

在 TFT 螢幕顯示動畫並疊加字元

NuMicro® 32 位元系列微控制器範例程式碼介紹

文件資訊

應用簡述	使用 QSPI Quad 模式搭配 PDMA 從 FLASH 中讀取動畫資料，然後疊加字元資料，最後利用 SPI 介面驅動 TFT 控制器進行顯示。
BSP 版本	M3331_Series_BSP_CMSIS_V3.00.002
開發平台	NuMaker-M3334KI V2.0 & NuFun-M3334KI V2.0

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

本範例程式碼利用 QSPI Quad 模式搭配 PDMA，依次從 FLASH 中讀取由動畫生成的多張圖片資料，然後再分別疊加字元資料，最後利用 SPI 搭配 PDMA，依次把資料發送到 TFT 控制器進行顯示。

1.1 原理

320x240 TFT 以 RGB565 模式顯示，一個像素需要 16 位元資料，全螢幕顯示需要 153,600 (320 x 240 x 2) 個位元組資料。開發板上的 FLASH (W25Q32) 儲存容量為 4 Mbytes，最多可儲存 27 張圖片資料。執行「Tool」資料夾中的「*ImagesToBin_Uncompress.exe*」（該 Tool 僅供學習參考），將當前路徑下 27 張解析度為 320 x 240，位深度為 16 的 bmp 格式圖片轉換為 RGB565 格式資料（高位在前），並依序合併最終在當前路徑下生成一個「*SPI_Flash.bin*」檔案。每張圖片的掃描方式如圖 1-1 所示。



圖 1-1 圖片掃描示意圖

利用 ICP Programming Tool，將 Bin 檔燒錄到 NuFun-M3334KI V2.0 板上的 FLASH 晶片。操作如下：

1. 連按兩下「Tool」資料夾中的「*NuMicro ICP Programming Tool 3.22.7959a.exe*」進行工具安裝（安裝後在安裝目錄下才會有「*SPI_Flash.ini*」檔案）。
2. 在安裝目錄下找到「*SPI_Flash.ini*」並打開，參考圖 1-2 進行修改（待燒錄 Bin 檔路徑經由圖示步驟 2 設定；燒錄完成後，經由步驟 3 設定從 FLASH 中讀出用於校驗的 Bin 檔路徑）。

```
[Version]
Nu_LinkVersion=V1.00
[M031]
ProgramAlgorithm=FlashAlgorithm_M031_SPI.a
ProgramOffset=0x0
ProgramFile=C:\Program Files (x86)\Nuvoton Tools\ICPTool\SPI_Flash.bin
OutputFile=C:\Program Files (x86)\Nuvoton Tools\ICPTool\SPI_Flash_output.bin
[M3331]
ProgramAlgorithm=FlashAlgorithm_M3331_SPI.a
ProgramOffset=0x0
ProgramFile=C:\Program Files (x86)\Nuvoton Tools\ICPTool\SPI_Flash.bin
OutputFile=C:\Program Files (x86)\Nuvoton Tools\ICPTool\SPI_Flash_output.bin
```

1. Set the starting address for programming
2. Set the source file path
3. Set the path for the verification file

圖 1-2 燒錄參數設定

3. 用 USB 線連接 EVB 與 PC，並參考圖 1-3 進行操作（在進入步驟 1 過程中，若提示韌體與驅動版本不一致，則根據提示進行韌體更新）。注意在「**Programming**」選項中因為沒有 Internal Flash 需要燒錄，因此必須勾選「**Chip Setting**」。

