使用 PSIO 實作 ARGB2 LED 燈效

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 M252 PSIO 實作 ARGB2 LED 時序並實現彩虹燈、走馬燈等燈效
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
開發平台	NuMaker-M252KG Ver1.0

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

PSIO (Programmable Serial I/O) 提供簡易的方法達到各種序列傳輸的接收與傳送，例如: HDQ、DMX512、1-wire、IR、PS/2、Microwire、Wiegand、LED 等。本文描述如何使用 PSIO 實現 ARGB2 LED 燈效，並演示 ARGB2 LED 彩虹燈、走馬燈效果，此範例代碼使用一組 Timer 來定時更新 ARGB2 LED 彩虹燈、走馬燈的顏色，一組 UART 來選擇執行模式；其內容包含 PSIO 與 ARGB2 LED 的基本介紹、範例程式的說明以及運行結果。

1.1 原理

1.1.1 ARGB2 LED 資料傳輸時間 (TH + TL = 1.2 µs ± 600 ns)

ARGB2 LED 時序圖如圖 1-1 與圖 1-2 所示。邏輯 ‘0’ 定義高準位 0.3 µs ± 0.15 µs 與低準位 0.9 µs ± 0.15 µs，邏輯 ‘1’ 定義為高準位 0.9 µs ± 0.15 µs 與低準位 0.3 µs ± 0.15 µs。

在使用 PSIO 控制 ARGB2 LED 燈條，必須先設定好時序規則(slot)，按照預期的時序運作，將時間基本單位劃分約 0.3 µs 可以造出所需的波形。時序計算公式如下：

- PSIO clock source 選擇 PCLK1，PCLK1 clock 計算方式 HIRC (48 MHz) / 4 = 12 MHz
- PSIO clock 使用 PCLK1 除頻 4 = 3MHz
- PSIO one slot cycle time = 1 / 3MHz = 0.33 µs

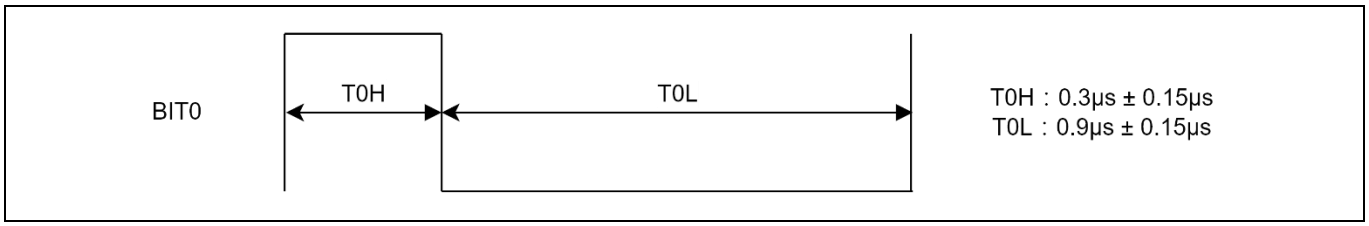

圖 1-1 邏輯 ‘0’ (bit 0)

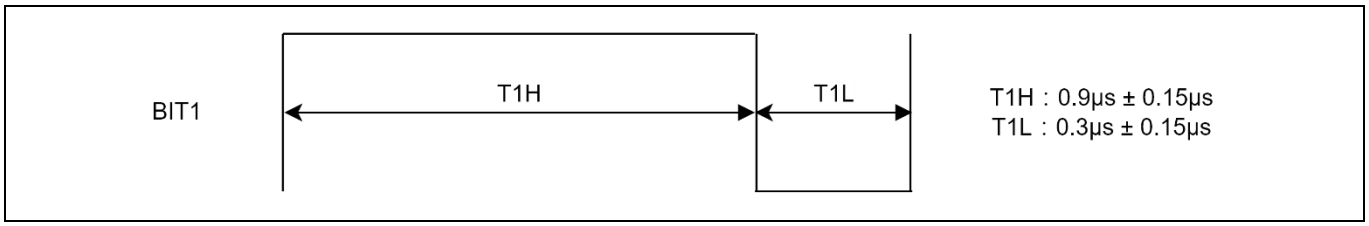

圖 1-2 邏輯 ‘1’ (bit 1)

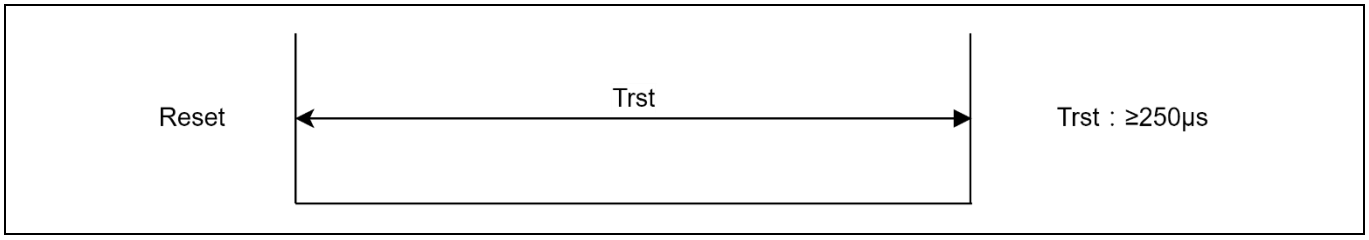

圖 1-3 重置 (Trst)

1.2 高階功能模式

1.2.1 Feedback 模式

ARGB2 LED 燈條除了能透過 MCU 傳遞信號做燈效，也可以回傳訊息給 MCU，內容有兩種：

- 回傳燈條上 LED 串接顆數與目前燈條被設定的 ID 號
- 回傳燈條上 LED 串接顆數與 LED 消耗電流大小

Feedback 模式指令波形如圖 1-4，下達 Feedback 模式指令給 ARGB2 LED 燈條後，ARGB2 LED 燈條會回傳 High Pulse。PSIO Control PIN 由 Output 切換成 Input 讀取 Pulse，藉由讀取 Pulse 的數目及寬度便可得到該燈條上有多少顆 LED 及所使用的最大電流、目前燈條被設定的 ID。若 160 μs 內沒有收到 High Pulse，MCU 要恢復到 Normal Mode，即 PSIO Control PIN 由 Input 切回 Output。

