M0A23 GPIO 实现 ARGB LED 跑马灯效果

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 M0A23 GPIO 模拟 ARGB LED 的时序并实现跑马灯效果
BSP 版本	M0A21_Series_BSP_CMSIS_V3.01.000
开发平台	Bling Bling Board Ver2.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

此范例代码使用 M0A21/M0A23 系列的 GPIO 来仿真 ARGB 的时序，并演示 ARGB LED 跑马灯效果。此范例代码使用一组 Timer 来定时更新 ARGB LED 跑马灯的颜色，使用一组 GPIO 来模拟 ARGB LED 操作，并一组 UART 用来输入显示文字。

1.1 原理

1.1.1 ARGB LED 数据传输时间(TH+TL=1200 ns ± 160 ns)

ARGB LED 时序图如图 1-1 与图 1-2 所示。逻辑「1」定义为高电平 300ns ± 80ns 与低电平 900ns ± 80ns; 逻辑「1」定义为高电平 900 ns ± 80ns 与低电平 300ns ± 80ns。图 1-3 重置时间需要大于 50 us。

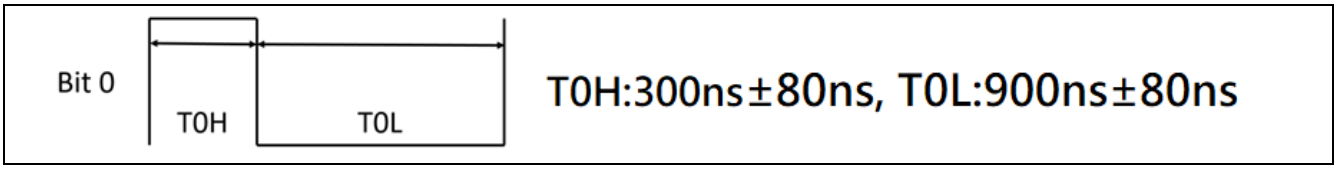

图 1-1 逻辑「0」 (Bit 0)

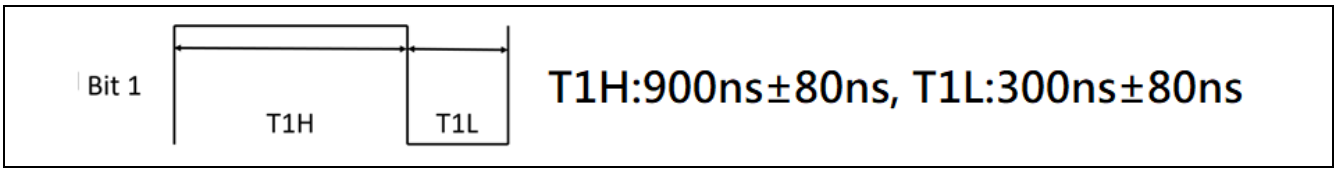

图 1-2 逻辑「1」 (Bit 1)

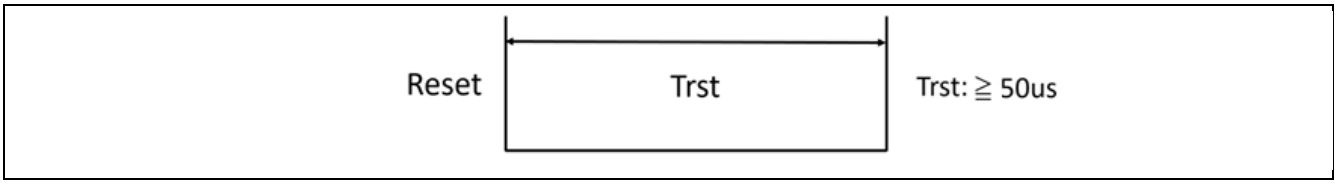

图 1-3 重置 (Trst)

1.1.2 ARGB LED 特性介绍及连接方式

用户只需一条信号线就可以控制 ARGB LED 颜色。与传统的 RGB LED 相比，ARGB LED 更容易设计并节省 Layout 空间。连接方式如图 1-4，从 MCU 的 GPIO 连接到 LED1 的输入(DIN)，然后 LED1 输出(DOUT)会串接到 LED2 的输入(DIN)。

