3.12 摄像头显示

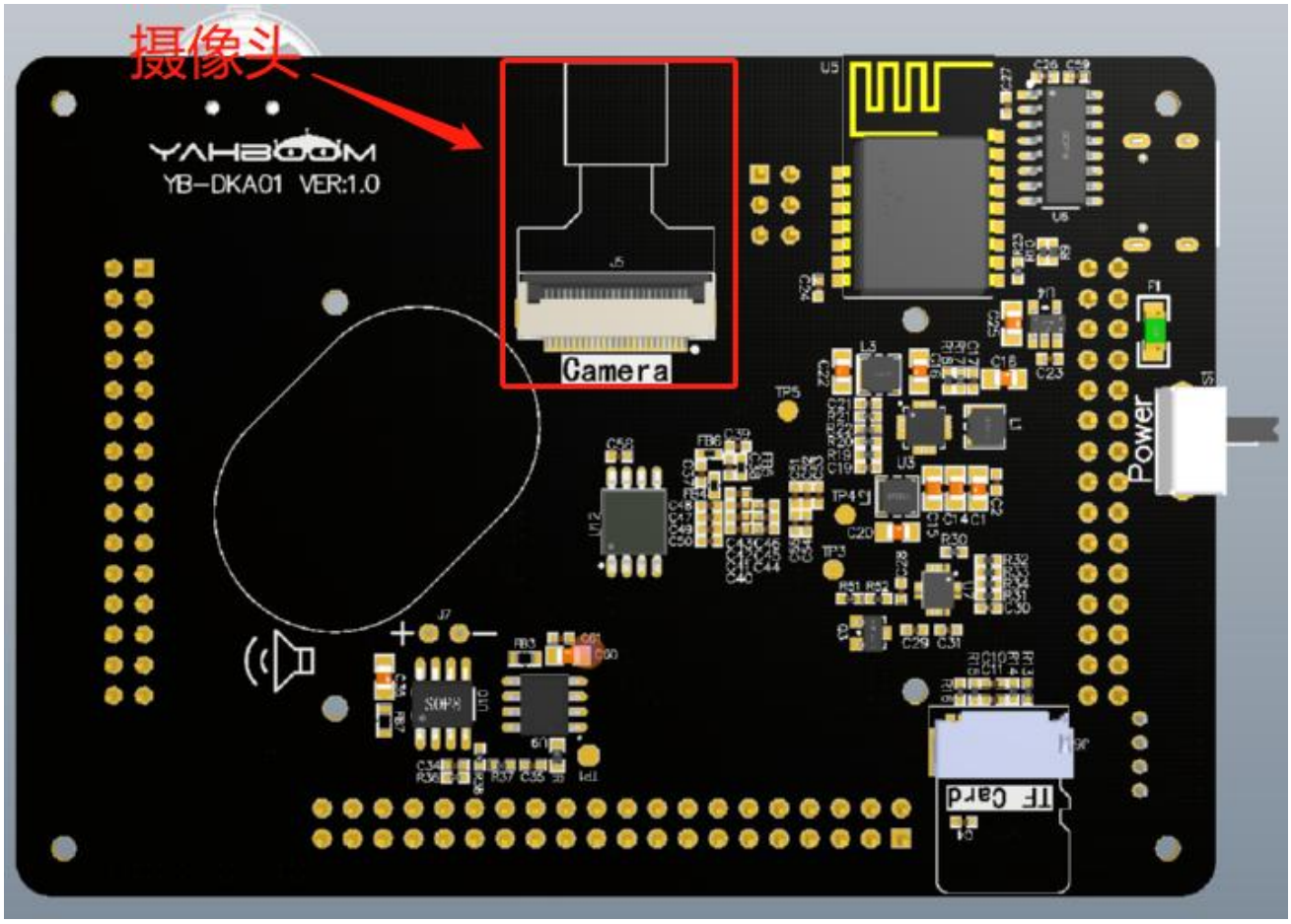
一、实验目的

本节课主要学习 K210 如何调用摄像头采集数据，然后通过 LCD 显示屏实时显示的功能。

二、实验准备

1.实验元件

ov2640 摄像头/ov9655 摄像头/GC2145 摄像头、LCD 显示屏



OV2640 是 Omni Vision 公司生产的一颗 CMOS 图像传感器，支持最高 200 万像素，支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹等自

