Driving KA32183A 9x9 LED Matrix with M483 I²C

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code uses M483 as an example to show how to drive the KA32183A LED extension board by using I²C interface.
BSP Version	M480_Series_BSP_CMSIS_V3.05.005
Hardware	NuMaker-M483KG V1.1 NuMaker-KA32183A_Extension_Board V1.0

1. Overview

This example code uses M483 to drive 81 dots LED matrix with KA32183A driver IC and manage demo sequence from the database to be displayed on the KA32183A LED extension board.

KA32183A is a LED driver IC that can provide a simple design solution for appealing illumination effect in various lighting application. The KA32183A development platform also provides a PC-GUI tool to generate LED patterns and then convert binary file into HEX format and view lighting effect. The PC-GUI tool can also help to complete LED effect easier. For more details, please visit KA32183A Website Page.

1.1 Principle

The following section describes 3 stages to drive KA32183A in the example code.

1. Edit LED patterns by using PC-GUI tool (*LED_Driver_Demo_Software_v1.00.exe*), as described in section 1.1.1. Table 1-1 lists related files in LED_DRV_DEMO_SW (PC-GUI)_v1.00 folder.

Related Folders and Files		Description
Folder	Pattern	To save LED pattern file(*.ptf) created and its command file(*.cmf) which is auto-generated.
	Playlist	To save playlist file(*.pls) of selected LED patterns.
	Setup	To save matrix setup file(*.mts) used for new pattern design
File	CyUSB.dll	Essential file that cannot be removed, renamed and relocated.
	Fx2hid.iic	Essential file that cannot be removed, renamed and relocated.
	LED_Driver_Demo_Software	Executeble file (*.exe) of software.

Table 1-1 Related Files and Folders in the LED_DRV_DEMO_SW (PC-GUI)_v1.00 Folder

2. Generate a LED pattern header file by using PC-GUI tool (*KA3218xPatternConverter_v1.00.exe*). This is called demo_pattern.h by default, as described in section 1.1.2.

3. Merge the LED pattern header file into software project, as described in section 1.1.3.

After building and loading code into MCU, the effect can be applied to display on the KA32183A LED extension board.

1.1.1 Creating and Editing LED Patterns by LED_Driver_Demo_Software Tool

Follow the steps below to create and edit customized LED patterns.

