M258 GPIO 实现 ARGB2 LED 灯效

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码使用 M258 GPIO 模拟 ARGB2 LED 时序并实现彩虹灯、跑马灯等灯效
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
开发平台	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

此范例代码使用 M251、M252、M254、M256、M258 系列的 GPIO 来模拟第二代灯条 (ARGB2) 的时序，并演示 ARGB2 LED 彩虹灯、跑马灯效果。此范例代码使用一组 Timer 来定时更新 ARGB2 LED 彩虹灯、跑马灯的颜色，一组 UART 用来选择执行模式。

1.1 原理

1.1.1 ARGB2 LED 资料传输时间(TH+TL=1.2 us ± 600 ns)

ARGB2 LED 时序图如 图 1-1 与 图 1-2 所示。逻辑 ’0’ 定义为高电平 0.3us ± 0.15us 与低电平 0.9us ± 0.15us，逻辑 ’1’ 定义为高电平0.9us ± 0.15us与低电平 0.3us ± 0.15us。

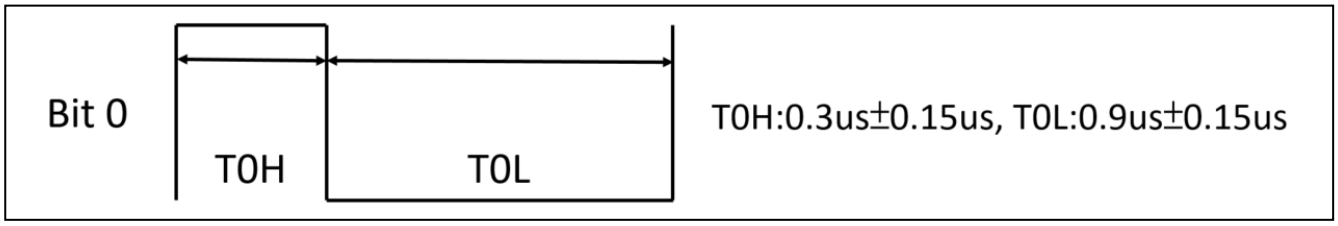

图 1-1 逻辑’0’ (Bit 0)

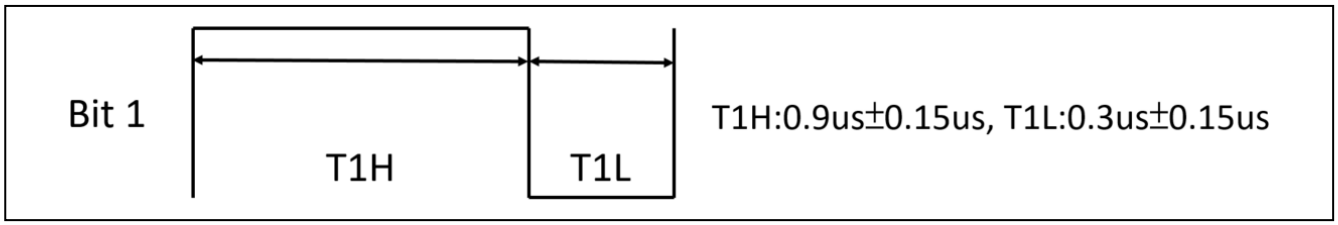

图 1-2 逻辑’1’ (Bit 1)

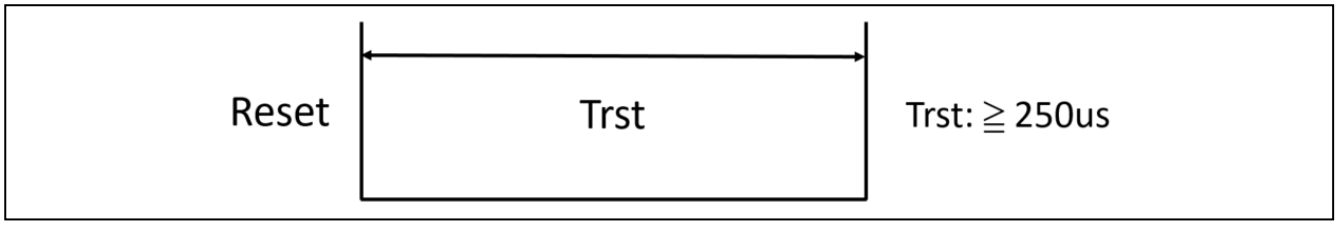

图 1-3 重置 (Trst)

1.1.2 GPIO 时钟频率与逻辑 Bit 设计

使用GPIO 控制 ARGB2 灯条，主要透过 NOP() 指令让高低电平往后延迟一段时间，程式中设定系统时钟 HIRC 为 48MHz，执行一个命令时间约 1 / 48MHz (20.83ns)，由执行命令时间即可得到需要执行几次 NOP() 指令。

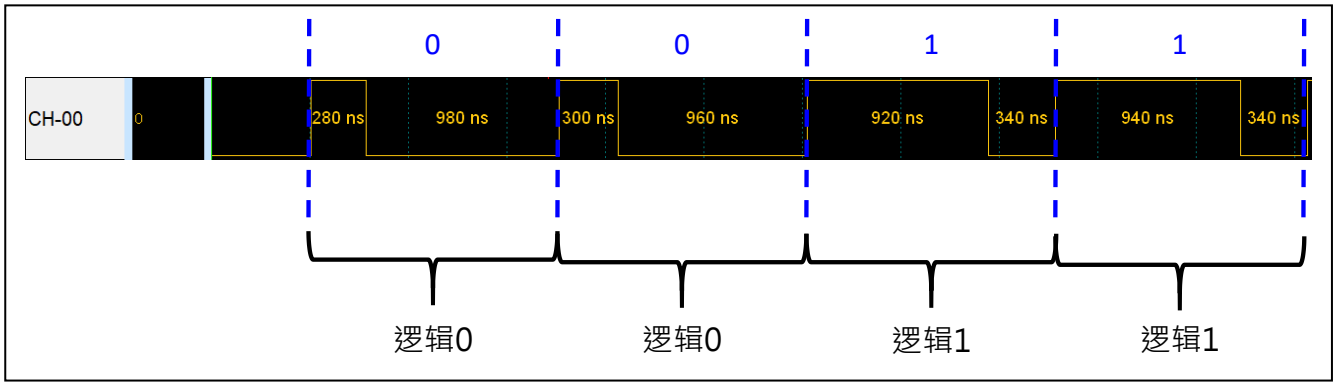

图 1-4 使用 NOP 指令让高低电平延迟一段时间

1.1.3 Feedback 模式

ARGB2 灯条除了能透过 MCU 传递信号做灯效，也可以回传讯息给 MCU，内容有两种：

1. 回传灯条上 LED 串接颗数 与 LED 消耗电流大小

2. 回传灯条上 LED 串接颗数 与 目前灯条被设定的 ID 号

Feedback 模式指令波形如图 1-5，下达 Feedback 模式指令给 ARGB2 灯条后，ARGB2 灯条会回传Pulse，MCU 必须立刻把 GPIO 由 Output 切换成 Input 读取 Pulse，藉由读取 Pulse 的数目及宽度便可得到该灯条上有多少颗 LED 及所使用的最大电流、目前灯条被设定的ID；进入 TH20 SET 模式改变设定可以切换上方两种回传内容，请参考1.1.4。

