AN1704 ALIENTEK 摄像头模块使用

本应用文档（AN1704，对应 **ALIENTEK MiniSTM32 开发板（V3.0）扩展实验 9**）将教大家如何在 ALIENTEK MiniSTM32 开发板上使用 ALIENTEK OV7725 和 OV7670 摄像头模块。

本文档分为如下几部分：

1， OV7725 简介
2， OV7670 简介
3， 硬件连接
4， 软件实现
5， 验证

1、OV7725 简介

OV7725 是 OV（OmniVision）公司生产的一颗 1/4 寸的 CMOS VGA 图像传感器。该传感器体积小、工作电压低（典型电压 3.3V），提供单片 VGA 摄像头和影像处理器的所有功能。通过 SCCB 总线控制，可以输出整帧、子采样、取窗口等方式的各种分辨率，10 位或 8 位影像数据输出。该产品 VGA 图像输出最高可达 60 帧/秒。用户可以完全控制图像质量、数据格式和传输方式。所有图像处理功能过程包括伽玛曲线、白平衡、度、色度等都可以通过 SCCB 接口编程。OmmiVision 图像传感器应用独有的传感器技术，通过减少或消除光学或电子缺陷如固定图案噪声、托尾、浮散等，提高图像质量，得到清晰的稳定的彩色图像。

OV7725 的特点有：

- 高灵敏度、低电压适合嵌入式应用
- 标准的 SCCB 接口，兼容 IIC 接口
- 支持 RawRGB、RGB(GBR4:2:2，RGB565/RGB555/RGB444)，YUV(4:2:2)和 YCbCr（4:2:2）输出格式
- 支持 VGA、QVGA，和从 CIF 到 40*30 的各种尺寸输出
- 支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹、自动黑电平校准等自动控制功能。同时支持色饱和度、色相、伽马、锐度等设置。
- 支持图像缩放

OV7725 的功能框图如图 1.1 所示：

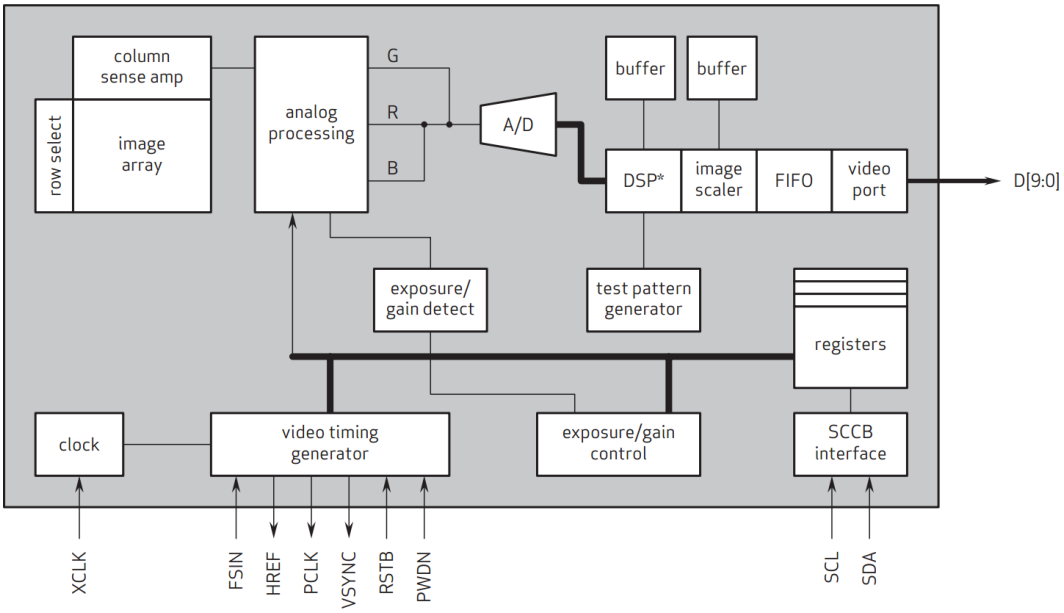

图 1.1 OV7725 功能框图

OV7725 传感器包括如下一些功能模块。

1.感光整列（Image Array）

OV7725 总共有 656*488 个像素，其中 640*480 个有效（即有效像素为 30W）。

2.时序发生器（Video Timing Generator）

时序发生器具有的功能包括：整列控制和帧率发生（7 种不同格式输出）、内部信号发生器和分布、帧率时序、自动曝光控制、输出外部时序（VSYNC、HREF/HSYNC 和 PCLK）。

3.模拟信号处理（Analog Processing）

模拟信号处理所有模拟功能，并包括：自动增益（AGC）和自动白平衡（AWB）。

4.A/D 转换（A/D）

原始的信号经过模拟处理器模块之后 ，分 G 和 BR 两路进入一个 10 位的 A/D 转换器，A/D 转换器工作在 12M 频率，与像素频率完全同步（转换的频率和帧率有关）。

除 A/D 转换器外，该模块还有以下三个功能：

- 黑电平校正（BLC)
- U/V 通道延迟
- A/D 范围控制

A/D 范围乘积和 A/D 的范围控制共同设置 A/D 的范围和最大值，允许用户根据应用调整图片的亮度。

5.测试图案发生器（Test Pattern Generator）

测试图案发生器功能包括：八色彩色条图案、渐变至黑白彩色条图案和输出脚移位“1”。

6.数字处理器（DSP）

这个部分控制由原始信号插值到 RGB 信号的过程，并控制一些图像质量：

- 边缘锐化（二维高通滤波器）
- 颜色空间转换（ 原始信号到 RGB 或者 YUV/YCbYCr)
- RGB 色彩矩阵以消除串扰
- 色相和饱和度的控制
- 可编程的伽玛
- 十位到八位数据转换

7.缩放功能（Image Scaler）

这个模块按照预先设置的要求输出数据格式，能将 YUV/RGB 信号从 VGA 缩小到 CIF 以下的任何尺寸。

8.数字视频接口（Digital Video Port）

通过寄存器 COM2[1:0]，调节 IOL/IOH 的驱动电流，以适应用户的负载。

9.SCCB 接口（SCCB Interface）

SCCB 接口控制图像传感器芯片的运行，详细使用方法参照光盘的《OmniVision Technologies Seril Camera Control Bus(SCCB) Specification》这个文档

OV7725 的寄存器通过 SCCB 时序访问并设置，SCCB 时序和 IIC 时序十分类似，在本章我们不做介绍，请大家参考光盘的 SCCB 相关文档。接下来我们介绍一下 OV7725 的图像数据输出格式及时序分析。首先我们简单介绍几个定义：

