

GigaDevice Semiconductor Inc.

Arm[®] Cortex[®]- M3/M4/M23/M33 32-bit MCU

**Application Note
AN044**

Table of Contents

Table of Contents	2
List of Figures	3
List of Tables	4
1. Introduction	5
2. IRC8M calibration principle	6
3. Software implementation	8
4. Experimental results	12
5. Revision history	13

List of Figures

Figure 2-1. The principle of LXTAL calibration IRC8M	7
Figure 2-2. Hardware connection block diagram.....	7
Figure 4-1. Serial output IRC8M measuring frequency	12
Figure 4-2. Oscilloscope output IRC8M actual frequency	12

List of Tables

Table 5-1. Revision history.....	13
----------------------------------	----

1. Introduction

GD32F10x microcontrollers can use the internal IRC8M oscillator to run. When temperature is 25°C and working voltage is 3.3V, the accuracy range of IRC8M is $\pm 1\%$. When temperature is 40 to 105°C, the accuracy range of IRC8M is -2.5% to 1.5%. Therefore, the actual application environment will determine the clock frequency of the IRC8M. GD32F10x microcontroller provides the possibility of IRC8M calibration, which can calibrate the working frequency of IRC8M to achieve the compensation of working temperature.

This application note describes how to use LXTAL (low-speed external clock) to calibrate the internal high-speed clock IRC8M.

2. IRC8M calibration principle

IRC8M frequency of the GD32F10x microcontroller can be calibrated to $8\text{MHz} \pm 1\%$ (25°C , 3.3V) by the factory. When the system is powered on, IRC8MCALIB[7:0] in the RCU_CTL register will be initialized to the factory calibration value, and the value of IRC8MADJ[4:0] in the RCU_CTL register is initialized to 16. The system can calibrate IRC8M by modifying the value of IRC8MADJ[4:0], and the calibration step is about 40KHz. That is, every time a step is increased, the frequency of IRC8M will increase by 40KHz.

Calibration principle refer to [Figure 2-1. The principle of LXTAL calibration IRC8M](#). In the calibration process, the 64 divider of the RTC clock can be used as the reference clock. The calibration steps are as follows:

1. Select the LXTAL (32.768KHz) as RTC clock source, set the COEN bit in the BKP_OCTL register, and turn on the RTC clock calibration output function. The output signal frequency is 512Hz.
2. Connect the RTC clock calibration output pin (PC13) to the TIMER capture input pin and use the TIMER capture value to calculate the actual operating frequency of the IRC8M.
3. Use the error between the actual operating frequency and the ideal value of 8M to determine how to modify the value of IRC8MADJ[4:0], and finally select the adjustment value with the smallest error to calibrate the IRC8M.

Assuming that N times update events and 1 time rising edge capture event are generated during one RTC calibration output cycle, the time of one capture cycle T_{capture} is:

$$T_{\text{capture}} = ((N * \text{CARL}) + \text{capture_value}) / F_{\text{timerclk}} \quad (2-1)$$

and

$$T_{\text{capture}} = 1 / F_{\text{ref}} \quad (2-2)$$

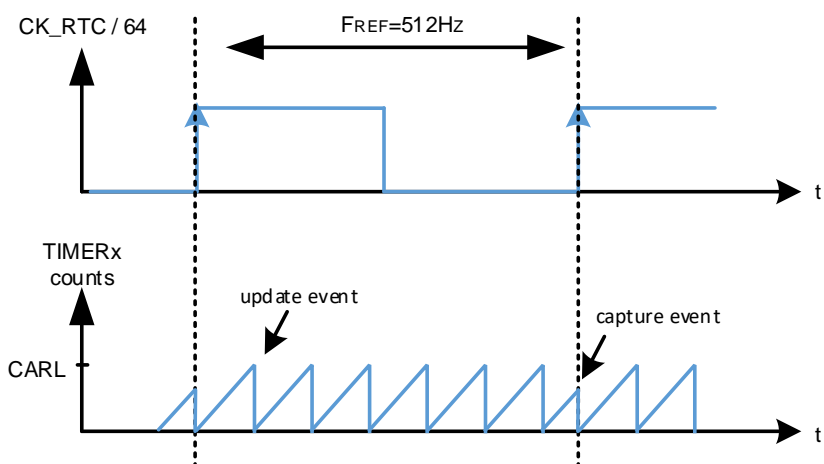
In among, the capture value is capture_value, the update count value is CARL, the timer count frequency is F_{timerclk} , and F_{ref} is the reference clock.

