透過TIMER Capture抓取輸入訊號頻率

NuMicro® 32 位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼使用 M251 TIMER Capture 功能抓取輸入訊號頻率和工作週期
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
開發平台	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

此範例代碼使用 M251、M252、M254、M256、M258 系列 TIMER2 透過 capture 功能抓取輸入訊號頻率、工作週期，以及 TIMER0 透過 PWM 功能輸出模擬訊號，可輸入訊號頻率範圍為 20 ~ 80kHz，將此兩組 TIMER 接腳互接，確認 TIMER Capture 功能抓取的頻率、工作週期，另外透過一組 UART 印出擷取結果。

1.1 原理

1.1.1 TIMER capture 功能抓取頻率計算

TIMER capture主要透過設定中斷，並根據中斷次數用以計算擷取的頻率數值，以及工作週期，TIMER capture中斷方式可選擇上升緣觸發、下降緣觸發、高準位週期時間、低準位週期時間，搭配系統時鐘頻率，即可得知擷取頻率數值。

本範例代碼將系統時鐘設定為 24MHz，因此一個Clock時間為0.042us，如圖 1-1，TIMER的一個Clock時間也會是0.042us，TIMER capture中斷方式分別設定為高準位週期時間、低準位週期時間，並分別抓取輸入訊號高準位時TIMER的計數次數，以及低準位時TIMER的計數次數，如圖 1-2，將這兩個計數次數相加後，再乘以Clock時間的0.042us，即可得知輸入訊號一個訊號的週期時間，如圖 1-3，最後再將週期時間倒數即為輸入頻率。

