HiKey970

Camera Introduce

Issue	**01**
Date	**2018-03-11**

HiSilicon Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	http://www.hisilicon.com
Email:	support@hisilicon.com

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 01 (2018-03-11)

The first version.

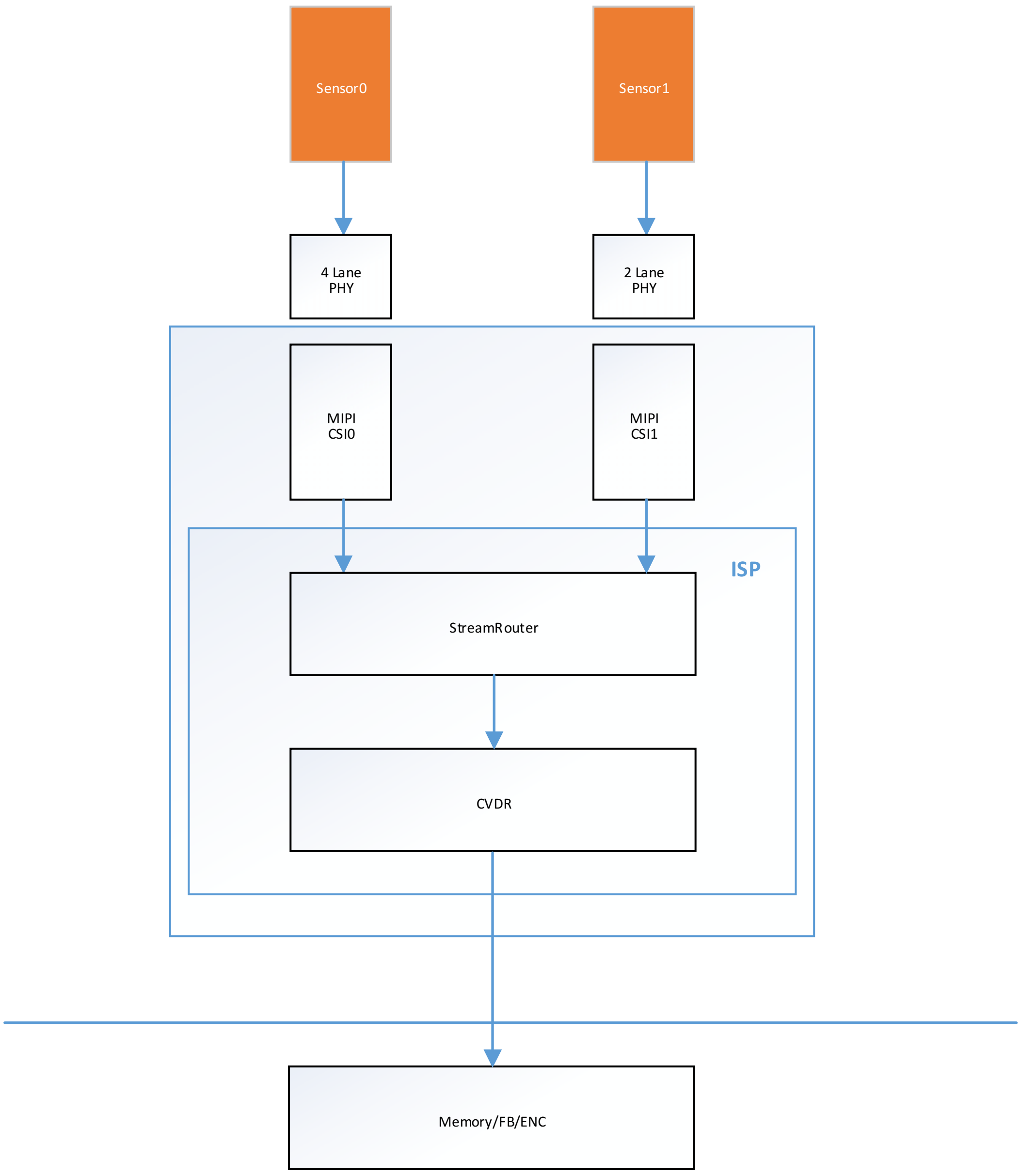
Contents

1 Description

1.1 ISP Overview

The Image Signal Processor(ISP) is the key enabler for still image capture and video capture use cases. It interfaces with up to 2 external camera modules, to of which data streams can be processed by the ISP simultaneously. The ISP simply sends the data stream of the camera to DRAM for processing outside the ISP. The ISP receives data from the cameras and processes it to make it usable by the GPU, display controller or encoder.