Assuming that the scale factor between the Timer counting period and IRC8M is m, that is:

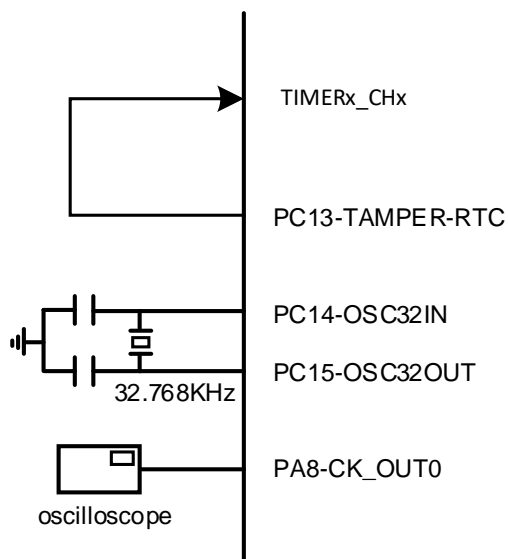
$$F_{\text{timerclk}} \approx m * F_{\text{IRC8M}} \quad (2-3)$$

then:

$$F_{\text{IRC8M}} = F_{\text{ref}} * ((N * \text{CARL}) + \text{capture_value}) / m \quad (2-4)$$

Figure 2-1. The principle of LXTAL calibration IRC8M


During the calibration process, the IRC8M clock can be output to the oscilloscope for observation through the CK_OUT0 (PA8) pin. The hardware connection block diagram is as follows:

Figure 2-2. Hardware connection block diagram


3. Software implementation

In this software implementation, the system frequency works at 108MHz. In among, the system clock source is selected as the PLL clock, and the PLL clock source is selected as IRC8M. The TIMER clock is 1MHz working frequency after pre-frequency division. The code implementation part is as follows:

1. Select LXTAL as the clock source of RTC and enable RTC clock calibration output.

```
void rtc_calibration_output_config(void) {
    /* enable PMU and BKPI clock */
    rcu_periph_clock_enable(RCU_PMU);
    rcu_periph_clock_enable(RCU_BKPI);
    pmu_backup_write_enable();
    /* turn on the LXTAL clock */
    rcu_osc_on(RCU_LXTAL);
    while(!rcu_osc_stab_wait(RCU_LXTAL));
    /* configure the RTC clock source as LXTAL */
    rcu_rtc_clock_config(RCU_RTCSRC_LXTAL);
    /* enable RTC clock calibration output */
    bkp_rtc_calibration_output_enable();
}
```