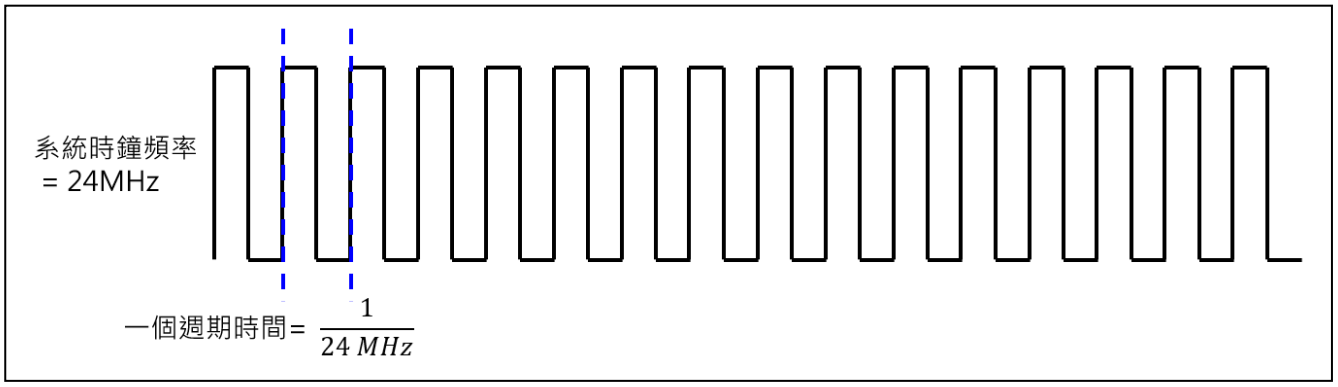

圖 1-1 系統時鐘一個週期時間

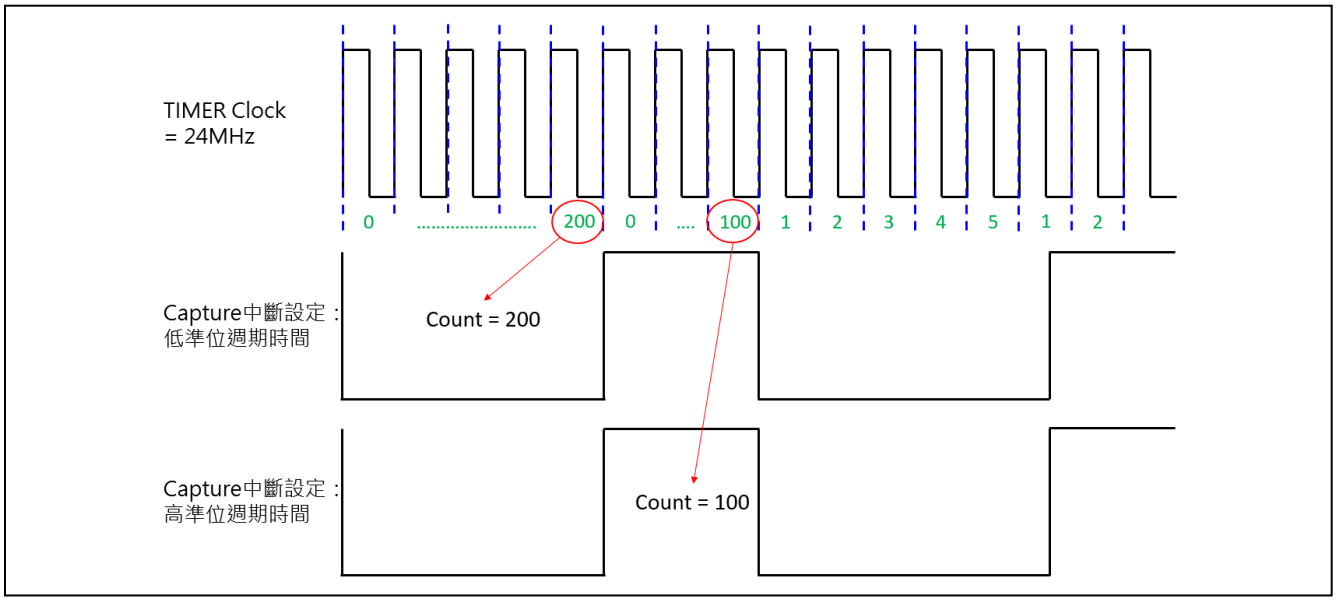

圖 1-2 TIMER capture 中斷方式

1.1.2 TIMER capture 功能抓取工作週期計算

TIMER capture中斷方式各別設定為高準位週期時間、低準位週期時間，每發生一次Capture中斷就Reset計數值，最後將高準位週期時間、低準位週期時間抓取到的計數值做計算即可得到輸入訊號的工作週期，如圖 1-3。

$$工作週期 = \frac{\text{高準位週期時間Capture計數值}}{\text{高準位週期時間Capture計數值} + \text{低準位週期時間Capture計數值}} \times 100\%$$

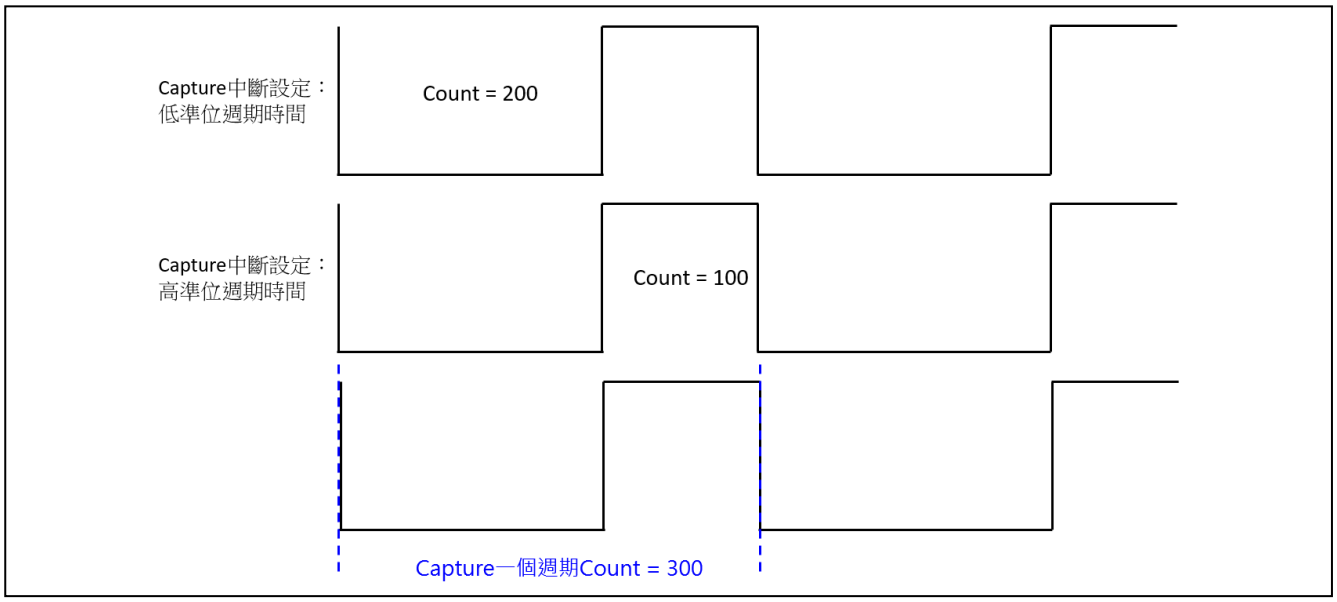

圖 1-3 TIMER capture 抓取一個訊號的週期時間計數

1.2 執行結果

TIMER0開啟PWM功能，分別輸出20 Hz、80 kHz訊號，工作週期分別設定為10%、50%、100%，TIMER2開啟Capture功能，抓取TIMER0輸出的訊號頻率、工作週期，確認計算結果是否正確，並透過終端軟體顯示結果，請參考圖 1-4 ~ 圖 1-15。

