Using PSIO to Achieve ARGB2 LED Effects

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This sample code uses M252 PSIO to implement ARGB2 LED timing and realize lighting effects such as rainbow and theater chase rainbow.
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.001
Hardware	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. Overview

PSIO (Programmable Serial I/O) provides an easy way to receive and transmit various serial transmissions, such as HDQ, DMX512, 1-wire, IR, PS/2, Microwire, Wiegand, and LED. This example code shows how to use PSIO to implement a control ARGB2 LED timing sequence and rainbow wave effect, and uses a set of timer to regularly update the color of the ARGB2 LED, and a set of UART to select the execution mode.

1.1 Principle

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

The ARGB2 LED timing sequence is shown in Figure 1-1 and Figure 1-2. Logic '0' defines high level 0.3 μs ± 0.15 μs and low level 0.9 μs ± 0.15 μs, and logic '1' is defined as high level 0.9 μs ± 0.15 μs and low level 0.3 μs ± 0.15 μs.

To use PSIO to control ARGB2 LED strips, the user must first set the timing rules (slot) and operate according to the expected timing. Divide the basic unit of time into about 0.3 μs to create the desired waveform. The timing calculation formula is as follows:

- PSIO clock source selected PCLK1, PCLK1 is calculated as HIRC (48MHz) / 4 = 12 MHz.
- PSIO clock uses PCLK1 divided by 4 = 3 MHz.
- PSIO one slot cycle time = 1 / 3 MHz = 0.33 μs.

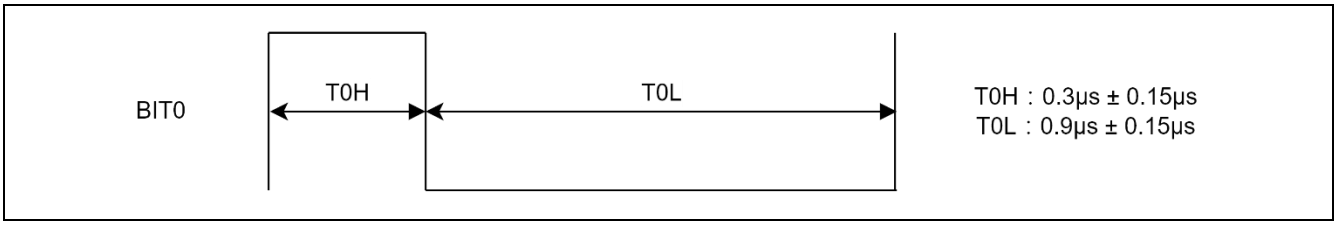

Figure 1-1 Logic ‘0’ (Bit 0)

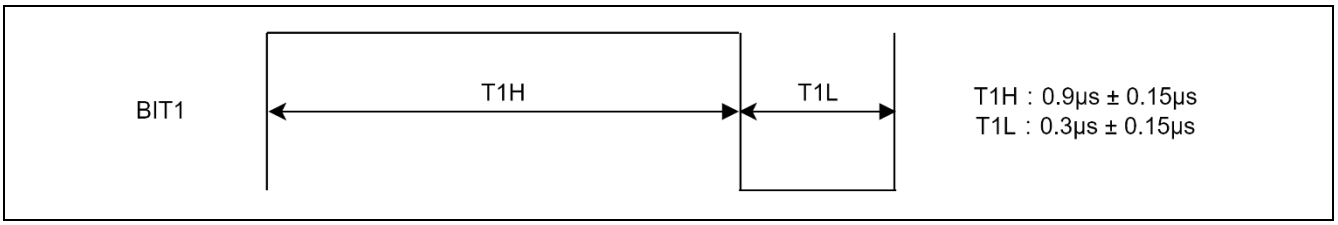

Figure 1-2 Logic ‘1’ (Bit 1)

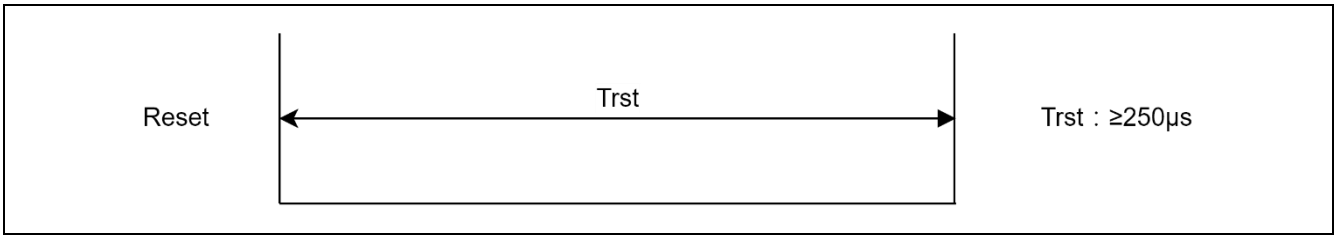

Figure 1-3 Reset (Trst)

1.2 Advanced Function Mode

1.2.1 Feedback Mode

In addition to transmitting signals through the microcontroller (MCU) for lighting effects, the RGB LED strips can send messages back 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.

In the feedback mode command waveform as shown in Figure 1-4, after issuing the feedback mode command to the ARGB2 LED strip, the ARGB2 LED strip will return high pulse. The PSIO control pin is switched from Output to Input to read the pulse, by reading the number and width of the pulse, the user can get how many LEDs on the strip and the maximum current used, and the ID of the current strips is set. If the high pulse is not received within 160 μs, the MCU will be restored to Normal Mode, that is, the PSIO control pin is switched from Input to Output.

