Application Note

ATK-MC2640 模块使用说明

高性能 200W 高清摄像头模块

使用说明

正点原子

广州市星翼电子科技有限公司

修订历史

版本	日期	原因
V1.0	2022/06/25	第一次发布
V1.1	2023/03/11	添加对阿波罗 STM32F429 开发板的阿波罗 STM32F767 开发板的支持
V1.2	2023/04/15	添加对阿波罗 STM32H743 开发板的支持

目 录

1，硬件连接

1.1 正点原子精英 STM32F103 开发板

ATK-MC2640 模块可直接与正点原子精英 STM32F103 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
精英 STM32F103 开发板	V3.3	PD6	PG14	PG15	PC1	PC3	PC5	PC7	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
精英 STM32F103 开发板	GND	PD3	PG13	PC0	PC2	PC4	PC6	PB4	PB3

表 1.1.1 ATK-MC2640 模块与精英 STM32F103 开发板连接关系

1.2 正点原子战舰 STM32F103 开发板

ATK-MC2640 模块可直接与正点原子战舰 STM32F103 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
战舰 STM32F103 开发板	V3.3	PD6	PG14	PG15	PC1	PC3	PC5	PC7	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
战舰 STM32F103 开发板	GND	PD3	PG13	PC0	PC2	PC4	PC6	PB4	PB3

表 1.2.1 ATK-MC2640 模块与战舰 STM32F103 开发板连接关系

1.3 正点原子探索者 STM32F407 开发板

ATK-MC2640 模块可直接与正点原子探索者 STM32F407 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
探索者 STM32F407 开发板	V3.3	PB7	PA4	PG15	PC7	PC9	PB6	PE6	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
探索者 STM32F407 开发板	GND	PD6	PD7	PC6	PC8	PC11	PE5	PA6	PG9

表 1.3.1 ATK-MC2640 模块与探索者 STM32F407 开发板连接关系

1.4 正点原子 MiniSTM32H750 开发板

ATK-MC2640 模块可直接与正点原子 MiniSTM32H750 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
MiniSTM32H750 开发板	V3.3	PB7	PA4	PA7	PC7	PC9	PD3	PB9	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
MiniSTM32H750 开发板	GND	PB10	PB11	PC6	PC8	PC11	PB8	PA6	PC4

表 1.4.1 ATK-MC2640 模块与 MiniSTM32H750 开发板连接关系

1.5 正点原子阿波罗 STM32F429 开发板

ATK-MC2640 模块可直接与正点原子阿波罗 STM32F429 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
阿波罗 STM32F429 开发板	V3.3	PB7	PH8	PA15	PC7	PC9	PD3	PB9	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
阿波罗 STM32F429 开发板	GND	PB4	PB3	PC6	PC8	PC11	PB8	PA6	EX_P2

表 1.5.1 ATK-MC2640 模块与阿波罗 STM32F429 开发板连接关系

1.6 正点原子阿波罗 STM32F767 开发板

ATK-MC2640 模块可直接与正点原子阿波罗 STM32F767 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系								
ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
阿波罗 STM32F767 开发板	V3.3	PB7	PH8	PA15	PC7	PC9	PD3	PB9	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
阿波罗 STM32F767 开发板	GND	PB4	PB3	PC6	PC8	PC11	PB8	PA6	EX_P2

表 1.6.1 ATK-MC2640 模块与阿波罗 STM32F767 开发板连接关系

1.7 正点原子阿波罗 STM32H743 开发板

ATK-MC2640 模块可直接与正点原子阿波罗 STM32H743 开发板板载的 CAMERA 摄像头接口进行连接，连接后可通过 SCCB 等相关协议进行通讯，具体的连接关系，如下图所示：

模块对应开发板	连接关系

ATK-MC2640 模块	3.3V	VSYNC	HREF	RST	D1	D3	D5	D7	FLASH
阿波罗 STM32H743 开发板	V3.3	PB7	PH8	PA15	PC7	PC9	PD3	PB9	PA8
模块对应开发板	**连接关系**								
ATK-MC2640 模块	GND	SCL	SDA	D0	D2	D4	D6	PCLK	PWDN
阿波罗 STM32H743 开发板	GND	PB4	PB3	PC6	PC8	PC11	PB8	PA6	EX_P2