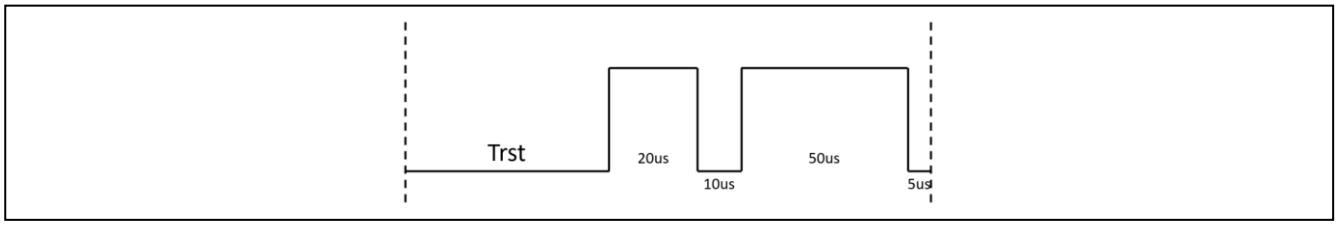

图 1-5 Feedback 模式波形

Pulse宽度代表意义：

1. 收到 20us Pulse：LED 耗电流为 12mA

2. 收到 10us Pulse：LED 耗电流为 5mA

1.1.4 TH20 SET 模式

由 MCU 传送 20us (高电平) + 250us (低电平) 信号至 ARGB2 灯条即可进入TH20 SET模式，后面再接续 24 bits ，如图 1-6，接续在 TH20 SET 模式后设置数据 24 bits 内容与设定，可参考表 1-1、表 1-2，TH20 SET 模式 Dimming 功能可把原最大电流设置更细致，达到最佳灯效。

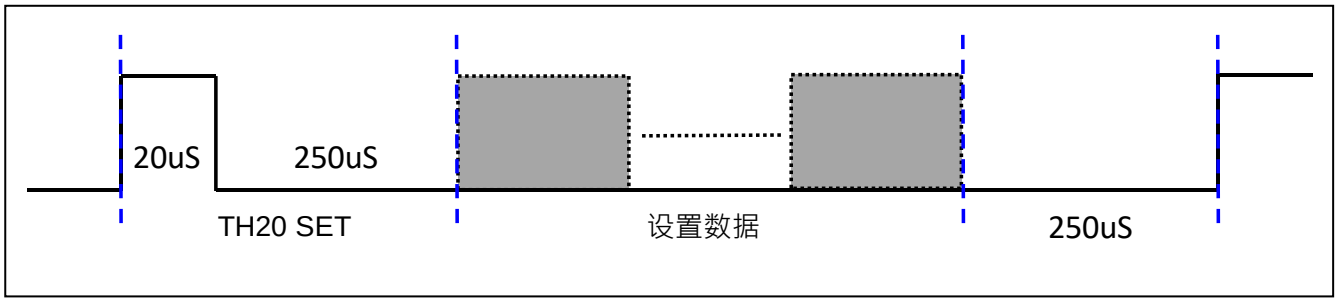

图 1-6 TH20 SET 模式波形

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 模式后 24 bits 设置数据

SS[0]	设置接收新资料后，PWM是否重新计数，default：0 0：不重新计数　1：PWM一律会由 0 开始计数
SS[2：1]	Imax (请勿更动)，default：00
SS[3]	为数据更新时机设置，default：0 0：接收到 250us Trst 信号才同步更新 1：收满 24 bits 立即更新

SS[5：4]	设置 PWM 频率，default：11 00：1.25 KHz　01：2.5 KHz　10：10 KHz　11：20 KHz
Sel_CUR	设置 Feedback 模式回传内容，default：0 0：回传 LED 颗数 与 LED 消耗电流大小 1：回传 LED 颗数 与 目前灯条被设定的ID号
G[4：0]	GLED Dimming 功能，共31阶，default：0x1F
R[4：0]	RLED Dimming 功能，共31阶，default：0x1F
B[4：0]	BLED Dimming 功能，共31阶，default：0x1F

表 1-2 TH20 SET 模式后 24 bits 设置数据设定

Dimming功能控制输出电流公式如下，以 GLED 为例：

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

1.1.5 并联模式 Y Cable 功能

透过 Y Cable 功能可以对 ARGB2 灯条做 ID设定，开启 Y Cable 功能如图 1-7，进入前必须有一个 250us (低电平) Pulse 的 Reset 信号，并联模式 Y Cable 功能的 Command有四种。

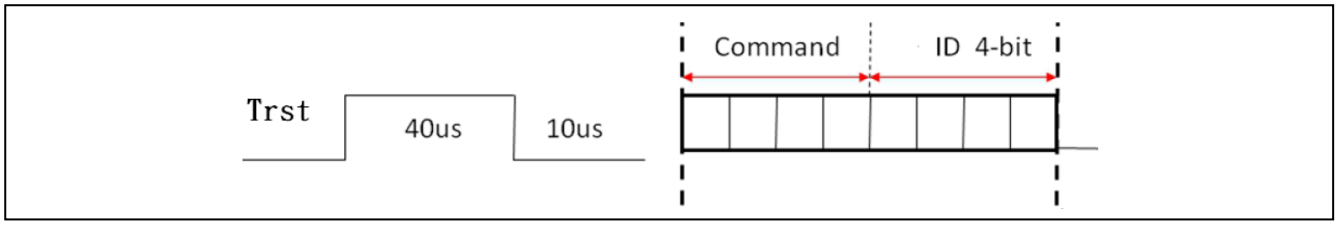

图 1-7 开启 Y Cable 功能波形

Command[7：4]	0001：设置灯条 ID 号 (SET_ID) 0010：清除灯条 ID 号 (CLR_ID) 0011：确认 ID 号是否存在或重复 (CHK_ID) 0100：指定使用 ID 号灯条 (Specify_ID)
ID[3：0]	范围为：0000 ~ 1111

表 1-3 并联模式 Y Cable 功能 Command 设定

◆ 设置 ARGB2 灯条 ID (SET_ID)：

SET_ID + ID 指令传送完毕后，如设置ID成功，ARGB2 灯条会回传 Hi Pulse，因此必须立刻将 MCU 的 GPIO 由 Output 切到 Input，用以判断 ID 设定情况。依此方法将所有并联的 ARGB2 灯条给予 ID 号，当所有并联灯条都有 ID 号后再下 SET_ID， ARGB2 灯条就不会回覆 Hi Pulse。

