤醒后将进入 RTC 唤醒中断，在中断中关闭自动唤醒功能。

RTC 自动唤醒相关配置如下：

1. RTC 自动唤醒配置

```
void rtc_configuration(void)
{
    rtc_parameter_struct     rtc_initpara;
    __IO uint32_t prescaler_a = 0x7F, prescaler_s = 0xFF;

    /* enable PMU and BKPI clocks */
    rcu_periph_clock_enable(RCU_PMU);
    rcu_periph_clock_enable(RCU_BKP);
    /* allow access to BKP domain */
    pmu_backup_write_enable();
    rcu_periph_clock_enable(RCU_RTC);

    /* enable LXTAL */
    rcu_osci_on(RCU_LXTAL);
    /* wait for LXTAL stabilization flag */
```

```
    rcu_osci_stab_wait(RCU_LXTAL);
    rcu_lxtal_clock_monitor_enable();
    /* configure the RTC clock source selection */
    rcu_rtc_clock_config(RCU_RTCSRC_LXTAL);

    rtc_register_sync_wait();
    /* setup RTC time value */
    rtc_initpara.factor_asyn = prescaler_a;
    rtc_initpara.factor_syn = prescaler_s;
    rtc_initpara.year = 0x16;
    rtc_initpara.day_of_week = RTC_WEDNESDAY;
    rtc_initpara.month = RTC_SEP;
    rtc_initpara.date = 0x07;
    rtc_initpara.display_format = RTC_24HOUR;
    rtc_initpara.am_pm = RTC_AM;
    rtc_initpara.hour = 0x09;
    rtc_initpara.minute = 0x28;
    rtc_initpara.second = 0;
    rtc_init(&rtc_initpara);

    /* EXTI line 20 configuration */
    nvic_irq_enable(RTC_WKUP_IRQn, 2);
    exti_flag_clear(EXTI_20);
    exti_init(EXTI_20, EXTI_INTERRUPT, EXTI_TRIG_RISING);
    rtc_flag_clear(RTC_STAT_WTF);

    /* wakeup clock configuration */
    rtc_wakeup_clock_set(WAKEUP_CKSPRE);
    rtc_wakeup_timer_set(2);
    rtc_interrupt_enable(RTC_INT_WAKEUP);
    rtc_wakeup_disable();
}
```

2. RTC 自动唤醒中断处理

```
void RTC_WKUP_IRQHandler(void)
{
    if(RESET != rtc_flag_get(RTC_FLAG_WT)) {
        /* clear EXTI line 20 pending and rtc wakeup flag */
        rtc_flag_clear(RTC_FLAG_WT);
        exti_flag_clear(EXTI_20);
        /* disable rtc auto wakeup function */
        rtc_wakeup_disable();
    }
}
```

3. 按键中断处理

```
void EXTI10_15_IRQHandler(void)
{
    if(SET == exti_interrupt_flag_get(EXTI_13)) {
        /* clear EXTI line 13 pending flag */
        exti_interrupt_flag_clear(EXTI_13);
        enter_deepsleep_flag = 1;
        /* enable RTC auto wakeup function */
        rtc_wakeup_enable();
    }
}
```

2.2.2. RTC 闹钟唤醒

系统上电后，LED1 闪烁。当 Tamper 按键按下后，产生 EXTI 中断，在此中断中更新 RTC 闹钟时间并开启闹钟功能；退出 EXTI 中断后，MCU 进入进入深度睡眠模式 1 且 LED1 停止闪烁。当闹钟时间到达后，产生 RTC 闹钟事件并将系统从深度睡眠模式 1 唤醒，LED 继续闪烁。。由于事先开启了 RTC 闹钟 0 中断，在唤醒后将进入 RTC 闹钟中断，在中断中关闭闹钟 0 功能。

