Implementing PUSH Dimming on NDA102

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code implements the PUSH function in the NDA102 project, enabling the PUSH switch to be connected through the DALI port for dimming function.
BSP Version	102_207_NDA102EC1_v03_01_Lib
Hardware	DALI_SLAVE_NDA102EC1 V1.0

1. Overview

This example can enable LED lighting power supplies to recognize PUSH signals through the DALI (Digital Addressable Lighting Interface) bus interface, thereby enabling DALI control gears to have PUSH dimming function. This reduces PUSH specialized interfaces and circuits, reduces costs, and provides another networking dimming method when there is no DALI host (as shown in Figure 1-1).

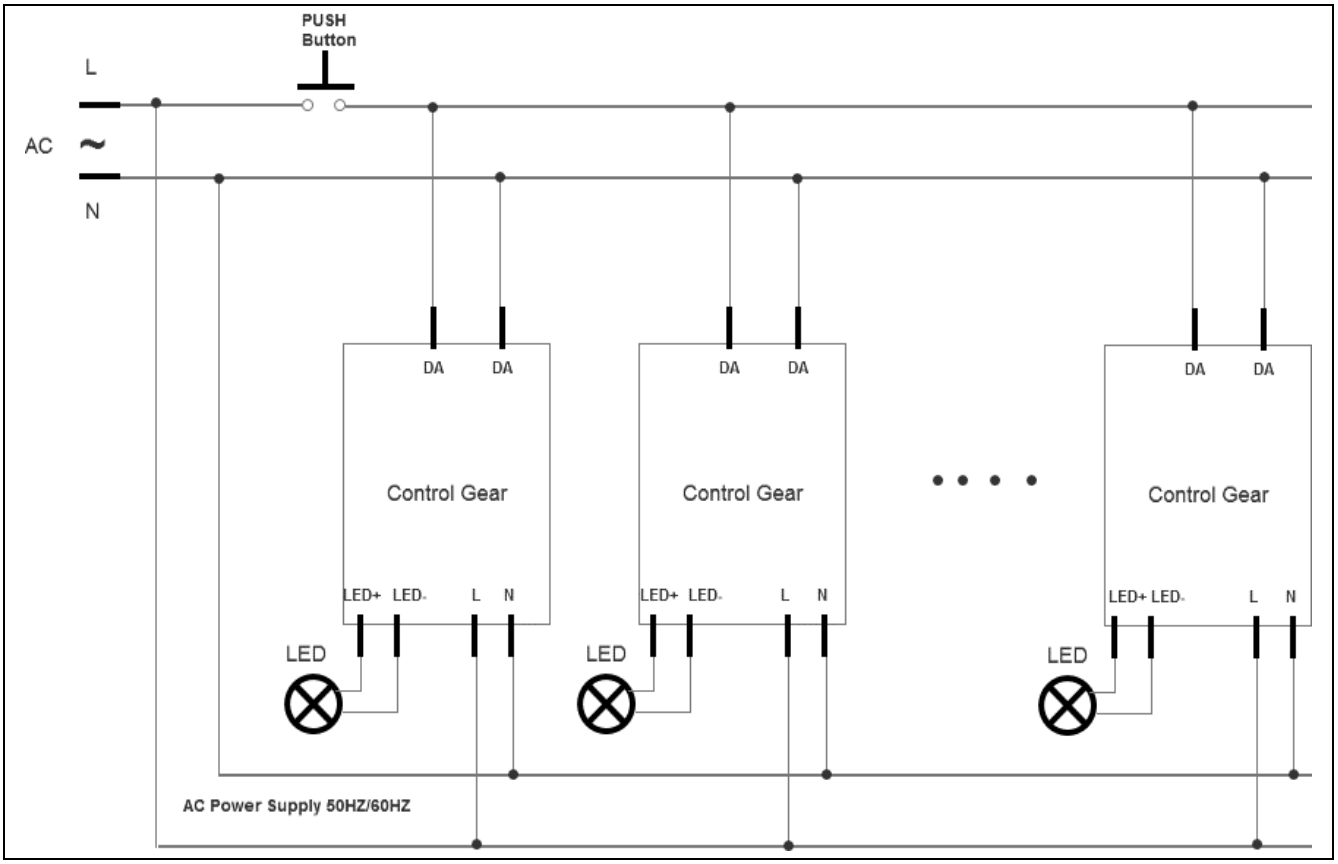

Figure 1-1 PUSH Dim Multiple Connection Diagram

1.1 Principle

DALI signal complies with IEC-62386-101 2014 regulations, which is a 1200bps Manchester code waveform, with a pulse width of 416us and DC 0~22.5V. Due to the IEC-62386-102 2014 requirement that the DALI interface also has a protection function of AC signal blind connection with a withstand voltage of more than 1 minute, the DALI signal terminal hardware can support the input AC signal as a dimming signal. The use of AC signal as a dimming function is commonly referred to as PUSH dimming function in the industry.