1. Open LED_Driver_Demo_Software.exe, as shown in Figure 1-1.

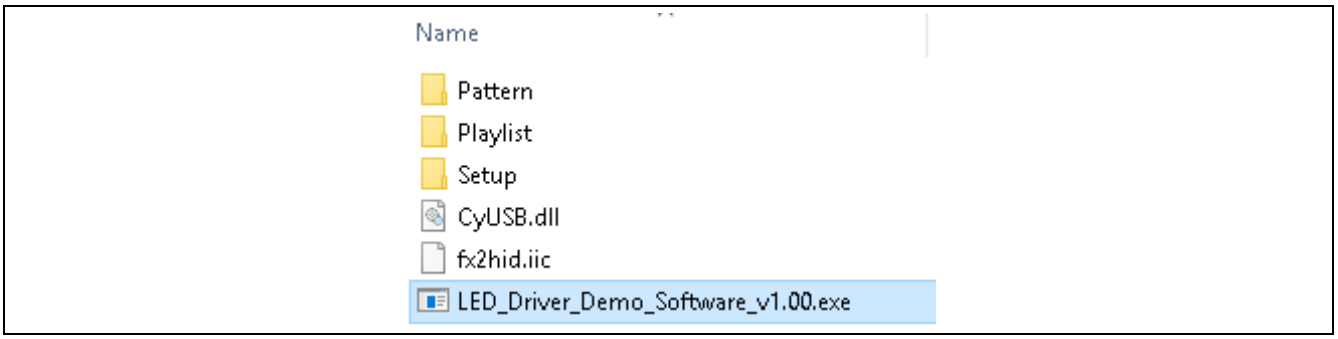

Figure 1-1 Open EXE Application

2. Click **Tools** in the toolbar and select **Pattern Editor**, as shown in Figure 1-2.

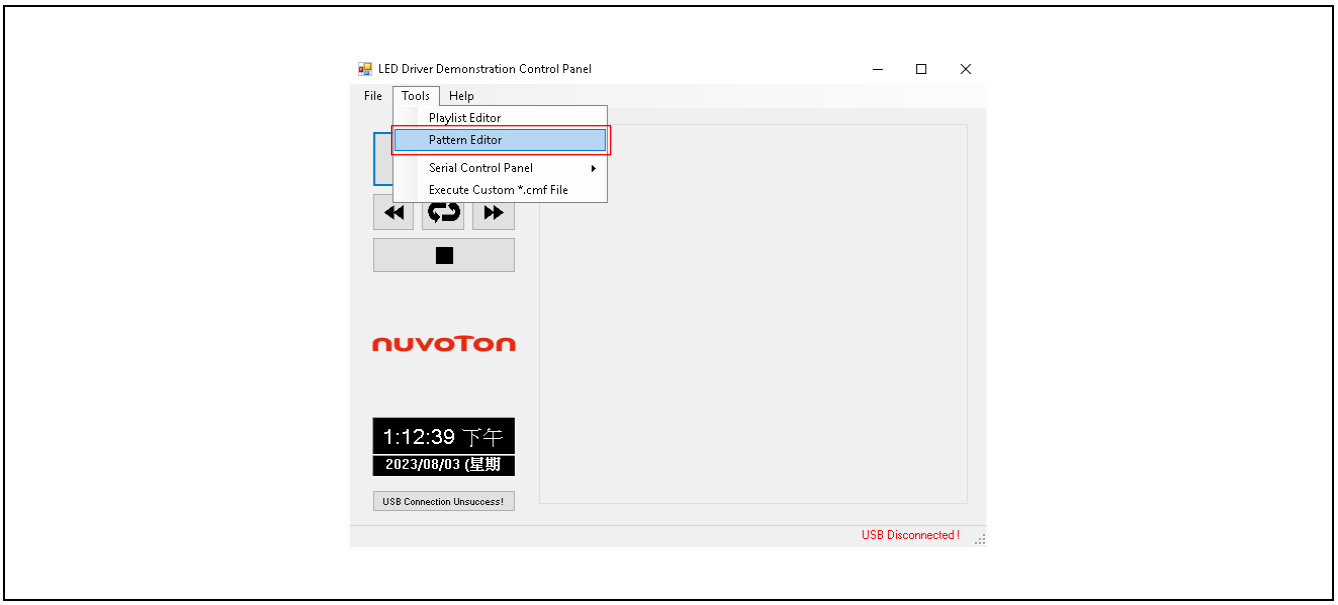

Figure 1-2 Start a New Pattern Editor Project in Main Screen

3. Use matrix setup file (*32183A_default.mts*) for KA32183A to establish default settings when you open the tool for the first time. The default setup file is saved on Setup folder, as shown in Figure 1-3.

