# M258 GPIO 實現 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 M258 GPIO 模擬 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-M258KE Ver1.2 |

*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

此範例代碼使用 M251、M252、M254、M256、M258 系列的 GPIO 來模擬第二代燈條 (ARGB2) 的時序，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 用來選擇執行模式。

## 1.1 原理

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

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

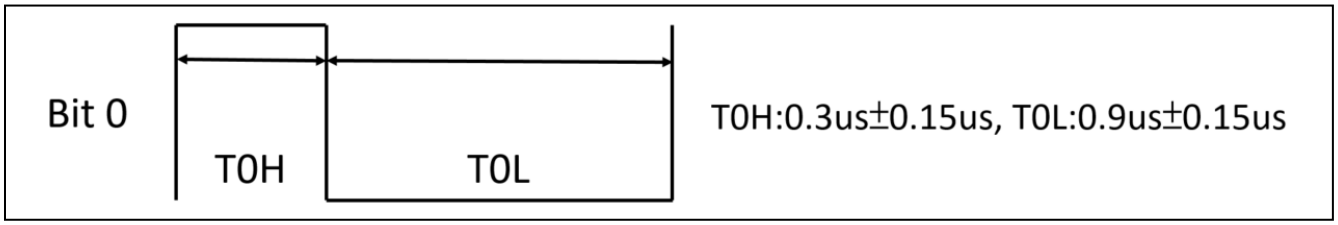


圖 1-1 邏輯’0’ ( Bit 0 )

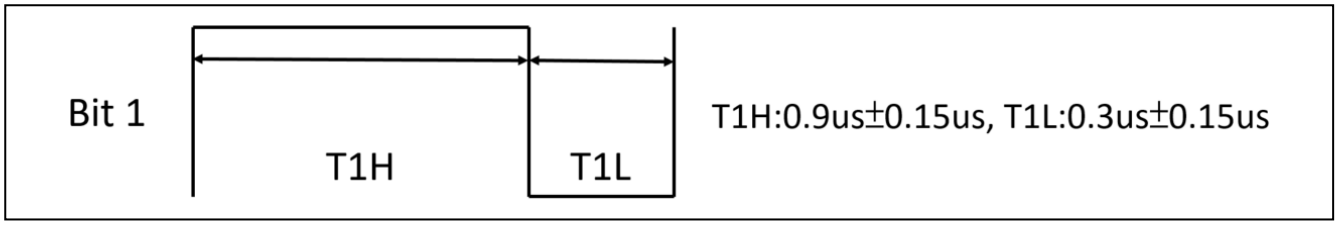


圖 1-2 邏輯’1’ ( Bit 1 )

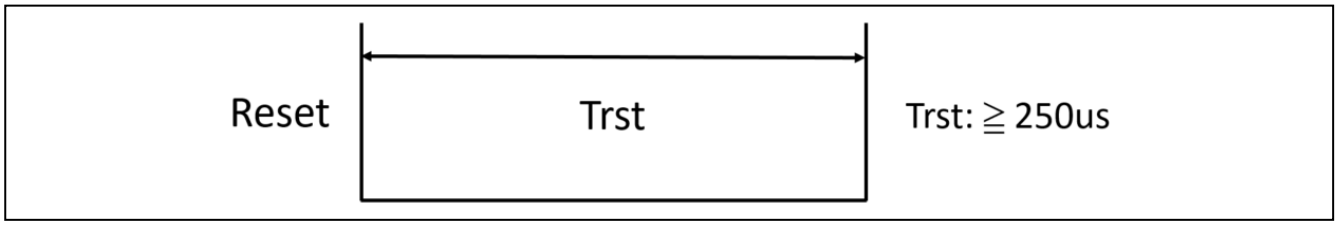


圖 1-3 重置 ( Trst )

### 1.1.2 GPIO 時鐘頻率與邏輯 Bit 設計

使用GPIO 控制 ARGB2 燈條，主要透過 NOP() 指令讓高低準位往後延遲一段時間，程式中設

定系統時鐘 HIRC 為 48MHz，執行一個命令時間約 1 / 48MHz ( 20.83ns )，由執行命令時間即可得到需要執行幾次 NOP() 指令。

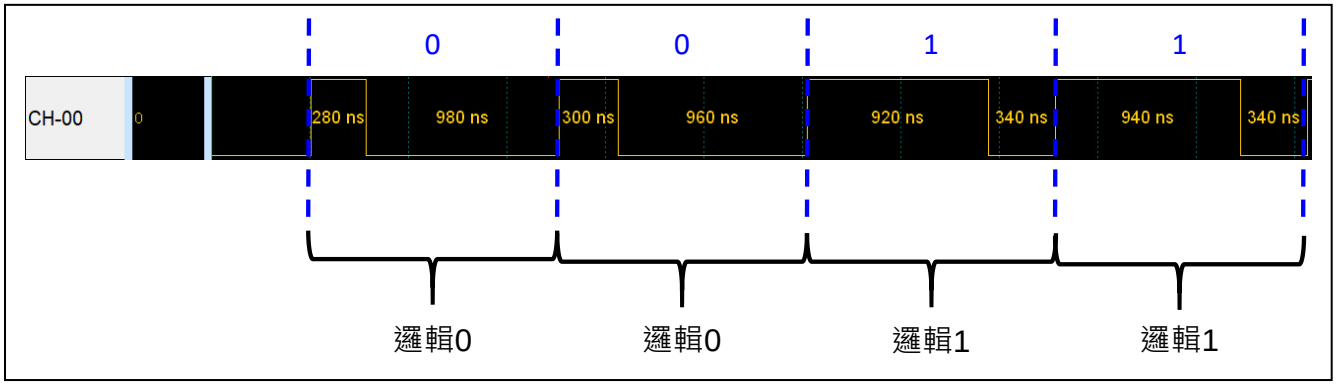


圖 1-4 使用 NOP 指令讓高低準位延遲一段時間

### 1.1.3 Feedback 模式

ARGB2 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：

1. 回傳燈條上 LED 串接顆數 與 LED 消耗電流大小

2. 回傳燈條上 LED 串接顆數 與 目前燈條被設定的 ID 號

Feedback 模式指令波形如圖 1-5，下達 Feedback 模式指令給 ARGB2 燈條後，ARGB2 燈條會回傳Pulse，MCU 必須立刻把 GPIO 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；進入 TH20 SET 模式改變設定可以切換上方兩種回傳內容，請參考1.1.4。

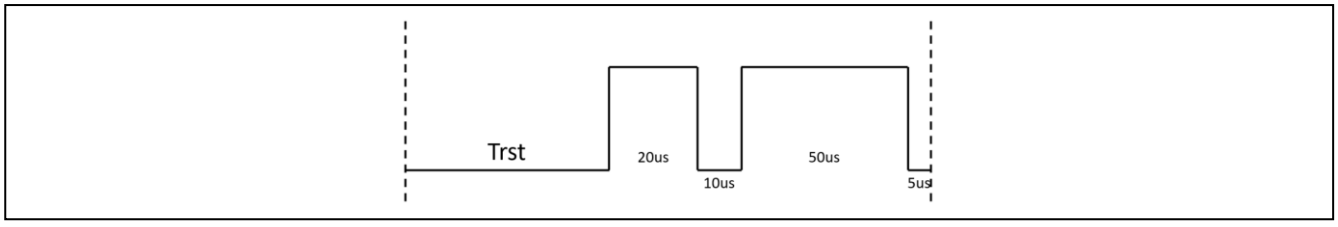


圖 1-5 Feedback 模式波形

Pulse寬度代表意義：

1. 收到 20us Pulse：LED 耗電流為 12mA

2. 收到 10us Pulse：LED 耗電流為 5mA

### 1.1.4 TH20 SET 模式

由 MCU 傳送 20us (高準位) + 250us (低準位) 訊號至 ARGB2 燈條即可進入TH20 SET模式，後面再接續 24 bits ，如圖 1-6，接續在 TH20 SET 模式後設置數據 24 bits 內容與設定，可參考表 1-1、表 1-2，TH20 SET 模式 Dimming 功能可把原最大電流設置更細緻，達到最佳燈效。