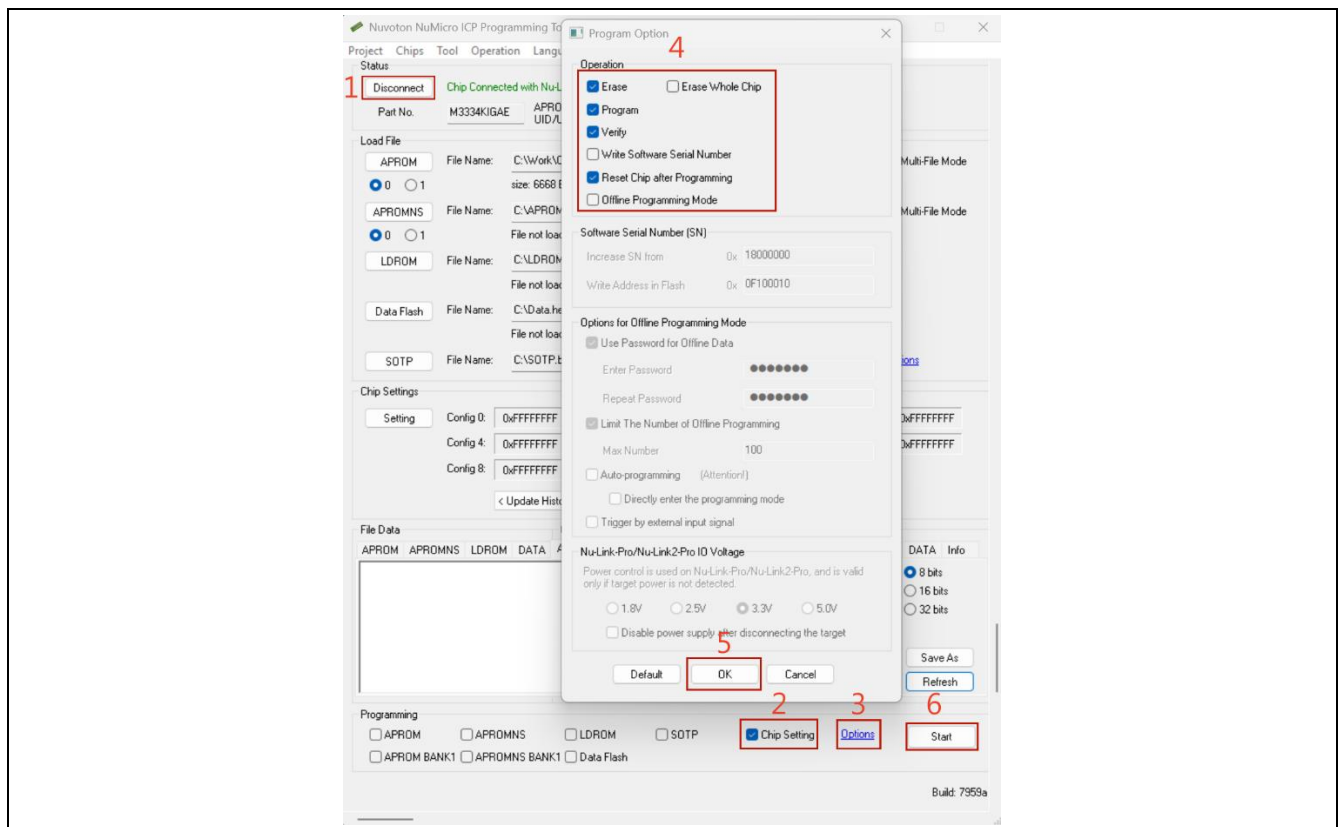


圖 1-3 燒錄選項設定

MCU 利用 QSPI 介面與外部 FLASH 進行通信。為了提高資料傳輸速率，啟用了 Quad 模式，並使用了 RX PDMA，資料寬度為 32 Bit。MCU 利用 SPI 介面與 TFT 控制器相連，在此使用了 TX PDMA 和雙傳輸緩衝區，資料寬度為 16 Bit。