动控制功能。

OV9655 也是 Omni Vision 公司生产的一颗 CMOS 图像传感器，支持最高 130 万像素，支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹等自动控制功能。

GC2145 摄像头是格科微公司生产的一颗 CMOS 图像传感器，支持最高 200 万像素，支持自动曝光控制、自动增益控制、自动白平衡、自动消除灯光条纹等自动控制功能。

2.元件特性

OV2640/OV9655/GC2145 摄像头与 K210 开发板通过数字摄像头接口（DVP）连接，DVP 是常用的摄像头接口模块，具有以下特性：

a.支持支持 SCCB 协议配置摄像头寄存器；

b.支持 640×480（30 万像素）及以下分辨率，每帧大小可配置；

c.支持 YUV422 和 RGB565 格式的图像输入；

d.支持图像同时输出到 KPU 和显示屏：

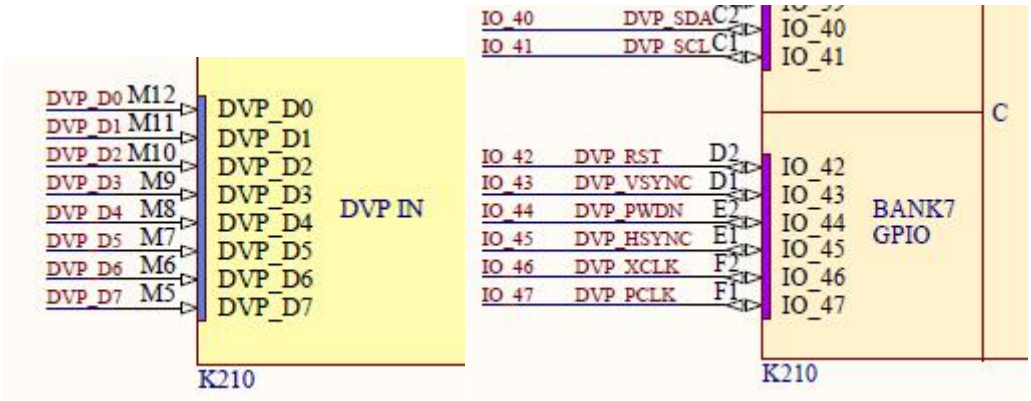
-输出到 KPU 的格式可选 RGB888，或 YUV422 输入时的 Y 分量，