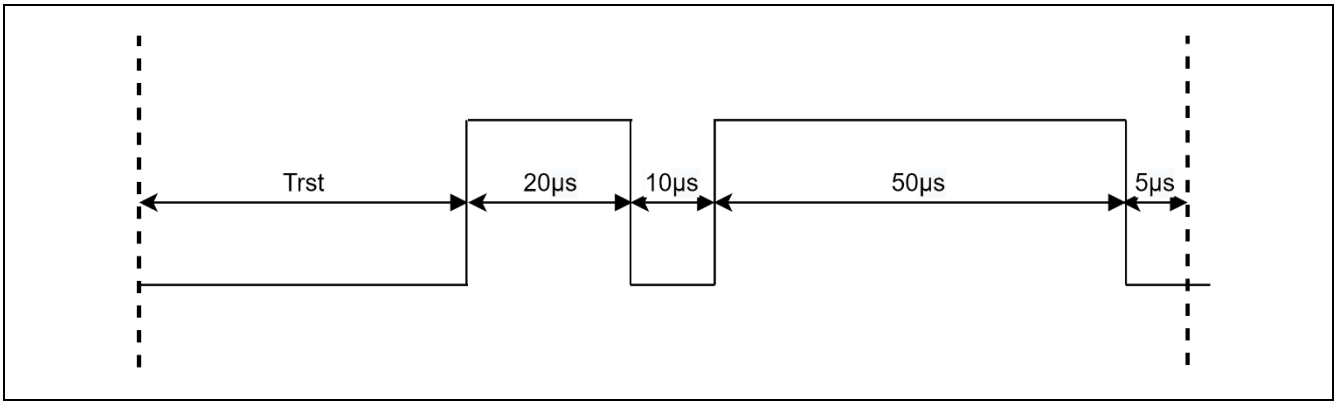

Figure 1-4 Feedback Mode Waveform

Pulse width represents the following meanings:

- When receiving 20 μs high pulse, the LED consumes 12 mA.
- When receiving 10 μs high pulse, the LED consumes 5 mA.

1.2.2 TH20 SET Mode

Send a 20 μs (high level) + 250 μs (low level) signal from the MCU to the ARGB2 LED strip to enter the TH20 SET mode, followed by 24 bits, as shown in Figure 1-5. After the TH20 SET mode, there are 24 bits of setting data. Please refer to Table 1-1 and Table 1-2.

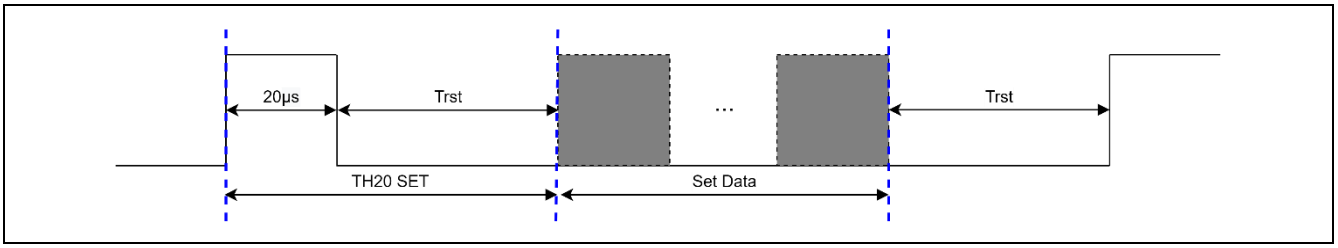

Figure 1-5 TH20 SET Mode Waveform

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>

Table 1-1 TH20 SET Mode 24 Bits Data Format

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: 1.25kHz 01: 2.5kHz 10: 10kHz 11: 20kHz
SS[6]	Optional bit to change the feedback information, default: 0 0: feedback the max. sink current of G, R, B port 1: feedback the strip fixed ID
G[4:0]	G LED Dimming, default: 0x1F
R[4:0]	R LED Dimming, default: 0x1F
B[4:0]	B LED Dimming, default: 0x1F

Table 1-2 TH20 SET Mode Setting Function Description

The dimming function controls the output current formula as follows:

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

1.2.3 Parallel Mode (Y Cable Mode)

Through the Y Cable function, the user can set the ID of the ARGB2 LED strip. To enable the Y Cable function, as shown in Figure 1-6, there must be a Reset signal of 250 μs (low level) Pulse before entering Y Cable mode. There are four kinds of commands for the Y Cable function in parallel mode. Please refer to Table 1-3.

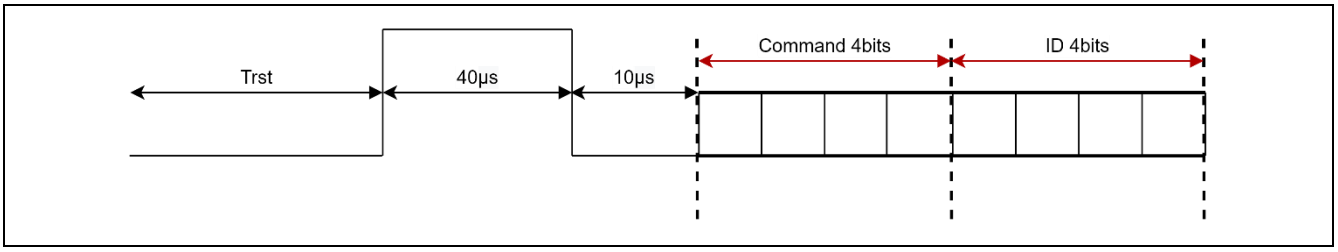

Figure 1-6 Y Cable Mode Waveform

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 Description

1.2.3.1 Set the Strip's ID

After the SET_ID + ID command is sent, if the ID setting is successful, the ARGB2 LED strip will return high pulse. The PSIO should switch from Output to Input immediately. In this way, assign ID numbers to all parallel ARGB2 LED strips. When all parallel strips have ID numbers and then SET_ID, the ARGB2 LED strips will not respond to high pulse.[1]