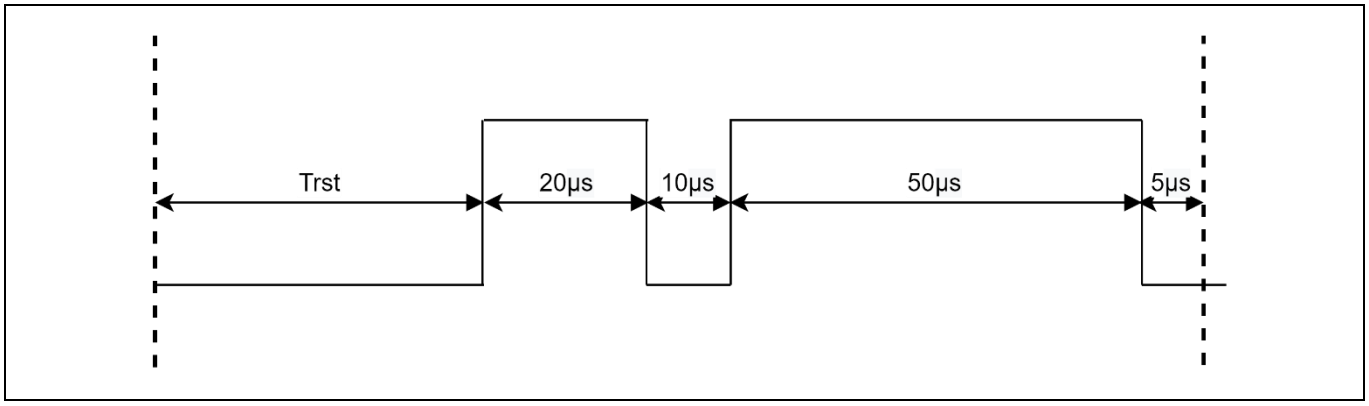

圖 1-4 進入 Feedback 模式指令波形

Pulse 寬度代表意義：

- 收到 20 μs High Pulse：LED 耗電流為 12mA
- 收到 10 μs High Pulse：LED 耗電流為 5mA

1.2.2 TH20 SET 模式

由 MCU 傳送 20 µs (高準位) + 250 µs (低準位) 訊號至 ARGB2 LED 燈條即可進入 TH20 SET 模式，後面再接續 24 bits，如圖 1-5。接續在 TH20 SET 模式後是 24 bits 設定資料，可參考表 1-1、表 1-2。

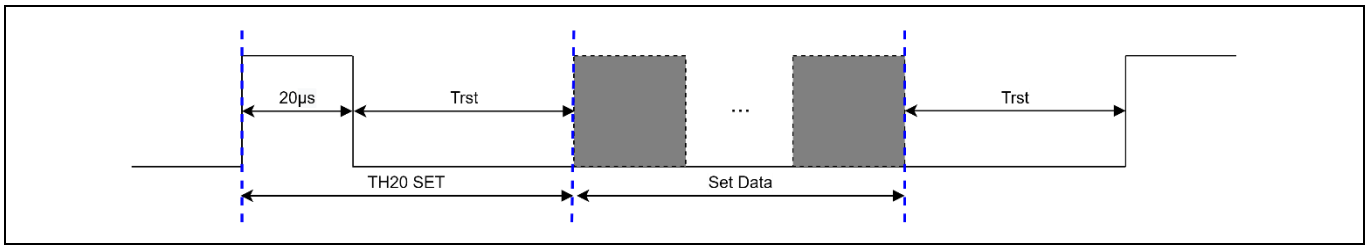

圖 1-5 TH20 SET 模式波形

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS <5>	SS <4>	0	G <4>	G <3>	G <2>	G <1>	G <0>	SS <3>	SS <2>	SS <6>	R <4>	R <3>	R <2>	R <1>	R <0>	SS <1>	SS <0>	0	B <4>	B <3>	B <2>	B <1>	B <0>

表 1-1 TH20 SET 模式 24 bits 資料格式

SS[0]	設置接收新資料後，PWM 是否重新計數，default：0 0：不重新計數 1：PWM 一律會由 0 開始計數
SS[2:1]	Imax (請勿更動)，default：00
SS[3]	為數據更新時機設置，default：0 0：接收到 250 µs Trst 信號才同步更新 1：收滿 24 bits 立即更新
SS[5:4]	設置 PWM 頻率，default：11 00：1.25 kHz 01：2.5 kHz 10：10 kHz 11：20 kHz
SS[6]	設置 Feedback 模式回傳內容，default：0

	0：回傳 LED 顆數與 LED 消耗電流大小 1：回傳 LED 顆數與目前燈條被設定的 ID 號
G[4:0]	G LED Dimming 功能，共 31 階，default：0x1F
R[4:0]	R LED Dimming 功能，共 31 階，default：0x1F
B[4:0]	B LED Dimming 功能，共 31 階，default：0x1F

表 1-2 TH20 SET 模式設置功能說明

Dimming 功能可把原最大電流設置更細緻，達到最佳燈效。控制輸出電流公式如下，以 G LED 為例：

$$Iout = \frac{\mathrm{Im} \times (15.5 + 0.5 \times \mathrm{G}[0] + 1 \times \mathrm{G}[1] + 2 \times \mathrm{G}[2] + 4 \times \mathrm{G}[3] + 8 \times \mathrm{G}[4])}{31}$$

1.2.3 並聯模式 Y Cable 功能

透過 Y Cable 功能可以對 ARGB2 LED 燈條做 ID 設定。開啟 Y Cable 功能如圖 1-6，進入前必需有一個 250 µs (低準位) Pulse 的 Reset 訊號。並聯模式 Y Cable 功能的 Command 有四種，可參考表 1-3。

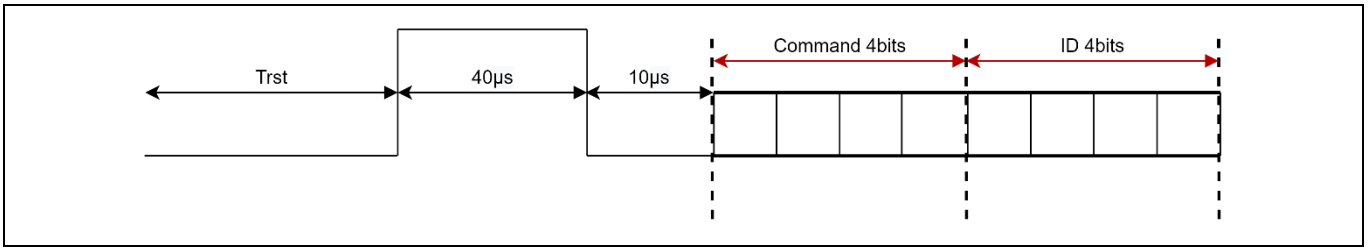

圖 1-6 開啟 Y Cable 指令波形

Command [7：4]	0001：設置燈條 ID 號 (SET_ID) 0010：清除燈條 ID 號 (CLR_ID) 0011：確認 ID 號是否存在或重複 (CHK_ID) 0100：指定使用 ID 號燈條 (Specify_ID)
ID [3：0]	範圍為：0000 ~ 1111

表 1-3 並聯模式 Y Cable 指令設定說明

1.2.3.1 設置 ARGB2 LED 燈條 ID (SET_ID)

SET_ID + ID 指令傳送完畢後，如設置 ID 成功，ARGB2 LED 燈條會回傳 High Pulse，PSIO 需要立刻由 Output 切換成 Input，用以判斷 ID 設定情況。依此方法將所有並聯的 ARGB2 LED 燈條給予 ID 號，當所有並聯燈條都有 ID 號後再下 SET_ID，ARGB2 LED 燈條就不會回覆 High Pulse。[1]