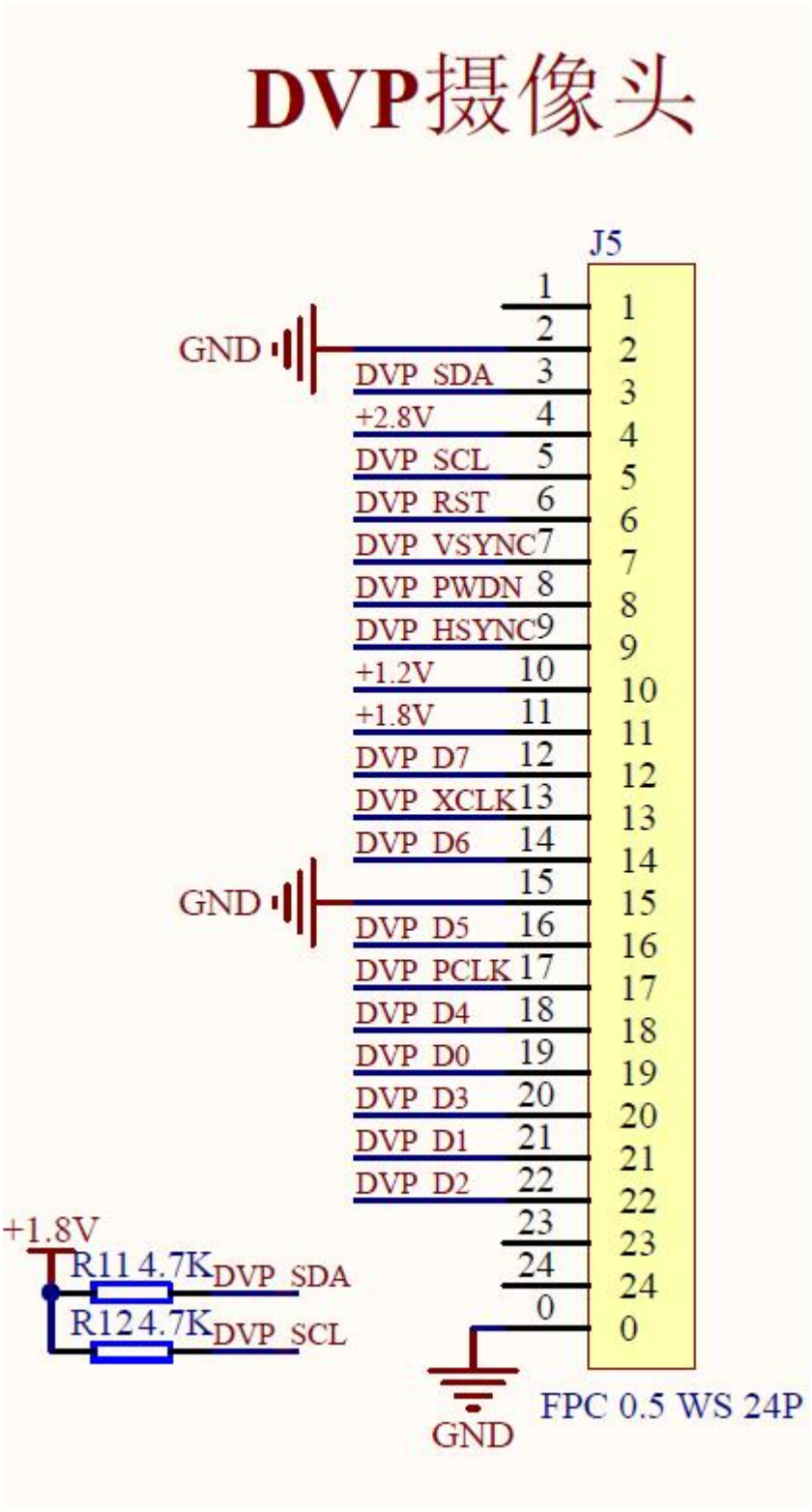
-输出到显示屏的格式为 RGB565；

e.检测到一帧开始或一帧图像传输完成时可向 CPU 发送中断。

3.硬件连接

K210 开发板出厂默认已经安装好摄像头和显示器，只需要使用 type-C 数据线连接 K210 开发板与电脑即可。


DVP_D0 M12
DVP_D1 M11
DVP_D2 M10
DVP_D3 M9
DVP_D4 M8
DVP_D5 M7
DVP_D6 M6
DVP_D7 M5
DVP_D0
DVP_D1
DVP_D2
DVP_D3
DVP_D4
DVP_D5
DVP_D6
DVP_D7
DVP IN
K210
IO_40 DVP_SDA C2
IO_41 DVP_SCL C1
IO_42 DVP_RST D2
IO_43 DVP_VSYNC D1
IO_44 DVP_PWDN E2
IO_45 DVP_HSYNC E1
IO_46 DVP_XCLK F2
IO_47 DVP_PCLK F1
IO_40
IO_41
IO_42
IO_43
IO_44
IO_45
IO_46
IO_47
BANK7
GPIO
C
K210



4. SDK 中对应 API 功能

对应的头文件 dvp.h

为用户提供以下接口：

- dvp_init ：DVP 初始化。
- dvp_set_output_enable：设置输出模式（内存或 AI）使能或禁止。
- dvp_set_image_format：设置图像接收模式，RGB 或 YUV。
- dvp_set_image_size：设置 DVP 图像采集尺寸。
- dvp_set_ai_addr：设置 AI 存放图像的地址，供 AI 模块进行算法处理。
- dvp_set_display_addr：设置采集图像在内存中的存放地址，可以用来显示。
- dvp_config_interrupt：设置图像开始和结束中断状态，使能或禁用。
- dvp_get_interrupt：判断是否是输入的中断类型，返回值：0 为否，非 0 为是。
- dvp_clear_interrupt：清除中断。
- dvp_start_convert：开始采集图像，在确定图像采集开始中断后调用。
- dvp_enable_burst：使能突发传输模式。
- dvp_disable_burst：禁用突发传输模式。
- dvp_enable_auto：使能自动接收图像模式。
- dvp_disable_auto：禁用自动接收图像模式。
- dvp_sccb_send_data：通过 sccb 协议发送数据。
- dvp_sccb_receive_data：通过 SCCB 接收数据。
- dvp_sccb_set_clk_rate：设置 sccb 的速率。
- dvp_set_xclk_rate：设置输入时钟的速率。

