低功耗模式下的脉冲计数

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序使用 TIMER0 来计数脉冲的数量，并使用 RTC 来唤醒 MCU。当 MCU 被唤醒后，它会输出计数到的脉冲数量。
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.006
开发平台	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

1. 概述

本范例代码先设定了 RTC 和 TIMER0，然后将 MCU 切换到低功耗模式(Power-down mode)。TIMER0 会开始计算 PB5 上的脉冲信号，直到 RTC 唤醒 MCU 后，停止计数并且将脉冲的数量输出到计算机。

1.1 原理

图 1-1 的流程图可以帮助你在低功耗模式下实现脉冲计数。

1. 系统初始化

   确保 HIRC、LXT、UART0、RTC 和 TIMER0 都已经启用，并将 PB5 设定为脉冲计数输入引脚。

2. TIMER0 设定

   启用 TIMER0 的事件计数模式，并且开始计数脉冲。

3. RTC 设定

   设定当前 RTC 时间，并且设定 MCU 的唤醒时间后，启用 RTC 中断。

4. 低功耗模式

将 MCU 置于掉电模式，并等待 RTC 唤醒。在这种掉电模式下，大多数时钟都会被禁用，仅保留 LIRC 和 LXT 处于活动状态。此外，配置为使用 LIRC 或 LXT 的外围设备时钟（如 RTC、WDT、计时器和 UART）将保持活动状态，以确保它们在低功耗模式下正常运行。

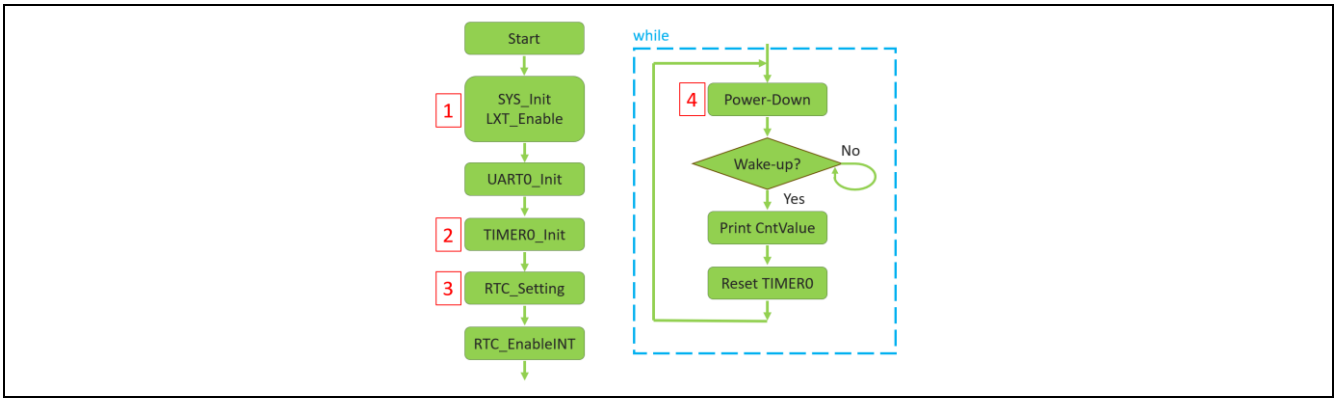

图 1-1 程序流程图

1.2 执行结果

最大输入脉冲频率应该是 LIRC 频率的三分之一。如果启用了 TMx 除弹跳功能（CNTDBEN (TIMERx_EXTCTL[7]) = 1），则最大输入脉冲频率应该是 LIRC 频率的八分之一。然而，为了稳定性和安全性，我们建议最大输入脉冲频率为 LIRC 频率的十分之一。