※备注：每次 Power ON 后，ARGB2 灯条的 ID 都会变成 0

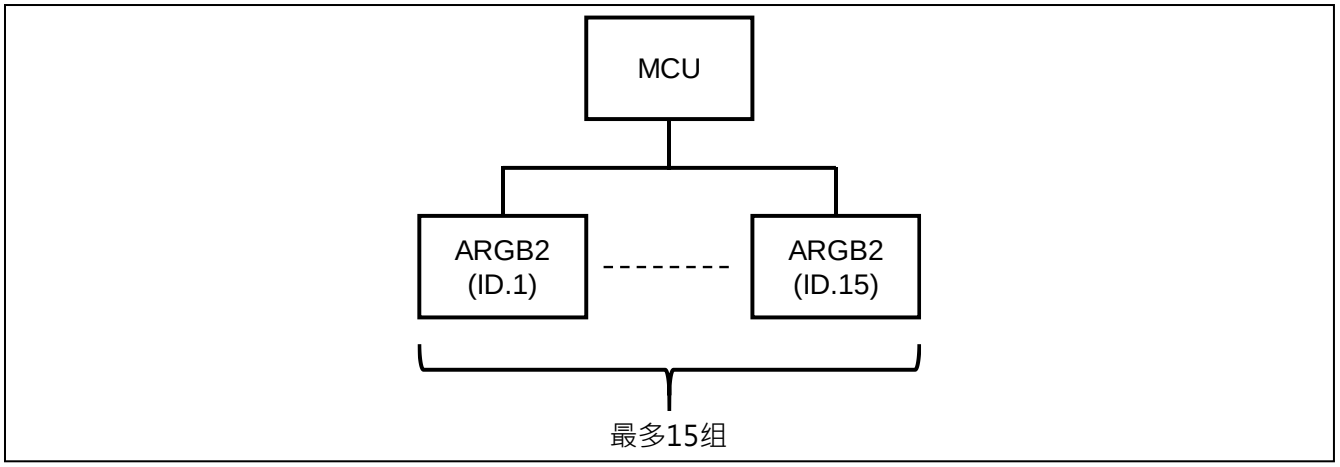

图 1-8 SET_ID 示意图

◆ 清除 ARGB2 灯条 ID (CLR_ID)：

当 ID = 0000 时可以清除所有 ARGB2 灯条 ID，当 ID = 0001 ~ 1111 时可以清除指定 ID 为 0，设置 ID 前建议做一次 CLR_ID + 0000，确保所有并联 ARGB2 灯条 ID 皆为 0。

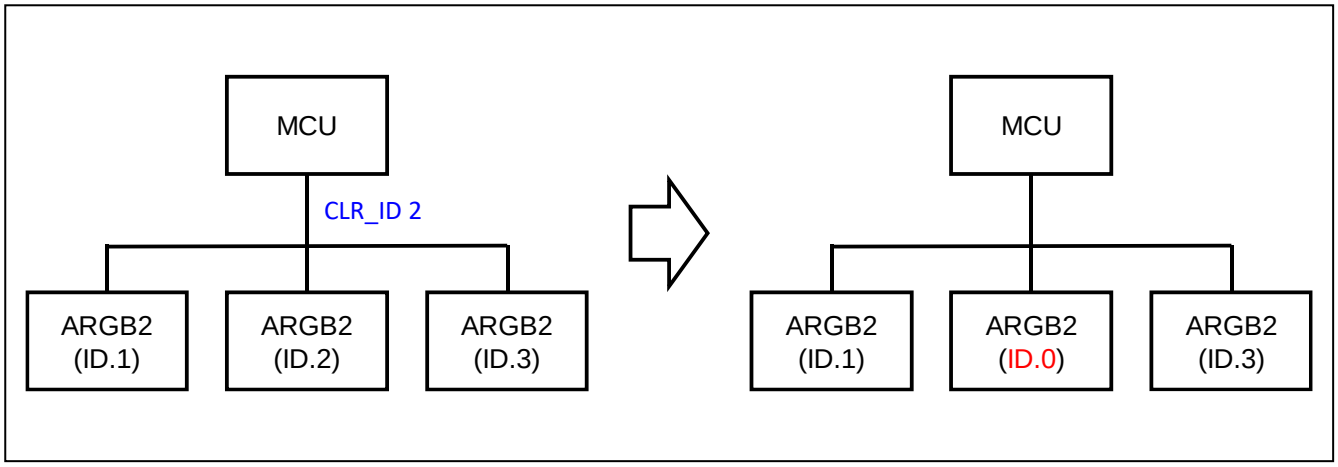

图 1-9 CLR_ID 示意图

◆ 确认 ARGB2 灯条 ID (CHK_ID)：

与 SET_ID 指令类似， MCU 传送 CHK_ID 指令后，如有符合指定的 ID ， ARGB2 灯条会回传 Hi Pulse，假如并联的 ARGB2 灯条有 ID 被重复设定，此时发送 CHK_ID 指令，MCU 会收到被重复设定数量的 Hi Pulse。