The PUSH signal is generally supplied from AC main power directly to the PUSH dimming signal interface. This type of signal has the characteristics of strong high-voltage anti-interference capability due to the direct use of main AC. Moreover, due to the requirement of DALI bus that the bus current of a single device is less than 2mA, PUSH dimming bus also has the characteristic of low power consumption. The PUSH dimming signal is characterized by a

50Hz/60Hz sine wave, which is converted into a 5V square waves of 50Hz/60Hz through the DALI bus circuit. ThNDA102 will determine how many cycles there are based on I/O square wave counting, calculate the duration of the button time based on the number of cycles, and then define the dimming function based on the duration of the button. The block diagram is as follows.

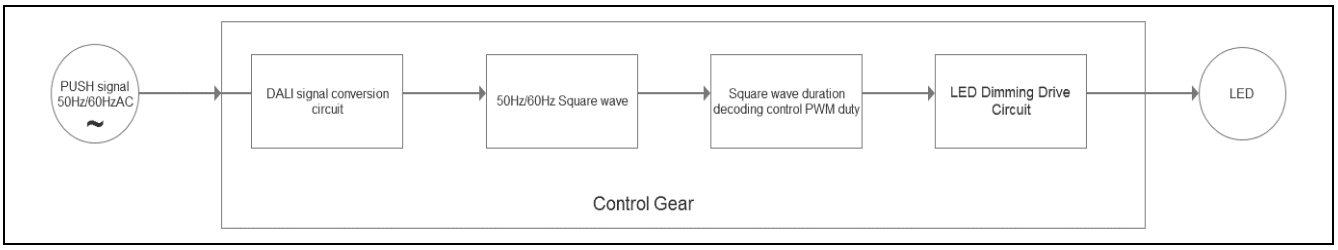

Figure 1-2 PUSH Function Block Diagram

The PUSH button function definition is shown in Table 1-1.

	PUSH Button Description	**Press to lift time Δ *t***	**Function Description**
PUSH Mode	Short press to switch on/off	0.1S <Δ*t*<0.6S	In PUSH mode, short press to turn on or off the lights
	Long press for dimming	Δ*t* >0.6S	In PUSH mode, long press and hold to dimming, dimming stroke (0-100% brightness)
	Long press for reset	Δ*t* >9S	In PUSH mode, press and hold for more than 9S to reset the brightness to half brightness (50%)

Table 1-1 PUSH Button Function Definition

1.2 ICP Download

Connect to the ICP tool and download the firmware to the chip. Download *NDA102_PUSH_DIM_Control.bin* to one evaluation board (hereinafter referred to as 2#), and the chip configuration is shown in Figure 1-3 and Figure 1-4.

