定时器触发 PDMA 驱动段码 LED 显示

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码实现当 M032 正处于更新 Flash 的时候，仍可通过 Timer0 触发 PDMA 驱动两位 Segment LED 显示。
BSP 版本	M031_Series_BSP_CMSIS_V3.07.000
开发平台	NuMaker-M032SE V1.3 NuEdu-Basic01 V2.0

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

本范例代码基于 M032 实现即使在更新 Flash 数据的同时，也可以正常驱动两位数码管显示。

市场上某些产品，在应用程序运行的过程需要更新 Flash 数据，如果更新的数据量大或更新频繁，就会导致其他程序功能处于阻塞状态。例如应用程序需要更新 Flash 还需要驱动 Segment LED 显示，当更新 Flash 时间超过 20 ms，就会导致显示闪烁，因为 LED 刷新间隔超过 20 ms。此参考程序就可以解决此问题。

1.1 原理

每个段码数码管有 1 个位选引脚，当位选引脚输出高电平（或低电平），数码管点亮；反之则熄灭。此参考设计位选引脚为高电平有效，即高电平点亮数码管。

每个段码数码管通常由 8 个发光二极管（A~G 和 DOT）组成，通过控制每个二极管引脚的电平实现亮灭。此参考设计二极管引脚为低电平有效，即低电平点亮相应段。

两位段码数码管及控制原理如图 1-1。