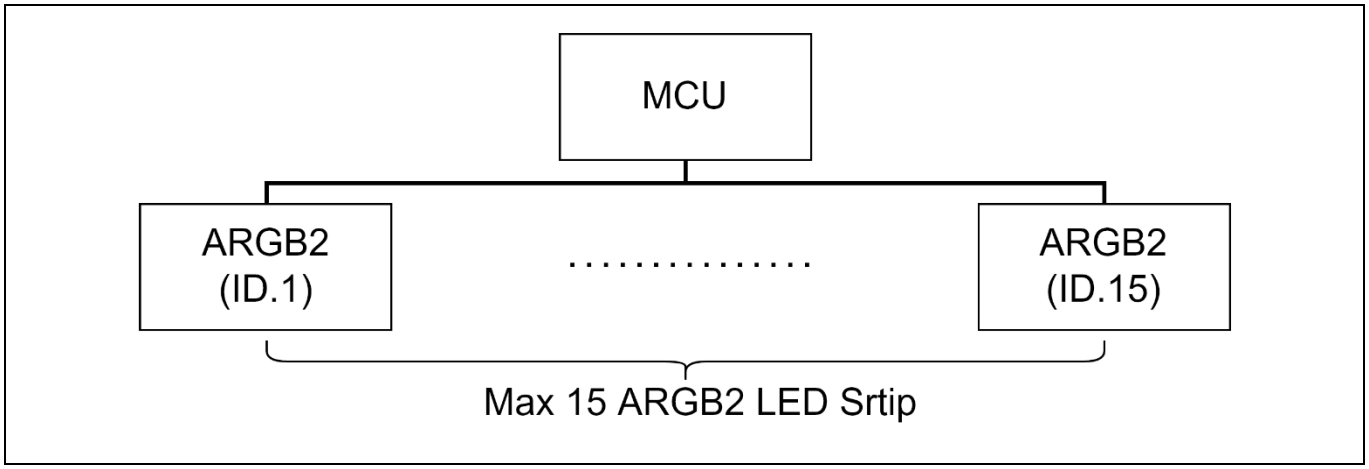

圖 1-7 Set ID 示意圖

1.2.3.2 清除 ARGB2 LED 燈條 ID (CLR_ID)

當傳送的 ID = 0000 時可以清除所有 ARGB2 LED 燈條 ID，當 ID = 0001 ~ 1111 時可以清除指定 ID 為 0。設置 ID 前建議做一次 CLR_ID + 0000，確保所有並聯 ARGB2 LED 燈條 ID 皆為 0。

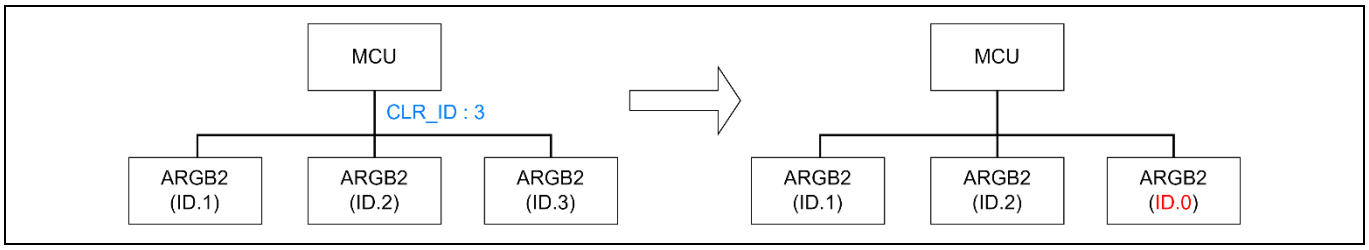

圖 1-8 CLR_ID 示意圖

[1] 每次 Power ON 後，ARGB2 燈條的 ID 都會變成 0

1.2.3.3 確認 ARGB2 LED 燈條 ID (CHK_ID)

MCU 傳送 CHK_ID 指令後，如有符合指定的 ID，ARGB2 LED 燈條會回傳 High Pulse，假如並聯的 ARGB2 LED 燈條有 ID 被重複設定，此時發送 CHK_ID 指令，MCU 會收到被重複設定數量的 High Pulse。

1.2.3.4 指定使用 ID 號燈條 (Specify_ID)

只有被指定到的 ID 燈條會有動作，當並聯的 ARGB2 LED 燈條 ID 為 0 時則會一起動作。波形示意圖，如圖 1-9。

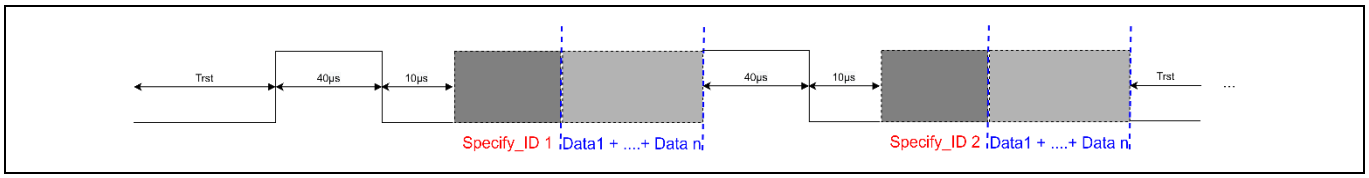

圖 1-9 Specify_ID 示意圖

並聯模式 Y Cable 功能控制流程如下：

- CLR_ID：把所有並聯的 ARGB2 LED 燈條 ID 設定為 0
- SET_ID：分配不同 ID 給所有 ARGB2 LED 燈條，分配 ID 成功後，ARGB2 LED 燈條會回傳 Pulse
- CHK_ID：對分配到 ID 的 ARGB2 LED 燈條分別做確認，如分配成功 ARGB2 LED 燈條會回傳 Pulse
- Specify_ID：可以指定燈條做控制

1.3 執行結果

使用終端軟體透過 UART 選擇執行模式，波特率設定 115200，終端軟體執行結果如圖 1-10，可選擇模式如下，其中 1 ~ 3 為 ARGB2 LED 燈效顯示，4 ~ 7 為開啟燈條不同模式：

1. 輪流執行顯示綠燈、紅燈、藍燈、燈效
2. 執行彩虹燈效果
3. 執行走馬燈效果 (以彩虹方式)
4. 執行 Feedback 模式
5. 執行 TH20 SET 模式
6. 執行 Y Cable 模式
7. 設置 RGB 顯示亮度

```
================================================
        Enter the select test function:
================================================
[01] RGB Loop
[02] Rainbow cycle
[03] Theater chase rainbow
[04] Feedback mode
[05] Th20 set mode
[06] Y cable mode
[07] Set RGB Brightness
```