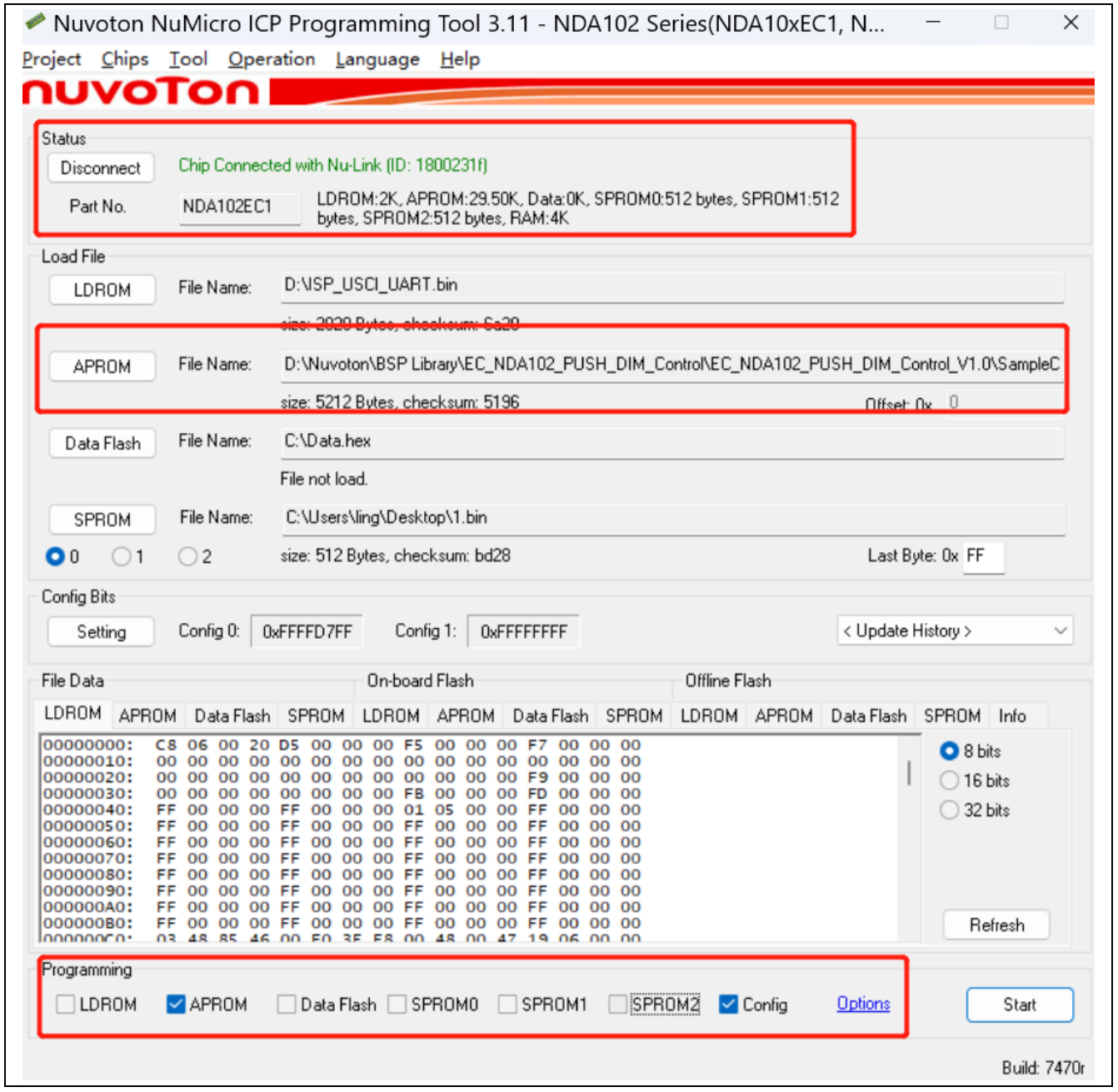

Figure 1-3 ICP Configuration (2#)

Chip settings are shown as follows.

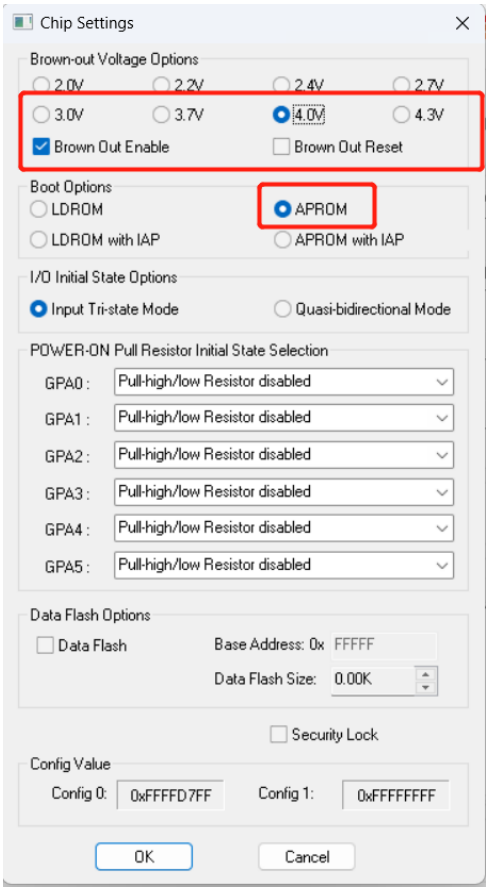

Figure 1-4 Chip Settings (2#)

Connect to the ICP tool and download the firmware to the other chip. Download *NDA102_PUSH_Wave_Out.bin* to the other evlauation board (hereinafter referred to as 1#), and the chip configuration is shown in Figure 1-5 and Figure 1-6.

