# PDMA 搬移 FLASH 数据到 TFT 屏显示

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 M031 系列芯片将 SPI FLASH 中图片数据通过 PDMA 搬移到 TFT 屏显示 |
| BSP 版本 | M031_Series_BSP_CMSIS_V3.06.000 |
| 开发平台 | NuMaker-M031KI V1.0<br>NuTFT-SPI_320x240 Daughter Board_V1.3 |

*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

本范例代码使用 M031KIAAE 的 SPI、USPI 外设 PDMA 功能将 SPI FLASH 中图片数据搬移到 TFT 屏进行显示。

首先通过 SPI 外设将图片数据写入到 FLASH 中，然后通过 SPI 外设 RX PDMA 功能将 FLASH 中图片数据传输到 USPI 外设 TX，由 TX 驱动 TFT 屏进行显示。

## 1.1 原理

320x240 TFT 以 RGB565 模式显示，显示一屏图片需要 153600 (320*240*2) 个字节数据。FLASH (W25Q16) 存储容量为 2M Bytes，上电后将图片数据保存到 FLASH。M031KIAAE 通过 SPI 与 FLASH 进行通信，通过 USPI 与 TFT 控制芯片 (ILI9341) 进行通信。SPI 外设及 USPI 外设都支持 PDMA 功能。通过使能 SPI 外设 RX PDMA 功能实现将 FLASH 中的图片数据直接传输到 USPI 外设 TX，由 TX 驱动 TFT 进行显示。除了设置 FLASH 读数据起始地址和设置 TFT 显示初始位置之外，其他数据传输都由 PDMA 完成，进而提升 CPU 的运行效率。