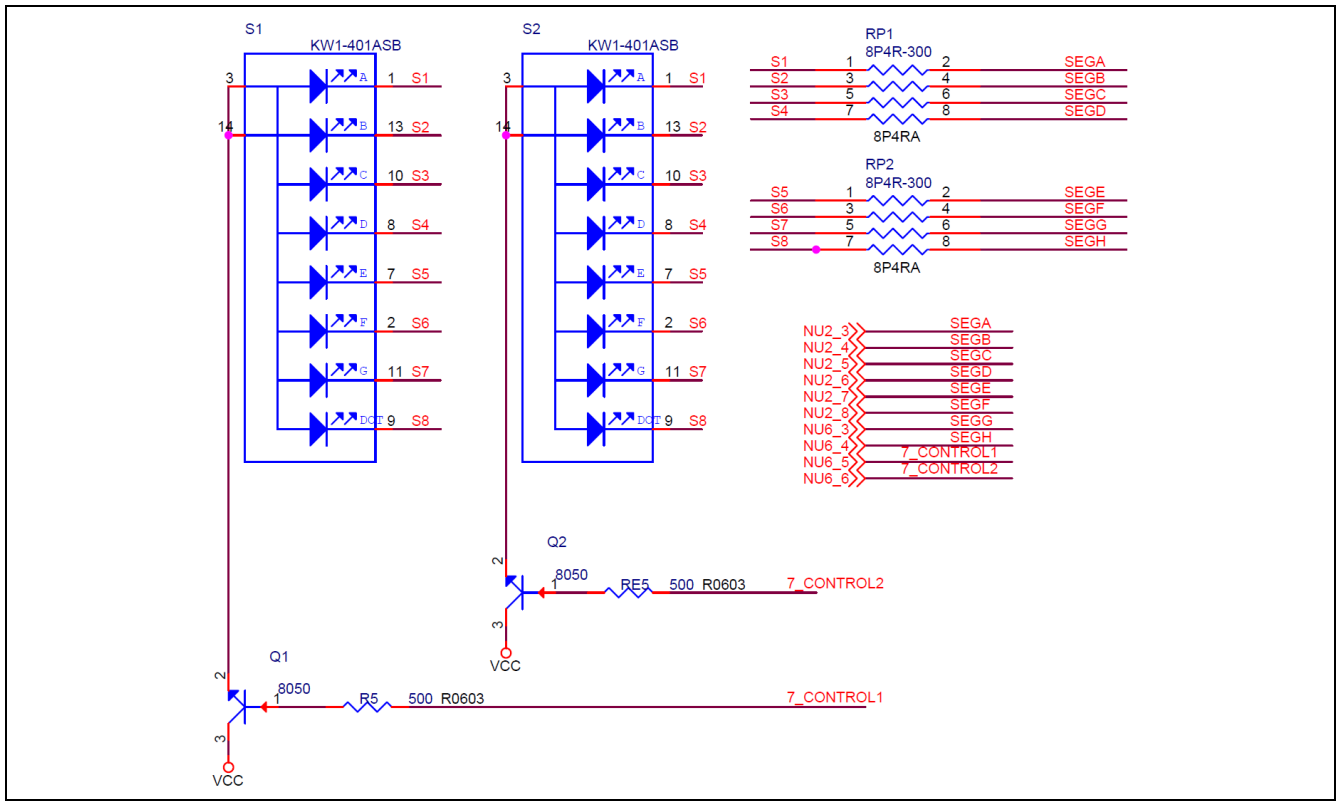

图 1-1 两位段码数码管及控制原理

通过分时扫描两位数码管，利用人眼视觉暂留效应实现连续显示。

若 CPU 要读取的执行指令和正更新的 Flash 在同一 bank，则 CPU 将 pending 直到 Flash 更新完成才能读到指令，如果此时仍通过 CPU 执行 LED 驱动代码则必然会出现闪烁。而 PDMA 控制器的运行是不需要 CPU 介入，如果此时由 PMDA 传输 LED 显示的相关数据，则可以避免闪烁的问题。

Timer0 定时触发 PDMA，PDMA 设置为 Scatter-Gather 模式，第 1 个描述符表传输第 1 位数码管显示数据，第 2 个描述符表传输第 2 位数码管显示数据。设置为 Ping-Pong 模式即第 1 个描述符表传输完之后，传输第 2 个描述符表；第 2 个描述符表传输完之后再传输第 1 个描述符表，依此循环传输。

1.2 执行结果

本范例代码执行编译和烧录后串口输出，如图 1-2 所示。

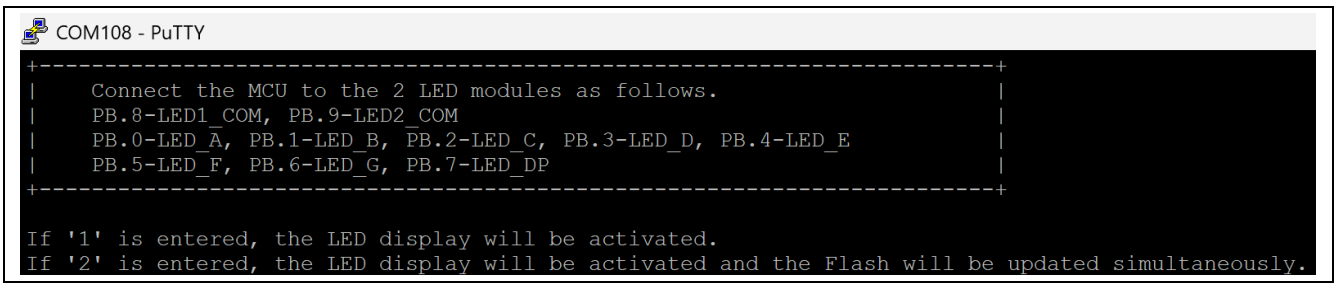

图 1-2 执行编译和烧录结果

输入字符“1”，两位 LED 将分别显示“1”、“2”，延时 900 ms，接着分别显示“3”、“4”，延时 900 ms，接着分别显示“5”、“6”，延时 900 ms，依此类推，直到显示“E”、“F”。串口输出，如图 1-3 所示。

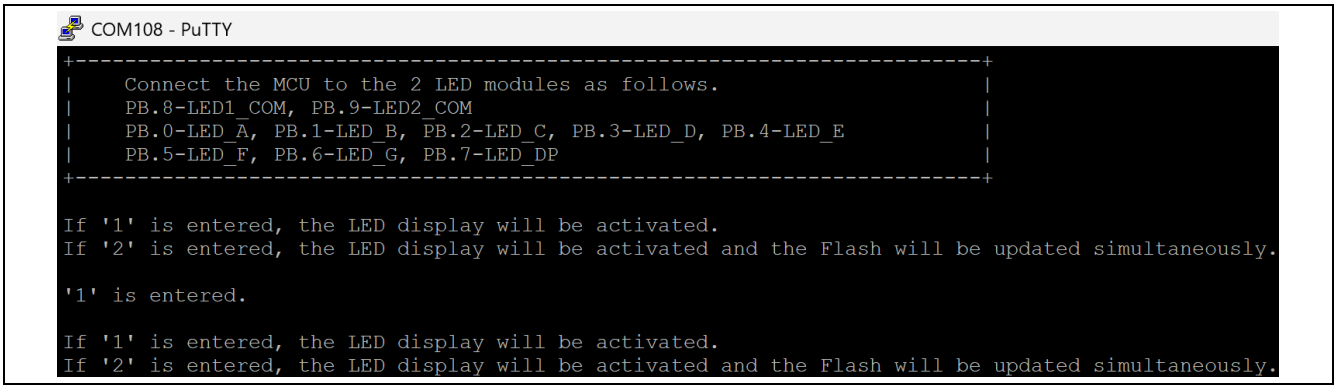

图 1-3 串口输入字符“1”