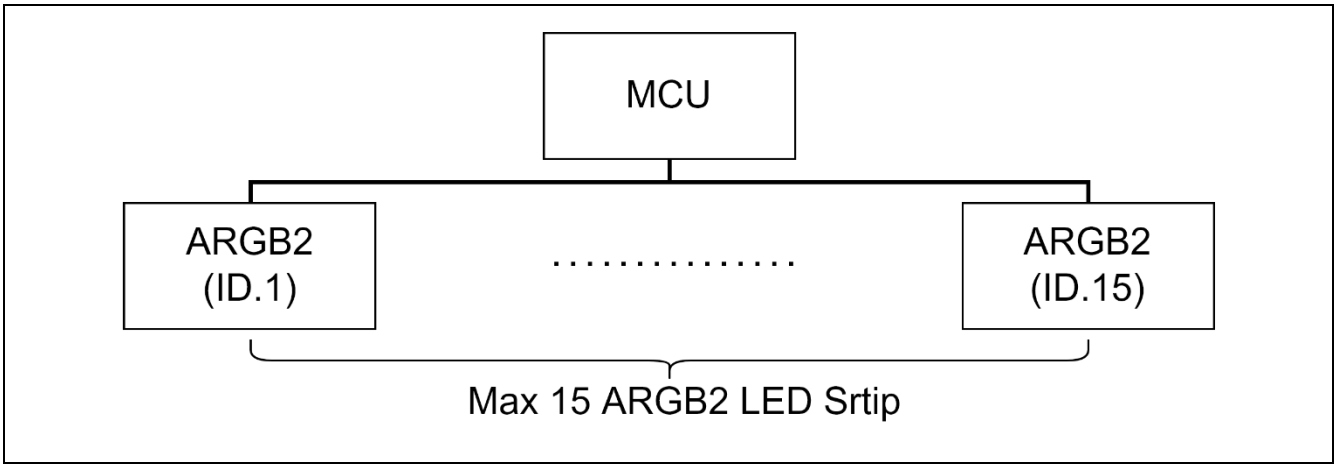

Figure 1-7 Set ID Schematic Diagram

[1] Every time powered ON, the ID of the ARGB2 LED strip will become 0.

1.2.3.2 Clear the Strip's ID

All ARGB2 LED strip IDs can be cleared when the transmitted ID = 0000, and when ID = 0001 ~ 1111, the specified ID can be cleared to 0. Before setting the ID, it is recommended to do CLR_ID + 0000 once, ensure that the IDs of all parallel ARGB2 LED strips are 0.

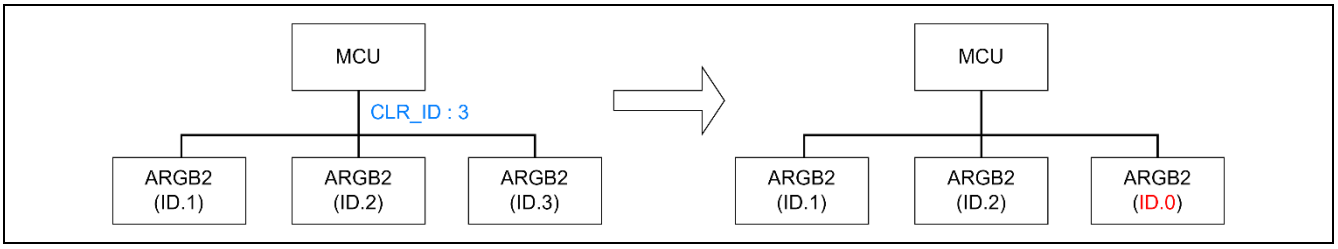

Figure 1-8 Clear ID Schematic Diagram

1.2.3.3 Check the Strip's ID

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

1.2.3.4 Assign a Specific Strip to Receive Data

Only the assigned ID light bar will act. When the ID of the parallel ARGB2 LED strip is 0, they will act together. A diagram of the waveform is shown in Figure 1-9.

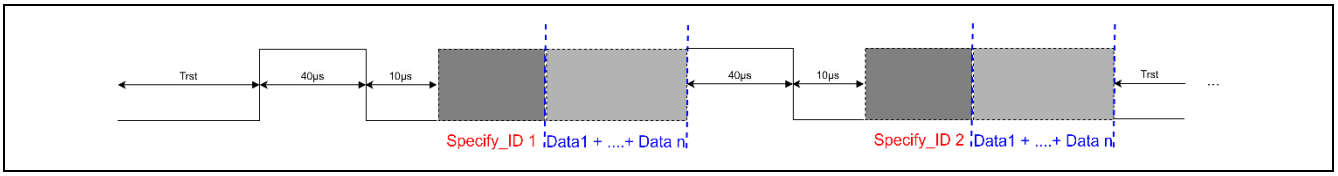

Figure 1-9 Specific ID Schematic Diagram

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

- SET_ID: Allocate different IDs to all ARGB2 LED strips. After the ID assignment is successful, the ARGB2 LED strips will return pulse.
- CLR_ID: Set the ID of all parallel ARGB2 LED strips to 0.
- CHK_ID: Confirm the ARGB2 LED strips assigned to the ID, if the assignment is successful, the ARGB2 LED strip will return high pulse.
- Specify_ID: User can specify the strips for control.