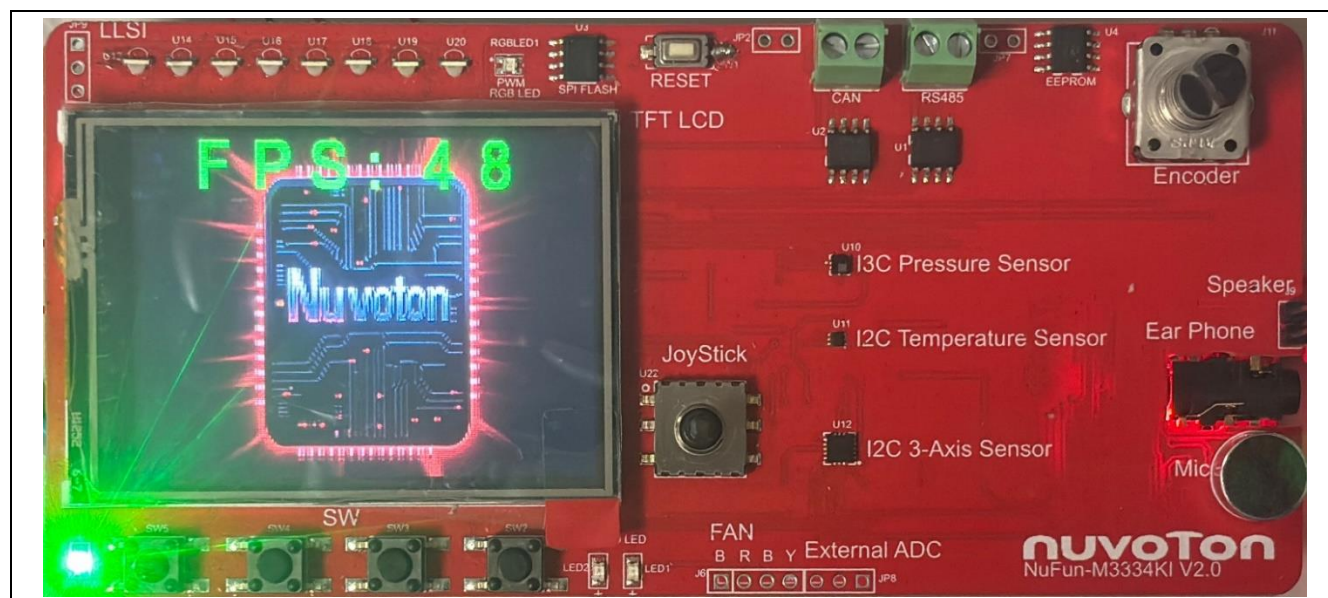
MCU 上電後先初始化相關介面，然後依序讀取外部 FLASH 中每張圖片資料，並分別疊加字元資料，最後依次發送到 TFT 控制器進行動畫顯示。

此示例的整體軟體流程如下：

1. 上電後初始化 MCU 系統時鐘、GPIO、UART、Timer0、QSPI、SPI、以及外部 LCD 等。
2. MCU 將外部 FLASH 配置為 QSPI Quad 模式，並啟用接收端 PDMA 以高效讀取圖像資料。
3. 動畫幀資料按順序從外部 FLASH 中讀取並傳入顯示緩衝區。
4. 諸如即時幀率之類的字元資料會逐行與圖像資料在顯示緩衝區中進行合併。
5. 合併後的幀資料利用 SPI 介面使用 TX PDMA 並結合雙緩衝機制傳輸至 TFT 控制器。
6. Timer0 用於測量顯示幕的刷新頻率，該頻率的計算結果會通過 UART 介面進行輸出。
7. 上述過程 3~6 會針對每一幀重複進行，從而在 TFT 顯示幕上生成連續的動畫效果。

1.2 執行結果

本範例程式碼執行編譯和燒錄後，會顯示動畫並疊加字元，其中一個畫面如圖 1-4 所示。



FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.
 FPS : 48.

2. 程式碼介紹

以下將介紹此示例的關鍵代碼部分。除非另有說明，所有代碼均位於 *main.c* 檔案中。

以下代碼定義了此示例中所使用的巨集。

```
#define PDMA_FOR_LCD_CH0    0
#define PDMA_FOR_FLASH_CH1 1
#define STRING_WIDTH      (5*8) /* String width */
#define STRING_HEIGHT     46    /* String height */
#define LCD_WIDTH         320
#define LCD_HEIGHT        240
#define PIC_WIDTH         320    /* Picture width */
#define PIC_HEIGHT        240    /* Picture height */
#define LV_HOR_RES_MAX    320
#define LV_VER_RES_MAX    240
#define FLASH_QSPI_CLK_FREQ 60000000 /* QSPI clock speed */
#define LCD_SPI_CLK_FREQ   60000000  /* SPI clock speed */
#define COLOR_WHITE       0xFFFF
#define COLOR_BLACK       0x0000
#define COLOR_BLUE        0x001F
#define COLOR_BLUE2       0x051F
#define COLOR_RED          0xF800
#define COLOR_MAGENTA     0xF81F
#define COLOR_GREEN       0x07E0
#define COLOR_CYAN        0x7FFF
#define COLOR_YELLOW      0xFFE0
#define LCD_RST           PD14
#define LCD_DC            PH10
#define LCD_EN            PH11
#define LCD_DC_CLR()      PH10 = 0
#define LCD_DC_SET()      PH10 = 1
#define QSPI_FLASH_PORT   QSPI0
#define SPI_LCD_PORT      SPI1
```