VGA，即分辨率为 640*480 的输出模式；

QVGA，即分辨率为 320*240 的输出格式；

QQVGA，即分辨率为 160*120 的输出格式；

PCLK，即像素时钟，一个 PCLK 时钟，输出一个像素(或半个像素)。

VSYNC，即帧同步信号。

HREF /HSYNC，即行同步信号。

OV7725 时序分析：

我们的 LCD 数据格式为 RGB565，下面以 OV7725 输出 RGB565 模式分析一下时序。OV7725 输出时序图如下图 1.2 所示：

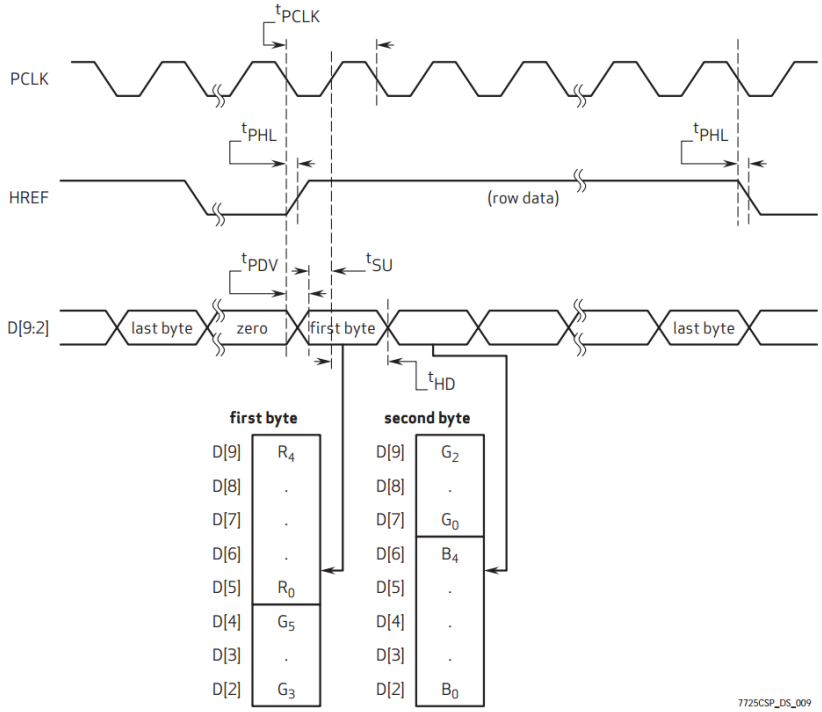

图 1.2 OV7725 RGB565 输出时序

从上图可看出，OV7725 的图像数据通过 D[9:2]输出一个字节，first byte 和 second byte 组成一个 16 位 RGB565 数据。时序上，HREF 为高时开始传输一行数据，1 个 PCLK 传输 1 个字节，传输完一行数据最后一个字节（last byte）后 HREF 则变为低。

再来看看 OV7725 帧时序图（VGA 模式），如下图 1.3 所示：

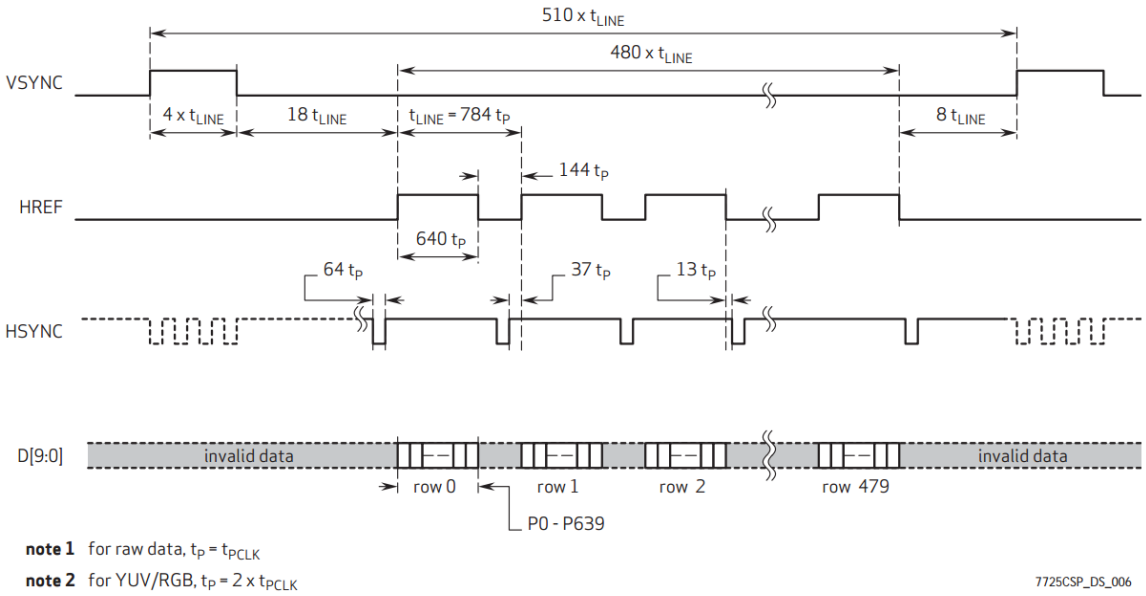

图 1.3 OV7725 帧时序

从上图可看出，1 个 HREF 周期由 640t_p 高电平和低电平 144t_p 组成。对于 YUV/RGB 模式，t_P=2×t_{PCLK} 即一个 t_{PCLK} 对应传输一个字节（RGB565 格式传输一行数据的时间 T=640×2t_{PCLK}）。其中 144t_p 是传输一行数据的间隔时间。当传输了 480 个 HREF 周期（480×t_{LINE}）后刚好完成一个 VSYNC（帧）数据传输，等 8 t_{LINE} 后会产生一个 VSYNC 上升沿表示一帧数据传输完成。程序中我们就可以根据 VSYNC 上升沿来判断一帧图像数据传输完成。注意，图中的 HSYNC 和 HREF 其实是同一个引脚产生的信号，只是在不同场合下面，使用不同的信号方式，我们本章用到的是 HREF。

因为 OV7725 的像素时钟（PCLK）最高可达 24Mhz，我们用 STM32F103RCT6 的 IO 口直接抓取，是非常困难的，也十分占耗 CPU（可以通过降低 PCLK 输出频率，来实现 IO 口抓取，但是不推荐）。所以，本章我们并不是采取直接抓取来自 OV7725 的数据，而是通过 FIFO 读取，ALIENTEK OV7725 摄像头模块自带了一个 FIFO 芯片（AL422B），用于暂存图像数据，有了这个芯片，我们就可以很方便的获取图像数据了，而不再需要单片机具有高速 IO，也不会耗费多少 CPU，可以说，只要是个单片机，都可以通过 ALIENTEK OV7725 摄像头模块实现拍照的功能。

