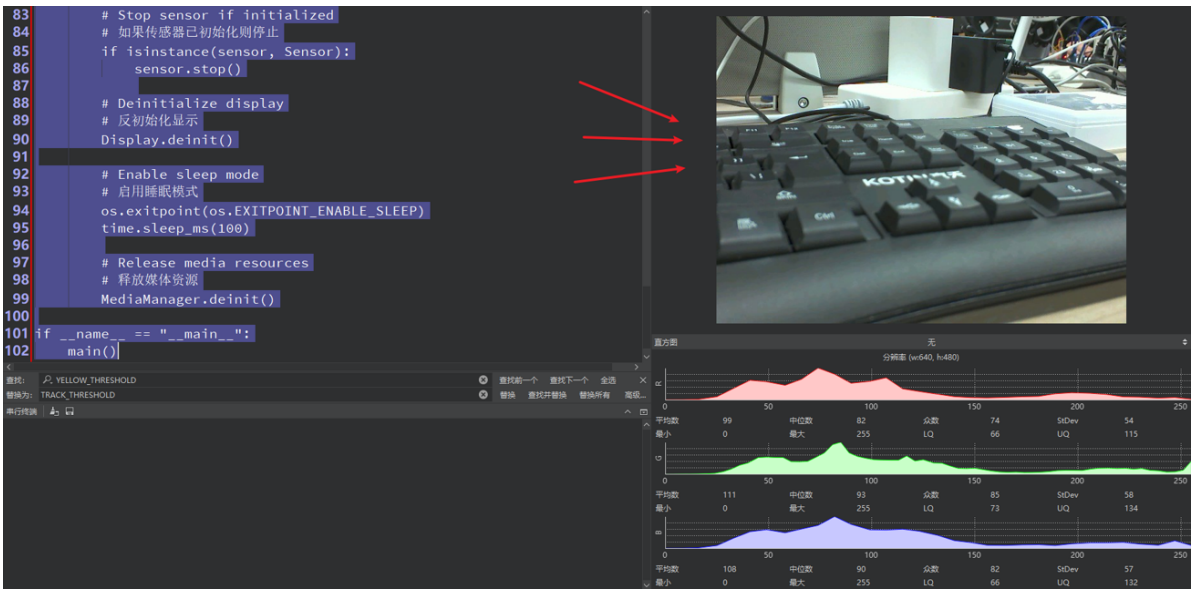
Camera display

Introduction to the results of routine experiments

In this section, we will call the camera and display the image captured by the camera in the IDE frame buffer.

We use CanMV IDE to open the sample code and connect K230 to the computer via USB.

Click the Run button in the lower left corner of the CanMV IDE, and you can see the image captured by the camera displayed in the frame buffer in the upper right corner of the IDE.



Code Explanation

The source code file for this tutorial is located in [Source Code/02.Basic/17.sensor.py]