接着输入字符“2”，两位 LED 将分别显示“1”、“2”，同时执行 Flash 擦、写、读，耗时约 516 ms，通过逻辑分析仪抓取 PA0 引脚波形如图 1-4；接着分别显示“3”、“4”，同时执行 Flash 擦、写、读，耗时约 516 ms；接着分别显示“5”、“6”，同时执行 Flash 擦、写、读，耗时约 516 ms；依此类推，直到显示“E”、“F”。串口输出，如图 1-5 所示。

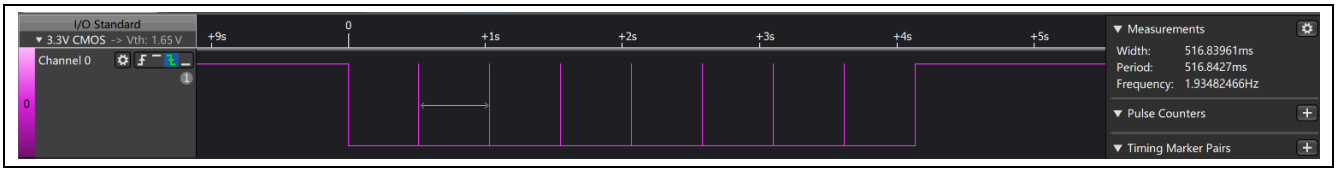

图 1-4 Flash 擦、写、读时间

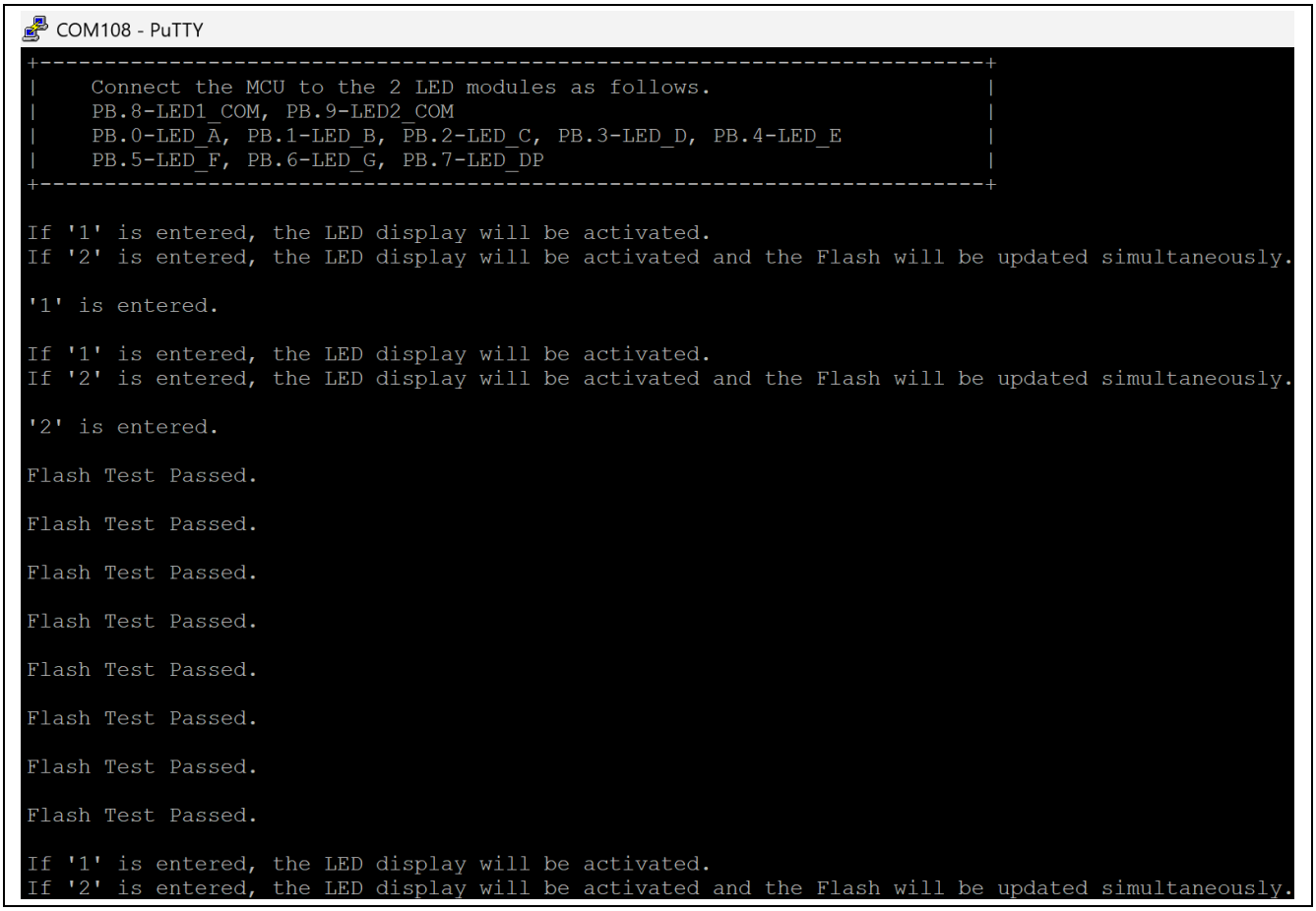

图 1-5 串口输入字符“2”

2. 代码介绍

宏定义及数码管显示数值码定义，代码位于 *main.c*。

```
#define UART0_Baudrate  115200
#define TIMER0_Freq     1000            // Set Timer0 interrupt frequency
#define LED1_COM        0x01            // Select LED1
#define LED2_COM        0x02            // Select LED2
#define TEST_PATTERN    0x5A5A5A00      // Flash programming value
#define PDMA_CHN        1
#define APROM_TEST_BASE 0x00008000UL    // Flash programming start address
#define APROM_TEST_LEN  0x00002000UL    // Flash programming length

uint8_t const g_u8Value[16] =
{
    0xC0,  //"0"
    0xF9,  //"1"
    0xA4,  //"2"
    0xB0,  //"3"
    0x99,  //"4"
    0x92,  //"5"
    0x82,  //"6"
    0xF8,  //"7"
    0x80,  //"8"
    0x90,  //"9"
    0x88,  //"A"
    0x83,  //"B"
    0xC6,  //"C"
    0xA1,  //"D"
    0x86,  //"E"
    0x8E,  //"F"
};
```

程序初始化，包括系统初始化、UART 初始化、显示初值初始化、PDMA 初始化及定时器初始化，代码位于 *main.c*。

```
/* Variable initialization */
char u8Option;
uint16_t i;

/* Init System, IP clock and multi-function I/O. */