接下来我们介绍一下 ALIENTEK OV7725 摄像头模块。OV7725 模块的外观如图 1.4：

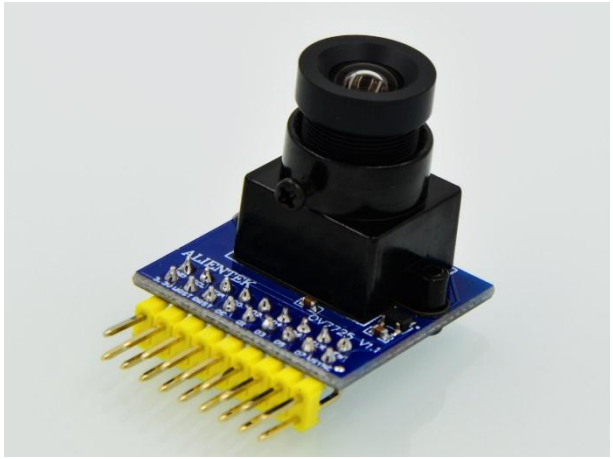

图 1.4 ALIENTEK OV7725 摄像头模块外观图

OV7725 模块原理图如下图 1.5 所示：

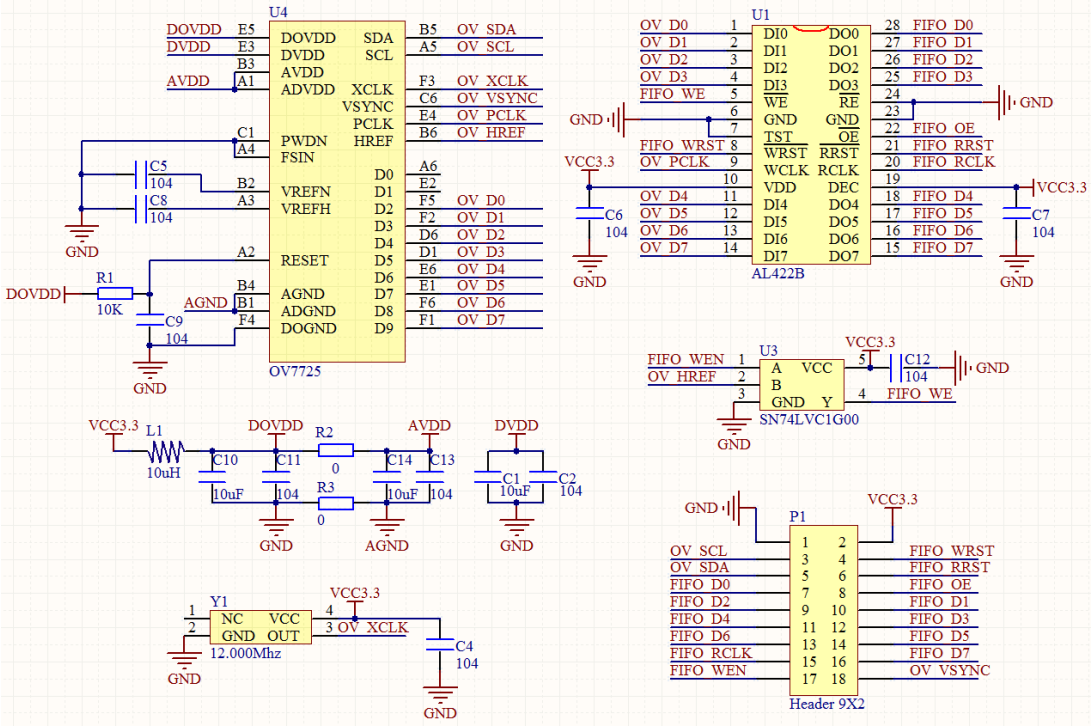

图 1.5 ALIENTEK OV7725 摄像头模块原理图

从上图可以看出，ALIENTEK OV7725 摄像头模块自带了有源晶振，用于产生 12M 时钟作为 OV7725 传感器的 XCLK 输入；带有一个 FIFO 芯片（AL422B），该 FIFO 芯片的容量是 384K 字节，足够存储 2 帧 QVGA 的图像数据。模块通过一个 2*9 的双排排针（P1）与外部通信，与外部的通信信号如表 1.1 所示：

信号	作用描述	信号	作用描述
VCC3.3	模块供电脚，接 3.3V 电源	FIFO_WEN	FIFO 写使能
GND	模块地线	FIFO_WRST	FIFO 写指针复位
OV_SCL	SCCB 通信时钟信号	FIFO_RRST	FIFO 读指针复位
OV_SDA	SCCB 通信数据信号	FIFO_OE	FIFO 输出使能（片选）
FIFO_D[7:0]	FIFO 输出数据（8 位）	OV_VSYNC	帧同步信号
FIFO_RCLK	读 FIFO 时钟		

表 1.1 P1 接口信号描述

下面我们来看看如何使用 ALIENTEK OV7725 摄像头模块（以 QVGA 模式，RGB565 格式为例）。对于该模块，我们只关心两点：1，如何存储图像数据；2，如何读取图像数据。

首先，我们来看如何存储图像数据。

摄像头模块存储图像数据的过程为：等待 OV7725 帧同步信号→FIFO 写指针复位→FIFO 写使能→等待第二个 OV7725 帧同步信号→FIFO 写禁止。通过以上 5 个步骤，我们就可以完成 1 帧图像数据在 AL422B 的存储。注意：FIFO 写禁止操作不是必须的，只有当你想将一帧图片数据存储在 FIFO，并在外部 MCU 读取完这帧图片数据之前，不再采集新的图片数据的时候，才需要进行 FIFO 写禁止。

接下来，我们来看看如何读取图像数据。

在存储完一帧图像以后，我们就可以开始读取图像数据了。读取过程为：FIFO 读指针复位→给 FIFO 读时钟（FIFO_RCLK）→读取第一个像素高字节→给 FIFO 读时钟→读取第一个像素低字节→给 FIFO 读时钟→读取第二个像素高字节→循环读取剩余像素→结束。

可以看出，摄像头模块数据的读取也是十分简单，比如 QVGA 模式，RGB565 格式，

我们总共循环读取 320*240*2 次，就可以读取 1 帧图像数据，把这些数据写入 LCD 模块，我们就可以看到摄像头捕捉到的画面了。

了解了数据存储和读取，我们就可以开始设计代码了，本章，我们用一个外部中断，来捕捉帧同步信号（VSYNC），然后在中断里面启动图像数据存储，等待下一次 VSYNC 信号到来，我们就关闭数据存储，然后一帧数据就存储完成了，在主函数里面就可以慢慢的将这一帧数据读出来，放到 LCD 即可显示了，同时开始第二帧数据的存储，如此循环，实现摄像头功能。

本章，我们将使用 ALIENTEK OV7725 摄像头模块的以 QVGA 模式（分辨率 320*240），直接输出显示到我们 LCD 上。注意：摄像头模块自带的 FIFO（AL422B）是没办法缓存一帧的 VGA 图像的，如果使用 VGA 全屏分辨率输出，那么你必须在 FIFO 写满之前开始读 FIFO 数据，保证数据不被覆盖。OV7725 还可以对输出图像进行各种设置，数据手册和应用笔记详见光盘《OV7725_datasheet.pdf》和《OV7725 Software Application Note.pdf》。对 AL422B 的操作时序，请大家参考 AL422B 的数据手册。

2、OV7670 简介

OV7670 是 OV（OmniVision）公司生产的一颗 1/6 寸的 CMOS VGA 图像传感器。OV760 功能及特点和 OV7725 非常类似，相同之处这里不细说了，主要不同之处是 OV7670 典型工作电压为 2.8V、VGA 图像输出最高为 30 帧/秒、图像数据输出只有 8 位 D[7:0]。下面我们分析 OV7670 的时序。

OV7670 时序分析：

OV7670 的图像数据输出（通过 D[7:0]）就是在 PCLK，VSYNC 和 HREF/ HSYNC 的控制下进行的。首先看看行输出时序，如图 2.1 所示：