圖 1-10 終端軟體顯示畫面

1.3.2 邏輯 ’1’ (bit 1)、邏輯 ’0’ (bit 0)、重置 (Trst) 執行結果

圖 1-11 邏輯’1’和邏輯’0’執行結果

圖 1-12 重置 (Trst) 執行結果

1.3.3 GRB 循環執行結果

圖 1-13 綠燈執行結果

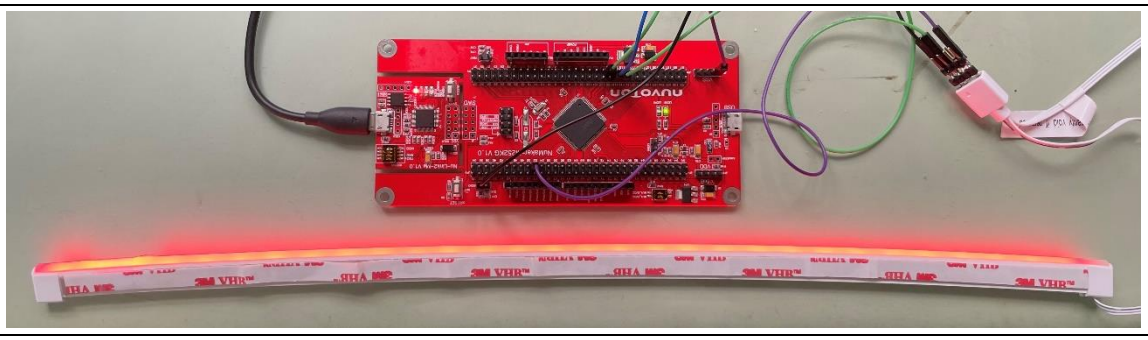

圖 1-14 紅燈執行結果

圖 1-15 藍燈執行結果

1.3.4 彩虹燈執行結果

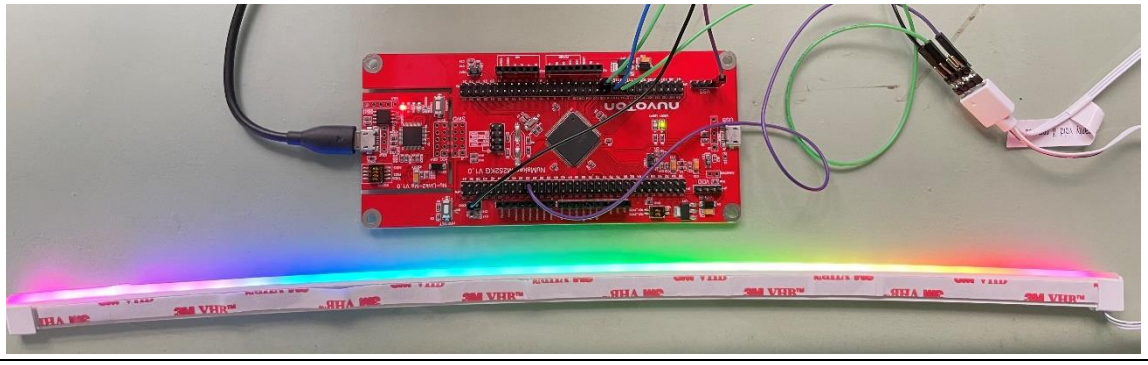

圖 1-16 彩虹燈執行結果

1.3.5 走馬燈執行結果

圖 1-17 走馬燈執行結果

1.3.6 Feedback 模式執行結果

ARGB2 規範進入 Feedback 的時序需要 20 µs (高準位) + 10 µs (低準位) + 50 µs (高準位)。如圖 1-18 所示。

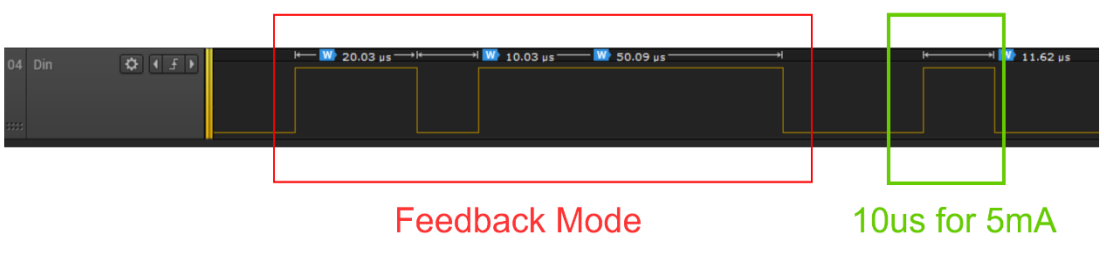

圖 1-18 Feedback 模式波形和 Pulse 寬度代表最大耗電流

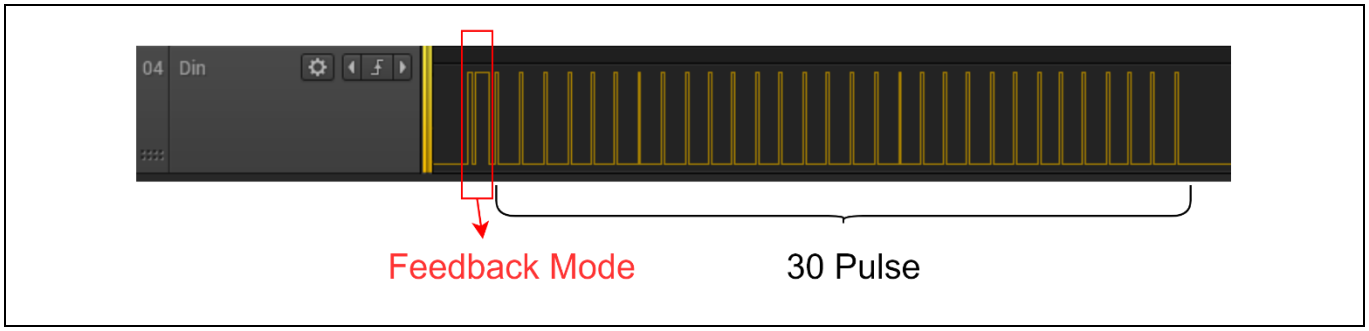

圖 1-19 ARGB2 LED 燈條回傳 Pulse

1.3.7 TH20 Set 模式執行結果

由 MCU 傳送 20 µs (高準位) + 250 µs (低準位) 訊號至 ARGB2 LED 燈條即可進入 TH20 SET 模式，後面再接續 24 bits 資料。