◆ 指定使用 ID 号灯条 (Specify_ID)：

只有被指定到的 ID 灯条会动作，当并联的 ARGB2 灯条 ID为 0 时则会一起动作，运作原理如图 1-10。

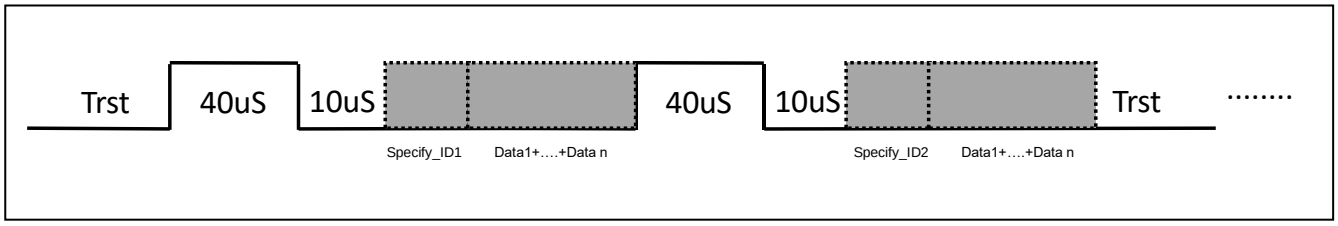

图 1-10 Specify_ID 示意图

并联模式 Y Cable 功能控制流程如下：

1. CLR_ID：把所有并联的 ARGB2 灯条 ID 设定为 0

2. SET_ID：分配不同 ID 给所有 ARGB2 灯条，分配 ID 成功后，ARGB2 灯条会回传 Pulse

3. CHK_ID：对分配到 ID 的 ARGB2 灯条分别做确认，如分配成功，ARGB2 灯条会回传 Pulse

4. Specify_ID：可以指定灯条做控制

1.2 执行结果

1.2.1 逻辑’ 1’ (Bit 1)、逻辑’ 0’ (Bit 0)、重置 (Trst) 执行结果

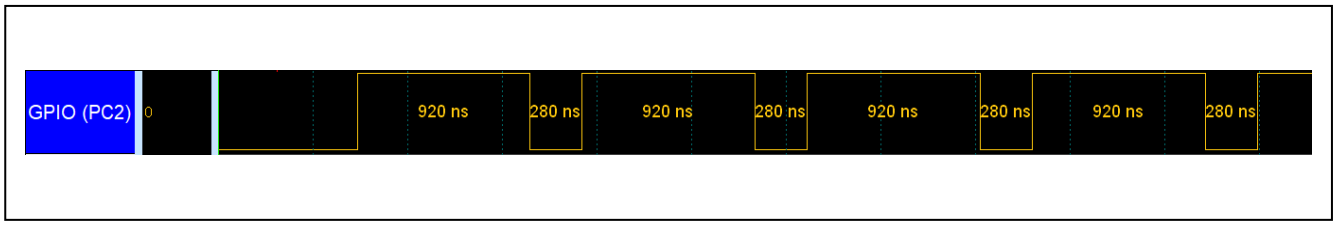

图 1-11 逻辑’1’ 執行結果

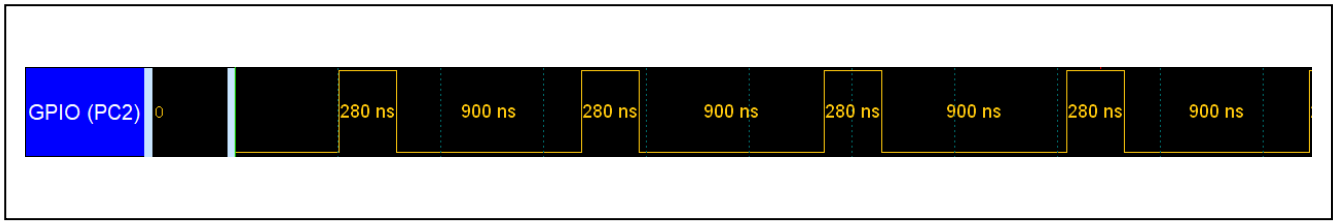

图 1-12 逻辑’0’ 執行結果

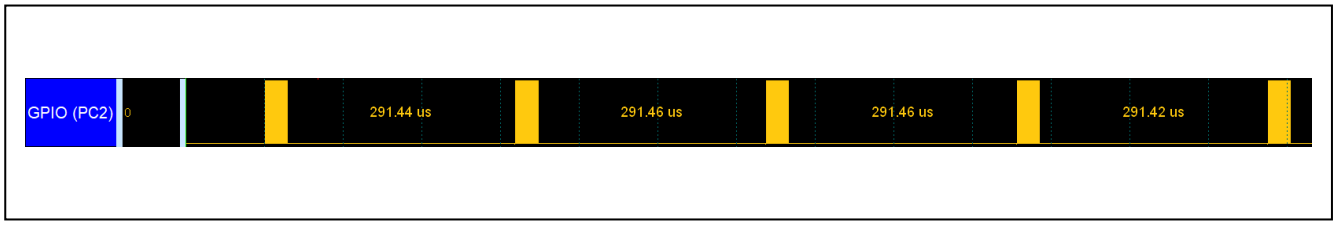

图 1-13 重置 Trst 执行结果

1.2.2 GRB 循环执行结果

GRB循环执行结果如图 1-14 ~ 图 1-16，代码说明请参考 2.2.1。

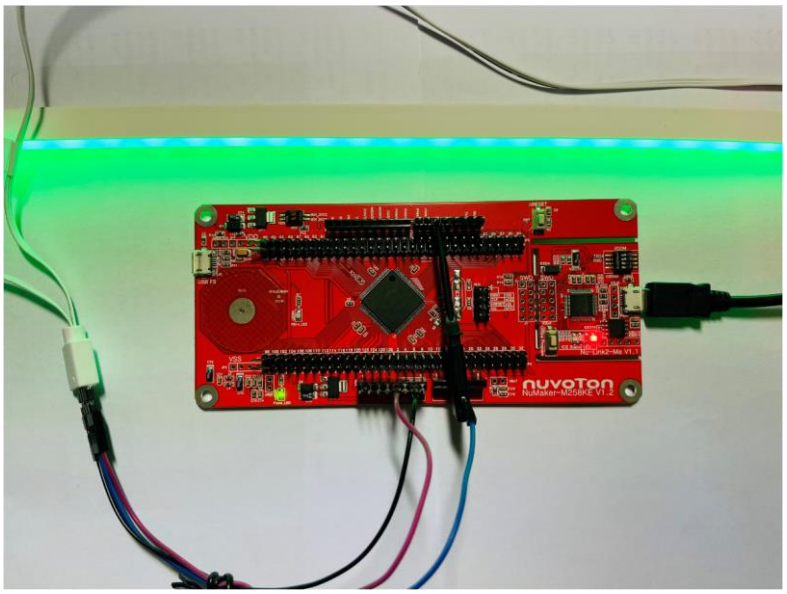

图 1-14 绿灯执行结果

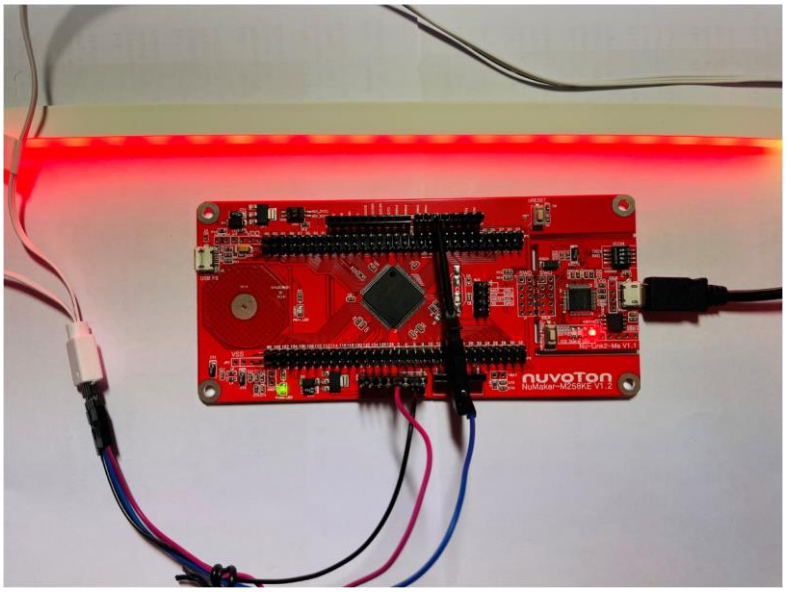

