Application Note

ATK-MC5640 模块使用说明

高性能 500W 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 正点原子探索者 STM32F407 开发板

ATK-MC5640 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-MC5640 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-MC5640 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 1.1.1 ATK-MC5640 模块与探索者 STM32F407 开发板连接关系

1.2 正点原子 MiniSTM32H750 开发板

ATK-MC5640 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-MC5640 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-MC5640 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 1.2.1 ATK-MC5640 模块与 MiniSTM32H750 开发板连接关系

1.3 正点原子阿波罗 STM32F429 开发板

ATK-MC5640 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-MC5640 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-MC5640 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 1.3.1 ATK-MC5640 模块与阿波罗 STM32F429 开发板连接关系

1.4 正点原子阿波罗 STM32F767 开发板

ATK-MC5640 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-MC5640 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-MC5640 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 1.4.1 ATK-MC5640 模块与阿波罗 STM32F767 开发板连接关系

1.5 正点原子阿波罗 STM32H743 开发板

ATK-MC5640 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-MC5640 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-MC5640 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 1.5.1 ATK-MC5640 模块与阿波罗 STM32H743 开发板连接关系

2，实验功能

2.1 ATK-MC5640 模块测试实验（DCMI）

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-MC5640 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-MC5640 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。

2.1.2 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_MC5640 子文件夹，该文件夹中就包含了 ATK-MC5640 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MC5640/
|-- atk_mc5640.c
|-- atk_mc5640.h
|-- atk_mc5640_cfg.h
|-- atk_mc5640_dcmi.c
|-- atk_mc5640_dcmi.h
|-- atk_mc5640_sccb.c
`-- atk_mc5640_sccb.h
```

图 2.1.2.1 ATK-MC5640 模块驱动代码

2.1.2.1 ATK-MC5640 模块接口驱动

在图 2.1.2.1 中，atk_mc5640_sccb.c 和 atk_mc5640_sccb.h 是开发板与 ATK-MC5640 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-MC5640 模块 DCMI 接口驱动

在图 2.2.2.1 中，atk_mc5640_dcmi.c 和 atk_mc5640_dcmi.h 是开发板与 ATK-MC5640 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。

2.1.2.3 ATK-MC5640 模块驱动

在图 2.1.2.1 中，atk_mc5640.c 和 atk_mc5640.h 是 ATK-MC5640 模块的驱动文件，包含了 ATK-MC5640 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_mc5640_init()

该函数用于初始化 ATK-MC5640 模块，具体的代码，如下所示：

```
/**
 * @brief   初始化 ATK-MC5640 模块
 * @param   无
 * @retval  ATK_MC5640_EOK: ATK-MC5640 模块初始化成功
 *          ATK_MC5640_ERROR: 通讯出错，ATK-MC5640 模块初始化失败
```

```
 */
uint8_t atk_mc5640_init(void)
{
    uint16_t chip_id;

    atk_mc5640_hw_init();                /* ATK-MC5640 模块硬件初始化 */
    atk_mc5640_exit_power_down();        /* ATK-MC5640 模块退出掉电模式 */
    atk_mc5640_hw_reset();               /* ATK-MC5640 模块硬件复位 */
    atk_mc5640_sccb_init();              /* ATK-MC5640 SCCB 接口初始化 */
    atk_mc5640_sw_reset();               /* ATK-MC5640 模块软件复位 */

    chip_id = atk_mc5640_get_chip_id(); /* 获取芯片 ID */
    if (chip_id != ATK_MC5640_CHIP_ID)
    {
        return ATK_MC5640_ERROR;
    }

    atk_mc5640_init_reg();               /* 初始化 ATK-MC5640 寄存器配置 */

    atk_mc5640_dcmi_init();              /* 初始化 ATK-MC5640 模块 DCMI 接口 */

    return ATK_MC5640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc5640_init()就是通过模拟 SCCB 协议与 ATK-MC5640 模块进行通讯，首先通过 SCCB 读取 ATK-MC5640 模块寄存器中的芯片 ID 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-MC5640 模块的初始化，最后还会初始化与 ATK-MC5640 模块通讯的 DCMI 接口。

2. 函数 atk_mc5640_set_xxx()