The following figure illustrates connections between ISP module and camera module. Camera sensor0/1 transmits stream data to ISP StreamRouter via CSI Phy, and then sends it to CVDR, finally writes to memory. The hikey970 ISP supports two MIPI CSI-2 with 4 DL interfaces.The ISP includes several hardware controller, ex: StreamRouter,CVDR.

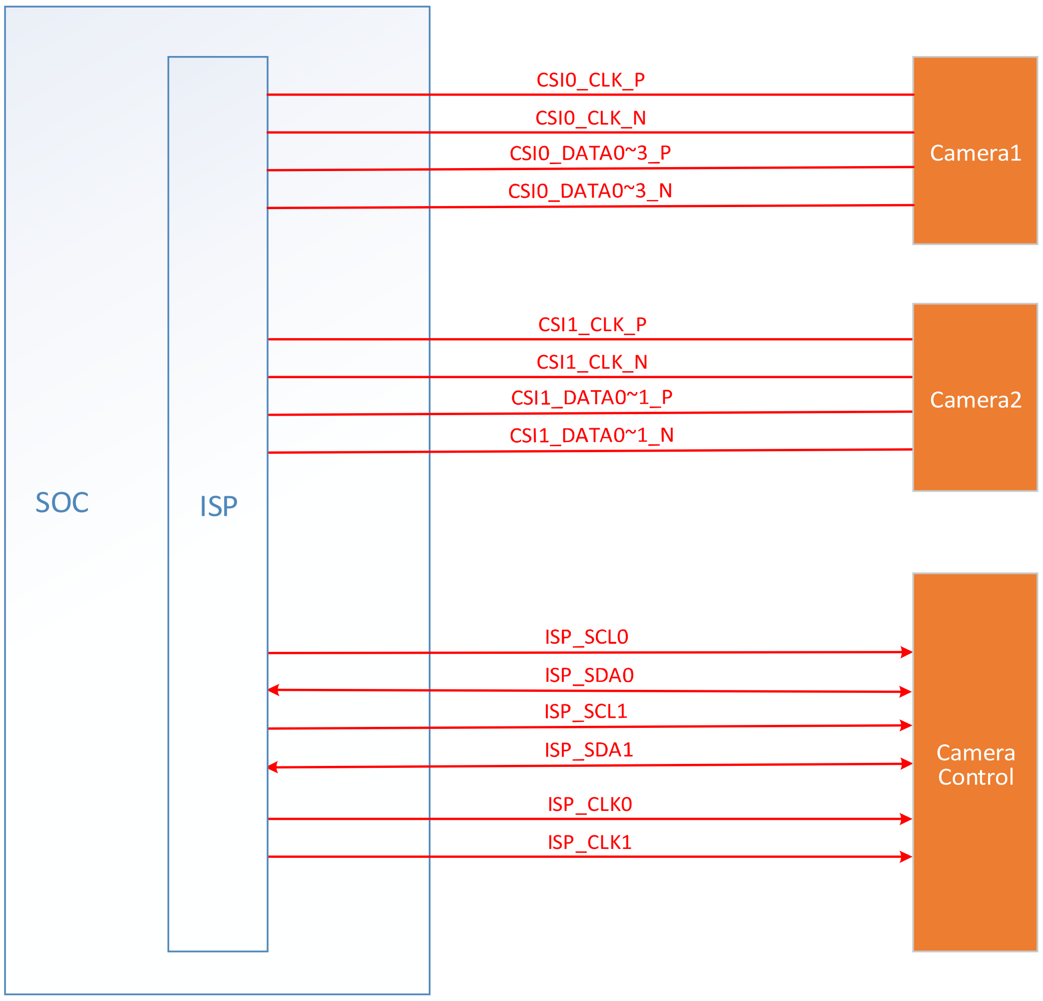


1.2 Functional Description

1.2.1 MIPI CSI-2

The CSI-2 specification defines standard data transmission and control interfaces between transmitter and receiver. Data transmission interface (referred as CSI-2) is unidirectional differential serial interface with data and clock signals; the physical layer of this interface is the MIPI Alliance Specification for D-PHY. The following figure illustrates connections between CSI-2 transmitter and receiver, which typically are a camera module and a ISP module. The hikey970 ISP supports two MIPI CSI-2 with 4+2 DL interfaces. The P/N represents differential signal.

Figure--ISP And Camera connect interface



When you configure CSI , you should indicate how many lanes to use to transfer stream data(If you set 4 lanes, the camera sensor must support 4 lanes). You could configure lanes as ISP CSI datasheet. Detailed configuration please refer to the hisi_isp_csi.c file.