三、实验原理

ov2640/ov9655/GC2145 摄像头模组利用透镜成像的原理，采集图像，通过感光芯片及相关电路来记录和传输图像信号，按一定的分辨率，以隔行扫描的方式采集图像上的点，当扫描到某点时，就通过图像传感芯片将该点处图像的灰度转换成灰度一一对应的电压值，然后将此电压值通过视频信号端输出。

四、实验过程

下面以 OV2640 为例，OV9655/GC2145 摄像头的思路基本一样。

1. 首先根据上面的硬件连接引脚图，完成硬件引脚和软件功能的映射关系

```
/*****************************HARDWARE-PIN*****************************
// 硬件IO口，与原理图对应
#define PIN_LCD_CS            (36)
#define PIN_LCD_RST           (37)
#define PIN_LCD_RS            (38)
#define PIN_LCD_WR            (39)

// camera
#define PIN_DVP_PCLK          (47)
#define PIN_DVP_XCLK          (46)
#define PIN_DVP_HSYNC         (45)
#define PIN_DVP_PWDN          (44)
#define PIN_DVP_VSYNC         (43)
#define PIN_DVP_RST           (42)
#define PIN_DVP_SCL           (41)
#define PIN_DVP_SDA           (40)

/*****************************SOFTWARE-GPIO*****************************
// 软件GPIO口，与程序对应
#define LCD_RST_GPIONUM       (0)
#define LCD_RS_GPIONUM        (1)


/*****************************FUNC-GPIO*****************************
// GPIO口的功能，绑定到硬件IO口
#define FUNC_LCD_CS           (FUNC_SPI0_SS3)
#define FUNC_LCD_RST          (FUNC_GPIOHS0 + LCD_RST_GPIONUM)
#define FUNC_LCD_RS           (FUNC_GPIOHS0 + LCD_RS_GPIONUM)
#define FUNC_LCD_WR           (FUNC_SPI0_SCLK)
```

```
void hardware_init(void)
{
    /* lcd */
    fpioa_set_function(PIN_LCD_CS,  FUNC_LCD_CS);
    fpioa_set_function(PIN_LCD_RST, FUNC_LCD_RST);
    fpioa_set_function(PIN_LCD_RS,  FUNC_LCD_RS);
    fpioa_set_function(PIN_LCD_WR,  FUNC_LCD_WR);

    /* DVP camera */
    fpioa_set_function(PIN_DVP_RST,   FUNC_CMOS_RST);
    fpioa_set_function(PIN_DVP_PWDN,  FUNC_CMOS_PWDN);
    fpioa_set_function(PIN_DVP_XCLK,  FUNC_CMOS_XCLK);
    fpioa_set_function(PIN_DVP_VSYNC, FUNC_CMOS_VSYNC);
    fpioa_set_function(PIN_DVP_HSYNC, FUNC_CMOS_HREF);
    fpioa_set_function(PIN_DVP_PCLK,  FUNC_CMOS_PCLK);
    fpioa_set_function(PIN_DVP_SCL,   FUNC_SCCB_SCLK);
    fpioa_set_function(PIN_DVP_SDA,   FUNC_SCCB_SDA);

    /* 使能SPI0和DVP */
    sysctl_set_spi0_dvp_data(1);
}
```

2. 初始化电源域电压，摄像头和显示器都需要 1.8V 电平信号，根据所在电源域设置 bank6 和 bank7 的电压为 1.8V。

```
void io_set_power(void)
{
    sysctl_set_power_mode(SYSCTL_POWER_BANK6, SYSCTL_POWER_V18);
    sysctl_set_power_mode(SYSCTL_POWER_BANK7, SYSCTL_POWER_V18);
}
```

3. 设置系统时钟，初始化系统中断服务，使能全局中断。

```
/* 设置系统时钟和DVP时钟 */
sysctl_pll_set_freq(SYSCTL_PLL0, 800000000UL);
sysctl_pll_set_freq(SYSCTL_PLL1, 300000000UL);
sysctl_pll_set_freq(SYSCTL_PLL2, 45158400UL);
uarths_init();

/* 系统中断初始化，使能全局中断*/
plic_init();
sysctl_enable_irq();
```