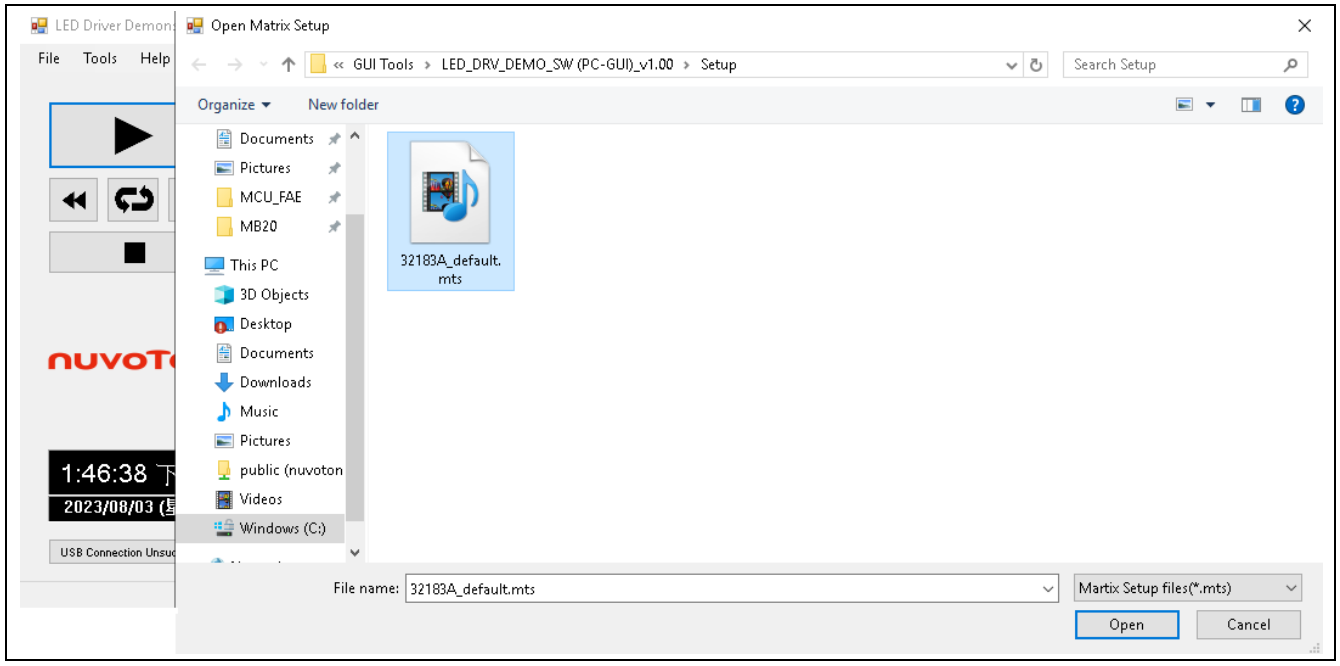

Figure 1-3 Use Default Matrix Setup File to Establish Default Settings

4. In pattern editor screen, click “+” button to add multiple frames of LED pattern and configure hold time for each frame. User can click dot matrix for creating LED pattern in the red box and configure the parameters of each LED(e.g. Brightness, Firefly and Fad-in/out, and so on), as shown in Figure 1-4.

Note: This matrix consists of 4 matrix (9x9) and can support four KA32183A chips. The matrix for KA32183A in the LED_Driver_Demo_Software tool and the corresponding zone on the KA32183A extension board is labelled with yellow box. Each dot in matrix represents one LED with individual settings. The “0” on the dot represents the setting of brightness is “0”.