1.2.2 StreamRouter

The StreamRouter functionalities are:

The main function of the StreamRouter module is to select the output path for the incoming camera data stream.

Routing the different pixels or data streams coming from the cameras through the CSI2 interfaces to DMAs to send data directly to DRAM.

Absorbing the back pressure from ISP pipelines and from the centralized video data router(CVDR) due to the DRAM latency.

Reordering/decompressing incomoing pixels from CSI2 interfaces.

The StreamRouter controls the frame processing rat by setting the GO bit CSIFILTER_GO. FILTER_GO[7:0].If the GO is not set, data received from the camera are discarded.

Step1 (initialization)

Set the StreamRouter static configurations(initialization only) and configurations that vary from frame to frame

The StreamRouter receives stream data from cameras through CSI2 interface, so we need configure CSI index to tell streamRouter that receives data from the specified csi. Of course, we also need to configure ID_Router to specify the data output channel.

Step2

Set the GO bit CSIFILTERGO.FILTERGO[7:0] for the all active channels for a given use case.

Go back to step1 to prepare next frame processing be setting new configuration in register shadows.

Detailed configuration please refer to the hisi_isp_sr.c file.

1.2.3 CVDR

The CVDR receives stream data from StreamRouter's ID Router, so we need configure cvdr port to specify the data input channel. And, we also need write stream data to memory through CVDR, so we must define the memory address to CVDR.

Detailed configuration please refer to the hisi_isp_cvdr.c file.

1.2.4 ISP Interrupts

The ISP generates several interrupts routed to either the ISP MPU or to the MPU-Host.

IRQ_MERGER_2 routs the ISP core interrupt towards the MPU_host, The IRQ_MERGER_2 merges all ISP interrupts into groups of 32 named FRPROC_x, ERROR_x, DEBUG_x.

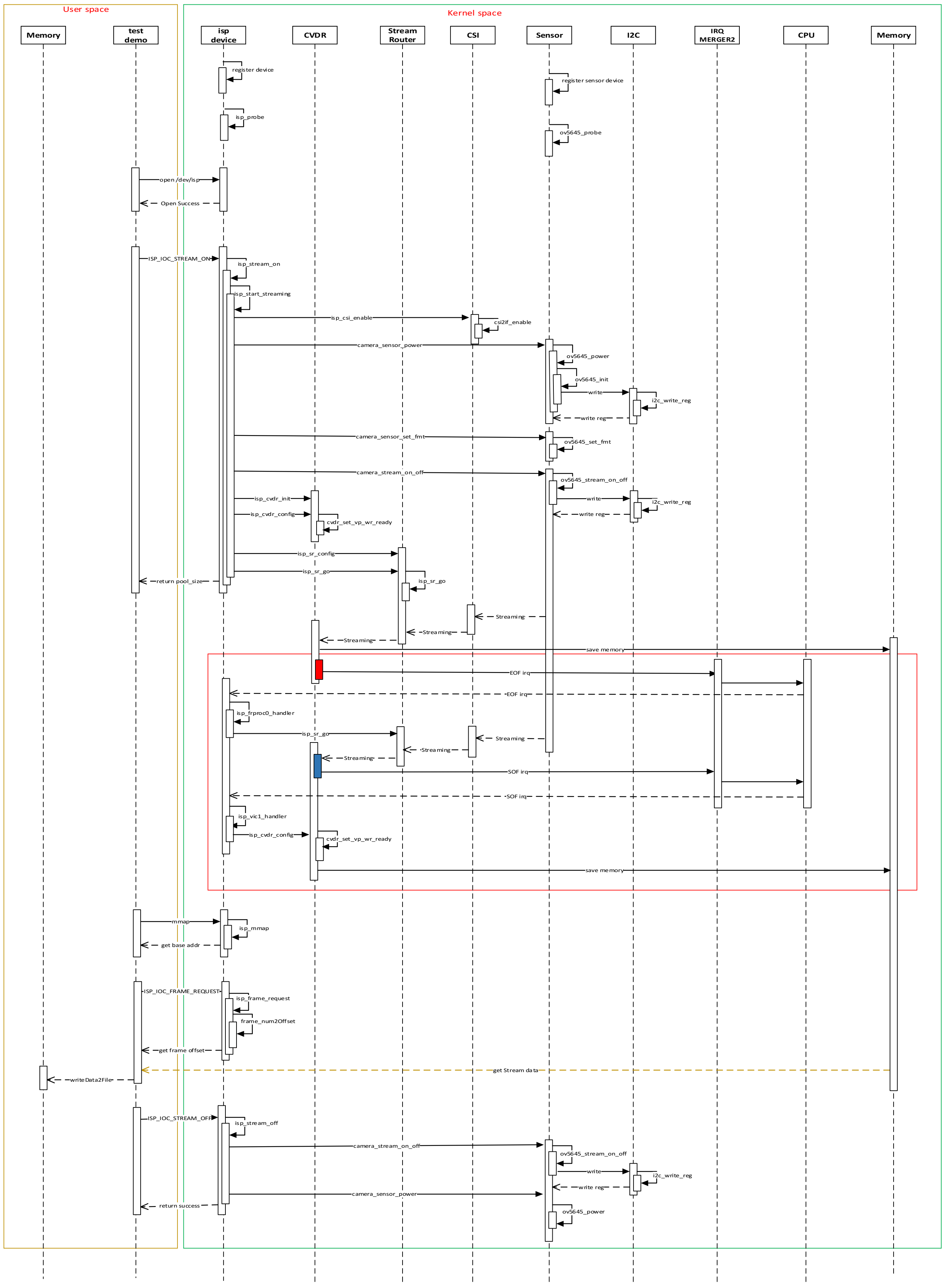
FRPROC_x are interrupt occurring during the Frame processing and more subject to be used by the software for the ISP control

