Using M258 GPIO to Achieve ARGB2 LED Effect

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code uses GPIO to simulate ARGB2 LED timing sequence and achieve the Rainbow Wave ARGB2 LED effect on the M258 series microcontroller (MCU).
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
Hardware	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 Overview

This sample code uses GPIO to simulate ARGB2 LED timing sequence and produce LED Rainbow Wave effect. This code uses one timer to schedule ARGB2 LED to do ARGB2 LED toning, one set of GPIO to simulate ARGB2 LED strip timing sequence and one set of UART to select the execution mode.

1.1 Principle

1.1.1 ARGB2 LED Data Transfer Time (TH + TL = 1.2us ± 600ns)

The ARGB2 LED timing sequence is shown in Figure 1-1 and Figure 1-2. Logic 0 is defined as a high pulse of 0.3 ± 0.15 us followed by a low pulse of 0.9 ± 0.15 us. Logic 1 is defined as a high pulse of 0.9 ± 0.15 us followed by a low pulse of 0.3us ± 0.15us.

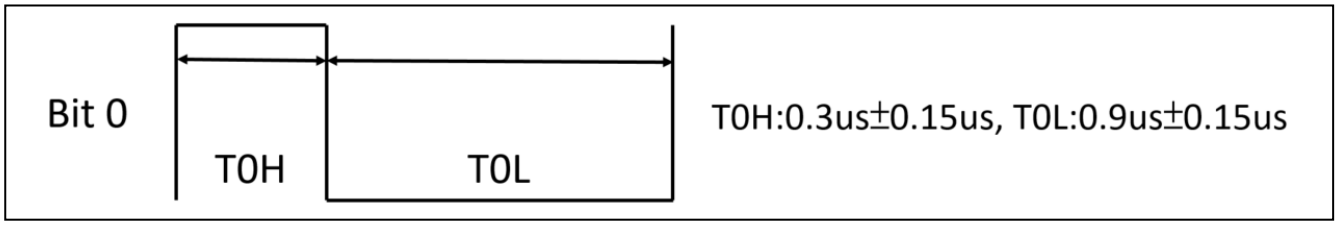

Figure 1-1 Logic 0 (Bit 0)

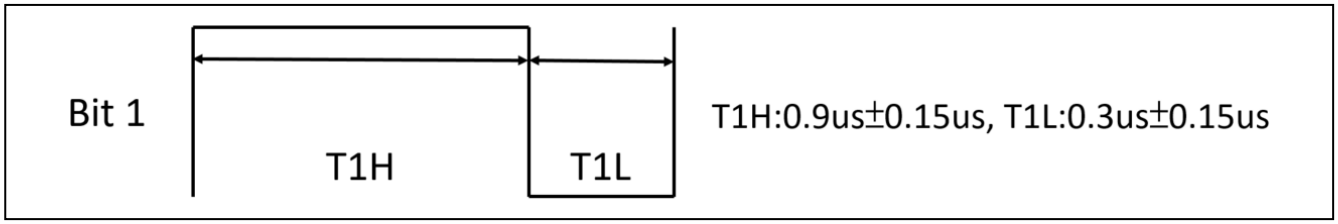

Figure 1-2 Logic 1 (Bit 1)

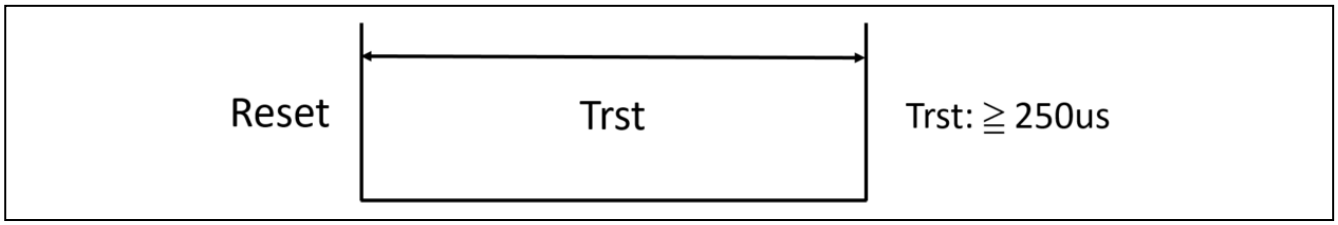

Figure 1-3 Reset (Trst)

1.1.2 GPIO Clock Frequency and Logic Bit Design

Use GPIO to control ARGB2 strip. Delay the high and low levels for a period of time through the NOP() instruction. The system clock HIRC is 48 MHz. Thus, the time to execute a command is about 1 / 48 MHz (20.83ns). It can be obtained that the NOP() instruction needs to be executed several times.

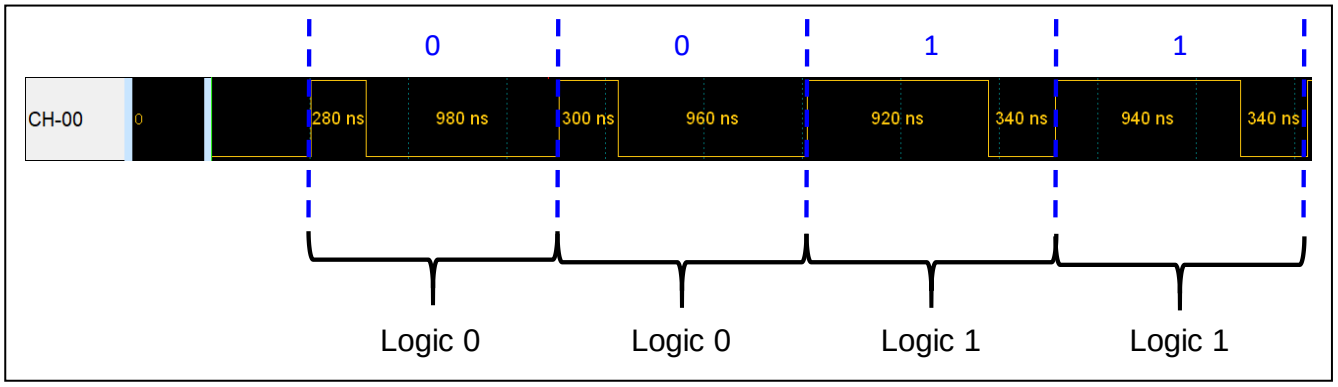

Figure 1-4 Delay High and Low Levels for a Period through NOP Instruction

1.1.3 Feedback Mode

In addition to transmitting signals through the MCU for lighting effects, the ARGB2 strip can also send back messages to the MCU. There are two types of content:

- The number of LEDs on the strip and the current consumption of the LEDs
- The number of LEDs on the strip and the currently set ID of the strip

The waveform of feedback mode is shown in Figure 1-5. After the Feedback mode command is sent to the ARGB2 strip, the ARGB2 strip will return a pulse. The MCU must immediately switch the GPIO from Output to Input to receive the pulse. By reading the number and width of the pulse can get the number of LEDs on the strip, the maximum consumption of current, and the currently set ID of the strip. Entering the TH20 SET mode to change the setting can switch between the above two return contents. Please refer to section 1.1.4.