1.3 Demo Result

Use terminal software through UART select execution mode, and set the baud rate to 115200. The terminal software execution results is shown in Figure 1-10. The selectable mode is as follows, where 1 ~ 3 is the ARGB2 LED strip effect display, and 4 ~ 7 is to open the light bar in different modes:

1. Run light effects of green, red, and blue cycle
2. Run light effects of rainbow cycle
3. Run light effects of theater chase rainbow
4. Run Feedback mode
5. Run TH20 SET mode
6. Run Y Cable function
7. Run set green, red, and blue brightness

```
================================================
        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
```

Figure 1-10 Result of Terminal Software Demo

1.3.2 Logic ‘0’ (Bit 0), Logic ‘1’ (Bit 1), Reset (Trst) Execution Result

Figure 1-11 Result of the Logic ‘1’ (Bit 1) and Logic ‘0’ (Bit 0)

Figure 1-12 Result of the Reset (Trst) Demo

1.3.3 GRB Loop Demo Result

Figure 1-13 Result of Green Light Demo

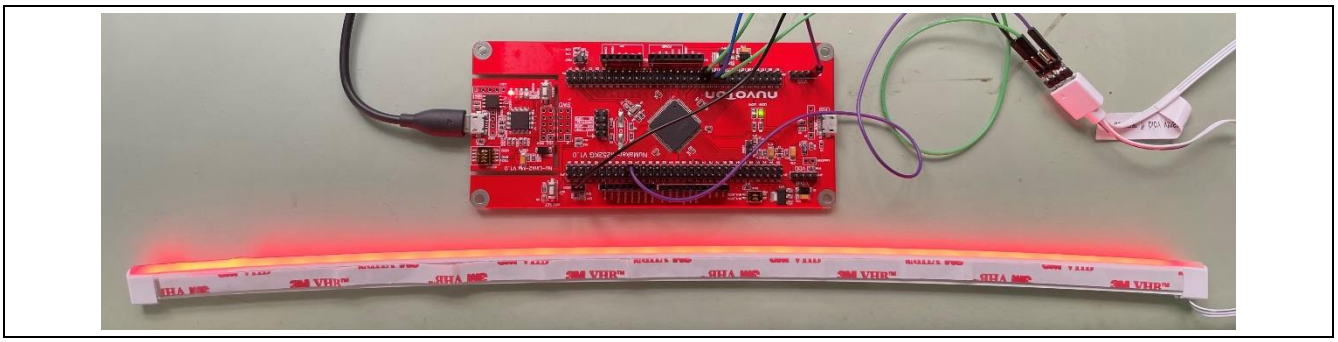

Figure 1-14 result of Red Light Demo

Figure 1-15 Result of Blue Light Demo

1.3.4 Rainbow Cycle Demo

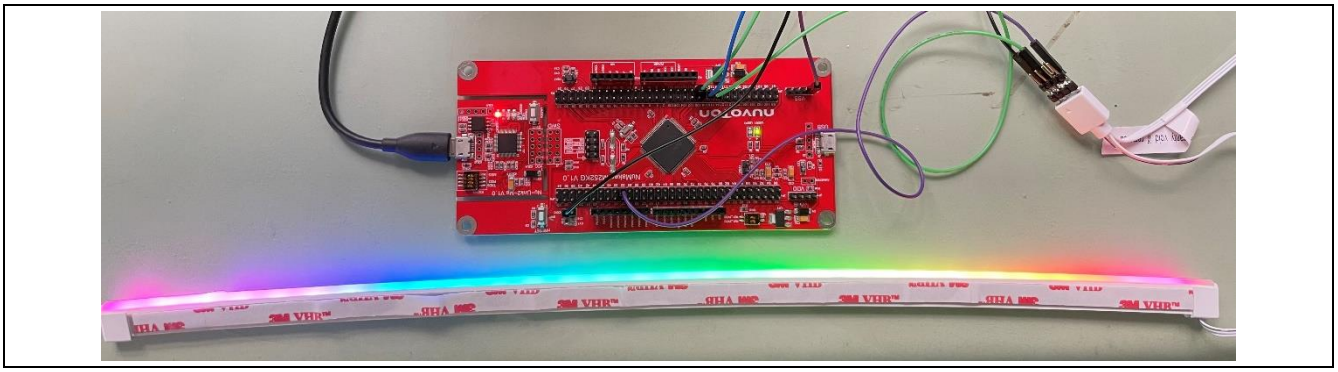

Figure 1-16 Result of Rainbow Cycle Demo

1.3.5 Theater Chase Rainbow Demo

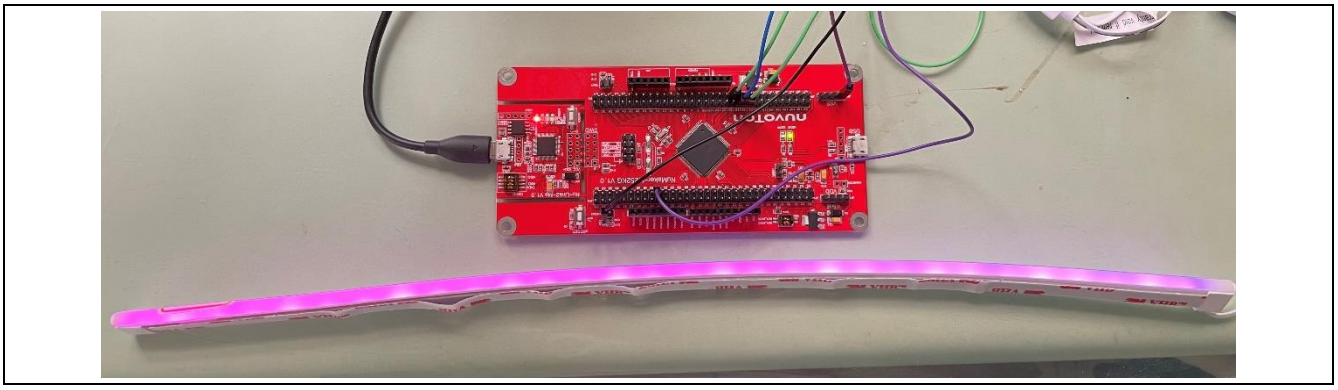

Figure 1-17 Result of Theater Chase Rainbow Demo

1.3.6 Feedback Mode Demo

The timing of the ARGB2 LED specification's entry into Feedback requires 20 μs (high level) + 10 μs (low level) + 50 μs (high level), as shown in Figure 1-18.