1.2.1 PWM 訊號(頻率 20 Hz，工作週期 10%)

圖 1-4 為 PWM 頻率 20 Hz，工作週期 10% 時的波形圖，圖 1-5 為 TIMER2 Capture 實際抓取訊號結果，透過終端軟體呈現。

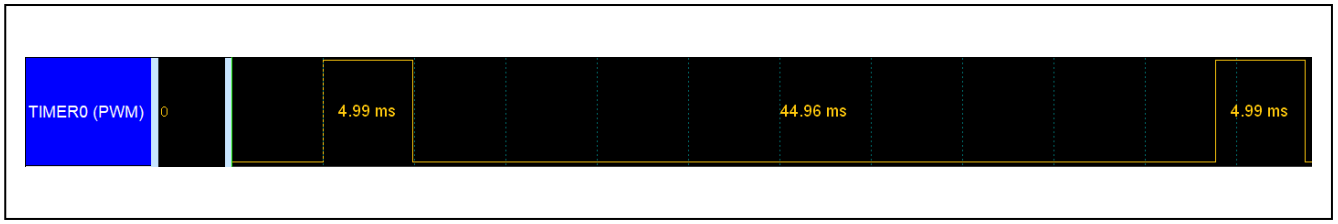

圖 1-4 TIMER0 輸出頻率 20 Hz，工作週期 10%

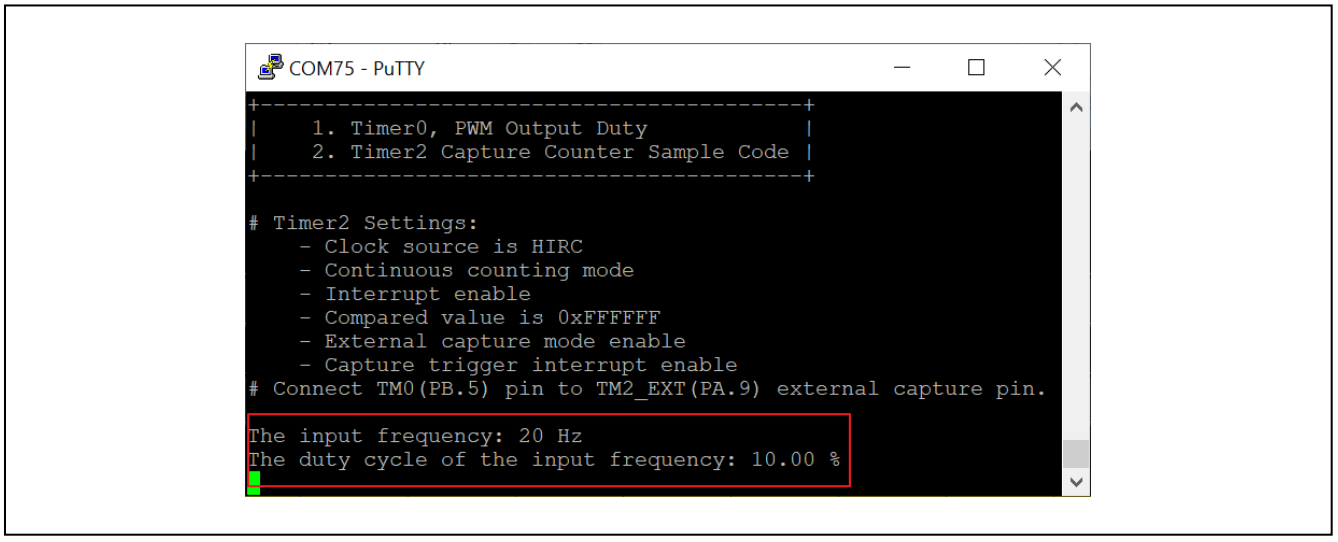

圖 1-5 TIMER2 capture 抓取結果(頻率 20 Hz，工作週期 10%)