SYS_Init();

/* Init UART0 to 115200-8n1 for print message */
UART_Open(UART0, UART0_Baudrate);

printf("+-----------------------------------------------------------------------+ \n");
printf("|    Connect the MCU to the 2 LED modules as follows.                   | \n");
printf("|    PB.8-LED1_COM, PB.9-LED2_COM                                       | \n");
printf("|    PB.0-LED_A, PB.1-LED_B, PB.2-LED_C, PB.3-LED_D, PB.4-LED_E         | \n");
printf("|    PB.5-LED_F, PB.6-LED_G, PB.7-LED_DP                                | \n");
printf("+-----------------------------------------------------------------------+ \n");
```

```
/* Set PB[9:0] as output mode */
GPIO_SetMode(PB, 0x03FF, GPIO_MODE_OUTPUT);
/* Set PA0 as output mode */
GPIO_SetMode(PA, BIT0, GPIO_MODE_OUTPUT);
PA0  = 1;

/* Variable initialization */
u8Index = 0;
u8Times = 1;

/* Display value initialization */
g_u16LED1[0] = ((uint16_t)LED1_COM << 8) + g_u8Value[u8Index];
g_u16LED2[0] = ((uint16_t)LED2_COM << 8) + g_u8Value[u8Index + 1];

/* PDMA Init */
PDMA_Init();

/* Set Timer0 frequency to 1 kHz */
TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, TIMER0_Freq);

/* Select Timer0 as PDMA trigger source */
TIMER_SetTriggerTarget(TIMER0, TIMER_TRG_TO_PDMA);

/* Start Timer0 */
TIMER_Start(TIMER0);
```

