3.12 Camera display

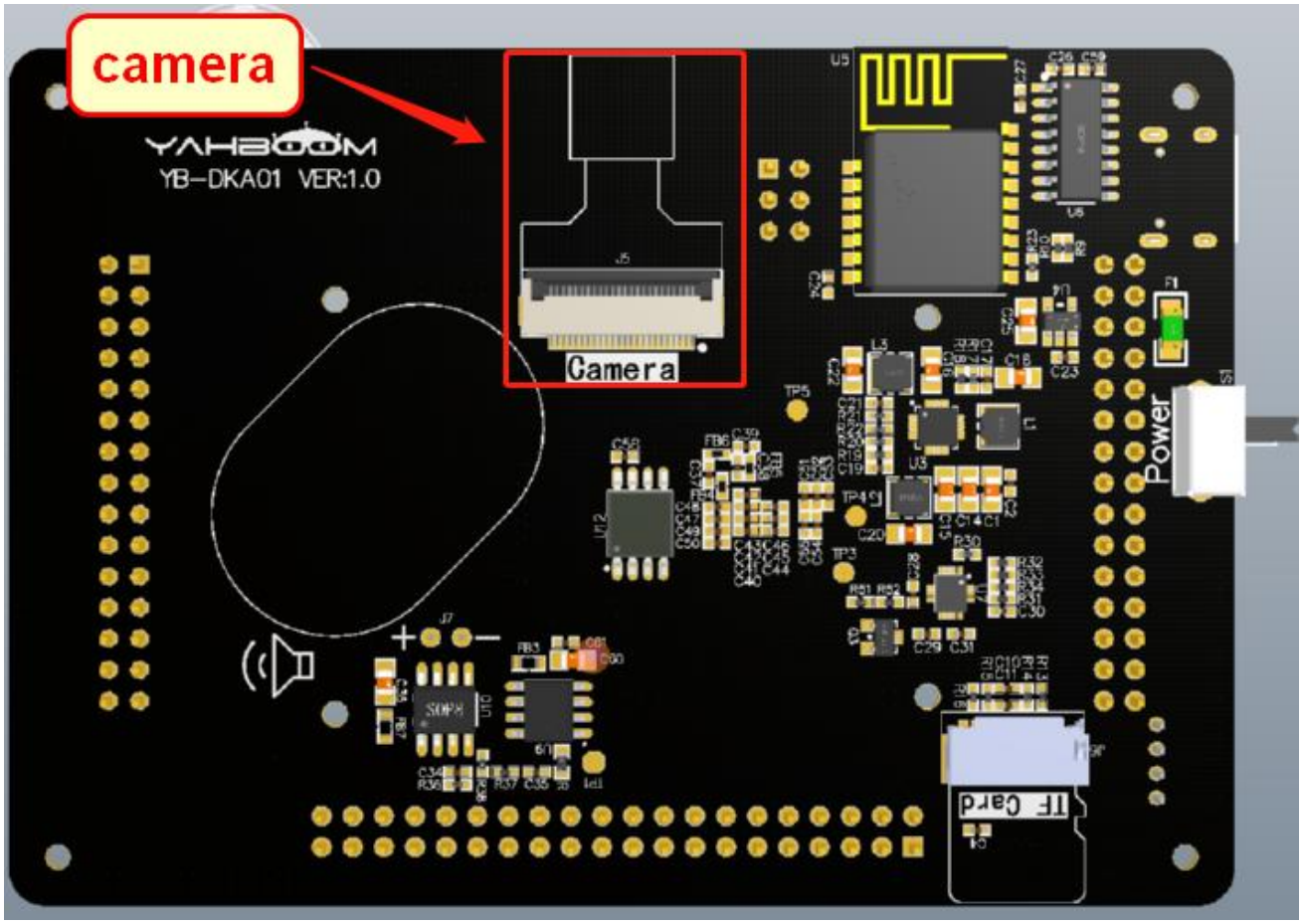
1. Experiment purpose

In this lesson, we will learn how to call the K210 camera to collect data and then display it in real time through the LCD screen.

