GigaDevice Semiconductor Inc.

Calibration method of internal low-speed oscillator using TIMER

Application Note
AN064
Table of Contents

Table of Contents ................................................................. 2
List of Figures ................................................................. 3
List of Tables ................................................................. 4
1. Introduction ................................................................. 5
2. Calibration principle ...................................................... 6
3. Method of implementation ............................................... 8
   3.1. IRC40K edge capture ................................................ 8
   3.2. IRC40K frequency calculation .................................... 9
4. Revision history ............................................................ 11
List of Figures

- Figure 2-1. TIMER4_CH3 alternate function remapping ................................................................. 6
- Figure 2-2. TIMER4_CH3 capture IRC40K edges ................................................................. 7
List of Tables

Table 3-1. Configuration of TIMER ........................................................................................................... 8
Table 3-2. Interrupt handing .......................................................................................................................... 9
Table 3-3. IRC40K actual frequency calculation ............................................................................................ 10
Table 4-1. Revision history ............................................................................................................................ 11
1. **Introduction**

The low-speed oscillator (IRC32K or IRC40K) inside the GD32 MCU can be used as a low-power clock source, which has the advantage of not requiring external devices and thus reducing hardware costs. Its frequency is about 40kHz or 32kHz, and can provide the clock for peripherals such as free watchdog timer (FWDGT) and the real-time clock (RTC). The disadvantage is that the accuracy is low, and it is not suitable for many occasions, but it can be captured by TIMER for calibration. This article takes GD32F30x series MCU as an example to introduce the method of using TIMER to calibrate the internal low-speed oscillator.
2. Calibration principle

Even if it is the same series of MCUs, the internal low-speed clock of each MCU may be different and have a certain deviation, so when it provides a clock to RTC or FWDGT, the software sets the prescaler of RTC or FWDGT to a fixed value of 40K, which will cause a large error (For example: if the RTC prescaler is set at 40K clock to get the time base of 1S, then the prescaler value of RTC is set to 39999. When the actual frequency of IRC40K is less than 40K, the timing effect of RTC will be too fast).

GD32F30x series MCUs can be remapped through software configuration to connect the TIMER4_CH3 channel and IRC40K internally, refer to Figure 2-1. TIMER4_CH3 alternate function remapping.

Figure 2-1. TIMER4_CH3 alternate function remapping

<table>
<thead>
<tr>
<th>Alternate function</th>
<th>TIMER4CH3_IREMAP = 0</th>
<th>TIMER4CH3_IREMAP = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMER4_CH3</td>
<td>TIMER4_CH3 is connected to PA3</td>
<td>IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose</td>
</tr>
</tbody>
</table>

1. Remap available only for High-density and Extra-density and Connectivity lines devices.

The IRC40K edge can be captured by TIMER4_CH3 channels to obtain an accurate clock frequency, refer to Figure 2-2. TIMER4_CH3 capture IRC40K edges. Thereby modifying the clock divider value of peripherals such as RTC or FWDGT to obtain a more accurate time base. Then:

\[
T = t_1 - t_0 = \frac{CNT}{f_{ck}} \tag{2-1}
\]

\[
f_{IRC40K} = \frac{1}{T} = \frac{f_{ck}}{CNT} \tag{2-2}
\]

Where, \(t_0\) is the time when TIMER4 captures the falling edge of IRC40K for the first time, \(t_1\) is the time when TIMER4 captures the falling edge of IRC40K for the second time, \(T\) is the period of IRC40K, and \(f_{ck}\) is the clock of the TIMER4 counter. \(f_{ck}\) is known, CNT can be obtained by reading the TIMER counter by two captures, then \(f_{IRC40K}\) is calculated by formula (2-2).
Taking the application of RTC as an example, after calculating $f_{IRC40K}$, the actual IRC40K frequency can be used to configure the prescale value of RTC, which is used to generate the time base SC_CLK of RTC. Usually, the prescaler value is set as $f_{IRC40K-1}$ to obtain the RTC time base of 1S, so as to achieve the purpose of calibration.
3. Method of implementation

3.1. IRC40K edge capture

The falling edge of IRC40K is captured by configuring TIMER4_CH3, the code refer to Table 3-1. Configuration of TIMER.