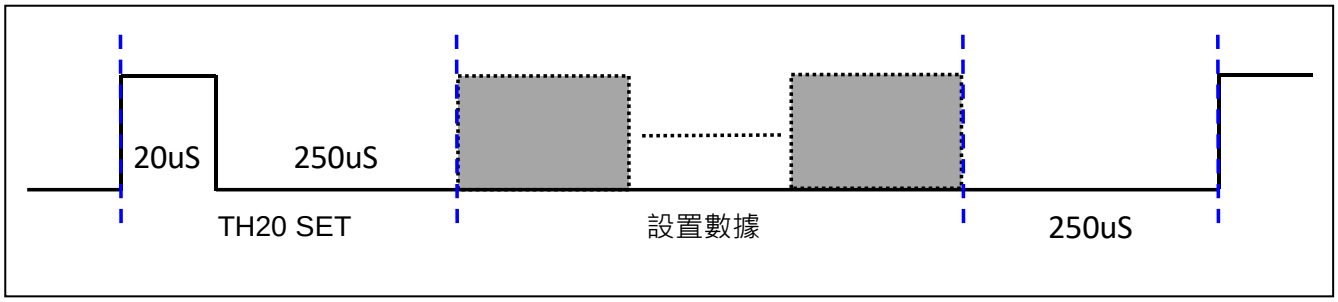


圖 1-6 TH20 SET 模式波形

| 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> | Sel_ CUR | 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 模式後 24 bits 設置數據

| | |
|---|---|
| SS[0] | 設置接收新資料後，PWM是否重新計數，default：0<br>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<br>0：接收到 250us Trst 信號才同步更新<br>1：收滿 24 bits 立即更新 |
| SS[5：4] | 設置 PWM 頻率，default：11<br>00：1.25 KHz　01：2.5 KHz　10：10 KHz　11：20 KHz |
| Sel_CUR | 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<br>1：回傳 LED 顆數 與 目前燈條被設定的ID號 |
| G[4：0] | GLED Dimming 功能，共31階，default：0x1F |
| R[4：0] | RLED Dimming 功能，共31階，default：0x1F |
| B[4：0] | BLED Dimming 功能，共31階，default：0x1F |

表 1-2 TH20 SET 模式後 24 bits 設置數據設定

Dimming功能控制輸出電流公式如下，以 GLED 為例：

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

### 1.1.5 並聯模式 Y Cable 功能

透過 Y Cable 功能可以對 ARGB2 燈條做 ID設定，開啟 Y Cable 功能如圖 1-7，進入前必須有一個 250us (低準位) Pulse 的 Reset 訊號，並聯模式 Y Cable 功能的 Command有四種。

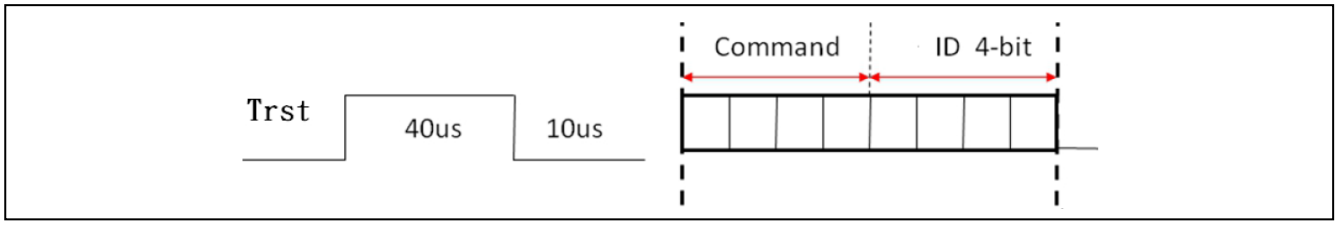


圖 1-7 開啟 Y Cable 功能波形

| | |
|---|---|
| Command[7：4] | 0001：設置燈條 ID 號 (SET_ID)<br>0010：清除燈條 ID 號 (CLR_ID)<br>0011：確認 ID 號是否存在或重複 (CHK_ID)<br>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 功能 Command 設定

◆ 設置 ARGB2 燈條 ID (SET_ID)：

SET_ID + ID 指令傳送完畢後，如設置ID成功，ARGB2 燈條會回傳 Hi Pulse，因此必須立刻將 MCU 的 GPIO 由 Output 切到 Input，用以判斷 ID 設定情況。依此方法將所有並聯的 ARGB2 燈條給予 ID 號，當所有並聯燈條都有 ID 號後再下 SET_ID， ARGB2 燈條就不會回覆 Hi Pulse。

※備註：每次 Power ON 後，ARGB2 燈條的 ID 都會變成 0

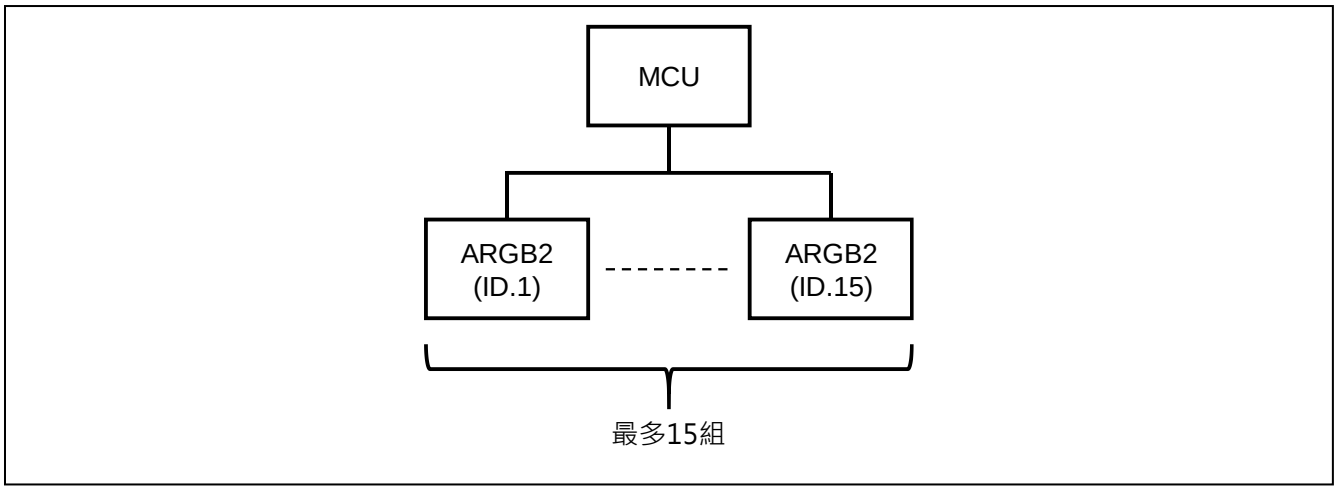


圖 1-8 SET_ID 示意圖

◆ 清除 ARGB2 燈條 ID (CLR_ID)：

當 ID = 0000 時可以清除所有 ARGB2 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 燈條 ID 皆為 0。