RTC 闹钟唤醒相关配置如下：

1. RTC 闹钟唤醒配置

```
void rtc_configuration(void)
{
    rtc_parameter_struct     rtc_initpara;
    __IO uint32_t prescaler_a = 0x7F, prescaler_s = 0xFF;

    /* enable PMU and BKPI clocks */
    rcu_periph_clock_enable(RCU_PMU);
    rcu_periph_clock_enable(RCU_BKP);
    /* allow access to BKP domain */
    pmu_backup_write_enable();
    rcu_periph_clock_enable(RCU_RTC);

    /* enable LXTAL */
    rcu_osci_on(RCU_LXTAL);
    /* wait for LXTAL stabilization flag */
    rcu_osci_stab_wait(RCU_LXTAL);
    rcu_lxtal_clock_monitor_enable();
    /* configure the RTC clock source selection */
    rcu_rtc_clock_config(RCU_RTCSRC_LXTAL);

    rtc_register_sync_wait();
    /* setup RTC time value */
```

```
    rtc_initpara.factor_asyn = prescaler_a;
    rtc_initpara.factor_syn = prescaler_s;
    rtc_initpara.year = 0x16;
    rtc_initpara.day_of_week = RTC_WEDNESDAY;
    rtc_initpara.month = RTC_SEP;
    rtc_initpara.date = 0x07;
    rtc_initpara.display_format = RTC_24HOUR;
    rtc_initpara.am_pm = RTC_AM;
    rtc_initpara.hour = 0x09;
    rtc_initpara.minute = 0x28;
    rtc_initpara.second = 0;
    rtc_init(&rtc_initpara);

    /* RTC alarm configuration */
    rtc_alarm_struct    rtc_alarm;
    rtc_alarm_disable(RTC_ALARM0);
    rtc_alarm.alarm_mask    =    RTC_ALARM_DATE_MASK    |    RTC_ALARM_HOUR_MASK    |
RTC_ALARM_MINUTE_MASK;
    rtc_alarm.weekday_or_date = RTC_ALARM_DATE_SELECTED;
    rtc_alarm.alarm_day = 0x31;
    rtc_alarm.am_pm = RTC_AM;
    rtc_alarm.alarm_hour = 0x09;
    rtc_alarm.alarm_minute = 0x28;
    rtc_alarm.alarm_second = 0x00;
    rtc_alarm_config(RTC_ALARM0, &rtc_alarm);

    /* EXTI line 17 configuration */
    nvic_irq_enable(RTC_Alarm_IRQn, 0);
    exti_flag_clear(EXTI_17);
    exti_init(EXTI_17, EXTI_INTERRUPT, EXTI_TRIG_RISING);
    rtc_flag_clear(RTC_STAT_ALRM0F);

    /* enable alram 0 interrupt */
    rtc_interrupt_enable(RTC_INT_ALARM0);
    rtc_alarm_disable(RTC_ALARM0);
}
```

2. RTC 闹钟中断处理

```
void RTC_Alarm_IRQHandler(void)
{
    if(RESET != rtc_flag_get(RTC_FLAG_ALARM0)) {
        /* clear EXTI line 20 pending and rtc alarm flag */
        rtc_flag_clear(RTC_FLAG_ALARM0);
        exti_flag_clear(EXTI_17);
```

```
        /* disable rtc alarm 0 function */
        rtc_alarm_disable(RTC_ALARM0);
    }
}
```

3. 按键中断处理

```
void EXTI10_15_IRQHandler(void)
{
    if(SET == exti_interrupt_flag_get(EXTI_13)) {
        /* clear EXTI line 13 pending flag */
        exti_interrupt_flag_clear(EXTI_13);
        enter_deepsleep_flag = 1;
        /* update rtc alarm time*/
        rtc_alarm_update(0x03);
    }
}
```

2.3. USART 唤醒

系统上电后，LED1 闪烁。按下 Tamper 按键按下后，系统进入深度睡眠模式 1，LED1 停止闪烁；当 USART1 接收到数据后，系统从深度睡眠模式 1 唤醒，LED1 继续闪烁。此后，USART 可正常收发数据。用户可以通过