2.Experiment preparation

2.1 components

ov2640 camera/ov9655 camera/GC2145 camera, LCD screen



OV2640 is a CMOS image sensor manufactured by Omni Vision. It supports up to 2 million pixels, automatic exposure control, automatic gain control, automatic white balance, automatic light stripe elimination and other automatic control functions.

OV9655 is also a CMOS image sensor produced by Omni Vision. It supports up to 1.3 million pixels and supports automatic control functions such as automatic exposure control, automatic gain control, automatic white balance, and automatic elimination of light streaks.

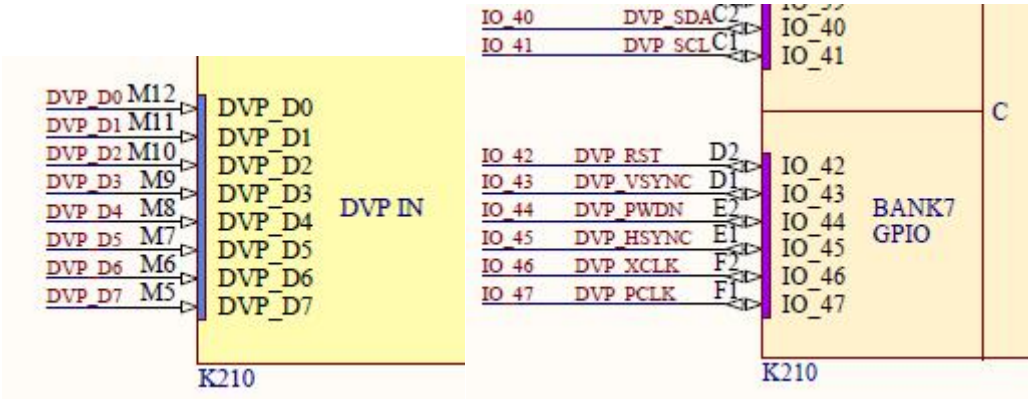
GC2145 is a CMOS image sensor produced by GC Micro. It supports up to 2 million pixels and supports automatic exposure control, automatic gain control, automatic white balance, automatic elimination of light streaks and other automatic control functions.

