計時器觸發 PDMA 驅動段碼 LED 顯示

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼實現當 M032 正處於更新 Flash 的時候，仍可通過 Timer0 觸發 PDMA 驅動兩位 Segment LED 顯示。
BSP 版本	M031_Series_BSP_CMSIS_V3.07.000
開發平台	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

本範例代碼基於 M032 實現即使在更新 Flash 資料的同時，也可以正常驅動兩位數碼管顯示。

市場上某些產品，在應用程式運行的過程需要更新 Flash 資料，如果更新的資料量大或更新頻繁，就會導致其他程式功能處於阻塞狀態。例如應用程式需要更新 Flash 還需要驅動 Segment LED 顯示，當更新 Flash 時間超過 20 ms，就會導致顯示閃爍，因为 LED 刷新間隔会超過 20 ms。此參考程式就可以解決此問題。

1.1 原理

每個段碼數碼管有 1 個位選引腳，當位選引腳輸出高電平（或低電平），數碼管點亮；反之則熄滅。此參考設計位選引腳為高電平有效，即高電平點亮數碼管。

每個段碼數碼管通常由 8 個發光二極體（A~G 和 DOT）組成，通過控制每個二極體引腳的電平實現亮滅。此參考設計二極體引腳為低電平有效，即低電平點亮相應段。

兩位段碼數碼管及控制原理如圖 1-1。

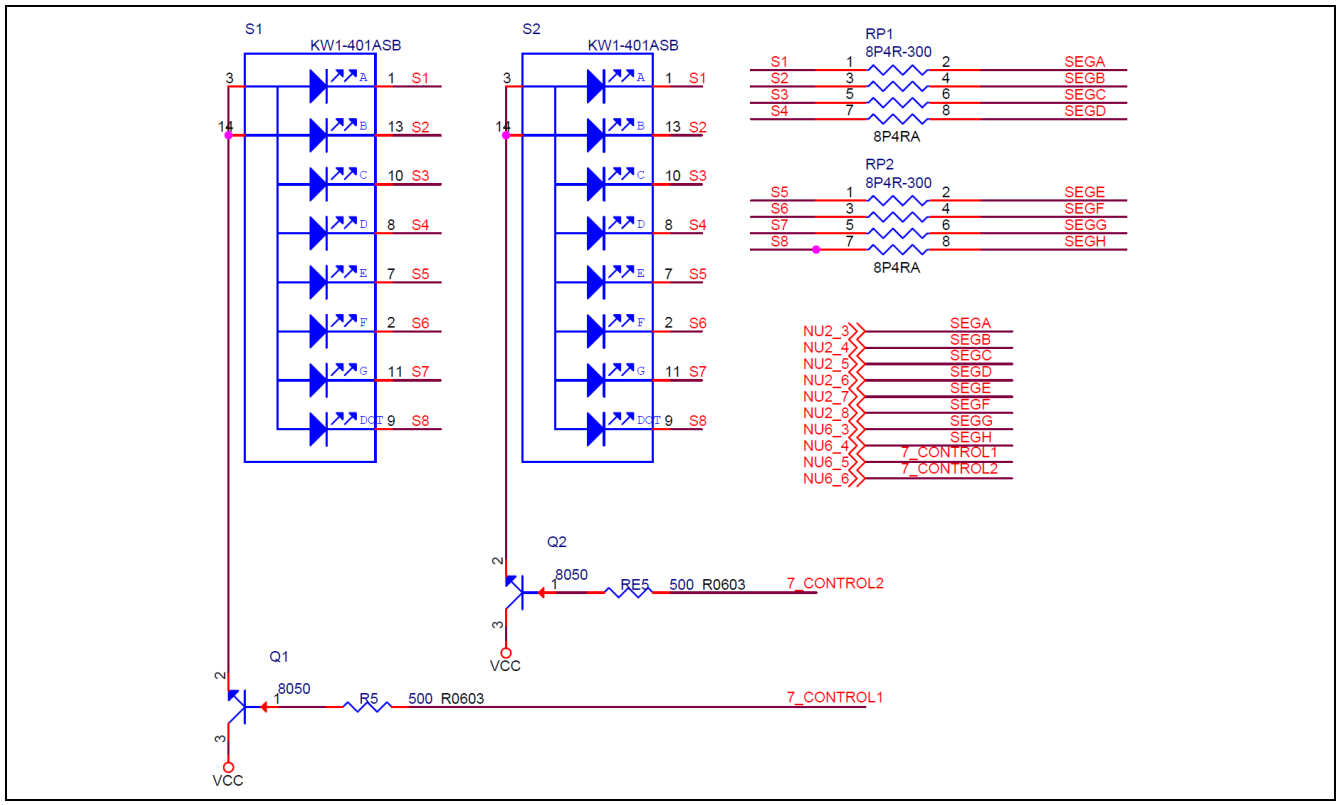

