M251 DMX512 控制范例

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 M251 实现 DMX512 控制 ARGB2 LED
BSP 版本	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.02.007
开发平台	NuMaker-M251KG V1.1

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

DMX512 协议是由美国舞台灯光协会（USITT）提出的一种数据调光协议，它给出了灯光控制器与灯具设备之间通信的协议标准。应用它能够对舞台、剧场、演播室等场所的调光器及其他控制设备进行数字控制，适用于一主多从式控制系统，其连线简单可靠性高。本文描述如何使用 M251 UART 解码 DMX512 数据包，并控制 ARGB2 LED 灯。

1.1 原理

1.1.1 ARGB2 LED 数据传输时间（TH + TL = 1.2 µs ± 600 ns）

ARGB2 LED 时序图如图 1-1 与图 1-2 所示。逻辑 ‘0’ 定义高电平 0.3 µs ± 0.15 µs 与低电平 0.9 µs ± 0.15 µs，逻辑 ‘1’ 定义为高电平 0.9 µs ± 0.15 µs 与低电平 0.3 µs ± 0.15 µs。

在使用 PSIO 控制 ARGB2 LED，必须先设定好时序规则（slot），按照预期的时序运作，将时间基本单位划分约 0.3 µs 可以造出所需的波形。时序计算公式如下：

- PSIO clock source 选择 PCLK1，PCLK1 clock 计算方式 HIRC (48 MHz) / 4 = 12 MHz
- PSIO clock 使用 PCLK1 除频 4 = 3 MHz
- PSIO one slot cycle time = 1 / 3 MHz = 0.33 µs

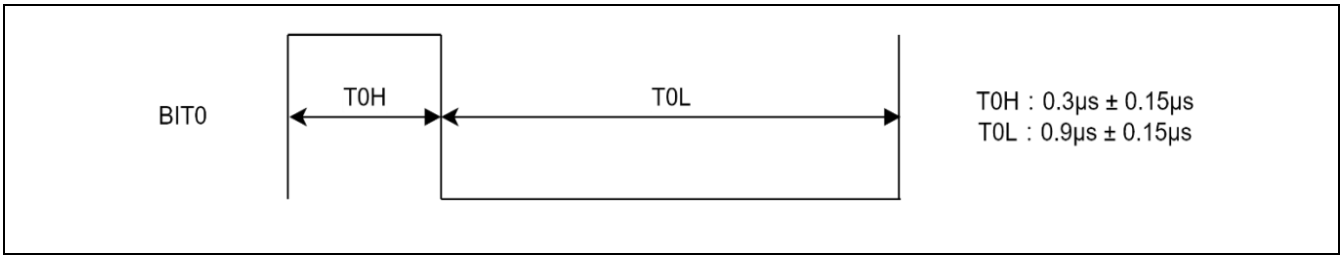

图 1-1 逻辑 ‘0’ (bit 0)

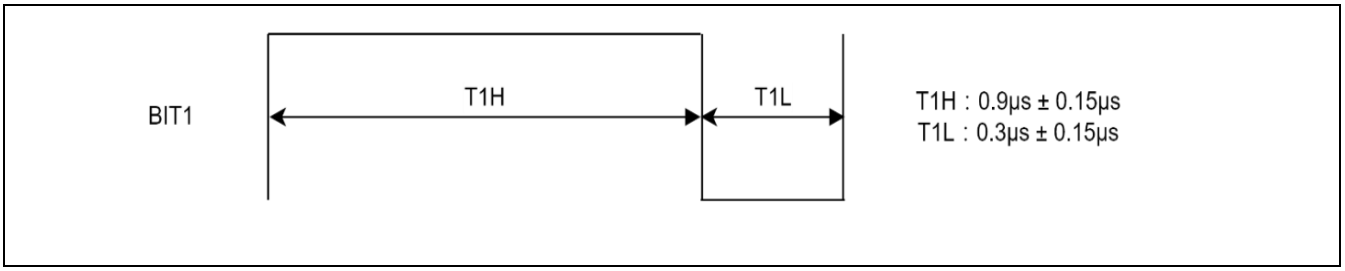

图 1-2 逻辑 ‘1’ (bit 1)