函数 atk_mc5640_set_xxx()为一系列用于配置 ATK-MC5640 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、色相、特殊效果、曝光度、锐度、镜像/翻转和测图案，这些函数都是通过读写 ATK-MC5640 模块上摄像头传感器的寄存器来完成的，以设置 ATK-MC5640 模块输出图像对比度的函数 atk_mc5640_set_contrast()为例，其具体的代码如下所示：

```
/**
 * @brief   设置 ATK-MC5640 模块对比度
 * @param   contrast:   ATK_MC5640_CONTRAST_0   : +4
 *                      ATK_MC5640_CONTRAST_1   : +3
 *                      ATK_MC5640_CONTRAST_2   : +2
 *                      ATK_MC5640_CONTRAST_3   : +1
 *                      ATK_MC5640_CONTRAST_4   : 0
 *                      ATK_MC5640_CONTRAST_5   : -1
 *                      ATK_MC5640_CONTRAST_6   : -2
 *                      ATK_MC5640_CONTRAST_7   : -3
 *                      ATK_MC5640_CONTRAST_8   : -4
```

```
 * @retval  ATK_MC5640_EOK      : 设置 ATK-MC5640 模块对比度成功
 *          ATK_MC5640_EINVAL   : 传入参数错误
 */
uint8_t atk_mc5640_set_contrast(atk_mc5640_contrast_t contrast)
{
    switch (contrast)
    {
        case ATK_MC5640_CONTRAST_0:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x30);
            atk_mc5640_write_reg(0x5585, 0x30);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_1:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x2C);
            atk_mc5640_write_reg(0x5585, 0x2C);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_2:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x28);
            atk_mc5640_write_reg(0x5585, 0x28);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_3:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x24);
            atk_mc5640_write_reg(0x5585, 0x24);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_4:
```

```
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x20);
            atk_mc5640_write_reg(0x5585, 0x20);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_5:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x1C);
            atk_mc5640_write_reg(0x5585, 0x1C);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_6:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x18);
            atk_mc5640_write_reg(0x5585, 0x18);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_7:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x14);
            atk_mc5640_write_reg(0x5585, 0x14);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
        case ATK_MC5640_CONTRAST_8:
        {
            atk_mc5640_write_reg(0x5001, 0xFF);
            atk_mc5640_write_reg(0x5580, 0x04);
            atk_mc5640_write_reg(0x5586, 0x10);
            atk_mc5640_write_reg(0x5585, 0x10);
            atk_mc5640_write_reg(0x5588, 0x41);
            break;
        }
```

```
        default:
        {
            return ATK_MC5640_EINVAL;
        }
    }

    return ATK_MC5640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc5640_set_contrast()就是通过往 ATK-MC5640 模块内存的各个寄存器写入不同的值，以完成配置 ATK-MC5640 模块输出图像对比度的操作。

3. 函数 atk_mc5640_read_reg()和函数 atk_mc5640_write_reg()

这两个函数该函数用于读写 ATK-MC5640 模块的寄存器，具体的代码如下所示：

```
/**
 * @brief   ATK-MC5640 模块写寄存器
 * @param   reg: 寄存器地址
 *          dat: 待写入的值
 * @retval  无
 */
static void atk_mc5640_write_reg(uint16_t reg, uint8_t dat)
{
    atk_mc5640_sccb_3_phase_write(ATK_MC5640_SCCB_ADDR, reg, dat);
}

/**
 * @brief   ATK-MC5640 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_mc5640_read_reg(uint16_t reg)
{
    uint8_t dat = 0;

    atk_mc5640_sccb_2_phase_write(ATK_MC5640_SCCB_ADDR, reg);
    atk_mc5640_sccb_2_phase_read(ATK_MC5640_SCCB_ADDR, &dat);

    return dat;
}
```

从上面的代码中可以看出，读写 ATK-MC5640 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_mc5640_get_frame()

该函数用于获取 ATK-MC5640 模块输出的一帧图像，具体的代码图下所示：

```
/**
 * @brief   获取 ATK-MC5640 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_MC5640_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_MC5640_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_MC5640_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_MC5640_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_MC5640_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_MC5640_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_MC5640_EOK      : 获取 ATK-MC5640 模块输出的一帧图像数据成功
 *          ATK_MC5640_EINVAL   : 传入参数错误
 */
uint8_t atk_mc5640_get_frame(  uint32_t dts_addr,
                               atk_mc5640_get_type_t type,
                               void (*before_transfer)(void))
{
    uint32_t meminc;
    uint32_t memdataalignment;
    uint32_t len;

    switch (type)
    {
        case ATK_MC5640_GET_TYPE_DTS_8B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint8_t);
            break;
        }
        case ATK_MC5640_GET_TYPE_DTS_8B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_BYTE;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint8_t);
            break;
        }
```