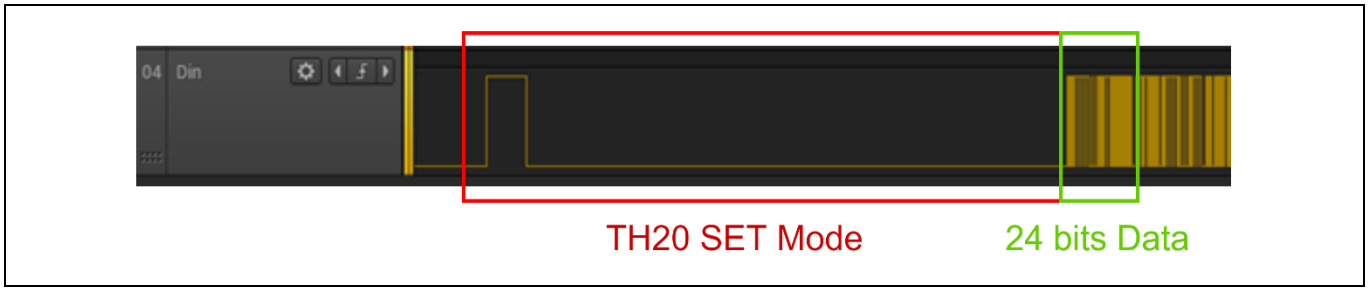

圖 1-20 開啟 TH20 模式模式波形

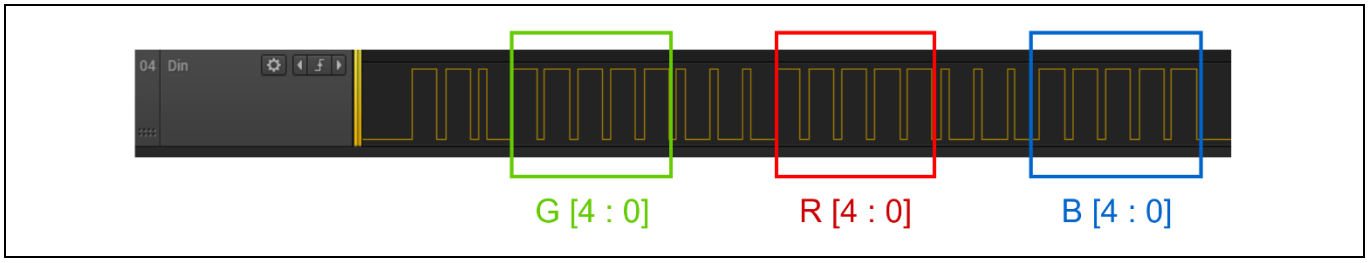

圖 1-21 TH20 設置 24 bits 資料波形

1.3.8 並聯模式 Y Cable 執行結果

ARGB2 規範支持多燈條並聯控制模式 (最多 15 條) 。使用指定的命令，可以識別每個燈條分配的唯一的燈條動態 ID (通過 set dynamic ID 命令) 。命令完成後，MCU 可以單獨控制並將 " 清除 ID " 、 " 檢查 ID " 、 " 指定 ID " 命令發送到每個燈條。