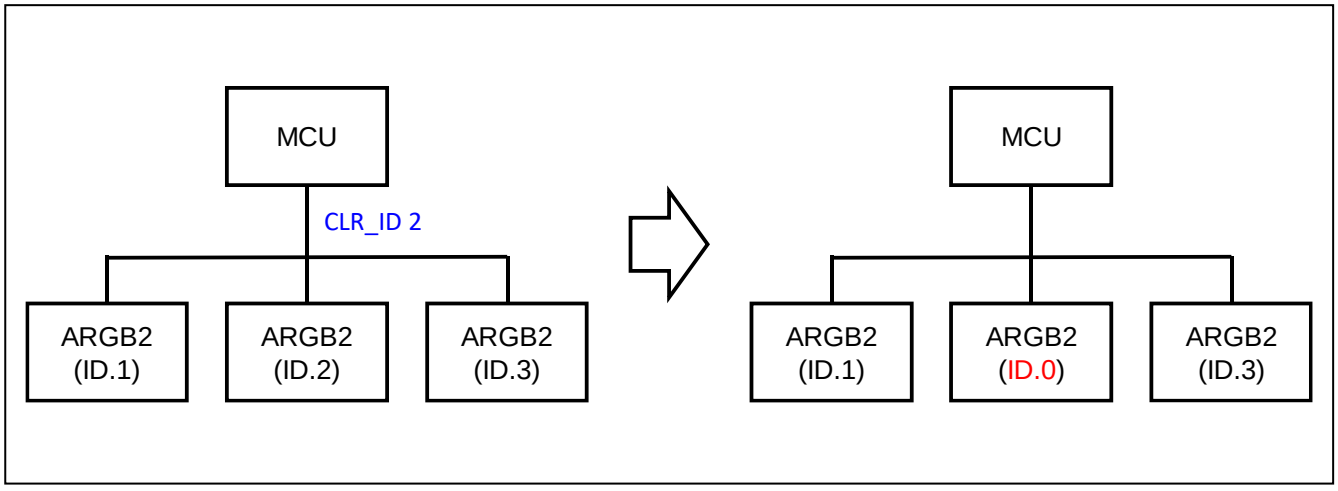


圖 1-9 CLR_ID 示意圖

◆ 確認 ARGB2 燈條 ID (CHK_ID)：

與 SET_ID 指令類似， MCU 傳送 CHK_ID 指令後，如有符合指定的 ID ， ARGB2 燈條會回傳 Hi Pulse，假如並聯的 ARGB2 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 Hi Pulse。