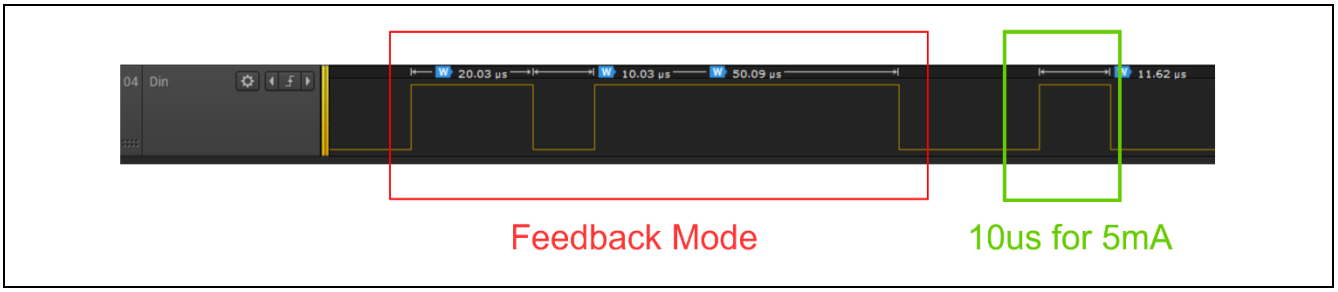

Figure 1-18 Results of Feedback Mode Waveforms and Pulse Width Represent the Maximum Current Consumption

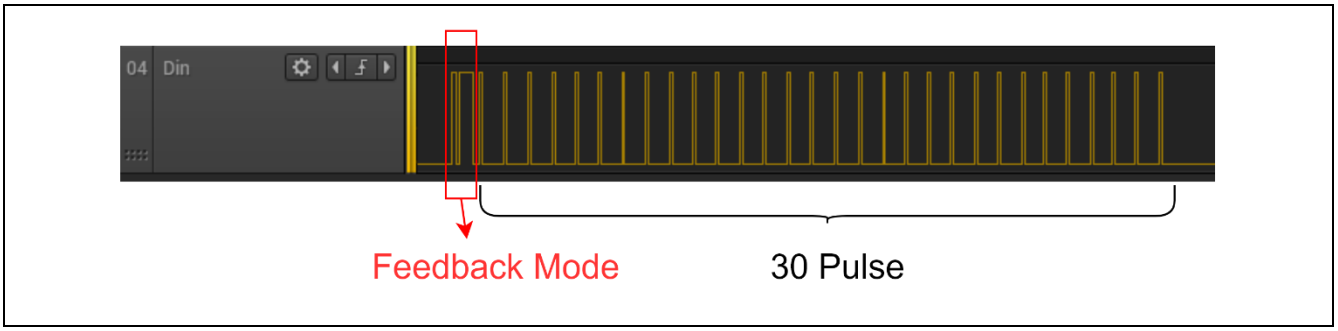

Figure 1-19 Result of ARGB2 Strip Response Pulse

1.3.7 TH20 SET Mode Demo

The MCU transmits a 20 µs (high level) + 250 µs (low level) signal to the ARGB2 LED strip to enter TH20 SET mode, followed by 24 bits data.

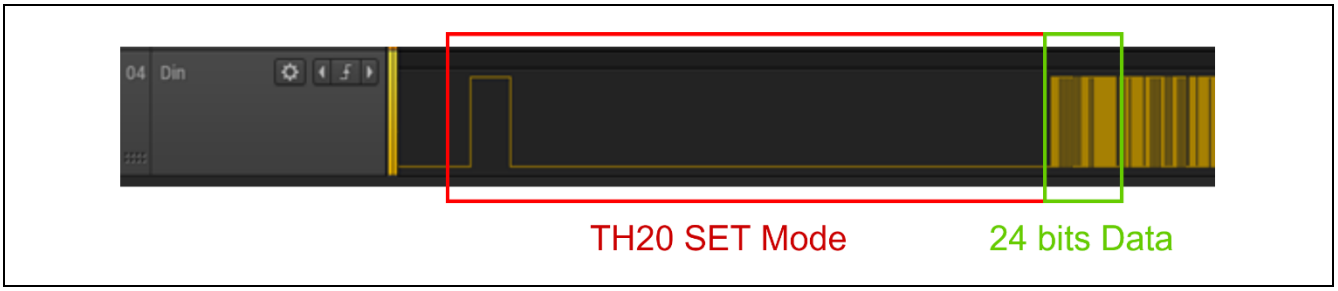

Figure 1-20 Result of TH20 SET Waveform

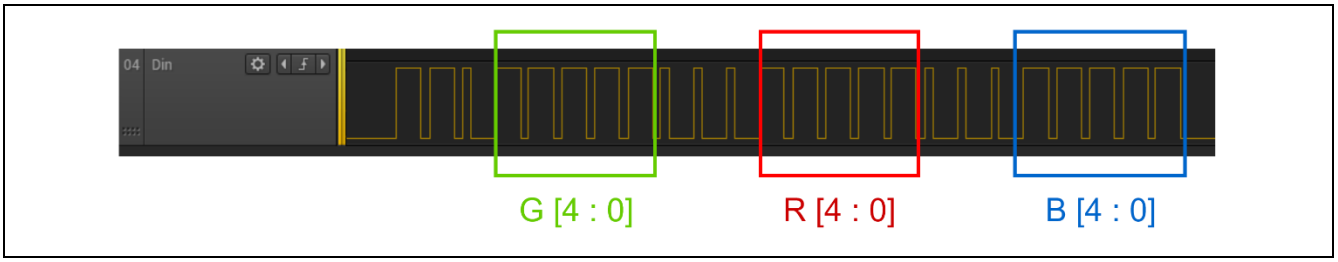

Figure 1-21 Result of TH20 SET 24 Bits Data Waveform

1.3.8 Parallel Mode (Y Cable) Demo

The ARGB2 specification supports multiple simultaneous control modes (up to 15 LED strips can be specified for control). strips that can identify the IDs dynamically assigned to each strip (set dynamic ID command). After the command is completed, the MCU sends "Clear ID", "Check ID", "Specify ID" commands to each strip.

