GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

Application Note
AN021
Table of Contents

Table of Contents ................................................................. 2
List of Figures ................................................................. 3
List of Tables ................................................................. 4
1. Introduction .............................................................. 5
2. Implementation of measuring code running time ....................... 6
   2.1. Measurement with Timer ............................................ 6
   2.2. Measurement with systick counter .................................. 10
   2.3. Measure with gpio flip ............................................ 13
   2.4. Measure with Keil online simulation ............................... 13
3. Revision history ............................................................ 16
List of Figures

Figure 2-1. Timer 1 measures the running time of the code and prints the result .......................... 10
Figure 2-2. Systick measurement code running time result printing .............................................. 12
Figure 2-3. Use logic analyzer to measure GPIO flip time result ................................................ 13
Figure 2-4. Configure SW port download mode ................................................................................ 14
Figure 2-5. Configure Trace interface ............................................................................................ 14
Figure 2-6. Reset running time ....................................................................................................... 15
Figure 2-7. Code runtime measurement .......................................................................................... 15
List of Tables

Table 2-1. Timer1 configuration parameters ................................................................. 6  
Table 2-2. Timer1 interrupt handle function ................................................................. 7  
Table 2-3. Test code function ....................................................................................... 7  
Table 2-4. Timer measures start and end functions ....................................................... 8  
Table 2-5. Timer measures start and end functions ....................................................... 9  
Table 2-6. Systick configuration function ..................................................................... 10  
Table 2-7. The main function of systick measuring code running time ....................... 11  
Table 2-8. GPIO flip configuration and main function ............................................... 13  
Table 3-1. Revision history ......................................................................................... 16
1. Introduction

In the process of MCU development, there are some problems, such as measuring the execution time of some codes or algorithms, comparing the running time of codes before and after optimization, and whether the waiting period designed in the code is accurate enough. This application note is based on gd32f10x series, using gd32f103c-eval development board, and the development environment is keil 5.27, four methods to measure code execution time are provided.

- Measure with timer
- Measure with systick counter
- Measure with gpio flip
- Measure with Keil online simulation
2. Implementation of measuring code running time

2.1. Measurement with Timer

The working frequency of MCU system clock is 108MHz, the division factor of Timer1 is 108, and the counter value is 10000. Set timer update interrupt and enable Timer1 update event interrupt. The configuration of Timer1 is shown in Table 2-1. Timer1 configuration parameters.

Table 2-1. Timer1 configuration parameters

```c
/**
 * brief configure the TIMER peripheral
 * param[in]  none
 * param[out] none
 * retval    none
 */
void timer_config(void)
{
    /* -------------------------------------
    TIMER1 Configuration:
    TIMER1CLK = SystemCoreClock/108 = 1MHz, the period is 1ms (10000/1000000 = 1ms).
    ------------------------------------- */
    timer_parameter_struct timer_initpara;

    rcu_periph_clock_enable(RCU_TIMER1);
    timer_deinit(TIMER1);
    /* initialize TIMER init parameter struct */
    timer_struct_para_init(&timer_initpara);
    /* TIMER1 configuration */
    timer_initpara.prescaler         = 107;
    timer_initpara.alignedmode       = TIMER_COUNTER_EDGE;
    timer_initpara.counterdirection  = TIMER_COUNTER_UP;
    timer_initpara.period            = 9999;
    timer_initpara.clockdivision     = TIMER_CKDIV_DIV1;

    timer_init(TIMER1, &timer_initpara);
    timer_interrupt_flag_clear(TIMER1, TIMER_INT_FLAG_UP);
    timer_interrupt_enable(TIMER1, TIMER_INT_UP);
}
/**
 * brief configure the TIMER1 interrupt
 * param[in]  none
 * param[out] none
 * retval    none
 */
```
Several methods of measuring MCU code running time

```c
void nvic_config(void)
{
    nvic_priority_group_set(NVIC_PRIGROUP_PRE1_SUB3);
    nvic_irq_enable(TIMER1_IRQHandler, 1, 1);
}
```

Define the external variable cnt_cycle. When the timer is updated with 1ms event, cnt_cycle plus one in the update interrupt.

**Table 2-2. Timer1 interrupt handle function**

```c
extern uint32_t cnt_cycle;

/*! 
\brief this function handles TIMER1 interrupt request.
\param[in] none
\param[out] none
\retval none
*/
void TIMER1_IRQHandler(void)
{
    if(SET == timer_interrupt_flag_get(TIMER1, TIMER_INT_FLAG_UP)){
        /* clear the update event interrupt flag bit */
        timer_interrupt_flag_clear(TIMER1, TIMER_INT_FLAG_UP);
        ++cnt_cycle;
    }
}
```

Write the code segment to be measured. In this project, 108 __nop () is written in the function `delay_1us`, that is, the code runs for 108 clock cycles for 1us. The code is shown in **Table 2-3. Test code function**.

**Table 2-3. Test code function**

```c
/*! 
\brief delay 1us
\param[in] none
\param[out] none
\retval none
*/
define delay_1us()
{
    do{
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
        __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
```
Several methods of measuring MCU code running time

```c
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
__nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
#endif

while(0)

/*! 
 \brief delay nms
 \param[in] nms: delay nms ms
 \param[out] none
 \retval none */
void delay_nms(uint32_t nms)
{
    uint32_t nus=nms*1000;
    while(--nus)
    {
        delay_1us();
    }
}
```

Time start function measure_ runtime_ start and measure_ runtime_end is shown in Table 2-4. Timer measures start and end functions.