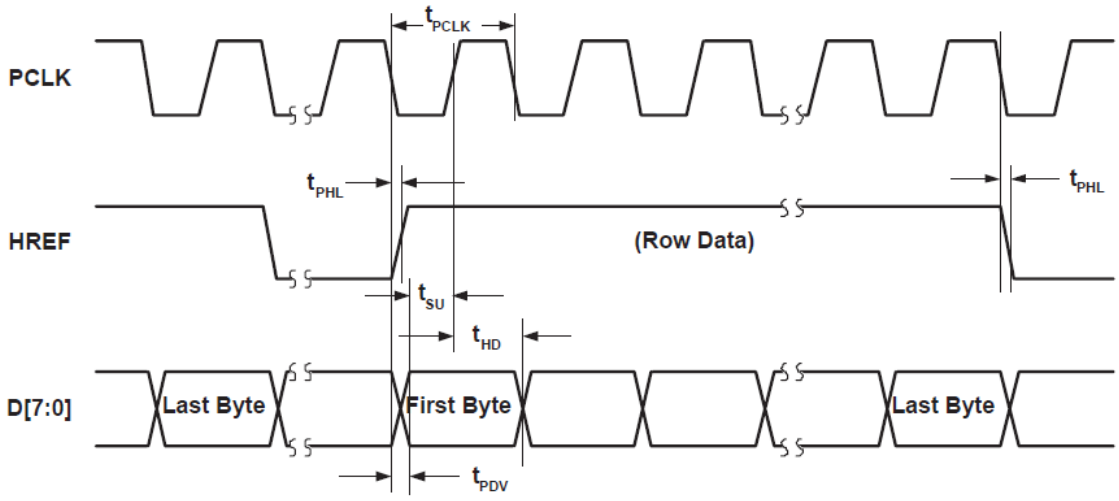

图 2.1 OV7670 行输出时序

从上图可以看出，图像数据在 HREF 为高的时候输出，当 HREF 变高后，每一个 PCLK 时钟，输出一个字节数据。比如我们采用 VGA 时序，RGB565 格式输出，每 2 个字节组成一个像素的颜色（高字节在前，低字节在后），这样每行输出总共有 640*2 个 PCLK 周期，输出 640*2 个字节。

再来看看帧时序（VGA 模式），如图 2.2 所示：

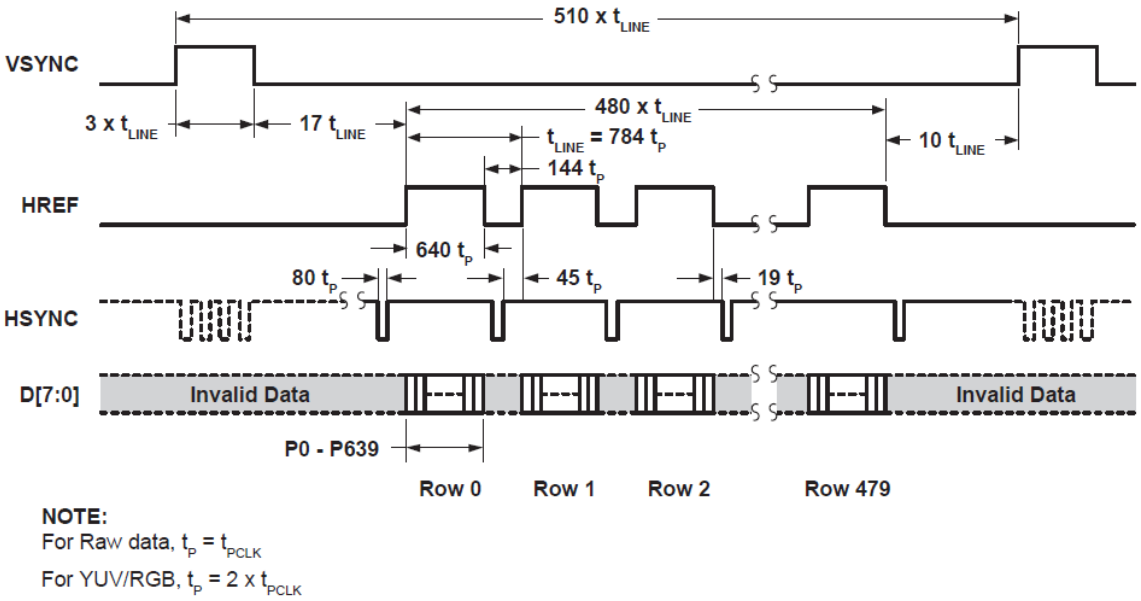

图 2.2 OV7670 帧时序

OV7670 帧时序和 OV7725 帧时序非常类似，不同之处就是 VSYNC 高电平时间和数据传输完成等待 VSYNC 上升沿时间，另外 OV7670 数据输出是 D[7:0]。注意，图中的 HSYNC 和 HREF 其实是同一个引脚产生的信号，只是在不同场合下面，使用不同的信号方式，我们本章用到的是 HREF。