系統初始化包括 UART、Timer0、QSPI、SPI 和 LCD 的配置。在初始化過程中，受保護的寄存器會被解鎖，系統時鐘和 I/O 會被配置，LCD 會以預設顏色清零。Timer0 被配置用於測量 LCD 的刷新率。設置 PDMA 中斷優先級為最高，確保資料高效傳輸。

```
uint8_t i;
uint32_t Flash_Addr;
uint16_t Rate = 0;

/* Unlock protected registers */
SYS_UnlockReg();

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

/* Configure PD14[LCD_RST] PH10[LCD_DC] PH11[LCD_EN] as push-pull output */
GPIO_SetMode(PD, BIT14, GPIO_MODE_OUTPUT);
```

```

GPIO_SetMode(PH, BIT10 | BIT11, GPIO_MODE_OUTPUT);

/* Configure UART0: 115200, 8-bit word, no parity bit, 1 stop bit. */
UART_Open(UART0, 115200);

/* Timer0 initialization. The main purpose is to measure the LCD refresh rate. */
Timer0_Init();

/* QSPI0 initialization, used for driving the Flash */
QSPI_FLASH_Init();

/* SPI1 initialization, used for driving the LCD */
SPI_LCD_Init();

/* LCD initialization */
LCD_Init();

/* Set full-screen display with green color. */
LCD_Color_Set(0, 0, (LCD_WIDTH - 1), (LCD_HEIGHT - 1), COLOR_GREEN);

/* Set interrupt priority */
NVIC_SetPriority(PDMA0_IRQn, 0);
NVIC_SetPriority(TMR0_IRQn, 1);

/* Delay 200ms */
CLK_SysTickLongDelay(200000);

```

主迴圈負責顯示動畫並疊加字元資料。

對於每一幀：

重置刷新計時器計數器。

從外部 FLASH 中讀取圖像資料，並將其與字元資料合併。

利用 SPI 結合 PDMA 將合併後的資料傳送至 LCD。

計算刷新頻率，並透過 UART 匯出顯示。

```

/* Set the starting address of Flash */
Flash_Addr = 0;

for (i = 0; i < 27; i++)
{
    /* Reset the time counter */
    u8TimerCnt = 0;

    /* Display the image and superimpose the characters */
    Flash_Addr = LCD_Display_Picture_And_String(Flash_Addr,
                                                0, 0,
                                                PIC_WIDTH, PIC_HEIGHT,
                                                COLOR_BLACK, Rate,
                                                40, 0,

```

```

                                STRING_WIDTH, STRING_HEIGHT,
                                COLOR_RED);

    /* Calculate the refresh rate */
    Rate = (uint16_t)(10000.0/(u8TimerCnt*1));

    /* Print the refresh rate */
    printf("FPS = %d.\n", Rate);
}