注意：在使用 USART1 的唤醒模式时，需配置 USART1 时钟源为 IRC16M 或 LXTAL 并且在进入深度睡眠模式前需禁用 DMA 功能。

USART 唤醒相关配置如下：

1. USART1 唤醒配置

```
void usart1_init(void)
{
    /* enable COM GPIO clock */
    rcu_periph_clock_enable(RCU_GPIOA);

    /* USART configuration the CK_IRC16M as USART clock */
    rcu_usart_clock_config(IDX_USART1, RCU_USARTSRC_IRC16MDIV);
    /* enable USART clock */
    rcu_periph_clock_enable(RCU_USART1);

    /* connect port to USARTx_Tx */
    gpio_af_set(GPIOA, GPIO_AF_7, GPIO_PIN_2);
    /* connect port to USARTx_Rx */
    gpio_af_set(GPIOA, GPIO_AF_7, GPIO_PIN_3);

    /* configure USART Tx as alternate function push-pull */
```

```
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_2);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_2);
    /* configure USART Rx as alternate function push-pull */
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_3);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_3);

    /* USART configure */
    usart_deinit(USART1);
    usart_baudrate_set(USART1, 115200U);
    usart_receive_config(USART1, USART_RECEIVE_ENABLE);
    usart_transmit_config(USART1, USART_TRANSMIT_ENABLE);

    rcu_periph_clock_enable(RCU_SYSCFG);
    nvic_irq_enable(USART1_WKUP_IRQn, 2);
    /* USART1 Wakeup EXTI line configuretion */
    exti_init(EXTI_26, EXTI_INTERRUPT, EXTI_TRIG_RISING);

    /* use start bit wakeup MCU */
    usart_wakeup_mode_config(USART1, USART_WUM_STARTB);
    /* enable USART */
    usart_enable(USART1);

    /* ensure USART is enabled */
    while(RESET == usart_flag_get(USART1, USART_FLAG_REA)) {
    }
    /* check USART is not transmitting */
    while(SET == usart_flag_get(USART1, USART_FLAG_BSY)) {
    }

    usart_wakeup_enable(USART1);
    /* enable the WUIE interrupt */
    usart_interrupt_enable(USART1, USART_INT_WU);
}
```

2. USART1 中断处理

```
void USART1_WKUP_IRQHandler(void)
{
    if(SET == usart_interrupt_flag_get(USART1, USART_INT_FLAG_WU)) {
        /* clear EXTI line 26 pending and wakeup flag */
        usart_flag_clear(USART1, USART_FLAG_WU);
        exti_flag_clear(EXTI_26);
    }
}
```

2.4. LPUART 唤醒

系统上电后，LED1 闪烁。按下 Tamper 按键按下后，系统进入深度睡眠模式 1，LED1 停止闪烁；当 LPUART 接收到数据后，系统从深度睡眠模式 1 唤醒，LED1 继续闪烁。此后，LPUART 可正常收发数据。

注意：在使用 LPUART 的唤醒模式时，需配置 LPUART 时钟源为 IRC16M 或 LXTAL 并且在进入深度睡眠模式前需禁用 DMA 功能。

LPUART 唤醒相关配置如下：

1. LPUART 唤醒配置

```
void lpuart_init(void)
{
    /* enable COM GPIO clock */
    rcu_periph_clock_enable(RCU_GPIOA);
    /* configure the CK_IRC16M as LPUART clock */
    rcu_lpuart_clock_config(RCU_LPUARTSRC_IRC16MDIV);
    /* enable LPUART clock */
    rcu_periph_clock_enable(RCU_LPUART);

    /* connect port to LPUART_TX */
    gpio_af_set(GPIOA, GPIO_AF_8, GPIO_PIN_2);
    /* connect port to LPUART_RX */
    gpio_af_set(GPIOA, GPIO_AF_8, GPIO_PIN_3);

    /* configure LPUART Tx as alternate function push-pull */
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_2);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_2);
    /* configure LPUART Rx as alternate function push-pull */
    gpio_mode_set(GPIOA, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_3);
    gpio_output_options_set(GPIOA, GPIO_OTYPE_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_3);

    /* LPUART configure */
    lpuart_deinit();
    lpuart_word_length_set(LPUART_WL_8BIT);
    lpuart_stop_bit_set(LPUART_STB_1BIT);
    lpuart_parity_config(LPUART_PM_NONE);
    lpuart_baudrate_set(115200U);
    lpuart_receive_config(LPUART_RECEIVE_ENABLE);
    lpuart_transmit_config(LPUART_TRANSMIT_ENABLE);

    rcu_periph_clock_enable(RCU_SYSCFG);
    nvic_irq_enable(LPUART_WKUP_IRQn, 2);
```

```
    /* LPUART Wakeup EXTI line configuretion */
    exti_init(EXTI_28, EXTI_INTERRUPT, EXTI_TRIG_RISING);

    /* use start bit wakeup MCU */
    lpuart_wakeup_mode_config(LPUART_WUM_STARTB);
    /* enable LPUART */
    lpuart_enable();

    /* ensure LPUART is enabled */
    while(RESET == lpuart_flag_get(LPUART_FLAG_REA)) {
    }

    /* check LPUART is not transmitting */
    while(SET == lpuart_flag_get(LPUART_FLAG_BSY)) {
    }

    lpuart_wakeup_enable();
    /* enable the WUIE interrupt */
    lpuart_interrupt_enable(LPUART_INT_WU);
}
```