◆ 指定使用 ID 號燈條 (Specify_ID)：

只有被指定到的 ID 燈條會動作，當並聯的 ARGB2 燈條 ID為 0 時則會一起動作。運作原理如圖 1-10。

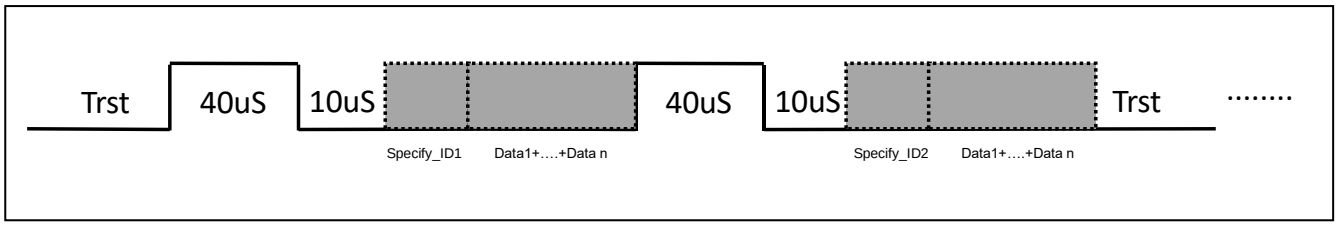


圖 1-10 Specify_ID 示意圖

並聯模式 Y Cable 功能控制流程如下：

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

## 1.2 執行結果

### 1.2.1 邏輯’1’ (Bit 1)、邏輯’0’ (Bit 0)、重置 (Trst) 執行結果

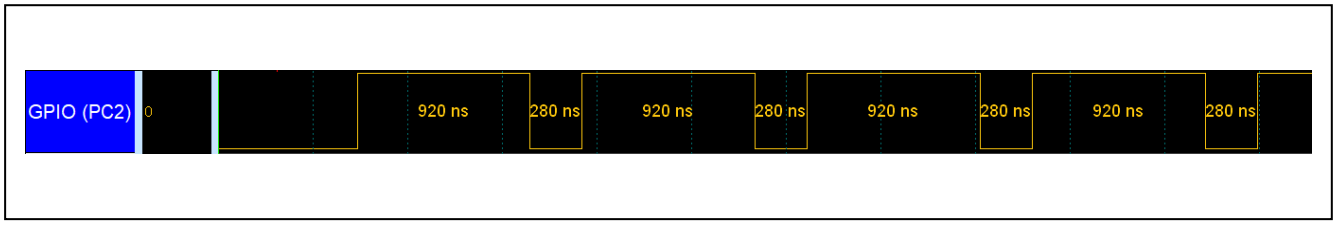


圖 1-11 邏輯’1’ 執行結果

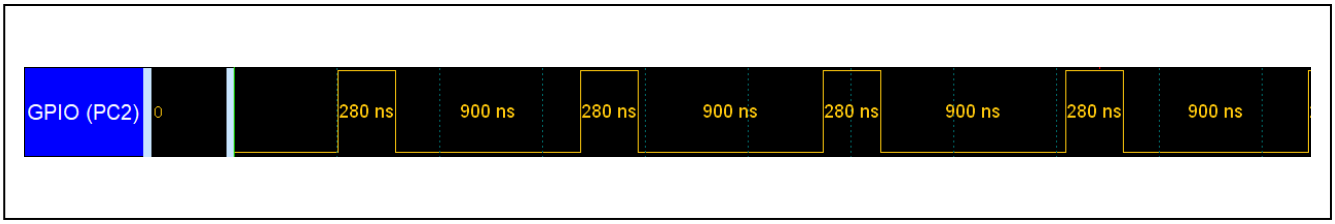


圖 1-12 邏輯'0' 執行結果

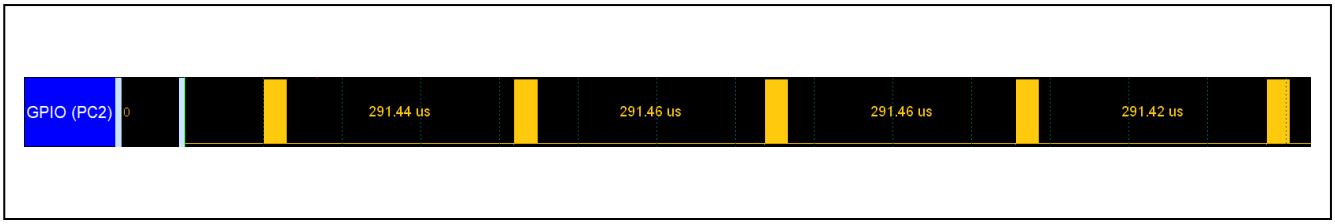


圖 1-13 重置 Trst 執行結果

### 1.2.2 GRB 循環執行結果

GRB循環執行結果如圖 1-14 ~ 圖 1-16，代碼說明請參考 2.2.1。

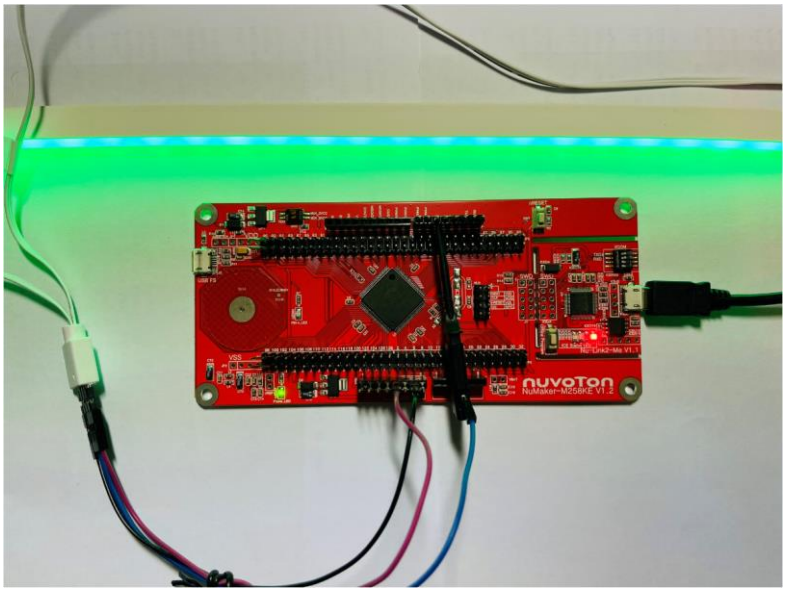

圖 1-14 綠燈執行結果

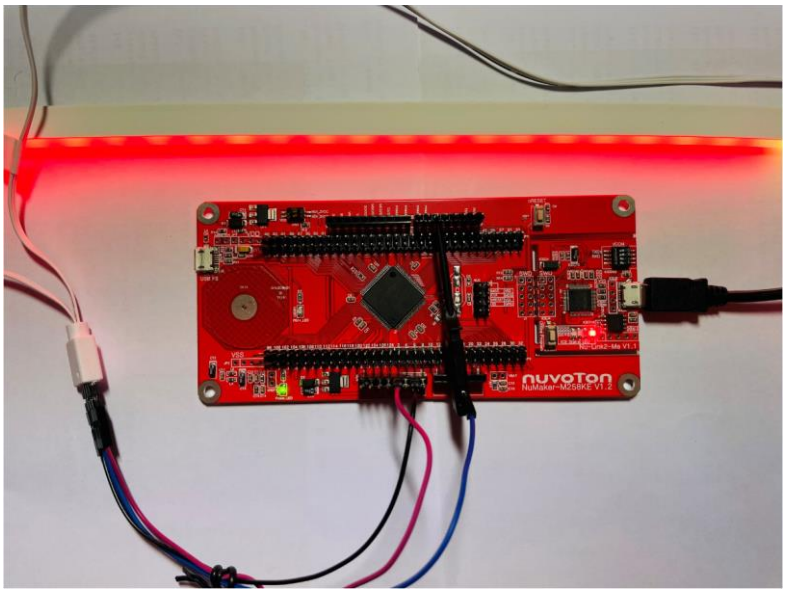

圖 1-15 紅燈執行結果

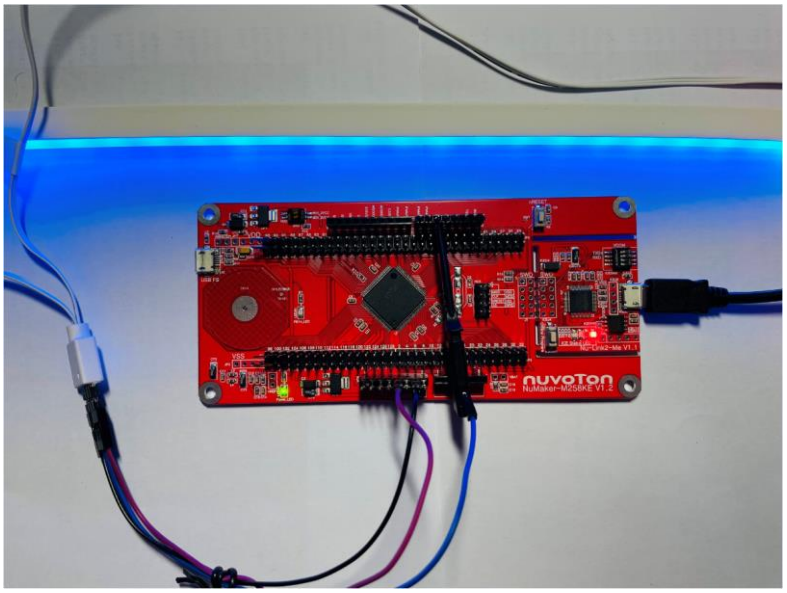

圖 1-16 藍燈執行結果

### 1.2.3 彩虹燈執行結果

彩虹燈執行結果如圖 1-17，代碼說明請參考 2.2.2。

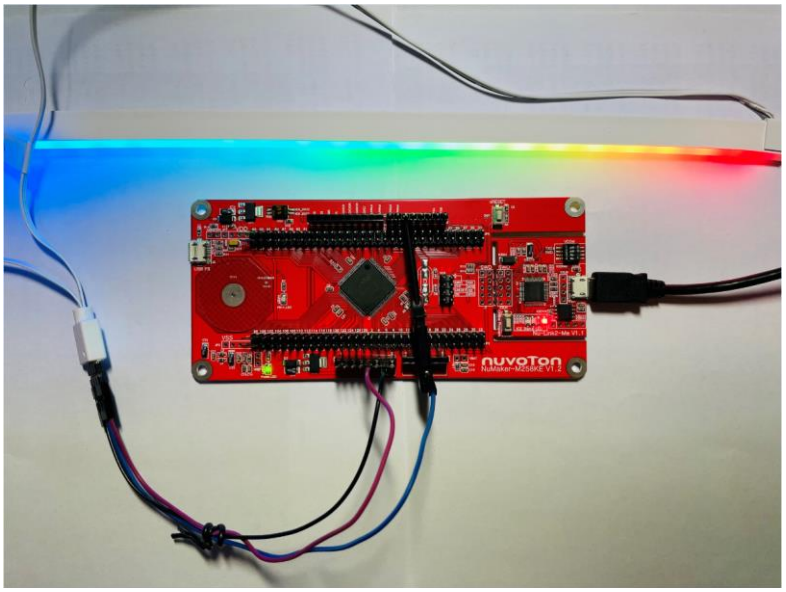

圖 1-17 彩虹燈執行結果

### 1.2.4 走馬燈執行結果

走馬燈執行結果如圖 1-18，代碼說明請參考 2.2.3。

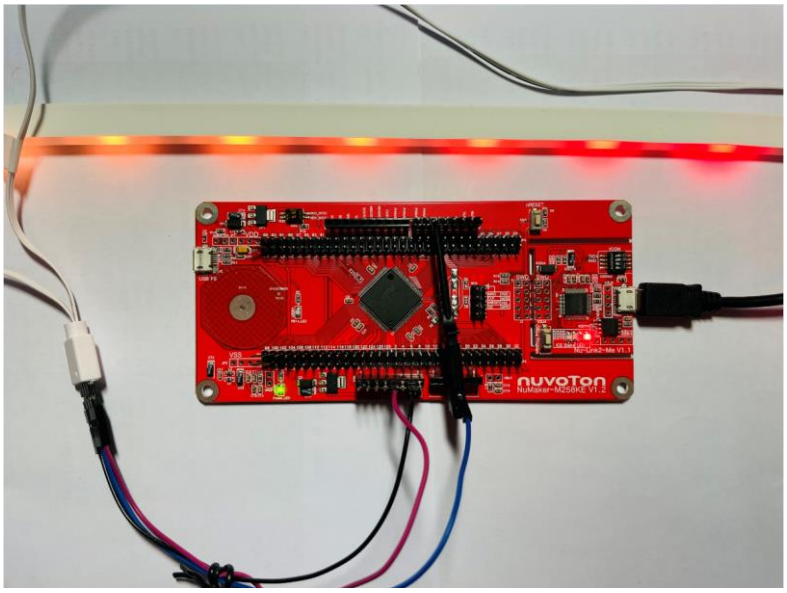

圖 1-18 走馬燈執行結果

### 1.2.5 Feedback 模式執行結果

由 MCU 下達 20us (Hi) + 10us (Lo) + 50us (Hi) 至 ARGB2 燈條開啟 Feedback 模式，波形如

圖 1-19。 ARGB2 燈條接收到開啟 Feedback 模式波形後，會回傳 Pulse 代表一根燈條內有幾顆 LED燈，如圖 1-20，以及單顆 LED 最大耗電流數值，如圖 1-21。