圖 1-1 兩位段碼數碼管及控制原理

通過分時掃描兩位數碼管，利用人眼視覺暫留效應實現連續顯示。

若 CPU 要讀取的執行指令和正更新的 Flash 在同一 bank，則 CPU 將 pending 直到 Flash 更新完成才能讀到指令，如果此時仍通過 CPU 執行 LED 驅動代碼則必然會出現閃爍。而 PDMA 控制器的運行是不需要 CPU 介入，如果此時由 PMDA 傳輸 LED 顯示的相關資料，則可以避免閃爍的問題。

Timer0 定時觸發 PDMA，PDMA 設置為 Scatter-Gather 模式，第 1 個描述符表傳輸第 1 位數碼管顯示資料，第 2 個描述符表傳輸第 2 位數碼管顯示資料。設置為 Ping-Pong 模式即第 1 個描述符表傳輸完之後，傳輸第 2 個描述符表；第 2 個描述符表傳輸完之後再傳輸第 1 個描述符表，依此循環傳輸。

1.2 執行結果

本範例代碼執行編譯和燒錄後序列埠輸出，如圖 1-2 所示。

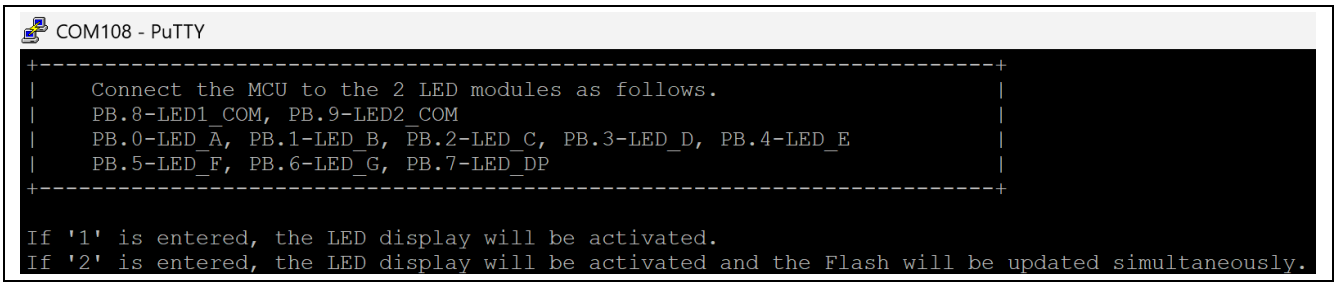

圖 1-2 執行編譯和燒錄結果

輸入字元“1”，兩位 LED 將分別顯示“1”、“2”，延時 900 ms，接著分別顯示“3”、“4”，延時 900 ms，接著分別顯示“5”、“6”，延時 900 ms，依此類推，直到顯示“E”、“F”。序列埠輸出，如圖 1-3 所示。

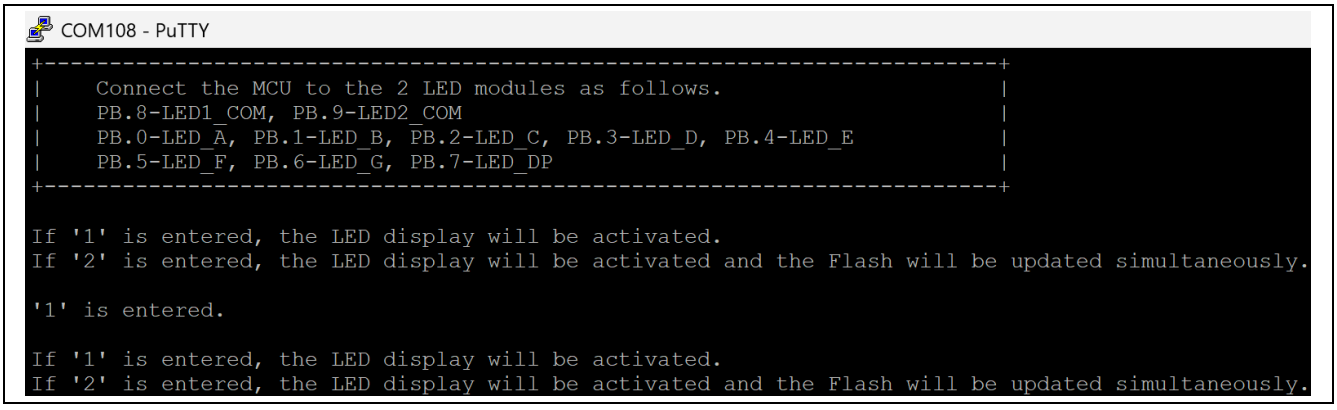

圖 1-3 序列埠輸入字元“1”