Complete code

```
# 导入必要的模块：时间、数学、操作系统、垃圾回收、系统
# (Import necessary modules: time, math, os, garbage collection, system)
import time, math, os, gc, sys

# 导入媒体相关模块：传感器、显示、媒体管理
# (Import media-related modules: sensor, display, media manager)
from media.sensor import *
from media.display import *
from media.media import *

# 定义图像宽度和高度常量
# (Define image width and height constants)
WIDTH = 640
HEIGHT = 480

# 初始化传感器变量为空
# (Initialize sensor variable as None)
sensor = None

try:
    # 使用默认配置构造传感器对象，设置指定宽度和高度
    # (Construct a Sensor object with default configuration, setting specified
width and height)
    sensor = Sensor(width = WIDTH, height = HEIGHT, fps=30)

    # 传感器复位
    # (Reset the sensor)
    sensor.reset()

    # 设置水平镜像（当前被注释）
    # (Set horizontal mirror - currently commented out)
    # sensor.set_hmirror(False)

    # 设置垂直翻转（当前被注释）
    # (Set vertical flip - currently commented out)
    # sensor.set_vflip(False)

    # 设置通道0的输出尺寸
    # (Set channel 0 output size)
    sensor.set_framesize(width = WIDTH, height = HEIGHT)

    # 设置通道0的输出格式为RGB565
    # (Set channel 0 output format to RGB565)
    sensor.set_pixformat(Sensor.RGB565)

    # 使用IDE作为输出目标初始化显示
    # (Initialize display using IDE as output target)
    Display.init(Display.ST7701, width = WIDTH, height = HEIGHT, to_ide = True)

    # 初始化媒体管理器
    # (Initialize the media manager)
    MediaManager.init()

    # 启动传感器运行
    # (Start the sensor running)
```

```
    sensor.run()

    # 创建时钟对象用于计算帧率
    # (Create a clock object to calculate frames per second)
    fps = time.clock()

    # 主循环
    # (Main loop)
    while True:
        # 帧率计时器tick
        # (Tick the FPS timer)
        fps.tick()

        # 检查是否应该退出程序
        # (Check if the program should exit)
        os.exitpoint()

        # 从传感器获取一帧图像
        # (Capture a frame from the sensor)
        img = sensor.snapshot()

        # 在屏幕上显示结果图像
        # (Display the resulting image on screen)
        Display.show_image(img)

        # 执行垃圾回收
        # (Perform garbage collection)
        gc.collect()

        # 打印当前帧率
        # (Print the current frames per second)
        print(fps.fps())

except KeyboardInterrupt as e:
    # 捕获键盘中断异常（用户手动停止）
    # (Catch keyboard interrupt exception - user manually stops)
    print(f"user stop")
except BaseException as e:
    # 捕获所有其他异常
    # (Catch all other exceptions)
    print(f"Exception '{e}'")
finally:
    # 无论如何都执行清理工作
    # (Perform cleanup regardless of how the program exits)

    # 停止传感器运行（如果传感器对象存在）
    # (Stop the sensor if the sensor object exists)
    if isinstance(sensor, Sensor):
        sensor.stop()

    # 反初始化显示
    # (Deinitialize the display)
    Display.deinit()

    # 设置退出点，允许进入睡眠模式
    # (Set exit point to enable sleep mode)
    os.exitpoint(os.EXITPOINT_ENABLE_SLEEP)
```

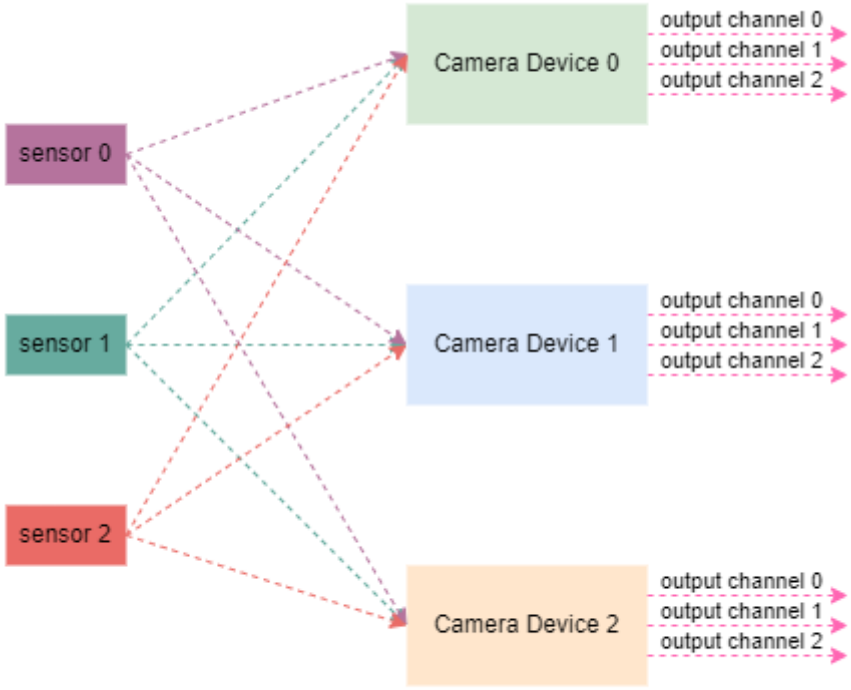
```
    # 短暂延时100毫秒
    # (Brief delay of 100 milliseconds)
    time.sleep_ms(100)

    # 释放媒体缓冲区
    # (Release media buffer)
    MediaManager.deinit()
```

Camera related API

> The following content is excerpted from the CanMV Sensor API chapter

The module of the CanMV K230 platform `sensor` is responsible for image acquisition and data processing. The module provides a set of high-level APIs that developers can use to easily obtain images of different formats and sizes without having to understand the specific implementation of the underlying hardware.