MCU 設置輸出時序 40 µs + 10 µs + 8 bits (Command + ID) 進入 Y Cable 模式，波形如圖 1-22。

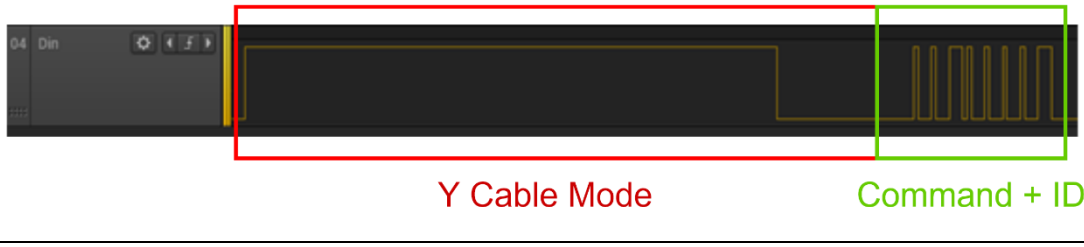

圖 1-22 開啟 Y Cable 功能和 8 bits (Command + ID) 波形

- 設置 ARGB2 燈條 ID (SET_ID) 執行結果

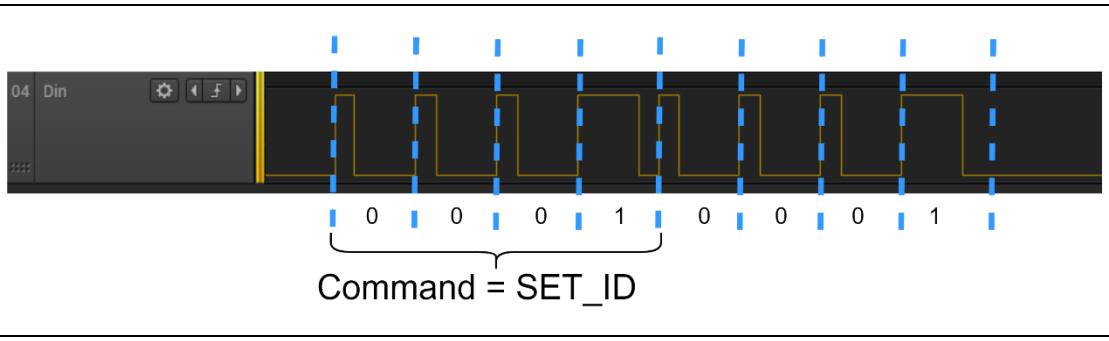

圖 1-23 設置 ARGB2 燈條 (SET_ID) 波形

- 清除 ARGB2 燈條 ID (CLR_ID) 執行結果

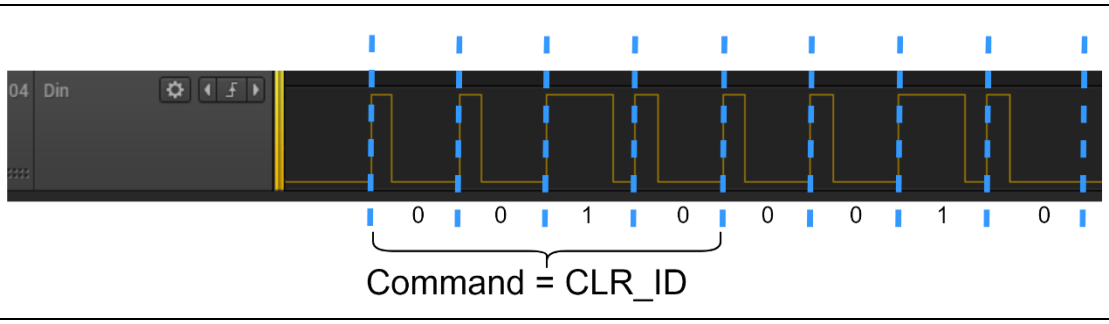

圖 1-24 清除 ARGB2 燈條 (CLR_ID) 波形

- 確認 ARGB2 燈條 ID (CHK_ID) 執行結果

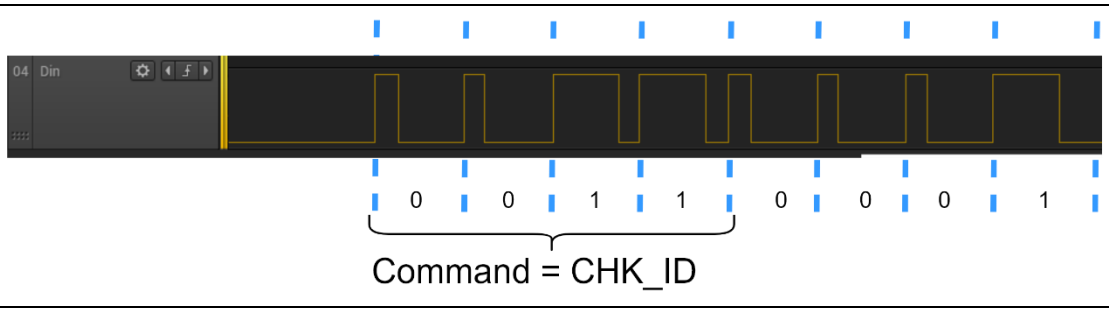

圖 1-25 確認 ARGB2 燈條 (CHK_ID) 波形

圖 1-26 確認 ARGB2 燈條 ID 成功示意圖

- 指定使用 ID 號碼 ARGB2 燈條 (Specify_ID) 執行結果

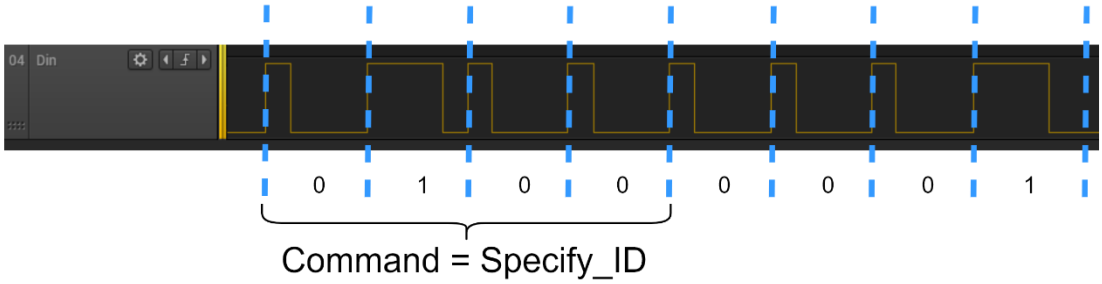

圖 1-27 指定使用 ID 號燈條 (Specify_ID)

2. 代碼介紹

2.1 主代碼介紹

代碼工作流程如下:

1. *main.c* 中初始化 M252 系列相關外設，包含 HIRC、PSIO、TIMER0、UART0
2. ARGB2StripTestMenu() function 中 while() 迴圈選擇執行 GRB 循環、彩虹燈、走馬燈、Feedback 模式、TH20 SET 模式、並聯 Y Cable 模式功能
3. 邏輯 '0'、邏輯 '1' 波形使用 PSIO_SendDataWithSize(uint32_t u32DataSize, uint32_t u32Data) 實現

```
int main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();

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

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

    /* Lock protected registers */
    SYS_LockReg();

    Delay_Init();

    /* Use slot control 0 and PIN 0 */
    PSIO_StripInit(PSIO_SC0, PSIO_PIN0);

    PSIO_StripSetBrightness(255);

    ARGB2StripTestMenu();
}
```

main.c PSIO_StripInit() 初始化 PSIO Slot Control 和 PSIO PIN，請參考如下代碼。

```
void PSIO_StripInit(uint8_t u8SlotCtrl, uint8_t u8DataPin)
{
    gsPSIO_ARGB2_Cfg.u8SlotCtrl = u8SlotCtrl;
    gsPSIO_ARGB2_Cfg.u8DataPin = u8DataPin;

    PSIO_Strip_OpenOutput(PSIO_PIN_ENABLE);
}
```

PSIS_ARGB2_Driver.c PSIO_Strip_OpenOutput() 在初始化 Output Mode 時，使用 PSIO_SET_GENCTL() 設定 Pin 的狀態，內容為使用的 Pin、Slot controller、模式、初始化狀態及結束狀態等。PSIO_SET_WIDTH() 與 PSIO_SET_OUTPUT_DEPTH() 則是與資料的格式有關，此範例初始設定輸出寬度為 8 bit、輸入寬度為 1 bit，深度皆為 1。

```
static void PSIO_Strip_OpenOutput(uint32_t u32Switch)
{
    /* PSIO UART TX PIN general setting */
    PSIO_SET_GENCTL(PSIO,
                    gsPSIO_ARGB2_Cfg.u8DataPin,
                    u32Switch,
                    gsPSIO_ARGB2_Cfg.u8SlotCtrl,
                    PSIO_OUTPUT_MODE,
                    PSIO_LOW_LEVEL,
                    PSIO_LOW_LEVEL);

    /* Set data order ad MSB */
    PSIO_SET_ORDER(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, PSIO_MSB);

    /* Set output data buffer width as 8 bit, input data width as 1 bit*/
    PSIO_SET_WIDTH(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, 1, 8);

    /* Set output data buffer depth as 1 */
    PSIO_SET_OUTPUT_DEPTH(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, PSIO_DEPTH1);

    /* Set slot controller trigger source as software trigger */
    PSIO_SET_TRIGSRC(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SW_TRIGGER);
}
```

main.c while() 迴圈實現選擇功能執行，請參考如下代碼。

```
void ARGB2StripTestMenu(void)
{
    uint8_t u8Index = 0;
    struct
    {
        char *chName;
        int (*funcptr)(void);
    } sMainMenu[] =
    {
        {"", NULL},
        {"RGB Loop", &Strip_RGBLoop},
        {"Rainbow cycle", &Strip_RainbowCycle},
        {"Theater chase rainbow", &Strip_TheaterChaseRainbow},
        {"Feedback mode", &Strip_FeedbackMode},
        {"Th20 set mode", &Strip_Th20SetMode},
        {"Y cable mode", &Strip_YCableSetupMenu},
        {"Set RGB Brightness", &Strip_SetBrightness},
    };

    while (1)
    {
        printf("\r\n\r\n");
```

```
        printf("============================================\r\n");
        printf("\tEnter the select test function: \r\n");
        printf("============================================\r\n");

        for (u8Index = 1; u8Index < sizeof(sMainMenu) / sizeof(sMainMenu[0]); u8Index++)
        {
            printf("[%02d] %s\r\n",
                   u8Index,
                   sMainMenu[u8Index].chName);
        }

        printf("\r\n");

        u8Index = sysGetNum();

        if ((u8Index == 0) || (u8Index >= sizeof(sMainMenu) / sizeof(sMainMenu[0])))
        {
            continue;
        }

        if (sMainMenu[u8Index].funcptr != NULL)
        {
            sMainMenu[u8Index].funcptr();
        }
    }
}
```