接著輸入字元“2”，兩位 LED 將分別顯示“1”、“2”，同時執行 Flash 擦、寫、讀，耗時約 516 ms，通過邏輯分析儀抓取 PA0 引腳波形如圖 1-4；接著分別顯示“3”、“4”，同時執行 Flash 擦、寫、讀，耗時約 516 ms；接著分別顯示“5”、“6”，同時執行 Flash 擦、寫、讀，耗時約 516 ms；依此類推，直到顯示“E”、“F”。序列埠輸出，如圖 1-5 所示。

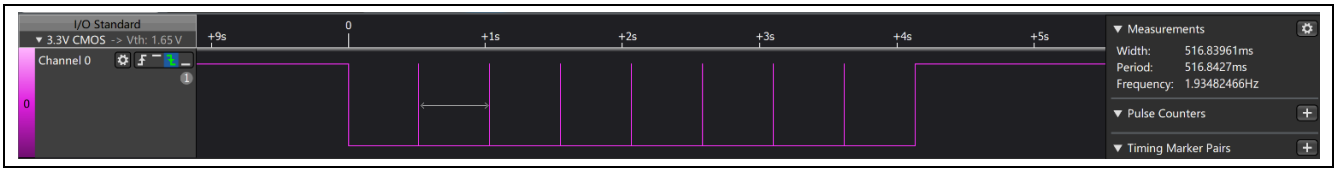

圖 1-4 Flash 擦、寫、讀時間

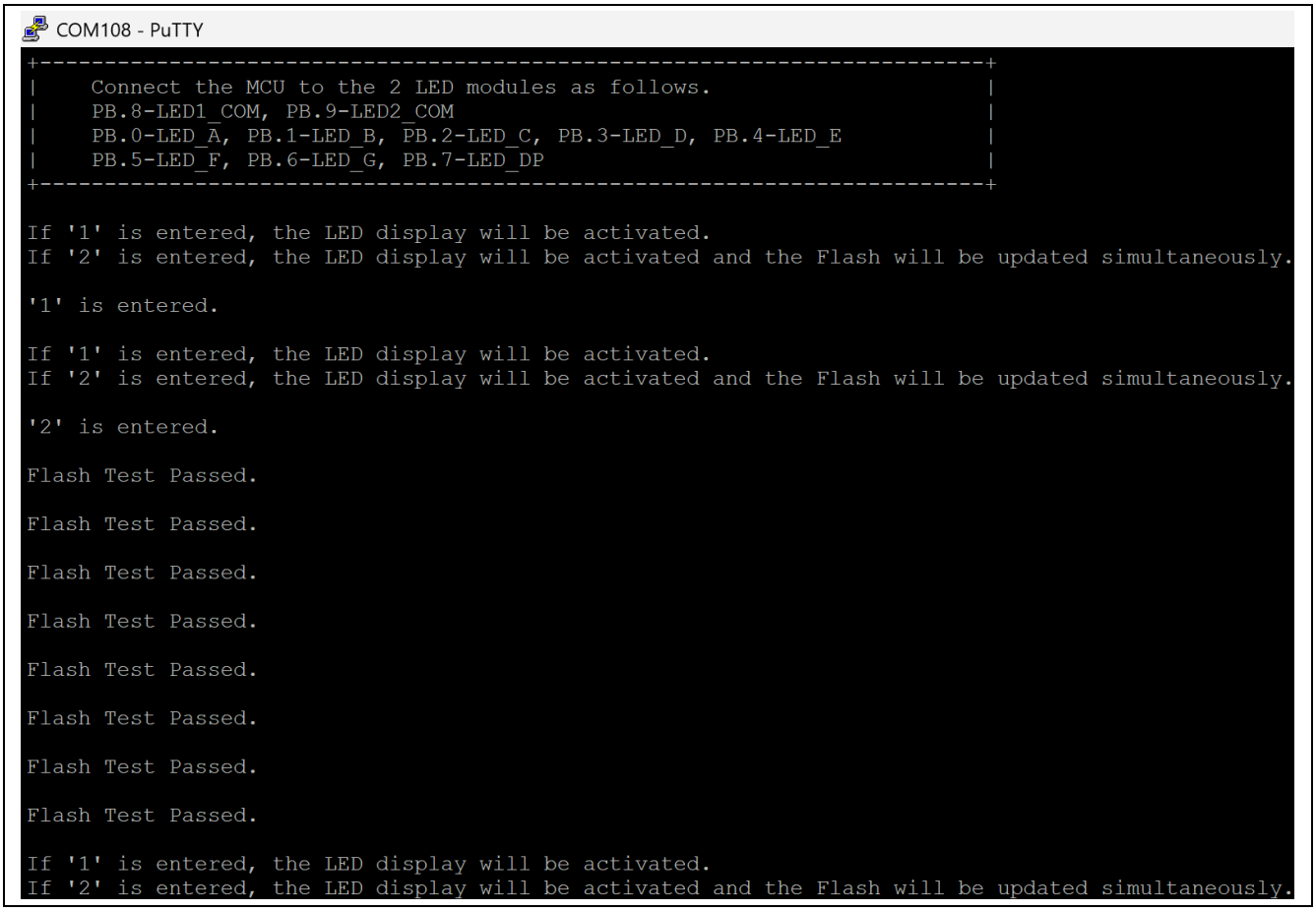

圖 1-5 序列埠輸入字元“2”

2. 代碼介紹

巨集定義及數碼管顯示數值碼定義，代碼位於 *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"
};
```

程式初始化，包括系統初始化、UART 初始化、顯示初值初始化、PDMA 初始化及計時器初始化，代碼位於 *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，代碼位於 *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 編程及檢查函式，代碼位於 *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 */