## 1.2 執行結果

本范例代码执行编译和烧录后，TFT 屏显示如图 1-1 所示。




图 1-1 执行编译和烧录结果

串口打印消息如图 1-2 所示。

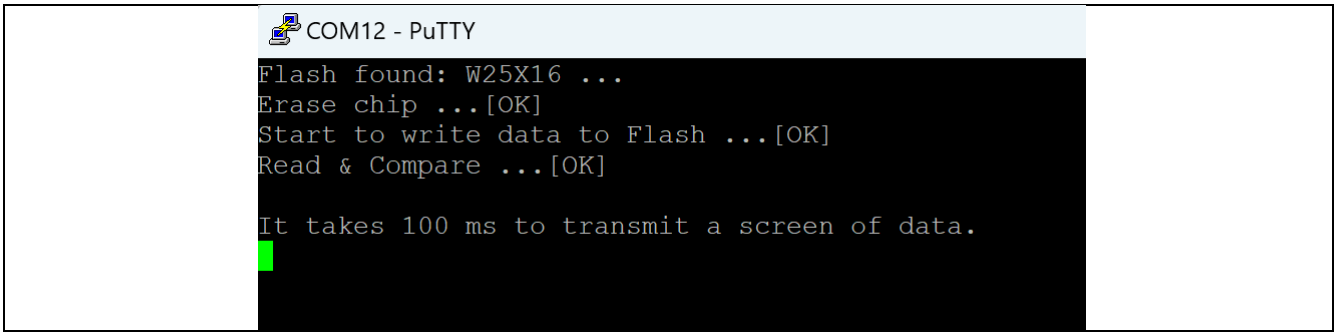



图 1-2 串口打印消息

## 2. 代码介绍

图片数据数组定义，代码位于 *Image.c*。

```
const uint8_t u8Image[DATA_PER_SCREEN] =
{
    0XF7,0XBE,0XF7,0XDE,0XF7,0XDE,0XF7,0XDE,0XF7,0XDE,0XF7,0XBE,0XF7,0XBE,0XF7,0XBE,
    0XEF,0X9D,0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,
    0XE7,0X7C,0XE7,0X7C,0XE7,0X7C,0XE7,0X9C,0XE7,0X9D,0XE7,0X9D,0XE7,0X9C,0XE7,0X9C,
    0XE7,0X7C,0XE7,0X9C,0XE7,0X9C,0XE7,0X9D,0XE7,0X9D,0XE7,0X9D,0XE7,0X9D,0XEF,0X9D,
    0XEF,0X9D,0XEF,0X9E,0XEF,0XBD,0XEF,0XBE,0XEF,0XBE,0XEF,0XBD,0XEF,0XBD,0XEF,0X9D,
    ……
    0X3B,0X67,0X32,0XC6,0X2A,0X24,0X19,0X83,0X19,0X83,0X21,0XA4,0X2A,0X05,0X2A,0X25,
    0X21,0XE4,0X21,0XC3,0X21,0XC3,0X21,0X83,0X19,0X43,0X21,0X63,0X2A,0X04,0X32,0X45,
    0X2A,0X04,0X32,0X65,0X3A,0XC7,0X2A,0X25,0X19,0X83,0X21,0XC4,0X29,0XE5,0X21,0XA4,
    0X19,0X43,0X19,0X43,0X21,0X83,0X21,0X63,0X19,0X22,0X19,0X02,0X10,0XE2,0X10,0XC2,
    0X10,0XC2,0X10,0XC2,0X10,0XC2,0X10,0XC2,0X10,0XC2,0X10,0XC2,0X10,0XC2,0X10,0XA2,
};
```

图片数据数组大小定义，代码位于 *Image.h*。

```
#define DATA_PER_SCREEN   (153600UL)
extern const uint8_t u8Image[DATA_PER_SCREEN];
```

主函数包括对系统，对 UART0、USCI_SPI0、PWM、SPI0，对 TFT 驱动芯片，对 SPI FLASH，对 Timer0 初始化，最后执行主循环。代码位于 main 函数。

```
    int main()
    {
        /* System initialization */
        SYS_Init();
        /* Initial UART0 to 115200-8n1 for print message */
        UART_Open(UART0, 115200);
        /* Initial USCI_SPI0 */
        USCI_SPI_Init();
        /* Initial PWM for backlight brightness control */
        PWM_Init();
        /* Initial SPI0 */
        SPI_Init();
        /* Initial ILI9341 LCD Panel Hardware */
        Drv_ILI9341_LCD_Init();
        /* Identify flash type, erase flash, write picture datas to flash,
        check if write successful */
        SpiFlash_Setting();
        /* Initial Timer0 */
        Timer0_Init();
        while (1)
        {
            /* Data read from FLASH is sent to TFT via PDMA */
            Copy_FlashData_To_TFT_WithPDMA();
        }
    }
```

主循环函数，用于实现将 FLASH 中数据搬移到 TFT 显示。当一屏数据传输完成后，串口输出传输所花的时间。代码位于 Copy_FlashData_To_TFT_WithPDMA 函数。

```
/* Data read from FLASH is sent to TFT via PDMA */
void Copy_FlashData_To_TFT_WithPDMA(void)
{
    uint32_t u32RegValue, u32Abort;

    /* Data Width is swtich to 8 Bits */
    Switch_DataWidth_Is_Byte(TRUE);
    /* Reset LCD display position */
    LCD_Set_Pos();
    /* Reset read start address */
    SpiFlash_SetReadAddress(0x000000);
    /* Data Width is swtich to 16 Bits */
    Switch_DataWidth_Is_Byte(FALSE);
    /* Init PDMA */
    PDMA_Init();
    /* CS = 0 : select chip */
    USPI_CS_CLR;
    /* DC = 1 : write data */
    LCM_DC_SET;
    /* Enable SPI RX PDMA function */
    SPI_TRIGGER_RX_PDMA(SPI0);
    while (1)
    {
        /* Get interrupt status */
        u32RegValue = PDMA_GET_INT_STATUS(PDMA);
        /* Check the PDMA transfer done interrupt flag */
        if (u32RegValue &PDMA_INTSTS_TDIF_Msk)
        {
            /* Check the PDMA transfer done flags */
            if ((PDMA_GET_TD_STS(PDMA) & (0x01 << SPI_MASTER_RX_DMA_CH)) ==
                    (0x01 << SPI_MASTER_RX_DMA_CH))
            {
                /* Clear the PDMA transfer done flags */
                PDMA_CLR_TD_FLAG(PDMA, (0x01 << SPI_MASTER_RX_DMA_CH));
                /* Enable SPI RX PDMA function */
                SPI_TRIGGER_RX_PDMA(SPI0);
                u8TransferCnt++;
                if (u8TransferCnt >= TRANSFER_TIMES)
                {
                    u8TransferCnt = 0;
                    printf("\nIt takes %d ms to transmit a screen of data. \n",
                           (u8TimerCnt*10));
                    /* Disable Timer0 NVIC */
                    NVIC_DisableIRQ(TMR0_IRQn);
                    while (1);
                }
            }
        }
        /* Check the DMA transfer abort interrupt flag */
        if (u32RegValue &PDMA_INTSTS_ABTIF_Msk)
        {
            /* Get the target abort flag */
            u32Abort = PDMA_GET_ABORT_STS(PDMA);
            /* Clear the target abort flag */
            PDMA_CLR_ABORT_FLAG(PDMA, u32Abort);
```

```
            printf("\nabort happen\n");
        }

        /* Check the DMA time-out interrupt flag */
        if (u32RegValue &(PDMA_INTSTS_REQTOF0_Msk | PDMA_INTSTS_REQTOF1_Msk))
        {
            /* Clear the time-out flag */
            PDMA->INTSTS = u32RegValue &(PDMA_INTSTS_REQTOF0_Msk |
                                         PDMA_INTSTS_REQTOF1_Msk);
            printf("\ntime-out happen\n");
        }
    }
}
```