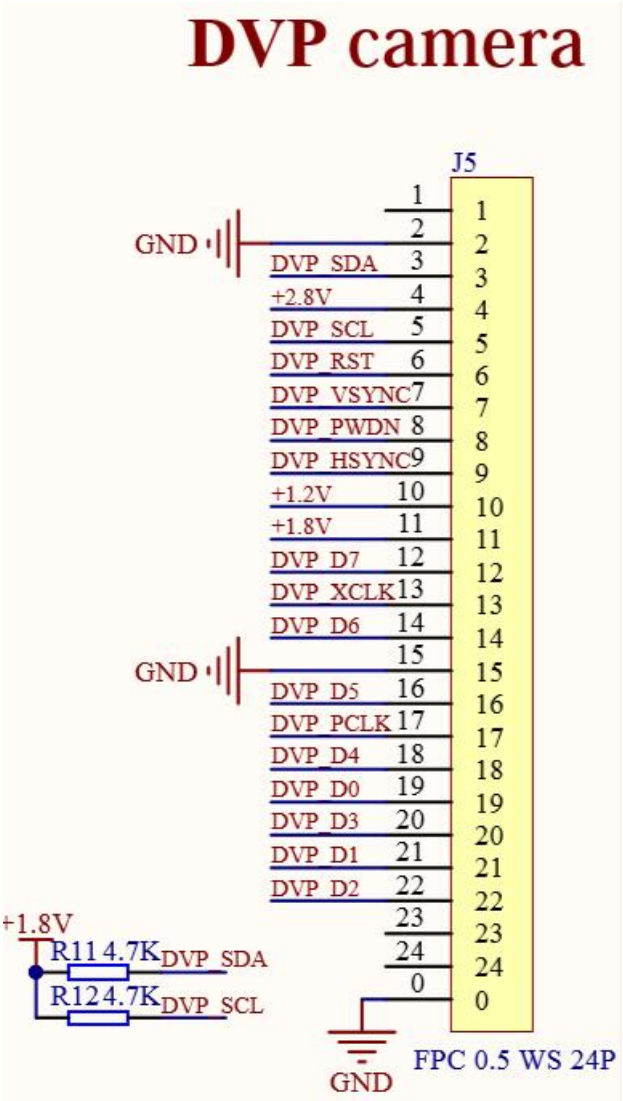
2.2 Component characteristics

OV2640/OV9655/GC2145 camera and K210 development board are connected by digital camera interface (DVP). DVP is a commonly used camera interface module with the following features:

a. SCCB protocol is supported to configure camera register;
b. Supports 640×480 (300,000 pixels) or less resolutions, and size of each frame can be configured;
c. Support YUV422 and RGB565 format image input;
d. Support image output to KPU and display screen simultaneously.
-The output format to KPU is optional: RGB888 or YUV422.
- Output to the display with RGB565 format.
e. An interrupt can be sent to the CPU when the start of a frame is detected or when a frame image transfer is completed.

2.3 Hardware connection
The K210 development board is shipped with a camera and monitor installed by default.





2.4 SDK API function

The header file is dvp.h

We will provide following interfaces to users.

dvp_init: DVP initialization.

dvp_set_output_enable: Set output mode (memory or AI) enabled or disabled.

dvp_set_image_format: Set image receive mode, RGB or YUV.

dvp_set_image_size: Set the DVP image acquisition size.

dvp_set_ai_addr: Set the address where AI can store the image for AI module to process the algorithm.

dvp_set_display_addr: Set the location of the acquisition image in memory, which can be used for display.

dvp_config_interrupt: Sets the image start and end interrupt states,enable or disable.

dvp_get_interrupt: Determine if the input interrupt type, return value: 0 for no, non-0 for yes.

dvp_clear_interrupt: clear interrupt

dvp_start_convert: Starting collecting images,

dvp_enable_burst: Enable burst transmission mode.
dvp_disable_burst: Disable burst transport mode.
dvp_enable_auto: Enable automatic receiving of image mode.
dvp_disable_auto: Disable automatic receiving of image mode.
dvp_sccb_send_data: Data is sent through the SCCB protocol.
dvp_sccb_receive_data: Receive data through SCCB.
dvp_sccb_set_clk_rate: Set the rate of SCCB.
dvp_set_xclk_rate: Set the rate of the input clock.

3. Experimental principle

Ov2640/Ov9655/GC2145 camera module uses the principle of lens imaging to collect images. It record and transmit image signals through the photosensitive chip and related circuits. It can collect points on the image in an interlaced scan at a certain resolution.
When a certain point is scanned, the image sensor chip converts the gray level of the image at that point into a corresponding voltage value, then this voltage value is output through the video signal terminal.

4. Experiment procedure

Taking OV2640 as an example below, the idea of OV9655/GC2145 camera is basically the same.
4.1 According to the above hardware connection pin diagram, K210 hardware pins and software functions use FPIOA mapping relationship.

```
/*****************************HARDWARE-PIN*********************************/
//Hardware IO port, corresponding Schematic
#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********************************/
// Software GPIO port, corresponding program
#define LCD_RST_GPIONUM         (0)
#define LCD_RS_GPIONUM          (1)


/*****************************FUNC-GPIO************************************/
//Function of GPIO port, bound to hardware IO port
#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)
```

```
/**
* Function       hardware_init
* @author        Gengyue
* @date          2020.05.27
* @brief         Hardware initialization, binding GPIO port
* @param[in]     void
* @param[out]    void
* @retval        void
* @par History   NO
*/
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);

    /* Enable SPI0 and DVP */
    sysctl_set_spi0_dvp_data(1);
}
```

4.2 Initialize the voltage in the power domain, a 1.8V level signal is required for both the camera and the display.

The voltage of Bank6 and Bank7 is set to 1.8V according to the power domain where they are located.

```
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);
}
```