表 1.7.1 ATK-MC2640 模块与阿波罗 STM32H743 开发板连接关系

2，实验功能

2.1 ATK-MC2640 模块测试实验（GPIO）

2.1.1 功能说明

在本实验中，开发板主控芯片通过模拟 SCCB 协议对 ATK-MC2640 模块中的摄像头传感器进行配置等通讯，并直接通过 GPIO 操作 8 位数据并口以及其他控制引脚完成获取 ATK-MC2640 模块输出的图像数据，然后将获取到的图像数据实时地显示至 LCD。

2.1.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MC2640 子文件夹，该文件夹中就包含了 ATK-MC2640 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MC2640/
|-- atk_mc2640.c
|-- atk_mc2640.h
|-- atk_mc2640_cfg.h
|-- atk_mc2640_sccb.c
`-- atk_mc2640_sccb.h
```

图 2.1.2.1 ATK-MC2640 模块驱动代码

2.1.2.1 ATK-MC2640 模块接口驱动

在图 2.1.2.1 中，atk_mc2640_sccb.c 和 atk_mc2640_sccb.h 是开发板与 ATK-MC2640 模块通讯而使用的模拟 SCCB 驱动文件，关于模拟 SCCB 的驱动介绍，请查看正点原子各个开发板对应的开发指南中模拟 SCCB 对应的章节。

2.1.2.2 ATK-MC2640 模块驱动

在图 2.1.2.1 中，atk_mc2640.c 和 atk_mc2640.h 是 ATK-MC2640 模块的驱动文件，包含了 ATK-MC2640 模块初始化、读写寄存器操作以及获取输出图像等相关 API 函数。函数比较多，下面仅介绍几个重要的 API 函数。

1. 函数 atk_mc2640_init()

该函数用于初始化 ATK-MC2640 模块，具体的代码，如下所示：

```
/**
 * @brief   初始化 ATK-MC2640 模块
 * @param   无
 * @retval  ATK_MC2640_EOK      : ATK-MC2640 模块初始化成功
 *          ATK_MC2640_ERROR    : 通讯出错，ATK-MC2640 模块初始化失败
 *          ATK_MC2640_ENOMEM   : 内存不足，ATK-MC2640 模块初始化失败
 */
uint8_t atk_mc2640_init(void)
{
    uint16_t mid;
    uint16_t pid;
```

```
    atk_mc2640_hw_init();            /* ATK-MC2640 模块硬件初始化 */
    atk_mc2640_exit_power_down();    /* ATK-MC2640 模块退出掉电模式 */
    atk_mc2640_hw_reset();           /* ATK-MC2640 模块硬件复位 */
    atk_mc2640_sccb_init();          /* ATK-MC2640 SCCB 接口初始化 */
    atk_mc2640_sw_reset();           /* ATK-MC2640 模块软件复位 */

    mid = atk_mc2640_get_mid();      /* 获取制造商 ID */
    if (mid != ATK_MC2640_MID)
    {
        return ATK_MC2640_ERROR;
    }

    pid = atk_mc2640_get_pid();      /* 获取产品 ID */
    if (pid != ATK_MC2640_PID)
    {
        return ATK_MC2640_ERROR;
    }

    atk_mc2640_init_reg();/* 初始化 ATK-MC2640 寄存器配置 */

    g_atk_mc2640_sta.read.line_buf = mymalloc(SRAMIN,
                        g_atk_mc2640_sta.output.width * sizeof(uint16_t));

    if (g_atk_mc2640_sta.read.line_buf == NULL)
    {
        return ATK_MC2640_ENOMEM;
    }

    return ATK_MC2640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc2640_init()就是通过模拟 SCCB 协议与 ATK-MC2640 模块进行通讯，首先通过读取 ATK-MC2640 模块中寄存器的值来判断通讯是否无误，然后再向其寄存器写入对应的配置值，以此完成对 ATK-MC2640 模块的初始化，此外，在初始化函数中还申请了一块用于存放 ATK-MC2640 模块输出的一行数据的内存空间这在获取 ATK-MC2640 模块输出的一帧图像数据时，会使用到。

2. 函数 atk_mc2640_set_xxx()