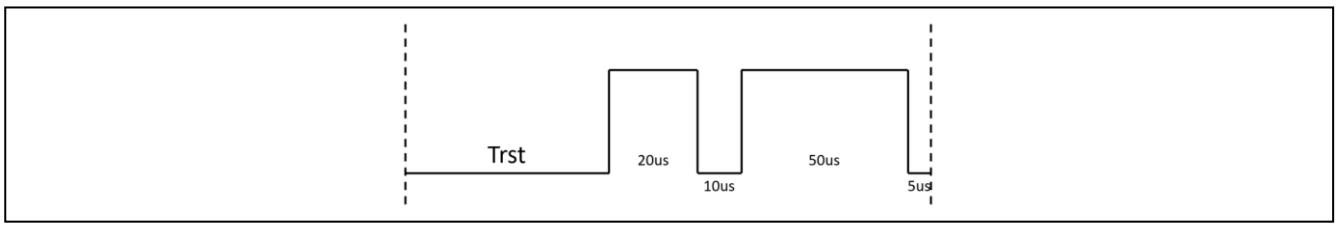

Figure 1-5 Waveform of Feedback Mode

The high pulse width of ARGB2 strip return denotes:

- Receive the 20 us width pulse: the current consumption of the LEDs is 12 mA
- Receive the 10 us width pulse: the current consumption of the LEDs is 5 mA

1.1.4 TH20 SET Mode

Send 20 us (high level) + 250 us (low level) signal from MCU to ARGB2 strip to enter TH20 SET mode, followed by 24 bits as shown in Figure 1-6. For the content of setting 24 bits data, refer to Table 1-1 and Table 1-2.

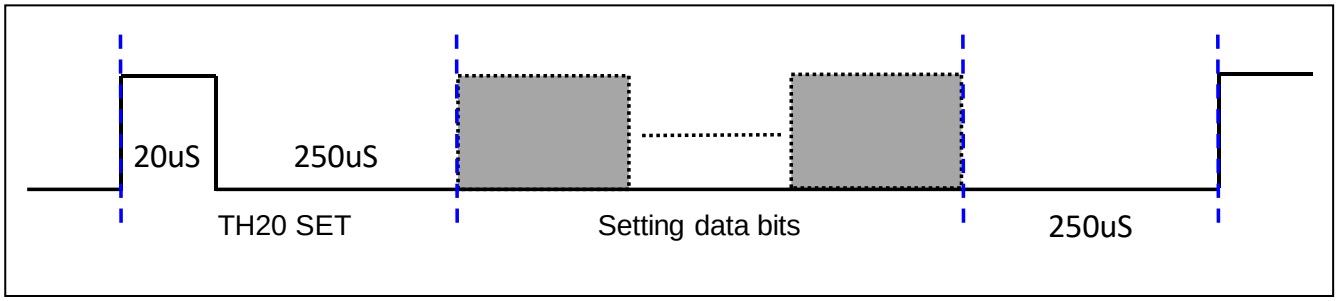

Figure 1-6 Waveform of TH20 SET Mode

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>

Table 1-1 24 Bits of Setting Data

SS[0]	PWM reset and synchronization command setting, default: 0 0: non-synchronization 1: synchronized for PWM (PWM internal counter is reset, re-started and synchronized with Trst end point)
SS[2：1]	Imax (don't change), default：00
SS[3]	Display data update and validation, default: 0 0: display (PWM data) is valid and synchronized with Trst end point 1: non-synchronized with Trst end point (data is valid immediately after PWM data is received)
SS[5：4]	Display re-fresh rate, default: 11 00: 1.25 kHz　01: 2.5 kHz　10: 10 kHz　11: 20 kHz
Sel_CUR	Optional bit to change the feedback information, default: 0 0: feedback the max. sink current of R, G, B port 1: feedback the strip fixed ID
G[4: 0]	GLED Dimming, default: 0x1F
R[4: 0]	RLED Dimming, default: 0x1F
B[4: 0]	BLED Dimming, default: 0x1F

Table 1-2 Content of Setting 24 Bits Data

The Dimming function controls the output current formula as follows:

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

1.1.5 Parallel Mode (Y Cable Function)

The ID of ARGB2 strip can be set through Y Cable function. Send 250 us (low level) + 40 us (high level) + 10 us (low level) to ARGB2 strip to open Y Cable function as shown in Figure 1-7. Parallel mode has four commands as shown in Table 1-3.

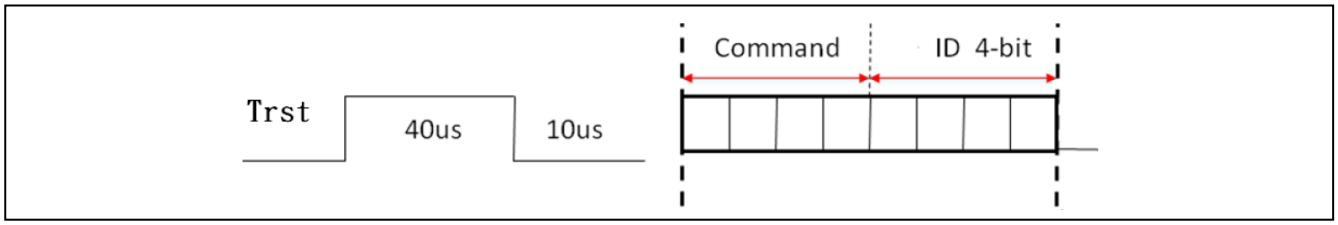

Figure 1-7 Waveform of Parallel Mode Using Open Y Cable Mode

Command[7:4]	0001: Set the strip's ID (SET_ID) 0010: Clear the strip's ID (CLR_ID) 0011: Check the strip's ID (CHK_ID) 0100: Assign a specific strip to receive data (Specify_ID)
ID[3:0]	Range: 0000 ~ 1111

Table 1-3 Command Setting of Y Cable Function

◆ Set the strip's ID (SET_ID)

If setting ID to ARGB2 strip is successful when sending SET_ID + ID command to ARGB2 strip, ARGB2 strip will reply a high pulse. The MCU must immediately switch the GPIO from Output to Input to receive the pulse. In this way, all parallel ARGB2 strips are given with ID numbers. When all parallel strips have ID numbers and then SET_ID command is sent, the ARGB2 strips will not reply high pulse.

Note: The default dynamic ID number of each strip is 0 after power-on reset.