4. 初始化 LCD 显示屏，并且显示图片和字符串，显示时间为 1 秒。

```
/* 初始化LCD */
lcd_init();

/* LCD显示图片 */
uint16_t *img = &gImage_logo;
lcd_draw_picture_half(0, 0, 320, 240, img);
lcd_draw_string(16, 40, "Hello Yahboom!", RED);
lcd_draw_string(16, 60, "Nice to meet you!", BLUE);
sleep(1);
```

5. 接下来是初始化 dvp。设置摄像头输出的格式以及图像大小和保存的图像地址等参数。

```c
/* dvp初始化 */
void dvp_cam_init(void)
{
    /* DVP初始化，设置sccb的寄存器长度为8bit */
    dvp_init(8);
    /* 设置输入时钟为24000000*/
    dvp_set_xclk_rate(24000000);
    /* 使能突发传输模式 */
    dvp_enable_burst();
    /* 关闭AI输出模式，使能显示模式 */
    dvp_set_output_enable(DVP_OUTPUT_AI, 0);
    dvp_set_output_enable(DVP_OUTPUT_DISPLAY, 1);
    /* 设置输出格式为RGB */
    dvp_set_image_format(DVP_CFG_RGB_FORMAT);
    /* 设置输出像素大小为320*240 */
    dvp_set_image_size(CAM_WIDTH_PIXEL, CAM_HIGHT_PIXEL);

    /* 设置DVP的显示地址参数和中断 */
    display_buf = (uint32_t*)iomem_malloc(CAM_WIDTH_PIXEL * CAM_HIGHT_PIXEL * 2);
    display_buf_addr = display_buf;
    dvp_set_display_addr((uint32_t)display_buf_addr);
    dvp_config_interrupt(DVP_CFG_START_INT_ENABLE | DVP_CFG_FINISH_INT_ENABLE, 0);
    dvp_disable_auto();
}
```

6. 设置 dvp 中断服务

```c
void dvp_cam_set_irq(void)
{
    /* DVP 中断配置：中断优先级，中断回调，使能DVP中断 */
    printf("DVP interrupt config\r\n");
    plic_set_priority(IRQN_DVP_INTERRUPT, 1);
    plic_irq_register(IRQN_DVP_INTERRUPT, on_dvp_irq_cb, NULL);
    plic_irq_enable(IRQN_DVP_INTERRUPT);

    /* 清除DVP中断位 */
    g_dvp_finish_flag = 0;
    dvp_clear_interrupt(DVP_STS_FRAME_START | DVP_STS_FRAME_FINISH);
    dvp_config_interrupt(DVP_CFG_START_INT_ENABLE | DVP_CFG_FINISH_INT_ENABLE, 1);
}
```

7. dvp 中断回调，把摄像头的内容保存到 display_buf_addr 这个地址变量中。

```c
/* dvp中断回调函数 */
static int on_dvp_irq_cb(void *ctx)
{
    /* 读取DVP中断状态，如果完成则刷新显示地址的数据，并清除中断标志，否则读取摄像头数据*/
    if (dvp_get_interrupt(DVP_STS_FRAME_FINISH))
    {
        dvp_set_display_addr((uint32_t)display_buf_addr);
        dvp_clear_interrupt(DVP_STS_FRAME_FINISH);
        g_dvp_finish_flag = 1;
    }
    else
    {
        if (g_dvp_finish_flag == 0)
            dvp_start_convert();
        dvp_clear_interrupt(DVP_STS_FRAME_START);
    }
    return 0;
}
```

8. 初始化 OV2640，直接把 OV2640 的寄存器以及对应的数据发送就可以。具体的寄存器以及功能可以查看硬件资料中的摄像头资料。

```c
/* 初始化ov2640摄像头 */
int ov2640_init(void)
{
    dvp_cam_init();
    dvp_cam_set_irq();

    uint16_t v_manuf_id;
    uint16_t v_device_id;
    ov2640_read_id(&v_manuf_id, &v_device_id);
    printf("manuf_id:0x%04x,device_id:0x%04x\n", v_manuf_id, v_device_id);
    uint16_t index = 0;
    for (index = 0; ov2640_config[index][0]; index++)
        dvp_sccb_send_data(OV2640_ADDR, ov2640_config[index][0], ov2640_config[index][1]);

    return 0;
}
```