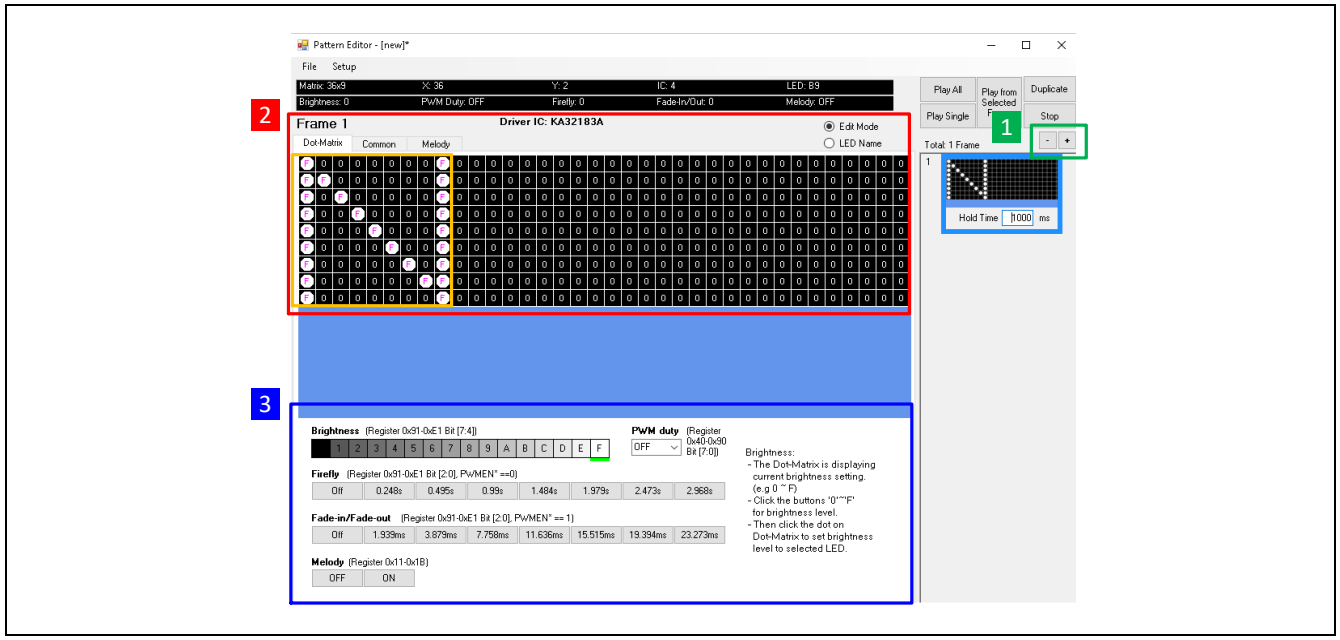

Figure 1-4 Editing the LED Patterns

5. When editing is completed, save as pattern file (*.cmf and *.ptf) in Pattern folder.

1.1.2 Converting Pattern File to Header File by Using KA3218xPatternConverter.exe

Follow the steps below and shown in Figure 1-5 to convert the pattern file to a header file.

1. Choose the **KA32183A** tab in *KA3218xPatternConverter.exe*
2. Select the folder to import the pattern file.
3. Import the pattern file (*.cmf) into *KA3218xPatternConverter.exe*.
4. Click **Convert** to generate the LED pattern header file, which is called demo_pattern.h by default.

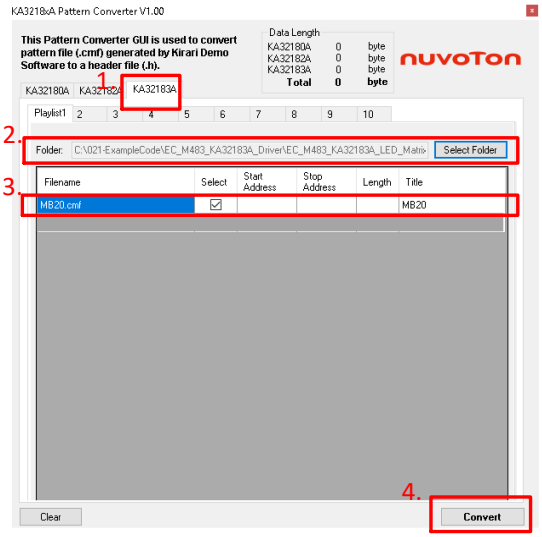

Figure 1-5 Generate LED Pattern Header File

1.1.3 Merging LED Pattern Files into Project

Replace *demo_pattern.h* under inc directory with the one generated from KA3218xPatternConverter. The file path is shown in Figure 1-6.

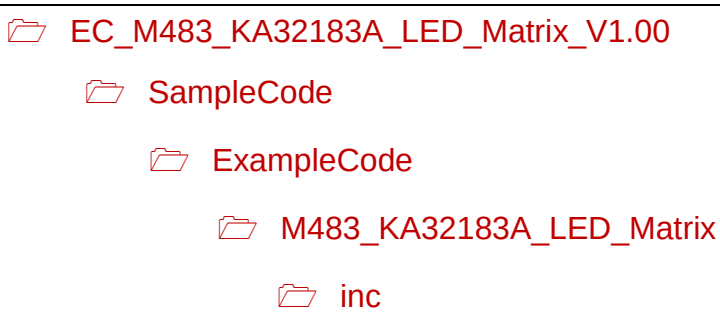

Figure 1-6 File Path of demo_pattern.h

After building and loading code into MCU, the effect can be applied to display on the KA32183A LED extension board.

1.2 Demo Result

In this example code, “NTC♡” are displayed in sequence on the KA32183A LED extension board, as shown in the Figure 1-7.