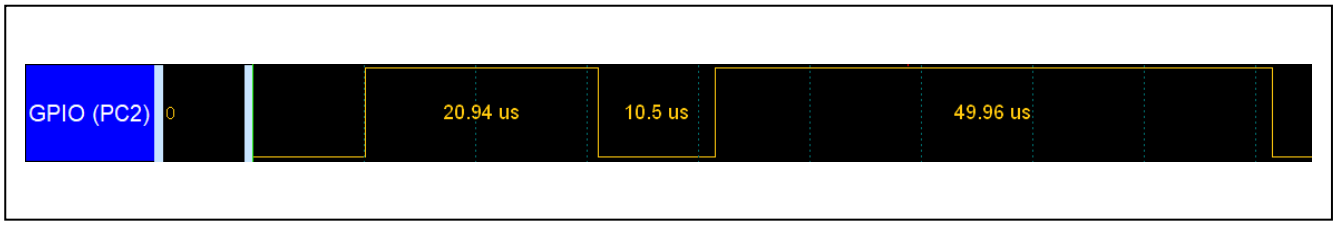


圖 1-19 開啟 Feedback 模式波形

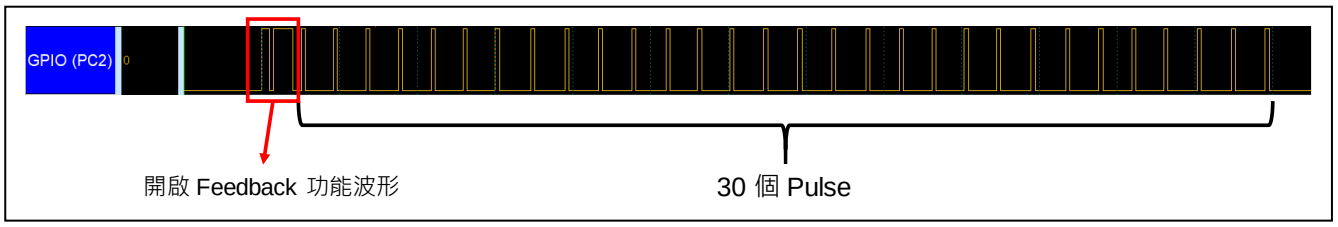


圖 1-20 ARGB2 燈條回傳 Pulse

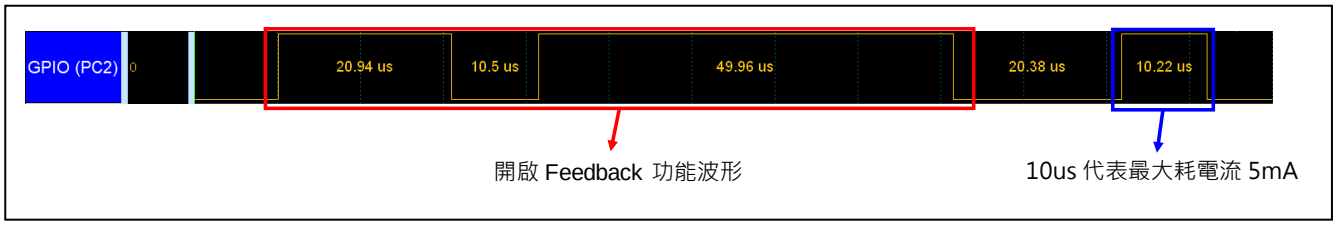


圖 1-21 Pulse 寬度代表最大耗電流

### 1.2.6 TH20 SET 模式執行結果

由 MCU 下達 20us (Hi) + Trst (Lo) 至 ARGB2 燈條開啟 TH20 SET 模式，波形如圖 1-22，設置 24 bits 如圖 1-23。

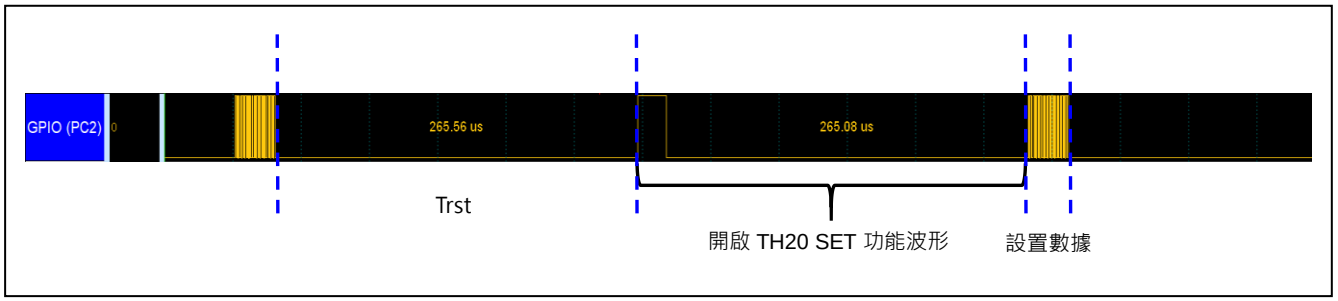


圖 1-22 開啟 TH20 SET 模式波形

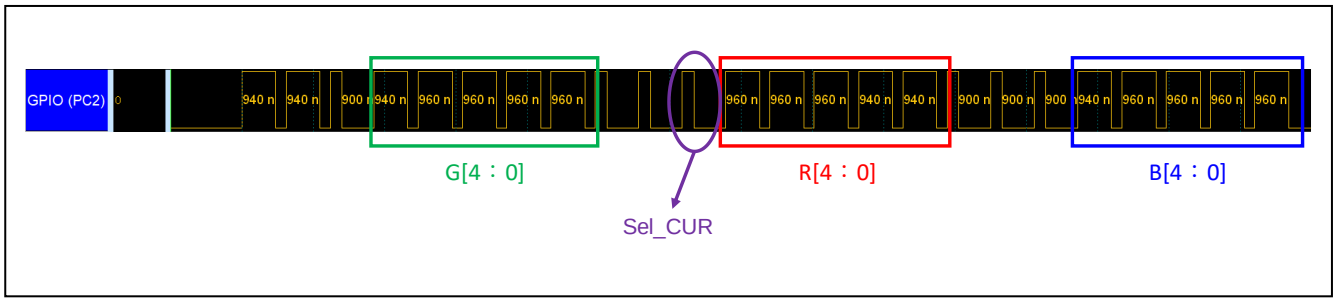


圖 1-23 設置 24 bits 資料

### 1.2.7 並聯模式 Y Cable 功能執行結果

由 MCU 下達 40us (Hi) + 10us (Lo) + 8 bits (Command + ID) 至 ARGB2 燈條開啟 Y Cable 模式，波形如圖 1-24，Command 波形如圖 1-25 ~ 圖 1-30。