1.2.2 PWM 訊號(頻率 20 Hz，工作週期 50%)

圖 1-6 為 PWM 頻率 20 Hz，工作週期 50% 時的波形圖，圖 1-7 為 TIMER2 Capture實際抓取訊號結果，透過終端軟體呈現。

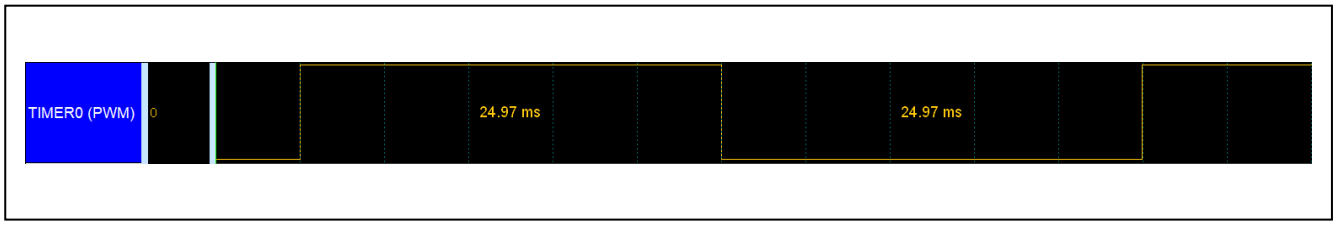

圖 1-6 TIMER0 輸出頻率 20 Hz，工作週期 50%

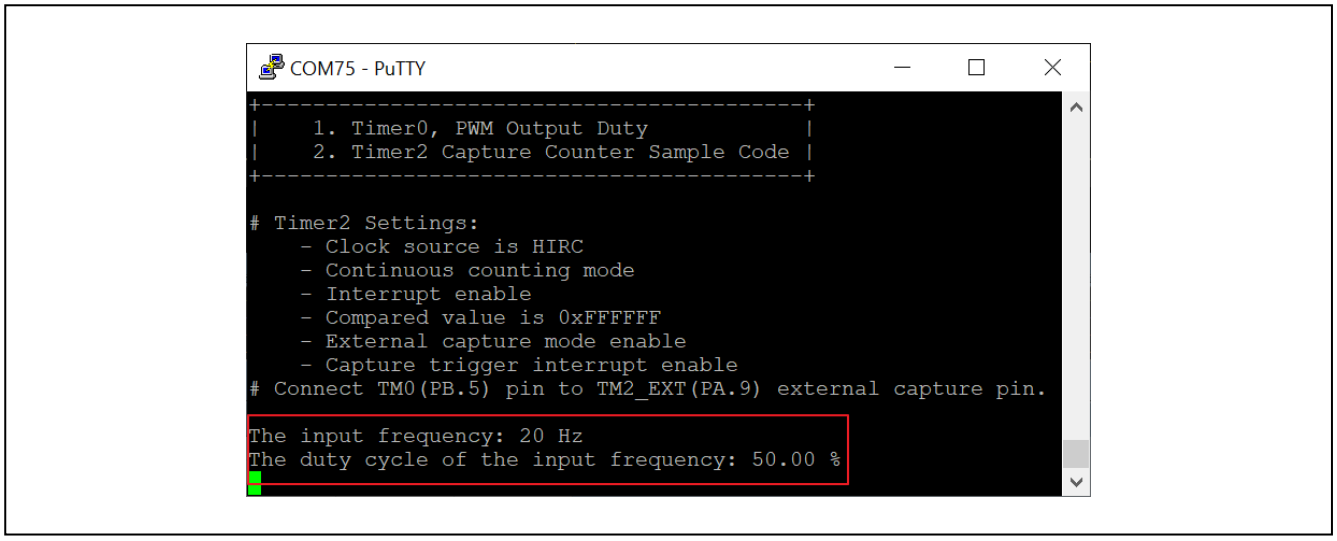

圖 1-7 TIMER2 capture 抓取結果(頻率 20 Hz，工作週期 50%)