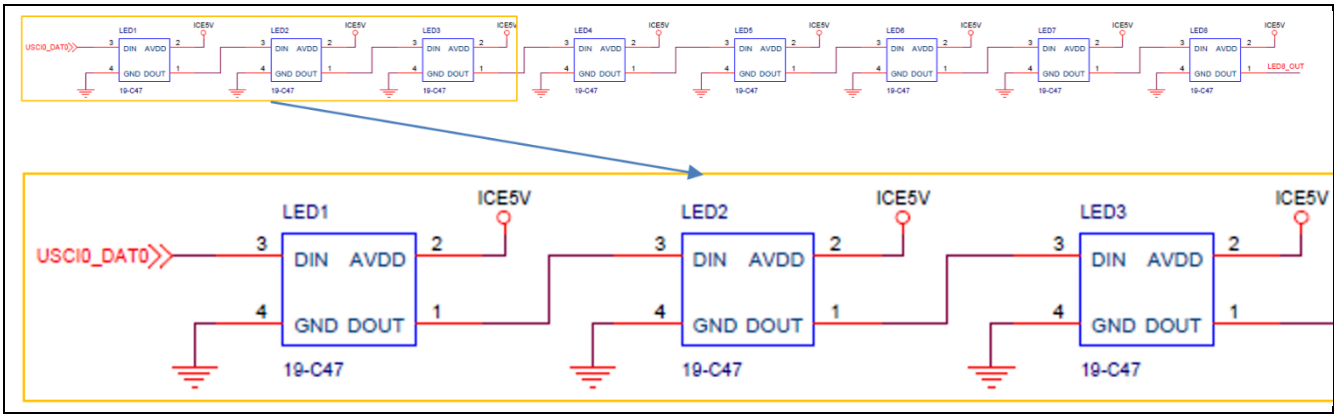

图 1-4 ARGB LED 串接

1.1.3 GPIO 时钟频率与逻辑 Bit 设计

使用 GPIO 控制 ARGB LED，主要透过 NOP() 命令让高低电平往后延迟一段时间，程序中设定系统时钟 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() 命令。

如图 1-1 和图 1-2，逻辑「0」定义为高电平 300ns ± 80ns(220ns≤TH0≤380ns) 与低电平 900ns ± 80ns(820ns≤ TL0≤980ns);逻辑「1」定义为高电平 900 ns ± 80ns (820ns≤TH1≤980ns)与低电平 300ns ± 80ns(220ns≤TL1≤380ns)。此范例代码中逻辑「0」子程序为 ARGB_sendBIT0()，逻辑「1」子程序为 ARGB_sendBIT1()，逻辑「0」及逻辑「1」子程序中使用 NOP() 命令，执行一个 NOP()时间约 1 / 48MHz (20.83ns)。

由上述得知在设计时逻辑「0」高电平最少需要 220/20.83≈11NOP()，最多需要 380/20.83≈19NOP()。逻辑「1」低电平最少需要 820/20.83≈40NOP()，最多需要 980/20.83≈48NOP()。逻辑「0」高电平最少需要 820/20.83≈40NOP()，最多需要 980/20.83≈48NOP()。逻辑「1」低电平最少需要 220/20.83≈11NOP()，最多需要 380/20.83≈19NOP()，ARGB LED 的时钟频率可依上述条件设计。

如图 1-5 逻辑「0」高电平为 280ns(280/20.83≈14NOP())，低电平为 980ns(980/20.83≈48NOP())，逻辑「1」高电平为 920ns(920/20.83≈45NOP())，低电平为 340ns(340/20.83≈17NOP())。