```

函式 LCD_Display_Picture_And_String() 從外部 FLASH 中指定地址讀取圖像資料，將其顯示在預定義的 LCD 區域內，並疊加諸如幀率值之類的文本字元。

```

/* Read the image data from a specific address in the flash memory and display it in a
specific area of the image, while also overlaying numbers on the image. */
uint32_t LCD_Display_Picture_And_String(uint32_t Addr, /* Flash address */
    uint16_t Pic_x, uint16_t Pic_y, /* picture display coordinates */
    uint16_t Pic_Width, uint16_t Pic_Height, /* picture size */
    uint16_t String_Backcolor, uint16_t Rate, /* string background color, display numerical */
    uint16_t String_x, uint16_t String_y, /* string display coordinates */
    uint16_t String_Width, uint16_t String_Height, /* string size */
    uint16_t String_Color) /* string color */
{
    uint32_t i;

    /* Initialize the pointer */
    uint16_t *Display_Buffer = LCD_Display_Buffer0;
    uint8_t Y_Number = 0;

    if (Addr == 0)
    {
        QSPI_Close(QSPI0);

        while ((QSPI0->STATUS & QSPI_STATUS_SPIENSTS_Msk) != 0)
        {}

        /* Disable MCU quad I/O mode. */
        QSPI_DISABLE_QUAD_MODE(QSPI0);

        /* Set the QSPI data length to 8 */
        QSPI_SET_DATA_WIDTH(QSPI0, 8);

        QSPI0->CTL |= (QSPI_CTL_SPIEN_Msk);

        /* Enable Flash quad mode */
        SpiFlash_Enable_QSPI_Quad();

        /* Set the starting address for Flash reading */
        SpiFlash_SetReadAddress(Addr);

        QSPI_Close(QSPI0);

        while ((QSPI0->STATUS & QSPI_STATUS_SPIENSTS_Msk) != 0)
        {}
    }
}

```

```

    /* Set the QSPI data length to 32 */
    QSPI_SET_DATA_WIDTH(QSPI0, 32);

    QSPI0->CTL |= (QSPI_CTL_SPIEN_Msk);
}

SPI_Close(SPI_LCD_PORT);

while ((SPI_LCD_PORT->STATUS & SPI_STATUS_SPIENSTS_Msk) != 0)
{}

/* Set the SPI data width to 8 */
SPI_SET_DATA_WIDTH(SPI_LCD_PORT, 8);

SPI_LCD_PORT->CTL |= (SPI_CTL_SPIEN_Msk);

/* Set the display area of the LCD */
LCD_Display_Zone_Set(Pic_x, Pic_y, (Pic_x + Pic_Width - 1), (Pic_y + Pic_Height - 1));

SPI_Close(SPI_LCD_PORT);

while ((SPI_LCD_PORT->STATUS & SPI_STATUS_SPIENSTS_Msk) != 0)
{}

/* Set the SPI data width to 16 */
SPI_SET_DATA_WIDTH(SPI_LCD_PORT, 16);

SPI_LCD_PORT->CTL |= (SPI_CTL_SPIEN_Msk);

/* Initialization flag */
PDMA_For_LCD_Send_End_Flag = 1;

for (i = 0; i < Pic_Height; i++)
{
    /* QSPI master PDMA RX Initialization */
    PDMA_For_FLASH_Trigger((uint8_t *)Display_Buffer, LCD_WIDTH / 2);

    /* Clear RX Buffer */
    QSPI_ClearRxFIFO(QSPI_FLASH_PORT);

    /* Enable RX PDMA transfer */
    QSPI_TRIGGER_RX_PDMA(QSPI0);

    /* Initialization flag */
    PDMA_For_Flash_Read_End_Flag = 0;

    /* Waiting for the reading to be completed */
    while (PDMA_For_Flash_Read_End_Flag == 0);

    /* Finished reading */
    PDMA_For_Flash_Read_End_Flag = 0;

    /* Merge the image data with the character data */
    if ((Y_Number >= String_y) && (Y_Number < (String_y + String_Height)))
    {
        /* Merge "FPS:" character data with the image data. */

```

```

String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[10]),
String_x, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);
String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[11]),
String_x + String_Width * 1, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);
String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[12]),
String_x + String_Width * 2, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);
String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[13]),
String_x + String_Width * 3, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);

/* Merge "Rate" value character data with the image data. */
String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[Rate / 10]),
String_x + String_Width * 4, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);
String_Line_Fill(Display_Buffer, (uint8_t *)(&XL_Word[Rate % 10]),
String_x + String_Width * 5, String_Width, (Y_Number - String_y),
String_Color, String_Backcolor, 0);
}

Y_Number++;

/* Send the merged data */
Display_Buffer = PDMA_For_LCD_Data_Send(Display_Buffer, Pic_Width);
}

/* Waiting for the sending to be completed */
PDMA_For_LCD_Wait_Send_End();

/* Return the address of the next image data in Flash */
return (Addr + Pic_Width * Pic_Height * 2);
}