1.1.2 DMX512 数据帧

一个字节为一个数据帧，称作一个控制通道，可以控制灯光设备的一个或几个功能。一个数据帧由 1 个开始位（S），8 个数据位（D0~D7）和 2 个结束位（E）共 11 位构成，采用单向异步串行传输，如图 1-3 所示：

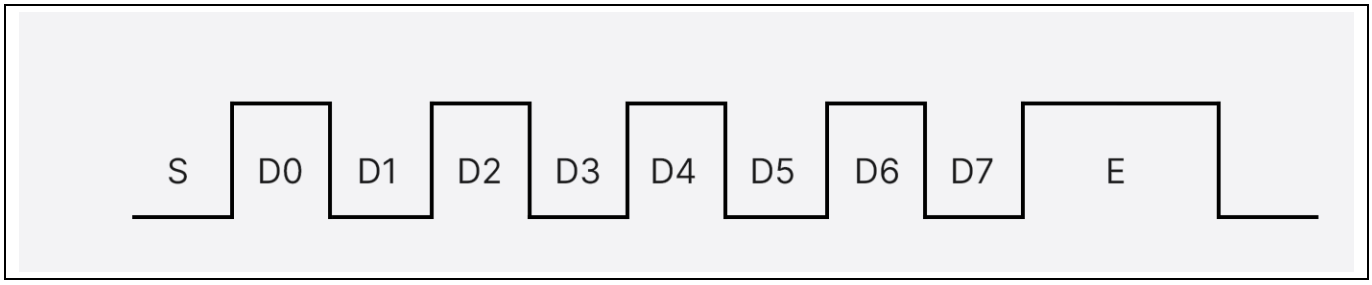

图 1-3 DMX512 数据帧

D0~D7 为 8 位控制数据，控制灯光亮度时可产生 256 个亮度等级，0 对应灯光最暗，255 对应最亮。位宽 4 us，每帧 11 位，指令帧宽度为 44 us，传输速率为 1/44=250 kbps。

1.1.3 DMX512 信息包

一个完整的 DMX512 数据包由一个 MTBP 位，一个 Break 位，一个 MAB 位，一个 SC 帧和 512 个数据帧构成。如下图图 1-4：

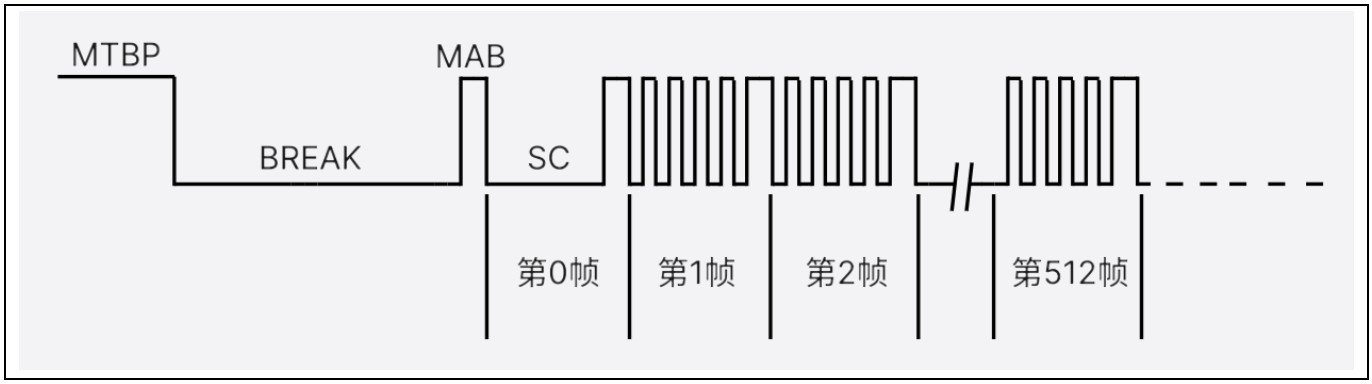

图 1-4 DMX512 数据包

- MTBP（Mark Time Between Packets）位，标志着一个完整的数据包发送完毕，高电平有效。
- Break 位，对应一个数据包结束后的程序复位阶段，宽度不少于两个帧（22 比特）。