函数 atk_mc2640_set_xxx()为一系列用于配置 ATK-MC2640 模块输出图像的函数，其中包括灯光模式、色彩饱和度、亮度、对比度和特殊效果，这些函数都是通过读写 ATK-MC2640 模块上摄像头传感器的寄存器来完成的，以设置 ATK-MC2640 模块输出图像对比度的函数 atk_mc2640_set_contrast()为例，其具体的代码如下所示：

```
/**
 * @brief   设置 ATK-MC2640 模块对比度
 * @param   contrast:   ATK_MC2640_CONTRAST_0: +2
 *                      ATK_MC2640_CONTRAST_1: +1
```

```
 *                      ATK_MC2640_CONTRAST_2: 0
 *                      ATK_MC2640_CONTRAST_3: -1
 *                      ATK_MC2640_CONTRAST_4: -2
 * @retval  ATK_MC2640_EOK      : 设置 ATK-MC2640 模块对比度成功
 *          ATK_MC2640_EINVAL   : 传入参数错误
 */
uint8_t atk_mc2640_set_contrast(atk_mc2640_contrast_t contrast)
{
    switch (contrast)
    {
        case ATK_MC2640_CONTRAST_0:
        {
            atk_mc2640_reg_bank_select(ATK_MC2640_REG_BANK_DSP);
            atk_mc2640_write_reg(0x7C, 0x00);
            atk_mc2640_write_reg(0x7D, 0x04);
            atk_mc2640_write_reg(0x7C, 0x07);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x28);
            atk_mc2640_write_reg(0x7D, 0x0C);
            atk_mc2640_write_reg(0x7D, 0x06);
            break;
        }
        case ATK_MC2640_CONTRAST_1:
        {
            atk_mc2640_reg_bank_select(ATK_MC2640_REG_BANK_DSP);
            atk_mc2640_write_reg(0x7C, 0x00);
            atk_mc2640_write_reg(0x7D, 0x04);
            atk_mc2640_write_reg(0x7C, 0x07);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x24);
            atk_mc2640_write_reg(0x7D, 0x16);
            atk_mc2640_write_reg(0x7D, 0x06);
            break;
        }
        case ATK_MC2640_CONTRAST_2:
        {
            atk_mc2640_reg_bank_select(ATK_MC2640_REG_BANK_DSP);
            atk_mc2640_write_reg(0x7C, 0x00);
            atk_mc2640_write_reg(0x7D, 0x04);
            atk_mc2640_write_reg(0x7C, 0x07);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x06);
```

```
            break;
        }
        case ATK_MC2640_CONTRAST_3:
        {
            atk_mc2640_reg_bank_select(ATK_MC2640_REG_BANK_DSP);
            atk_mc2640_write_reg(0x7C, 0x00);
            atk_mc2640_write_reg(0x7D, 0x04);
            atk_mc2640_write_reg(0x7C, 0x07);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x1C);
            atk_mc2640_write_reg(0x7D, 0x2A);
            atk_mc2640_write_reg(0x7D, 0x06);
            break;
        }
        case ATK_MC2640_CONTRAST_4:
        {
            atk_mc2640_reg_bank_select(ATK_MC2640_REG_BANK_DSP);
            atk_mc2640_write_reg(0x7C, 0x00);
            atk_mc2640_write_reg(0x7D, 0x04);
            atk_mc2640_write_reg(0x7C, 0x07);
            atk_mc2640_write_reg(0x7D, 0x20);
            atk_mc2640_write_reg(0x7D, 0x18);
            atk_mc2640_write_reg(0x7D, 0x34);
            atk_mc2640_write_reg(0x7D, 0x06);
            break;
        }
        default:
        {
            return ATK_MC2640_EINVAL;
        }
    }

    return ATK_MC2640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc2640_set_contrast()就是通过往 ATK-MC2640 模块内存的各个寄存器写入不同的值，以完成配置 ATK-MC2640 模块输出图像对比度的操作。

3. 函数 atk_mc2640_read_reg()和函数 atk_mc2640_write_reg()

这两个函数该函数用于读写 ATK-MC2640 模块的寄存器，具体的代码如下所示：

```
/**
 * @brief   ATK-MC2640 模块写寄存器
 * @param   reg: 寄存器地址
 *          dat: 待写入的值
 * @retval  无
 */
```

```
static void atk_mc2640_write_reg(uint8_t reg, uint8_t dat)
{
    atk_mc2640_sccb_3_phase_write(ATK_MC2640_SCCB_ADDR, reg, dat);
}