The MCU sets the output timing 40 µs + 10 µs + 8 bits (Command + ID) into Y Cable mode, and the waveform is shown in Figure 1-22.

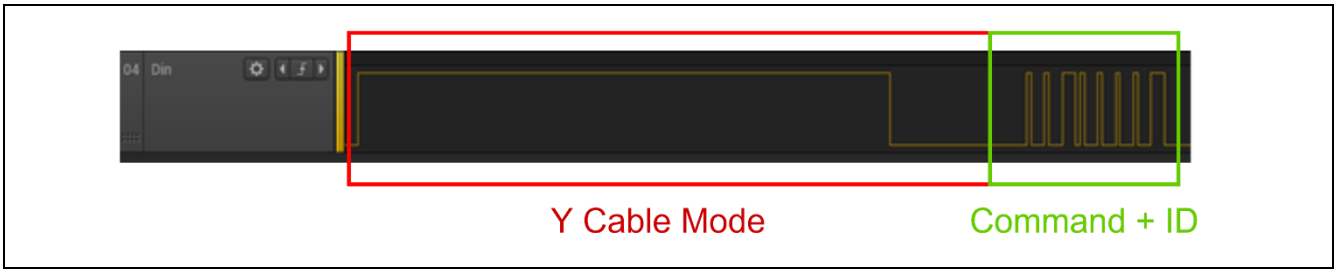

Figure 1-22 Result of Parallel Mode (Y Cable) and 8 Bits (Command + ID) Waveform

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

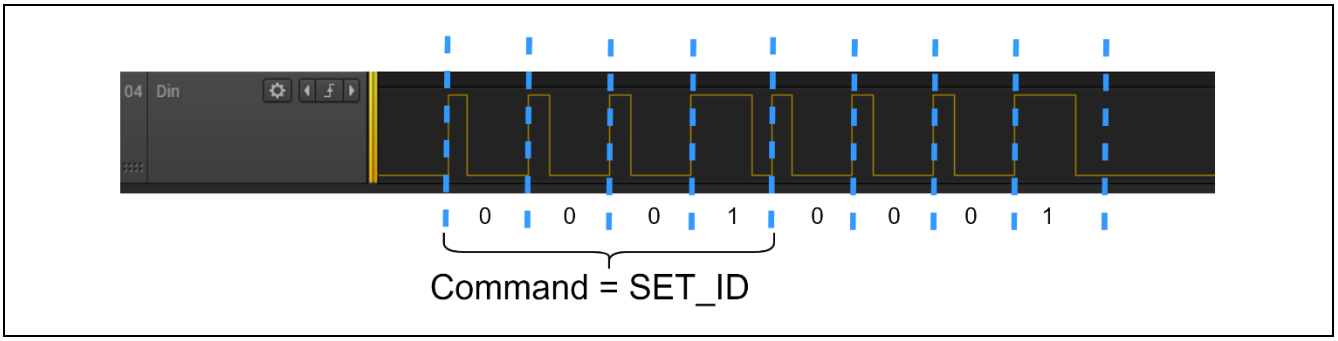

Figure 1-23 Result of SET_ID Command Waveform

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

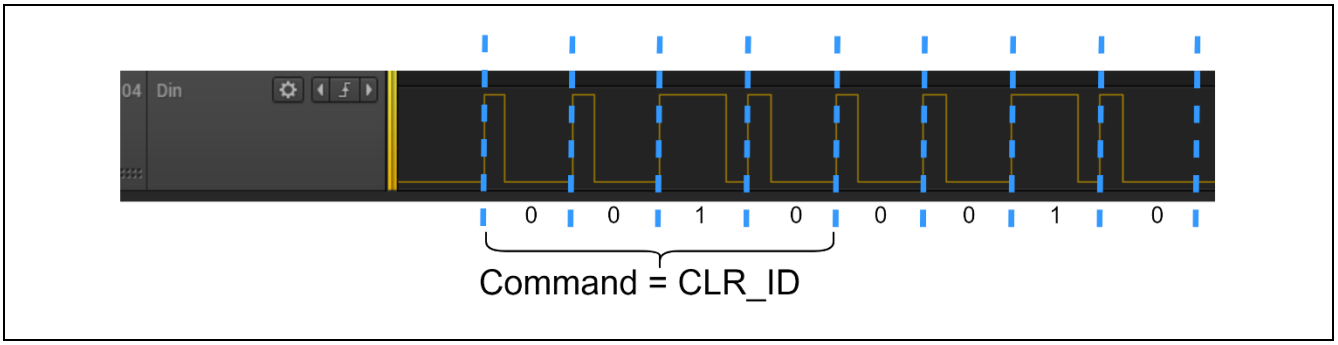

Figure 1-24 Result of CLR_ID Command Waveform

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

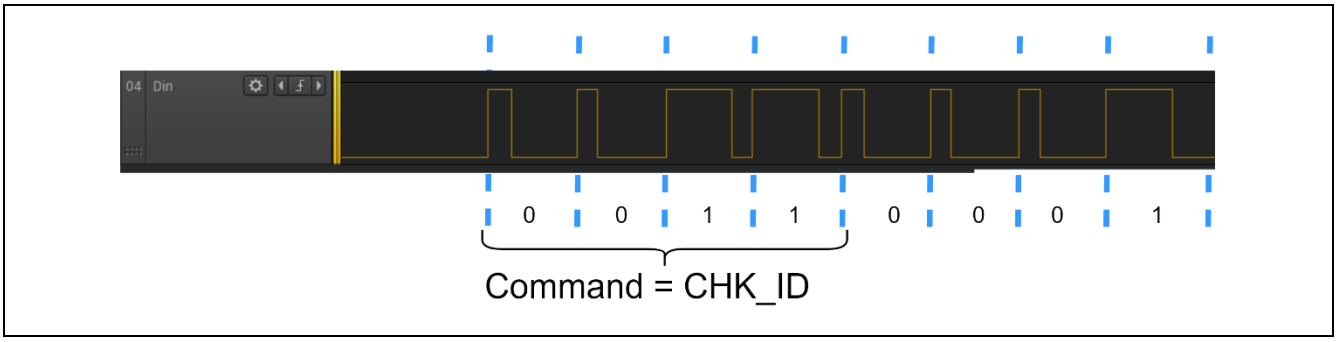

Figure 1-25 Result of CHK_ID Command Waveform

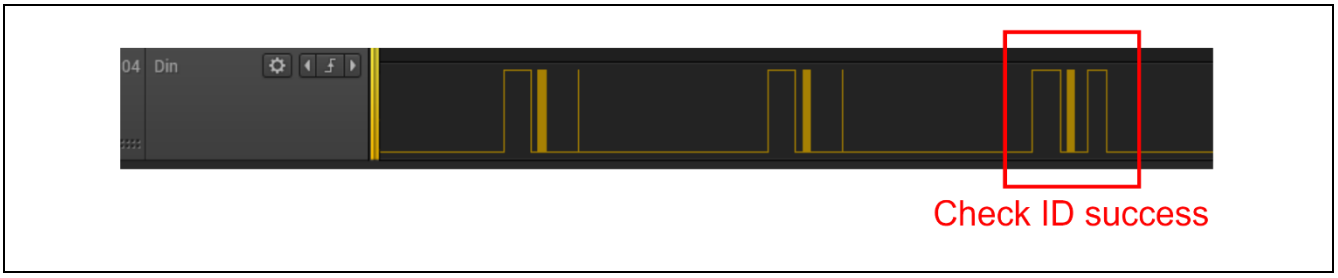

Figure 1-26 Result of Confirm Check ID Success Waveform

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

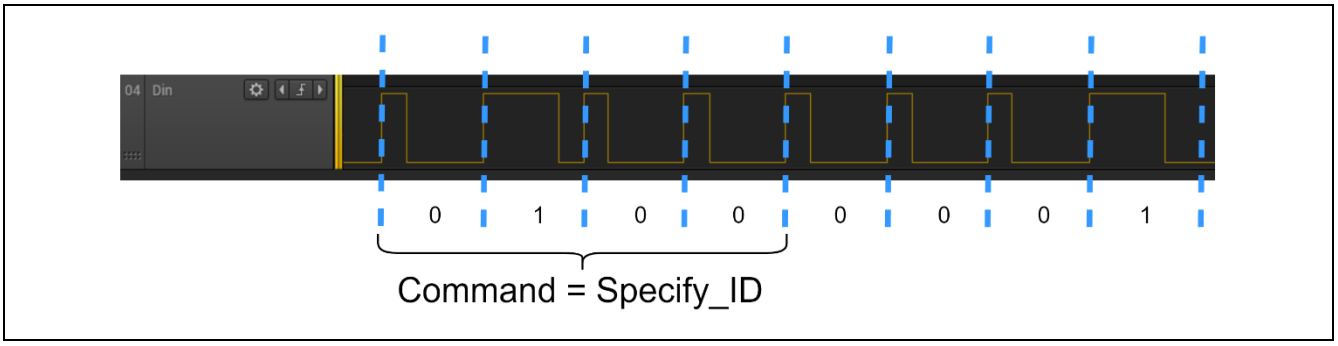

Figure 1-27 Result of Specify_ID Command Waveform

2. Code Description

2.1 Main Code Description

The code workflow is as follows:

1. The M252 series related peripherals are initialized in the *main.c*, including HIRC, PSIO, TIMER0, and UART0
2. In the ARGB2 StripTestMenu() function, the while() loop selects to perform the GRB loop, rainbow cycle, theater chase rainbow, Feedback mode, TH20 SET mode, and parallel Y Cable mode functions
3. Logic '0', logic '1' waveforms are implemented using PSIO_SendDataWithSize (uint32_t u32DataSize, uint32_t u32Data)

```
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();
}
```

Use *PSIO_ARGB2_Driver.c* PSIO_StripInit() to initialize PSIO Slot Control and PSIO pin.

```
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);
}
```

Use *PSIS_ARGB2_Driver.c* PSIO_Strip_OpenOutput() when initializing the output mode. Use PSIO_SET_GENCTL() to set the pin controller, mode, start state and end state. PSIO_SET_WIDTH() and PSIO_SET_OUTPUT_DEPTH() are related to the data format. In this

example, the output width is initially set to 8 bits, the input width is 1 bit, and the depth is both 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);
}
```

Use *main.c* while() loop to implement the selection function execution.