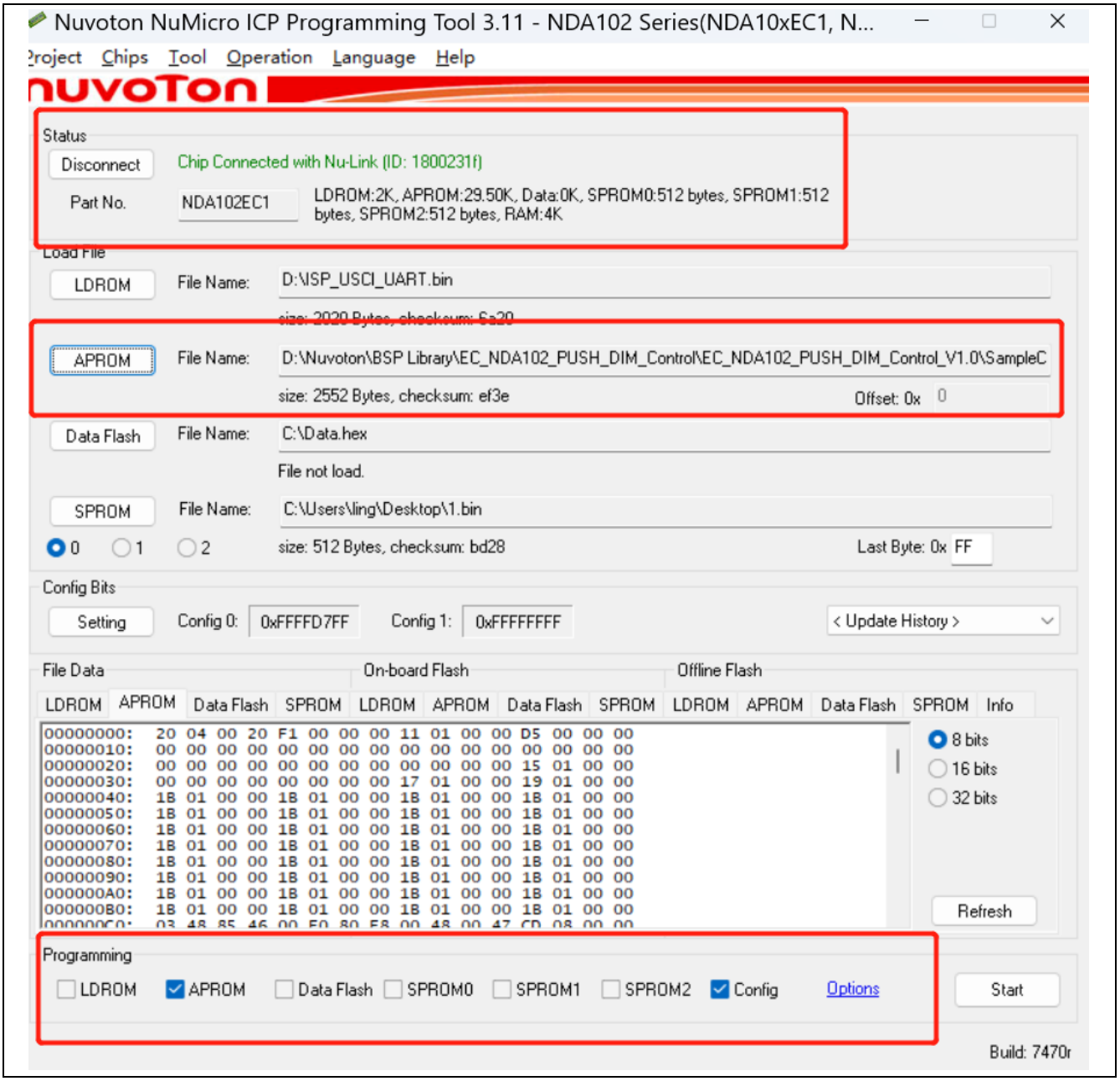

Figure 1-5 ICP Configuration (1#)

Chip settings are shown as follows.

