# 透过TIMER Capture抓取输入信号频率

NuMicro® 32 位系列微控制器范例代码介绍

## 文件信息

| | |
|---|---|
| 应用简述 | 本范例代码使用 M251 TIMER Capture 功能抓取输入信号频率和工作周期 |
| BSP 版本 | M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001 |
| 开发平台 | NuMaker-M251SD Ver1.1 |

*The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.*

*All data and specifications are subject to change without notice.*

*For additional information or questions, please contact: Nuvoton Technology Corporation.*

www.nuvoton.com

# 1. 概述

此范例代码使用M251、M252、M254、M256、M258系列TIMER2透过capture功能抓取输入信号频率、工作周期，以及TIMER0透过PWM功能输出模拟信号，可输入信号频率范围为 20 ~ 80kHz，将此两组TIMER管脚互接，确认TIMER Capture功能抓取的频率、工作周期，另外透过一组UART印出抓取结果。

## 1.1 原理

### 1.1.1 TIMER capture 功能抓取频率计算

TIMER capture主要透过设定中断，并根据中断次数用以计算抓取的频率数值，以及工作周期，TIMER capture中断方式可选择上升沿触发、下降沿触发、高电平周期时间、低电平周期时间，搭配系统时钟频率，即可得知抓取频率数值。

本范例代码将系统时钟设定为 24MHz，因此一个Clock时间为0.042us，如图 1-1，TIMER的一个Clock时间也会是0.042us，TIMER capture中断方式分别设定为高电平周期时间、低电平周期时间，并分别抓取输入信号高电平时TIMER的计数次数，以及低电平时TIMER的计数次数，如图 1-2，将这两个计数次数相加后，再乘以Clock时间的0.042us，即可得知输入信号一个信号的周期时间，如图 1-3，最后再将周期时间倒数即为输入频率。