图 1-15 红灯执行结果

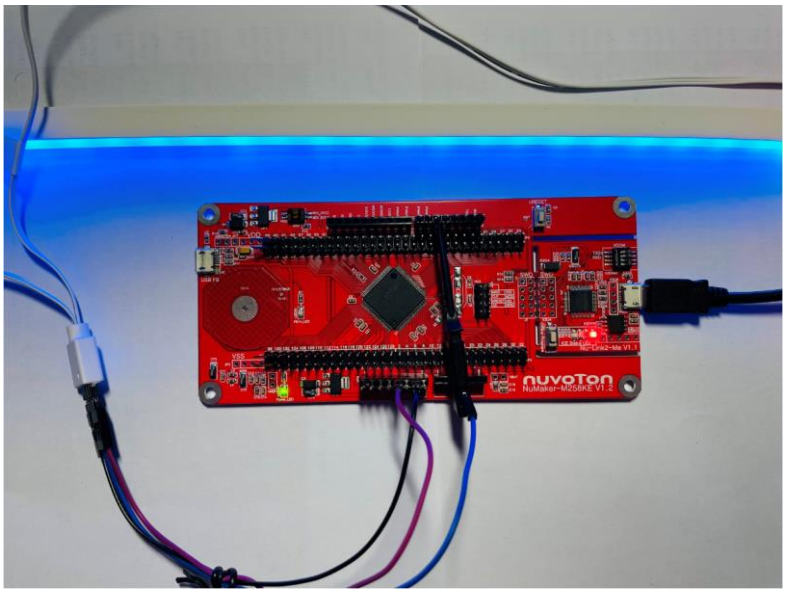

图 1-16 蓝灯执行结果

1.2.3 彩虹灯执行结果

彩虹灯执行结果如图 1-17，代码说明请参考 2.2.2。

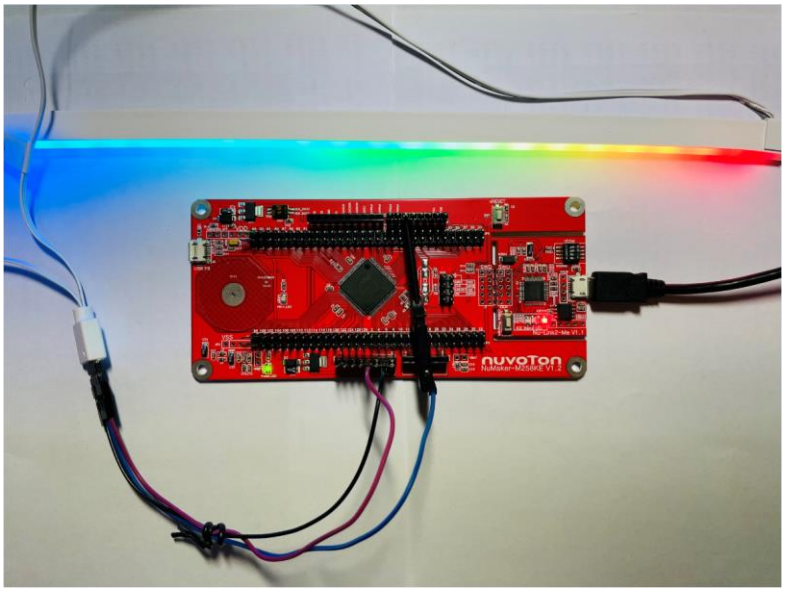

图 1-17 彩虹灯执行结果

1.2.4 跑马灯执行结果

跑马灯执行结果如图 1-18，代码说明请参考2.2.3。

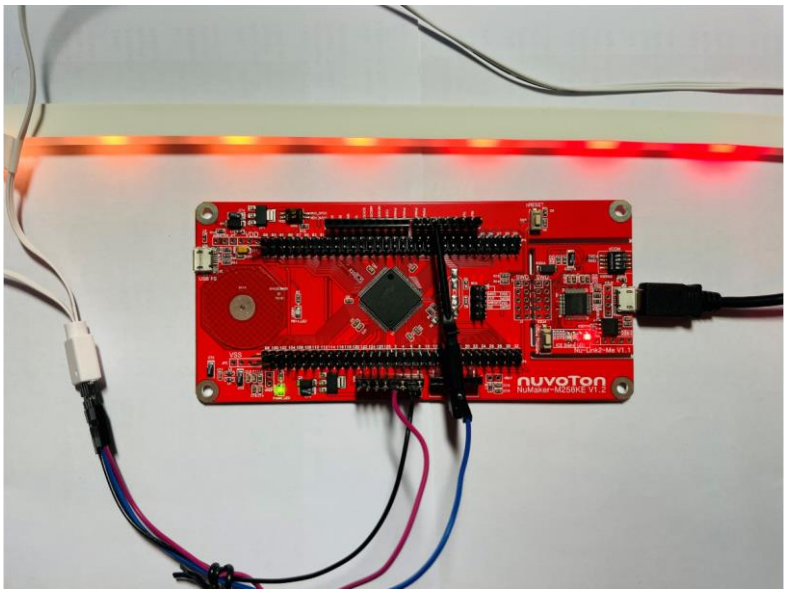

图 1-18 跑马灯执行结果

1.2.5 Feedback 模式执行结果

由 MCU 下达 20us (Hi) + 10us (Lo) + 50us (Hi) 至 ARGB2 灯条开启 Feedback 模式，波形如图 1-19。 ARGB2 灯条接收到开启 Feedback 模式波形后，会回传 Pulse 代表一根灯条内有几颗 LED灯，如图 1-20，以及单颗 LED 最大耗电流数值，如图 1-21。

图 1-19 开启 Feedback 模式波形

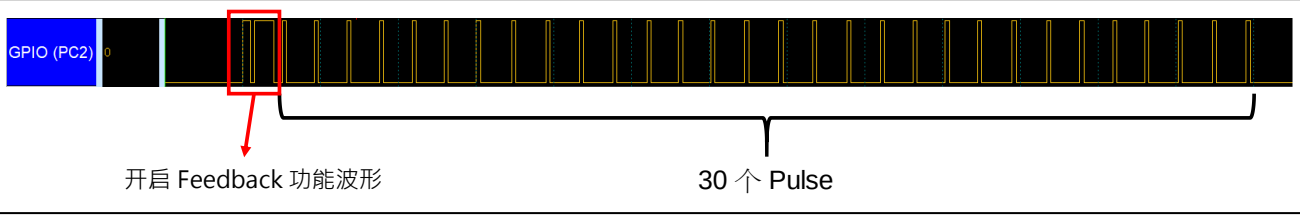

图 1-20 ARGB2 灯条回传 Pulse

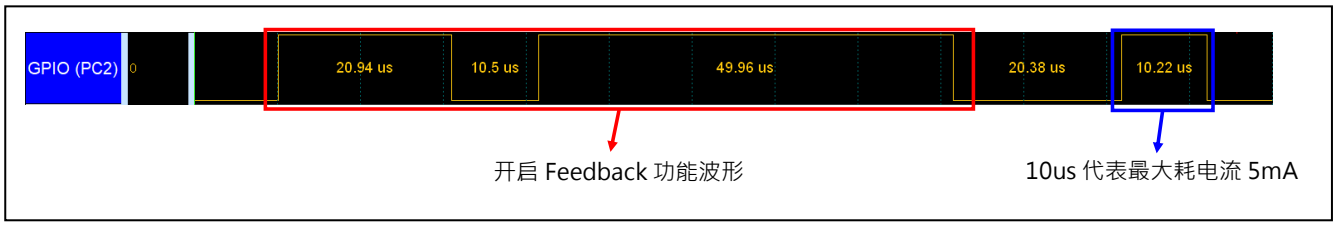

图 1-21 Pulse 宽度代表最大耗电流

1.2.6 TH20 SET 模式执行结果

由 MCU 下达 20us (Hi) + Trst (Lo) 至 ARGB2 灯条开启 TH20 SET 模式，波形如图 1-22，设置 24 bits 如图 1-23。