- MAB 位，由于每一个数据帧的第一位(即开始位)为低电平，所以必须用一个高电平脉冲间隔前后两个低电平脉冲，这个起间隔、分离作用的高电平脉冲即 MAB（Mark After Break）。
- SC（Start Code）帧，同数据帧一样，也是由 11 位构成，除最后的两个高电平的结束位之外，其他 9 位全部是低电平，通常将其叫做开始帧、第 0 帧或第 0 通道，不控制设备。

1.1.4 UART 解码 DMX512 数据

1. 设置 UART 波特率为 250 kbps，8 位数据长度，2 个 STOP 位。
2. 使能 RDA/RLS 中断。
3. 接收到 BREAK 位和 SC 帧后开始接收控制通道数据帧（1~512）。
4. 控制通道数据帧数据用于控制 LED 亮度。

2. 代码介绍

PSIO_StripInit()函数分别初始化 2 条灯带对应的 PSIO Slot Control 和 PSIO PIN。

```
void PSIO_StripInit(void)
{
    uint8_t i;

    gsPSIO_ARGB2_Cfg[0].u8SlotCtrl = PSIO_SC0;
    gsPSIO_ARGB2_Cfg[0].u8DataPin = PSIO_PIN0;

    gsPSIO_ARGB2_Cfg[1].u8SlotCtrl = PSIO_SC1;
    gsPSIO_ARGB2_Cfg[1].u8DataPin = PSIO_PIN1;

    for (i = 0; i < 2; i++)
    {
        PSIO_Strip_OpenOutput(gsPSIO_ARGB2_Cfg[i], PSIO_PIN_ENABLE);
    }
}
```

PSIO_Strip_OpenOutput()函数在初始化 Output Mode 时，使用 PSIO_SET_GENCTL() 设定 Pin 的状态，内容为使用的 Pin、Slot controller、模式、初始化状态及结束状态等。PSIO_SET_WIDTH()与 PSIO_SET_OUTPUT_DEPTH()则是与数据的格式有关，此范例初始设定输出宽度为 8 bit、输入宽度为 1 bit，深度皆为 1。

```
static void PSIO_Strip_OpenOutput(S_PSIO_ARGB2_STRIP_CFG_T gsPSIO_Cfg, uint32_t u32Switch)
{
    /* PSIO UART TX PIN general setting */
    PSIO_SET_GENCTL(PSIO,
                    gsPSIO_Cfg.u8DataPin,
                    u32Switch, //PSIO_PIN_ENABLE
                    gsPSIO_Cfg.u8SlotCtrl,
                    PSIO_OUTPUT_MODE,
                    PSIO_LOW_LEVEL,
                    PSIO_LOW_LEVEL);

    /* Set data order ad MSB */
    PSIO_SET_ORDER(PSIO, gsPSIO_Cfg.u8DataPin, PSIO_MSB);

    /* Set output data buffer width as 8 bit, input data width as 1 bit*/
    PSIO_SET_WIDTH(PSIO, gsPSIO_Cfg.u8DataPin, 1, 8);

    /* Set output data buffer depth as 1 */
    PSIO_SET_OUTPUT_DEPTH(PSIO, gsPSIO_Cfg.u8DataPin, PSIO_DEPTH1);

    /* Set slot controller trigger source as software trigger */
    PSIO_SET_TRIGSRC(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SW_TRIGGER);
}
```