1.2.3 PWM 訊號(頻率 20 Hz，工作週期 90%)

圖 1-8 為 PWM 頻率 20 Hz，工作週期 90% 時的波形圖，圖 1-9 為 TIMER2 Capture實際抓取訊號結果，透過終端軟體呈現。

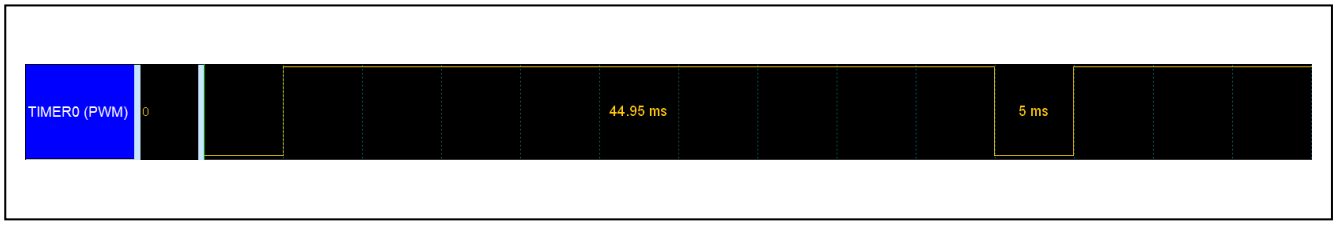

圖 1-8 TIMER0 輸出頻率 20 Hz，工作週期 90%

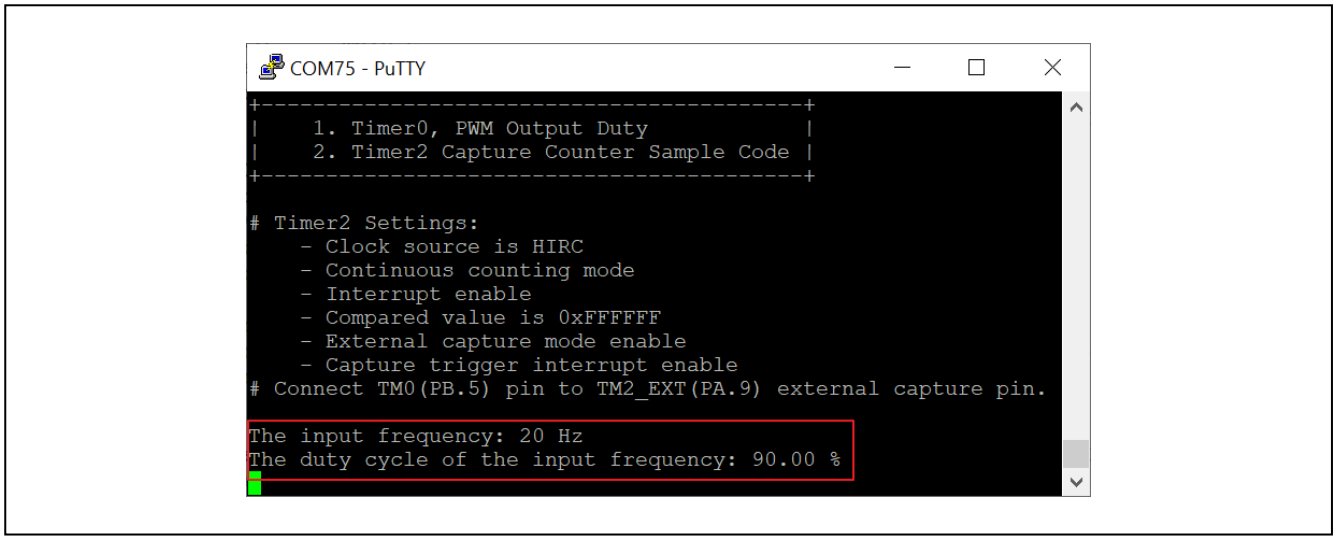

圖 1-9 TIMER2 capture抓取結果(頻率20 Hz，工作週期90%)