```

函式 **String_Line_Fill()** 通過將字元點陣圖資料與圖像資料合併的方式更新顯示緩衝區。像素值是根據字體點陣圖定義和顏色參數進行更新的。

```

/* Update the character data to the buffer. */
void String_Line_Fill(uint16_t *Buff, uint8_t *String, uint8_t x, uint16_t String_Width,
uint8_t Line_Num, uint16_t String_Color, uint16_t Back_Color,
uint8_t Backcolor_Enable)
{
uint16_t i = 0;
uint8_t Bit_Number = 0;
uint16_t Pixel_Cnt = 0;
uint16_t Width = (String_Width / 8);

for (i = 0; i < Width; i++)
{
for (Bit_Number = 0; Bit_Number < 8; Bit_Number++)
{
if ((String[Line_Num * Width + i] >> (7 - Bit_Number)) & 0x01)
{
Buff[x+Pixel_Cnt] = String_Color;
}
}
}
}

```

```

    }
    else
    {
        if (Backcolor_Enable == 1)
        {
            Buff[x+Pixel_Cnt] = Back_Color;
        }
    }

    Pixel_Cnt++;
}
}
}

```

函式 PDMA_For_LCD_Data_Send() 和 PDMA_For_LCD_Trigger() 實現了使用 PDMA 進行的雙緩衝式 SPI 資料傳輸，以確保液晶顯示螢幕的持續且高效更新。

```

/* Use PDMA to send data of a specific length from the cache. */
uint16_t *PDMA_For_LCD_Data_Send(uint16_t *buffer, uint16_t count)
{
    while (1)
    {
        /* Wait for the data transmission to be completed. */
        if (PDMA_For_LCD_Send_End_Flag == 1)
        {
            PDMA_For_LCD_Send_End_Flag = 0;

            /* Trigger data transmission */
            PDMA_For_LCD_Trigger(buffer, count);

            /* Switch the sending buffer */
            if (buffer == LCD_Display_Buffer0)
            {
                buffer = LCD_Display_Buffer1;
            }
            else
            {
                buffer = LCD_Display_Buffer0;
            }

            break;
        }
    }

    return buffer;
}

void PDMA_For_LCD_Trigger(uint16_t *buffer, uint16_t count)
{
    PDMA_Open(PDMA0, (1<<PDMA_FOR_LCD_CH0));

    /*=====
    SPI master PDMA TX channel configuration:
    -----
    Word length = 16 bits
    */
}

```

```

    Transfer Count = count
    Source = buffer
    Source Address = Increasing
    Destination = SPI1->TX
    Destination Address = Fixed
    Burst Type = Single Transfer

    =====*/
    PDMA_SetTransferCnt(PDMA0, PDMA_FOR_LCD_CH0, PDMA_WIDTH_16, count);
    PDMA_SetTransferAddr(PDMA0, PDMA_FOR_LCD_CH0, (uint32_t)(buffer), PDMA_SAR_INC,
                        (uint32_t)&SPI1->TX, PDMA_DAR_FIX);
    PDMA_SetTransferMode(PDMA0, PDMA_FOR_LCD_CH0, PDMA_SPI1_TX, FALSE, 0);
    PDMA_SetBurstType(PDMA0, PDMA_FOR_LCD_CH0, PDMA_REQ_SINGLE, PDMA_BURST_4);

    /* Enable PDMA channel 0 interrupt */
    PDMA_EnableInt(PDMA0, PDMA_FOR_LCD_CH0, PDMA_INT_TRANS_DONE);

    NVIC_EnableIRQ(PDMA0_IRQn);

    SPI_TRIGGER_TX_PDMA(SPI1);
}