/**
 * @brief   ATK-MC2640 模块读寄存器
 * @param   reg: 寄存器的地址
 * @retval  读取到的寄存器值
 */
static uint8_t atk_mc2640_read_reg(uint8_t reg)
{
    uint8_t dat = 0;

    atk_mc2640_sccb_2_phase_write(ATK_MC2640_SCCB_ADDR, reg);
    atk_mc2640_sccb_2_phase_read(ATK_MC2640_SCCB_ADDR, &dat);

    return dat;
}
```

从上面的代码中可以看出，读写 ATK-MC2640 模块的寄存器还是比较简答的，仅需通过 SCCB 分别进行 2 相写、2 相读传输和 3 相写传输即可完成。

4. 函数 atk_mc2640_get_frame()

该函数用于获取 ATK-MC2640 模块输出的一帧图像，具体的代码图下所示：

```
/**
 * @brief   获取 ATK-MC2640 模块输出的一帧图像数据
 * @param   dts_addr        : 帧数据的接收缓冲的首地址
 *          type            :   ATK_MC2640_GET_TYPE_DTS_8B_NOINC    :
 *                                  图像数据以字节方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_8B_INC      :
 *                                  图像数据以字节方式写入目的地址，目的地址自动增加
 *                              ATK_MC2640_GET_TYPE_DTS_16B_NOINC   :
 *                                  图像数据以半字方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_16B_INC     :
 *                                  图像数据以半字方式写入目的地址，目的地址自动增加
 *                              ATK_MC2640_GET_TYPE_DTS_32B_NOINC   :
 *                                  图像数据以字方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_32B_INC     :
 *                                  图像数据以字方式写入目的地址，目的地址自动增加
 *          before_transfer: 帧数据传输前，需要完成的事务，可为 NULL
 * @retval  ATK_MC2640_EOK      : 获取 ATK-MC2640 模块输出的一帧图像数据成功
 *          ATK_MC2640_EINVAL   : 传入参数错误
 *          ATK_MC2640_EEMPTY   : 图像数据为空
 */
uint8_t atk_mc2640_get_frame(  uint32_t dts_addr,
```

```
                                atk_mc2640_get_type_t type,
                                void (*before_transfer)(void))
{
    uint32_t meminc;
    uint32_t memdataalignment;
    uint16_t pixel_cnt = 0;
    uint16_t line_cnt = 0;
    uint16_t dts_offset;

    switch (type)
    {
        case ATK_MC2640_GET_TYPE_DTS_8B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            dts_offset = 0;
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_8B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            dts_offset = g_atk_mc2640_sta.output.width << 1;
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_16B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            dts_offset = 0;
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_16B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            dts_offset = g_atk_mc2640_sta.output.width << 1;
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_32B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            dts_offset = 0;
```

```
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_32B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            dts_offset = g_atk_mc2640_sta.output.width << 1;
            break;
        }
        default:
        {
            return ATK_MC2640_EINVAL;
        }
    }

    atk_mc2640_dma_init(meminc, memdataalignment);

    if (ATK_MC2640_READ_VSYNC() != 0)
    {
        return ATK_MC2640_EEMPTY;
    }

    if (before_transfer != NULL)
    {
        before_transfer();
    }

    while (line_cnt < g_atk_mc2640_sta.output.height)
    {
        while (ATK_MC2640_READ_HREF() != 0)
        {
            while (ATK_MC2640_READ_PCLK() == 0);
            g_atk_mc2640_sta.read.line_buf[pixel_cnt++] =
                                                atk_mc2640_get_byte_data();
            while (ATK_MC2640_READ_PCLK() != 0);
            while (ATK_MC2640_READ_PCLK() == 0);
            g_atk_mc2640_sta.read.line_buf[pixel_cnt++] =
                                                atk_mc2640_get_byte_data();
            while (ATK_MC2640_READ_PCLK() != 0);
        }

        if (pixel_cnt != 0)
        {
            HAL_DMA_Abort(&g_atk_mc2640_sta.read.dma_handle);
```

```
            HAL_DMA_Start(  &g_atk_mc2640_sta.read.dma_handle,
                            (uint32_t)g_atk_mc2640_sta.read.line_buf,
                            dts_addr,
                            g_atk_mc2640_sta.output.width);
            dts_addr += dts_offset;
            pixel_cnt = 0;
            line_cnt++;
        }
    }

    return ATK_MC2640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc2640_get_frame()就是通过 GPIO 操作 8 位并口和其他的控制引脚获取 ATK-MC2640 模块输出的图像数据的，并且该函数还提供了两种目的地址的操作方式，一种为固定目的地址，这种方式一般可以用于将图像数据传输至 FSMC 或 FMC 方式连接的 TFTLCD 上进行显示，第二种为目的地址自动递增的方式，这种方式一般适用于将图像数据存放值内存中，并且还支持了多种目的地址的多种数据位宽。