2. TIMER input capture configuration

```
void timer_capture_config (void)
{
    timer_ic_parameter_struct timer_icinitpara;
    timer_parameter_struct timer_initpara;
    /* configure the NVIC and TIMER interrupt */
    nvic_priority_group_set(NVIC_PRIGROUP_PRE1_SUB3);
    nvic_irq_enable(TIMER2_IRQn, 1, 1);
    rcu_periph_clock_enable(RCU_GPIOA);
    rcu_periph_clock_enable(RCU_AF);
    /* configure PA6 (TIMER2 CH0) as alternate function */
    gpio_init(GPIOA, GPIO_MODE_IN_FLOATING, GPIO_OSPEED_50MHZ, GPIO_PIN_6);
    /* configure TIMER2 CH0 pin (PA6) */
    rcu_periph_clock_enable(RCU_TIMER2);
    timer_deinit(TIMER2);
    timer_initpara.prescaler      = 107;
    timer_initpara.alignedmode    = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection = TIMER_COUNTER_UP;
    timer_initpara.period        = TIMER_PERIOD_VALUE -1 ;
    timer_initpara.clockdivision  = TIMER_CKDIV_DIV1;
    timer_initpara.repetitioncounter = 0;
    timer_init(TIMER2, &timer_initpara);
}
```



```

/* TIMER2 CH0 input capture configuration */
timer_icinitpara.icpolarity = TIMER_IC_POLARITY_RISING;
timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;
timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;
timer_icinitpara.icfilter = 0x0;
timer_input_capture_config(TIMER2, TIMER_CH_0,&timer_icinitpara);

/* auto-reload preload enable */
timer_auto_reload_shadow_enable(TIMER2);
/* clear channel 0 interrupt bit */
timer_interrupt_flag_clear(TIMER2, TIMER_INT_FLAG_CH0);
timer_interrupt_flag_clear(TIMER2, TIMER_INT_FLAG_UP);
/* channel 0 interrupt enable */
timer_interrupt_enable(TIMER2, TIMER_INT_CH0);
timer_interrupt_enable(TIMER2, TIMER_INT_UP);
/* TIMER2 counter enable */
timer_enable(TIMER2);
}

```

3. TIMER interrupt processing

```

void TIMER2_IRQHandler(void)
{
    if(SET == timer_interrupt_flag_get(TIMER2, TIMER_INT_UP)) {
        timer_interrupt_flag_clear(TIMER2, TIMER_INT_UP);
        /* count the update event */
        up_event_counts ++;
    }
    if(SET == timer_interrupt_flag_get(TIMER2, TIMER_INT_CH0)) {
        timer_counter_value_config(TIMER2, 0);
        timer_interrupt_flag_clear(TIMER2, TIMER_INT_CH0);
        if(0 != capture_event_counts)
        {
            /* get the capture value and calculate n times capture period */
            capture_value =
timer_channel_capture_value_register_read(TIMER2, TIMER_CH_0);
            ref_period_counts = up_event_counts*(TIMER_PERIOD_VALUE) +
capture_value;
            n_time_ref_period_counts += ref_period_counts;
        }
        up_event_counts = 0;
        if(capture_event_counts++ == TIMER_CAPTURE_NUMS)
        {
            /* calculate average capture period */
            n_time_average_counts = (uint32_t)(n_time_ref_period_counts /

```

Use LXTAL to calibrate IRC8M in GD32F10x

```

TIMER_CAPTURE_NUMS);
        n_time_ref_period_counts = 0;
        capture_event_counts = 0;
    }
}
}

```

4. According to the adjustment value, calculate the IRC8M measurement frequency under each adjustment value.

```

void get_irc8m_adjust_value_array(uint32_t irc8m_measure_array[], uint8_t array_len)
{
    uint8_t i = 0;
    do {
        /* get the IRC8M measurement frequency of every adjust value */
        rcu_irc8m_adjust_value_set(i);
        delay_1ms(100);
        irc8m_measure_value[i] = IRC_8M_REAL;
    } while(i++ < array_len);
}

```

5. According to the measurement frequency of IRC8M, find the adjustment value with the smallest error accuracy.

```

uint8_t get_min_error_adjust_value(uint32_t irc8m_measure_array[], uint8_t array_len)
{
    uint8_t i = 0;
    uint8_t min_error_adjust_value = 0;
    uint32_t min_measure_error_value = 0xFFFFFFFF;
    uint32_t measure_error_absolute = 0;
    /* get the adjust value of smallest error */
    do{
        if(irc8m_measure_array[i] > IRC8M_IDEAL_VALUE){
            measure_error_absolute = irc8m_measure_array[i] - IRC8M_IDEAL_VALUE;
        }else{
            measure_error_absolute = IRC8M_IDEAL_VALUE - irc8m_measure_array[i];
        }
        if(measure_error_absolute < min_measure_error_value){
            min_measure_error_value = measure_error_absolute;
            min_error_adjust_value = i;
        }
    }while(i++ < array_len);
    return min_error_adjust_value;
}