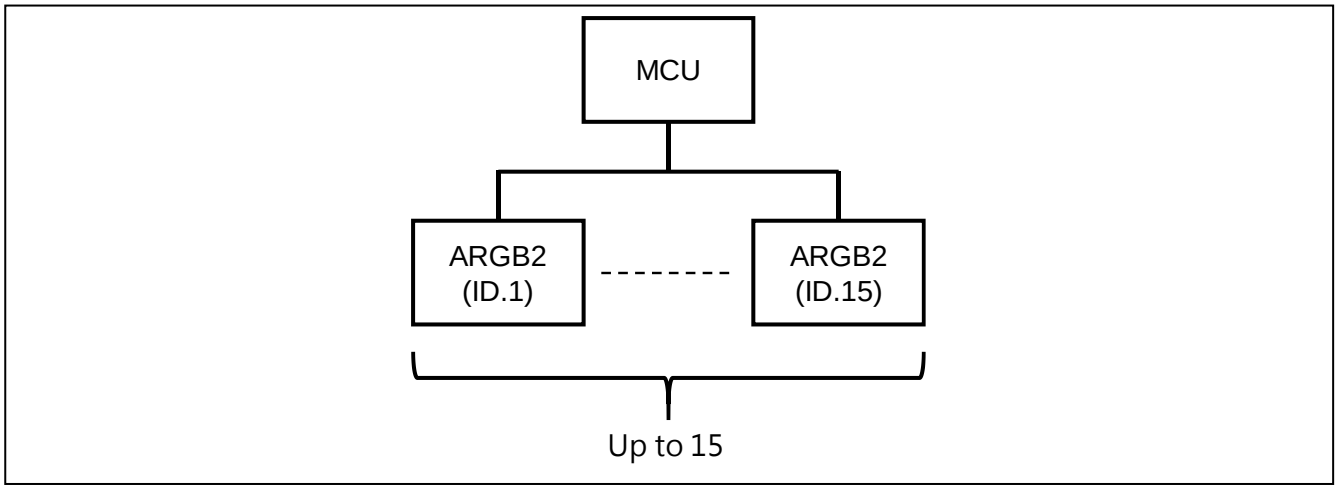

Figure 1-8 SET_ID Concept

◆ Clear the strip's ID (CLR_ID)

Clear all ARGB2 strip's ID when sending CLR_ID =0000. Clear the specific ID number when ID's range is 0001 ~ 1111. It is suggested to use one CLR_ID + 0000 command to assure all parallel strip's ID is 0.

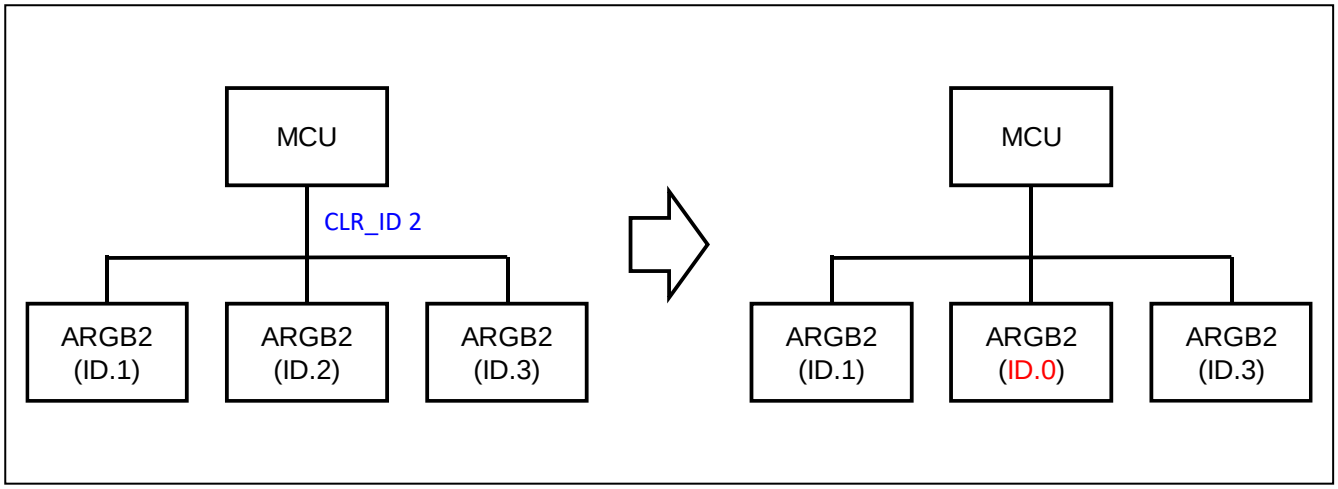

Figure 1-9 CLR_ID Concept

◆ Check the strip's ID (CHK_ID)

It is the same as the SET_ID command. After the MCU sends the CHK_ID command, if it matches the specified ID, the ARGB2 strip will return high pulse. If the ID of the parallel ARGB2 strip has been set repeatedly, send the CHK_ID command at this time, and the MCU will receive the repeatedly set number of high pulses.

◆ Assign a specific strip to receive data (Specify_ID)

Only the assigned ID strip will act, and when the ID of the parallel ARGB2 strip is 0, it will act together. The working principle is shown in Figure 1-10.

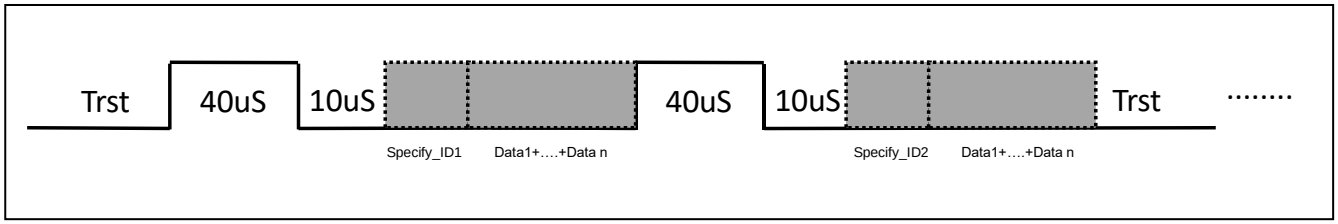

Figure 1-10 Specify_ID Concept

The control flow of the Y Cable function in parallel mode is as follows:

1. CLR_ID: Set the ID of all parallel ARGB2 strips to 0
2. SET_ID: Allocate different IDs to all ARGB2 strips. After the ID assignment is successful, the ARGB2 strips will return Pulse
3. CHK_ID: Confirm the ARGB2 strips assigned to the ID; if the assignment is successful, the ARGB2 light bar will return Pulse
4. Specify_ID: Specify the strips for control.