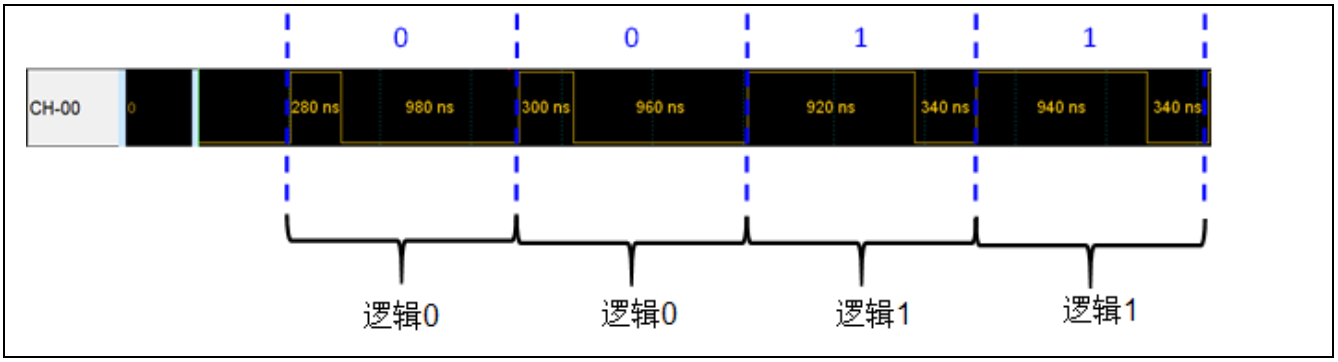

图 1-5 使用 NOP 命令让高低电平延迟一段时间

此范例代码使用 Keil C，编译时会将 C 语言转换成汇编语言，由于 C 语言的命令运行时间如循环或是赋值…等可能会需要耗数个 NOP()，故在实际设计程序中需要一些调整 NOP()数量。

此范例代码 ARGB_sendOne()子程序中，用来判断逻辑「0」和逻辑「1」，其中 ARGB_sendBIT0()为逻辑「0」子程序，ARGB_sendBIT1()为逻辑「1」子程序，在 ARGB_sendBIT0()子程序中高电平含有 11 个 NOP()，由于受到 ARGB_sendOne()子程序中的变量值指定及判断式(if…else)及循环(for)影响执行中多出 7 个 NOP()，故逻辑「0」高电平会是(11+7=18)18 个 NOP()，如图 1-6 实际逻辑「0」高电平 357ns(357/20.83≈18 NOP())。在 ARGB_sendBIT0()子程序中低电平含有 29 个 NOP()，由于受到 ARGB_sendOne()子程序中的赋值及判断语句(if…else)及循环(for)影响执行中多出 14 个 NOP()，故逻辑「0」低电平会是(29+14=43)43 个 NOP()，如图 1-7 实际逻辑「0」低电平 892ns(892/20.83≈43 NOP())。

