GigaDevice Semiconductor Inc.

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

应用笔记

AN044

目录

图索引

表索引

1. 介绍

GD32F10x 微控制器可以使用内部 IRC8M 振荡器运行。在温度 25℃，工作电压 3.3V 条件下，IRC8M 精度范围为±1%；在-40~105℃下，IRC8M 精度范围为-2.5%~1.5%。因此，实际应用环境将决定 IRC8M 的时钟频率。GD32F10x 微控制器提供了软件校准 IRC8M 的可能性，可校准 IRC8M 的工作频率，以达到对工作温度的补偿。

该应用笔记讲述如何使用 LXTAL（低速外部时钟）校准内部高速时钟 IRC8M。

2. IRC8M 校准原理

GD32F10x 微控制器在出厂时将 IRC8M 频率校准到 8MHz±1%（25℃，3.3V）。系统上电时，RCU_CTL 寄存器中 IRC8MCALIB[7:0]将被初始化为出厂校准值，且 RCU_CTL 寄存器中 IRC8MADJ[4:0]值初始化为 16。系统可通过修改 IRC8MADJ[4:0]的值来校准 IRC8M，其单位步长约为 40KHz，即每增加一个步长，IRC8M 将增加 40KHz 频率。

校准原理参考*图 2-1. LXTAL 校准 IRC8M 原理*。在校准过程中，可以使用 RTC 时钟的 64 分频作为参考时钟。校准步骤如下：

1. 选择 RTC 的时钟源为 LXTAL（32.768KHz），置位 BKP_OCTL 寄存器中的 COEN 位，开启 RTC 时钟校准输出功能，该输出信号频率为 512Hz；

2. 将 RTC 时钟校准输出引脚（PC13）连接至 TIMER 捕获输入引脚；通过 TIMER 捕获值来计算 IRC8M 的实际工作频率。

3. 根据实际工作频率与理想值 8M 之间的误差来决定如何修改 IRC8MADJ[4:0]值，最后选取具有最小的误差的调整值来校准 IRC8M。

设在一个 RTC 校准输出周期内产生 N 次定时器更新事件和 1 次上升沿捕获事件，则一次捕获周期的时间为：

$$Tcapture = ((N * CARL) + capture\_value) / Ftimerclk \quad (2\text{-}1)$$

且

$$Tcapture = 1 / Fref \quad (2\text{-}2)$$

其中，捕获值为 capture_value，更新计数值为 CARL，Timer 计数频率为 Ftimerclk，Fref 为参考时钟。

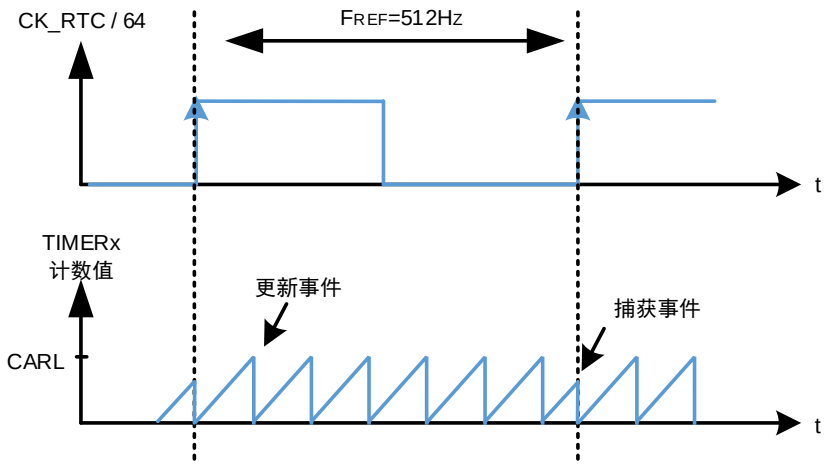
假设 Timer 计数周期与 IRC8M 的比例因子为 m，即：

$$Ftimerclk \approx m * FIRC8M \quad (2\text{-}3)$$

则：

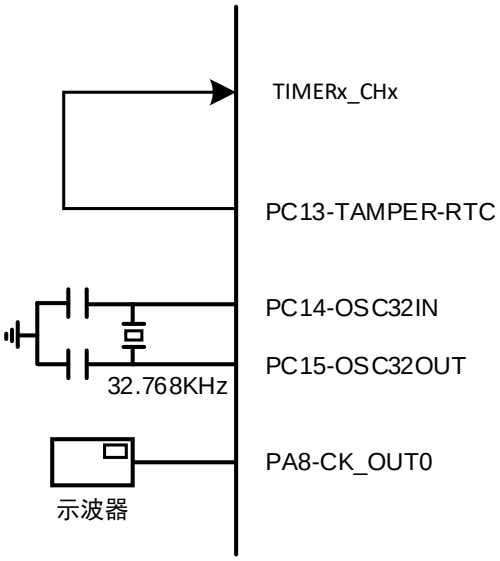
$$FIRC8M = Fref*((N * CARL) + capture\_value) / m \quad (2\text{-}4)$$

图 2-1. LXTAL 校准 IRC8M 原理



在校准过程中可通过 CK_OUT0（PA8）引脚将 IRC8M 时钟输出到示波器上观测，硬件连接框图如下：

图 2-2. 硬件连接框图



3. 软件实现

在此软件实现中，系统工作在 108MHz 频率下。其中，系统时钟源选择为 PLL 时钟，PLL 时钟源选择为 IRC8M。TIMER 时钟经预分频后为 1MHz 工作频率。代码实现部分如下：

1. 选择 LXTAL 作为 RTC 的时钟源并使能 RTC 时钟校准输出

```
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_osci_on(RCU_LXTAL);
    while(!rcu_osci_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 输入捕获配置

```
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 中断处理

```
    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        /
TIMER_CAPTURE_NUMS);
```

```
                    n_time_ref_period_counts = 0;
                    capture_event_counts = 0;
                }
            }
        }
```

4. 根据调整值，计算各调整值下的 IRC8M 测量频率

```
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. 根据 IRC8M 的测量频率，找出误差精度最小的调整值

```
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. 主程序及实验现象

```
#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 配置

```
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. 实验现象

*图 4-1. 串口输出 IRC8M 测量频率*与*图 4-2 示波器输出 IRC8M 实际频率*分别显示了 IRC8M 校准之后的测量频率和实际频率。

图 4-1. 串口输出 IRC8M 测量频率

```
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
```

图 4-2 示波器输出 IRC8M 实际频率

5. 版本历史

表 5-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2021 年 10 月 27 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.