輸出邏輯 ‘0’、邏輯 ‘1’ 波形訊號函式需要三種不同的動作，分別是輸出高準位、從緩衝區輸出資料及輸出低準位，時序的規劃如圖 1-1 和圖 1-2 所述，並整理於表 2-1。Slot 0 ~ Slot 2 可視為 ARGB2 LED 協定的 1 bit。PSIO_SET_WIDTH() 根據輸出的資料大小設定輸出寬度，例如設定 ARGB2 LED 的顏色需要設定輸出寬度為 24 bit。PSIO_SET_SCCTL() 利用 PSIO 的迴圈功能，將此行為再複製 23 次，即輸出 24 bit 資料。設定完成後，用軟體觸發 PSIO_START_SC() 的方式，讓 slot controller 開始執行。

	Slot0	Slot1	Slot2
Check Point	Check Point 0	Check Point 1	Check Point 2
Engine Clock	1	2	1
PIN Action	Output High	Output Data Form Buffer	Output Low

表 2-1 PSIO 設定輸出邏輯’0’、邏輯’1’

```
void PSIO_SendDataWithSize(uint32_t u32DataSize, uint32_t u32Data)
{
    S_PSIO_CP_CONFIG sDataConfig = {0};

    /* Slot 0 ~ 7 */
    sDataConfig.CKPT0SLT = PSIO_SLOT0;
    sDataConfig.CKPT1SLT = PSIO_SLOT1;
    sDataConfig.CKPT2SLT = PSIO_SLOT2;
```

```
    sDataConfig.CKPT3SLT = PSIO_SLOT_DISABLE;
    sDataConfig.CKPT4SLT = PSIO_SLOT_DISABLE;
    sDataConfig.CKPT5SLT = PSIO_SLOT_DISABLE;
    sDataConfig.CKPT6SLT = PSIO_SLOT_DISABLE;
    sDataConfig.CKPT7SLT = PSIO_SLOT_DISABLE;

    /* Actions 0 ~ 7 */
    sDataConfig.CKPT0ACT = PSIO_OUT_HIGH;
    sDataConfig.CKPT1ACT = PSIO_OUT_BUFFER;
    sDataConfig.CKPT2ACT = PSIO_OUT_LOW;
    sDataConfig.CKPT3ACT = PSIO_NO_ACTION;
    sDataConfig.CKPT4ACT = PSIO_NO_ACTION;
    sDataConfig.CKPT5ACT = PSIO_NO_ACTION;
    sDataConfig.CKPT6ACT = PSIO_NO_ACTION;
    sDataConfig.CKPT7ACT = PSIO_NO_ACTION;

    /* Set check point configuration */
    PSIO_SET_CP_CONFIG(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, &sDataConfig);

    /* Set slot count */
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SLOT0, 1);
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SLOT1, 2);
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SLOT2, 1);

    /* Enable repeat slot0 ~ slot2 u32DataSize - 1 times */
    PSIO_SET_SCCTL(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SLOT0, PSIO_SLOT2, u32DataSize
- 1, PSIO_REPEAT_DISABLE);

    /* Set out data width as u32DataSize bit */
    PSIO_SET_WIDTH(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, 0, u32DataSize);

    /* Set out data depth as 1 */
    PSIO_SET_OUTPUT_DEPTH(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, PSIO_DEPTH1);

    /* Set output data to buffer */
    PSIO_SET_OUTPUT_DATA(PSIO, gsPSIO_ARGB2_Cfg.u8DataPin, u32Data);

    /* Trigger slot controller */
    PSIO_START_SC(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl);

    /* Wait for slot controller is not busy */
    while (PSIO_GET_BUSY_FLAG(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl));

    /* Stop slot controller */
    PSIO_STOP_SC(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl);

    /* Disable repeat slot0 ~ slot2 */
    PSIO_SET_SCCTL(PSIO, gsPSIO_ARGB2_Cfg.u8SlotCtrl, PSIO_SLOT_DISABLE,
PSIO_SLOT_DISABLE, 0, PSIO_REPEAT_DISABLE);
}
```

2.2 燈效代碼說明

2.2.1 GRB 循環燈效

PSIO_StripSetRGBColor(uint8_t u8Red, uint8_t u8Green, uint8_t u8Blue) 用來組合單顆燈顏色的 24 bits 資料，可設定單顆燈單一顏色，PIXEL_NUM 代表要設定的 LED 燈數量，利用 Timer 函數 TIMER_Delay() 來調整 LED 燈顯示的間隔時間。

```
int Strip_RGBLoop(void)
{
    uint8_t u8i = 0;

    while (kbhit())
    {
        for (u8i = 0; u8i < PIXEL_NUM; u8i++)
        {
            PSIO_StripSetRGBColor(0xFF, 0x00, 0x00);
        }

        TIMER_Delay(TIMER0, 100000);

        for (u8i = 0; u8i < PIXEL_NUM; u8i++)
        {
            PSIO_StripSetRGBColor(0x00, 0xFF, 0x00);
        }

        TIMER_Delay(TIMER0, 100000);

        for (u8i = 0; u8i < PIXEL_NUM; u8i++)
        {
            PSIO_StripSetRGBColor(0x00, 0x00, 0xFF);
        }

        TIMER_Delay(TIMER0, 100000);
    }

    PSIO_StripClear();

    return 0;
}
```

```
void PSIO_StripSetRGBColor(uint8_t u8Red, uint8_t u8Green, uint8_t u8Blue)
{
    int32_t i32Data = 0;

    u8Red   /= (256 / ((uint16_t) gsPSIO_ARGB2_Cfg.u8Brightness + 1));
    u8Green /= (256 / ((uint16_t) gsPSIO_ARGB2_Cfg.u8Brightness + 1));
    u8Blue  /= (256 / ((uint16_t) gsPSIO_ARGB2_Cfg.u8Brightness + 1));

    i32Data |= (u8Green << 16);
    i32Data |= (u8Red << 8);
    i32Data |= u8Blue;
```

```
    PSIO_SendDataWithSize(24, i32Data);
}
```

2.2.2 彩虹燈

PSIO_StripSetPixelColor(uint16_t n, uint32_t GRBColor) 函數用來存放每顆燈的顏色，PIXEL_NUM 為欲設定燈的數量， PSIO_StripWheel(uint8_t wheelPos) 函數用來設定顏色存放位置，用以產生輪轉效果，PSIO_StripShow() 函數用來輸出 BIT0、BIT1 波形至 ARGB2 燈條，以實現彩虹燈效果。

```
int Strip_RainbowCycle(void)
{
    uint16_t u16i = 0, u16j = 0;

    /* 5 cycles of all colors on wheel */
    for (u16j = 0; u16j < (256 * 5); u16j++)
    {
        for (u16i = 0; u16i < PIXEL_NUM; u16i++)
        {
            PSIO_StripSetPixelColor(u16i, PSIO_StripWheel(((u16i * 256 / PIXEL_NUM) +
u16j) & 255));
        }

        PSIO_StripShow();

        PSIO_StripReset();
    }

    PSIO_StripClear();

    return 0;
}
```