1.2 Demo Result

1.2.1 Logic'1' (Bit 1), Logic'0' (Bit 0), Reset (Trst) Demo Result

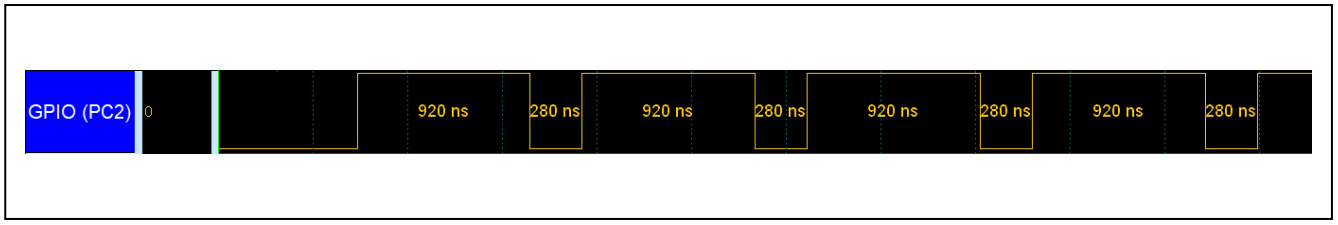

Figure 1-11 Logic'1' Demo Result

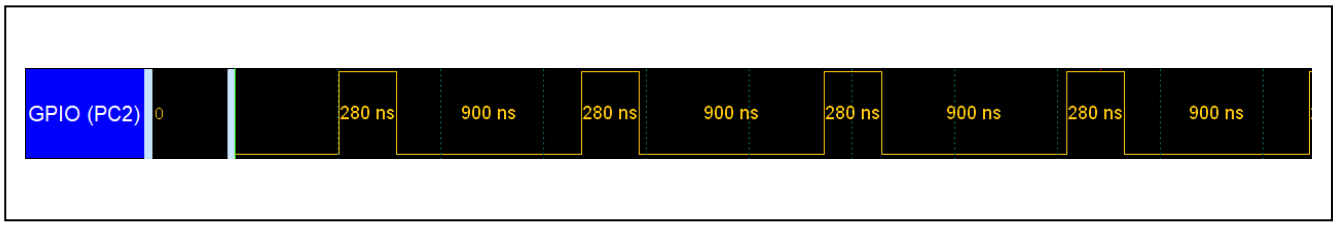

Figure 1-12 Logic'0' Demo Result

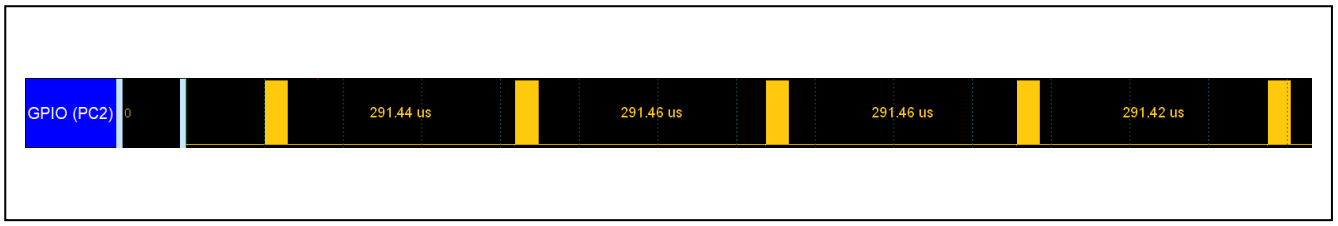

Figure 1-13 Trst Demo Result

1.2.2 GRB Loop Demo Result

The GRB loop demo result is shown as Figure 1-14 ~ Figure 1-16. For code description, please refer to section 2.2.1.

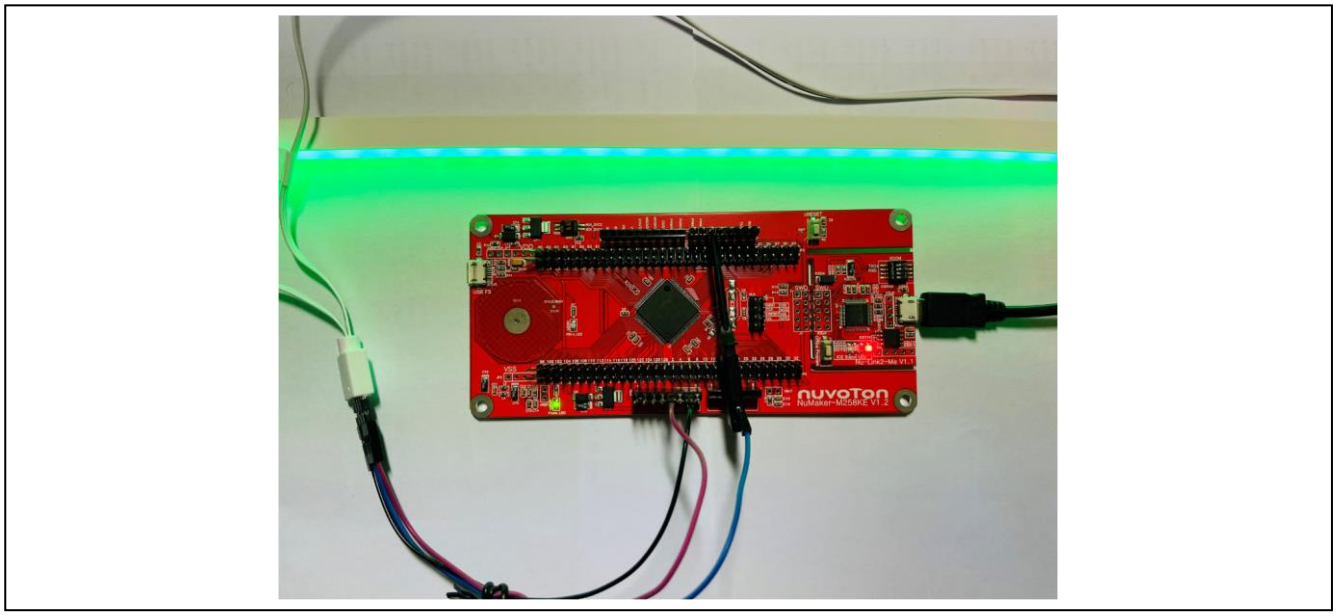

Figure 1-14 Green Light Demo Result

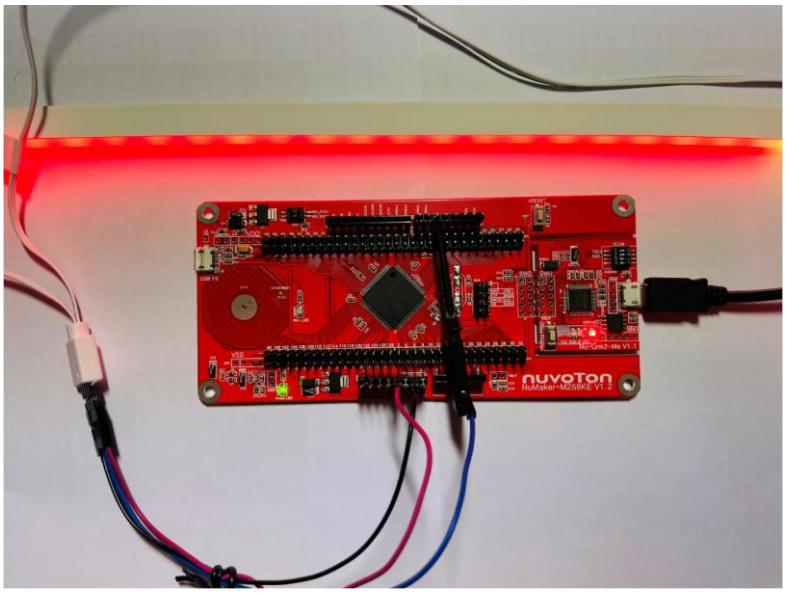

Figure 1-15 Red Light Demo Result

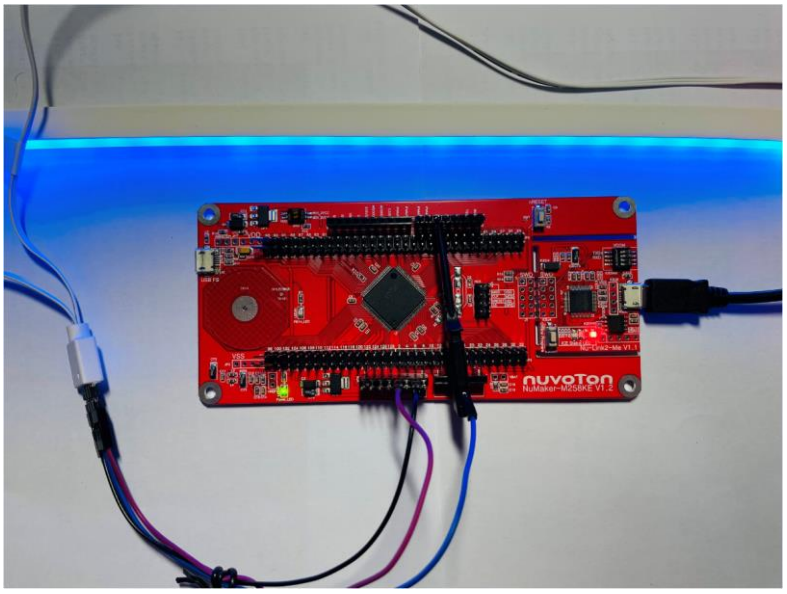

Figure 1-16 Blue Light Demo Result

1.2.3 Rainbow Cycle Demo Result

The rainbow cycle demo result is shown as Figure 1-17. For code description, please refer to section 2.2.2.