}
```

主循環，代碼位於 *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. 軟體與硬體需求

3.1 軟體需求

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

3.2 硬體需求

- 電路元件
  - NuMaker-M032SE V1.3
  - NuEdu-Basic01 V2.0
- 線路示意圖
  - 將 NuMaker-M032SE V1.3 相關引腳與 NuEdu-Basic01 V2.0 相關引腳按圖 3-1 進行連接。

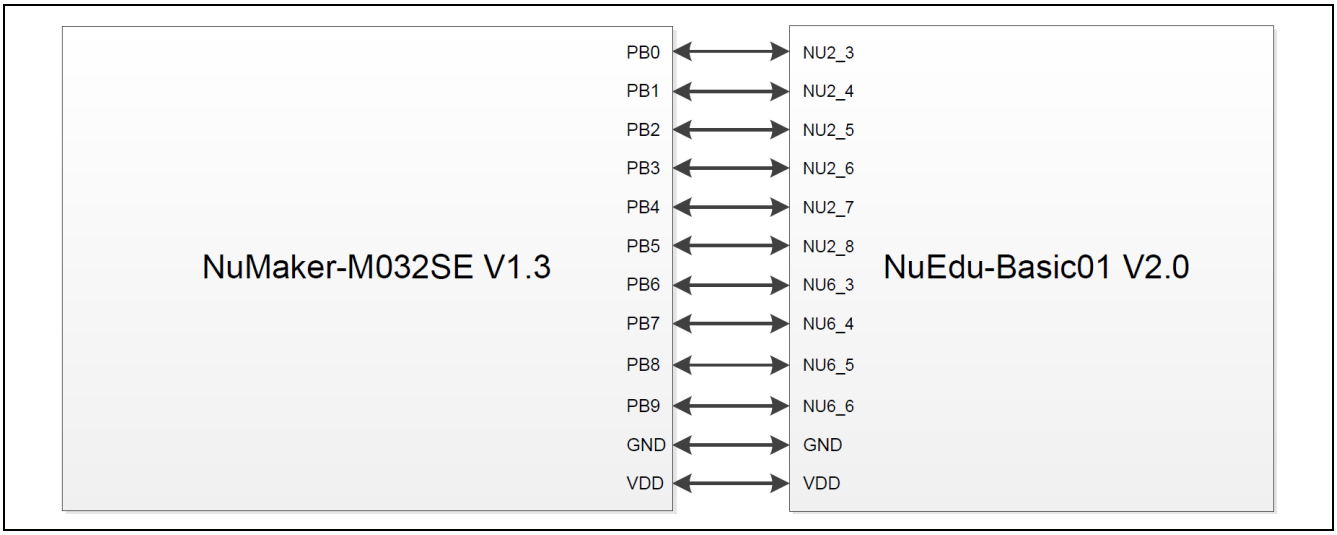

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

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

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 *Timer_Trigger_PDMA_To_Drive_LED.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
2025.10.15	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.