2. LPUART 中断处理

```
void LPUART_WKUP_IRQHandler(void)
{
    if(SET == lpuart_interrupt_flag_get(LPUART_INT_FLAG_WU)) {
        /* clear EXTI line 28 pending and wakeup flag */
        lpuart_flag_clear(LPUART_FLAG_WU);
        exti_flag_clear(EXTI_28);
    }
}
```

2.5. LPTIMER 唤醒

系统上电后，LED1 闪烁。当 Tamper 按键按下后，LPTIMER 启动；退出中断后，MCU 进入进入深度睡眠模式 1 且 LED1 停止闪烁。当 LPTIMER 计数器时间到达后，产生唤醒事件并将系统从深度睡眠模式 1 唤醒，LED 继续闪烁。由于事先开启了 LPTIMER 自动重装载中断，在唤醒后将进入 LPTIMER 全局中断，在中断中关闭 LPTIMER 功能。

注意：在使用 LPTIMER 的唤醒模式时，需配置 LPTIMER 时钟源为 IRC32K 或 LXTAL。

LPTIMER 唤醒相关配置如下：

1. LPTIMER 配置

```
void lptimer_config(void)
{
```

```
    /* LPTIMER clock */
    rcu_osci_on(RCU_IRC32K);
    rcu_osci_stab_wait(RCU_IRC32K);
    rcu_lptimer_clock_config(RCU_LPTIMERSRC_IRC32K);
    rcu_periph_clock_enable(RCU_LPTIMER);

    /* ----------------------------------------------------------------------
    LPTIMER Configuration:
    LPTIMER count with internal clock IRC32K, the prescaler is 16, the period is 1000.
    LPTIMERCLK = IRC32K / 32 = 1KHz
    ---------------------------------------------------------------------- */
    /* LPTIMER configuration */
    lptimer_parameter_struct lptimer_structure;
    /* deinit a LPTIMER */
    lptimer_deinit();
    /* initialize LPTIMER init parameter struct */
    lptimer_struct_para_init(&lptimer_structure);
    /* LPTIMER configuration */
    lptimer_structure.clocksource          = LPTIMER_INTERNALCLK;
    lptimer_structure.prescaler            = LPTIMER_PSC_32;
    lptimer_structure.extclockpolarity = LPTIMER_EXTERNALCLK_RISING;
    lptimer_structure.extclockfilter     = LPTIMER_EXTERNALCLK_FILTEROFF;
    lptimer_structure.triggermode          = LPTIMER_TRIGGER_SOFTWARE;
    lptimer_structure.extriggersource    = LPTIMER_EXTRIGGER_GPIO;
    lptimer_structure.extriggerfilter    = LPTIMER_TRIGGER_FILTEROFF;
    lptimer_structure.outputpolarity     = LPTIMER_OUTPUT_NOTINVERTED;
    lptimer_structure.outputmode           = LPTIMER_OUTPUT_PWMORSINGLE;
    lptimer_structure.countersource      = LPTIMER_COUNTER_INTERNAL;
    lptimer_init(&lptimer_structure);

    /* disable the registers shadow function */
    lptimer_register_shadow_disable();
    lptimer_timeout_disable();

    /* EXTI line 29 configuration */
    nvic_irq_enable(LPTIMER_IRQn, 2U);
    exti_init(EXTI_29, EXTI_INTERRUPT, EXTI_TRIG_RISING);
    exti_interrupt_flag_clear(EXTI_29);

    /* enable the LPTIMER interrupt */
    lptimer_interrupt_flag_clear(LPTIMER_INT_FLAG_CARM);
    lptimer_interrupt_enable(LPTIMER_INT_CARM);
    lptimer_stop();
}
```