2.1.2.4 实验测试代码

实验的测试代码为文件 demo.c，在工程目录下的 User 子目录中。测试代码的入口函数为 demo_run()，具体的代码，如下所示：

```
/**
 * @brief   例程演示入口函数
 * @param   无
 * @retval  无
 */
void demo_run(void)
{
    uint8_t ret;

    /* 初始化内部 SRAM 内存池 */
    my_mem_init(SRAMIN);
    /* 初始化 ATK-MC2640 模块 */
    ret  = atk_mc2640_init();
    /* 输出图像格式 */
    ret |= atk_mc2640_set_output_format(ATK_MC2640_OUTPUT_FORMAT_RGB565);
    /* 输出图像分辨率 */
    ret |= atk_mc2640_set_output_size(lcddev.width, lcddev.height);
    if (ret != 0)
    {
        printf("ATK-MC2640 Init Failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
```

```
        }
    }

    /* 输出速率 */
    demo_set_outspeed_suit_lcd();
    /* 设置灯光模式 */
    atk_mc2640_set_light_mode(ATK_MC2640_LIGHT_MODE_SUNNY);
    /* 设置色彩饱和度 */
    atk_mc2640_set_color_saturation(ATK_MC2640_COLOR_SATURATION_1);
    /* 设置亮度 */
    atk_mc2640_set_brightness(ATK_MC2640_BRIGHTNESS_1);
    /* 设置对比度 */
    atk_mc2640_set_contrast(ATK_MC2640_CONTRAST_2);
    /* 设置特殊效果 */
    atk_mc2640_set_special_effect(ATK_MC2640_SPECIAL_EFFECT_NORMAL);

    while (1)
    {
        /* 将获取到的图像数据，显示至 LCD */
        atk_mc2640_get_frame(   (uint32_t)&LCD->LCD_RAM,
                                ATK_MC2640_GET_TYPE_DTS_16B_NOINC,
                                demo_reset_lcd);
    }
}
```

上面的代码还是比较简单的，就是将 ATK-MC2640 模块输出的图像数据显示至 LCD，不过要注意的是，为了适应不同尺寸的 LCD，需要通过函数 demo_set_outspeed_suit_lcd()配置 ATK-MC2640 模块的输出速率。

2.1.3 实验现象

将 ATK-MC2640 模块按照第一节“硬件连接”中介绍的连接方式与开发板连接，并将实验代码编译烧录至开发板中，如果此时开发板连接 LCD，那么 LCD 显示的内容，如下图所示：

图 2.1.3.1 LCD 显示内容一

同时，通过串口调试助手输出实验信息，如下图所示：

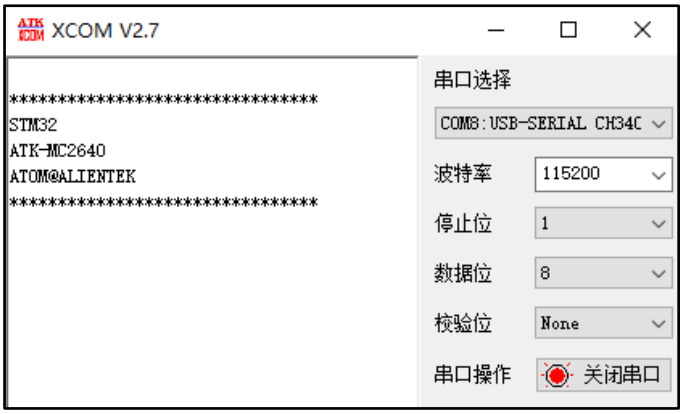

图 2.1.3.2 串口调试助手显示内容一

接下来，如果 ATK-MC2640 模块初始化成功，则会在 LCD 上显示 ATK-MC2640 模块输出的图像，如下图所示：

图 2.1.3.3 LCD 显示内容二

2.2 ATK-MC2640 模块测试实验（DCMI）

2.2.1 功能说明

在本实验中，开发板主控芯片通过模拟 SCCB 协议对 ATK-MC2640 模块中的摄像头传感器进行配置等通讯，并通过 DCMI 接口获取 ATK-MC2640 模块输出的图像数据，然后将获取到的图像数据实时地显示至 LCD。