4.3 Sets the system clock, initializes the system interrupt service, and enables global interrupt.

```
/* Set the system clock and the DVP clock */
sysctl_pll_set_freq(SYSCTL_PLL0, 800000000UL);
sysctl_pll_set_freq(SYSCTL_PLL1, 300000000UL);
sysctl_pll_set_freq(SYSCTL_PLL2, 45158400UL);
uarths_init();

/* System interrupt initialization, enabling global interrupt */
plic_init();
sysctl_enable_irq();
```

4.4 Initializes the LCD display and displays images and strings for 1 second.

```
/* Initialize LCD */
lcd_init();

/* LCD display picture */
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);
```

4.5 Initialize the DVP. Set the format of the camera output as well as the image size and saved image address and other parameters.

```
/* dvp Initialization */
void dvp_cam_init(void)
{
    /* DVP is initialized, and the register length of SCCB is set to 8 bits*/
    dvp_init(8);
    /* Set the input clock to 24000000*/
    dvp_set_xclk_rate(24000000);
    /* Enable burst mode */
    dvp_enable_burst();
    /* Turn off the AI output mode, enable display mode*/
    dvp_set_output_enable(DVP_OUTPUT_AI, 0);
    dvp_set_output_enable(DVP_OUTPUT_DISPLAY, 1);
    /* Set the output format to RGB*/
    dvp_set_image_format(DVP_CFG_RGB_FORMAT);
    /* Set the output pixel size to 320*240 */
    dvp_set_image_size(CAM_WIDTH_PIXEL, CAM_HIGHT_PIXEL);

    /* Set the DVP display address parameters and interrupt */
    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();
}
```

4.6 Set up the DVP interrupt service

```
void dvp_cam_set_irq(void)
{
    /* DVP interrupt configuration: Interrupt priority, interrupt callback, enable DVP interrupt */
    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);

    /* Clear the DVP interrupt bits */
    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);
}
```

4.7 DVP interrupts the callback and saves the contents of the camera to the address variable display_buf_addr.

```
/* dvp Interrupt callback function */
static int on_dvp_irq_cb(void *ctx)
{
    /*Read the DVP interrupt status, refresh the data of the display addr
    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;
}
```

4.8 Initialize OV2640. More detail, please view [8.About Hardware]---[camera]

```c
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;
}
```

4.9 After the DVP transmission is completed, the LCD displays the address variable data.

```c
while (1)
{
    /* Wait for the end of the camera acquisition, and then clear the end sign */
    while (g_dvp_finish_flag == 0)
        ;
    g_dvp_finish_flag = 0;

    /* Display picture*/
    lcd_draw_picture(0, 0, 320, 240, display_buf_addr);
}
```

4.10 Compile and debug, burn and run

Copy the camera to the src directory in the SDK.

Then, enter the build directory and run the following command to compile.