PSIO_SendDataWithSize()函数在输出逻辑 ‘0’ 、逻辑 ‘1’ 波形信号需要三种不同的动作，分别是输出高电平、从缓冲区输出数据及输出低电平，时序的规划如图 1-1 和图 1-2 所述。Slot 0 ~ Slot 2 可视为 ARGB2 LED 协议的 1 bit。PSIO_SET_WIDTH() 根据输出的数据大小设定输出宽度，例如设定 ARGB2 LED 的颜色需要设定输出宽度为 24 bit。PSIO_SET_SCCTL()利用 PSIO 的循环功能，将此行为再复制 23 次，即输出 24 bit 数据。设定完成后，用软件触发 PSIO_START_SC()的方式，让 slot controller 开始执行。

```
void PSIO_SendDataWithSize(S_PSIO_ARGB2_STRIP_CFG_T gsPSIO_Cfg, 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_Cfg.u8DataPin, &sDataConfig);

    /* Set slot count */
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SLOT0, 1);
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SLOT1, 2);
    PSIO_SCSLOT_SET_SLOT(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SLOT2, 1);

    /* Enable repeat slot0 ~ slot2 u32DataSize - 1 times */
    PSIO_SET_SCCTL(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SLOT0, PSIO_SLOT2,
                   u32DataSize - 1, PSIO_REPEAT_DISABLE);

    /* Set out data width as u32DataSize bit */
    PSIO_SET_WIDTH(PSIO, gsPSIO_Cfg.u8DataPin, 0, u32DataSize);

    /* Set out data depth as 1 */
    PSIO_SET_OUTPUT_DEPTH(PSIO, gsPSIO_Cfg.u8DataPin, PSIO_DEPTH1);
```

```
    /* Set output data to buffer */
    PSIO_SET_OUTPUT_DATA(PSIO, gsPSIO_Cfg.u8DataPin, u32Data);

    /* Trigger slot controller */
    PSIO_START_SC(PSIO, gsPSIO_Cfg.u8SlotCtrl);

    /* Wait for slot controller is not busy */
    while (PSIO_GET_BUSY_FLAG(PSIO, gsPSIO_Cfg.u8SlotCtrl));

    /* Stop slot controller */
    PSIO_STOP_SC(PSIO, gsPSIO_Cfg.u8SlotCtrl);

    /* Disable repeat slot0 ~ slot2 */
    PSIO_SET_SCCTL(PSIO, gsPSIO_Cfg.u8SlotCtrl, PSIO_SLOT_DISABLE, PSIO_SLOT_DISABLE,
                   0, PSIO_REPEAT_DISABLE);
}
```

DMX512_UART_Init()函数对 UART 初始化。

```
void DMX512_UART_Init(void)
{
    /* Configure DMX512_UART_nRTS as OUTPUT mode */
    GPIO_SetMode(DMX512_UART_nRTS_PORT, DMX512_UART_nRTS_BIT, GPIO_MODE_OUTPUT);
    DMX512_UART_nRTS_PIN = 0;

    /* Set multi-function pins for DMX512_UART RX and TX */
    DMX512_UART_RX_MFP &= ~(DMX512_UART_RX_MFP_MSK | DMX512_UART_TX_MFP_MSK);
    DMX512_UART_TX_MFP |= (DMX512_UART_RX_MFP_FUNC | DMX512_UART_TX_MFP_FUNC);

    /* Configure DMX512_UART and set DMX512_UART baud rate */
    UART_Open(DMX512_UART, DMX512_UART_BAUDRATE);
    DMX512_UART->LINE = UART_WORD_LEN_8 | UART_PARITY_NONE | UART_STOP_BIT_2;

    /* Enable RDA\RLS Interrupt */
    NVIC_EnableIRQ(DMX512_UART_IRQn);
    UART_EnableInt(DMX512_UART, (UART_INTEN_RDAIEN_Msk | UART_INTEN_RLSIEN_Msk));
}
```