设置命令的时候将传输位宽设为 8 bits，传输数据的时候将位宽设置为 16 bits。16 bits 传输位宽比 8 bits 传输位宽更高效。代码位于 Switch_DataWidth_Is_Byte 函数。

```
void Switch_DataWidth_Is_Byte(_Bool bFlag)
{
    if (bFlag)
    {
        /* USPI0 Transfer Data width is 8 Bits */
        USPI0->LINECTL &= ~USPI_LINECTL_DWIDTH_Msk;
        USPI0->LINECTL |= (0x08 << USPI_LINECTL_DWIDTH_Pos);
        /* SPI0 Transfer Data width is 8 Bits*/
        SPI0->CTL &= ~SPI_CTL_DWIDTH_Msk;
        SPI0->CTL |= (0x08 << SPI_CTL_DWIDTH_Pos);
    }
    else
    {
        /* USPI0 Transfer Data width is 16 Bits */
        USPI0->LINECTL &= ~USPI_LINECTL_DWIDTH_Msk;
        USPI0->LINECTL |= (0 << USPI_LINECTL_DWIDTH_Pos);
        /* SPI0 Transfer Data width is 16 Bits */
        SPI0->CTL &= ~SPI_CTL_DWIDTH_Msk;
        SPI0->CTL |= (0x10 << SPI_CTL_DWIDTH_Pos);
    }
}
```

Timer0 初始化，设置每隔 10 ms 进入一次中断，计数器初始化为 0，代码位于 Timer0_Init 函数中。定时器中断函数，每进入中断对计数器进行加 1，代码位于 TMR0_IRQHandler 函数。

```
void Timer0_Init(void)
{
    /* Open Timer0 in periodic mode, enable interrupt
    and 1 interrupt tick per 10 millisecond */
    TIMER_Open(TIMER0, TIMER_PERIODIC_MODE, 100);
    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++;
    }
}
```