```

Timer0 以週期性模式進行配置，具有固定的中斷間隔。中斷服務程式會遞增一個計數器值，該計數器用於計算顯示刷新率。

```

void Timer0_Init(void)
{
    /* Open Timer0 in periodic mode, enable interrupt
    and 1 interrupt tick per 100 microsecond */
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 10000);
    TIMER_EnableInt(TIMER0);

    /* Start Timer0 counting */
    TIMER_Start(TIMER0);

    /* Clear timer counter */
    u8TimerCnt = 0;

    /* Enable Timer0 NVIC */
    NVIC_EnableIRQ(TMR0_IRQn);
}

void TMR0_IRQHandler(void)
{
    if (TIMER_GetIntFlag(TIMER0) == 1)
    {
        /* Clear Timer0 time-out interrupt flag */
        TIMER_ClearIntFlag(TIMER0);

        /* Timer counter plus one */
        u8TimerCnt++;
    }
}

```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
 - M3331_Series_BSP_CMSIS_V3.00.002
- IDE 版本
 - Keil uVision 5.41
 - VS Code 1.1111.0

3.2 硬體需求

- 電路元件
 - NuMaker-M3334KI V2.0
 - NuFun-M3334KI V2.0
- 線路示意圖
 - 將 NuMaker-M3334KI V2.0 板與 NuFun-M3334KI V2.0 板，透過板載連接器對插連接起來。如圖 3-2 所示，兩組黃色框起的連接器對插，標識「A」方向對齊；兩組藍黑色框起的連接器對插，標識「B」方向對齊。MCU 位於 NuMaker-M3334KI V2.0 上，TFT 及 FLASH 位於 NuFun-M3334KI V2.0 上，接線示意圖如圖 3-1 所示。
注意：為避免 FLASH 發生寫保護，「/WP、/HOLD、/CS」引腳建議接上拉電阻。