DMX512_UART_IRQHandler ()函数为 UART 中断处理程序。

```
void DMX512_UART_IRQHandler(void)
{
    uint32_t u32IntSts = DMX512_UART->INTSTS;
    uint8_t u8InChar;

    if (u32IntSts & UART_INTSTS_RLSIF_Msk)
    {
        /* Break Interrupt Flag */
        /* This bit is set to logic "1" whenever the received data input (RX) is held in
           the "spacing state" (logic 0) for longer than a full word transmission time
```

```
           (that is, the total time of "START bit" + data bits + parity + STOP bits). */
        if (DMX512_UART->FIFOSTS & UART_FIFOSTS_BIF_Msk)
        {
            u8BreakFlag = 1;
            u16RecvLen = 0;
            u8ScFlag = 0;
        }

        /* Framing Error Flag */
        /* This bit is set to logic "1" whenever the received character does not have a
           valid "STOP bit" (that is, the STOP bit following the last data bit or PARITY
           bit is detected as logic 0). */
        else if (DMX512_UART->FIFOSTS & UART_FIFOSTS_FEF_Msk)
        {
            u8BreakFlag = 0;
            u16RecvLen = 0;
            u8ScFlag = 0;
        }

        u8InChar = UART_READ(DMX512_UART);
    }

    else if (u32IntSts & UART_INTSTS_RDAINT_Msk)
    {
        /* Get all the input characters */
        while (UART_GET_RX_EMPTY(DMX512_UART) == 0)
        {
            /* Get the character from UART Buffer */
            u8InChar = UART_READ(DMX512_UART);

            if (u8BreakFlag)
            {
                if (u8ScFlag == 0)
                {
                    if (u8InChar == 0x00)
                    {
                        u8ScFlag = 1;
                        g_u8Dmx512Buff[u16RecvLen++] = u8InChar;
                    }
                    else
                        u8BreakFlag = 0;
                }
                else
                {
                    g_u8Dmx512Buff[u16RecvLen++] = u8InChar;
                }
            }

            if (u16RecvLen>512)
                u16RecvLen = 0;
        }

        /*Forces a write of all user-space buffered data for the given output*/
        fflush(stdout);
    }
```

```
    if (DMX512_UART->FIFOSTS & (UART_FIFOSTS_BIF_Msk | UART_FIFOSTS_FEF_Msk |
                                UART_FIFOSTS_PEF_Msk | UART_FIFOSTS_RXOVIF_Msk))
    {
        DMX512_UART->FIFOSTS = (UART_FIFOSTS_BIF_Msk | UART_FIFOSTS_FEF_Msk |
                                UART_FIFOSTS_PEF_Msk | UART_FIFOSTS_RXOVIF_Msk);
    }
}
```

Strip_PSIO_DMX512_Test()函数对 DMX512 数据帧的数据进行处理，通过 PSIO 模块控制 ARGB2 LED。g_u8Dmx512Buff[1]数据帧 1 到 g_u8Dmx512Buff[512]数据帧 512 分别对应第 1-512 颗灯的亮度，g_u8Addr 为 DMX512 控制板的器件地址。

```
void Strip_PSIO_DMX512_Test(void)
{
    uint8_t i, j;
    uint8_t u8addr;

    u8addr = g_u8Addr;

    for (i = 0; i < sizeof(g_u8TransferTbl); i++)
    {
        g_u8TransferTbl[i] = 0;
    }

    for (i = 0; i < sizeof(LED_POS_TBL); i++)
    {
        g_u8TransferTbl[(LED_POS_TBL[i]-1)*3] = g_u8Dmx512Buff[i*3+1+u8addr];
        g_u8TransferTbl[(LED_POS_TBL[i]-1)*3+1] = g_u8Dmx512Buff[i*3+2+u8addr];
        g_u8TransferTbl[(LED_POS_TBL[i]-1)*3+2] = g_u8Dmx512Buff[i*3+3+u8addr];
    }

    for (i = 0; i < STRIP_NUM; i++)
    {
        uint8_t channel = (i == 0) ? 1 : 0;

        for (j = 0; j < LED_NUM; j++)
        {
            PSIO_StripSetRGBColor(channel, g_u8TransferTbl[i * 36+j*3],
                 g_u8TransferTbl[i * 36+j*3+1], g_u8TransferTbl[i * 36+j*3+2]);
        }
    }
}
```

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_V3.02.007
- IDE 版本
  - Keil uVersion V5.41.0.0