```
        case ATK_MC5640_GET_TYPE_DTS_16B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint16_t);
            break;
        }
        case ATK_MC5640_GET_TYPE_DTS_16B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_HALFWORD;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint16_t);
            break;
        }
        case ATK_MC5640_GET_TYPE_DTS_32B_NOINC:
        {
            meminc = DMA_MINC_DISABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint32_t);
            break;
        }
        case ATK_MC5640_GET_TYPE_DTS_32B_INC:
        {
            meminc = DMA_MINC_ENABLE;
            memdataalignment = DMA_MDATAALIGN_WORD;
            len =   (   g_atk_mc5640_sta.output.width *
                        g_atk_mc5640_sta.output.height) /
                    sizeof(uint32_t);
            break;
        }
        default:
        {
            return ATK_MC5640_EINVAL;
        }
    }

    if (before_transfer != NULL)
    {
```

```
        before_transfer();
    }

    atk_mc5640_dcmi_start(dts_addr, meminc, memdataalignment, len);

    return ATK_MC5640_EOK;
}
```

从上面的代码中可以看出，函数 atk_mc5640_get_frame()就是通过 DCMI 接口获取 ATK-MC5640 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。

2.1.2.4 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 ATK-MC5640 模块 */
    ret  = atk_mc5640_init();
    /* 设置 ATK-MC5640 输出 RGB565 图像数据 */
    ret += atk_mc5640_set_output_format(ATK_MC5640_OUTPUT_FORMAT_RGB565);
    /* 初始化 ATK-MC5640 模块自动对焦 */
    ret += atk_mc5640_auto_focus_init();
    /* ATK-MC5640 模块持续自动对焦 */
    ret += atk_mc5640_auto_focus_continuance();
    /* 设置 ATK-MC5640 模块灯光模式 */
    ret += atk_mc5640_set_light_mode(ATK_MC5640_LIGHT_MODE_ADVANCED_AWB);
    /* 设置 ATK-MC5640 模块色彩饱度 */
    ret += atk_mc5640_set_color_saturation(ATK_MC5640_COLOR_SATURATION_4);
    /* 设置 ATK-MC5640 模块亮度 */
    ret += atk_mc5640_set_brightness(ATK_MC5640_BRIGHTNESS_4);
    /* 设置 ATK-MC5640 模块对比度 */
    ret += atk_mc5640_set_contrast(ATK_MC5640_CONTRAST_4);
    /* 设置 ATK-MC5640 模块色相 */
    ret += atk_mc5640_set_hue(ATK_MC5640_HUE_6);
    /* 设置 ATK-MC5640 模块特殊效果 */
    ret += atk_mc5640_set_special_effect(ATK_MC5640_SPECIAL_EFFECT_NORMAL);
```

```
    /* 设置 ATK-MC5640 模块曝光度 */
    ret += atk_mc5640_set_exposure_level(ATK_MC5640_EXPOSURE_LEVEL_5);
    /* 设置 ATK-MC5640 模块锐度 */
    ret += atk_mc5640_set_sharpness_level(ATK_MC5640_SHARPNESS_OFF);
    /* 设置 ATK-MC5640 模块镜像/翻转 */
    ret += atk_mc5640_set_mirror_flip(ATK_MC5640_MIRROR_FLIP_1);
    /* 设置 ATK-MC5640 模块测试图案 */
    ret += atk_mc5640_set_test_pattern(ATK_MC5640_TEST_PATTERN_OFF);
    /* 设置 ATK-MC5640 模块输出图像尺寸 */
    ret += atk_mc5640_set_output_size(lcddev.width, lcddev.height);
    if (ret != 0)
    {
        printf("ATK-MC5640 init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

    while (1)
    {
        /* 将获取到的图像数据，显示至 LCD */
        atk_mc5640_get_frame(   (uint32_t)&LCD->LCD_RAM,
                                ATK_MC5640_GET_TYPE_DTS_16B_NOINC,
                                demo_reset_lcd);
    }
}
```

