Pulse Counting in Power-down Mode

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses TIMER0 to count the number of pulses, and uses RTC to wake up the microcontroller (MCU) and output the number of pulses.
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
Hardware	NuMaker_M251KG_V1.1

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

5. Overview

This example code starts by setting up the RTC and TIMER0, then switches the microcontroller (MCU) to Power-down mode. TIMER0 will count the pulses on PB5 until the RTC wakes up the MCU, at which point the counting will stop and the number of pulses will be printed.

5.1 Principle

The flow chart in Figure 5-1 can help you implement pulse counting in Power-down mode.

1. System Initialization

Ensure that HIRC, LXT, UART0, RTC, and TIMER0 are enabled, and set PB5 as the pulse counting input pin.

2. TIMER0 Setting

Enable the event counter mode of TIMER0 and start counting pulses.

3. RTC Setting

Set the current RTC time and configure the wake-up time for the MCU. Enable the RTC interrupt.

4. Power-down Mode

Put the MCU into Power-down mode and wait for the RTC to wake it up. In this Power-down mode, most clocks are disabled, with only the LIRC and LXT remaining active. Additionally, the peripheral clocks are configured to use LIRC or LXT, such as RTC, WDT, Timer, and UART, and remain active to ensure their proper operation in Low-power mode.

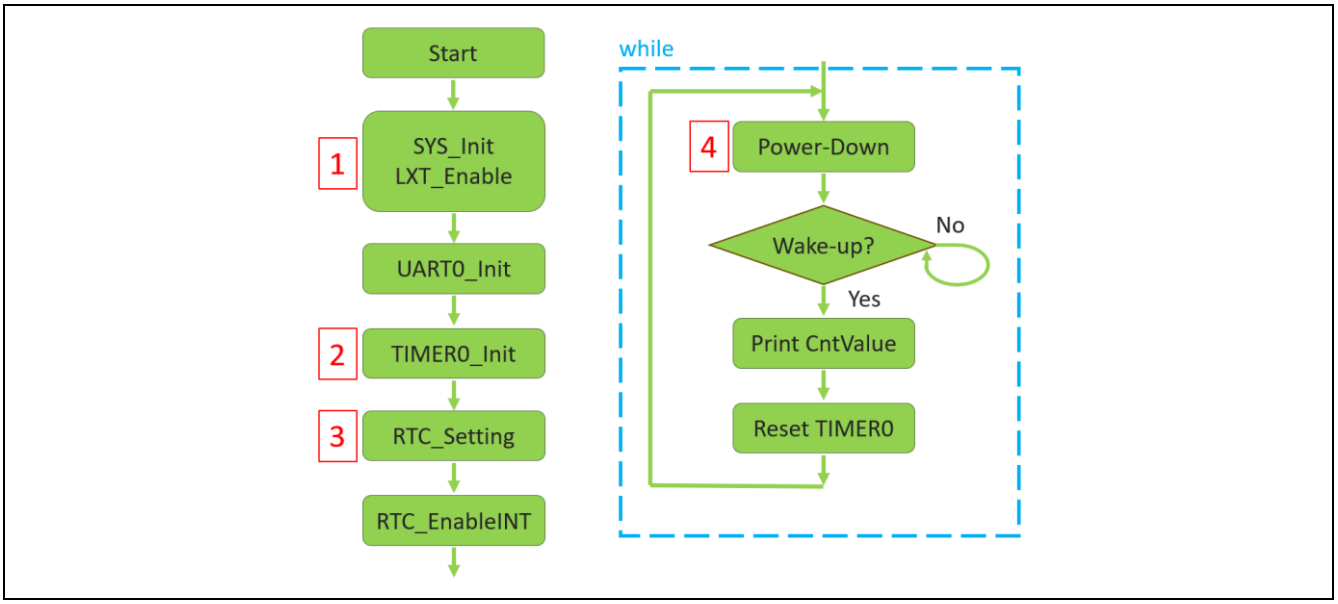

Figure 5-1 Example Code Flow Chart

5.2 Demo Result

The maximum input pulse frequency should be 1/3 of the LIRC frequency. If TMx de-bounce is enabled (CNTDBEN (TIMERx_EXTCTL[7]) = 1), the maximum input pulse frequency should be 1/8 of the LIRC frequency. However, for stability and safety, it is recommended that the maximum input pulse frequency be 1/10 of the LIRC frequency.

Execute this sample code after connecting the 200 Hz square wave generator signal to the PB5 pin and the GND pin. The execution results are shown in Figure 5-2 and Figure 5-3.