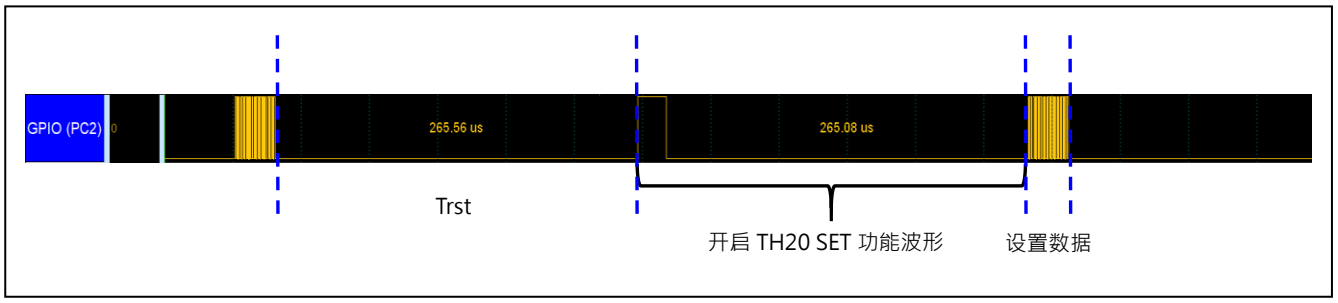

图 1-22 开启 TH20 SET 模式波形

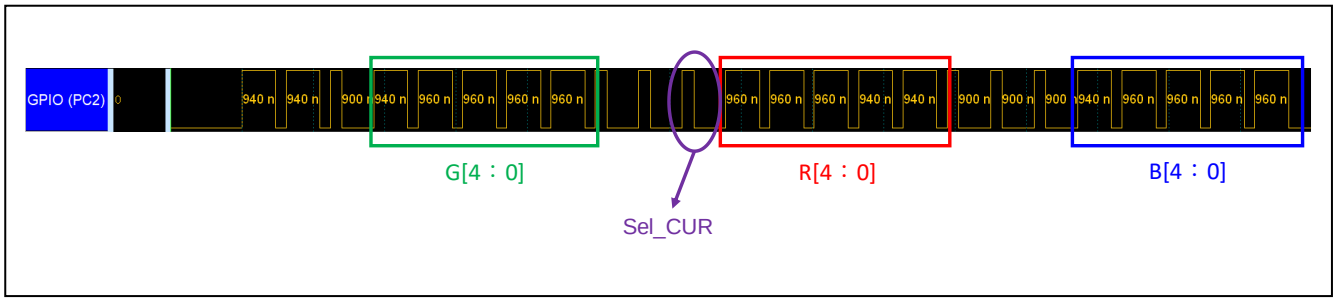

图 1-23 设置 24 bits 资料

1.2.7 并联模式 Y Cable 功能执行结果

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

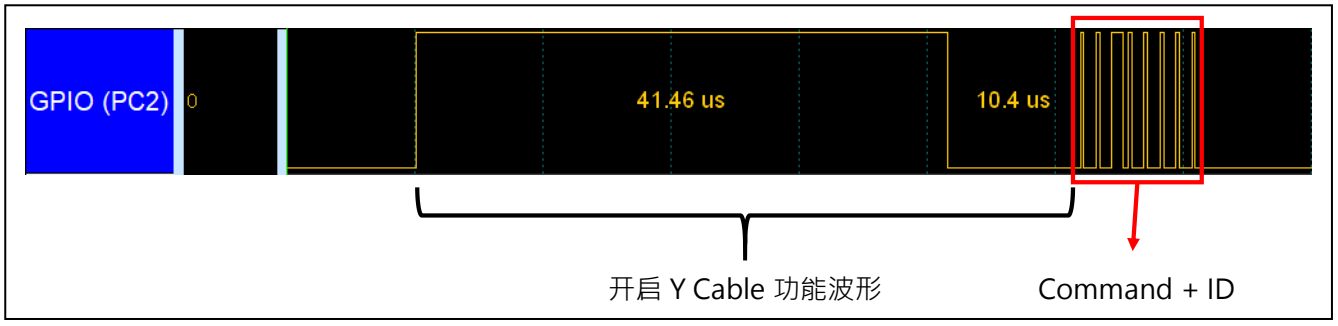

图 1-24 开启 Y Cable 功能波形

1. 设置 ARGB2 灯条 ID (SET_ID) 执行结果：

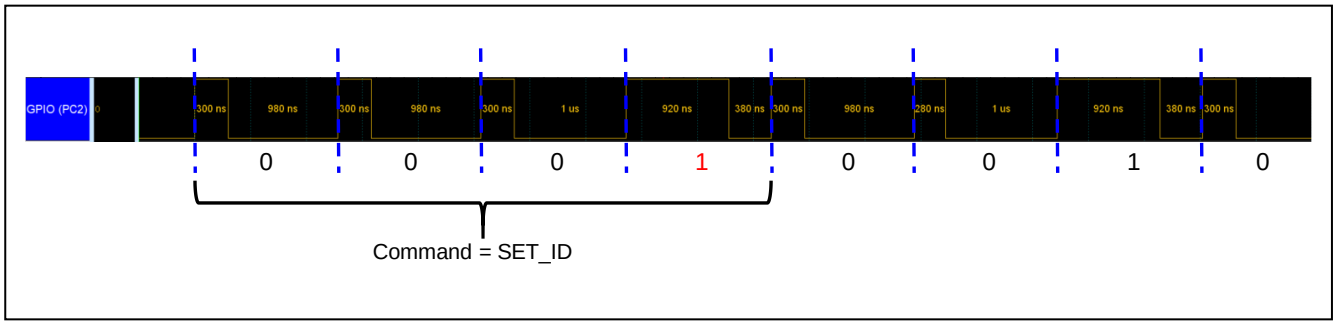

图 1-25 设置ARGB2 灯条 ID (SET_ID) 波形

图 1-26 设置 ARGB2 灯条 ID 成功

2. 清除 ARGB2 灯条 ID (CLR_ID) 执行结果：

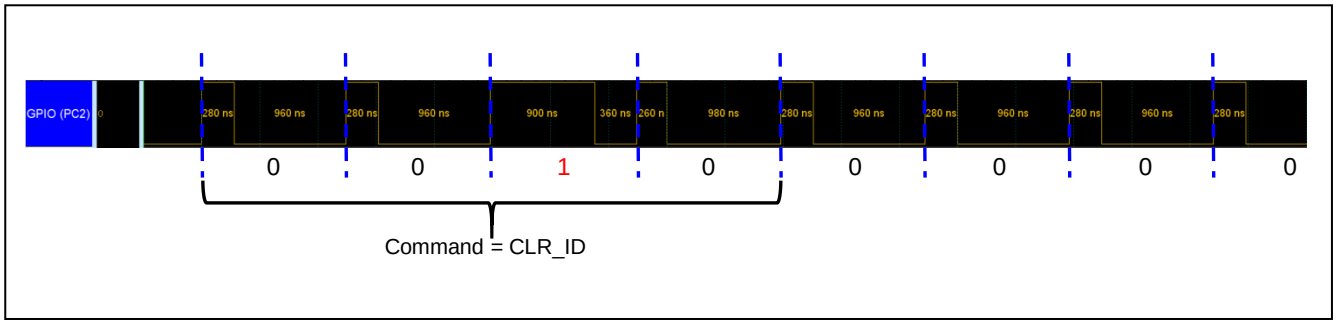

图 1-27 清除 ARGB2 灯条 ID (CLR_ID) 波形