上面的代码还是比较简单的，首先就是配置 ATK-MC5640 模块的各种参数，然后将 ATK-MC5640 模块输出的图像数据显示至 LCD。

2.1.3 实验现象

将 ATK-MC5640 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：

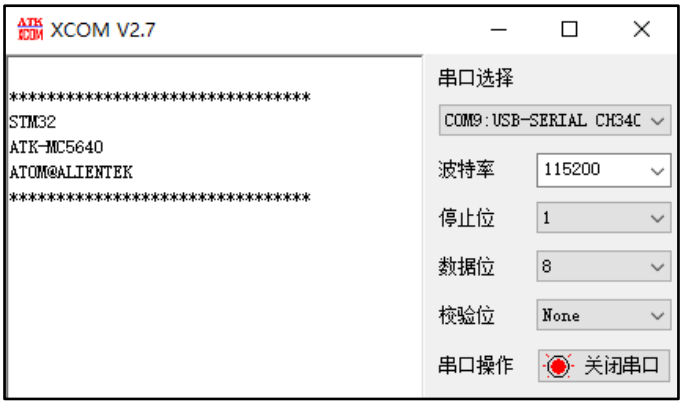

图 2.1.3.2 串口调试助手显示内容一

接下来，如果 ATK-MC5640 模块初始化成功，则会在 LCD 上显示 ATK-MC5640 模块输出的图像，如下图所示：

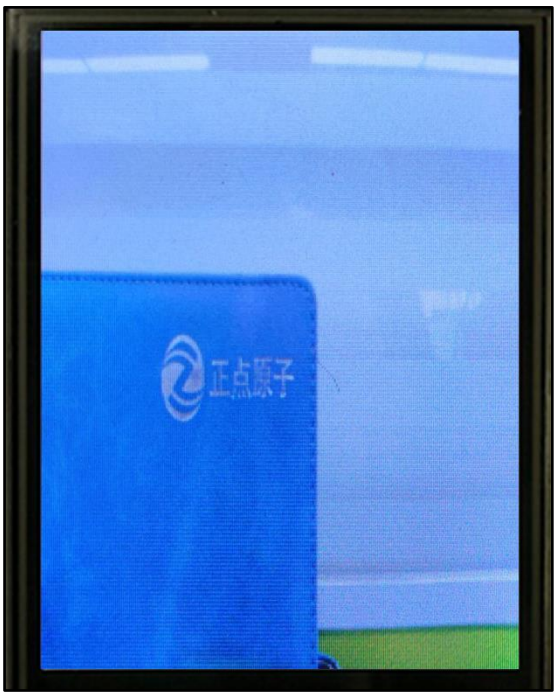

图 2.1.3.3 LCD 显示内容二

2.2 ATK-MC5640 模块测试实验（JPEG）

2.2.1 功能说明

本实验与第 2.2 小节《ATK-MC5640 模块测试实验（DCMI）》类似，只不过本实验配置 ATK-MC5640 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。