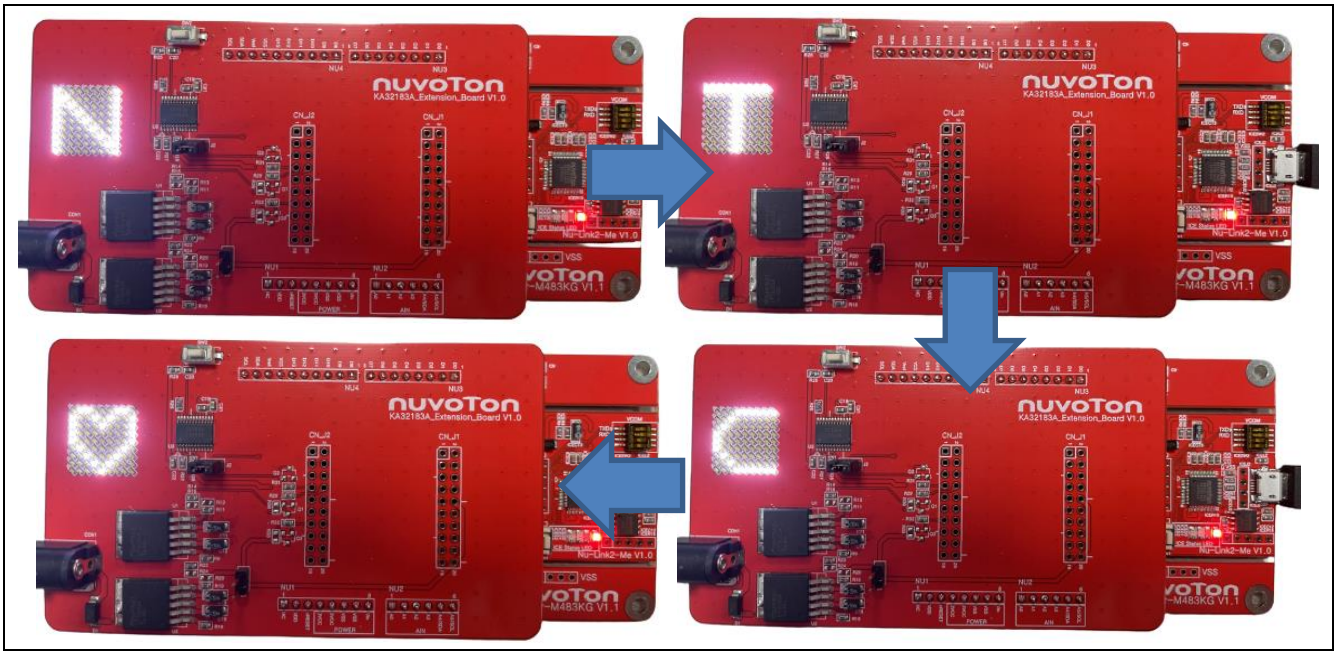

Figure 1-7 Demo Result

2. Code Description

The code located in *main.c* includes five parts:

- SYS_Init(): To initialize system and peripheral clock, and multi-function I/O.
- Timer0_Init(): To configure timer interrupt for 1ms.
- I2C1_Init(): To initialize I²C module and configure I²C clock as 1 MHz. Due to KA32183A SLAVESEL pin is tied to 'LOW" by hardware design, I²C slave address is 0b1011100x.
- LED_DeviceDetect():Auto-detection of KA32183A by using the unique I²C slave address. The code located in *led_app.c.*
- LED_APP_MainCtrl(): The code located in *led_app.c* includes two parts:
  - To manage state machine for demo control and select playlist being store in the binary pattern file.
  - Control KA32183A by using the I²C protocol.

```
int32_t main(void)
{

    …

    /* Init System, IP clock and multi-function I/O. */
    SYS_Init();

    …
    /* System tick */
    TIMER0_Init();
    /* Using I2C to control KA32183A */
    I2C1_Init();

    /* Detect device connected base on unique Slave Address of device */
    LED_DeviceDetect();

    while (1)
    {
         /* Polling LED_APP_MainCtrl() every 20 ms */
        if (u8MainLoopFlg)
        {
            u8MainLoopFlg = 0;
            LED_APP_MainCtrl();
        }
    }
}
```

After initializing the system, LED_APP_DemoCtrl() handling of KA32183A control with I²C protocol. The code is located in *led_app.c* includes three parts:

- Get the playlist address and pattern information.
- Configure KA32183A register.
- Control LED pattern sequence.

```
void LED_APP_DemoCtrl(UCHAR u8PlaylistNo, UCHAR u8PatternNo)
{

    …

    switch (u8LED_APP_DemoModeState)
    {
        /* Get the playlist address and pattern information. */
        case (STATE_START):
            LED_GetPlaylistAddr(u8PlaylistNo);
            LED_GetPatternInfo(u8PatternNo);

            u32Cyc = 0;
            u32TempAddr_demo = Demo_PatternInfo.u32TempAddress - 1;
            u32Length = Demo_PatternInfo.u32Length;
            break;

        /* Configure KA32183A register. */
        case (STATE_SINGLEWRITE):
            LED_SingleWrite();
            break;

        /* Control LED pattern sequence. */
        case (STATE_INCLOAD):
            do
            {
                LED_IncLoad();
                u32TempAddr_demo++;
                u8LED_APP_DemoModeState = *(MemoryPtr + (u32TempAddr_demo));
                u32Cyc++;
                u32TempAddr_demo++;
                u16Num = *(MemoryPtr + (u32TempAddr_demo));
                u32Cyc++;
            } while (u8LED_APP_DemoModeState == STATE_INCLOAD);

        case (STATE_INCWRITE):
            LED_IncWrite();
            break;

        /* Wait time between each frame of LED pattern display */
        case (STATE_WAIT):
            if (u8WaitFlgSet == 0)
            {
                LED_Wait();
                u8WaitFlgSet = 1;
            }
```

```
            if (u8LEDWaitLoopflg == 1)
            {
                u8LEDWaitLoopflg = 0;
                u8WaitFlgSet = 0;

                if (u32Cyc >= u32Length)
                {
                    u8LED_APP_DemoModeState = STATE_END;
                }
            };

            break;

        /* End of demo sequence */
        case (STATE_END):
            u8Start = 0;
            u8WaitFlgSet = 0;
            break;

        default:
            u8error_status = ERR_DEMOMODESTATE;
            u8Start = 2;        /* Assign for unknown state */
            break;
    }

    …
}
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M480_Series_BSP_CMSIS_V3.05.005
- IDE version
  - Keil uVersion 5.27

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M483KG V1.1
  - NuMaker-KA32183A_Extension_Board V1.0
- Pin Connect
  - Mount NuMaker-KA32183A LED extension board on NuMaker-M483KG. The pin connections between the M480 and the KA32183A is detailed below.

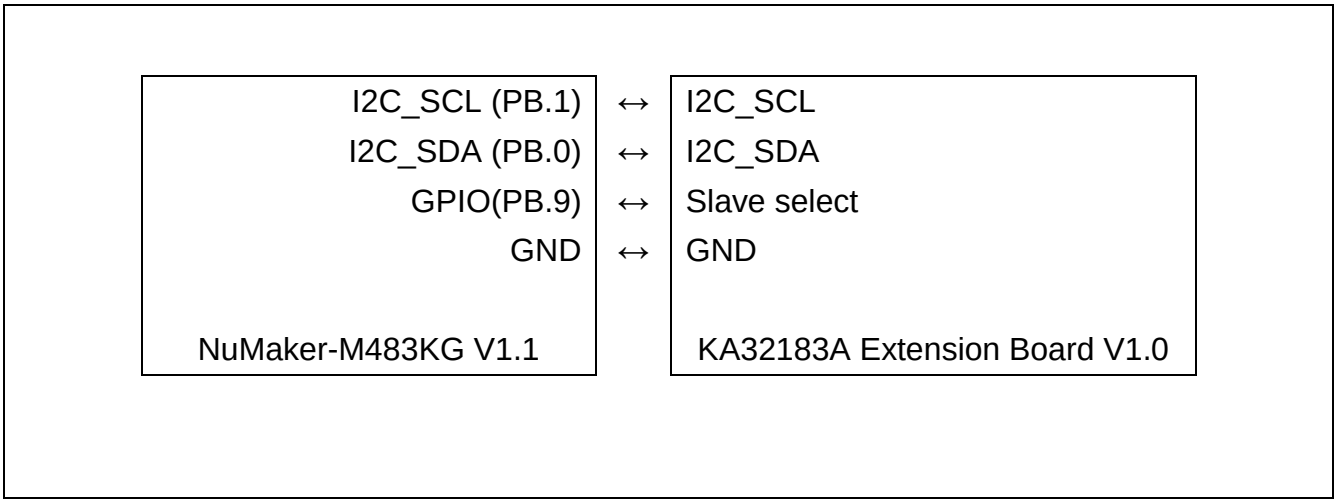

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M483_KA32183A_LED_Matrix_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
KA3218xA Reference GUI Tools	PC-GUI tools for creating customized LED pattern
GUI Tools	
LED_DRV_DEMO_SW (PC-GUI)_v1.00	LED_Driver_Demo_Software.exe for creating LED pattern
Pattern Converter GUI_v1.00	KA3218xPatternConverter.exe for generating LED pattern header file

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 *M483_KA32183A_LED_Matrix.uvproj*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2023.07.31	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.