注意：因为 STM32F1 没有 DCMI 接口，因此 STM32F1 系列开发板无法使用本实验。

2.2.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MC2640 子文件夹，该文件夹中就包含了 ATK-MC2640 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MC2640/
|-- atk_mc2640.c
|-- atk_mc2640.h
|-- atk_mc2640_cfg.h
|-- atk_mc2640_dcmi.c
|-- atk_mc2640_dcmi.h
|-- atk_mc2640_sccb.c
`-- atk_mc2640_sccb.h
```

图 2.2.2.1 ATK-MC2640 模块驱动代码

2.2.2.1 ATK-MC2640 模块 SCCB 接口驱动

请见第 2.1 小节《ATK-MC2640 模块测试实验（GPIO）》。

2.2.2.2 ATK-MC2640 模块 DCMI 接口驱动

在图 2.2.2.1 中，atk_mc2640_dcmi.c 和 atk_mc2640_dcmi.h 是开发板与 ATK-MC2640 模块通讯而使用的 DCMI 驱动文件，关于 DCMI 的驱动介绍，请查看正点原子各个开发板对应的开发指南中 DCMI 对应的章节。

2.2.2.3 ATK-MC2640 模块驱动

在图 2.1.2.1 中，atk_mc2640.c 和 atk_mc2640.h 是 ATK-MC2640 模块的驱动文件，包含了 ATK-MC2640 模块初始化、读写寄存器操作以及获取输出图像等相关 API 函数，这些函数大部分与第 2.1 小节《ATK-MC2640 模块测试实验（GPIO）》中的类似，最大的不同在于初始化函数 atk_mc2640_init()和获取图像数据函数 atk_mc2640_get_frame()，其中初始化函数的不同指出在于，多指行了一步 DCMI 的初始化，获取图像数据的函数差异比较大，具体的代码如下所示：

```
/**
 * @brief   获取 ATK-MC2640 模块输出的一帧图像数据
 * @param   dts_addr        : 帧数据的接收缓冲的首地址
 *          type            :   ATK_MC2640_GET_TYPE_DTS_8B_NOINC    :
 *                                  图像数据以字节方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_8B_INC      :
 *                                  图像数据以字节方式写入目的地址，目的地址自动增加
 *                              ATK_MC2640_GET_TYPE_DTS_16B_NOINC   :
 *                                  图像数据以半字方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_16B_INC     :
 *                                  图像数据以半字方式写入目的地址，目的地址自动增加
 *                              ATK_MC2640_GET_TYPE_DTS_32B_NOINC   :
 *                                  图像数据以字方式写入目的地址，目的地址固定不变
 *                              ATK_MC2640_GET_TYPE_DTS_32B_INC     :
 *                                  图像数据以字方式写入目的地址，目的地址自动增加
 *          before_transfer: 帧数据传输前，需要完成的事务，可为 NULL
 * @retval  ATK_MC2640_EOK      : 获取 ATK-MC2640 模块输出的一帧图像数据成功
 *          ATK_MC2640_EINVAL   : 传入参数错误
 *          ATK_MC2640_EEMPTY   : 图像数据为空
 */
uint8_t atk_mc2640_get_frame(  uint32_t dts_addr,
```

```
                                atk_mc2640_get_type_t type,
                                void (*before_transfer)(void))
{
    uint32_t meminc;
    uint32_t memdataalignment;
    uint32_t len;

    switch (type)
    {
        case ATK_MC2640_GET_TYPE_DTS_8B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint8_t);
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_8B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint8_t);
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_16B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint16_t);
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_16B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint16_t);
            break;
```

```
        }
        case ATK_MC2640_GET_TYPE_DTS_32B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint32_t);
            break;
        }
        case ATK_MC2640_GET_TYPE_DTS_32B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            len = ( g_atk_mc2640_sta.output.width*
                    g_atk_mc2640_sta.output.height) /
                    sizeof(uint32_t);
            break;
        }
        default:
        {
            return ATK_MC2640_EINVAL;
        }
    }

    if (before_transfer != NULL)
    {
        before_transfer();
    }

    atk_mc2640_dcmi_start(dts_addr, meminc, memdataalignment, len);

    return ATK_MC2640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc2640_get_frame()就是通过 DCMI 接口获取 ATK-MC2640 模块输出的图像数据，并且该函数还提供了两种目的地址的操作方式，一种为固定目的地址，这种方式一般可以用于将图像数据传输至 FSMC 或 FMC 方式连接的 TFTLCD 上进行显示，第二种为目的地址自动递增的方式，这种方式一般适用于将图像数据存放值内存中，并且还支持了多种目的地址的多种数据位宽。