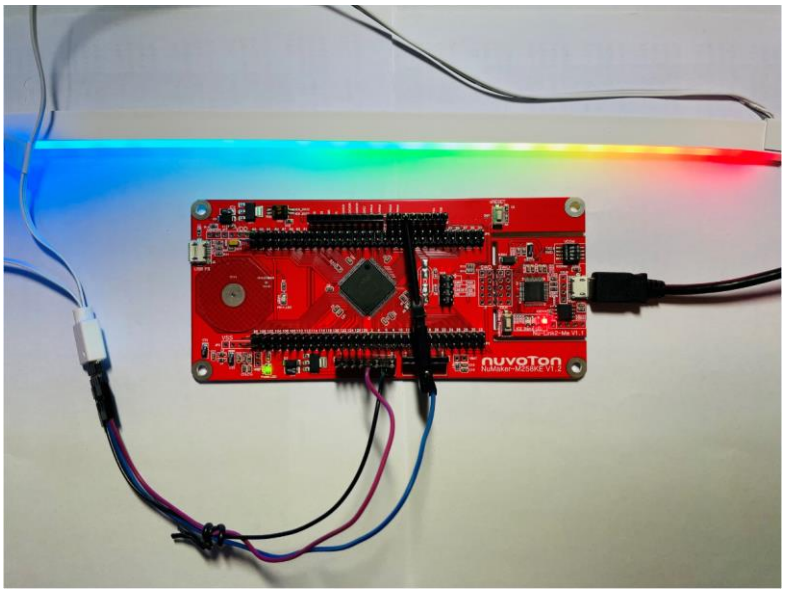

Figure 1-17 Rainbow Cycle Demo Result

1.2.4 Theater Chase Rainbow Demo Result

The Theater chase rainbow demo result is shown as Figure 1-18. For code description, please refer to section 2.2.3.

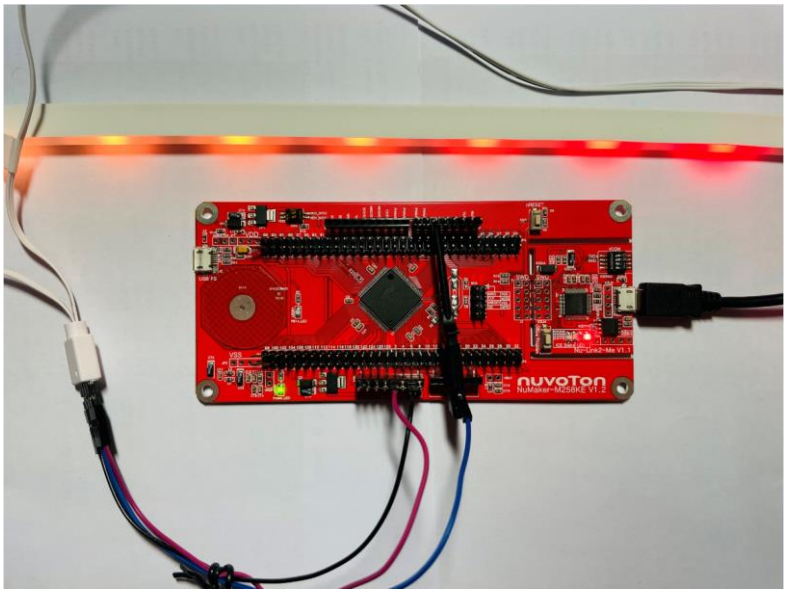

Figure 1-18 Theater Chase Rainbow Demo Result

1.2.5 Feedback Mode Demo Result

The MCU sends 20us (Hi) + 10us (Lo) + 50us (Hi) to the ARGB2 strip to enable the Feedback mode. The waveform is shown in Figure 1-19. After the ARGB2 strip receives the waveform of enabling Feedback mode, it will send back Pulse to indicate that there are several LED lights in a strip as shown in Figure 1-20, and the maximum current consumption value of a single LED as shown in Figure 1-21.

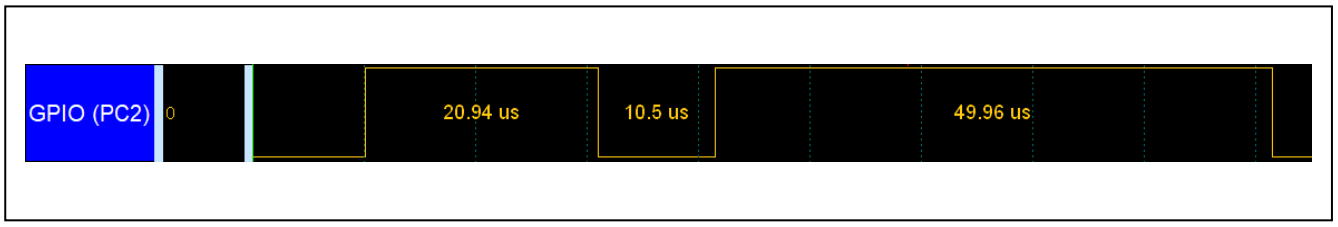

Figure 1-19 Waveform of Enabling Feedback Mode

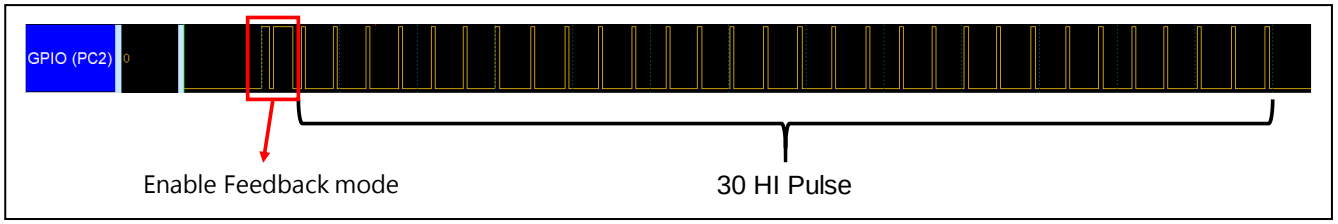

Figure 1-20 ARGB2 Strip Send Back Pulse

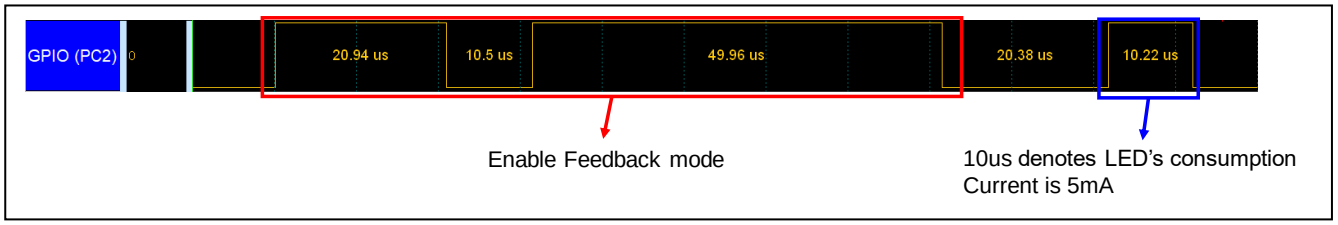

Figure 1-21 The Width of Pulse Denoting the Maximum Consumption Current

1.2.6 TH20 SET Mode Demo Result

The MCU sends 20us (Hi) + Trst (Lo) to the ARGB2 light bar to enable the TH20 SET mode. The waveform is shown in Figure 1-22, and the setting of 24 bits is shown in Figure 1-23.