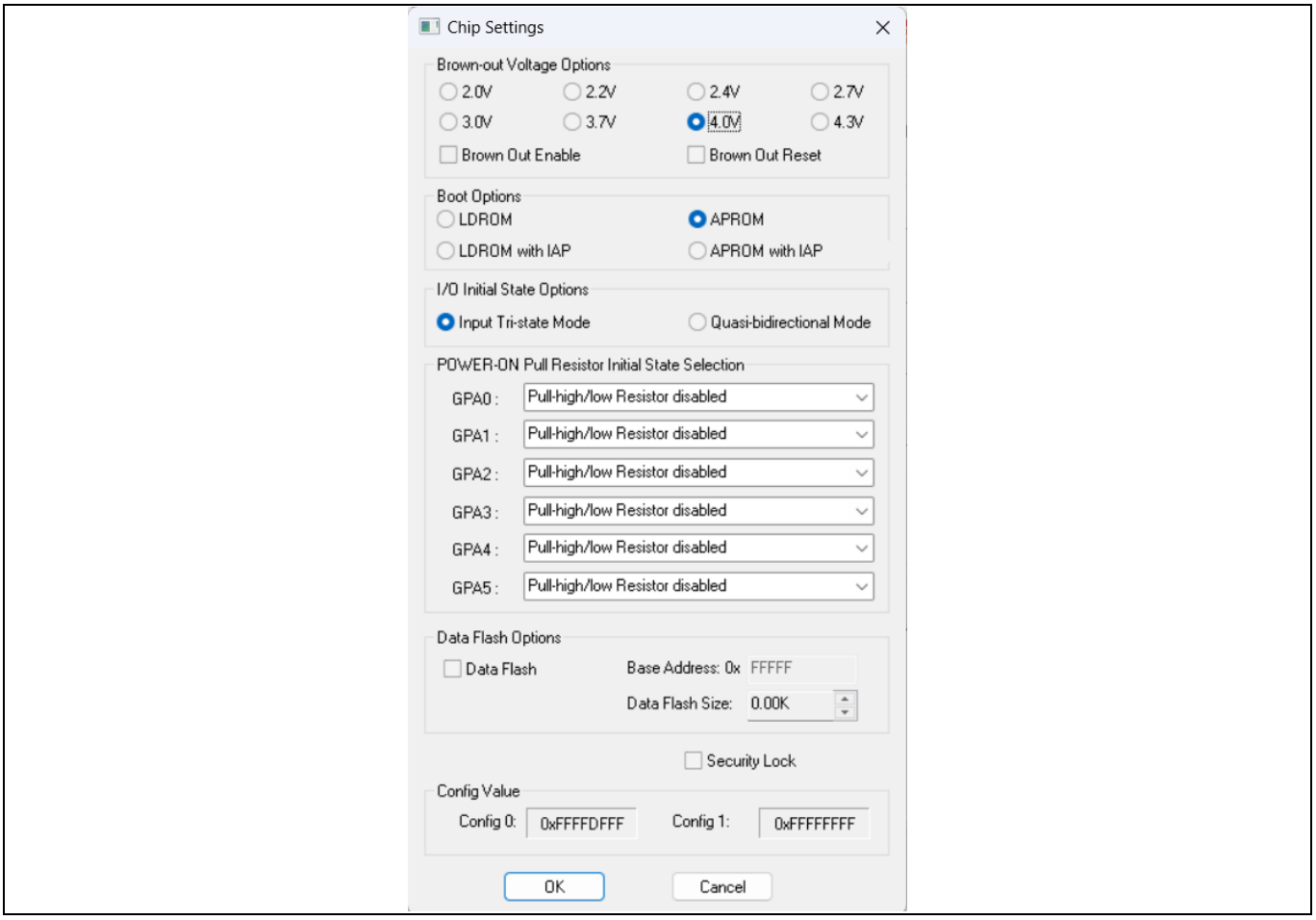

Figure 1-6 Chip Settings (1#)

1.3 Demo Result

When the sample code runs, start testing the PUSH function of DALI circuit. Due to the insufficient withstand voltage capability of the DALI interface of the DALI_SLAVE_NDA-102EC1 V1.0 evaluation board that is unable to be connected to the main AC, the other DALI_SLAVE_NDA102EC1 V1.0 evaluation board is used to output and simulate the PUSH signal input. The schematic diagram is as follows.