PDMA 初始化，代码位于 *main.c*。

```
void PDMA_Init(void)
{
    /* Open Channel 1 */
    PDMA_Open(PDMA, 1 << PDMA_CHN);

    PDMA_SetTransferMode(PDMA, PDMA_CHN, PDMA_TMR0, TRUE, (uint32_t)&DMA_DESC[0]);

    g_u32DMAConfig = \
                (0 << PDMA_DSCT_CTL_TXCNT_Pos) | /* Transfer count is 1 */ \
                PDMA_WIDTH_16 |  /* Transfer width is 16 bits(one word) */ \
                PDMA_SAR_FIX |   /* Source increment size is fixed(no increment) */ \
                PDMA_DAR_FIX |   /* Destination increment size is fixed(no increment */ \
                PDMA_REQ_BURST | /* Transfer type is burst transfer type */ \
                PDMA_BURST_1 |   /* Burst size is 1. No effect in single transfer type */ \
                PDMA_OP_SCATTER; /* Operation mode is scatter-gather mode */

    DMA_DESC[0].ctl = g_u32DMAConfig;
    /* Configure source address */
    DMA_DESC[0].src = (uint32_t)g_u16LED1;
    /* Configure destination address */
    DMA_DESC[0].dest = (uint32_t) &(PB->DOUT);
    /* Configure next descriptor table address */
    DMA_DESC[0].offset = (uint32_t)&DMA_DESC[1] - (PDMA->SCATBA);

    DMA_DESC[1].ctl = g_u32DMAConfig;
    /* Configure source address */
    DMA_DESC[1].src = (uint32_t)g_u16LED2;
```

```
    /* Configure destination address */
    DMA_DESC[1].dest = (uint32_t) &(PB->DOUT);
    /* Configure next descriptor table address */
    DMA_DESC[1].offset = (uint32_t)&DMA_DESC[0] - (PDMA->SCATBA);
}
```

Flash 编程及检查函数，代码位于 *main.c*。

```
int32_t Flash_Programming_Verify(uint32_t u32StartAddr, uint32_t u32EndAddr, uint32_t
u32Pattern)
{
    uint32_t    u32Addr;

    for (u32Addr = u32StartAddr; u32Addr < u32EndAddr; u32Addr += FMC_FLASH_PAGE_SIZE)
    {
        /* Erase page */
        if (FMC_Erase(u32Addr) != 0)
        {
            printf("FMC_Erase address 0x%x failed!\n", u32Addr);
            return -1;
        }

        /* Write test pattern to fill the whole page */
        if (Fill_Data_Pattern(u32Addr, u32Addr + FMC_FLASH_PAGE_SIZE, u32Pattern) < 0)
        {
            printf("Failed to write page 0x%x!\n", u32Addr);
            return -1;
        }

        /* Verify if page contents are all equal to test pattern */
        if (Verify_Data(u32Addr, u32Addr + FMC_FLASH_PAGE_SIZE, u32Pattern) < 0)
        {
            printf("\nData verify failed!\n ");          /* error message */
            return -1;                                   /* Program verify failed */
        }
    }

    printf("\nFlash Test Passed.\n");                    /* information message */
    return 0;                                            /* Flash test passed */
}
```