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>
```

After the compilation is complete, the **camera.bin** file will be generated in the build folder.

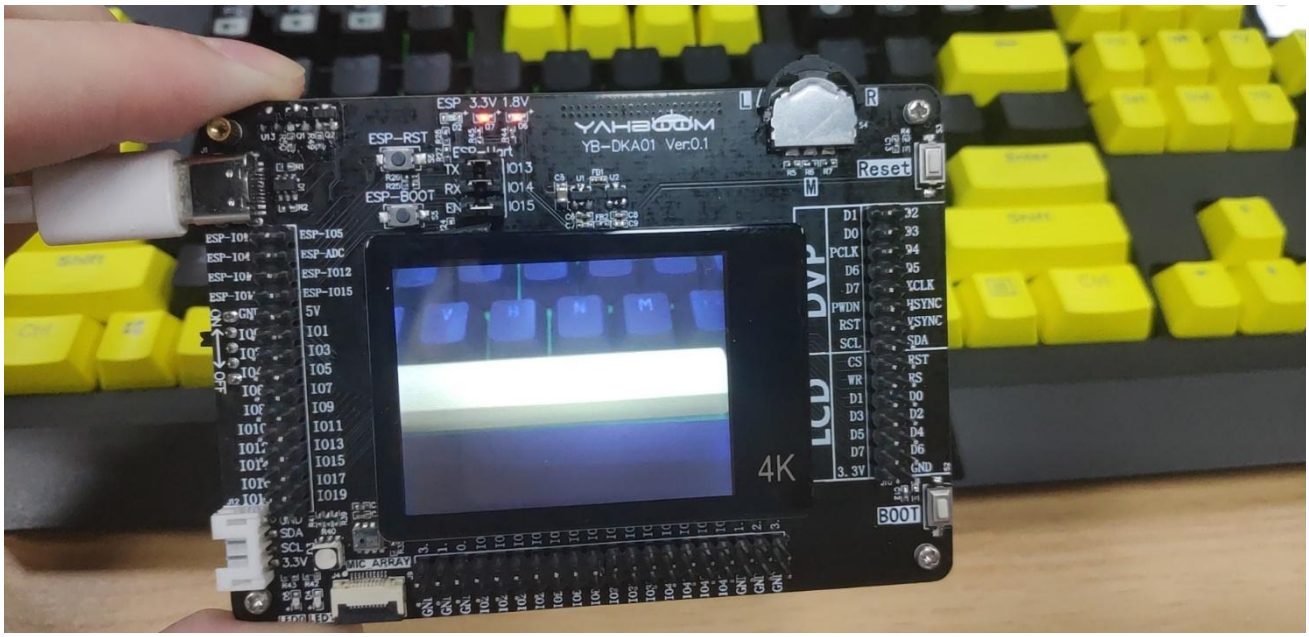
We need to use the type-C data cable to connect the computer and the K210 development board.

Open kflash, select the corresponding device, and then burn the **camera.bin** file to the K210 development board.

5. Experimental phenomenon

After the firmware is write,

First, LCD will display the picture logo and character string. After 1s, it will start capture picture video, and displays it on the LCD in real time.



6. Experiment summary

6.1 K210 development board DVP interface can be connected with OV2640/Ov9655 camera.

6.2 K210 development board display camera screen is achieved by refreshing the LCD interface frame by frame to achieve dynamic effect.

Appendix -- API

dvp_init

Description: Initialize DVP.

Function prototype: **void dvp_init(uint8_t reg_len)**

Parameter:

Parameter name	Description	Input/Ouput
reg_len	Length of sccb register	Input

Return value: No

dvp_set_output_enable

Description: Set output mode enable/disable

Function prototype: **void dvp_set_output_enable(dvp_output_mode_t index, int enable)**

Parameter:

Parameter name	Description	Input/Ouput
index	Image output to memory or AI	Input
enable	0-disable 1-enable	Input

Return value: No

dvp_set_image_format

Description: Set image receive mode, RGB or YUV.

Function prototype: void dvp_set_image_format(uint32_t format)

Parameter:

Parameter name	Description	Image mode	Input/Ouput
format	DVP_CFG_RGB_FORMAT	RGB mode	Input
	DVP_CFG_YUV_FORMAT	YUV mode	

Return value: No

dvp_set_image_size

Description: Set the DVP image acquisition size.

Function prototype: **void dvp_set_image_size(uint32_t width, uint32_t height)**

Parameter:

Parameter name	Description	Input/Ouput
width	width of image	Input
height	Height of image	Input

Return value: No

dvp_set_ai_addr

Description: Set the address where AI can store the image for AI module to process the algorithm.

Function prototype: **void dvp_set_ai_addr(uint32_t r_addr, uint32_t g_addr, uint32_t b_addr)**

Parameter:

Parameter name	Description	Input/Ouput
r_addr	Red component address	Input
g_addr	Green component address	Input
b_addr	Blue component address	Input

Return value: No

dvp_set_display_addr

Description: Set the location of the acquisition image in memory, which can be used for display.

Function prototype: **void dvp_set_display_addr(uint32_t addr)**

Parameter:

Parameter name	Description	Input/Ouput
addr	The memory address to store the image	Input

Return value: No

dvp_config_interrupt

Description: Configure the DVP interrupt type.

Function prototype: **void dvp_config_interrupt(uint32_t interrupt, uint8_t enable)**
Description: Sets the image start and end interrupt states to enable or disable.
Parameter:

Parameter name	Description	Interrupt type	Input/Output
interrupt	DVP_CFG_START_INT_ENABLE	Start image acquisition interrupts	Input
	DVP_CFG_FINISH_INT_ENABLE	End image acquisition interrupt	
enable	0-disable, 1-enable		Input

Return value: No