2.2.2.4 实验测试代码

实验的测试代码为文件 demo.c，在工程目录下的 User 子目录中。测试代码的入口函数为 demo_run()，具体的代码，如下所示：

```
/**
 * @brief    例程演示入口函数
```

```
 * @param   无
 * @retval  无
 */
void demo_run(void)
{
    uint8_t ret;

    /* 初始化内部 SRAM 内存池 */
    my_mem_init(SRAMIN);
    /* 初始化 ATK-MC2640 模块 */
    ret  = atk_mc2640_init();
    /* 输出图像格式 */
    ret |= atk_mc2640_set_output_format(ATK_MC2640_OUTPUT_FORMAT_RGB565);
    /* 输出图像分辨率 */
    ret |= atk_mc2640_set_output_size(lcddev.width, lcddev.height);
    if (ret != 0)
    {
        printf("ATK-MC2640 Init Failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    /* 设置灯光模式 */
    atk_mc2640_set_light_mode(ATK_MC2640_LIGHT_MODE_SUNNY);
    /* 设置色彩饱和度 */
    atk_mc2640_set_color_saturation(ATK_MC2640_COLOR_SATURATION_1);
    /* 设置亮度 */
    atk_mc2640_set_brightness(ATK_MC2640_BRIGHTNESS_1);
    /* 设置对比度 */
    atk_mc2640_set_contrast(ATK_MC2640_CONTRAST_2);
    /* 设置特殊效果 */
    atk_mc2640_set_special_effect(ATK_MC2640_SPECIAL_EFFECT_NORMAL);

    while (1)
    {
    /* 将获取到的图像数据，显示至 LCD */
    atk_mc2640_get_frame(   (uint32_t)&LCD->LCD_RAM,
                            ATK_MC2640_GET_TYPE_DTS_16B_NOINC,
                            demo_reset_lcd);
    }
}
```

上面的代码还是比较简单的，就是将 ATK-MC2640 模块输出的图像数据显示至 LCD。

2.2.3 实验现象

实验现象与第 2.1 小节《ATK-MC2640 模块测试实验（GPIO）》一致，请见第 2.1 小节《ATK-MC2640 模块测试实验（GPIO）》，但本实验使用 DCMI 接口读取 ATK-MC2640 模块输出的图像数据，因此帧率高了不少。

2.3 ATK-MC2640 模块测试实验（JPEG）

2.3.1 功能说明

本实验与第 2.2 小节《ATK-MC2640 模块测试实验（DCMI）》类似，只不过本实验配置 ATK-MC2640 模块输出 JPEG 图像数据，然后将通过 DCMI 接口读取到的 JPEG 图像数据通过串口输出至上位机显示。

注意：因为 STM32F1 没有 DCMI 接口，因此 STM32F1 系列开发板无法使用本实验。

2.3.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MC2640 子文件夹，该文件夹中就包含了 ATK-MC2640 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MC2640/
|-- atk_mc2640.c
|-- atk_mc2640.h
|-- atk_mc2640_cfg.h
|-- atk_mc2640_dcmi.c
|-- atk_mc2640_dcmi.h
|-- atk_mc2640_sccb.c
`-- atk_mc2640_sccb.h
```

图 2.2.2.1 ATK-MC2640 模块驱动代码

本实验的 ATK-MC2640 模块驱动代码与读 2.2 小节《ATK-MC2640 模块测试实验（DCMI）》一致。

2.3.2.1 实验测试代码

实验的测试代码为文件 demo.c，在工程目录下的 User 子目录中。测试代码的入口函数为 demo_run()，具体的代码，如下所示：

```
/**
 * @brief   例程演示入口函数
 * @param   无
 * @retval  无
 */
void demo_run(void)
{
    uint32_t *jpeg_buf;
    uint8_t *p_jpeg_buf;
    uint32_t jpeg_len;
    uint32_t jpeg_index;
```

```
/* 初始化与上位机通讯的串口 */
demo_uart_init();
/* 初始化内部 SRAM 内存池 */
my_mem_init(SRAMIN);