将 200 Hz 方波发生器信号连接到 PB5 接脚和 GND 接脚后执行此范例程序。执行结果可以参见在图 1-2 和图 1-3。

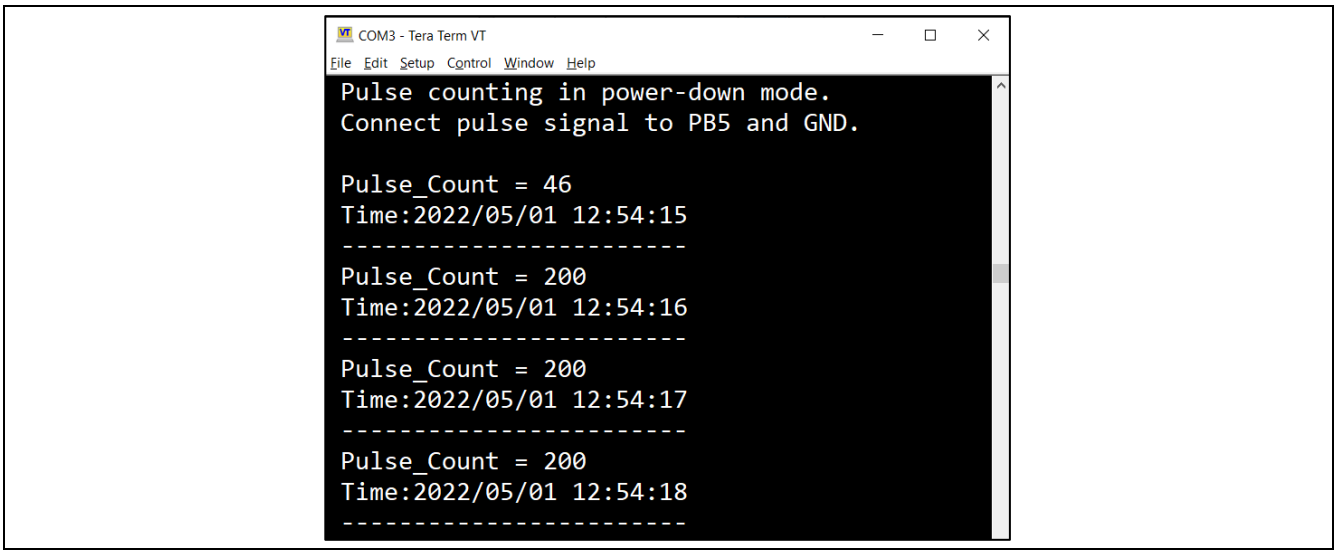

图 1-2 执行结果

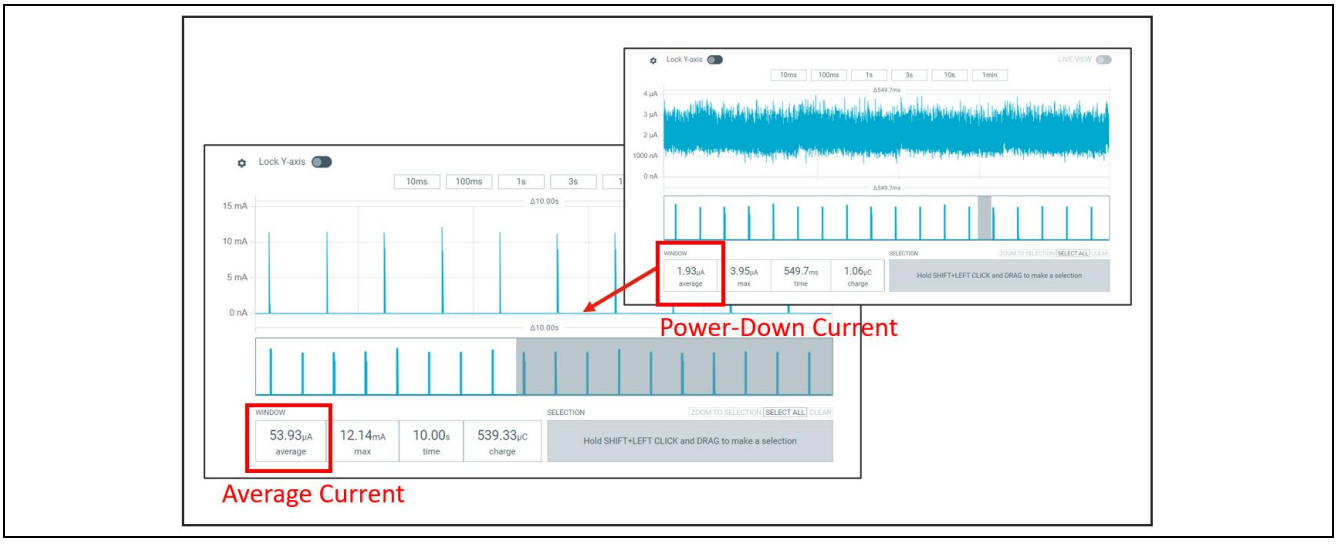

图 1-3 消耗电流

2. 代码介绍

初始化系统频率、TIMER0 和 UART0 后。再设定 RTC 的现在日期和时间，然后设定 MCU 的唤醒时间。最后启用 RTC 的中断执行功能。

```
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);
```

进入 while 循环后，先让 MCU 进入低功耗模式（power-down mode），并等待实时时钟（RTC）唤醒 MCU。当 RTC 唤醒 MCU 并输出脉冲数量后，重置 TIMER0 的计数并设定下一次的唤醒时间。完成一次计数后，回到 while 循环以开始新的循环。这样，MCU 将持续进行计数和低功耗管理的循环。

```
    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()) {};
    }
}
```

一旦达到 RTC 设定的时间，会执行 RTC 中断服务程序，读取由 TIMER0 计算的脉冲数，并透过 UART0 将其输出到计算机。

```
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);
    }
}
```

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_V3.02.006
- IDE 版本
  - Keil uVersion 5.36

3.2 硬件需求

- 电路组件
  - NuMaker_M251KG_V1.1
- 线路示意图
  - 将信号发生器的脉冲信号连接至 NuMaker-M251KG 的 PB5 引脚，并将 USB 连接至计算机如图 3-1。