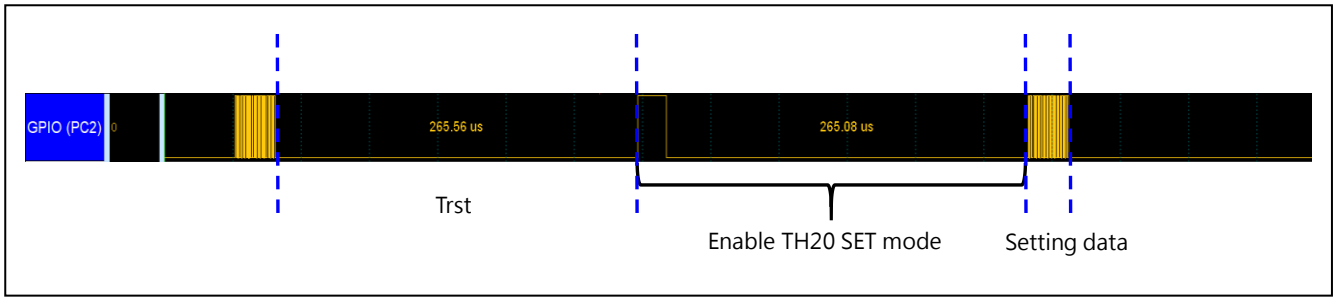

Figure 1-22 Waveform of Enabling TH20 SET Mode

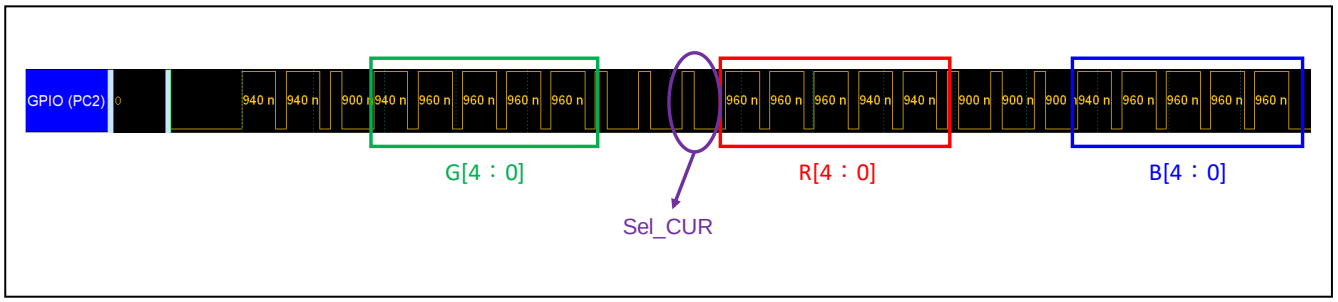

Figure 1-23 Setting 24 Bits Data

1.2.7 Parallel Mode (Y Cable Function) Demo Result

The MCU sends 40us (Hi) + 10us (Lo) + 8 bits (Command + ID) to the ARGB2 strip to enable the Y Cable function. The waveform is shown in Figure 1-24, and the command waveform is shown in Figure 1-25 ~ Figure 1-30.

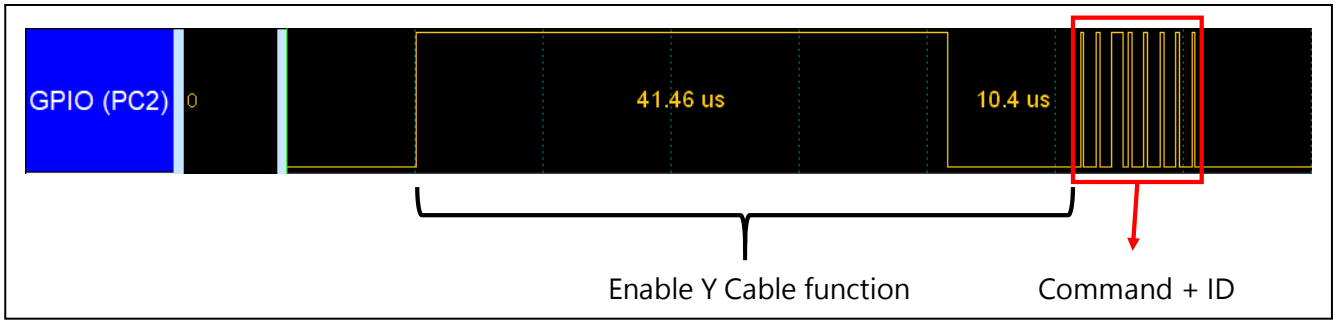

Figure 1-24 Waveform of Enabling Y Cable Function

- Set the strip's ID (SET_ID) demo result.

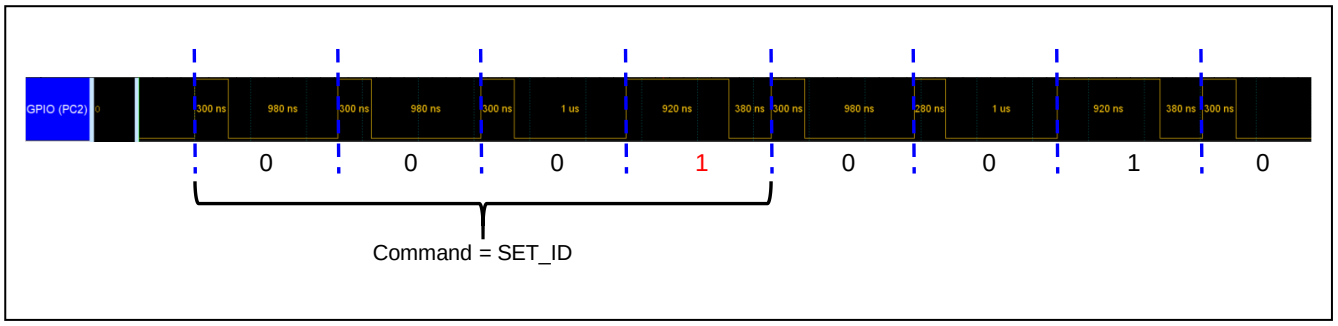

Figure 1-25 Waveform of SET_ID Command

Figure 1-26 SET_ID Command Successful

- Clear the strip's ID (CLR_ID) demo result.