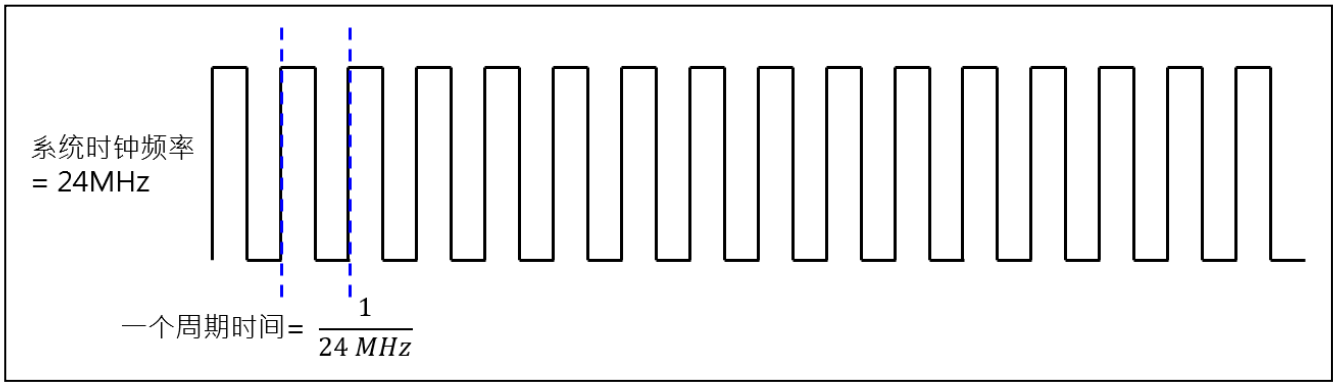


图 1-1 系统时钟一个周期时间

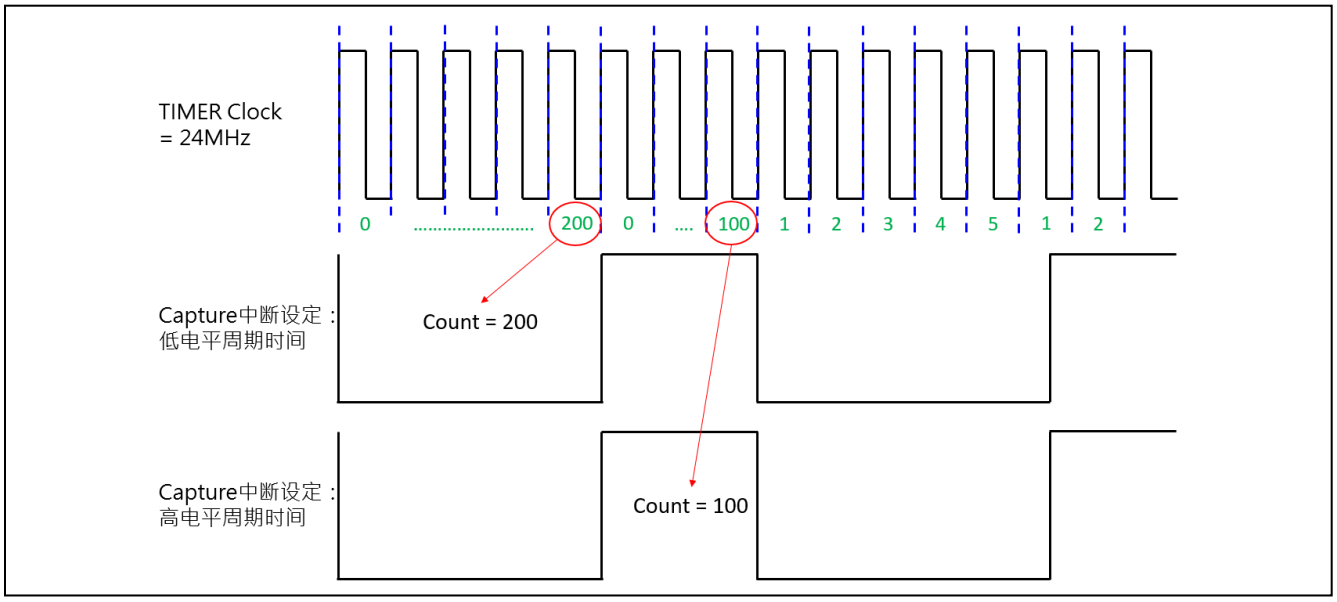


图 1-2 TIMER capture 中断方式

### 1.1.2 TIMER capture 功能抓取工作周期计算

TIMER capture中断方式分别设定为高电平周期时间、低电平周期时间，每发生一次Capture中断就Reset计数值，最后将高电平周期时间、低电平周期时间抓取到的计数值做计算即可得到输入信号的工作周期，如图 1-3。

$$工作周期 = \frac{\text{高电平周期时间Capture计数值}}{\text{高电平周期时间Capture计数值} + \text{低电平周期时间Capture计数值}} \times 100\%$$

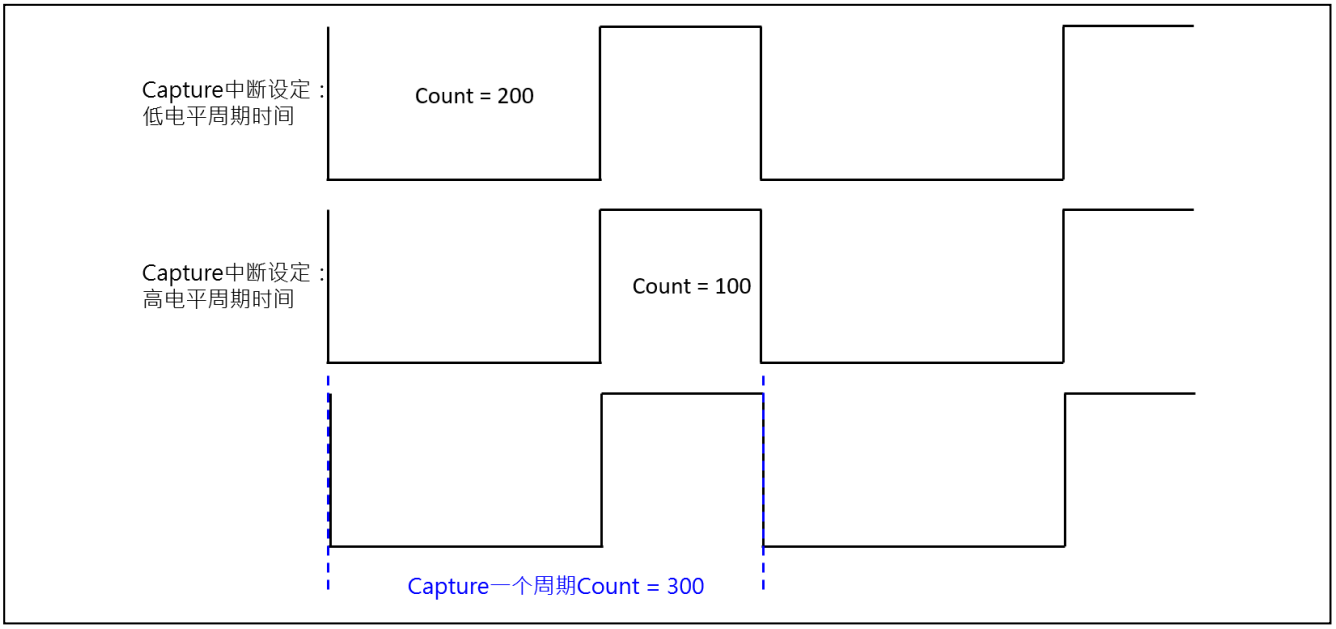


图 1-3 TIMER capture 抓取一个信号的周期时间计数

## 1.2 执行结果

TIMER0开启PWM功能，分别输出20 Hz、80 kHz信号，工作周期分别设定为10%、50%、100%，TIMER2开启Capture功能，抓取TIMER0输出的信号频率、工作周期，确认计算结果是否正确，并透过终端软件显示结果，请参考图 1-4 ~ 图 1-15。

### 1.2.1 PWM 信号(频率 20 Hz，工作周期 10%)

图 1-4 为 PWM 频率 20 Hz，工作周期 10% 时的波形图，图 1-5 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