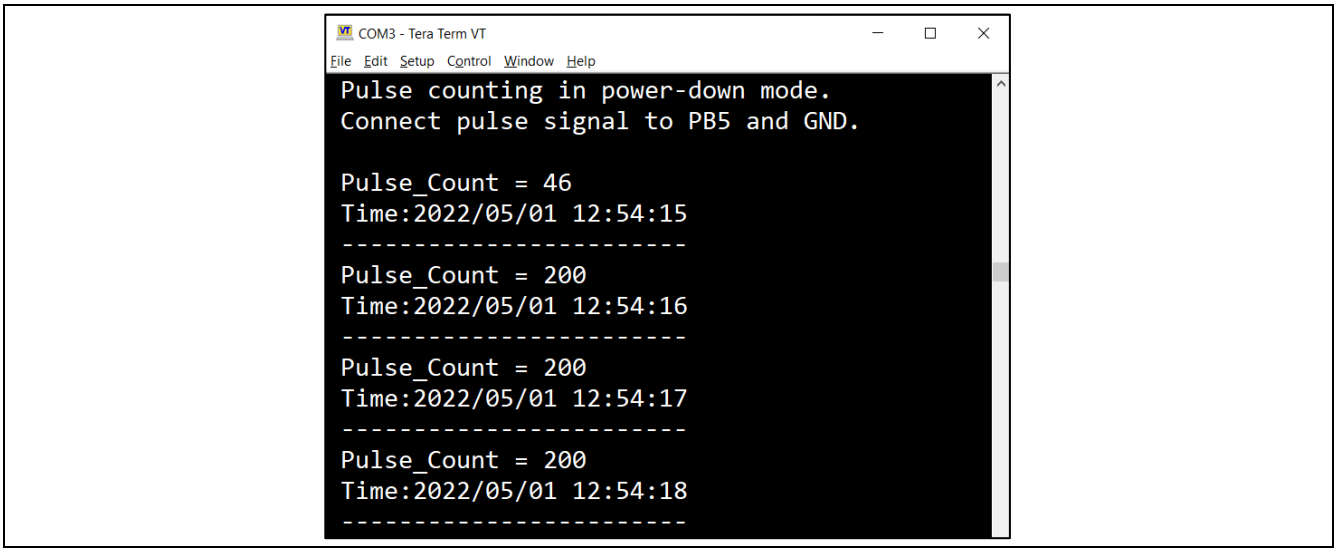

Figure 5-2 Execution Results

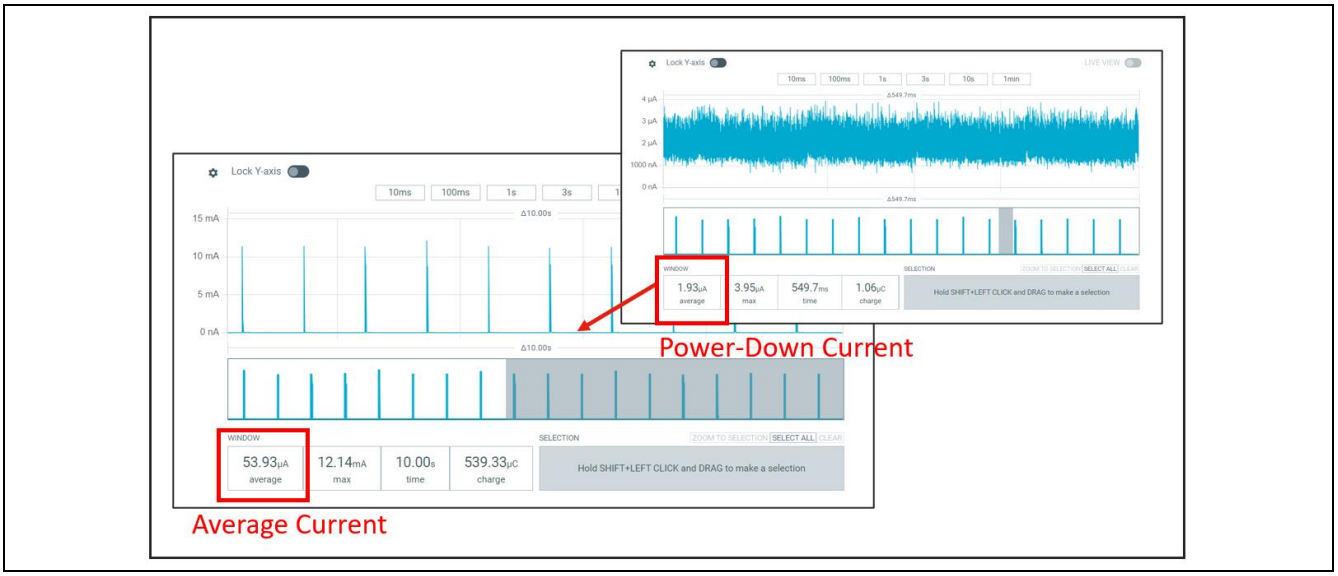

Figure 5-3 Power-down Current

6. Code Description

Initialize the system frequency, TIMER0, and UART0. Set the RTC's current date and time, and then configure the MCU wake-up time. Enable the RTC's interrupt execution function.

```
int32_t main(void)
{
    /* Init System, peripheral clock */
    SYS_Init();

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init GPIO */
    Set_All_GPIO_to_Quasi_Mode();

    /* Disable POR */
    SYS_DISABLE_POR();

    /* Disable LIRC */
    CLK_DisableXtalRC(CLK_PWRCTL_LIRCEN_Msk);
    TIMER_Delay(TIMER0, 100000);

    /* Init TIMER0 */
    TIMER0_Init();

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

    /* Lock protected registers */
    SYS_LockReg();

    printf(" Pulse counting in power-down mode.\n");
    printf(" Connect pulse signal to PB5 and GND.\n\n");

    /* Time setting */
    sInitTime.u32Year       = 2022;
    sInitTime.u32Month      = 5;
    sInitTime.u32Day        = 1;
    sInitTime.u32Hour       = 12;
    sInitTime.u32Minute     = 0;
    sInitTime.u32Second     = 0;
    sInitTime.u32DayOfWeek  = RTC_TUESDAY;
    sInitTime.u32TimeScale  = RTC_CLOCK_24;

    /* check rtc reset status */
    sInitTime_ptr = (RTC->INIT & RTC_INIT_ACTIVE_Msk) ? NULL : &sInitTime;

    if (RTC_Open(sInitTime_ptr) != 0)
    {
        printf("\n RTC initial fail!!");
        printf("\n Please check h/w setting!!");

        while (1);
```

```
    }

    /* Get the current time */
    RTC_GetDateAndTime(&sCurTime);

    /* Wait the UART FIFO buffer is empty */
    while (!IsDebugFifoEmpty()) {};