9. 等待 DVP 传输完成后，LCD 显示地址变量的数据。

```
while (1)
{
    /* 等待摄像头采集结束，然后清除结束标志 */
    while (g_dvp_finish_flag == 0)
        ;
    g_dvp_finish_flag = 0;

    /* 显示画面 */
    lcd_draw_picture(0, 0, 320, 240, display_buf_addr);
}
```

10.编译调试，烧录运行

把本课程资料中的 camera 复制到 SDK 中的 src 目录下，

然后进入 build 目录，运行以下命令编译。

cmake .. -DPROJ=camera -G "MinGW Makefiles"

make

```
[ 80%] Building C object lib/CMakeFiles/kendryte.dir/drivers/uarths.c.obj
[ 82%] Building C object lib/CMakeFiles/kendryte.dir/drivers/utils.c.obj
[ 85%] Building C object lib/CMakeFiles/kendryte.dir/drivers/wdt.c.obj
[ 87%] Linking C static library libkendryte.a
[ 87%] Built target kendryte
Scanning dependencies of target camera
[ 89%] Building C object CMakeFiles/camera.dir/src/camera/ov2640.c.obj
[ 91%] Linking C executable camera
Generating .bin file ...
[100%] Built target camera
PS E:\K210\kendryte-standalone-sdk-develop\build>
```

编译完成后，在 build 文件夹下会生成 camera.bin 文件，

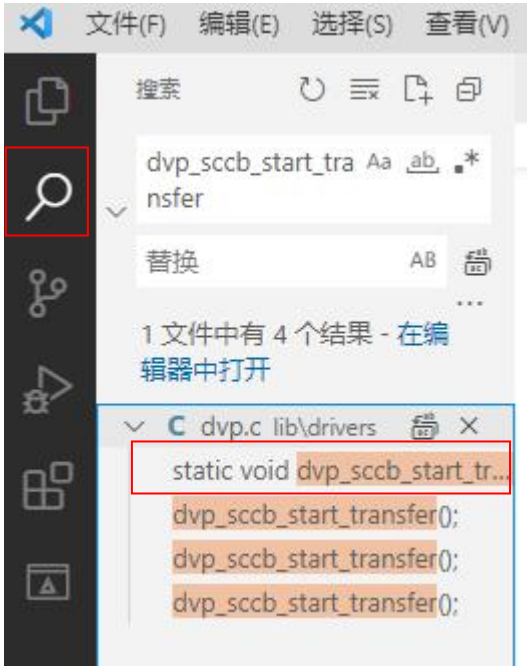
打开 kflash 将 camera.bin 文件烧录到 K210 开发板上。

五、注意事项(针对 OV9655 摄像头)

因为 OV9655 的通信时序比 OV2640 的通信时序慢一些，所以为了确保 OV9655 和 K210 能通信，一定要做以下步骤，否则 OV9655 将无法在 K210 使用。

第一步：打开一个完整的工程，找到 dvp.c 的源文件，一般在

D:\k210\kendryte-standalone-sdk-develop\lib\drivers 下，红色路径是工程的自定义路径，然后找到 dvp_sccb_start_transfer 这函数；或者用 Visual Studio Code 打开完整的工程，使用搜索功能，搜索 dvp_sccb_start_transfer 这函数，如图所示：