主循环，代码位于 *main.c*。

```
while (1)
{
    printf("\nIf '1' is entered, the LED display will be activated.\n");
    printf("If '2' is entered, the LED display will be activated and the Flash will be
     updated simultaneously.\n");

    u8Option = getchar();

    if (u8Option == '1')
    {
        printf("\n'1' is entered.\n");
```

```
        while (1)
        {
            /* Delay 900 ms */
            for (i = 0; i < 3; i++)
            {
                CLK_SysTickDelay(300000);
            }

            /* Update display data */
            u8Index = 2 * u8Times;
            g_u16LED1[0] = ((uint16_t)LED1_COM << 8) + g_u8Value[u8Index];
            g_u16LED2[0] = ((uint16_t)LED2_COM << 8) + g_u8Value[u8Index + 1];

            /* cycle count */
            u8Times++;

            if (u8Times >= (sizeof(g_u8Value)/2))
            {
                u8Times = 0;
                break;
            }
        }
    }

    if (u8Option == '2')
    {
        printf("\n'2' is entered.\n");

        while (1)
        {
            __NOP();
            SYS_UnlockReg();
            FMC_Open();

            FMC_ENABLE_AP_UPDATE();

            PA0 = 0; /* Testing the programming time */

            /* Program and check the data */
            if (Flash_Programming_Verify(APROM_TEST_BASE,
            (APROM_TEST_BASE+APROM_TEST_LEN), (TEST_PATTERN+u8Times)) < 0)
            {
                printf("\n\nAPROM test failed!\n");
            }

            PA0 = 1; /* Testing the programming time */

            FMC_DISABLE_AP_UPDATE();

            /* Disable FMC ISP function */
            FMC_Close();
            SYS_LockReg();

            /* Update display data */
            u8Index = 2*u8Times;
            g_u16LED1[0] = ((uint16_t)LED1_COM << 8) + g_u8Value[u8Index];
```

```
            g_u16LED2[0] = ((uint16_t)LED2_COM << 8) + g_u8Value[u8Index + 1];

            /* cycle count */
            u8Times++;

            if (u8Times >= (sizeof(g_u8Value)/2))
            {
                u8Times = 0;
                break;
            }
        }
    }
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M031_Series_BSP_CMSIS_V3.07.000
- IDE 版本
  - Keil uVision 5.41

3.2 硬件需求

- 电路组件
  - NuMaker-M032SE V1.3
  - NuEdu-Basic01 V2.0
- 线路示意图
  - 将 NuMaker-M032SE V1.3 相关引脚与 NuEdu-Basic01 V2.0 相关引脚按图 3-1 进行连接。

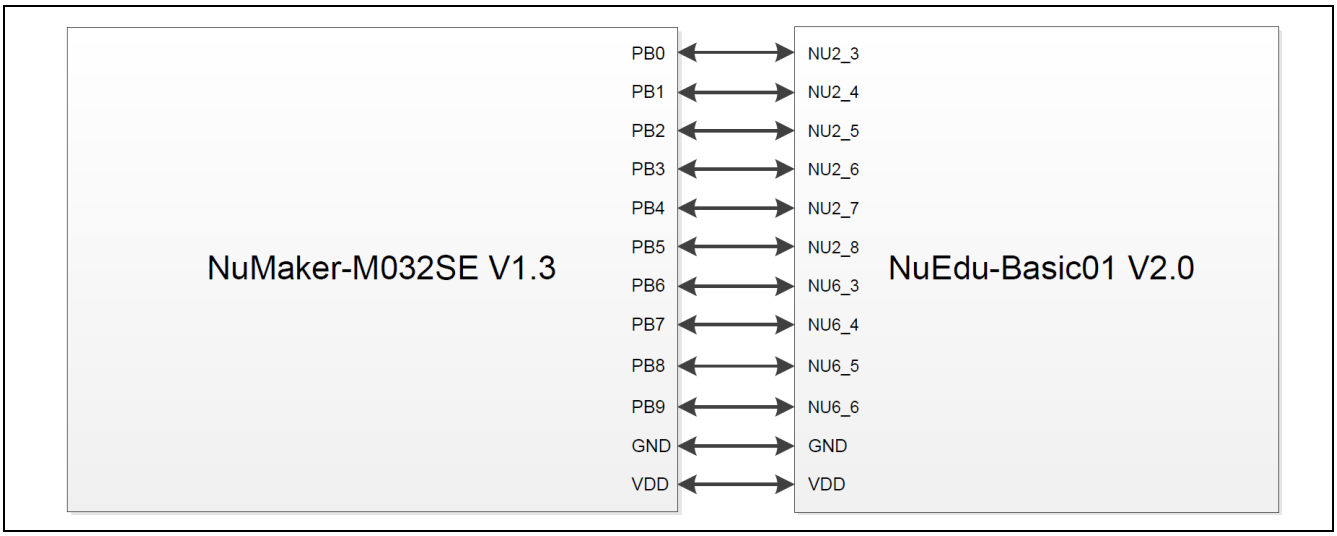

图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

EC_M032_Timer_Trigger_PDMA_To_Drive_LED_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 *Timer_Trigger_PDMA_To_Drive_LED.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
2025.10.15	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.