    /* Enable RTC Alarm Interrupt */
    RTC_EnableInt(RTC_INTEN_ALMIEN_Msk);
    NVIC_EnableIRQ(RTC_IRQn);
    NVIC_EnableIRQ(PWRWU_IRQn);
```

After entering the while loop, first let the MCU enter Low-power mode (Power-down mode) and wait for the real-time clock (RTC) to wake up the MCU. When the RTC wakes up the MCU and outputs the number of pulses, reset the count of TIMER0 and set the next wake-up time. After completing a counting process, return to the while loop to start a new cycle. In this way, the MCU will continue the cycle of counting and low-power management.

```
    while (1)
    {
        /* Set the alarm time for wake-up */
        Set_Alarm_time(User_Config.Wakeup_AlarmTime);

        g_i32Alarm = FALSE;

        /* Select mode to enter power down */
        EnterToPowerDown(User_Config.Power_Down_Mode);

        /* IOCTLSEL will automatically be set by hardware to 1 when system power is off
           and RTC_GPIOCTL0 had been configured. */
        /* when system wake up from PD mode, you need clear this bit if you want PF.4, 5,
           6 pin I/O function is controlled by GPIO module. */
        RTC->LXTCTL &= ~(RTC_LXTCTL_IOCTLSEL_Msk);

        /* Wait RTC interrupt occurred*/
        while (!g_i32Alarm) {};

        /* Get the current time */
        RTC_GetDateAndTime(&sCurTime);

        printf(" Time:%u/%02u/%02u %02u:%02u:%02u\n", sCurTime.u32Year,
                 sCurTime.u32Month, sCurTime.u32Day, sCurTime.u32Hour, sCurTime.u32Minute,
                 sCurTime.u32Second);
        printf(" ------------------------\n");

        /* Wait the UART FIFO buffer is empty */
        while (!IsDebugFifoEmpty()) {};
    }
}
```