3. 确认 ARGB2 灯条 ID (CHK_ID) 执行结果：

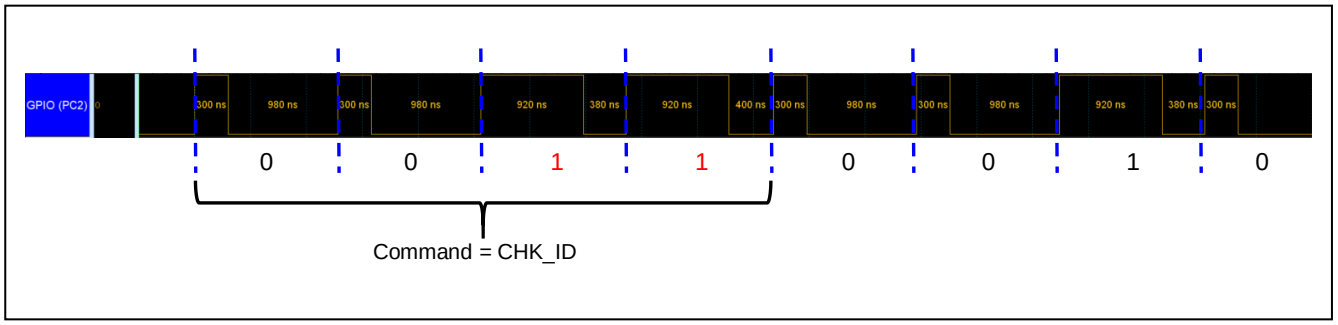

图 1-28 确认 ARGB2 灯条 ID (CHK_ID) 波形

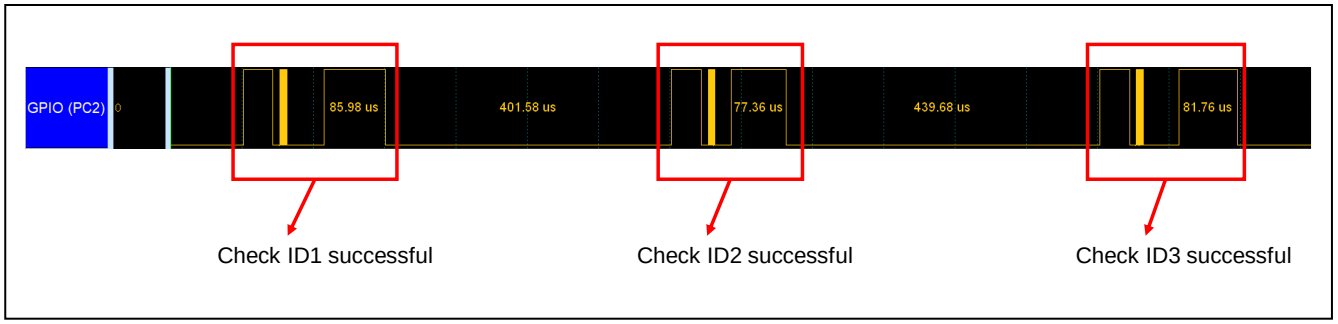

图 1-29 确认 ARGB2 灯条 ID 成功

4. 指定使用 ID 号灯条 (Specify_ID) 执行结果：

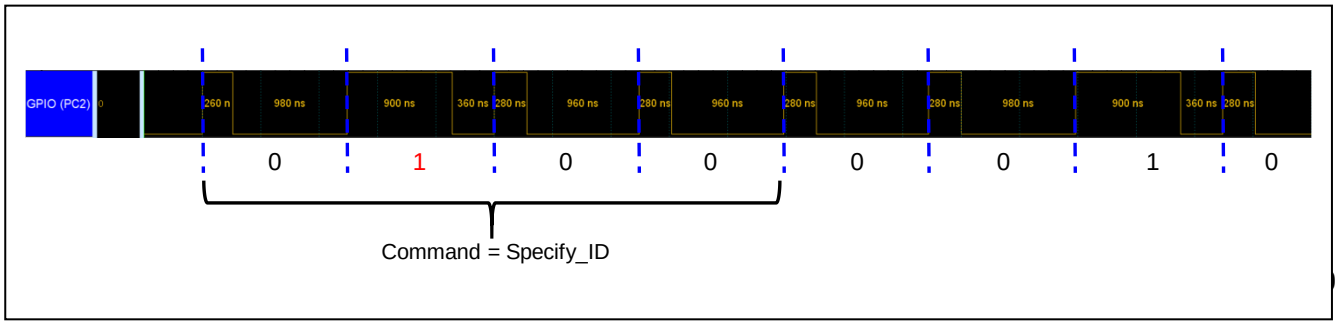

图 1-30 指定使用 ID 号灯条 (Specify_ID)

1.2.8 终端软件选择模式执行结果

使用终端软件透过 UART 选择执行模式，波特率请设定为115200，终端软件执行结果如图 1-31，可选择模式如下，其中 0 ~ 2 为灯效显示，3 ~ 5为开启灯条不同模式：

0. 执行轮流显示绿色、红色、蓝色灯效
1. 执行彩虹灯效果
2. 执行跑马灯效果 (以彩虹方式)
3. 执行 Feedback 模式
4. 执行 TH20 SET 模式
5. 执行 Y Cable 模式