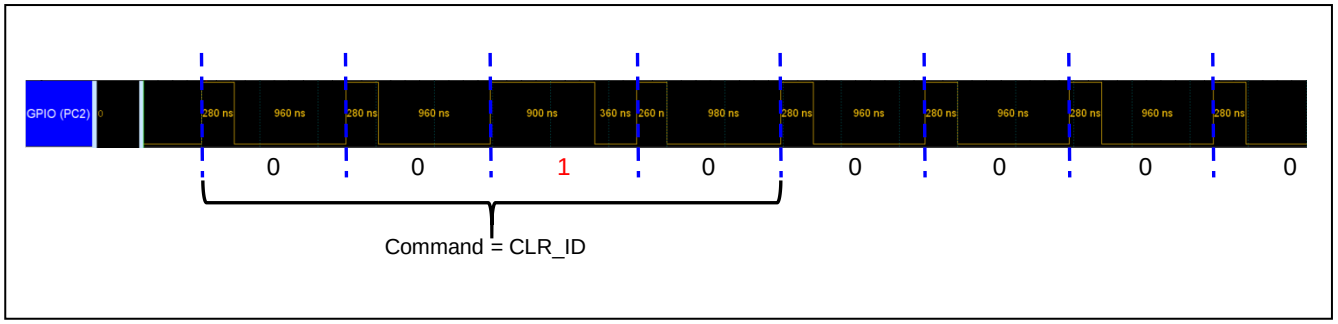

Figure 1-27 Waveform of CLR_ID Command

- Check the strip's ID (CHK_ID) demo result.

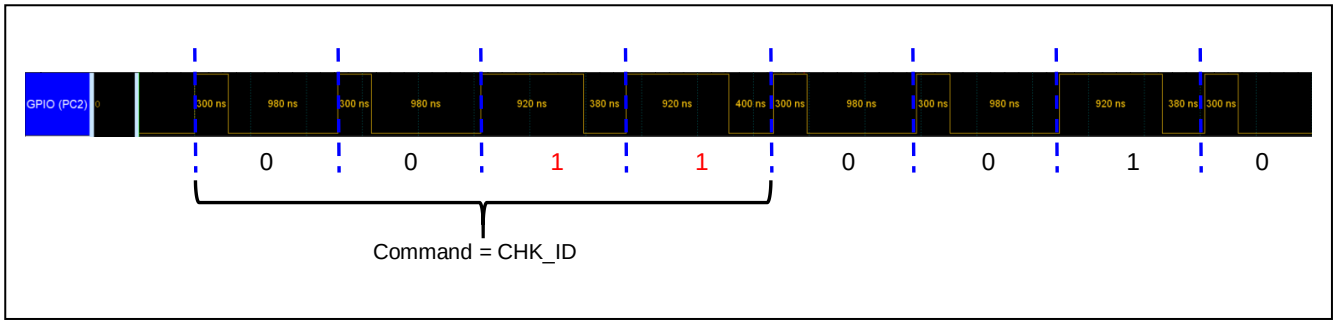

Figure 1-28 Waveform of CHK_ID Command

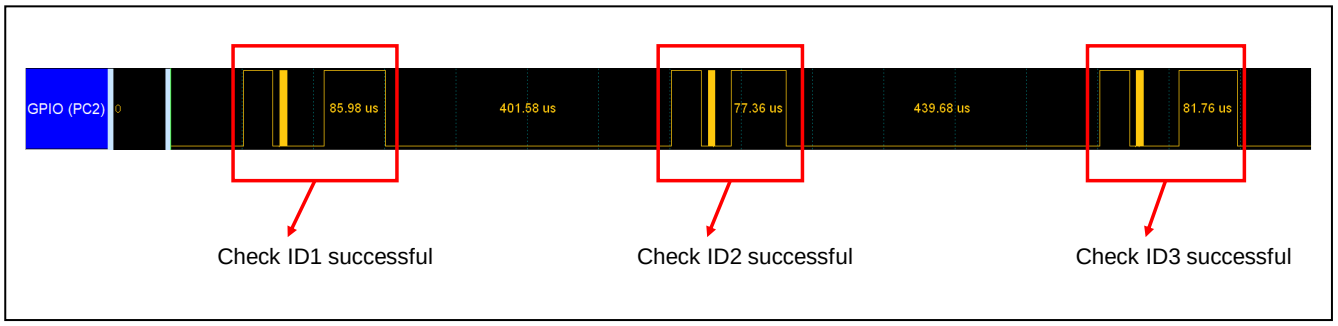

Figure 1-29 CHK_ID Command Successful

- Assign a specific strip to receive data (Specify_ID) demo result.

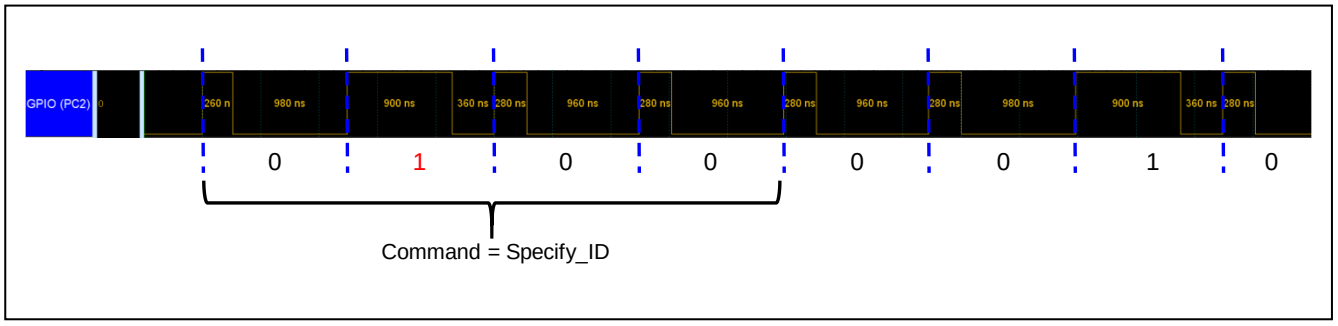

Figure 1-30 Waveform of Specify_ID Command

1.2.8 Terminal Software Demo Result

Use terminal software to select execution mode through UART. Please set baud rate to 115200. The demo result is shown in Figure 1-31. The mode is as follows:

0: Run light effect of green, red, blue cycle
1: Run light effect of rainbow cycle
2: Run light effect of theater chase rainbow
3: Run Feedback mode
4: Run TH20 SET mode
5: Run Y Cable Function