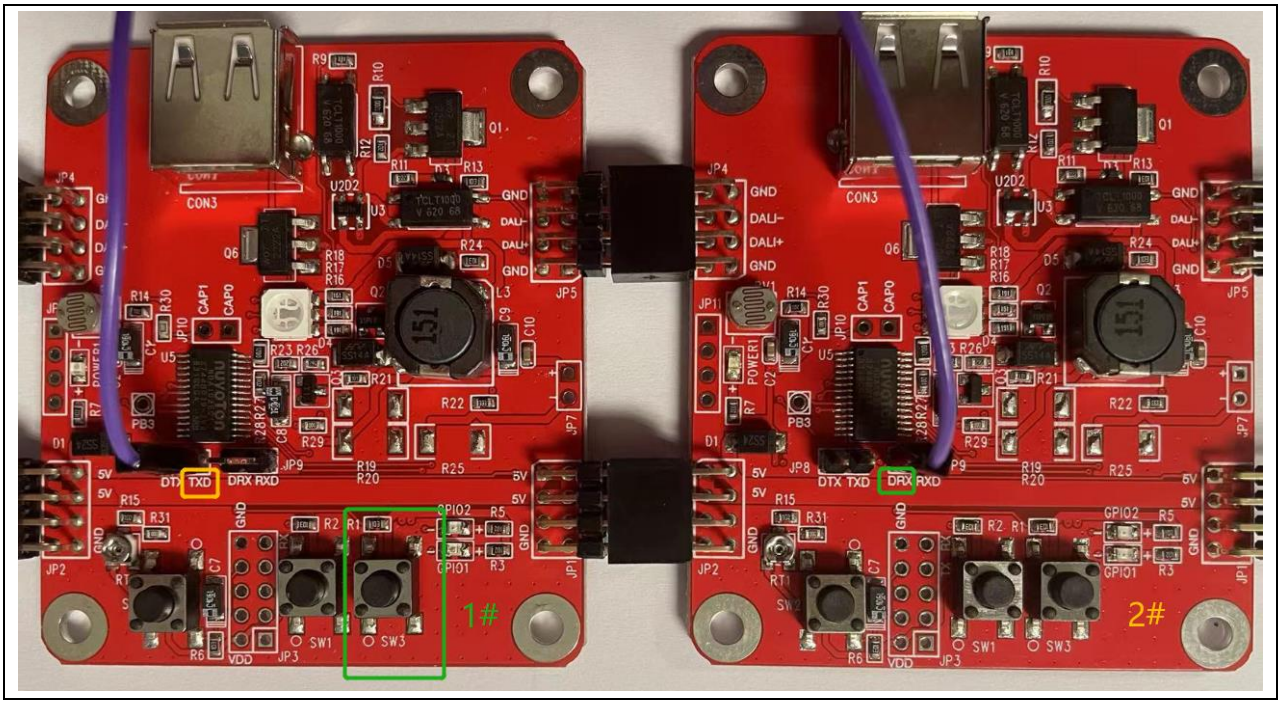

Figure 1-7 Physical Wiring Diagram for Testing PUSH Function

On 1# in the Figure 1-7, the simulated PUSH button uses SW3 (PD.6), and the simulated PUSH waveform output signal uses TXD (PD.3). The TXD (PD.3) of 1# is connected to the DRX (PB.0) of 2# through a DuPont cable.

When it is powered on for the first time, Unit 2 defaults to PUSH mode, and the light is at the maximum brightness.

In PUSH mode, short press to turn on the light.

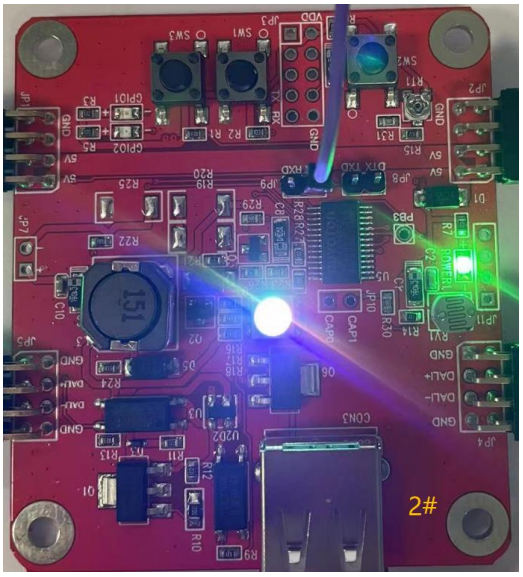

Figure 1-8 Short Press to Turn on the Light

In PUSH mode, short press to turn off the lights.

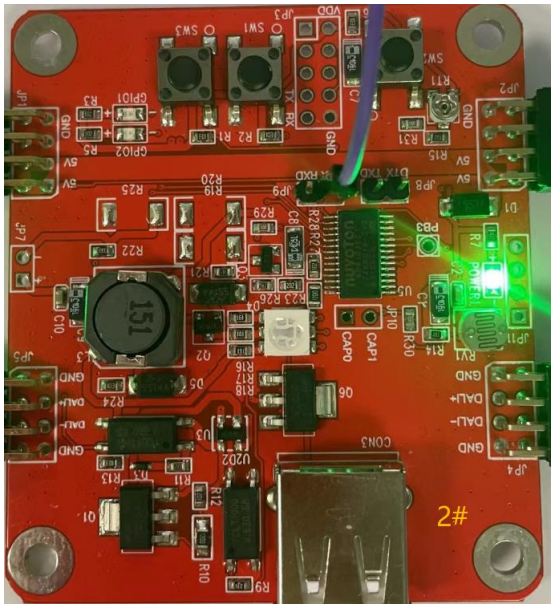

Figure 1-9 Short Press to Turn off the Light

In PUSH mode, long press and hold the PUSH button to dim the light from 0% to 100% brightness, and you can see that the light brightness gradually changed from 0% to 100% for about 4 seconds.