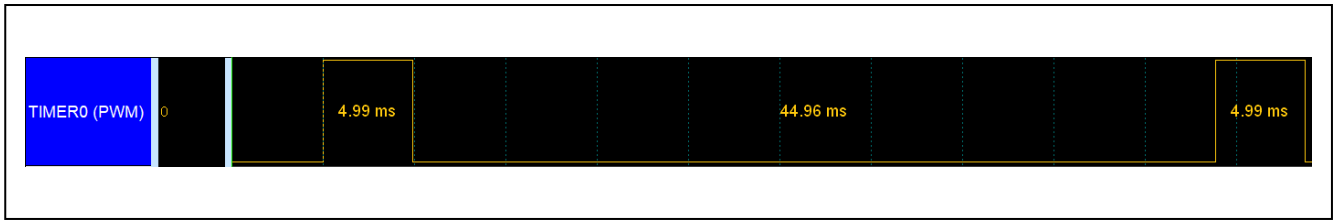


图 1-4 TIMER0 输出频率 20 Hz，工作周期 10%

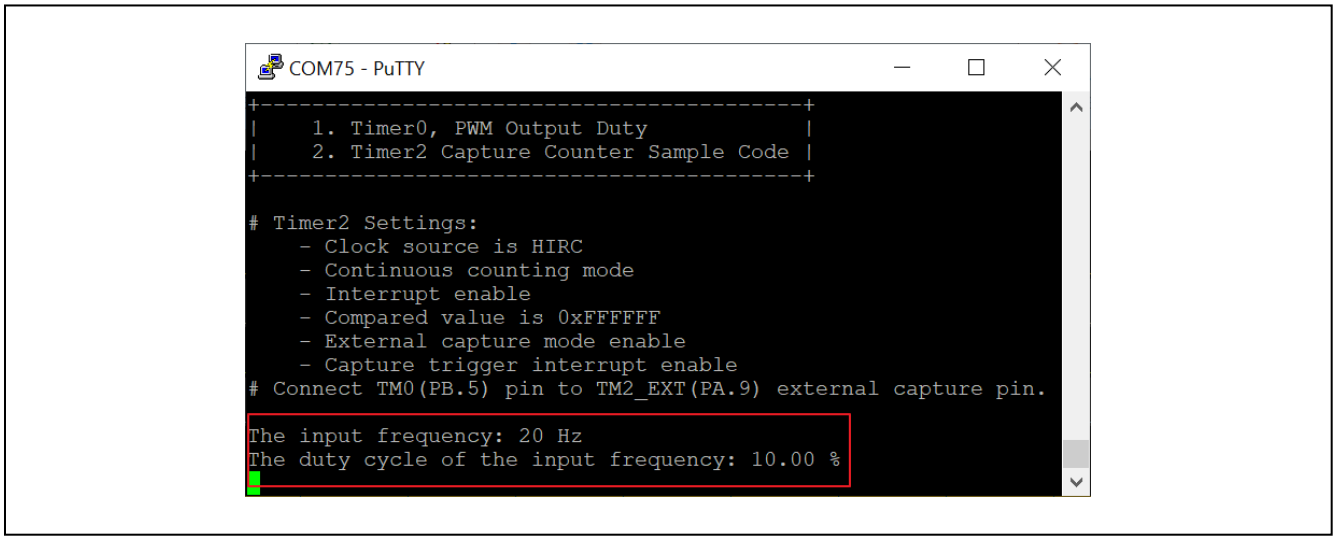


图 1-5 TIMER2 capture 抓取结果(频率 20 Hz，工作周期 10%)

### 1.2.2 PWM 信号(频率 20 Hz，工作周期 50%)

图 1-6 为 PWM 频率 20 Hz，工作周期 50% 时的波形图，图 1-7 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