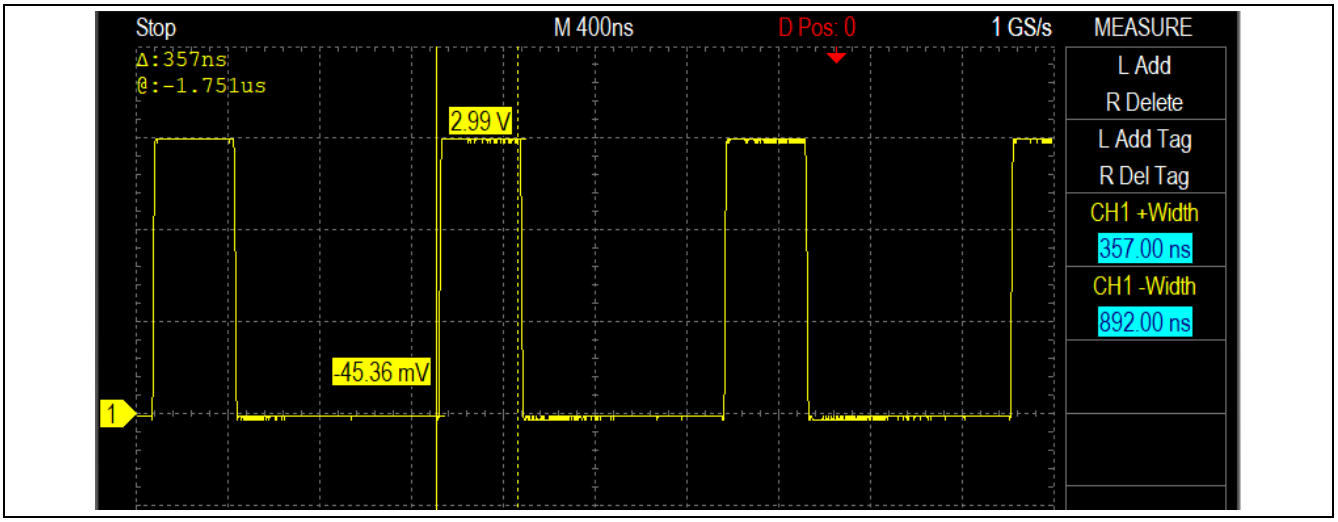

图 1-6 逻辑「0」高电平波形

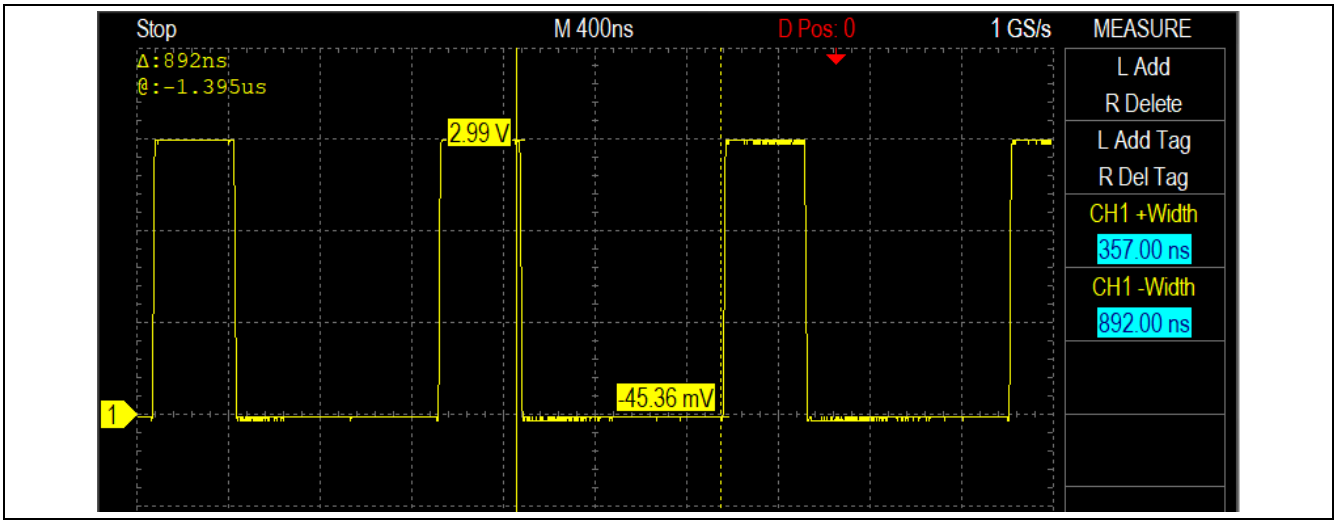

图 1-7 逻辑「0」低电平波形

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 1-8 逻辑「1」执行结果

图 1-9 逻辑「0」执行结果

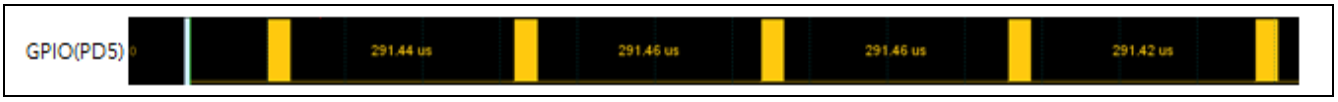

图 1-10 重置 Trst 执行结果

1.2.2 串口调试助手执行结果

使用串口调试助手软件，波特率设定为 115200，串口调试助手软件执行结果如图 1- 所示。

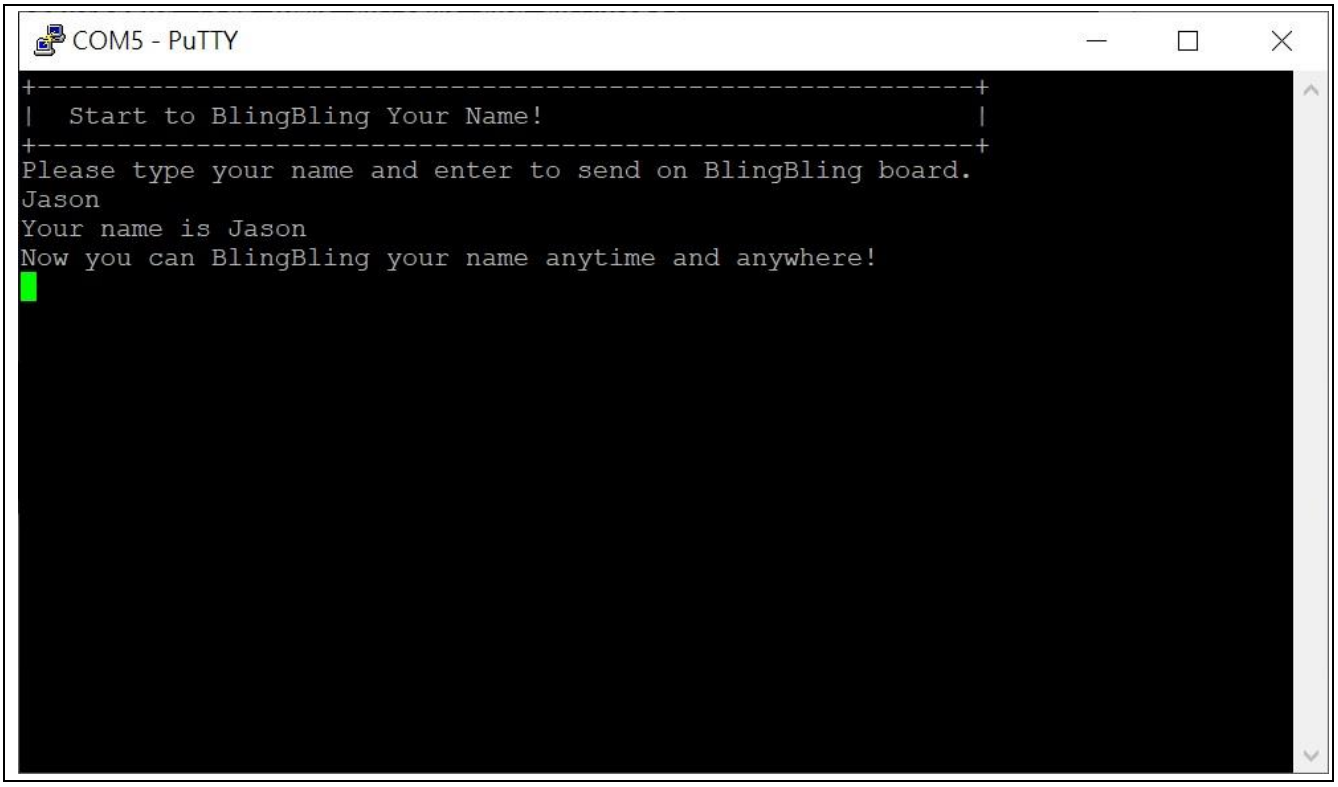

图 1-11 串口调试助手软件显示画面

1.2.3 跑马灯执行结果

ARGB LED 跑马灯执行结果，如图 1-所示。

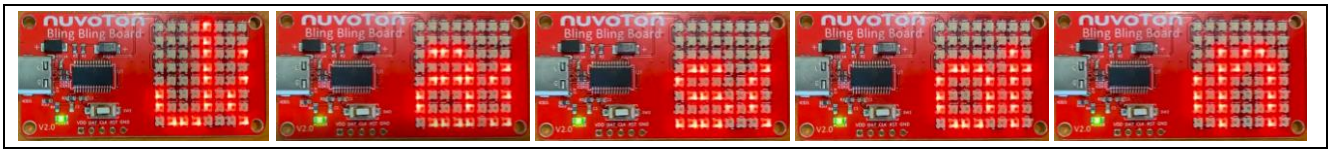

图 1-12 ARGB LED 跑马灯执行结果

2. 代码介绍

2.1 主代码说明

代码工作流程如下：

1. *main.c* 中初始化 M0A23 系列相关外设，包含 HIRC、UART0 与 Timer0
2. 使用 ARGB_init() 来设定 GPIO 和模拟 ARGB LED 时序
3. 使用 Read_DataFlash() 来判断 Data Flash 里面是否已有数据
4. 在 While 循环判断是否有新的字符串输入

```
int32_t main(void)
{
    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

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

    /* Init ARGB */
    ARGB_init();

    TIMER_Delay(TIMER0, 1000000);

    Read_DataFlash();

    Convert_FlashData();

    printf("+------------------------------------------------------------+\n");
    printf("|  Start to ""BlingBling"" Your Name!                          |\n");
    printf("+------------------------------------------------------------+\n");
    printf("Please type your name and enter to send on BlingBling board.\n");

    while(1)
    {
        if(receiveData == FALSE)
        {
            put_string(String);
        }
        else
        {
            Write_DataFlash();

            receiveData = FALSE;
        }
     }
}
```