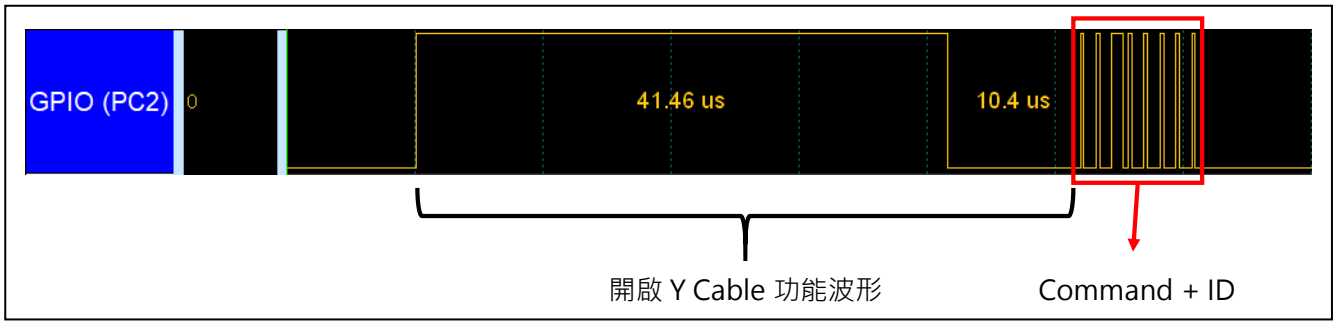


圖 1-24 開啟 Y Cable 功能波形

1. 設置 ARGB2 燈條 ID (SET_ID) 測試結果：

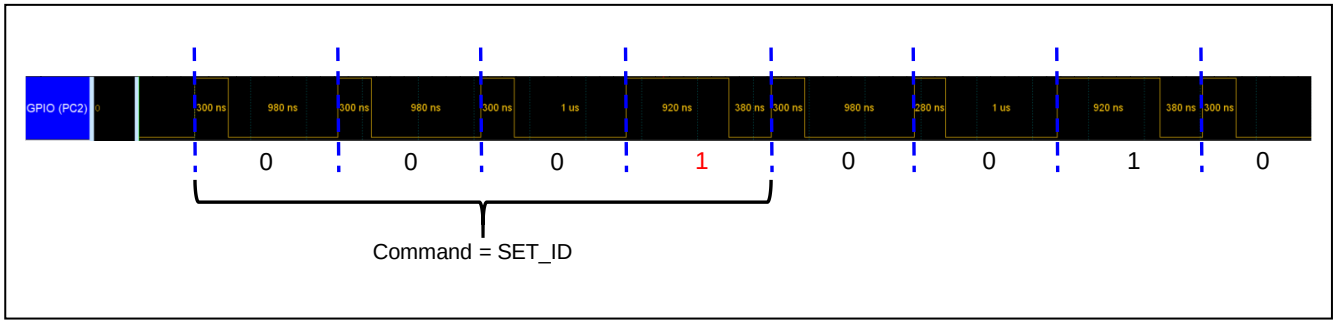


圖 1-25 設置 ARGB2 燈條 ID (SET_ID) 波形

圖 1-26 設置 ARGB2 燈條 ID 成功

2. 清除 ARGB2 燈條 ID (CLR_ID) 測試結果：

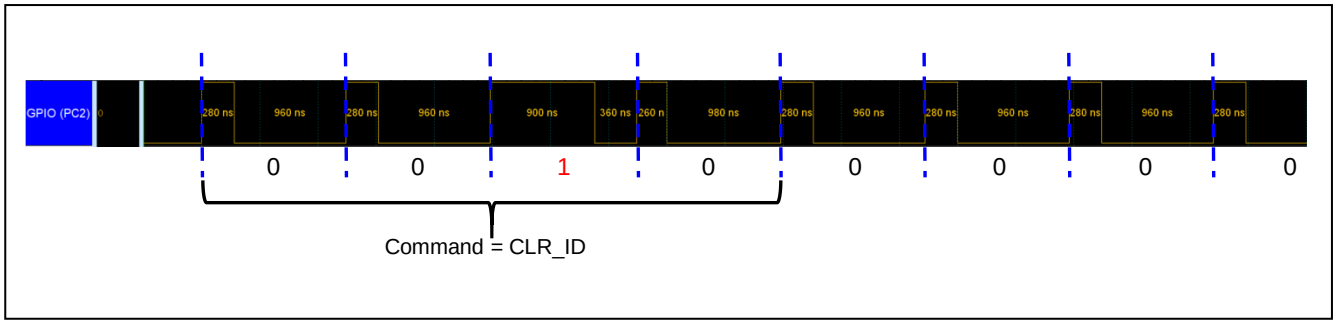


圖 1-27 清除 ARGB2 燈條 ID (CLR_ID) 波形

3. 確認 ARGB2 燈條 ID (CHK_ID) 測試結果：

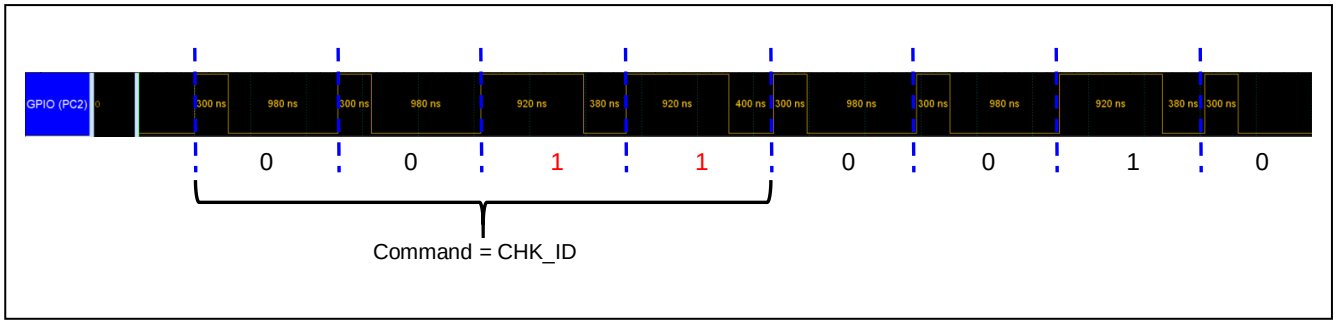


圖 1-28 確認 ARGB2 燈條 ID (CHK_ID) 波形

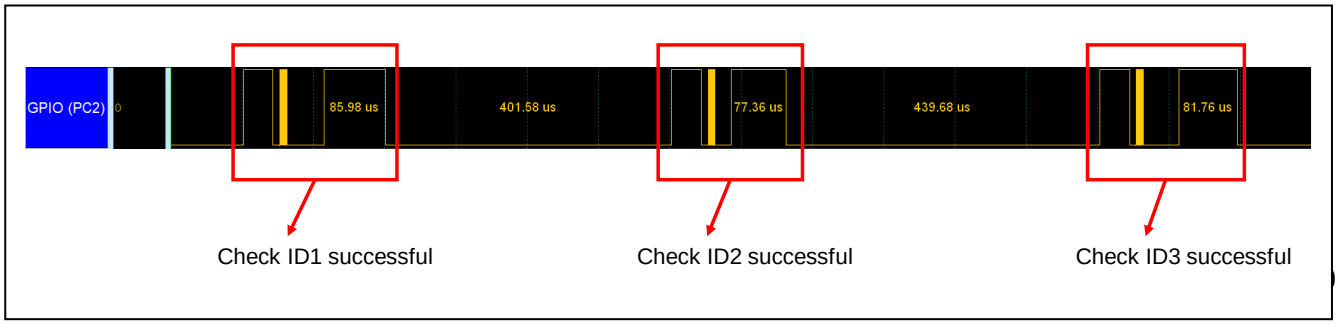

圖 1-29 確認 ARGB2 燈條 ID 成功

4. 指定使用 ID 號燈條 (Specify_ID) 測試結果：

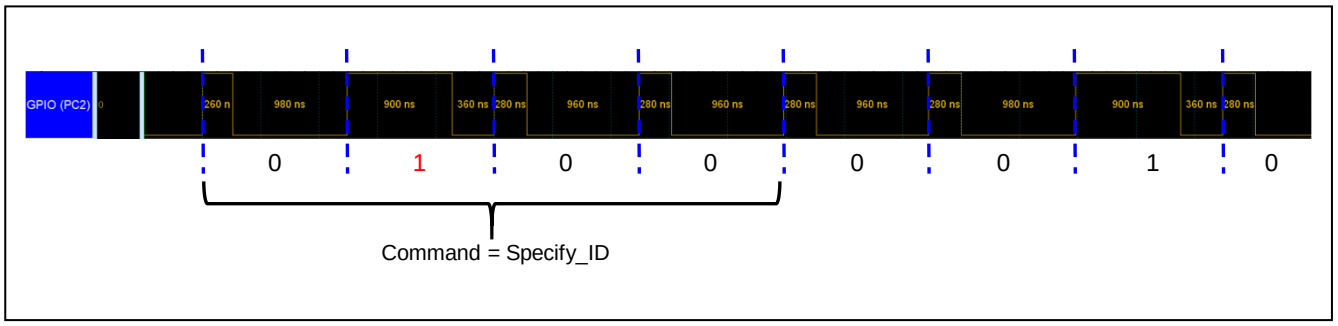


圖 1-30 指定使用 ID 號燈條 (Specify_ID)

### 1.2.8 終端軟體選擇模式執行結果

使用終端軟體透過 UART 選擇執行模式，鮑率請設定為115200，終端軟體執行結果如圖 1-31，可選擇模式如下，其中 0 ~ 2 為燈效顯示，3 ~ 5為開啟燈條不同模式：

0. 執行輪流顯示綠色、紅色、藍色燈效

1. 執行彩虹燈效果

2. 執行走馬燈效果 (以彩虹方式)

3. 執行 Feedback 模式

4. 執行 TH20 SET 模式

5. 執行 Y Cable 模式

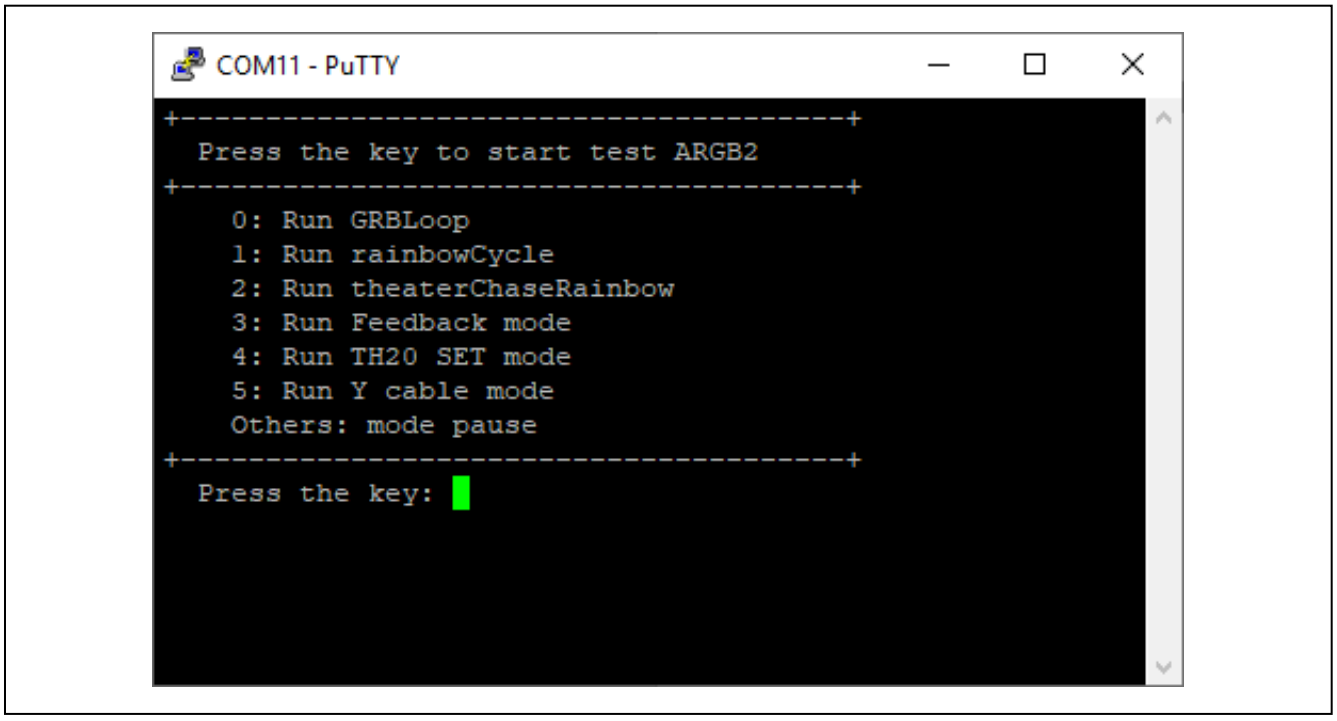


圖 1-31 終端軟體顯示畫面

# 2 代碼介紹

## 2.1 主代碼說明

代碼工作流程如下：

1. main.c 中初始 M251/M252/M254/M256/M258 系列相關外設，包含 HIRC、GPIO、UART0、Timer 0

2. while() 迴圈中使用 switch…case 選擇執行彩虹燈、走馬燈、Feedback 模式、TH20 SET 模式、並聯模式 Y Cable 功能

3. 邏輯'0'、邏輯'1'波形使用 NOP() 指令實現

```
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();

    /* Init Strip */
    Stripe_Init();

    /* Init display of first show in terminal*/
    TerminalShow_Init();

    while(1)
```

```
    {
        switch(MODE)
        {
            case '0':                                       //Run GRBLoop
                Strip_GRBLoop();
                break;
            case '1':                                       //Run rainbowCycle