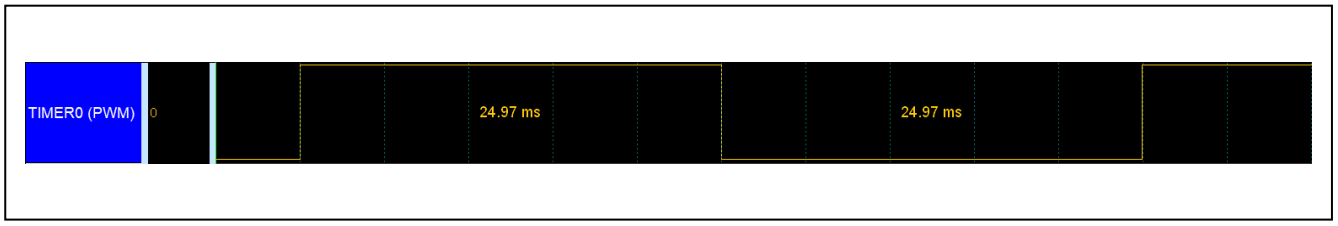


图 1-6 TIMER0 输出频率 20 Hz，工作周期 50%

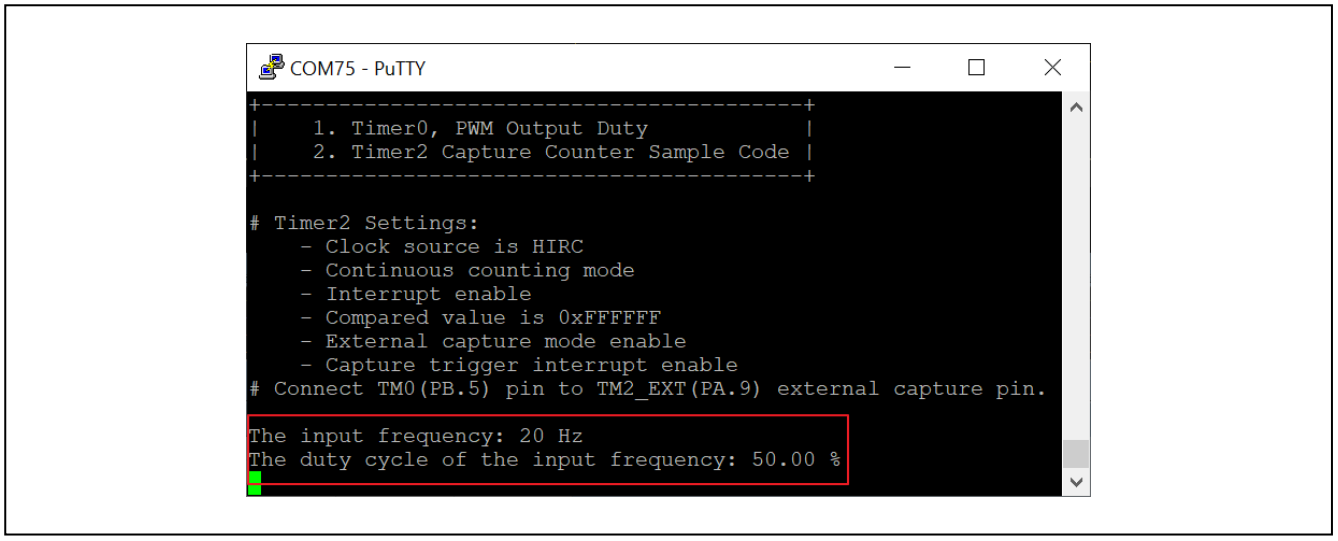


图 1-7 TIMER2 capture 抓取结果(频率 20 Hz，工作周期 50%)

### 1.2.3 PWM 信号(频率 20 Hz，工作周期 90%)

图 1-8 为 PWM 频率 20 Hz，工作周期 90% 时的波形图，图 1-9 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

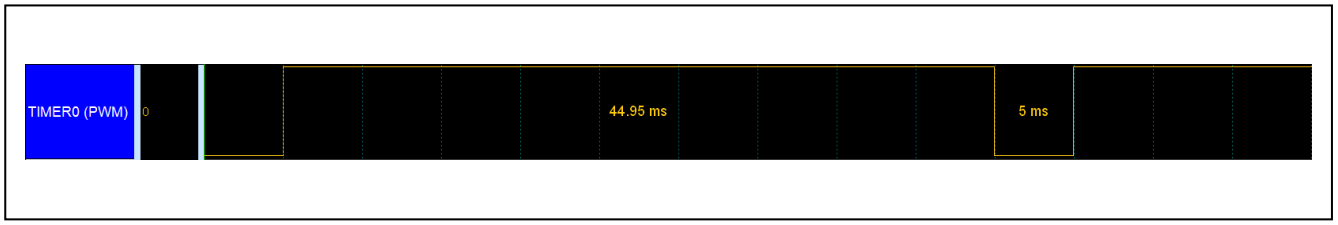


图 1-8 TIMER0 输出频率 20 Hz，工作周期 90%

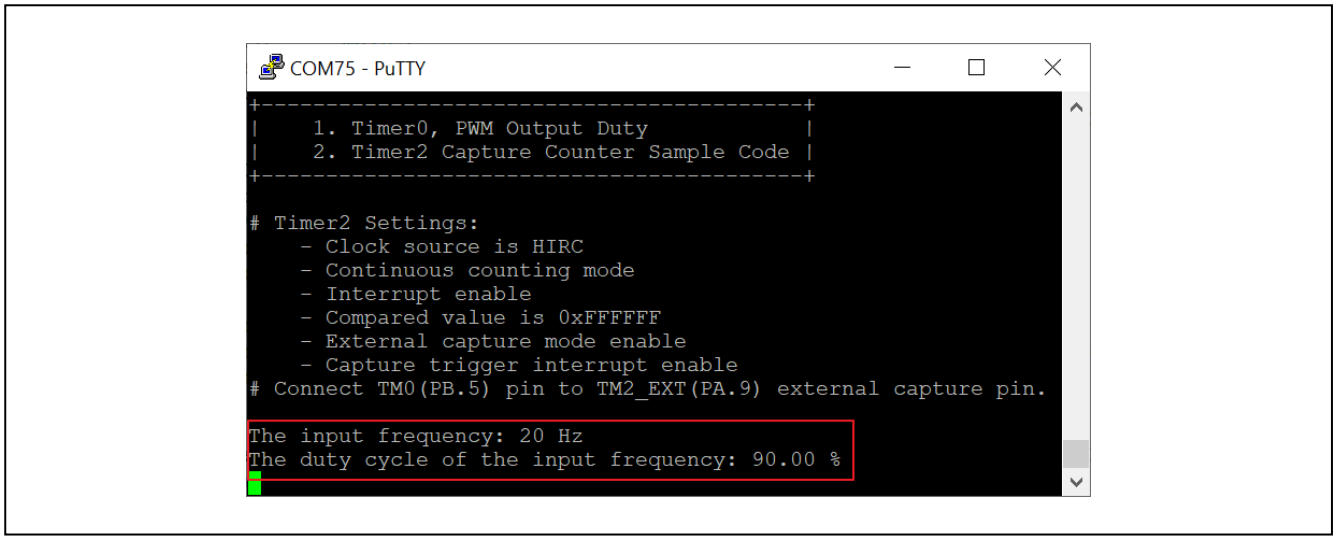


图 1-9 TIMER2 capture 抓取结果(频率 20 Hz，工作周期 90%)

### 1.2.4 PWM 信号(频率 80 kHz，工作周期 10%)

图 1-10 为 PWM 频率 80 kHz，工作周期 10% 时的波形图，图 1-11 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