ERROR_x are error events

DEBUG_x are debug events

1.3 ISP Process Flow

Figure--ISP Timing process


User space
Kernel space
Memory
test demo
isp device
CVDR
Stream Router
CSI
Sensor
I2C
IRQ MERGER2
CPU
Memory
register device
register sensor device
isp_probe
ov5645_probe
open /dev/isp
Open Success
ISP_IOC_STREAM_ON
isp_stream_on
isp_start_streaming
isp_csi_enable
csi2if_enable
camera_sensor_power
ov5645_power
ov5645_init
write
i2c_write_reg
write reg
camera_sensor_set_fmt
ov5645_set_fmt
camera_stream_on_off
ov5645_stream_on_off
isp_cvdr_init
write
i2c_write_reg
isp_cvdr_config
write reg
cvdr_set_vp_wr_ready
isp_sr_config
isp_sr_go
isp_sr_go
return pool_size
Streaming
Streaming
Streaming
save memory
EOF irq
EOF irq
isp_frproc0_handler
isp_sr_go
Streaming
Streaming
Streaming
SOF irq
SOF irq
isp_vic1_handler
isp_cvdr_config
cvdr_set_vp_wr_ready
save memory
mmap
isp_mmap
get base addr
ISP_IOC_FRAME_REQUEST
isp_frame_request
frame_num2Offset
get frame offset
get Stream data
writeData2File
ISP_IOC_STREAM_OFF
isp_stream_off
camera_stream_on_off
ov5645_stream_on_off
write
i2c_write_reg
write reg
camera_sensor_power
return success
ov5645_power

Attention: The red box represents the EOF of a frame received by CVDR; The blue box represents the EOF of a frame received by CVDR.

The picture shows the timing diagram:

1. After the kernel boots, go to register the device, then execute the ISP's probe, that include initialize, set the private data of the platform device, and then execute the probe of the sub-device sensor, then power on the sensor and detect whether it can be obtained sensor chipid.
2. In the test demo part, the isp device is operated by opening /dev/isp device node.
3. In the test demo part, send the ISP_IOC_STREAM_ON ioctl command to ISP device, at this time isp perform the following operations:
   - Initialize MIPI CSI channel.
   - Power up the camera sensor.
   - Initialize camera sensor register through i2c, and start streaming.
   - Initialize and configure CVDR.
   - Initialize and configure StreamRouter, and start receiving stream data from csi channel.
   - The data is sent from the Sensor to the SR through CSI, to CVDR, and finally to the memory buffer.
   - When the CVDR receives the first frame of EOF, EOF interrupt will be triggered. IRQ_MERGER2 will inform CPU and call ISP registered EOF interrupt function to configure SR and start receiving data; When CVDR receives the second frame's SOF , will trigger SOF interrupt at this moment, notify CPU by IRQ_MERGER2, and cpu will call ISP registered SOF interrupt function again, the interrupt function will configure CVDR and write the data from CVDR's W2_4 port into the memory.
4. In the test demo part, send the isp_frame_request ioctl command, ISP will return offset address of the request frame in the memory buffer; the test demo, through the mmap interface, the physical address of the memory mapped to the current process space, through the offset address to get the image of the specific address, finally the image data is written to the file.
5. In the test demo part, send the isp_stream_off ioctl command and the ISP performs stop streaming and power down the Sensor.