接下来我们介绍一下 ALIENTEK OV7670 摄像头模块。OV7670 模块的外观如图 2.3：

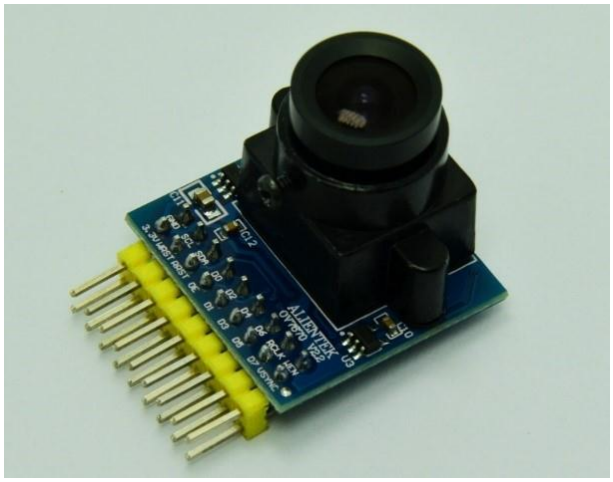

图 2.3 ALIENTEK OV7670 摄像头模块外观图

OV7670 摄像头模块原理图如图 2.4

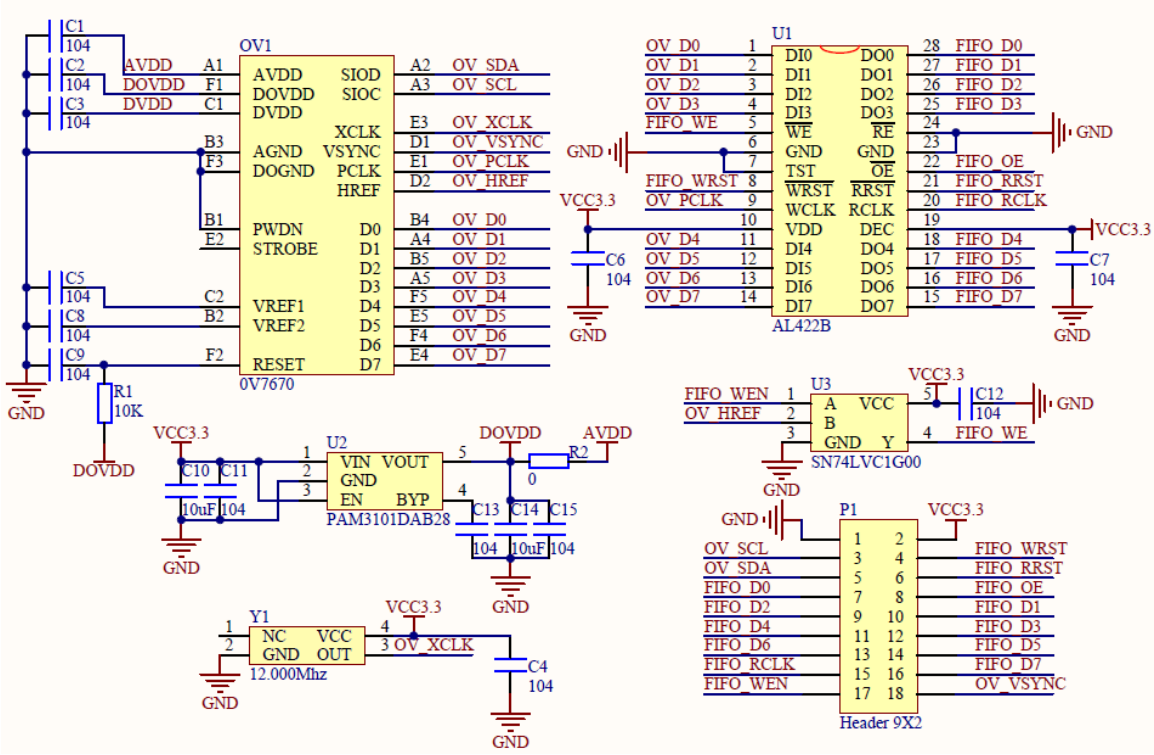

图 2.4 ALIENTEK OV7670 摄像头模块原理图

从上图可看出 OV7670 模块原理图和 OV7725 模块原理图有很大相同之处，不同之处是：

1、OV7670 带了 2.8V 稳压芯片，OV7670 典型工作电压为 2.8V，而 OV7725 典型工作电压为 3.3V。

2、OV7670 传感器数据位为 D[7:0]，而 OV7725 有效数据位为 D[9:2]。

如何存储 OV7670 图像数据至 FIFO 和从 FIFO 读取图像数据和前面 OV7725 的操作是一样的，这里就不赘述了。

本例程实验程序既支持 OV7725 又支持 OV7670，对于 OV7670，我们以 QVGA 模式（分辨率 320*240）直接输出显示到 LCD 上。OV7670 还可以对输出图像进行各种设置，详见光盘《OV7670 中文数据手册 1.01》和《OV7670 software application note》这两个文档。

3、硬件连接

本实验将实现如下功能：开机后，初始化摄像头模块（OV7725 或 OV7670），如果初始化成功，则在 LCD 模块上面显示摄像头模块所拍摄到的内容，实现摄像头驱动。通过串口，我们可以查看当前的帧率（这里是指 LCD 显示的帧率，而不是指 OV7725 和 OV7670 的输出帧率），同时可以借助 USMART 设置 OV7725 和 OV7670 的寄存器，方便大家调试。DS0 指示程序运行状态。

通过前面的介绍，我们知道，摄像头模块与 MCU 连接需要 16 根信号线，以及 2 根电源线。这 16 根信号线与 ALIENTEK MiniSTM32 开发板的连接关系如表 3.1 所示：

摄像头模块与开发板连接关系									
OV7670 摄像头模块	D0~D7	SCL	SDA	WRST	RCLK	RRST	OE	WEN	VSYNC
MiniSTM32 开发板	PB0~PB7	PC4	PC5	PA0	PA1	PA4	PA11	PA12	PA15

表 3.1 摄像头模块与开发板连接关系

从上表可以看出，我们的 PB0~PB7 用与连接摄像头的数据线，同时这几个 IO 口还连接了 LCD 模块，也就是 PB0~PB7 由 LCD 和摄像头模块分时复用。其他信号线的连接，就比较简单了，我们用杜邦线一一连接即可。

最后，我们连接摄像头模块的电源到开发板上即可。这里需要比较多的杜邦线（总共 18 根），大家得事先准备好。

4、软件实现

本例程我们在 ALIENTEK MiniSTM32 开发板 V3.0 标准例程（寄存器版）USMART 调试组件实验的基础上进行修改。

首先，在 HARDWARE 文件夹下新建 OV7725 和 OV7670 的文件夹。OV7725 文件夹如下文件：ov7725.c、ov7725.h、ov7725cfg.h、sccb.c、sccb.h 等 5 个文件。OV7670 文件夹如下文件：ov7670.c、ov7670.h、ov7670cfg.h、sccb.c、sccb.h 等 5 个文件。其中两个文件的 sccb.c 和 sccb.h 是一样的，程序工程中只添加一个 sccb.c 即可，另外将这两个文件夹加入头文件包含路径。另外，我们还需要用到 exti.c 和 timer.c，所以把这两个文件也加入 HARDWARE 组下。

本例程新增了多个文件，代码比较多，我们就不一一列出了，而且 ov7725.c 和 ov7670.c 有很多相同之处，这里我们仅以 OV7725 为例，并挑两个重要的地方进行讲解。首先，我们来看 ov7725.c 里面的 OV7725_Init 函数，该函数代码如下：

```
u8 OV7725_Init(void)
{
    u16 i=0;
    u16 reg=0;
    //设置 IO
    RCC->APB2ENR|=1<<2;          //先使能外设 PORTA 时钟
    RCC->APB2ENR|=1<<3;          //先使能外设 PORTB 时钟

    GPIOA->CRL&=0XFFF0FF00;
    GPIOA->CRL|=0X00030033;      //PA0/1/4  输出
    GPIOA->ODR|=1<<4;
    GPIOA->ODR|=3<<0;

    GPIOA->CRH&=0X00F00FFF;
    GPIOA->CRH|=0X83033000;      //PA15  输入、PA11/12/14 输出
    GPIOA->ODR|=3<<14;
    GPIOA->ODR|=3<<11;

    JTAG_Set(SWD_ENABLE);
    SCCB_Init();                 //初始化 SCCB  的 IO 口
    if(SCCB_WR_Reg(0x12,0x80))return 1;     //复位 SCCB
    delay_ms(50);
    reg=SCCB_RD_Reg(0X1c);       //读取厂家 ID  高八位
    reg<<=8;
    reg|=SCCB_RD_Reg(0X1d);      //读取厂家 ID  低八位
    if(reg!=OV7725_MID)
```

```
        {
            printf("MID:%d\r\n",reg);
            return 1;
        }
        reg=SCCB_RD_Reg(0X0a);          //读取厂家 ID 高八位
        reg<<=8;
        reg|=SCCB_RD_Reg(0X0b);         //读取厂家 ID 低八位
        if(reg!=OV7725_PID)
        {
            printf("HID:%d\r\n",reg);
            return 2;
        }
        //初始化 OV7725,采用 QVGA 分辨率(320*240)
        for(i=0;i<sizeof(ov7725_init_reg_tb1)/sizeof(ov7725_init_reg_tb1[0]);i++)
        {
            SCCB_WR_Reg(ov7725_init_reg_tb1[i][0],ov7725_init_reg_tb1[i][1]);
        }
        return 0x00;    //ok
    }
```

此部分代码先初始化 OV7725 相关的 IO 口（包括 SCCB_Init），然后最主要的是完成 OV7725 的寄存器序列初始化。OV7725 的寄存器特多（百几十个），配置特麻烦，幸好厂家有提供参考配置序列（详见《OV7725 software application note》），本章我们用到的配置序列，存放在 ov7725_init_reg_tbl 这个数组里面，该数组是一个 2 维数组，存储初始化序列寄存器及其对应的值，该数组存放在 ov7725cfg.h 里面。