3.2 硬件需求

- 电路组件
  - NuMaker-M251KG V1.1
  - NuMaker-Lighting-ARGB
- USB 转 DMX512 Bridge

  用户可以通过拨码开关设置器件地址，如图 3-1 所示，器件地址（0~255）分别对应 DMX512 控制通道数据帧（1~256）。

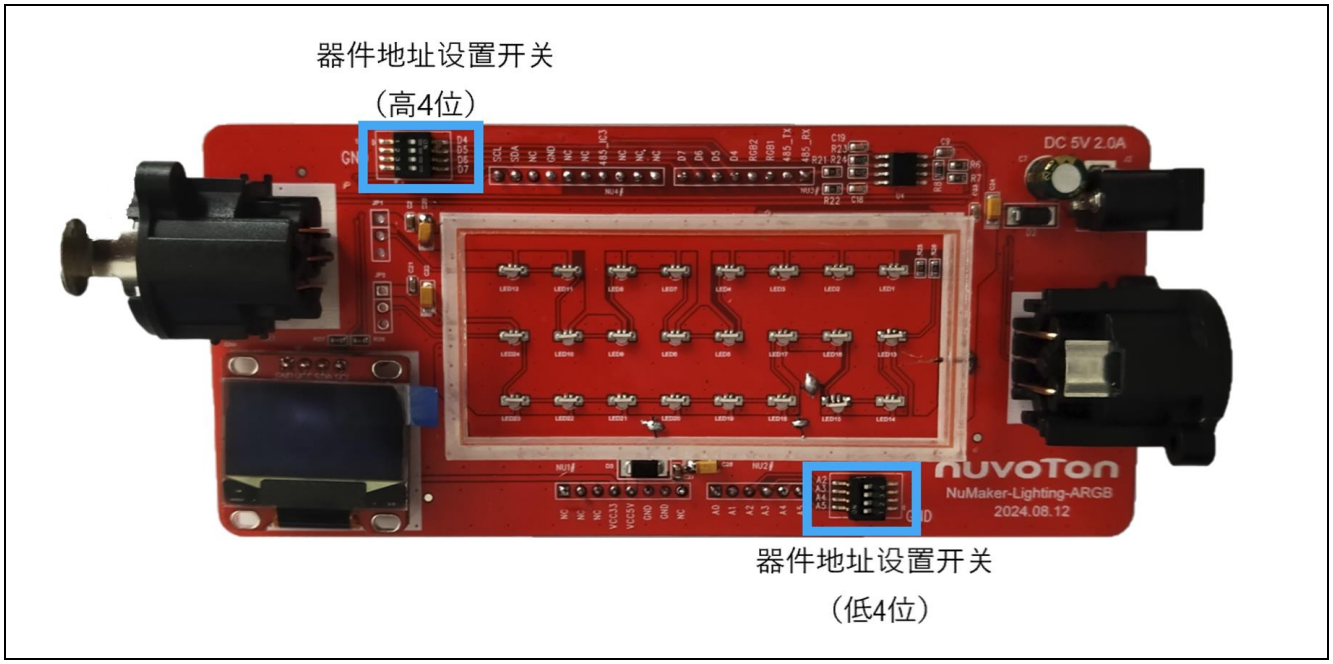

图 3-1 器件地址设置开关

4. 目录信息

目录结构如下图所示。

EC_M251_DMX512_Control_V1.00	
Tool	Tool
DMX512_Controller	Source file of DMX512_Controller
Library	Sample code header and source file
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	
Project	Source file of example code