Once the RTC set time is reached, the RTC interrupt service routine is executed, the pulse count calculated by TIMER0 is read, and it is output to the computer through UART0.

```
void RTC_IRQHandler(void)
{
    uint32_t Pulse_Count;

    /* alarm interrupt occurred */
    if ((RTC->INTEN & RTC_INTEN_ALMIEN_Msk) && RTC_GET_ALARM_INT_FLAG())
    {
         g_i32Alarm = TRUE;

        RTC_CLEAR_ALARM_INT_FLAG();

         /* Select HIRC as the clock source of Timer0 */
        Pulse_Count = TIMER_GetCounter(TIMER0);
         TIMER_ResetCounter(TIMER0);
        printf(" Pulse_Count = %d\n", Pulse_Count);
    }
}
```

7. Software and Hardware Requirements

7.1 Software Requirements

- BSP version
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
- IDE version
  - Keil uVersion 5.36

7.2 Hardware Requirements

- Circuit components
  - NuMaker_M251KG_V1.1
- Pin Connect
  - Connect the pulse signal from the function generator to the PB5 pin of the NuMaker-M251KG, and connect the USB to the computer as Figure 7-1.

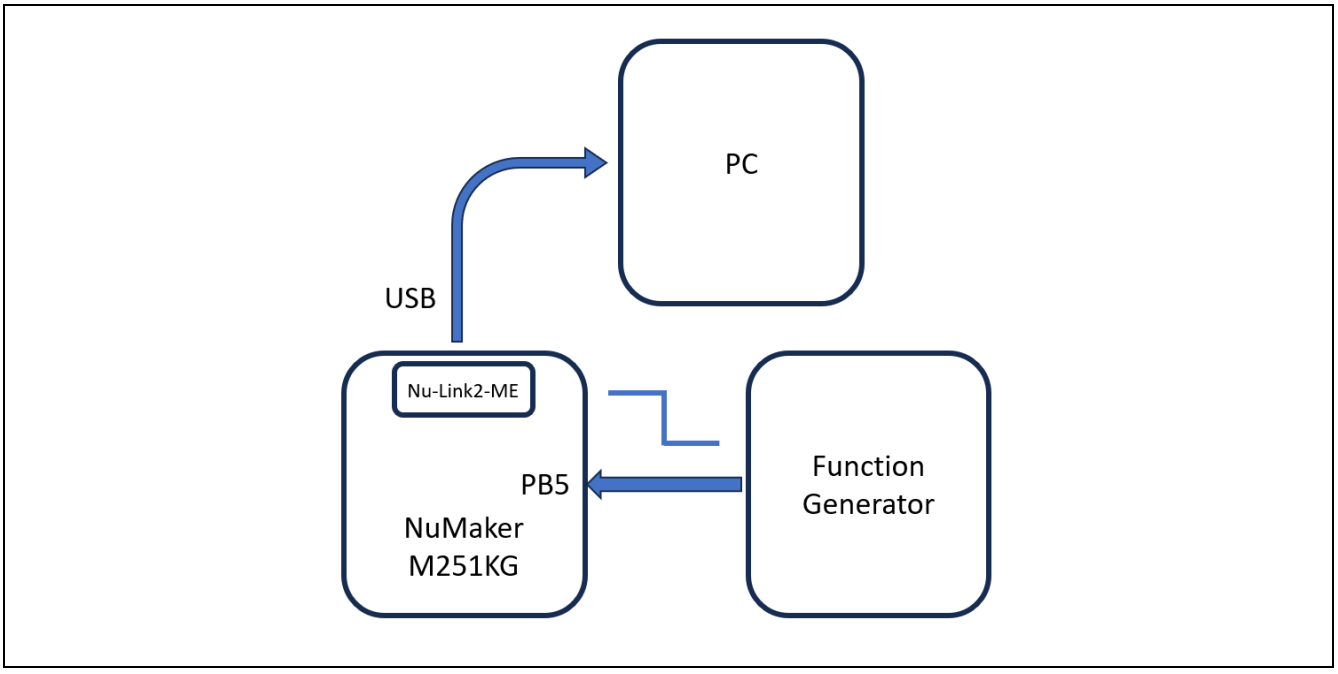

Figure 7-1 Pin Connect

8. Directory Information

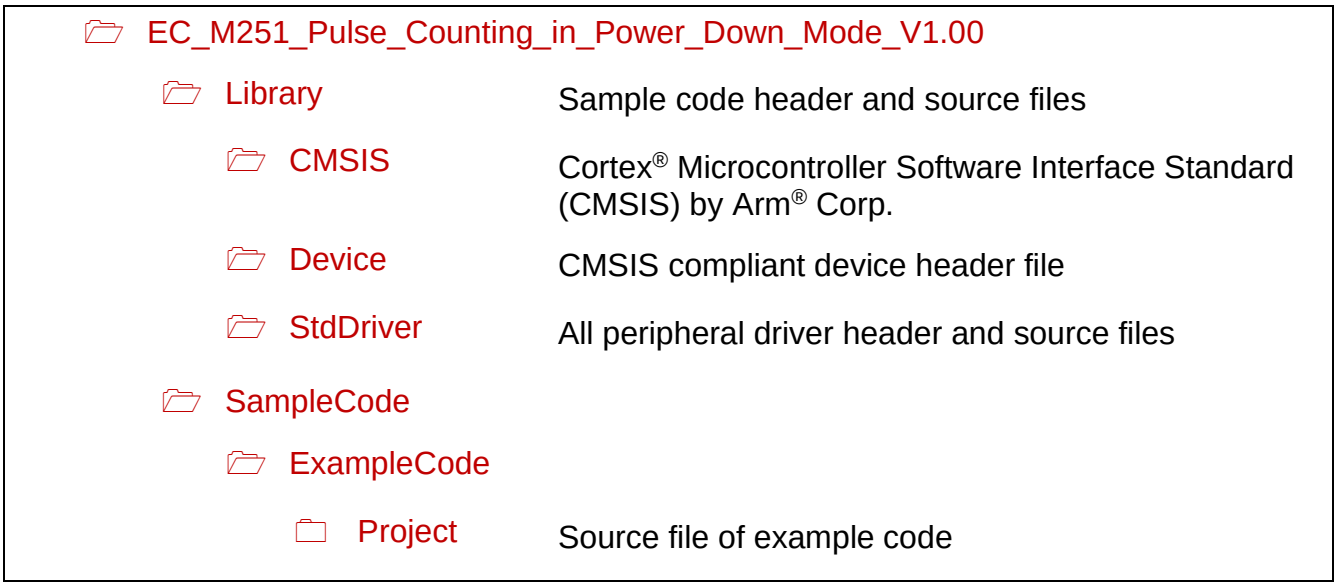

Figure 4-1 Directory Structure

9. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *Pulse_Counting_in_Power_Down_Mode.uvprojx*.
2. Enter Keil compile mode.

- Build
- Download
- Start/Stop debug session

3. Enter debug mode.

- Run

4. HyperTerminal will display as Figure 9-1.

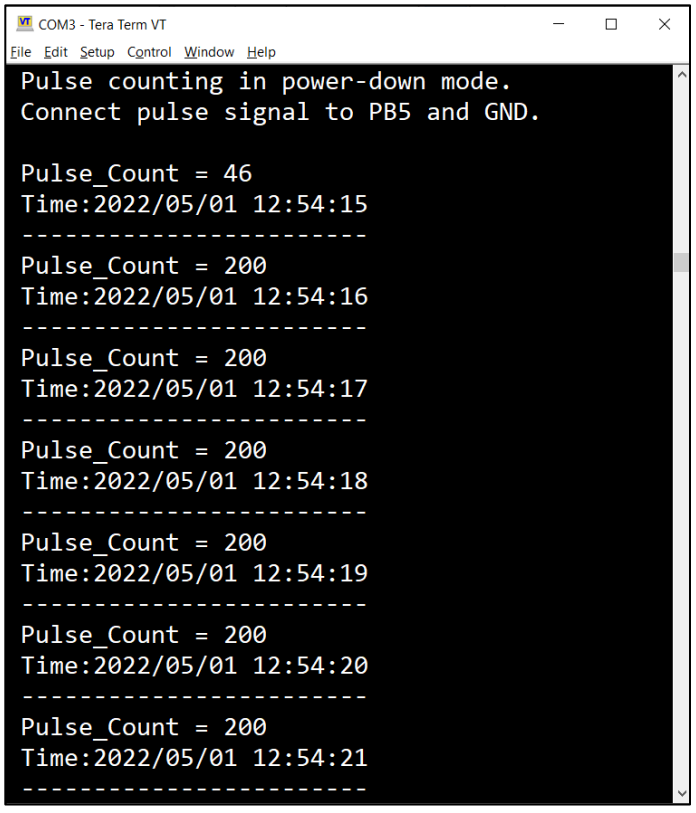

Figure 9-1 Display Count Number

10. Revision History

Date	Revision	Description
2024.06.13	1.00	Initial version.

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.