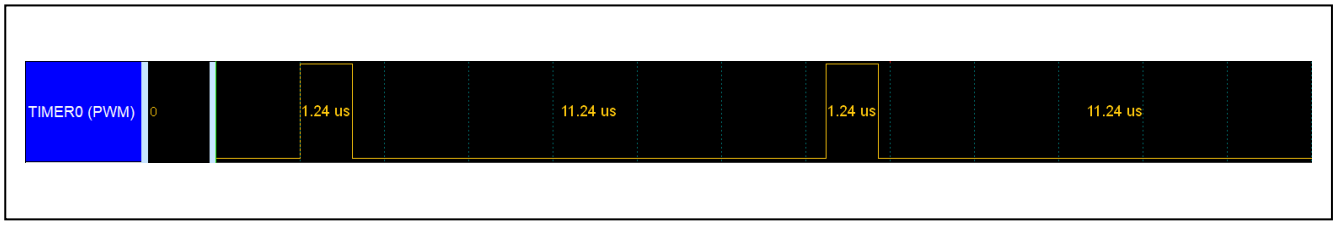


图 1-10 TIMER0 输出频率 80 kHz，工作周期 10%

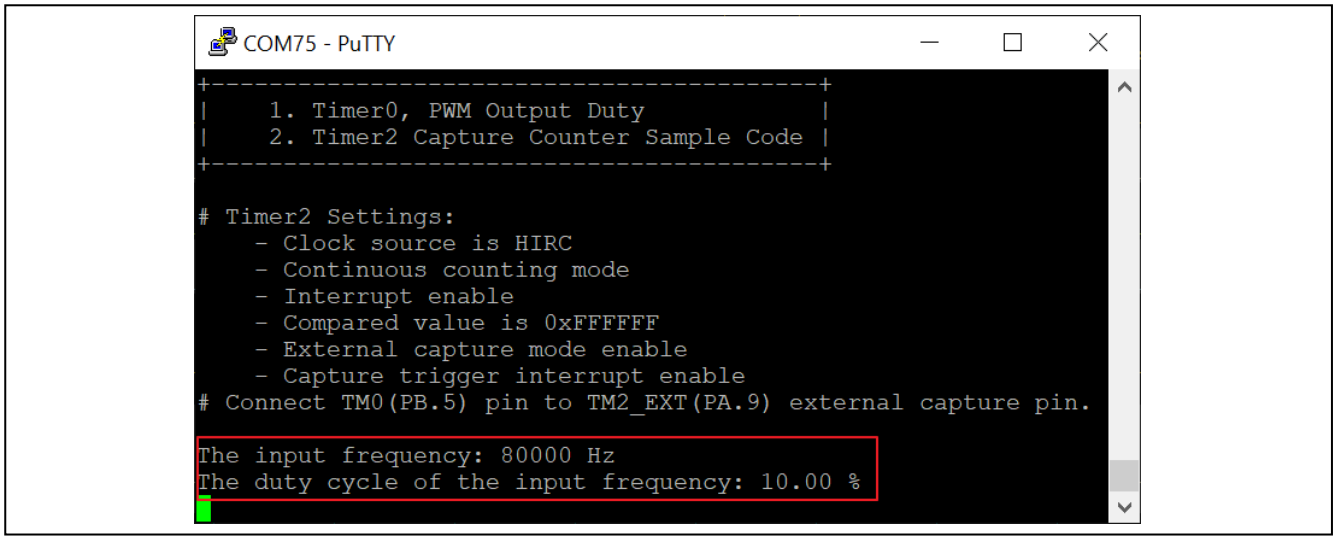


图 1-11 TIMER2 capture 抓取结果(频率 80 kHz，工作周期 10%)

### 1.2.5 PWM 信号(频率 80 kHz，工作周期 50%)

图 1-12 为 PWM 频率 80 kHz，工作周期 50% 时的波形图，图 1-13 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