                Strip_rainbowCycle();
                break;
            case '2':                                       //Run theaterChaseRainbow
                Strip_theaterChaseRainbow();
                break;
            case '3':                                       //feedback mode
                Strip_FeedbackMode();
                FeedbackMode_waitFeedback();
                break;
            case '4':                                       //TH20 SET mode
                Strip_Th20SetMode();
                break;
            case '5':                                       //Y cable mode
                Strip_YCableMode();
                break;
            default:
                break;
        }
    }
}
```

邏輯'0'、邏輯'1'波形使用 NOP()指令來實現，請參考如下代碼。

```
void Strip_sendBIT1 (void)
{
    int i = 0;
    int BIT1_HIGHcnt = (((SystemClock / 100) * BIT1_HIGHTime) / 10000000) - 5;
    int BIT1_LOWcnt = (((SystemClock / 100) * BIT1_LOWTime) / 10000000) - 5;

    Dout_Pin = 1;
    for(i = 0; i < BIT1_HIGHcnt; i++)
    {
        __NOP();
    }
```

```
    Dout_Pin = 0;
    for(i = 0; i < BIT1_LOWcnt; i++)
    {
        __NOP();
    }
}
void Strip_sendBIT0 (void)
{
    int i = 0;
    int BIT0_HIGHcnt = (((SystemClock / 100) * BIT0_HIGHTime) / 10000000) - 5;
    int BIT0_LOWcnt = (((SystemClock / 100) * BIT0_LOWTime) / 10000000) - 5;

    Dout_Pin = 1;
    for(i = 0; i < BIT0_HIGHcnt; i++)
    {
        __NOP();
    }

    Dout_Pin = 0;
    for(i = 0; i < BIT0_LOWcnt; i++)
    {
        __NOP();
    }
}
```

