GigaDevice Semiconductor Inc.

Wakeup methods from deep-sleep 1 mode for GD32L233 series

Application Note

AN094

Revision 1.0

(May. 2023)

Table of Contents

Table of Contents 2
List of Tables 3
1. Introduction 4
2. Deep-sleep 1 mode wakeup methods 5
2.1. EXTI wakeup 5
2.2. RTC wakeup 6
2.2.1. RTC auto wakeup 6
2.2.2. RTC alarm wakeup 8
2.3. USART wakeup 10
2.4. LPUART wakeup 12
2.5. LPTIMER wakeup 13
2.6. I2C wakeup 15
2.7. LVD wakeup 17
3. Experimental results 19
4. Revision history 21

List of Tables

Table 4-1. Revision history 21

1. Introduction

In the development of embedded system applications, low power application scenarios are often encountered. The GD32L233 microcontroller provides 10 power-saving modes, namely Run mode, Run1 mode, Run2 mode, Sleep mode, Sleep1 mode, Sleep2 mode, Deep-sleep mode, Deep-sleep 1 mode, Deep-sleep 2 mode and Standby mode. Deep-sleep 1 mode is often used in the development of GD32L233 low power system. When the system enters Deep-sleep 1 mode, the 1.1V domain clock, IRC16M, IRC48M, HXTAL and PLLs clocks are turned off, while NPLDO is turned off, LPLDO is turned on, SARAM and register contents are preserved. Any interruption or event from EXTI lines can wake up the system from Deep-sleep mode. For specific low power system, we need to provide different wakeup methods to meet the requirements of system design.

The application note describes how to wake the system from Deep-sleep 1 mode by using EXTI, RTC, USART, LPUART, LPTIMER, I2C, and LVD.

This application note is developed based on GD32L233R-EVAL V1.0 hardware board resources.

2. Deep-sleep 1 mode wakeup methods

2.1. EXTI wakeup

Using two buttons, one is to trigger the system to enter Deep-sleep 1 mode and the other is to wake up the system from Deep-sleep 1 mode. After the system is powered on, LED1 blinks. When Tamper button is pressed, the system enters Deep-sleep 1 mode and LED1 stops blinking. When the Wakeup button is pressed, the system wakes up from Deep-sleep 1 mode and the LED continues to blink. The system switches IRC16M as the system clock source after waking up from deepsleep. Therefore, you need to reconfigure the system clock to ensure that LED1 blinks at the previous frequency.

The configurations related to EXTI wakeup are as follows:

1. Configuration of buttons

```
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. Process of buttons interrupt

```
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 wakeup

2.2.1. RTC auto wakeup

After the system is powered on, LED1 blinks. When the Tamper button is pressed, the EXTI interrupt is generated and the RTC auto wakeup function is enabled. After exiting the EXTI interrupt, MCU enters Deep-sleep1 mode and LED1 stops blinking. When the auto wakeup time arrives, the RTC auto wakeup event will be generated and the system will be waked up from Deep-sleep 1 mode, and the LED will continue to blink. Due to the RTC wakeup interrupt is enabled in advance, the RTC wakeup interrupt will enter after wakeup event occurs, and the auto wakeup function will be disabled in the interrupt processing code.

The configurations related to RTC auto wakeup are as follows:

1. Configuration of RTC auto wakeup

```
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. Process of RTC auto wakeup interrupt

```
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. Process of button interrupt

```
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 alarm wakeup

After the system is powered on, LED1 blinks. When the Tamper button is pressed, the EXTI interrupt is generated and the RTC alarm time is updated and the alarm function is enabled. After exiting the EXTI interrupt, MCU enters Deep-sleep 1 mode and LED1 stops blinking. When the alarm time is reached, an RTC alarm event is generated and the system is waked up from Deep-sleep 1 mode, and the LED continues to blink. Due to the RTC alarm 0 interrupt is enabled in advance, the RTC alarm 0 interrup will enter after wakeup event occurs, and the alarm 0 function will be disabled in the interrupt processing code.

The configurations related to RTC alarm wakeup are as follows:

1. Configuration of RTC alarm wakeup

```
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. Process of RTC alarm interrupt

```
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. Process of button interrupt

```
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 wakeup

After the system is powered on, LED1 blinks. After pressing the Tamper button, the system enters Deep-sleep 1 mode and LED1 stops blinking. When USART1 receives the data, the system wakes up from Deep-sleep 1 mode, and LED1 continues to blink. After that, USART can send and receive data normally.

Note: When using the wakeup mode of USART1, set the clock source of USART1 to IRC16M or LXTAL and disable USART DMA function after enter the deepsleep mode.

The configurations related to USART wakeup are as follows:

1. Configuration of RTC USART1

```
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. Process of USART1 interrupt

```
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 wakeup

After the system is powered on, LED1 blinks. After pressing the Tamper button, the system enters to Deep-sleep 1 mode and LED1 stops blinking. When the LPUART receives the data, the system wakes up from Deep-sleep 1 mode and LED1 continues to blink. After that, LPUART can send and receive data normally.

Note: When using the wake up mode of LPUART, set the LPUART clock source to IRC16M or LXTAL and disable LPUART DMA function after enter the deepsleep mode.

The configurations related to LPUART wakeup are as follows:

1. Configuration of LPUART wakeup

```
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. Process of LPUART wakeup interrupt

```
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 wakeup

After the system is powered on, LED1 blinks. When Tamper button is pressed, the EXTI interrupt is generated and the LPTIMER starts. After exiting the EXTI interrupt, MCU enters Deep-sleep 1 mode and LED1 stops blinking. When the LPTIMER counter time is reached, a wakeup event is generated and the system is waked up from Deep-sleep 1 mode, and the

LED continues to blink. Due to the LPTIMER automatic reload interrupt is enabled in advance, it will enter to the global LPTIMER interrupt after wakeup event occurs, and disable the LPTIMER function during the interrupt.

Note: When using the wakeup mode of LPTIMER, set the clock source of LPTIMER to IRC32K or LXTAL.

The configurations related to LPTIMER wakeup are as follows:

1. Configuration of LPTIMER

```
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. Process of LPTIMER interrupt

```
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. Process of buttons interrupt

```
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 wakeup

After the system is powered on, LED1 blinks. After pressing the Tamper button, the system enters to Deep-sleep 1 mode and LED1 stops blinking. When I2C1 receives the matching address, the system wakes up from Deep-sleep 1 mode and LED1 continues to blink. After that, I2C can send and receive data normally.

Note: When using the I2C wakeup mode, the I2C clock source must be set to IRC16M.

The configurations related to I2C wakeup are as follows:

1. Configuration of I2C

```
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. Process of I2C interrupts

```
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 wakeup

After the system is powered on, LED1 blinks. After pressing the Tamper button, the system enters to Deep-sleep 1 mode and LED1 stops blinking. After the V_{DD} voltage falls below the low-voltage monitor threshold, the system wakes up from Deep-sleep 1 mode and the LED continues to blink.

The configurations related to LVD wakeup are as follows:

1. Configuration of LVD

```
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. Process of LVD interrupt

```
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. Experimental results

The main program code used in this experiment is as follows. The user can use the related macro to decide which method to use to wake the MCU from Deep-sleep 1 mode. Experimental phenomenon refer to the ***Deep-sleep 1 mode wakeup methods***。

```
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. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	May.25, 2023