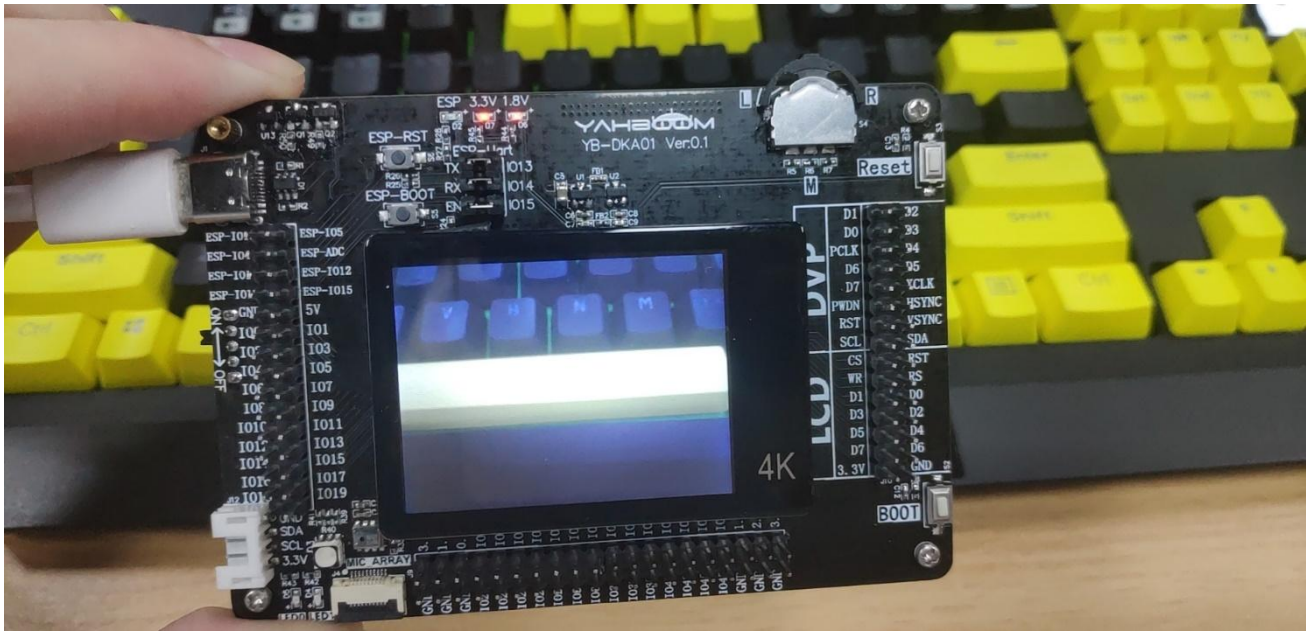
第二步：给 dvp_sccb_start_transfer 这函数增加这句代码 mdelay(5);如图所示

```
static void dvp_sccb_start_transfer(void)
{
    while(dvp->sts & DVP_STS_SCCB_EN)
        ;
    dvp->sts = DVP_STS_SCCB_EN | DVP_STS_SCCB_EN_WE;
    while(dvp->sts & DVP_STS_SCCB_EN)
        ;
    mdelay(5);//加上这句
}
```

第三步：如果 dvp_sccb_start_transfer 已经有了 mdelay(5)这句，前面两步可忽略。

六、实验现象

LCD 显示器先显示图片 logo 和字符串，一秒后打开摄像头采集的画面，并且实时显示在 LCD 上。



六、实验总结

1.K210 开发板板载 dvp 接口可以与兼容 dvp 接口的 ov2640/ov9655 摄像头连接使用。

2.K210 开发板显示摄像头画面是通过一帧一帧刷新 LCD 界面来达到动态效果的。

附：API

dvp_init

描述

初始化 DVP。

函数原型

```
void dvp_init(uint8_t reg_len)
```

参数

参数名称	描述	输入输出
reg_len	sccb 寄存器长度	输入

返回值

无

dvp_set_output_enable

描述

设置输出模式使能或禁用。

函数原型

```
void dvp_set_output_enable(dvp_output_mode_t index, int enable)
```

参数

参数名称	描述	输入输出
index	图像输出至内存或 AI	输入
enable	0：禁用 1：使能	输入

返回值

无。

dvp_set_image_format

描述

设置图像接收模式，RGB 或 YUV。

函数原型

```
void dvp_set_image_format(uint32_t format)
```

参数

参数名称	描述	图像模式	输入输出
format	DVP_CFG_RGB_FORMAT	RGB 模式	输入
	DVP_CFG_YUV_FORMAT	YUV 模式	

返回值

无

dvp_set_image_size

描述

设置 DVP 图像采集尺寸。

函数原型

```
void dvp_set_image_size(uint32_t width, uint32_t height)
```

参数

参数名称	描述	输入输出
width	图像宽度	输入
height	图像高度	输入

返回值

无

dvp_set_ai_addr

描述

设置 AI 存放图像的地址，供 AI 模块进行算法处理。

函数原型

```
void dvp_set_ai_addr(uint32_t r_addr, uint32_t g_addr, uint32_t
b_addr)
```