可以在Nuvoton_ARGB2_Strip.h 中對 ARGB2 燈條做初始設定，例如使用哪一個 GPIO 引腳、並聯的ARGB2 燈條數量、ARGB2 燈條內 LED 數量、TH20 SET 模式相關設定、Y Cable 設定。

```
//Basic Setting
#define Dout_Port      PC          //GPIO port of the using Digital output
#define Dout_PinMask   BIT2        //GPIO pin mask of the using Digital output
#define Dout_Pin       PC2         //Digital output pin
#define Dout_IRQn      GPC_IRQn    //GPIO Interrupt
#define Dout_IRQn_Pin  2           //GPIO Interrupt pin

#define STRIP_NUM      15          //ARGB2 strip's number
#define PIXEL_NUM      30          //LED's number in ARGB2 strip
#define GRB            24          //Total bits of RGB

//Th20 SET Mode Setting
#define TH20_24bits    0xdf1f1f
```

```
#define GLED_Dimming    0x1f        //default: 0x1f, range: 0x00~0x1f
#define RLED_Dimming    0x1f        //default: 0x1f, range: 0x00~0x1f
#define BLED_Dimming    0x1f        //default: 0x1f, range: 0x00~0x1f
#define Sel_CUR         0           //default: 0, 0: feedback the LED's number of strip &
                                                  Current
                                    //           1: feedback the LED's number of strip & ID
```

## 2.2 燈效代碼說明

### 2.2.1 GRB 循環

Strip_sendOne(int dat) 用來設定單顆燈顏色，一次傳送 24 bits，0xff0000 代表綠燈，0x00ff00 代表紅燈，0x0000ff 代表藍燈， PIXEL_NUM 為欲設定燈的數量，利用 Timer 函數 TIMER_Delay() 可以調整綠燈、紅燈、藍燈的間隔時間。

```
void Strip_GRBLoop()
{
    int i = 0;

    for(i = 0; i < PIXEL_NUM; i++)
    {
    Strip_sendOne(0xff0000);
    }

    TIMER_Delay(TIMER0, 1000000);

    for(i = 0; i < PIXEL_NUM; i++)
    {
        Strip_sendOne(0x00ff00);
    }

    TIMER_Delay(TIMER0, 1000000);

    for(i = 0; i < PIXEL_NUM; i++)
    {
        Strip_sendOne(0x0000ff);
    }

    TIMER_Delay(TIMER0, 1000000);
}
```

### 2.2.2 彩虹燈

Strip_setPixelColor(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，Strip_wheel(uint8_t wheelPos) 函數用來設定顏色存放位置，藉以產生輪轉效果，Strip_show() 函數用來輸出 BIT0、BIT1 波形至 ARGB2 燈條，以實現彩虹燈效果。

```
void Strip_rainbowCycle()
{
    int i, j;

    for(j=0; j<256*5; j++) // 5 cycles of all colors on wheel
    {
        for(i=0; i< PIXEL_NUM; i++)
        {
            Strip_setPixelColor(i, Strip_wheel(((i * 256 / PIXEL_NUM) + j) & 255));
        }
        Strip_show();

        Strip_sendReset();
    }
}
```

### 2.2.3 走馬燈

前半段代碼與彩虹燈一樣，Strip_setPixelColor(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，Strip_wheel(uint8_t wheelPos) 函數用來設定顏色存放位置，藉以產生輪轉效果，Strip_show() 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 Strip_setPixelColor(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。

```
void Strip_theaterChaseRainbow()
{
    uint16_t i;
    int j, q;

    for (j=0; j < 256; j++) // cycle all 256 colors in the wheel
    {
        for (q=0; q < 3; q++)
        {
            for (i=0; i < PIXEL_NUM; i=i+3)
```

```
            {
                Strip_setPixelColor(i+q, Strip_wheel( (i+j) % 255));
            }
            Strip_show();

            Strip_sendReset();

            for (i=0; i < PIXEL_NUM; i=i+3)
            {
                Strip_setPixelColor(i+q, 0);        //turn every third pixel off
            }
        }
    }
}
```

# 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 5.26

## 3.2 硬體需求

- 電路元件
  - NuMaker-M258KE Ver1.2
  - ARGB2 燈條
- 示意圖

使用 GPIO 腳位來模擬 ARGB2 燈條時序圖。

| | | |
|---|---|---|
| VCC | ↔ | VCC |
| GPIO (PC.2) | ↔ | Din |
| VDD | ↔ | VDD |
| M258 | | ARGB2 |

圖 3-1 線路連接

※備註：ARGB2 燈條工作電壓為 5 V，故 NuMaker-M258KE 開發板為 ARGB2 供應電壓必須更換為 5V。可以改變Nu-Link2-Me 上的ICEJPR1(MCUVCC)電阻使其輸出5V電壓。

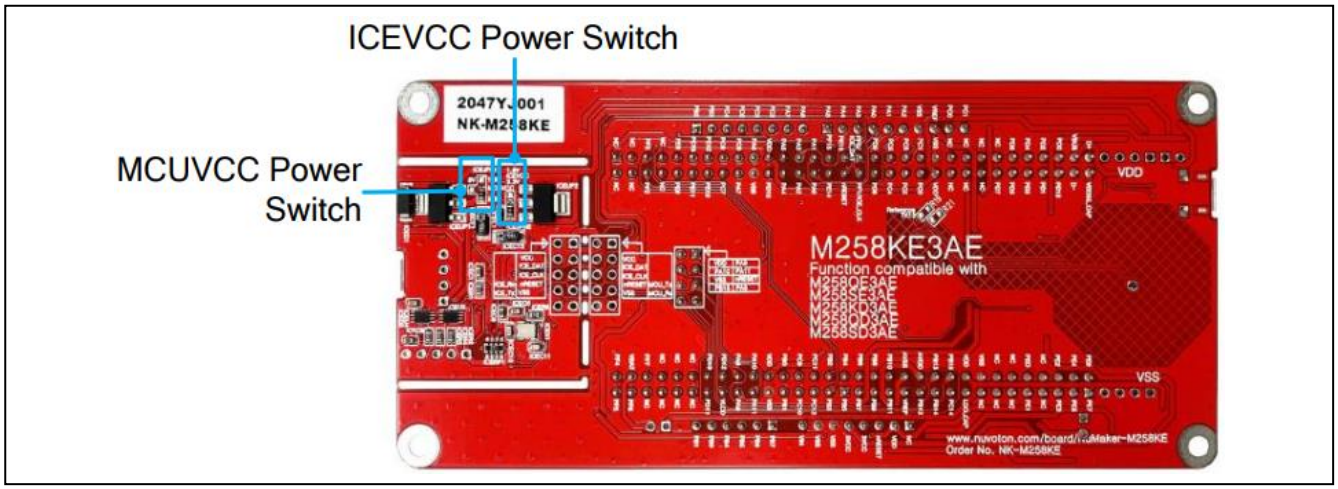


圖 3-2 MCUVCC 電源切換至 5V

# 4 目錄資訊

| | |
|---|---|
| EC_M258_GPIO_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 M258_GPIO_ARGB2_LED_Control.uvprojx。
2. 進入編譯模式介面
   a. 編譯
   b. 下載代碼至記憶體
   c. 進入 / 離開除錯模式
3. 進入除錯模式介面
   a. 執行代碼

# 6 修訂紀錄

| Date | Revision | Description |
|---|---|---|
| 2022.04.01 | 1.00 | 1. 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.*