```
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();
        }
    }
}
```

The waveform signal function of output logic '0' and logic '1' requires three different actions, namely, output high level, output data from the buffer and output low level, which are described and summarized in Table 2-1. Slot 0 ~ Slot 2 can be regarded as 1 bit of ARGB2 LED protocol. PSIO_SET_WIDTH() sets the output width according to the size of the output data. For example, to set the color of the ARGB2 LED, users must set the output width to 24 bits. PSIO_SET_SCCTL() uses the loop function of PSIO to replicate this behavior 23 times, that is, output 24 bit data. After the setting is completed, use the software triggers PSIO_START_SC() to start the slot controller execution.

	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

Table 2-1 Set Output Logic '0', Logic '1' with PSIO

```
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 Light Effect Code Description

2.2.1 GRB Cycle Function

PSIO_StripSetRGBColor(uint8_t u8Red, uint8_t u8Green, uint8_t u8Blue) 24 bits of data is used to combine the color of a single strip. Users can set a single color of a single strip, where PIXEL_NUM represents the number of LED strips to be set. Use the Timer function TIMER_Delay() to adjust the interval between LED display.

```
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 Rainbow Cycle Function

PSIO_StripSetPixelColor (uint16_t n, uint32_t GRBColor) function is used to store the color of each light. PIXEL_NUM is the number of lights to be set, and the PSIO_StripWheel(uint8_t wheelPos) function is used to set the color storage position to generate the wheel effect. The PSIO_StripShow() function is used to output the BIT0 and BIT1 waveforms to the ARGB2 LED strip to achieve the rainbow cycle effect.

```
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 Theater Chase Rainbow Function

Similar to the Rainbow Cycle function, one more for loop is in the outermost layer, and the PSIO_StripSetPixelColor (uint16_t n, uint32_t GRBColor) function is used to make it colorless for a period to achieve the effect of theater chase.

```
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. 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 5.28.0.0

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M251KG V1.0
  - ARGB2 LED Strip
- Pin Connect
  - Use the PSIO pin to simulate the ARGB2 strip timing diagram.

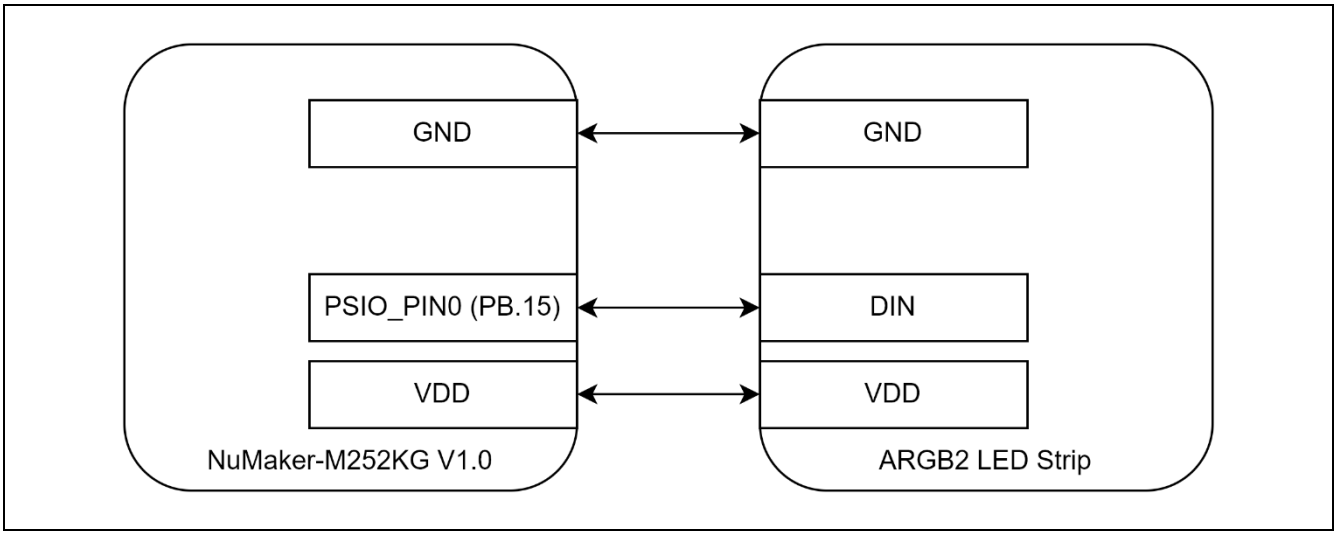

Figure 3-1 M252 PSIO_PIN0 and ARGB2 LED DIN Connections

- The operating voltage of the ARGB2 LED strip is 5V, so the supply voltage of the NuMaker-M252KG evaluation board for ARGB2 must be changed to 5V. Users can change the ICE JPR1 (MCU VCC) resistor on the Nu-Link2-Me to output 5V, or use the 5V pin of the USB to TTL module.

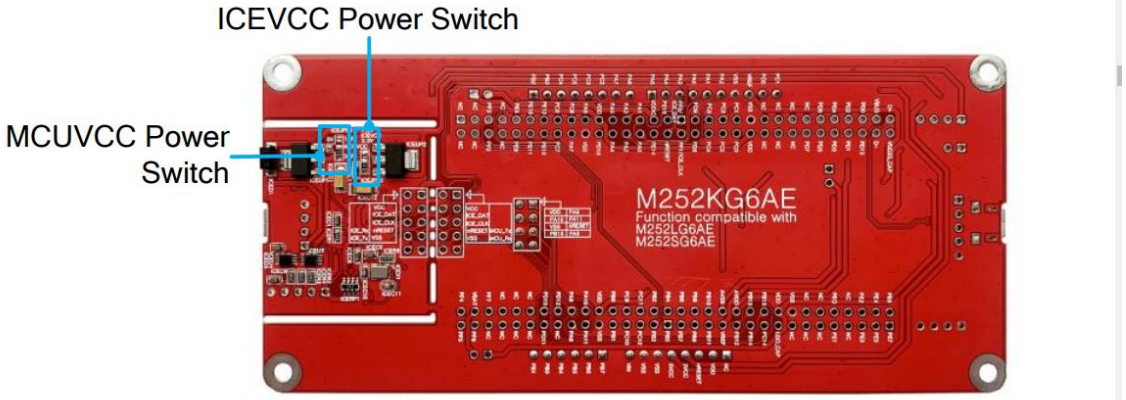

Figure 3-2 MCU VCC Power Switched to 5V

4. Directory Information

The directory structure is shown below.

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

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 *M252_PSIO_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
2022.06.02	1.00	Initial version.

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.