2.2.3 走馬燈

前半段代碼與彩虹燈一樣，PSIO_StripSetPixelColor(uint16_t n, uint32_t GRBColor) 函數用來存放每顆燈的顏色，PIXEL_NUM 為欲設定燈的數量，PSIO_StripWheel(uint8_t wheelPos) 函數用來設定顏色存放位置，用以產生輪轉效果，PSIO_StripShow() 函數用來輸出 BIT0、BIT1 波形至 ARGB2 燈條，最後再用一個 for 迴圈，透過 PSIO_StripSetPixelColor(uint16_t n, uint32_t GRBColor) 函數讓其一段時間間隔無顏色，以實現走馬燈效果。

```
int Strip_TheaterChaseRainbow(void)
{
    uint16_t u16i = 0, u16j = 0;
    uint8_t u8q = 0;

    /* cycle all 256 colors in the wheel */
    for (u16j = 0; u16j < 256; u16j++)
```

```
    {
        for (u8q = 0; u8q < 3; u8q++)
        {
            for (u16i = 0; u16i < PIXEL_NUM; u16i = u16i + 3)
            {
                PSIO_StripSetPixelColor(u16i + u8q, PSIO_StripWheel((u16i + u16j) % 255));
            }

            PSIO_StripShow();

            PSIO_StripReset();

            for (u16i = 0; u16i < PIXEL_NUM; u16i = u16i + 3)
            {
                /* turn every third pixel off */
                PSIO_StripSetPixelColor(u16i + u8q, 0);
            }
        }
    }

    PSIO_StripClear();

    return 0;
}
```

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 5.28.0.0

3.2 硬體需求

- 電路元件
  - NuMaker-M252KG V1.0
  - ARGB2 LED 燈條
- 示意圖
  - 使用 PSIO 腳位來模擬 ARGB2 LED 燈條時序圖。

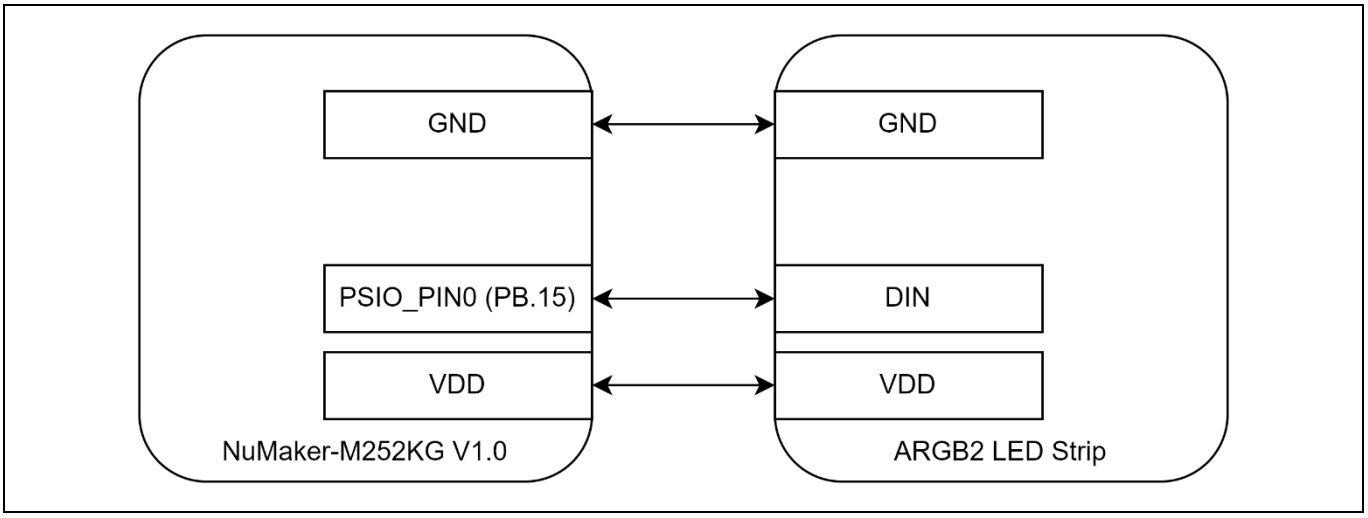

圖 3-1 M252 PSIO_PIN0 和 ARGB2 LED DIN 連接

- ARGB2 LED 燈條工作電壓為 5V，故 NuMaker-M252KG 開發板為 ARGB2 LED 燈條供應電壓必須更換為 5V。可以改變 Nu-Link2-Me 上的 ICE JPR1 (MCU VCC) 電阻使其輸出 5V 電壓，或是使用 USB to TTL module 的 5V PIN。

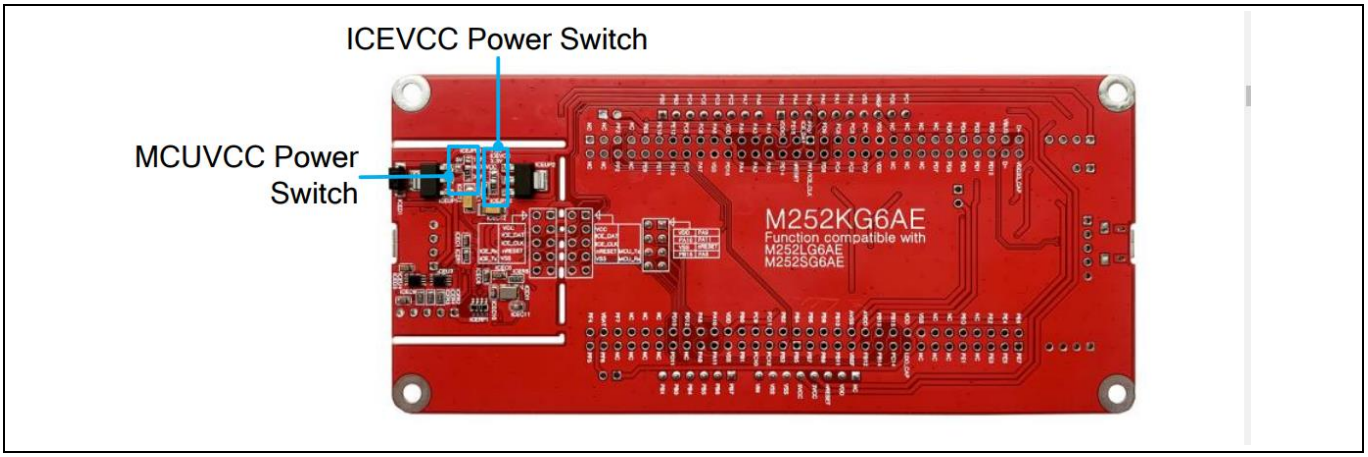

圖 3-2 MCU VCC 切換至 5V

4. 目錄資訊

EC_M252_PSIO_ARGB2_LED_Control_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 *M252_PSIO_ARGB2_LED_Control.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
2022.06.02	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.