Table 3-1. Configuration of TIMER

| /* TIMER4 configuration: input capture mode -------------------
| the RTC signal is connected to TIMER4 CH3
| the rising edge is used as active edge
| the TIMER4 CH3CV is used to compute the frequency value
| -------------------------------------------------------------- */
| timer_ic_parameter_struct timer_icinitpara;
| timer_parameter_struct timer_initpara;
| /* connect IRC40K clock to the TIMER4_CH3 input for calibration */
| rcu_periph_clock_enable(RCU_AF);
| gpio_pin_remap_config(GPIO_TIMER4CH3_IREMAP, ENABLE);
| rcu_periph_clock_enable(RCU_TIMER4);
| timer_deinit(TIMER4);
| /* initialize TIMER init parameter struct */
| timer_struct_para_init(&timer_initpara);
| /* TIMER4 configuration */
| timer_initpara.prescaler = 0;
| timer_initpara.alignedmode = TIMER_COUNTER_EDGE;
| timer_initpara.counterdirection = TIMER_COUNTER_UP;
| timer_initpara.period = 0xFFFF;
| timer_initpara.clockdivision = TIMER_CKDIV_DIV1;
| timer_init(TIMER4, &timer_initpara);
| /* TIMER4 CH3 input capture configuration */
| timer_icinitpara.icpolarity = TIMER_IC_POLARITY_FALLING;
| timer_icinitpara.icselection = TIMER_IC_SELECTION_DIRECTTI;
| timer_icinitpara.icprescaler = TIMER_IC_PSC_DIV1;
| timer_icinitpara.icfilter = 0x0;
| timer_input_capture_config(TIMER4, TIMER_CH_3, &timer_icinitpara);
Calibration method of internal low-speed oscillator using TIMER

```
/* auto-reload preload enable */
timer_auto_reload_shadow_enable(TIMER4);
/* clear channel 3 interrupt bit */
timer_interrupt_flag_clear(TIMER4, TIMER_INT_FLAG_CH3);
/* channel 3 interrupt enable */
timer_interrupt_enable(TIMER4, TIMER_INT_CH3);

/* TIMER4 counter enable */
timer_enable(TIMER4);
```

The falling edge of IRC40K captured by TIMER4_CH3 will trigger the capture interrupt, and the interrupt processing code is shown in Table 3-2, Interrupt handing.

**Table 3-2. Interrupt handing**

```c
void TIMER4_IRQHandler(void)
{
    if(SET == timer_interrupt_flag_get(TIMER4, TIMER_INT_CH3)){
        /* clear channel 3 interrupt bit */
        timer_interrupt_flag_clear(TIMER4, TIMER_INT_CH3);

        timer_capture_num++;
        if(1 == timer_capture_num){
            /* get the input capture value */
            timer_capture1 = timer_channel_capture_value_register_read(TIMER4, TIMER_CH_3);
        }else if(2 == timer_capture_num){
            /* get the input capture value */
            timer_capture2 = timer_channel_capture_value_register_read(TIMER4, TIMER_CH_3);

            /* end capture, disable TIMER4 and CH3 interrupt */
            timer_interrupt_disable(TIMER4, TIMER_INT_CH3);
            timer_disable(TIMER4);
        }
    }
}
```

### 3.2. IRC40K frequency calculation

After second capture, the values of the TIMER4 counters are timer_capture1 and timer_capture2 respectively, then the frequency of the IRC40K can be calculated and the prescaler value of the RTC can be configured, as shown in Table 3-3, IRC40K actual frequency calculation.
Calibration method of internal low-speed oscillator using TIMER

<table>
<thead>
<tr>
<th>Table 3-3. IRC40K actual frequency calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp_capture1 = timer_capture1;</td>
</tr>
<tr>
<td>temp_capture2 = timer_capture2;</td>
</tr>
<tr>
<td>timer_period = temp_capture2 - temp_capture1 + 1;</td>
</tr>
<tr>
<td>/* compute the actual frequency of the IRC40K (TIMER4_CLK = 2 * CK_APB1) */</td>
</tr>
<tr>
<td>if(0 != timer_period){</td>
</tr>
<tr>
<td>irc_freq = ((rcu_clock_freq_get(CK_APB1) * 2) / timer_period);</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>printf(&quot;timer_capture1 = %d, timer_capture2 = %d, irc_freq = %d\n&quot;, temp_capture1, temp_capture2, irc_freq);</td>
</tr>
<tr>
<td>/* adjust the RTC prescaler value to calibrate the clock */</td>
</tr>
<tr>
<td>rtc_lwoff_wait();</td>
</tr>
<tr>
<td>rtc_prescaler_set(irc_freq - 1);</td>
</tr>
<tr>
<td>rtc_lwoff_wait();</td>
</tr>
</tbody>
</table>
4. Revision history

Table 4-1. Revision history

<table>
<thead>
<tr>
<th>Revision No.</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Initial Release</td>
<td>May.20 2022</td>
</tr>
</tbody>
</table>