接下来，我们看看 ov7725cfg.h 里面 ov7725_init_reg_tbl 的内容，ov7725cfg.h 文件的代码如下：

```
//初始化寄存器序列及其对应的值
const u8 ov7725_init_reg_tbl[][2]=
{
    /*输出窗口设置*/
    {CLKRC,     0x00}, //clock config
    {COM7,      0x46}, //QVGA RGB565
    {HSTART,    0x3f}, //水平起始位置
    ……省略部分设置
    {COM3,      0x10},
    {COM5,      0xf5},
};
```

以上代码，我们省略了很多（全部贴出来太长了），我们大概了解下结构，每个条目的第一个字节为寄存器号（也就是寄存器地址），第二个字节为要设置的值，比如{CLKRC, 0x00}，就表示在 CLKRC（地址宏定义）地址，写入 0X00 这个值。

通过这么一长串寄存器的配置，我们就完成了 OV7725 的初始化，本章我们配置 OV7725 工作在 QVGA 模式，RGB565 格式输出。 在完成初始化之后，我们既可以开始读取 OV7725 的数据了。

OV7725 文件夹里面的其他代码我们就不逐个介绍了，请大家参考本例程源码。

因为我们还用到了帧率（LCD 显示的帧率）统计和中断处理，所以我们还需要修改 timer.c、exti.c 及 exti.h 这几个文件。

在 timer.c 里面，我们修改 TIM3_IRQHandler 这个函数，用于统计帧率，修改代码如下：

```
u8 ov_frame=0;
void TIM3_IRQHandler(void)
{
    if(TIM3->SR&0X0001)//溢出中断
    {
        printf("frame:%dfps\r\n",ov_frame);
        ov_frame=0;
    }
    TIM3->SR&=~(1<<0);//清除中断标志位
}
```

这里，我们用到基本定时器 TIM3 来统计帧率，也就是 1 秒钟中断一次，打印 ov_frame 的值，ov_frame 用于统计 LCD 帧率。

在 exti.c 里面添加 EXTI8_Init 和 EXTI9_5_IRQHandler 函数，用于 OV7670 模块的 FIFO 写控制，exti.c 文件新增部分代码（先屏蔽原来的 EXTI15_10_IRQHandler 函数）如下：

```
//中断服务函数
u8 ov_sta;
void EXTI15_10_IRQHandler(void)
{
    if(EXTI->PR&(1<<15))//是 15 线的中断
    {
        if(ov_sta<2)
        {
            if(ov_sta==0)
            {
                OV7725_WRST=0;      //复位写指针
                OV7725_WRST=1;
                OV7725_WREN=1;      //允许写入 FIFO
            }else OV7725_WREN=0;    //禁止写入 FIFO
            ov_sta++;
        }
    }
    EXTI->PR=1<<15;         //清除 LINE15 上的中断标志位
}
//外部中断初始化程序
//初始化 PA15 为中断输入.
void EXTI15_Init(void)
{
    RCC->APB2ENR|=1<<2;             //使能 PORTA 时钟
    JTAG_Set(SWD_ENABLE);           //关闭 JTAG
    GPIOA->CRH&=0X0FFFFFFF;  //PA15 设置成输入
    GPIOA->CRH|=0X80000000;
```

```
    GPIOA->ODR|=1<<15;              //PA15 上拉
    Ex_NVIC_Config(GPIO_A,15,FTIR);//下降沿触发
    MY_NVIC_Init(2,1,EXTI15_10_IRQChannel,2);//抢占 2，子优先级 1，组 2
}
```

因为 OV7725 的帧同步信号（OV_VSYNC）接在 PA15 上面，所以这里配置 PA15 作为中端输入，因为 STM32 的外部中断 10~15 共用一个中端服务函数（EXTI9_5_IRQHandler），所以在该函数里面，我们需要先判断中断是不是来自中断线 15 的，然后再做处理。

中断处理部分很简单，通过一个 ov_sta 来控制 OV7725 模块的 FIFO 写操作。当 ov_sta=0 的时候，表示 FIFO 存储的数据已经被成功读取了（ov_sta 在读完 FIFO 数据的时候被清零），然后只要 OV_VSYNC 信号到来，我们就先复位一下写指针，然后 ov_sta=1，标志着写指针已经复位，目前正在往 FIFO 里面写数据。再等下一个 OV_VSYNC 到来，也就表明一帧数据已经存储完毕了，此时我们设置 OV7725_WREN 为 0，禁止再往 OV7725 写入数据，此时 ov_sta 自增为 2。其他程序，只要读到 ov_sta 为 2，就表示一帧数据已经准备好了，可以读出，在读完数据之后，程序设置 ov_sta 为 0，则开启下一轮 FIFO 数据存储。

再在 exti.h 里面添加 EXTI15_Init 函数的定义，就完成对 exti.c 和 exti.h 的修改了。

最后，打开 test.c 文件，修改代码如下：

```
#define  OV7725 1
#define  OV7670 2
//由于 OV7725 传感器安装方式原因,OV7725_WINDOW_WIDTH 相当于 LCD 的高度，
  OV7725_WINDOW_HEIGHT 相当于 LCD 的宽度
//注意：此宏定义只对 OV7725 有效
#define  OV7725_WINDOW_WIDTH          320 // <=320
#define  OV7725_WINDOW_HEIGHT         240 // <=240
extern u8 ov_sta;   //在 exit.c 里面定义
extern u8 ov_frame;     //在 timer.c 里面定义
//更新 LCD 显示(OV7725)
void OV7725_camera_refresh(void)
{
    u32 i,j;
    u16 color;
    if(ov_sta==2)
    {
        LCD_Scan_Dir(U2D_L2R);      //从上到下,从左到右
        LCD_Set_Window((lcddev.width-OV7725_WINDOW_WIDTH)/2,
          (lcddev.height-OV7725_WINDOW_HEIGHT)/2,OV7725_WINDOW_WIDTH,
          OV7725_WINDOW_HEIGHT);//将显示区域设置到屏幕中央
        if(lcddev.id==0X1963)
          LCD_Set_Window((lcddev.width-OV7725_WINDOW_WIDTH)/2,
          (lcddev.height-OV7725_WINDOW_HEIGHT)/2,OV7725_WINDOW_HEIGHT
          ,OV7725_WINDOW_WIDTH);//将显示区域设置到屏幕中央
        LCD_WriteRAM_Prepare();     //开始写入 GRAM
        OV7725_CS=0;
        OV7725_RRST=0;              //开始复位读指针
        OV7725_RCK=0;
```

```
            OV7725_RCK=1;
            OV7725_RCK=0;
            OV7725_RRST=1;                  //复位读指针结束
            OV7725_RCK=1;
            for(i=0;i<OV7725_WINDOW_HEIGHT;i++)
            {
                for(j=0;j<OV7725_WINDOW_WIDTH;j++)
                {
                    GPIOB->CRL=0X88888888;
                    OV7725_RCK=0;
                    color=OV7725_DATA;  //读数据
                    OV7725_RCK=1;
                    color<<=8;
                    OV7725_RCK=0;
                    color|=OV7725_DATA; //读数据
                    OV7725_RCK=1;
                    GPIOB->CRL=0X33333333;
                    LCD_WR_DATA(color);
                }
            }
            OV7725_CS=1;
            OV7725_RCK=0;
            OV7725_RCK=1;
            EXTI->PR=1<<15;
            ov_sta=0;                       //清零帧中断标记
            ov_frame++;
            LCD_Scan_Dir(DFT_SCAN_DIR);     //恢复默认扫描方向
        }
    }
```