1.2.4 PWM 訊號(頻率 80 kHz，工作週期 10%)

圖 1-10 為 PWM 頻率 80 kHz，工作週期 10% 時的波形圖，圖 1-11 為 TIMER2 Capture實際抓取訊號結果，透過終端軟體呈現。

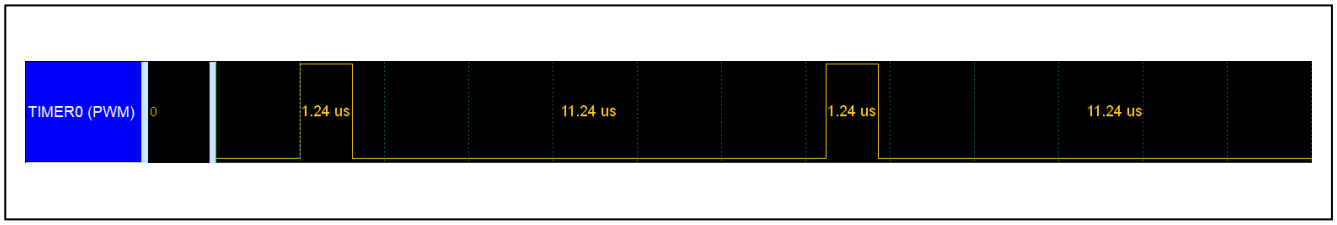

圖 1-10 TIMER0 輸出頻率 80 kHz，工作週期 10%

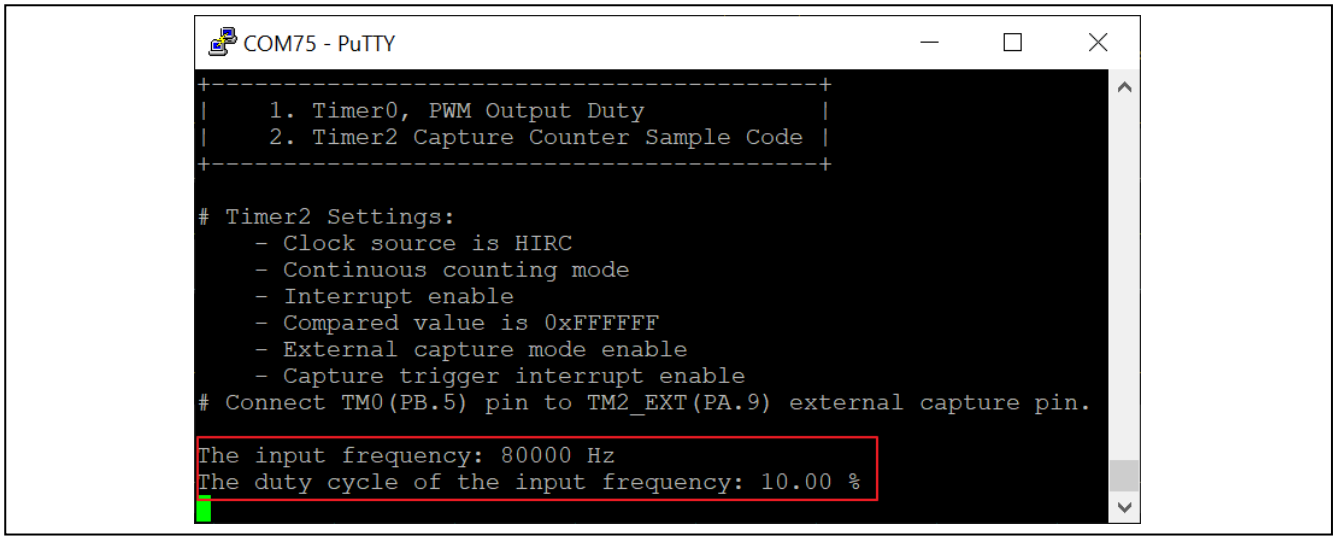

圖 1-11 TIMER2 capture抓取結果(頻率80 kHz，工作週期10%)