2.2.2 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_MC5640 子文件夹，该文件夹中就包含了 ATK-MC5640 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MC5640/
|-- atk_mc5640.c
|-- atk_mc5640.h
|-- atk_mc5640_cfg.h
|-- atk_mc5640_dcmi.c
|-- atk_mc5640_dcmi.h
|-- atk_mc5640_sccb.c
`-- atk_mc5640_sccb.h
```

图 2.2.2.1 ATK-MC5640 模块驱动代码

本实验的 ATK-MC5640 模块驱动代码与读 2.1 小节《ATK-MC5640 模块测试实验（DCMI）》一致。

2.2.2.1 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)
{
    uint8_t ret;
    uint8_t *p_jpeg_buf;
    uint32_t jpeg_len;
    uint32_t jpeg_index;
    uint32_t jpeg_start_index;
    uint32_t jpeg_end_index;

    /* 初始化与上位机通讯的串口 */
    demo_uart_init();

    /* 初始化 ATK-MC5640 模块 */
    ret  = atk_mc5640_init();
    ret += atk_mc5640_set_output_format(ATK_MC5640_OUTPUT_FORMAT_JPEG);
    ret += atk_mc5640_auto_focus_init();
    ret += atk_mc5640_auto_focus_continuance();
    ret += atk_mc5640_set_light_mode(ATK_MC5640_LIGHT_MODE_ADVANCED_AWB);
    ret += atk_mc5640_set_color_saturation(ATK_MC5640_COLOR_SATURATION_4);
    ret += atk_mc5640_set_brightness(ATK_MC5640_BRIGHTNESS_4);
    ret += atk_mc5640_set_contrast(ATK_MC5640_CONTRAST_4);
    ret += atk_mc5640_set_hue(ATK_MC5640_HUE_6);
    ret += atk_mc5640_set_special_effect(ATK_MC5640_SPECIAL_EFFECT_NORMAL);
    ret += atk_mc5640_set_exposure_level(ATK_MC5640_EXPOSURE_LEVEL_5);
    ret += atk_mc5640_set_sharpness_level(ATK_MC5640_SHARPNESS_OFF);
```

```
    ret += atk_mc5640_set_mirror_flip(ATK_MC5640_MIRROR_FLIP_1);
    ret += atk_mc5640_set_test_pattern(ATK_MC5640_TEST_PATTERN_OFF);
    ret += atk_mc5640_set_pre_scaling_window(4, 0);
    ret += atk_mc5640_set_output_size(  DEMO_JPEG_OUTPUT_WIDTH,
                                        DEMO_JPEG_OUTPUT_HEIGHT);
    if (ret != 0)
    {
        printf("ATK-MC5640 init failed!\r\n");
        while (1)
        {
            LED0_TOGGLE();
            delay_ms(200);
        }
    }

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

        /* 获取 ATK-MC5640 模块输出的一帧 JPEG 图像数据 */
        atk_mc5640_get_frame(   (uint32_t)g_jpeg_buf,
                                ATK_MC5640_GET_TYPE_DTS_32B_INC,
                                NULL);

        /* 获取 JPEG 图像数据起始位置 */
        for (   jpeg_start_index=UINT32_MAX, jpeg_index=0;
                jpeg_index<DEMO_JPEG_BUF_SIZE - 1;
                jpeg_index++)
        {
            if (    (p_jpeg_buf[jpeg_index] == 0xFF) &&
                    (p_jpeg_buf[jpeg_index + 1] == 0xD8))
            {
                jpeg_start_index = jpeg_index;
                break;
            }
        }
        if (jpeg_start_index == UINT32_MAX)
        {
            continue;
        }

        /* 获取 JPEG 图像数据结束位置 */
```

```
        for (   jpeg_end_index=UINT32_MAX, jpeg_index=jpeg_start_index;
                jpeg_index<DEMO_JPEG_BUF_SIZE - 1;
                jpeg_index++)
        {
            if (    (p_jpeg_buf[jpeg_index] == 0xFF) &&
                    (p_jpeg_buf[jpeg_index + 1] == 0xD9))
            {
                jpeg_end_index = jpeg_index;
                break;
            }
        }
        if (jpeg_end_index == UINT32_MAX)
        {
            continue;
        }

        /* 获取 JPEG 图像数据的长度 */
        jpeg_len = jpeg_end_index - jpeg_start_index + (sizeof(uint32_t) >> 1);

        /* 发送 JPEG 图像数据 */
        for (jpeg_index=jpeg_start_index; 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-MC5640 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至上位机进行显示。

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。

2.2.3 实验现象

将 ATK-MC5640 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 2.2.3.1 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：

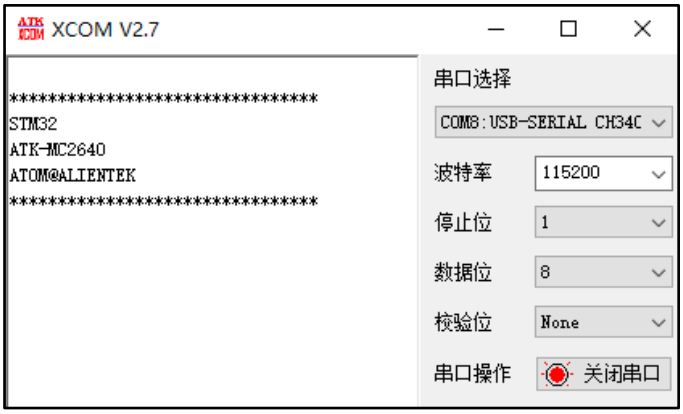

图 2.2.3.2 串口调试助手显示内容一

接下来，如果 ATK-MC5640 模块初始化成功，则会在上位机上显示 ATK-MC5640 模块输出的 JPEG 图像，如下图所示：

图 2.2.3.3 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/ov5640.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