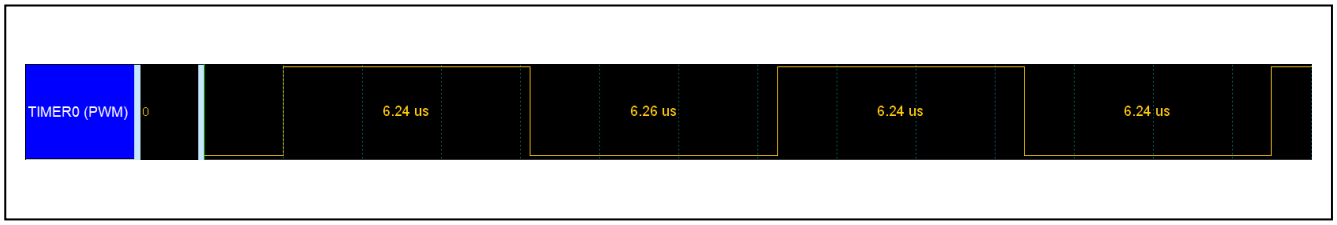


图 1-12 TIMER0 输出频率 80 kHz，工作周期 50%

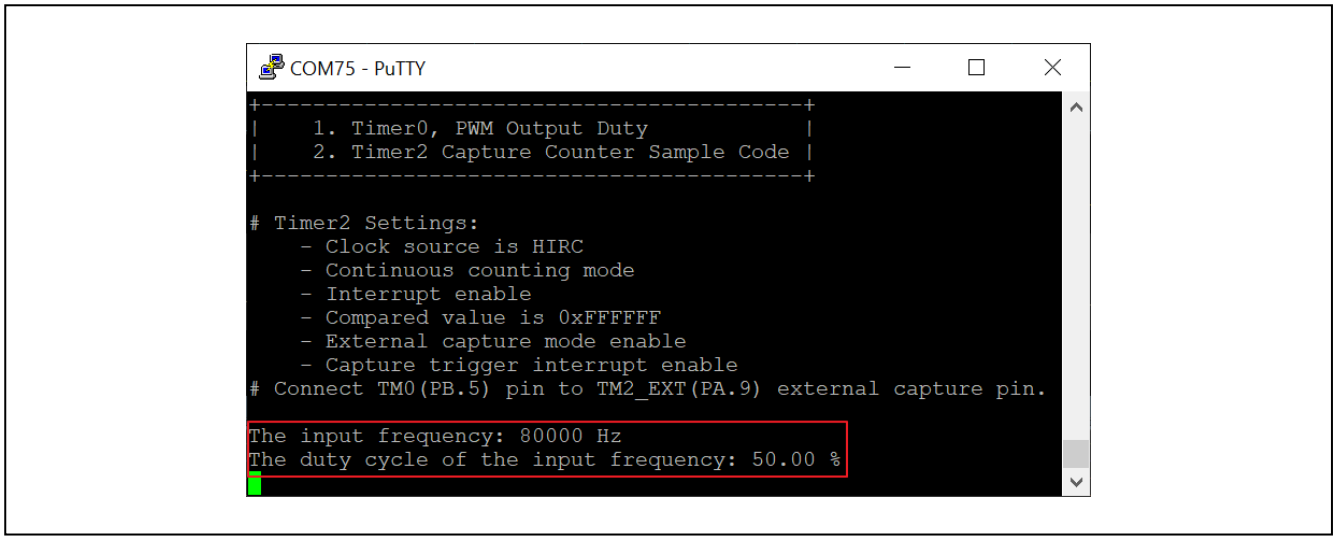


图 1-13 TIMER2 capture 抓取结果(频率 80 kHz，工作周期 50%)

### 1.2.6 PWM 信号(频率 80 kHz，工作周期 90%)

图 1-14 为 PWM 频率 80 kHz，工作周期 90% 时的波形图，图 1-15 为 TIMER2 Capture 实际抓取信号结果，透过终端软件呈现。

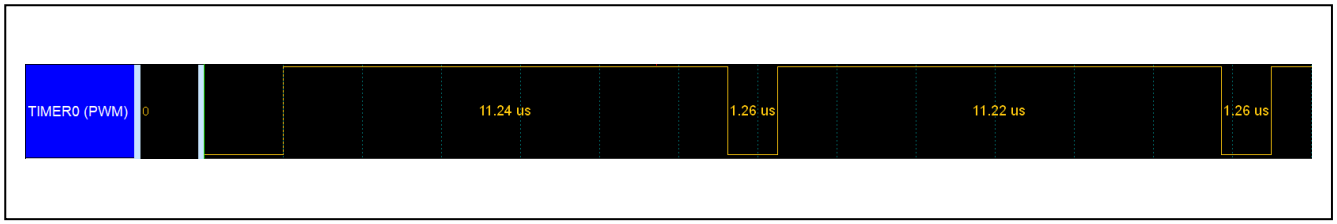


图 1-14 TIMER0 輸出頻率 80 kHz，工作週期 90%

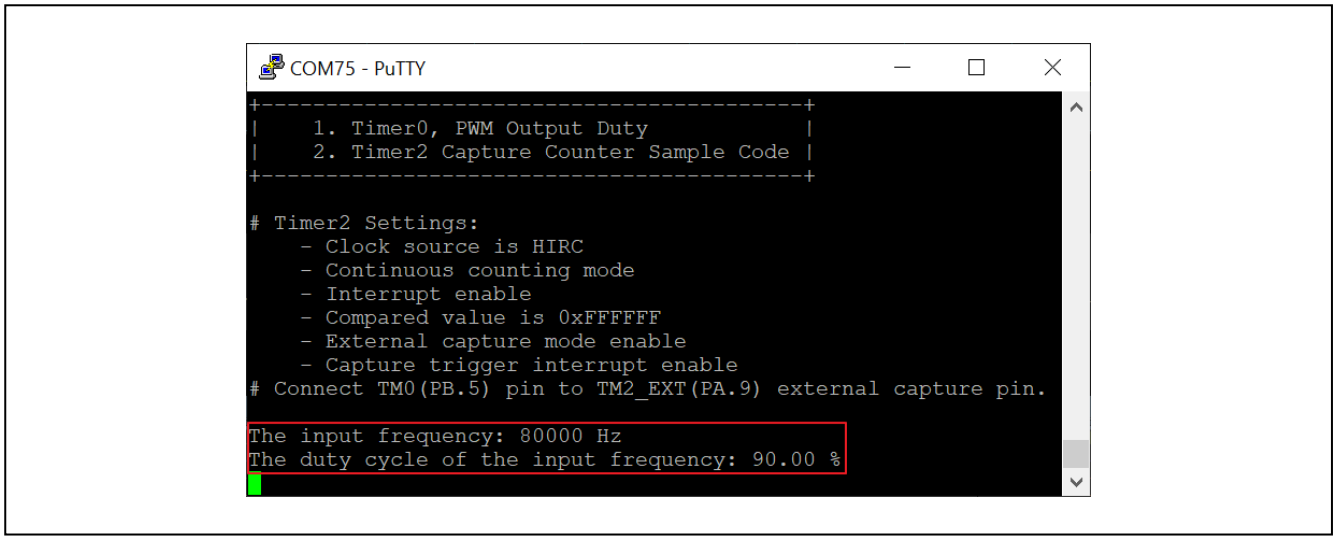


图 1-15 TIMER2 capture 抓取结果(频率 80 kHz，工作周期 90%)