1.2.5 PWM 訊號(頻率 80kHz，工作週期 50%)

圖 1-12 為 PWM頻率 80 kHz，工作週期 50% 時的波形圖，圖 1-13 為 TIMER2 Capture實際抓取訊號結果，透過終端軟體呈現。

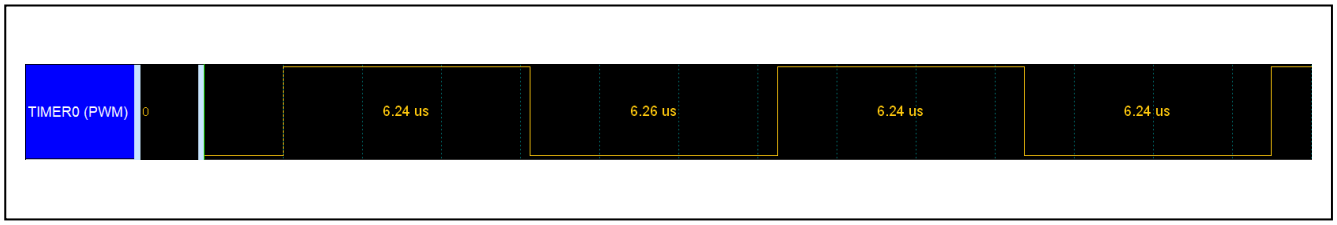

圖 1-12 TIMER0 輸出頻率 80 kHz，工作週期 50%

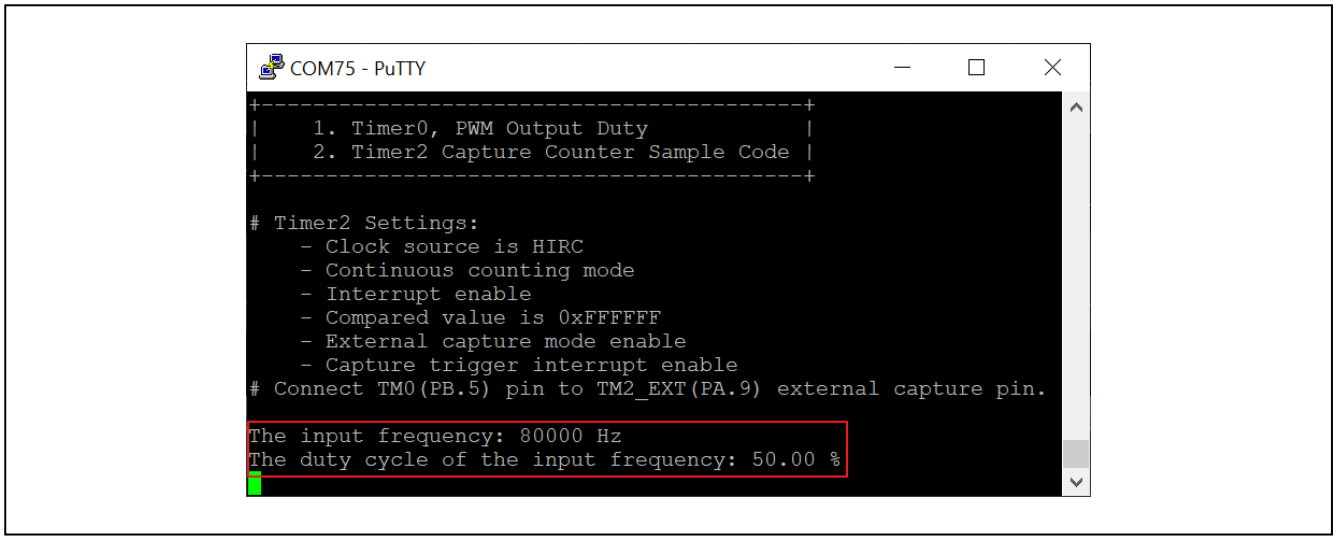

圖 1-13 TIMER2 capture 抓取結果(頻率 80 kHz，工作週期 50%)

1.2.6 PWM 訊號(頻率 80 kHz，工作週期 90%)

圖 1-14 為 PWM頻率 80 kHz，工作週期 90% 時的波形圖，圖 1-15 為 TIMER2 Capture實際抓取訊號結果，透過終端軟體呈現。

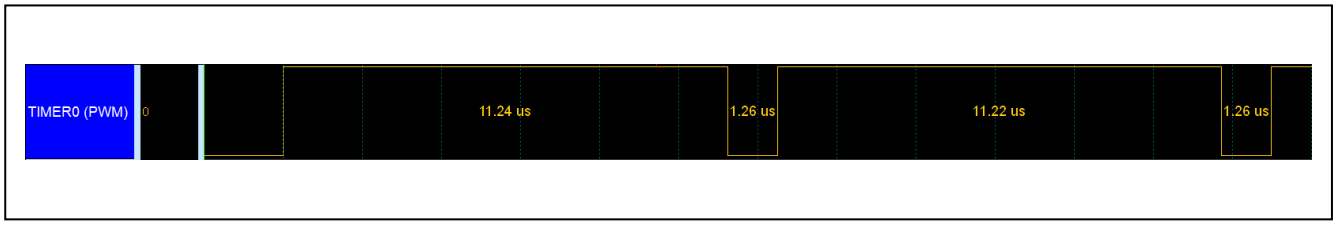

圖 1-14 TIMER0 輸出頻率 80 kHz，工作週期 90%

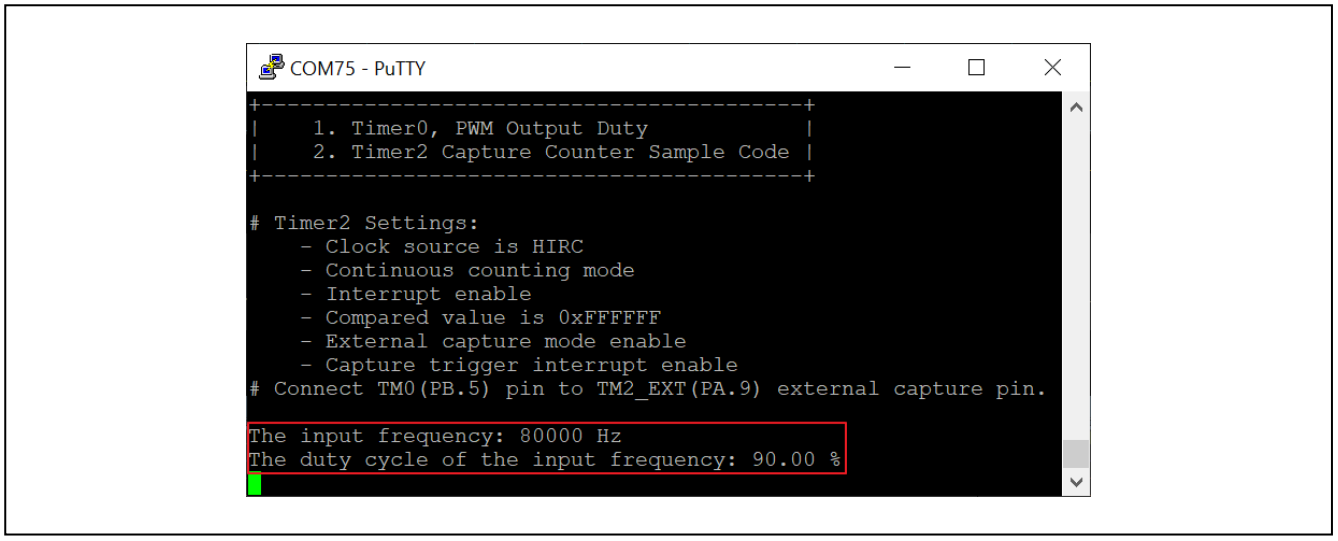

圖 1-15 TIMER2 capture 抓取結果(頻率 80 kHz，工作週期 90%)