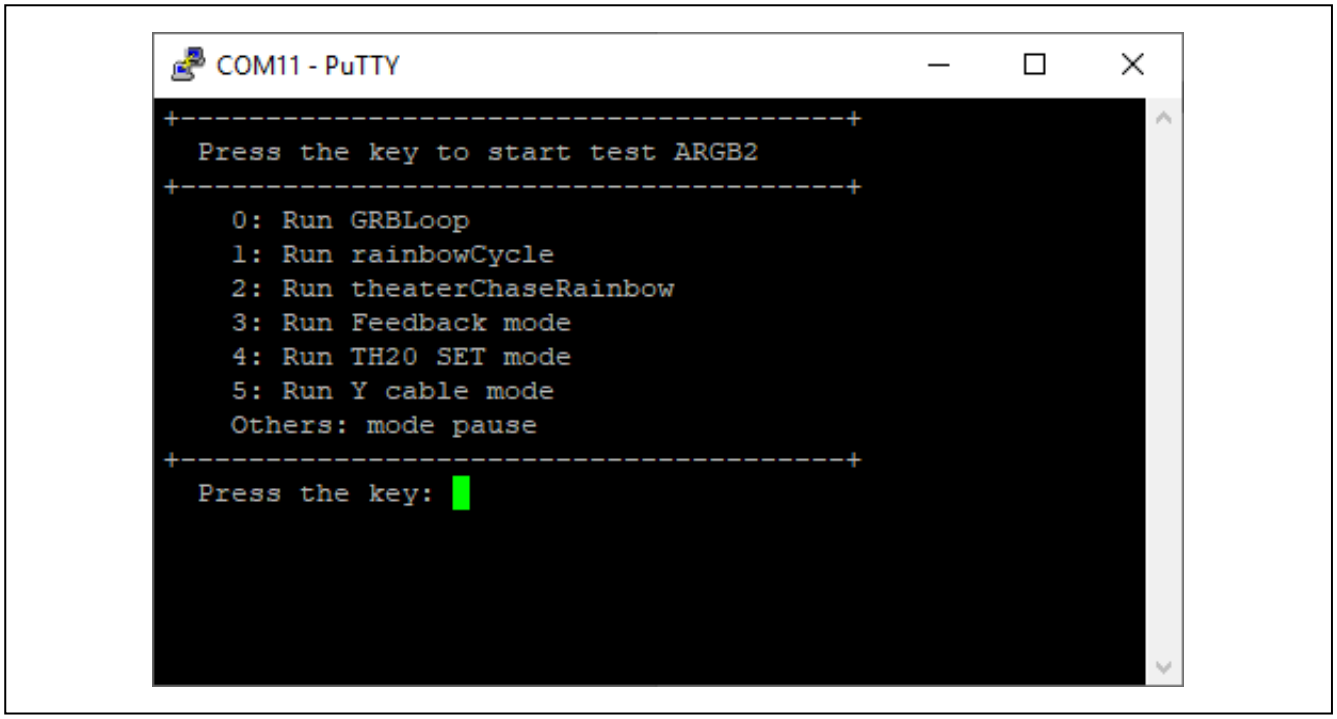

Figure 1-31 Demo Result of Terminal Software

2 Code Description

2.1 Main Code Description

The steps are listed below:

1. Initialize the M251/M252/M254/M256/M258 series in main.c, including HIRC, GPIO, UART0 and Timer0.

2. Select which function will run in a while loop, RainbowWave, TheaterChase, Feedback mode, TH20 SET mode or Parallel mode (Y Cable function).

```
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;
        }
    }
}
```

3. Realize logic'0', logic'1' by NOP() instruction.

```
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();
    }
}
```

In Nuvoton_ARGB2_Strip.h, you can make initial settings for ARGB2 strips, such as which GPIO pin to use, the number of parallel ARGB2 strips, the number of LEDs in ARGB2 strips, TH20 SET mode related settings, and Y Cable function settings.

```
//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 Light Effect Code Description

2.2.1 GRB Loop Function

The function Strip_sendOne(int dat) is used to set each color LED. Send 24 bits at the same time. 0xff0000 denotes green light, 0x00ff00 denotes red light, and 0x0000ff denotes blue light. PIXEL_NUM is the number of LED in a strip. The function TIMER_Delay() is used to set the interval time of green light, red light and blue light.

```
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 Rainbow Cycle Function

The function Strip_setPixelColor(uint16_t n, uint32_t GRBColor) is used to save each color LED. PIXEL_NUM is the number of LED in a strip. The function Strip_wheel(uint8_t wheelPos) is used to set the position of LED color to produce rolling effect. The function Strip_show() is used to output BIT0 and BIT1 signal to ARGB2 strip to realize the rainbow cycle effect.

```
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 Theater Chase Rainbow Function

The first half of the code is the same as the rainbow cycle function. The function Strip_setPixelColor(uint16_t n, uint32_t GRBColor) is used to save each color LED. PIXEL_NUM is the number of LED in a strip. The function Strip_wheel(uint8_t wheelPos) is used to set the position of LED color to produce rolling effect. The function Strip_show() is used to output BIT0 and BIT1 signal to ARGB2 strip. Finally, use the function Strip_setPixelColor(uint16_t n, uint32_t GRBColor) to set no LED color interval to realize the theater chase rainbow effect.

```
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 Software and Hardware Requirements

3.1 Software Requirements

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

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M258KE Ver1.2
  - ARGB2 strip
- Pin Connect

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

Figure 3-1 Pin Connect

Note: The operating voltage of ARGB2 strip is 5V, so the supply voltage for ARGB2 on NuMaker-M258KE evaluation board must be changed to 5V. The ICEJPR1 (MCUVCC) resistor on the Nu-Link2-Me can be changed to output 5V.

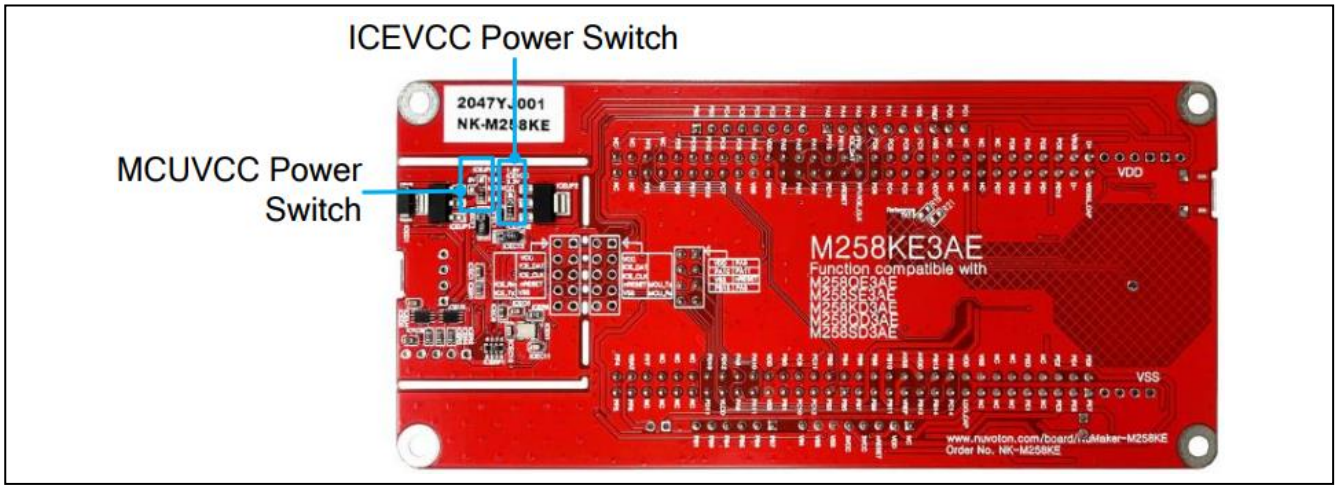

Figure 3-1 MCUVCC Power Switch to 5V

4 Directory Information

The directory structure is shown below.

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

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click M258_GPIO_ARGB2_LED_Control.uvprojx.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2022.04.19	1.00	1. Initially issued.

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.