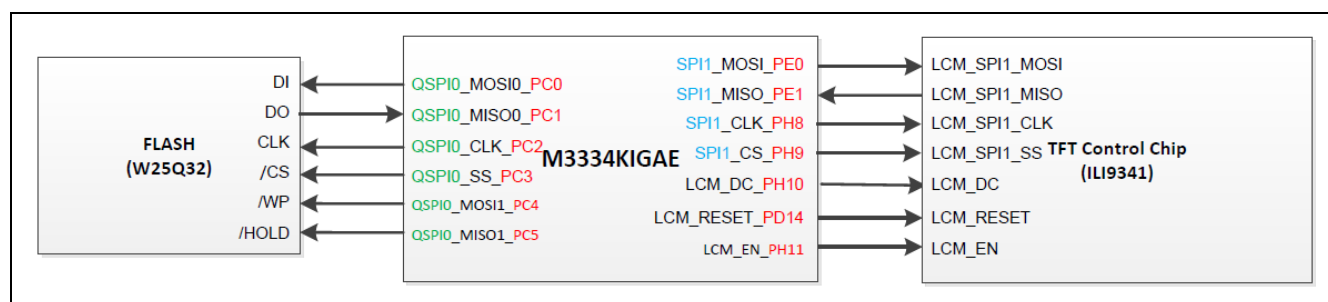


圖 3-1 接線示意圖

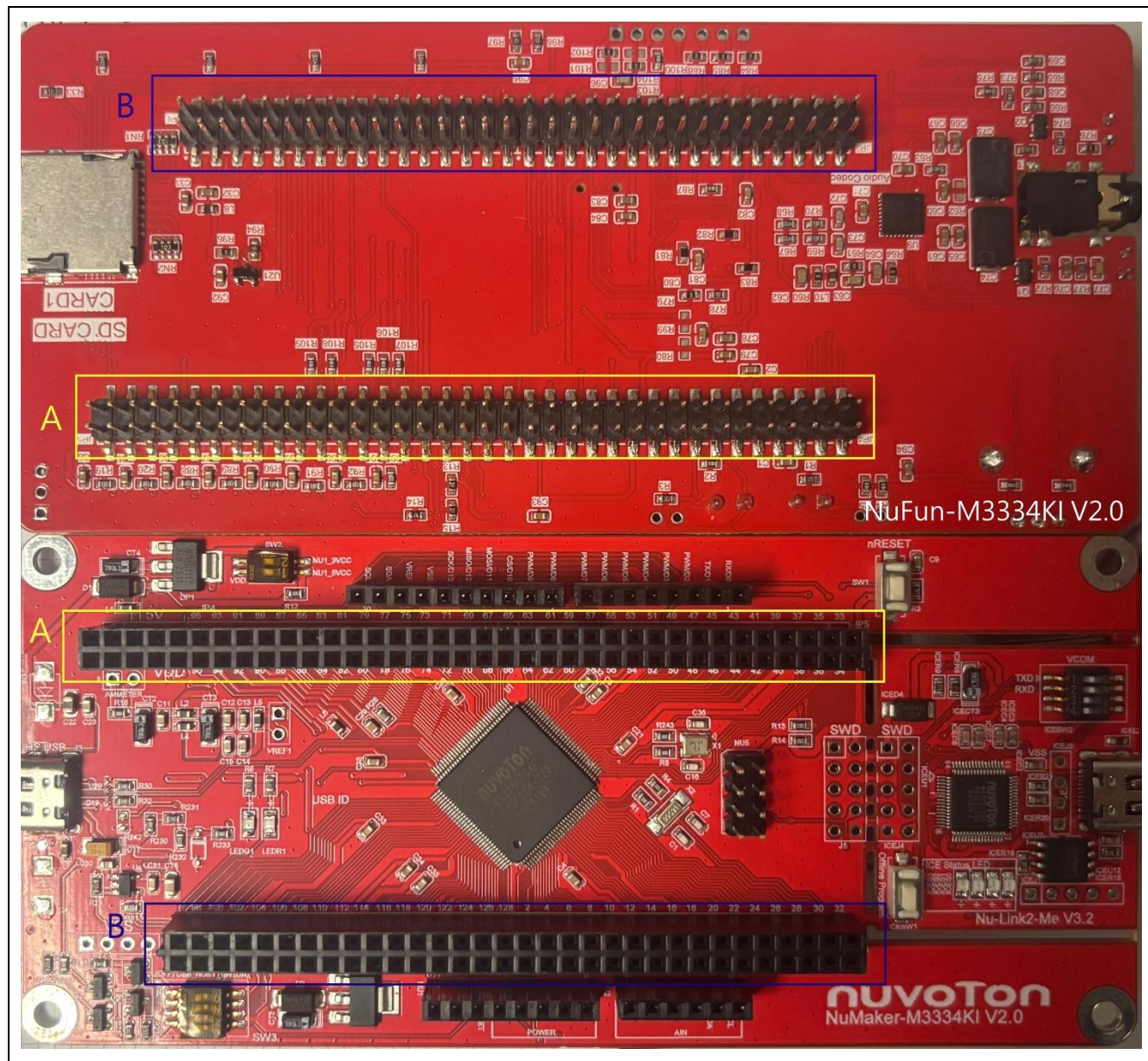


圖 3-2 實體接線圖

4. 目錄資訊

目錄結構如下圖所示。

 EC_M3331_Displaying_Animations_and_Overlaid_Characters_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	
 Project	Source file of example code
 Tool	ICP & ImagesToBin_Uncompress

圖 4-1 目錄資訊

5. 範例程式執行

5.1 Keil 平臺

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，點擊 *Display.uvprojx*。
2. 進入編譯模式介面
 - 編譯
 - 下載程式碼至記憶體
 - 進入 / 離開除錯模式
3. 進入除錯模式介面
 - 執行程式碼

5.2 VS Code 平臺

1. 選擇 **File**
 - 按一下 **Open Folder**
 - 根據目錄資訊章節選擇 Display_Animation 路徑下的 VSCode 資料夾
2. 選擇 **CMSIS**
 - 按一下 **Build solution** 編譯工程
 - 按一下 **Load & Run application** 下載程式碼至記憶體
 - 按一下 **Load & Debug application** 進入仿真模式

6. 修訂紀錄

Date	Revision	Description
2026.03.31	1.00	初始發布。
2026.08.13	1.01	修改LCD_RST巨集定義及添加Tool 源程式碼。
2026.09.20	1.02	1.在修改SPIx_CTL、SPIx_CLKDIV、 SPIx_SSCTL、SPIx_FIFOCTL暫存器前，先禁用SPI，修改完後，再使能SPI。 2.在修改QSPIx_CTL、QSPIx_CLKDIV、 QSPIx_SSCTL、QSPIx_FIFOCTL暫存器前，先禁用QSPI，修改完後，再使能QSPI。

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.*