# 2. 代码介绍

## 2.1 主代码说明

代码工作流程如下：

1. main.c 中初始 M251/M252/M254/M256/M258 系列相关外设，包含 HIRC、UART0、TIMER0、TIMER1、TIMER2。
2. 判断 TIMER0 设定频率、工作周期是否在要求范围内，并开启 TIMER0 的 PWM 功能、TIMER2 的 Capture 功能，以及 TIMER2 Capture 的中断功能。
3. 将 TIMER2 Capture 的中断模式设定 TIMER_CAPTURE_EVENT_GET_LOW_PERIOD，抓取到的计数值可用来计算抓取信号低周期时间，抓取后存至 g_au32TMRINTValue[ ]数组内，g_au32TMRINTValue[ ]数组的索引值 0 用来存放此计数值。
4. 将 TIMER2 Capture 的中断模式设定 TIMER_CAPTURE_EVENT_GET_HIGH_PERIOD，抓取到的计数值可用来计算抓取信号高周期时间，抓取后存至 g_au32TMRINTValue[ ]数组内，g_au32TMRINTValue[ ]数组的索引值 1 用来存放此计数值。
5. 从 g_au32TMRINTValue[ ]数组储存的计数值取出，并分别计算抓取到信号频率，以及工作周期。

主代码如下：

```
int main(void)
{
    /* Init System, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Init UART0 for printf and test */
    UART0_Init();

    printf("+-----------------------------------------+\n");
    printf("|    1. Timer0, PWM Output Duty           |\n");
    printf("|    2. Timer2 Capture Counter Sample Code |\n");
    printf("+-----------------------------------------+\n\n");
    printf("# Timer2 Settings:                         \n");
    printf("    - Clock source is PLL                  \n");
```

```
    printf("    - Continuous counting mode              \n");
    printf("    - Interrupt enable                      \n");
    printf("    - Compared value is 0xFFFFFF            \n");
    printf("    - External capture mode enable          \n");
    printf("    - Capture trigger interrupt enable      \n");
    printf("# Connect TM0(PB.5) pin to TM2_EXT(PA.9) external capture pin.\n\n");

    printf(" Please set frequency, then press the ENTER.: (Range: 20 ~ 80k Hz)\n");
    printf(" Frequency: ");

    while (1)
    {
        if(g_u8FinishSet == TRUE)
        {
            if((((int)g_u32Frequency < 20) || ((int)g_u32Frequency > 80000)) ||
                (((int)g_u32Duty < 0) || ((int)g_u32Duty > 100)))
            {
                printf(" Please check the frequency (range: 20 ~ 80k Hz) and duty cycle
                    (range: 0 ~ 100 %%) setting! \n\n");

                g_u8FinishSet = FALSE;

                printf(" Please set frequency, then press the ENTER.: \n");
                printf(" Frequency: ");
            }
            else
            {
                /* Enable Timer2 NVIC */
                NVIC_EnableIRQ(TMR2_IRQn);

                /* Change Timer to PWM counter mode */
                TPWM_ENABLE_PWM_MODE(TIMER0);

                /* Enable output of PWM_CH0 */
                TPWM_ENABLE_OUTPUT(TIMER0, TPWM_CH0);

                /* Set Timer0 PWM output frequency and duty cycle in up count type */
                if (TPWM_ConfigOutputFreqAndDuty(TIMER0, g_u32Frequency, g_u32Duty)
                    != g_u32Frequency)
                {
                printf("\n Set the frequency different from the user\n");
```

```
                }

                TPWM_START_COUNTER(TIMER0);

                TIMER_Delay(TIMER1, 500000);

                Get_PWMSignal();

                printf(" Please set frequency, then press the ENTER.: \n");
                printf(" Frequency: ");

                g_u32CaptureCount = 0;
                g_u8FinishSet = FALSE;
            }
        }
    }
}
```

Get_PWMSignal( ) 函数用来设定 TIMER2 的 Capture 方式，分别设定 Low period 与 High period 方式，并开启 Capture 中断，当 TIMER2 进入 Capture 中断，会将 capture 到的 counter 值存放至 au32TMRINTValue[ ]数组，最后透过 au32TMRINTValue[ ]数组储存的数值计算输入信号频率值，以及 Duty 值，请参考下方代码。

```
void Get_PWMSignal(void)
{
    /* Step1. Capture low period of input frequency */
    TIMER_SET_PRESCALE_VALUE(TIMER2, 0);
    TIMER_SET_CMP_VALUE(TIMER2, 0xFFFFFF);
    TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_COUNTER_RESET_MODE,
        TIMER_CAPTURE_EVENT_GET_LOW_PERIOD);
    TIMER_EnableCaptureInt(TIMER2);
    g_u32CaptureFinish = 0;

    /* Start Timer0 PWM counter and Timer2 counting*/
    TIMER_Start(TIMER2);
    while(g_u32CaptureFinish == 0);

    TIMER_StopCapture(TIMER2);
    TIMER_Stop(TIMER2);

    while (TIMER_IS_ACTIVE(TIMER2)){};
```

```
    TIMER_ClearCaptureIntFlag(TIMER2);

    /* Step2. Capture high period of input frequency */
    TIMER_SET_PRESCALE_VALUE(TIMER2, 0);
    TIMER_SET_CMP_VALUE(TIMER2, 0xFFFFFF);
    TIMER_EnableCapture(TIMER2, TIMER_CAPTURE_COUNTER_RESET_MODE,
        TIMER_CAPTURE_EVENT_GET_HIGH_PERIOD);
    TIMER_EnableCaptureInt(TIMER2);
    g_u32CaptureFinish = 0;

    /* Start Timer0 PWM counter and Timer2 counting*/
    TIMER_Start(TIMER2);
    while(g_u32CaptureFinish == 0);

    TIMER_StopCapture(TIMER2);
    TIMER_Stop(TIMER2);

    while (TIMER_IS_ACTIVE(TIMER2)){};
    TIMER_ClearCaptureIntFlag(TIMER2);

    printf("\n The input frequency: %d Hz\n", (int)PLL_CLOCK / (g_au32TMRINTValue[1] +
        g_au32TMRINTValue[0]));

    printf(" The duty cycle of the input frequency: %.2f %%\n\n\n",
        (float)(g_au32TMRINTValue[1] * 100) / (g_au32TMRINTValue[1] +
        g_au32TMRINTValue[0]));
}
```