图 4-1 目录信息

5. 范例程序执行

1. 将 USB 转 DMX512 Bridge 插入电脑 USB 口，卡侬头插入 NuMaker-Lighting-ARGB 板，如图 5-1：

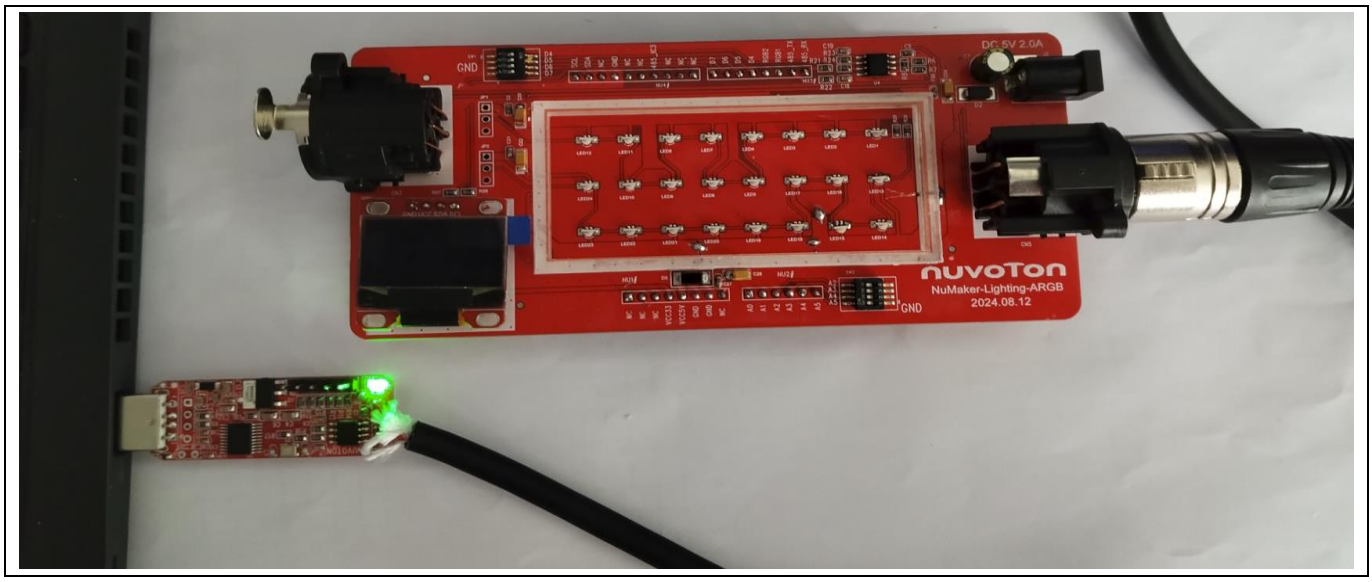

图 5-1 USB 转 DMX512 Bridge 接线图

2. 根据目录信息章节进入 *Tool\DMX512_Controller\bin\Debug* 路径中打开 DMX512 Controller，点击 Search->下拉选择框中选择 PID：5020->点击 Connect。根据 NuMaker-Lighting-ARGB 设置的器件地址，点击 Selection 右边下拉，选择对应的 Board，如图 5-2

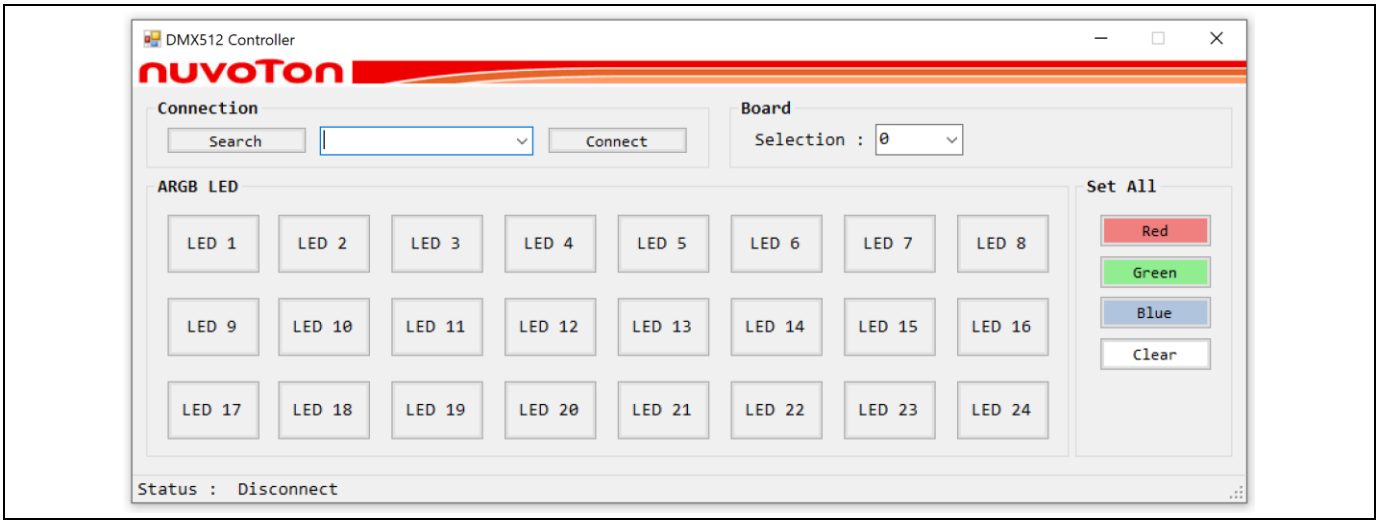

图 5-2 DMX512 Controller 界面图

3. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M251_DMX512_Control.uvprojx*。

4. 进入编译模式界面

   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式

5. 进入仿真模式界面

   - 执行代码
   - 在 DMX512 Controller 软件界面点击鼠标左键选择对应的 LED，调节对应选择的 LED 的 Value 值，控制 LED 亮度。

6. 修订纪录

Date	Revision	Description
2024.10.25	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.