```

6. Main program and experimental results

```

#define IRC8M_MEASURE      (4096*n_time_average_counts)

```

```

#define IRC8M_IDEAL_VALUE    (8000000)
#define ARRAY_LEN           (0x20)
/* measure value array of IRC8M */
uint32_t irc8m_measure_value[ARRAY_LEN] = {0};
extern uint32_t n_time_average_counts;
int main(void)
{
    uint8_t index;
    systick_config();
    rtc_calibration_output_config();
    timer_capture_config();
    ckout_config();
    gd_eval_com_init(EVAL_COM0);
    while(1){
        get_irc8m_adjust_value_array(irc8m_measure_value,ARRAY_LEN);
        index = get_min_error_adjust_value(irc8m_measure_value,ARRAY_LEN);
        /* adjust IRC8M as minium error */
        rcu_irc8m_adjust_value_set(index);
        delay_1ms(100);
        printf("\r\n%d==>%d\r\n", index, irc8m_measure_value[index]);
    }
}

```

7. CKOUT configuration

```

void ckout_config(void)
{
    rcu_periph_clock_enable(RCU_GPIOA);
    /* configure clock output pin */
    gpio_init(GPIOA, GPIO_MODE_AF_PP, GPIO_OSPEED_10MHZ, GPIO_PIN_8);
    rcu_ckout0_config(RCU_CKOUT0SRC_IRC8M);
}

```

4. Experimental results

[Figure 4-1. Serial output IRC8M measuring frequency](#) and [Figure 4-2. Oscilloscope output IRC8M actual frequency](#) show the measured frequency and actual frequency respectively after IRC8M calibration.

Figure 4-1. Serial output IRC8M measuring frequency

```

Adjust value:16
Minium error IRC8M frequency:8007680

Adjust value:16
Minium error IRC8M frequency:8011776

Adjust value:16
Minium error IRC8M frequency:8011776

Adjust value:16
Minium error IRC8M frequency:8011776

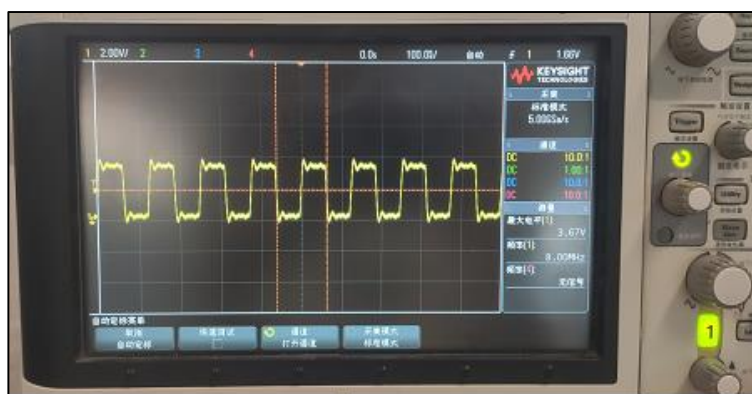
Adjust value:16
Minium error IRC8M frequency:8011776

Adjust value:16
Minium error IRC8M frequency:8007680

Adjust value:16
Minium error IRC8M frequency:8011776

Adjust value:16
Minium error IRC8M frequency:8011776
    
```

Figure 4-2. Oscilloscope output IRC8M actual frequency



5. Revision history

Table 5-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Oct.27,2021

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 its 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 its 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.