2. LPTIMER 中断处理

```
void LPTIMER_IRQHandler()
{
    if(RESET != lptimer_interrupt_flag_get(LPTIMER_INT_FLAG_CARM)) {
        /* clear EXTI line 29 pending and lptimer interrupt flag */
        lptimer_interrupt_flag_clear(LPTIMER_INT_FLAG_CARM);
        exti_interrupt_flag_clear(EXTI_29);
        /* stop lptimer */
        lptimer_stop();
    }
}
```

3. 按键中断处理

```
void EXTI10_15_IRQHandler(void)
{
    if(SET == exti_interrupt_flag_get(EXTI_13)) {
        /* clear EXTI line 13 pending flag */
        exti_interrupt_flag_clear(EXTI_13);
        enter_deepsleep_flag = 1;
        /* LPTIMER single start */
        lptimer_single_start(1999U, 999U);
    }
}
```

2.6. I2C 唤醒

系统上电后，LED1 闪烁。按下 Tamper 按键按下后，系统进入深度睡眠模式 1，LED1 停止闪烁；当 I2C1 接收到匹配地址后，系统从深度睡眠模式 1 唤醒，LED1 继续闪烁。此后，I2C 可正常收发数据。

注意：在使用 I2C 的唤醒模式时，需配置 I2C 时钟源为 IRC16M。

I2C 唤醒相关配置如下：

1. I2C1 配置

```
void i2c_config(void)
{
    /* select the I2C1 clock source */
    rcu_i2c_clock_config(IDX_I2C1, RCU_I2CSRC_IRC16MDIV);
    /* enable GPIOB clock */
    rcu_periph_clock_enable(RCU_GPIOB);
    /* enable I2C1 clock */
    rcu_periph_clock_enable(RCU_I2C1);
    /* enable PMU clock */
    rcu_periph_clock_enable(RCU_PMU);
```

```
    /* connect PB10 to I2C1_SCL */
    gpio_af_set(GPIOB, GPIO_AF_4, GPIO_PIN_10);
    /* connect PB11 to I2C1_SDA */
    gpio_af_set(GPIOB, GPIO_AF_4, GPIO_PIN_11);
    /* configure GPIO pins of I2C1 */
    gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_10);
    gpio_output_options_set(GPIOB,          GPIO_OTYPE_OD,          GPIO_OSPEED_50MHZ,
GPIO_PIN_10);
    gpio_mode_set(GPIOB, GPIO_MODE_AF, GPIO_PUPD_PULLUP, GPIO_PIN_11);
    gpio_output_options_set(GPIOB,          GPIO_OTYPE_OD,          GPIO_OSPEED_50MHZ,
GPIO_PIN_11);

    /* configure I2C timing */
    i2c_timing_config(I2C1, 0, 0x3, 0);
    /* configure I2C address */
    i2c_address_config(I2C1, I2C1_OWN_ADDRESS7, I2C_ADDFORMAT_7BITS);
    /* enable I2C1 */
    i2c_enable(I2C1);

    /* initialize the EXTI line 27 */
    nvic_irq_enable(I2C1_WKUP_IRQn, 0);
    nvic_irq_enable(I2C1_EV_IRQn, 2);
    exti_init(EXTI_27, EXTI_INTERRUPT, EXTI_TRIG_RISING);

    /* enable wakeup from deep-sleep mode */
    i2c_wakeup_from_deepsleep_enable(I2C1);
    /* enable the I2C1 interrupt */
    i2c_interrupt_enable(I2C1, I2C_INT_ADDM | I2C_INT_RBNE | I2C_INT_STPDET);
}
```