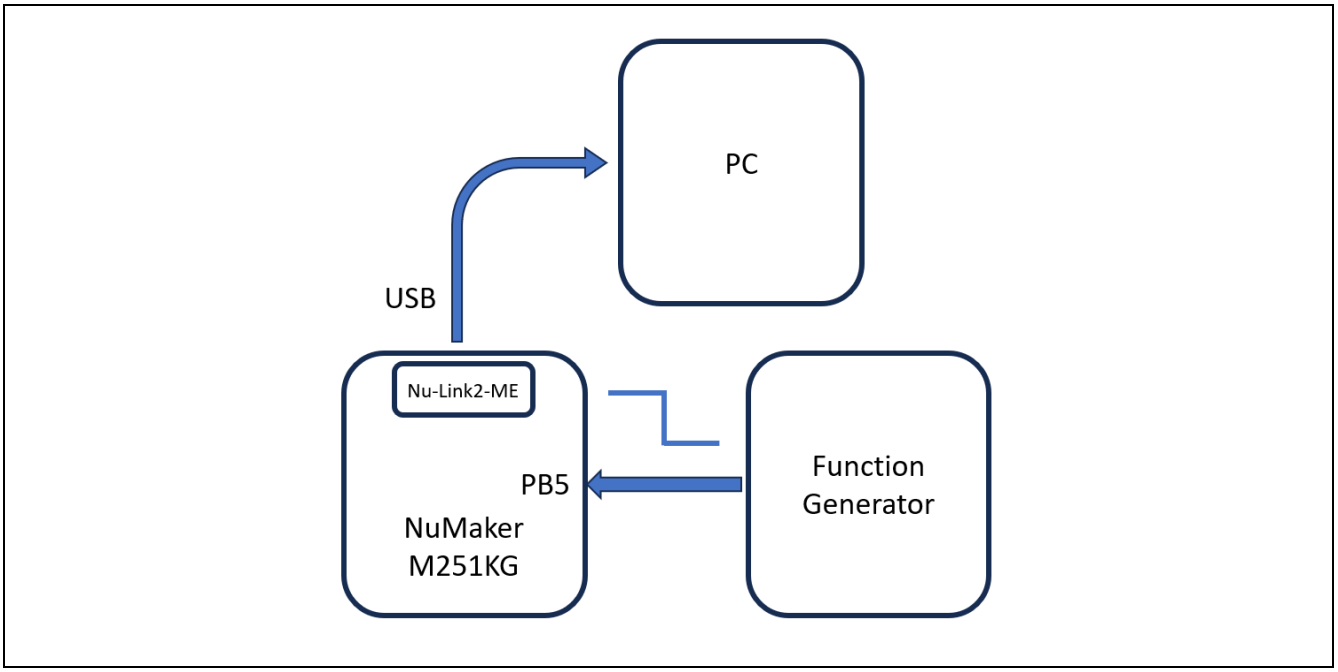

图 3-1 线路示意图

4. 目录信息

EC_M251_Pulse_Counting_in_Power_Down_Mode_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	
Project	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 *Pulse_Counting_in_Power_Down_Mode.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
4. 超级终端机画面如图 5-1

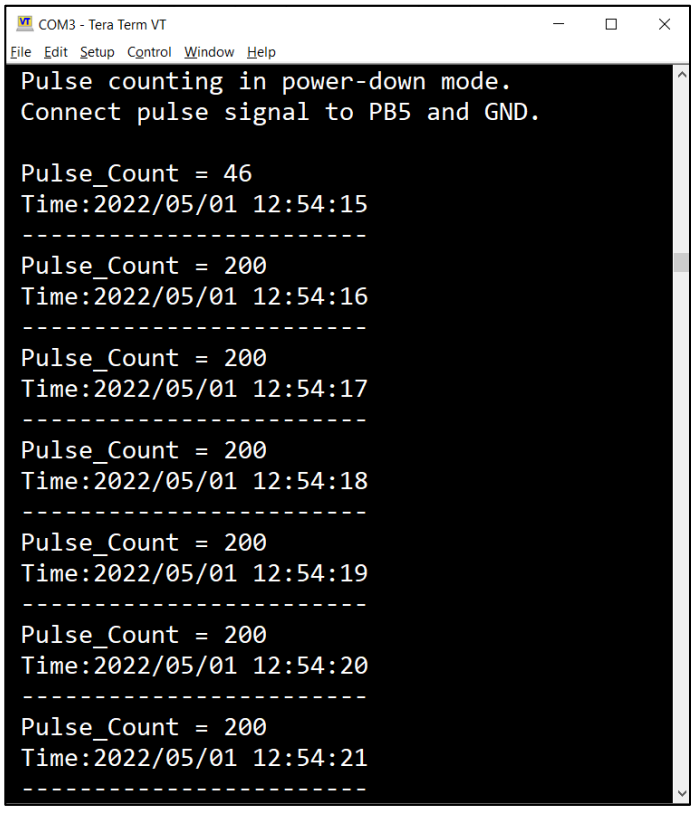

图 5-1 显示计数数量

6. 修订纪录

Date	Revision	Description
2024.06.13	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.