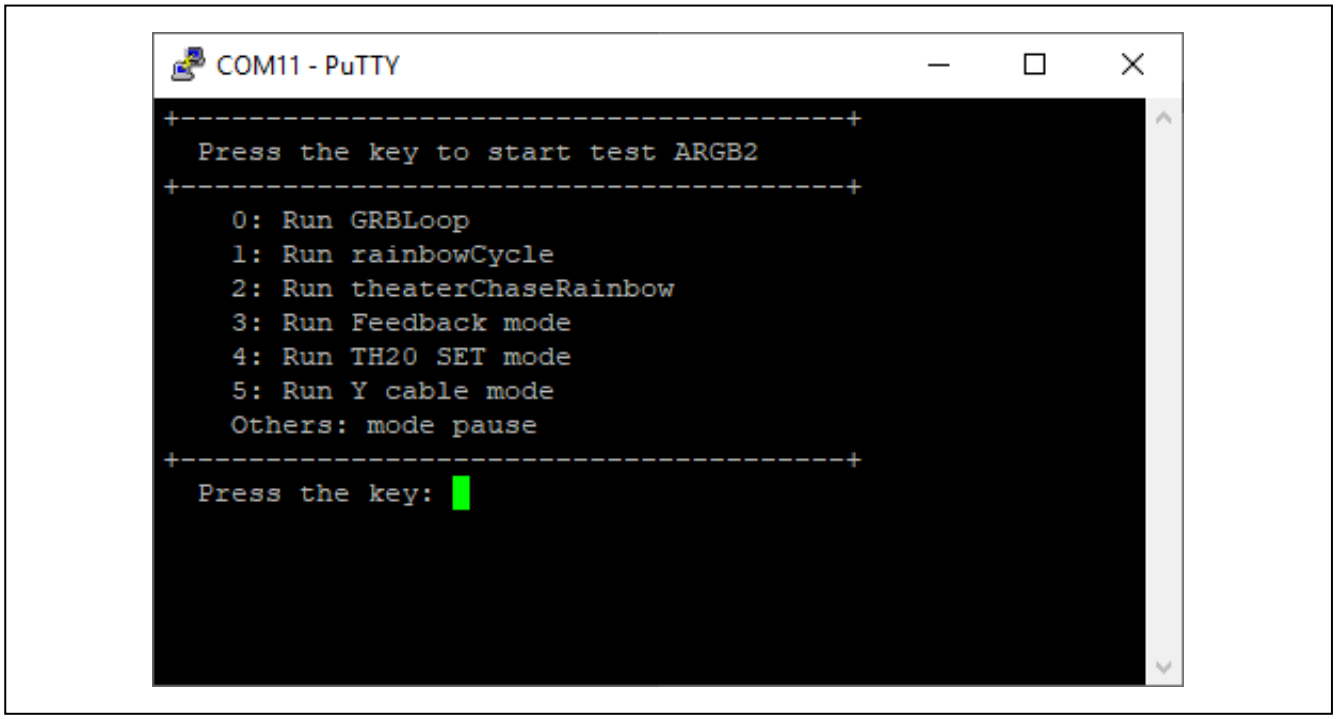

图 1-31 终端软件显示画面

2 代码介绍

2.1 主代码说明

代码工作流程如下：

1. main.c 中初始 M251/M252/M254/M256/M258 系列相关外设，包含 HIRC、GPIO、UART0、Timer 0
2. while() 循环中使用 switch…case 选择执行彩虹灯、跑马灯、Feedback 模式、TH20 SET 模式、并联模式 Y Cable 功能
3. 逻辑'0'、逻辑'1' 波形使用 NOP() 指令实现

```
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;
        }
    }
}
```

逻辑'0'、逻辑'1' 波形使用 NOP()指令来实现，请参考如下代码。

```
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 中对 ARGB2 灯条做初始设定，例如使用哪一个 GPIO 引脚、并联的ARGB2 灯条数量、ARGB2 灯条内 LED 数量、TH20 SET 模式相关设定、Y Cable 设定。

```
//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 灯效代码说明

2.2.1 GRB 循环

Strip_sendOne(int dat) 用来设定单颗灯颜色，一次传送 24 bits，0xff0000 代表绿灯，0x00ff00 代表红灯，0x0000ff 代表蓝灯， PIXEL_NUM 为欲设定灯的数量，利用 Timer 函数 TIMER_Delay() 可以调整绿灯、红灯、蓝灯的间隔时间。

```
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 彩虹灯

Strip_setPixelColor(uint16_t n, uint32_t GRBColor) 函数用来存放每颗灯的颜色，PIXEL_NUM 为欲设定灯的数量，Strip_wheel(uint8_t wheelPos) 函数用来设定颜色存放位置，借以产生轮转效果，Strip_show() 函数用来输出 BIT0、BIT1 波形至 ARGB2 灯条，以实现彩虹灯效果。

```
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 跑马灯

前半段代码与彩虹灯一样，Strip_setPixelColor(uint16_t n, uint32_t GRBColor) 函数用来存放每颗灯的颜色，PIXEL_NUM 为欲设定灯的数量，Strip_wheel(uint8_t wheelPos) 函数用来设定颜色存放位置，借以产生轮转效果，Strip_show() 函数用来输出 BIT0、BIT1 波形至 ARGB2 灯条，最后再用一个 for 循环，透过 Strip_setPixelColor(uint16_t n, uint32_t GRBColor) 函数让其一段时间间隔无颜色，以实现跑马灯效果。

```
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 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
- IDE 版本
  - Keil uVersion 5.26

3.2 硬件需求

- 电路组件
  - NuMaker-M258KE Ver1.2
  - ARGB2 灯条
- 示意图

使用 GPIO 引脚来模拟 ARGB2 灯条时序图。

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

图 3-1 线路连接

※备注：ARGB2 灯条工作电压为 5 V，故 NuMaker-M258KE 开发板为 ARGB2 供应电压必须更换为 5V。可以改变Nu-Link2-Me 上的ICEJPR1(MCUVCC)电阻使其输出5V电压。

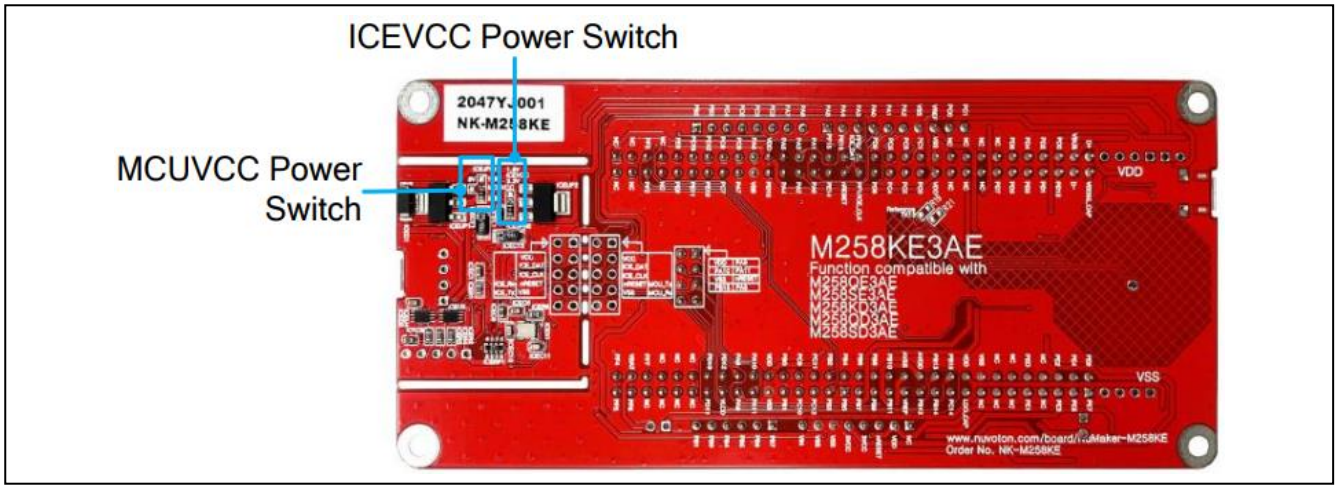

图 3-2 MCUVCC 电源切换至 5V

4 目录信息

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
Sample_Code	
ExampleCode	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 M258_GPIO_ARGB2_LED_Control.uvprojx。
2. 进入编译模式界面
   a. 编译
   b. 下载代码至内存
   c. 进入 / 离开仿真模式
3. 进入仿真模式界面
   a. 执行代码

6 修订纪录

Date	Revision	Description
2022.04.01	1.00	1. 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.