2. I2C1 中断处理

```
void I2C1_WKUP_IRQHandler(void)
{
    if(RESET != exti_interrupt_flag_get(EXTI_27)) {
        /* clear EXTI line 27 pending flag */
        exti_interrupt_flag_clear(EXTI_27);
    }
}

void I2C1_EV_IRQHandler(void)
{
    if(i2c_interrupt_flag_get(I2C1, I2C_INT_FLAG_ADDSEND)) {
        /* clear the ADDSEND bit */
```

```
            i2c_interrupt_flag_clear(I2C1, I2C_INT_FLAG_ADDSEND);
            /* enable wakeup from deep-sleep mode */
            i2c_wakeup_from_deepsleep_enable(I2C1);
        } else if(i2c_interrupt_flag_get(I2C1, I2C_INT_FLAG_RBNE)) {
            /* if reception data register is not empty, I2C1 will read a data from I2C_RDATA */
            i2c_data_receive(I2C1);
        } else if(i2c_interrupt_flag_get(I2C1, I2C_INT_FLAG_STPDET)) {
            /* clear STPDET interrupt flag */
            i2c_interrupt_flag_clear(I2C1, I2C_INT_FLAG_STPDET);
        }
}
```

2.7. LVD 唤醒

系统上电后，LED1 闪烁。按下 Tamper 按键按下后，系统进入深度睡眠模式 1，LED1 停止闪烁；当 VDD 电压低于低电压监测器阈值后，系统从深度睡眠模式 1 唤醒，LED 继续闪烁。

LVD 唤醒相关配置如下：

1. LVD 配置

```
void lvd_detect_init(void)
{
    nvic_irq_enable(LVD_IRQn, 2);
    rcu_periph_clock_enable(RCU_PMU);

    exti_init(EXTI_16, EXTI_INTERRUPT, EXTI_TRIG_RISING);
    exti_interrupt_flag_clear(EXTI_16);
    /* voltage threshold is 3.0V */
    pmu_lvd_select(PMU_LVDT_6);
}
```

2. LVD 中断处理

```
void LVD_IRQHandler(void)
{
    if(RESET != exti_interrupt_flag_get(EXTI_16)) {
        /* clear EXTI line 16 pending flag */
        exti_interrupt_flag_clear(EXTI_16);
    }
}
```

3. 实验

本实验使用的主程序代码如下，用户可以使用指定的宏来决定将使用哪种方式去将 MCU 从深度睡眠模式 1 唤醒。实验现象具体参考*深度睡眠模式 1 唤醒方法*。

```
int main(void)
{
    systick_config();
    /* init LED1 */
    gd_eval_led_init(LED1);
#ifdef WAKEUP_KEY
    wakeup_key_init();
#endif
#if defined (WAKEUP_RTC_AUTO) || defined (WAKEUP_RTC_ALARM)
    rtc_configuration();
#elif defined (WAKEUP_USART)
    usart1_init();
#elif defined (WAKEUP_LPUART)
    lpuart_init();
#elif defined (WAKEUP_LPTIMER)
    lptimer_config();
#elif defined (WAKEUP_I2C)
    i2c_config();
#elif defined (WAKEUP_LVD)
    lvd_detect_init();
#endif
    tamper_key_init();

    while(1) {
        if(1U == enter_deepsleep_flag) {
            enter_deepsleep_flag = 0;
            system_staus = RUN_DEEPSLEEP1;
            /* enter deep-sleep1 mode */
            system_enter_deepsleep1();
            exti_interrupt_disable(EXTI_13);
        }

      if((0U == enter_deepsleep_flag) && (RUN_DEEPSLEEP1 == system_staus)) {
            /* reconfig system clock */
            system_clock_reconfig();
            system_staus = RUN_NORMAL;
            exti_interrupt_enable(EXTI_13);
        }
```

```
    }
}
```

4. 版本历史

表格 4-1. 版本历史

版本号	描述	日期
1.0	首次发布	2023 年 5 月 25 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.

© 2023 GigaDevice – All rights reserved