OV7725_camera_refresh 函数，该函数用于读取摄像头模块自带 FIFO 里面的数据，并显示在 LCD 上面。对于 OV7725 我们可以通过宏定义设置图像窗口输出的大小，设置的宏定义：

```
//注意：此宏定义只对 OV7725 有效
#define  OV7725_WINDOW_WIDTH     320 // <=320
#define  OV7725_WINDOW_HEIGHT    240 // <=240
```

对分辨率大于 320*240 的屏幕，则通过开窗函数（LCD_Set_Window）将显示区域开窗在屏幕的正中央。注意，为了提高 FIFO 读取速度，我们将 FIFO_RCK 的控制，采用快速 IO 控制，关键代码如下（在 ov7725.h 里面）：

```
#define OV7725_RCK_H     GPIOB->BSRR=1<<4   //设置读数据时钟高电平
#define OV7725_RCK_L     GPIOB->BRR=1<<4    //设置读数据时钟低电平
```

OV7725_RCK_H 和 OV7725_RCK_L 就用到了 BSRR 和 BRR 这两个寄存器，以实现快速 IO 设置，从而提高读取速度。

下面我看一下 main 函数部分，代码如下：

```
int main(void)
```

```
{
    u8 sensor=0;
    u8 i;
    Stm32_Clock_Init(9);    //系统时钟设置
    uart_init(72,9600);     //串口初始化为 9600
    delay_init(72);         //延时初始化
    OV7670_Init();
    LED_Init();                 //初始化与 LED 连接的硬件接口
    LCD_Init();                 //初始化 LCD
    usmart_dev.init(72);    //初始化 USMART
    POINT_COLOR=RED;//设置字体为红色
    LCD_ShowString(30,50,200,200,16,"Mini STM32");
    LCD_ShowString(30,70,200,200,16,"OV7725_OV7670 TEST");
    LCD_ShowString(30,90,200,200,16,"ATOM@ALIENTEK");
    LCD_ShowString(30,110,200,200,16,"2017/11/1");
    LCD_ShowString(30,130,200,200,16,"Use USMART To Set!");
    LCD_ShowString(30,150,200,200,16,"OV7725_OV7670 Init...");
    while(1)//初始化 OV7725_OV7670
    {
        if(OV7725_Init()==0)
        {
            sensor=OV7725;
            LCD_ShowString(30,150,200,16,16,"OV7725 Init OK");
            delay_ms(1500);
            OV7725_Light_Mode(0);
            OV7725_Color_Saturation(0);
            OV7725_Brightness(0);
            OV7725_Contrast(0);
            OV7725_Special_Effects(0);
            OV7725_Window_Set(OV7725_WINDOW_WIDTH,
                OV7725_WINDOW_HEIGHT,0);//QVGA 模式输出
            OV7725_CS=0;
            break;
        }
        else if(OV7670_Init()==0)
        {
            sensor=OV7670;
            LCD_ShowString(30,150,200,16,16,"OV7670 Init OK");
            delay_ms(1500);
            OV7670_Light_Mode(0);
            OV7670_Color_Saturation(0);
            OV7670_Brightness(0);
            OV7670_Contrast(0);
            OV7670_Special_Effects(0);
```

```
                OV7670_Window_Set(12,176,240,320);//设置窗口
                OV7670_CS=0;
                break;
            }
            else
            {
                LCD_ShowString(30,150,200,16,16,"OV7725_OV7670 Error!!");
                delay_ms(200);
                LCD_Fill(30,150,200,246,WHITE);
                delay_ms(200);
            }
        }
        TIM3_Int_Init(10000,7199);          //TIM3,10Khz 计数频率,1 秒钟中断
        EXTI15_Init();                      //使能定时器捕获
        LCD_Clear(BACK_COLOR);
        while(1)
        {
            if(sensor==OV7725)OV7725_camera_refresh();      //更新显示
            else if(sensor==OV7670)OV7670_camera_refresh(); //更新显示
            if(i!=ov_frame)
                {i=ov_frame; LED0=!LED0;} //DS0 闪烁.
        }
    }
```

Main 函数前面初始化了用到的外设和 OV7725 或 OV7670，当检测到插入的 OV7725 模块则会给变量 sensor 赋值，sensor= OV7725。当插入 OV7670 模块，则 sensor=OV7670。在 while 循环里则会根据 sensor 选择显示函数 OV7725_camera_refresh() 或 OV7670_camera_refresh()。OV7670_camera_refresh()和 OV7725_camera_refresh()大同小异，这里就不贴出来说明了，大家可打开源码工程查看。

这里，我们可以通过 USMART 来设置和调节摄像头的参数，我们在 usmart_config.h 里面修改 usmart_nametab 的内容如下：

```
struct _m_usmart_nametab usmart_nametab[]=
{
#if USMART_USE_WRFUNS==1      //如果使能了读写操作
    (void*)read_addr,"u32 read_addr(u32 addr)",
    (void*)write_addr,"void write_addr(u32 addr,u32 val)",
#endif
    (void*)delay_ms,"void delay_ms(u16 nms)",
    (void*)delay_us,"void delay_us(u32 nus)",
    (void*)SCCB_WR_Reg,"u8 SCCB_WR_Reg(u8 reg,u8 data)",
    (void*)SCCB_RD_Reg,"u8 SCCB_RD_Reg(u8 reg)",
    (void*)OV7670_Light_Mode,"void OV7670_Light_Mode(u8 mode)",
    (void*)OV7670_Color_Saturation,"void OV7670_Color_Saturation(s8 sat)",
    (void*)OV7670_Brightness,"void OV7670_Brightness(s8 bright)",
    (void*)OV7670_Contrast,"void OV7670_Contrast(s8 contrast)",
```

```
        (void*)OV7670_Special_Effects,"void OV7670_Special_Effects(u8 eft)",
        (void*)OV7725_Light_Mode,"void OV7725_Light_Mode(u8 mode)",
        (void*)OV7725_Color_Saturation,"void OV7725_Color_Saturation(s8 sat)",
        (void*)OV7725_Brightness,"void OV7725_Brightness(s8 bright)",
        (void*)OV7725_Contrast,"void OV7725_Contrast(s8 contrast)",
        (void*)OV7725_Special_Effects,"void OV7725_Special_Effects(u8 eft)",
    (void*)OV7725_Window_Set,"voidOV7725_Window_Set(u16width,u16height,u8mode)",
    };
```

这样，我们就可以通过 USMART 设置摄像头的灯光模式、色饱和度、亮度、对比度和特效等。另外，我们还可以通过 SCCB_WR_Reg 和 SCCB_RD_Reg 这两个函数，来修改和读取 OV7725 和 OV7670 的各项设置，轻松实现摄像头的调试。

5、验证

ALIENTEK MiniSTM32 开发板与摄像头模块的连接，是通过杜邦线连接的，由于线比较长，容易受到外界干扰，导致图像显示错乱甚至显示不出来，这里我们推荐大家按以下方式对这些杜邦线进行分开捆绑，如图 5.1 所示：