In PUSH mode, press and hold the PUSH button again and adjust the brightness from 100% to 0.1%. You can see that the light brightness gradually changed from 100% to 0.1% for about 4 seconds. Adjust the effect to 0.1% as shown in the Figure 1-10.

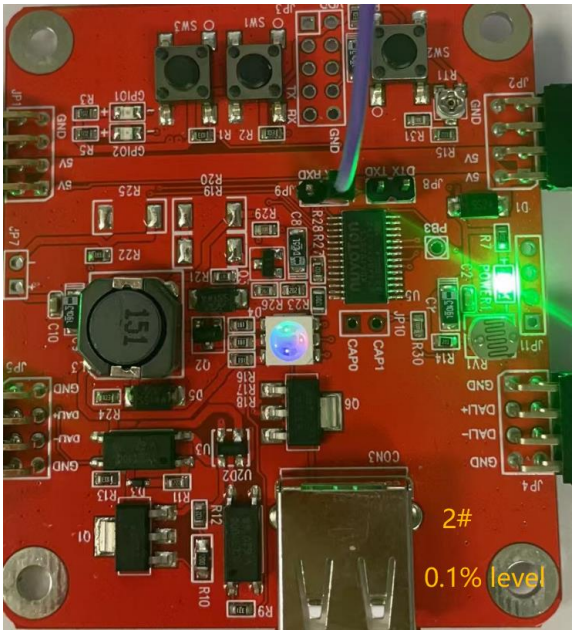

Figure 1-10 Long Press for Dimming the Light to the Darkest Position

In PUSH mode, press and hold for more than 9 seconds to see the LED brightness gradually returned to 50% brightness from 0 to 100%, and reset successfully.

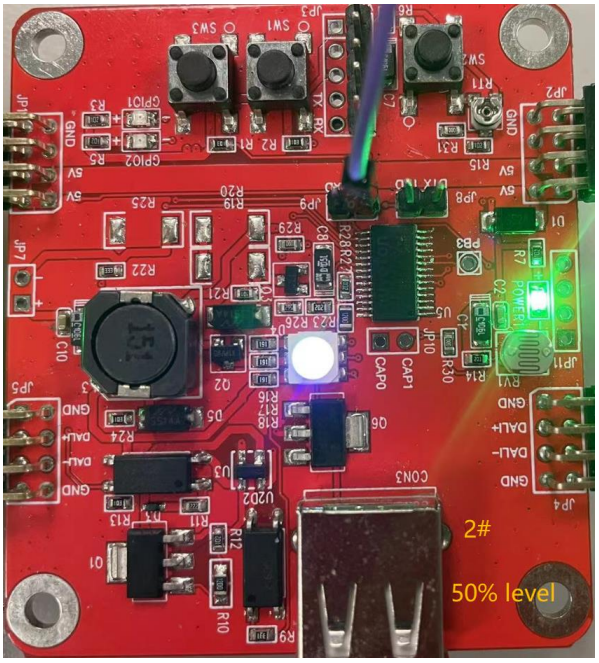

Figure 1-11 Long Press for More Than 9 Seconds to Reset

2. Code Description

In main.c, add PUSH signal identification code to the GPIO interrupt function.

```
void GPABCD_IRQHandler(void)
{
    GPIO_CLR_INT_FLAG(DRX_PORT, DRX_BIT);

    /*------------- ---------------------------------------------------------------*/
    /* PUSH Decode                                                                 */
    /*-------- ---------------------------------------------------------------*/
    /*push BUS IS HIGH,no push signal bus=low level so first signal change is high
level=io is low */
    if(DRX == HIGH_DRX)
    {
        /*Push signal recognition and Identify push mode */
        if((DRX_WAVE_Timer > 15) && (DRX_WAVE_Timer < 30)) /*Do not execute for the first
time*/
        {
            if(DimMode == 1) /*In push mode, in push dimming counter.*/
            {
                DRX_Push_Count++;/*counter*/

                if(65530 <= DRX_Push_Count)
                {
                    DRX_Push_Count = 65530;
                }
            }

          /*In non-push mode · Accumulate 10 consecutive pulses · then switch to PUSH mode.*/
            if(1 != DimMode)
            {
                DRX_toPUSH_Enable = 1;        /*Enable swtich mode*/

                DRX_toPUSH_Count++;

                if(DRX_toPUSH_Count == 1)
                    DRX_toPUSH_Timer = 0;      /*Clear and prepare the timing of 10
pulses*/
                else if(DRX_toPUSH_Count > 10)
                {

                    if((DRX_toPUSH_Timer > 600) && (DRX_toPUSH_Timer < 730))
                    {
                        DimMode = 1;           /*Confirm as PUSH signal*/
                        DRX_toPUSH_Enable = 0; /*Turn off to PUSH enable*/
                        DRX_toPUSH_Count = 0;      /*Clear, prepare for next count*/
                    }
                }
            }
        }

        DRX_WAVE_Timer = 40; /*Set the value to prepare for the next count*/
    }
    else
    {

    }
}
```