可在程序最上方的#define定义变量与数组，用来控制TIMER2 capture产生中断时得到的计数值。

```
#define PLL_CLOCK 24000000UL
#define RX_BUFFER_SIZE   50
#define RX_TIMEOUT_CNT   50
volatile uint32_t g_au32TMRINTCount = 0, g_u32CaptureFinish = 0, g_u32CaptureCount = 0;
volatile uint32_t g_au32TMRINTValue[2] = {0};
```

TIMER2 capture 中断代码如下：

主代码中断方式分别设定 TIMER_CAPTURE_EVENT_GET_LOW_PERIOD 与 TIMER_CAPTURE_EVENT_GET_HIGH_PERIOD，并依序透过 TIMER_GetCaptureData( )功能，将得到的计数值存放至 g_au32TMRINTValue[ ] 数组内。

```
void TMR2_IRQHandler(void)
{
    if (TIMER_GetCaptureIntFlag(TIMER2))
    {
        /* Clear Timer2 capture trigger interrupt flag */
        TIMER_ClearCaptureIntFlag(TIMER2);

        if(g_au32TMRINTCount == 0)
        {
            g_au32TMRINTCount = 1;
        }
        else
        {
            g_au32TMRINTValue[g_captureCount++] = TIMER_GetCaptureData(TIMER2);

            g_au32TMRINTCount = 0;
            g_captureFinish = 1;
        }
    }
}
```

# 3. 软件与硬件需求

## 3.1 软件需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
- IDE 版本
  - Keil uVersion 5.26

## 3.2 硬件需求

- 电路组件
  - NuMaker-M251SD Ver1.1
- 示意图

M251 的 PB.5 设定为 TIMER0，并开启 PWM 功能对外输出信号，PA.9 设定为 TIMER2_EXT，并开启 Capture 功能，用来抓取由 TIMER0 产生信号的频率、工作周期。

执行范例代码前，请先将 PB.5 和 PA.9 用跳线连接，硬件外部线路连接如下：

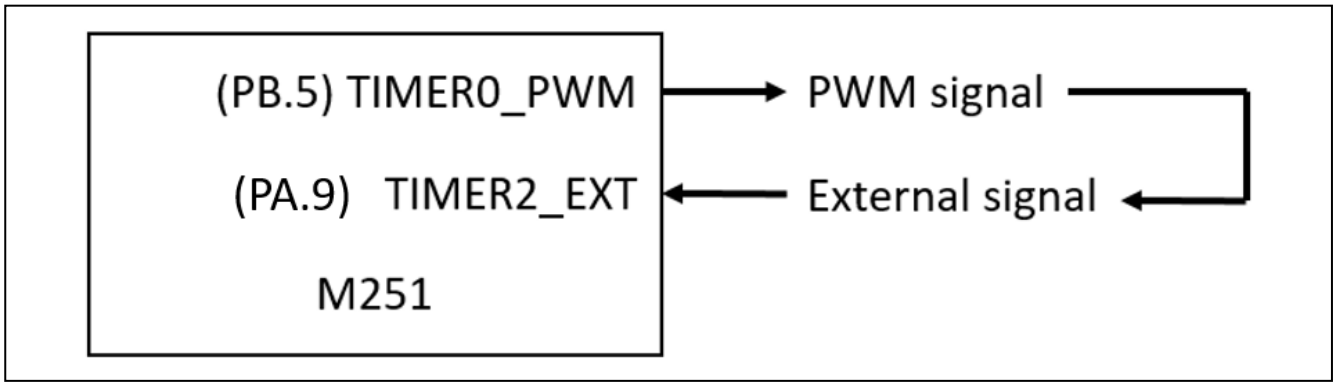


图 3-1 线路连接

# 4. 目录信息

| | |
|---|---|
| EC_M251_TIMER_Capture_Input_Signal_V1.00 | |
| Library | Sample code header and source files |
| CMSIS | Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp. |
| Device | CMSIS compliant device header file |
| StdDriver | All peripheral driver header and source files |
| Sample_Code | |
| ExampleCode | Source file of example code |

图 4-1 目录信息

# 5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 M251_TIMER_Capture_Input_Signal.uvprojx。
2. 进入编译模式界面
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

# 6. 修订纪录

| Date | Revision | Description |
|---|---|---|
| 2023.01.18 | 1.00 | 初始发布. |

## Important Notice

**Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".**

**Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.**

**All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.**

*Please note that all data and specifications are subject to change without notice.*
*All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*