因为 TFT 以 RGB565 模式显示，一个像素点需要 16 bits 来表示，且 MCU 与 TFT 之间传输基本单位也是 16 bits，因此传输一屏图片即 320x240 像素点需要传输 76800x16 bits 数据。因为一个 PDMA table 传输数量是用 16 bits 定义，最大传输数据为 65536 x16 bits，因此传输一屏图片需要分 2 次传输，分别传输 38400 x16 bits 数据。PDMA 操作模式设置为 Scatter-Gather。传输数量设置为 38400。传输位宽设置为 16 bits。传输类型设置为 signal。源地址固定为 SPI0 RX 寄存器，目标地址固定为 USPI0 TX 寄存器。代码位于 PDMA_Init 函数。

```
void PDMA_Init(void)
{
    /* Descriptor table 1 configuration */
    DMA_DESC[0].ctl = ((TRANSFER_LEN - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_16 |
                      PDMA_SAR_FIX | PDMA_DAR_FIX | PDMA_REQ_SINGLE | PDMA_OP_SCATTER;
    DMA_DESC[0].src = (uint32_t) &SPI0->RX;
    DMA_DESC[0].dest = (uint32_t) &USPI0->TXDAT;
    DMA_DESC[0].offset = (uint32_t) &DMA_DESC[1] - (PDMA->SCATBA);
    /* Descriptor table 2 configuration */
    DMA_DESC[1].ctl = ((TRANSFER_LEN - 1) << PDMA_DSCT_CTL_TXCNT_Pos) | PDMA_WIDTH_16 |
                      PDMA_SAR_FIX | PDMA_DAR_FIX | PDMA_REQ_SINGLE | PDMA_OP_BASIC;
    DMA_DESC[1].src = (uint32_t) &SPI0->RX;
    DMA_DESC[1].dest = (uint32_t) &USPI0->TXDAT;
    DMA_DESC[1].offset = 0;
    /* Open PDMA channel 1 for SPI RX */
    PDMA_Open(PDMA, (1 << SPI_MASTER_RX_DMA_CH));
    /* Configure PDMA transfer mode */
    PDMA_SetTransferMode(PDMA, SPI_MASTER_RX_DMA_CH, PDMA_SPI0_RX, 1, (uint32_t)
                         &DMA_DESC[0]);
}
```

# 3. 软件与硬件需求

## 3.1 软件需求

- BSP 版本
  - M031_Series_BSP_CMSIS_V3.06.000
- IDE 版本
  - Keil uVersion 5.36

## 3.2 硬件需求

- 电路组件
  - NuMaker-M031KI V1.0
  - NuTFT-SPI_320x240 Daughter Board_V1.3
- 线路示意图
  - FLASH 和 TFT Control Chip 位于 NuTFT-SPI_320x240 Daughter Board_V1.3 上
  - M031KIAAE 位于 NuMaker-M031KI V1.0 上

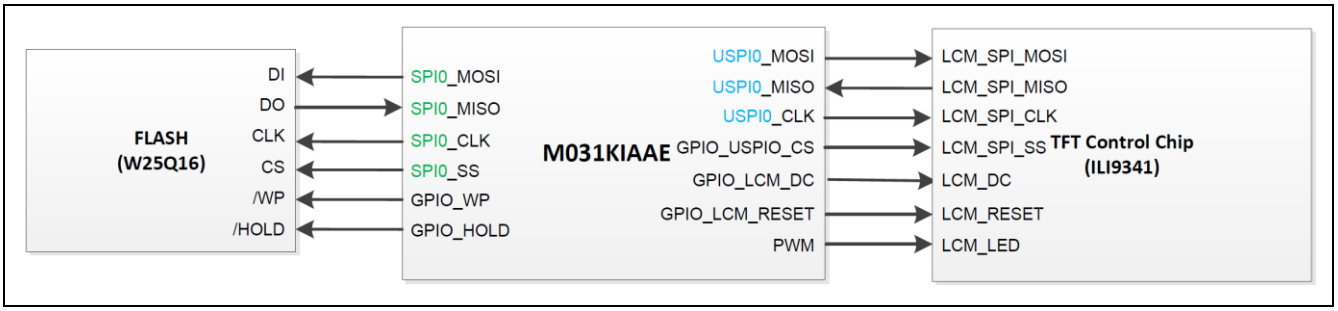



图 3-1 线路示意图

# 4. 目录信息

| | |
|---|---|
| EC_ M031_PDMA_FLASH_TO_TFT_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 |

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 *M031_PDMA_FLASH_TO_TFT.uvprojx*。
2. 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
3. 进入仿真模式界面
   - 执行代码

# 6. 修订纪录

| Date | Revision | Description |
|---|---|---|
| 2024.07.10 | 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.*