dvp_get_interrupt
Description: Determine whether it is the input interrupt type.
Function prototype: **int dvp_get_interrupt(uint32_t interrupt)**
Parameter:

Parameter name	Description	Interrupt type	Input/Output
interrupt	DVP_CFG_START_INT_ENABLE	Start image acquisition interrupts	Input
	DVP_CFG_FINISH_INT_ENABLE	End image acquisition interrupt	

Return value:

Return value	Description
0	no
!0	yes

dvp_clear_interrupt
Description: clearing interrupt
Function prototype: **void dvp_clear_interrupt(uint32_t interrupt)**
Parameter:

Parameter name	Description	Interrupt type	Input/Output
interrupt	DVP_CFG_START_INT_ENABLE	Start image acquisition interrupts	Input
	DVP_CFG_FINISH_INT_ENABLE	End image acquisition interrupt	

Return value: No

dvp_start_convert
Description: Start to collect images, and call after confirming that the image collection is interrupted.
Function prototype: **void dvp_start_convert(void)**
Parameter: no
Return value: No

dvp_enable_burst
Description: Enable burst mode
Function prototype: **void dvp_enable_burst(void)**
Parameter: no
Return value: no

dvp_disable_burst
Description: Disable burst mode
Function prototype: **void dvp_disable_burst(void)**
Parameter: no
Return value: no

dvp_enable_auto
Description: Enable automatic image receiving mode.
Function prototype: **void dvp_enable_auto(void)**
Parameter: no
Return value: no

dvp_disable_auto
Description: Disable automatic image receiving mode.
Function prototype: **void dvp_disable_auto(void)**
Parameter: no
Return value: no

dvp_sccb_send_data
Description: Send data through sccc.
Function prototype: **void dvp_sccb_send_data(uint8_t dev_addr, uint16_t reg_addr, uint8_t reg_data)**
Parameter:

Parameter name	Description	Input/Output
dev_addr	Peripheral image sensor SCCB address	Input
reg_addr	Peripheral image sensor register	Input
reg_data	Data sent	Input

Return value: no

dvp_sccb_receive_data
Description: Receive data through sccc.
Function prototype: **uint8_t dvp_sccb_receive_data(uint8_t dev_addr, uint16_t reg_addr)**
Parameter:

Parameter name	Description	Input/Output
dev_addr	Peripheral image sensor SCCB address	Input
reg_addr	Peripheral image sensor register	Input

Return value: Read the data of the register.

dvp_set_xclk_rate

Description: Set xclk rate.

Function prototype: **uint32_t dvp_set_xclk_rate(uint32_t xclk_rate)**

Parameter:

Parameter name	Description	Input/Output
xclk_rate	xclk rate	Input

Return value: xclk actual rate

dvp_sccb_set_clk_rate

Description: Set sccb rate.

Function prototype: **uint32_t dvp_sccb_set_clk_rate(uint32_t clk_rate)**

Parameter:

Parameter name	Description	Input/Output
clk_rate	sccb rate	Input

Return value:

Parameter name	Description
0	Failed, the set rate is too low to meet, please use I2C
!0	sccb actual rate

Data type

Relevant data types and data structures are defined as follows:

- dvp_output_mode_t: DVP output image mode.

dvp_output_mode_t

Description: DVP output image mode.

Define

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

member

Member name	Description
DVP_OUTPUT_AI	AI ouput
DVP_OUTPUT_DISPLAY	Output to memory for display