In push_DT6.c, the function DRX_PUSH_Count() performs PUSH cycle counting in 1ms tasks, with variable Push_Count calculating the time *Δt* that it takes to press and lift the PUSH button. Determine whether to perform the corresponding dimming action by long or short pressing based on *Δt*.

```
void DRX_PUSH_Count(void)
{
    if(DimMode == 1) /*Push Mode*/
    {

        if(DRX_Push_Count_OLD == DRX_Push_Count) /*Trigger and trigger time interval:
about 20ms . Customer Define. */
        {
            PUSH_Status_Count++;/*Release key counter*/
        }
        else
        {
            Push_Count++;/*Press key count about 20ms cycle*/
            PUSH_Status_Count = 0;
            PUSH_IN_Flag = 1;
        }

        DRX_Push_Count_OLD = DRX_Push_Count;
        Push_Scan(); /*Change level in push mode*/
    }
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - 102_207_NDA102EC1_v03_01_Lib
- IDE version
  - Keil uVersion 5.28

3.2 Hardware Requirements

- Circuit components
  - DALI_SLAVE_NDA102EC1 V1.0 (2pcs)
- Pin Connect
  - Use two sets of DALI_SLAVE_NDA102EC1 V1.0 evaluation boards. Among them, 1# simulates PUSH button and PUSH signal output, and 2# is used for PUSH signal reception and PUSH dimming. The connection is shown in Figure 3-1.

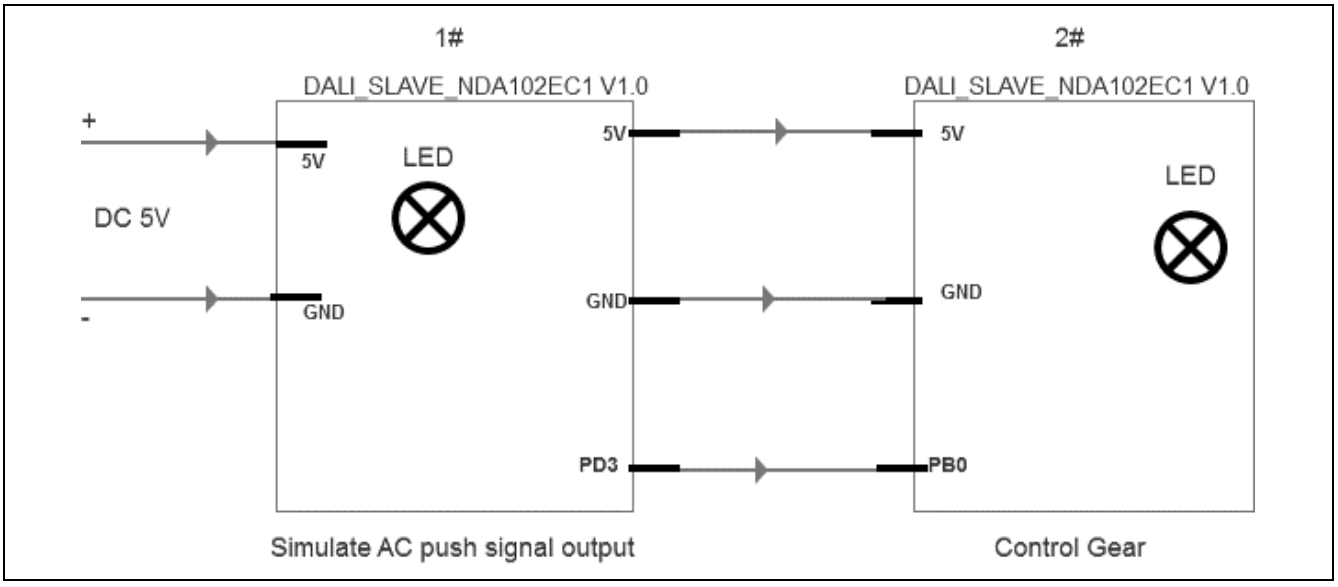

Figure 3-1 Hardware Connection Diagram

4. Directory Information

The directory structure is shown below.

EC_NDA102_PUSH_DIM_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 *NDA102_PUSH_DIM_Control.uvprojx (Corresponding to 2#)* and *NDA102_PUSH_Wave_Out.uvprojx (Corresponding to 1#)*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2023.03.29	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.