当 Read_DataFlash() 被调用时，会去读取指定地址的 Flash 数据。

```
void Read_DataFlash()
{
    SYS_UnlockReg();
```

```
    /* Enable FMC ISP function */
    FMC_Open();

    /* Enable Data Flash and set base address. */
    set_data_flash_base(DATA_FLASH_BASE);

    FMC_ENABLE_AP_UPDATE();

    flash_read(DATA_FLASH_BASE, DATA_FLASH_BASE + RX_BUFFER_SIZE);

    FMC_DISABLE_AP_UPDATE();

    FMC_Close();
}
```

当 Convert_FlashData() 被调用时，判断字符串字符数组是否有数据

```
void Convert_FlashData()
{
    uint8_t i = 0;

    if((uint8_t)flashData[0] != 0xFF)
    {
        for(i = 0; i < RX_BUFFER_SIZE; i++)
        {
            String[i] = 0;
        }

        for(i = 0; i < RX_BUFFER_SIZE; i++)
        {
            if((uint8_t)flashData[i] != 0xFF)
                String[i] = flashData[i];
            else
                break;
        }

        for(i = 0; i < RX_BUFFER_SIZE; i++)
        {
            flashData[i] = 0;
        }
    }
}
```

当 put_string(char inputString[]) 被调用时，会将字符串存放到跑马灯的数组。

```
void put_string(char inputString[])
{
    int8_t count;

    for(count = 0; count < strlen(inputString); count++)
    {
        select_word(inputString[count]);

        put_char(count, strlen(inputString));
    }

    if(showArray_column >8)
        Run_column = showArray_column;
    else
        Run_column = 8;

    ARGB_showString();
```

```
    TIMER_Delay(TIMER0, 1000000);

    color++;

    if(color == 3)
    {
        color = 0;
    }

    showArray_row = 0;
    showArray_column = 0;
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M0A21_Series_BSP_CMSIS_V3.01.000
- IDE 版本
  - Keil uVersion 5.36

3.2 硬件需求

- 电路组件
  - Bling Bling Board Ver2.0
  - ARGB LED
- 线路示意图

使用 GPIO 脚位来模拟 ARGB LED 时序图。

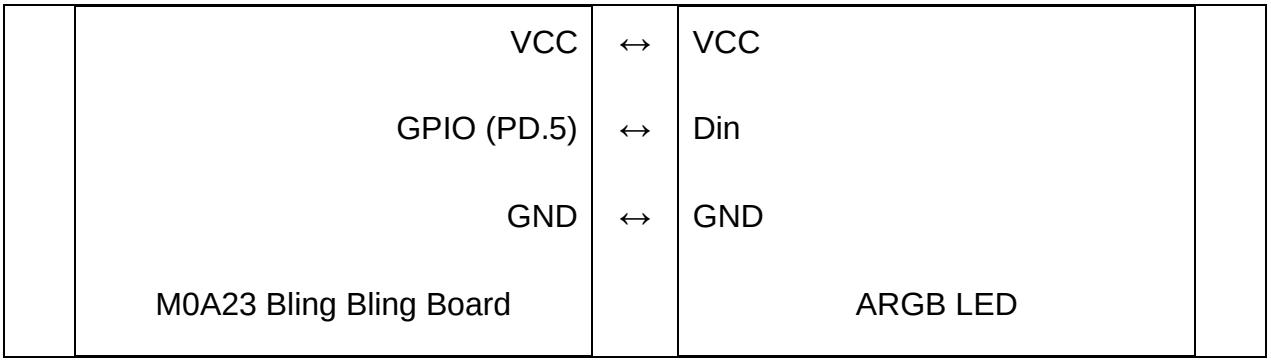

图 3-1 线路示意图

4. 目录信息

EC_M0A23_GPIO_ARGB_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	
BlingBling	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 *BlingBling.uvprojx*。
2. 进入编译模式接口
   - 编译
   - 下载代码至内存
   - 进入 / 离开除错模式
3. 进入除错模式接口
   - 执行代码

6. 修订纪录

Date	Revision	Description
2023.12.15	1.00	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.