**Table 2-4. Timer measures start and end functions**

```c
/*! 
 \brief start measure
 \param[in] none
 \param[out] none
 \retval none */
void measure_runtime_start()
{
    cnt_cycle = 0;
    timer_enable(TIMER1);
}

/*! 
 \brief end measurement
 \param[in] none
 \param[out] none
 \retval measure time */
```
Several methods of measuring MCU code running time

```c
float measure_runtime_end()
{
    float work_time;
    uint32_t cnt_num,tmp;
    cnt_num = TIMER_CNT(TIMER1);
    timer_disable(TIMER1);
    nvic_irq_disable(TIMER1_IRQn);
    tmp = cnt_cycle*10000 + cnt_num;
    work_time = (float)tmp/1000.0;
    return work_time;
}
```

Write code in the main function to measure the time of the code to be executed, and print it through the serial port. The code is shown in **Table 2-5. Timer measures start and end functions**.

**Table 2-5. Timer measures start and end functions**

```c
int main(void)
{
    gd_eval_com_init(EVAL_COM0);
    dbg_periph_enable(DBG_TIMER1_HOLD);
    timer_config();
    nvic_config();
    measure_runtime_start();
    delay_nms(1);
    rtimevla = measure_runtime_end();
    printf("The code run time is %f ms",rtimevla);
    while(1){
    }
}
```

The measurement results are printed by the upper computer, and the results are shown in **Figure 2-1. Timer 1 measures the running time of the code and prints the result**.
2.2. Measurement with systick counter

SysTick is a 24-bit countdown timer. When it counts to 0, it will automatically reload the initial value of the timing from the RELOAD register. In this section, systick uses the system clock as the clock input, and configures the systick interrupt to enter an interrupt every 1ms. Write the functions start_time, stop_time, and get_time to represent the start of the code timing, the end of the code timing and get the code running time. The function is shown in Table 2-6, Systick configuration function.

### Table 2-6. Systick configuration function

```c
uint32_t tick;
/*
 * 
 * brief strat time
 * param[in] none
 * param[out] none
 * retval none
 */

void start_time(void) {
    tick = 0;
    /* setup systick timer for 1000Hz interrupts */
    if (SysTick_Config(SystemCoreClock / 1000)){
        /* capture error */
        while (1){
        }
    }
}
```
Several methods of measuring MCU code running time

```c
/* configure the sysTick handler priority */
NVIC_SetPriority(SysTick_IRQn, 0x00U);

/*!
\brief stop time
\param[in] none
\param[out] none
\retval none
*/
void stop_time(void) {
    SysTick->CTRL &= SysTick_Counter_Disable;
    SysTick->VAL = SysTick_Counter_Clear;
}

/*!
\brief get time
\param[in] none
\param[out] none
\retval none
*/
uint32_t get_time(void) {
    uint32_t elapsed = (uint32_t)tick;
    return elapsed;
}

/*!
\brief this function handles SysTick exception
\param[in] none
\param[out] none
\retval none
*/
void SysTick_Handler(void)
{
    ++tick;
}
```

The code in the main function is shown in Table 2-7. The main function of sysTick measuring code running time.

Table 2-7. The main function of sysTick measuring code running time

```c
/*!
\brief main function
\param[in] none
```
Several methods of measuring MCU code running time

```c
/*
 * param[out] none
 * retval     none
 */

int main(void)
{
    uint32_t value;
    gd_eval_com_init(EVAL_COM0);
    start_time();
    delay_nms(3);
    stop_time();
    value=get_time();
    printf("The code run time is %d ms",value);
    while(1){
    }
}
```

The code running time measurement results are as follows:

**Figure 2-2. Systick measurement code running time result printing**

![Systick measurement code running time result printing](image)

**Note:** By changing the parameter of the SysTick_Config() function to change the base time of the systick timer entering the interrupt.
2.3. **Measure with gpio flip**

Configure GPIO PA1 to push-pull output mode, the default output is low level, configure the PA1 pin is pulled high before the code segment to be measured, and the level is pulled high after the code is run. Use the logic analyzer or oscilloscope trigger mode to test the code running time. The code is shown in *Table 2-8. GPIO flip configuration and main function*.

![Table 2-8. GPIO flip configuration and main function](image)

Use the logic analyzer to select the edge trigger, and the measurement results are as follows:

![Figure 2-3. Use logic analyzer to measure GPIO flip time result](image)

2.4. **Measure with Keil online simulation**

Use Jlink to connect to the target board, select SW Port in the Debug->setting tab, Enable Trace Setting in the Trace tab, configure Core clock as the system clock 108Mhz, click “OK”. The configuration is shown in *Figure 2-4. Configure SW port download mode*. 

![Figure 2-4. Configure SW port download mode](image)
Several methods of measuring MCU code running time

Figure 2.4. Configure SW port download mode

Configure SW port download mode

Figure 2.5. Configure Trace interface

Configure Trace interface

Enter the debugging interface, add breakpoints before and after the code to be measured, run to the breakpoint, select Reset Stop Watch(t1) in the lower right corner, clear the t1 running time, run at full speed, run to the end of the code, and view the code running time. The code running time result is shown in Figure 2.7. Code runtime measurement.
Several methods of measuring MCU code running time

Figure 2-6. Reset running time

Figure 2-7. Code runtime measurement
Several methods of measuring MCU code running time

3. 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>Apr.30, 2021</td>
</tr>
</tbody>
</table>
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.

© 2021 GigaDevice – All rights reserved