参数

参数名称	描述	输入输出
r_addr	红色分量地址	输入
g_addr	绿色分量地址	输入
b_addr	蓝色分量地址	输入

返回值

无

dvp_set_display_addr

描述

设置采集图像在内存中的存放地址，可以用来显示。

函数原型

```
void dvp_set_display_addr(uint32_t addr)
```

参数

参数名称	描述	输入输出
addr	存放图像的内存地址	输入

返回值

无

dvp_config_interrupt

配置 DVP 中断类型。

函数原型

```
void dvp_config_interrupt(uint32_t interrupt, uint8_t enable)
```

描述

设置图像开始和结束中断状态，使能或禁用。

参数

参数名称	描述	中断类型	输入输出
interrupt	DVP_CFG_START_INT_ENABLE	图像开始采集中断	输入

参数名称	描述	中断类型	输入输出
	DVP_CFG_FINISH_INT_ENABLE	图像结束采集中断	
enable	0：禁止 1：使能		输入

返回值

无。

dvp_get_interrupt

描述

判断是否是输入的中断类型。

函数原型

```
int dvp_get_interrupt(uint32_t interrupt)
```

参数

参数名称	描述	中断类型	输入输出
interrupt	DVP_CFG_START_INT_ENABLE	图像开始采集中断	输入
	DVP_CFG_FINISH_INT_ENABLE	图像结束采集中断	

返回值

返回值 描述

0 否

非 0 是

dvp_clear_interrupt

描述

清除中断。

函数原型

```
void dvp_clear_interrupt(uint32_t interrupt)
```

参数

参数名称	描述	中断类型	输入输出

参数名称	描述	中断类型	输入输出
interrupt	DVP_CFG_START_INT_ENABLE	图像开始采集中断	输入
	DVP_CFG_FINISH_INT_ENABLE	图像结束采集中断	

返回值

无。

dvp_start_convert

描述

开始采集图像，在确定图像采集开始中断后调用。

函数原型

```
void dvp_start_convert(void)
```

参数

无。

返回值

无。

dvp_enable_burst

描述

使能突发传输模式。

函数原型

```
void dvp_enable_burst(void)
```

参数

无。

返回值

无。

dvp_disable_burst

描述

禁用突发传输模式。

函数原型

```
void dvp_disable_burst(void)
```

参数

无。

返回值

无。

dvp_enable_auto

描述

使能自动接收图像模式。

函数原型

```
void dvp_enable_auto(void)
```

参数

无。

返回值

无。

dvp_disable_auto

描述

禁用自动接收图像模式。

函数原型

```
void dvp_disable_auto(void)
```

参数

无。

返回值

无。

dvp_sccb_send_data

描述

通过 sccb 发送数据。

函数原型

```
void dvp_sccb_send_data(uint8_t dev_addr, uint16_t reg_addr, uint8_t reg_data)
```

参数

参数名称	描述	输入输出
dev_addr	外设图像传感器 SCCB 地址	输入
reg_addr	外设图像传感器寄存器	输入
reg_data	发送的数据	输入

返回值

无

dvp_sccb_receive_data

描述

通过 SCCB 接收数据。

函数原型

```
uint8_t dvp_sccb_receive_data(uint8_t dev_addr, uint16_t reg_addr)
```

参数

参数名称	描述	输入输出
dev_addr	外设图像传感器 SCCB 地址	输入

参数名称	描述	输入输出
reg_addr	外设图像传感器寄存器	输入

返回值

读取寄存器的数据。

dvp_set_xclk_rate

描述

设置 xclk 的速率。

函数原型

```
uint32_t dvp_set_xclk_rate(uint32_t xclk_rate)
```

参数

参数名称　　描述　　输入输出

xclk_rate xclk 的速率 输入

返回值

xclk 的实际速率。

dvp_sccb_set_clk_rate

描述

设置 sccb 的速率。

函数原型

```
uint32_t dvp_sccb_set_clk_rate(uint32_t clk_rate)
```

参数

参数名称　　描述　　输入输出

clk_rate　sccb 的速率 输入

返回值

返回值 描述

0　　失败，设置的速率太低无法满足，请使用 I2C

返回值 描述

非 0　实际的 sccb 速率

数据类型

相关数据类型、数据结构定义如下：

- dvp_output_mode_t：DVP 输出图像的模式。

dvp_output_mode_t

描述

DVP 输入图像的模式。

定义

```
typedef enum _dvp_output_mode
{
    DVP_OUTPUT_AI,
    DVP_OUTPUT_DISPLAY,
} dvp_output_mode_t;
```

成员

成员名称	描述
DVP_OUTPUT_AI	AI 输出
DVP_OUTPUT_DISPLAY	向内存输出用于显示