/* 初始化 ATK-MC2640 模块 */
atk_mc2640_init();
atk_mc2640_set_output_format(ATK_MC2640_OUTPUT_FORMAT_JPEG);
atk_mc2640_set_output_size( DEMO_JPEG_OUTPUT_WIDTH,
                            DEMO_JPEG_OUTPUT_HEIGHT);
atk_mc2640_set_light_mode(ATK_MC2640_LIGHT_MODE_SUNNY);
atk_mc2640_set_color_saturation(ATK_MC2640_COLOR_SATURATION_1);
atk_mc2640_set_brightness(ATK_MC2640_BRIGHTNESS_1);
atk_mc2640_set_contrast(ATK_MC2640_CONTRAST_2);
atk_mc2640_set_special_effect(ATK_MC2640_SPECIAL_EFFECT_NORMAL);

/* 为 JPEG 缓存空间申请内存  */
jpeg_buf = (uint32_t *)mymalloc(SRAMIN, DEMO_JPEG_BUF_SIZE);

while (1)
{
    p_jpeg_buf = (uint8_t *)jpeg_buf;
    jpeg_len = DEMO_JPEG_BUF_SIZE / (sizeof(uint32_t));
    memset((void *)jpeg_buf, 0, DEMO_JPEG_BUF_SIZE);

    /* 获取 ATK-MC2640 模块输出的一帧 JPEG 图像数据 */
    atk_mc2640_get_frame(   (uint32_t)jpeg_buf,
                            ATK_MC2640_GET_TYPE_DTS_32B_INC,
                            NULL);

    /* 获取 JPEG 图像数据的长度 */
    while (jpeg_len > 0)
    {
        if (jpeg_buf[jpeg_len - 1] != 0)
        {
            break;
        }
        jpeg_len--;
    }
    jpeg_len *= sizeof(uint32_t);

    /* 发送 JPEG 图像数据 */
    for (jpeg_index=0; jpeg_index<jpeg_len; jpeg_index++)
```

```
        {
            USART3->DR = p_jpeg_buf[jpeg_index];
            while ((USART3->SR & 0x40) == 0);
        }
    }
}
```

上面的代码还是比较简单的，就是将 ATK-MC2640 模块输出的 JPEG 图像数据读取至缓冲空间，由于 JPEG 图像数据的大小是不确定的，因此首先就要计算出 JPEG 图像数据的大小，然后将 JPEG 图像数据通过串口输出只上位机进行显示。

注意：上面的代码中，与上位机通讯的串口使用的是 USART3，使用其他串口也是可以的，因为需要传输的数据量比较大，因此这里建议使用较高的串口通讯波特率，本实验使用的串口通讯波特率为 921600bps。

2.3.3 实验现象

将 ATK-MC2640 模块按照第一节“硬件连接”中介绍的连接方式与开发板连接，同时将开发板与上位机通讯的串口连接至 PC，并将实验代码编译烧录至开发板中，如果此时开发板连接 LCD，那么 LCD 显示的内容，如下图所示：

图 2.3.3.1 LCD 显示内容一

同时，通过串口调试助手输出实验信息，如下图所示：

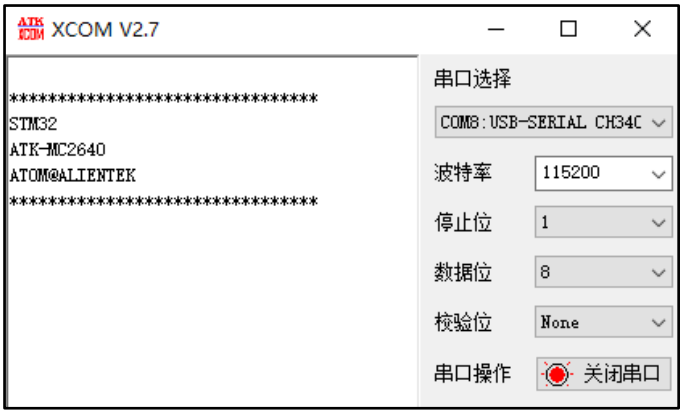

图 2.3.3.2 串口调试助手显示内容一

接下来，如果 ATK-MC2640 模块初始化成功，则会在上位机上显示 ATK-MC2640 模块输出的 JPEG 图像，如下图所示：

图 2.3.3.3 LCD 显示内容二

3，其他

1、 购买地址：

天猫：https://zhengdianyuanzi.tmall.com

淘宝：https://openedv.taobao.com

2、 资料下载

模块资料下载地址：http://www.openedv.com/docs/modules/camera/ov2640.html

3、 技术支持

公司网址：www.alientek.com

技术论坛：http://www.openedv.com/forum.php

在线教学：www.yuanzige.com

B 站视频：https://space.bilibili.com/394620890

传真：020-36773971

电话：020-38271790

正点原子