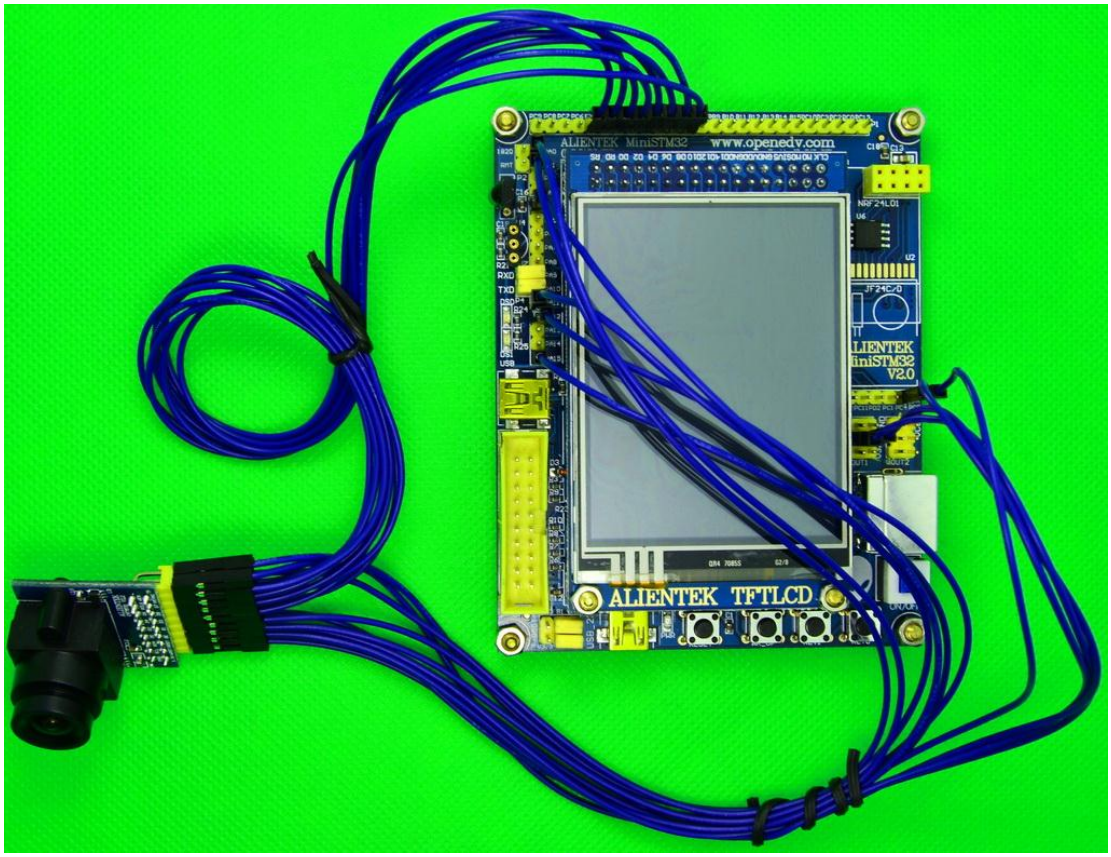

图 5.1 摄像头模块和 MiniSTM32 开发板连接示意图

如图 5.1 所示，我们将数据线：D0~D7 捆绑在一起，然后将其他线捆绑在一起，这样，可以有效防止图像错乱，甚至不出图像的问题。所以，推荐大家在连接好模块和开发板以后，按图 5.1 的方式进行捆绑处理。

然后，我们下载代码到 ALIENTEK MiniSTM32 开发板上，在摄像头模块初始化成功之后，即可得到摄像头拍摄到的画面，如图 5.2 所示：

图 5.2 OV725 摄像头模块拍摄效果图

此时，我们打开串口（9600 波特率），就可以从串口看到当前 LCD 的帧率，同时我们可以通过 USMART 调用相关函数来设置摄像头模块的不同模式，并且可以单独的设置和读取 OV7725 和 OV7670 的各个寄存器，实现对摄像头模块的调试，如图 5.2 所示：

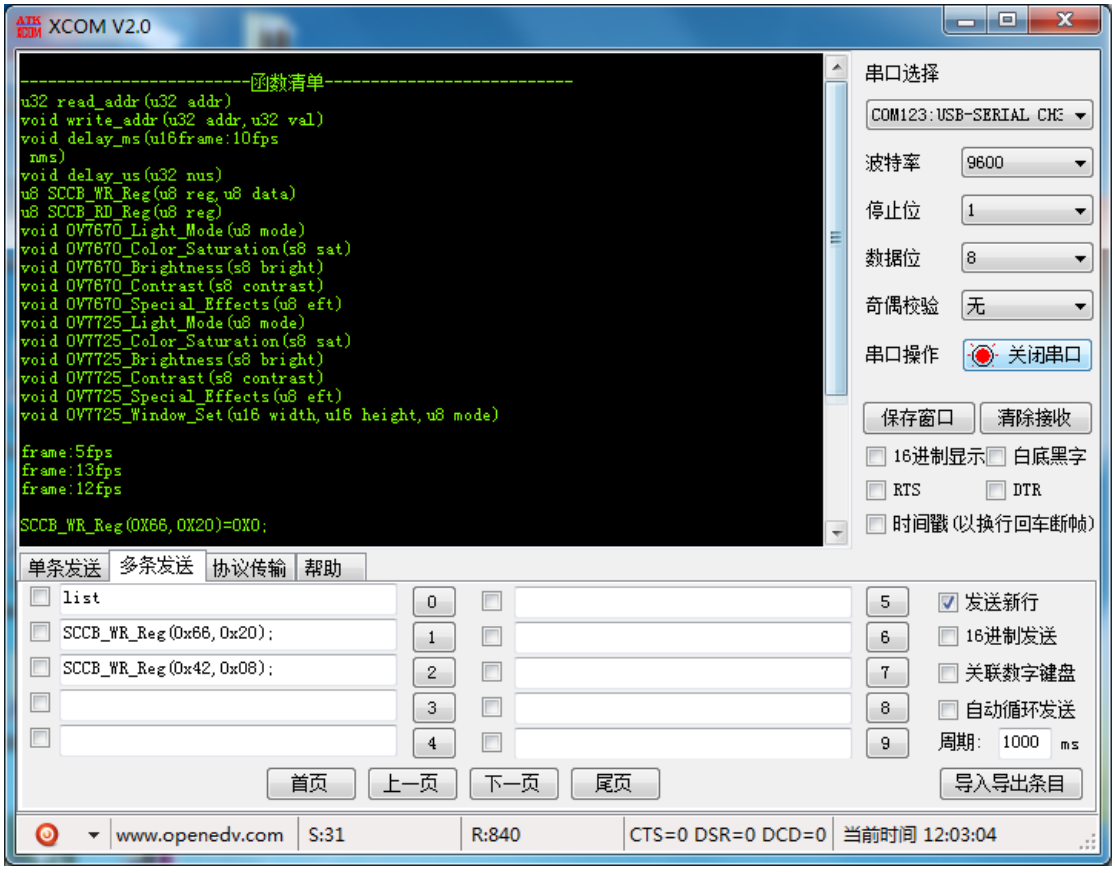

图 5.2 USMART 调试 OV7725

正点原子@ALIENTEK
2017-11-1
公司网址：www.alientek.com
技术论坛：www.openedv.com
资料下载地址：http://openedv.com/thread-232431-1-1.html
电话：020-38271790
传真：020-36773971