2. 代碼介紹

2.1 主代碼說明

代碼工作流程如下：

1. main.c 中初始 M251/M252/M254/M256/M258 系列相關外設，包含 HIRC、UART0、TIMER0、TIMER1、TIMER2。
2. 判斷 TIMER0 設定頻率、工作週期是否在要求範圍內，並開啟 TIMER0 的 PWM 功能、TIMER2 的 Capture 功能，以及 TIMER2 Capture 的中斷功能。
3. 將 TIMER2 Capture 的中斷模式設定 TIMER_CAPTURE_EVENT_GET_LOW_PERIOD，抓取到的計數值可用來計算抓取訊號低週期時間，抓取後存至 g_au32TMRINTValue[ ]陣列內，g_au32TMRINTValue[ ]陣列的索引值 0 用來存放此計數值。
4. 將 TIMER2 Capture 的中斷模式設定 TIMER_CAPTURE_EVENT_GET_HIGH_PERIOD，抓取到的計數值可用來計算抓取訊號高週期時間，抓取後存至 g_au32TMRINTValue[ ]陣列內，g_au32TMRINTValue[ ]陣列的索引值 1 用來存放此計數值。
5. 從 g_au32TMRINTValue[ ]陣列儲存的計數值取出，並分別計算擷取到訊號頻率，以及工作週期。

主代碼如下：

```
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( ) 函數用來設定 TIMER2 的 Capture 方式，分別設定 Low period 與 High period 方式，並開啟 Capture 中斷，當 TIMER2 進入 Capture 中斷，會將 capture 到的 counter 值存放至 au32TMRINTValue[ ]陣列，最後透過 au32TMRINTValue[ ]陣列儲存的數值計算輸入訊號頻率值，以及 Duty 值，請參考下方代碼。

```
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]));
}
```

可在程式最上方的#define 定義變數與陣列，用來控制 TIMER2 capture 產生中斷時得到的計數值。

```
#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 中斷代碼如下：

主代碼中斷方式分別設定為 TIMER_CAPTURE_EVENT_GET_LOW_PERIOD 與 TIMER_CAPTURE_EVENT_GET_HIGH_PERIOD，並依序透過 TIMER_GetCaptureData( )功能，將得到的計數值存放至 g_au32TMRINTValue[ ] 陣列內。

```
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. 軟體與硬體需求

3.1 軟體需求

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

3.2 硬體需求

- 電路元件
  - NuMaker-M251SD Ver1.1
- 示意圖

M251 的 PB.5 設定為 TIMER0，並開啟 PWM 功能對外輸出訊號，PA.9 設定為 TIMER2_EXT，並開啟 Capture 功能，用來抓取由 TIMER0 產生訊號的頻率、工作週期。

執行範例代碼前，請先將 PB.5 和 PA.9 用跳線連接，硬體外部線路連接如下：

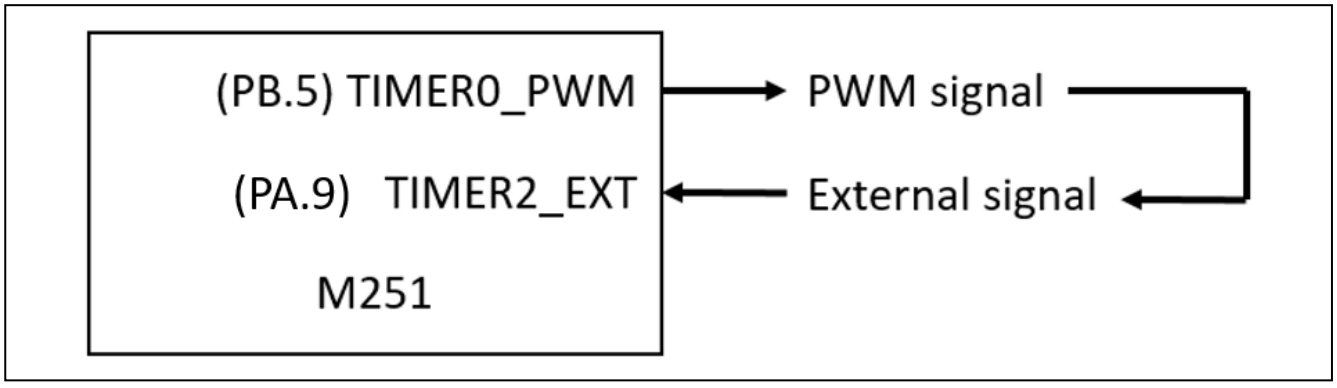

圖 3-1 線路連接

4. 目錄資訊

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
SampleCode	
ExampleCode	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 M251_TIMER_Capture_Input_Signal.uvprojx。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2023.01.18	1.00	初始發布.

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.