1.4 Sensor Drive Configuration

1.4.1 DTS Configuration

```
/* isp start */
hisiisp: isp@E8400000 {
    compatible = "hisilicon,hisi-isp","simple-bus";
    reg = <0x0 0xe8400000 0x0 0x200000>;
    interrupts = <0 262 4>, <0 266 4>;
    pinctrl-names = "default";
    pinctrl-0 = <&isp0_pmx_func &isp1_pmx_func
            &isp0_cfg_func &isp1_cfg_func
            &cam0_rst_pmx_func &cam1_rst_pmx_func
            &cam0_rst_cfg_func &cam1_rst_cfg_func
            &cam0_pwd_n_pmx_func &cam1_pwd_n_pmx_func
            &cam0_pwd_n_cfg_func &cam1_pwd_n_cfg_func>;
    clocks = <&crg_ctrl KIRIN970_CLK_GATE_ISP_SNCLK0>,
            <&crg_ctrl KIRIN970_CLK_GATE_ISP_SNCLK1>,
            <&crg_ctrl KIRIN970_CLK_GATE_ISP_SNCLK2>;
    clock-names = "clk_gate_isp_snclk0",
             "clk_gate_isp_snclk1",
             "clk_gate_isp_snclk2";
    clock-rates = <24000000>;
    DOVDD-supply = <&ldo4>;
    pool-size = <0x1000000>;
    status = "ok";

    hisi,ov5645 {
    compatible = "hisilicon,ov5645";
        sensor_index = <0>;
        pwdn-gpio = <&gpio6 5 1>;
        reset-gpio = <&gpio1 5 1>;
        status = "ok";
    };

    hisi,ov5640_main {
        compatible = "hisilicon,ov5640_main";
        sensor_index = <0>;
        pwdn-gpio = <&gpio6 5 1>;
        reset-gpio = <&gpio1 5 1>;
        status = "ok";
    };

    hisi,ov5640_sub {
        compatible = "hisilicon,ov5640_sub";
        sensor_index = <1>;
        pwdn-gpio = <&gpio3 7 1>;
        reset-gpio = <&gpio4 0 1>;
        status = "ok";
    };
   };
```

Isp is the parent device node, and the support sensor is the child device node. The child device node needs to configure sensor_index (the rear camera is 0 and the front camera is 1).

pwdn-gpio is the gpio corresponding to the sensor module powerdown pin. Reset-gpio refers to the gpio of the sensor module reset pin.

1.4.2 Add the corresponding sensor driver

1. Add the corresponding sensor's power-on interface, which is used to configure which channel and sensor clock needs to be configured. Refer to ov5645: ov5645_power (struct isp_i2c_client_t *client, int on).

2. Add sensor id interface, from which it can be used to read the sensor id and determine whether the corresponding sensor is detected, refer to ov5645_get_chip_id (struct isp_i2c_client_t * client, unsigned int * chip_id), when reading sensor id, you need to know that the sensor is mounted The i2c address, as well as the sensor id's storage specific address.

3. Add sensor register initialization sequence function, then initialize the sensor through i2c, and this sequence can be obtained from the sensor manufacturer, refer to ov5645_init (struct isp_i2c_client_t *client).

4. Add the function to set the sensor output format, and obtain the register sequence of the corresponding resolution format from the sensor manufacturer. Write the sequence to sensor through i2c to output the corresponding format data. Refer to ov5645_set_fmt (struct isp_i2c_client_t *client, Unsigned int w,unsigned int h)

5. Add the stop and start function of the sensor, and control the sensor's corresponding register address so that it starts to transmit image data or stop transmitting data. See ov5645_stream_on_off (struct isp_i2c_client_t *client, int enable)

6. Add sensor probe function, this function is used to detect the presence of sensor at power-on, if it exists, it will be registered, that the specific parameters for registering need to configure sensor are as follows. See ov5645_probe(struct platform_device *pdev).

   Need to configure the parameters:

   ```
   Idev->sensor[index].addr = OV5640_MAIN_SLAVE_ID;//mounted i2c address
   Idev->sensor[index].flag = I2C_ADDR_7BIT; //7 bit address and 1 bit read/write bit
   Idev->sensor[index].speed_cfg = I2C_SPEED_STD; //i2c read/write speed
   Idev->sensor[index].csi_index = 0; //The mounted csi index
   Idev->sensor[index].dt = YUV_DT_422_8BITS; //data format
   Idev->sensor[index].pf = DF_2PF8; //pixel format
   Idev->sensor[index].csi_lane = 2; // csi is connected via several lanes
   ```