K230 CanMV Camera示意图

Sensor 0, sensor 1, and sensor 2 represent three image input sensor devices; Camera Device 0, Camera Device 1, and Camera Device 2 correspond to the corresponding image processing units; output channel 0, output channel 1, and output channel 2 indicate that each image processing unit supports up to three output channels. Through software configuration, different sensor devices can be flexibly mapped to the corresponding image processing units.

The module of CanMV K230 `sensor` supports the simultaneous access of up to three image sensors, each of which can independently complete the acquisition, capture and processing of image data. In addition, each video channel can output three channels of image data in parallel for further processing by the back-end module. In actual applications, the specific number of sensors supported, input resolution and number of output channels will be limited by the hardware configuration and memory size of the development board, so a comprehensive evaluation is required based on project requirements.

Constructor

describe

`csi id` Construct an object from a and an image sensor type `Sensor`.

In image processing applications, users usually need to create an `Sensor` object first. CanMV K230 software can automatically detect the built-in image sensor, without the user manually specifying the specific model, just set the maximum output resolution and frame rate of the sensor. For information about supported image sensors, see the image sensor support list . If the set resolution or frame rate does not match the default configuration of the current sensor, the system will automatically adjust to the optimal configuration. The final configuration can be viewed in the log, for example `use sensor 23, output 640x480@90`.

grammar

```
sensor = Sensor(id, [width, height, fps])
```

parameter

Parameter name	describe	Input/Output	illustrate
id	`csi` Port, support `0-2`, please refer to the hardware schematic diagram for specific ports	enter	Optional. The default value is different for different development boards.
width	`sensor` Maximum output image width	enter	Optional, default `1920`
height	`sensor` Maximum output image height	enter	Optional, default `1080`
fps	`sensor` Maximum output image frame rate	enter	Optional, default `30`

Return Value

Return Value	describe
Sensor Object	Sensor Object

Example

```
sensor = Sensor(id=0)
sensor = Sensor(id=0, width=1280, height=720, fps=60)
sensor = Sensor(id=0, width=640, height=480)
```

2.1 sensor.reset

describe

Resets `sensor` the object. `Sensor` You must call this function after constructing the object to continue executing other operations.

grammar

```
sensor.reset()
```

parameter

Parameter name	describe	Input/Output
none		

Return Value

Return Value	describe
none	

Example

```
# Initialize sensor device 0 and sensor OV5647
sensor.reset()
```

2.2 sensor.set_framesize

describe

Sets the output image size for the specified channel. The user can configure the output image size via `framesize` the parameter or by specifying `width` and directly. **The width is automatically aligned to 16 pixels wide** . `height`

grammar

```
sensor.set_framesize(framesize=FRAME_SIZE_INVALID, chn=CAM_CHN_ID_0, alignment=0,
**kwargs)
```

parameter

Parameter name	describe	Input/Output
framesize	Sensor output image size	enter
chn	Sensor output channel number	enter
width	Output image width, *kw_arg*	enter
height	Output image height, *kw_arg*	enter

Return Value

Return Value	describe
none	

Precautions

- The output image size must not exceed the actual output capability of the image sensor.
- The maximum output image size for each channel is limited by the hardware.

Example

```
# Configure sensor device 0, output channel 0, output image size 640x480
sensor.set_framesize(chn=CAM_CHN_ID_0, width=640, height=480)

# Configure sensor device 0, output channel 1, output image size 320x240
sensor.set_framesize(chn=CAM_CHN_ID_1, width=320, height=240)
```

2.3 sensor.set_pixformat

describe

Configure the image sensor output image format for the specified channel.

grammar

```
sensor.set_pixformat(pix_format, chn=CAM_CHN_ID_0)
```

parameter

Parameter name	describe	Input/Output
pix_format	Output image format	enter
chn	Sensor output channel number	enter

Return Value

Return Value	describe
none	

Example

```
# Configure sensor device 0, output channel 0, output NV12 format
sensor.set_pixformat(sensor.YUV420SP, chn=CAM_CHN_ID_0)

# Configure sensor device 0, output channel 1, output RGB888 format
sensor.set_pixformat(sensor.RGB888, chn=CAM_CHN_ID_1)
```

2.4 sensor.set_hmirror

describe

Configure whether the image sensor is horizontally mirrored.

grammar

```
sensor.set_hmirror(enable)
```

parameter

Parameter name	describe	Input/Output
enable	`True` Enable horizontal mirror function `False` Disable horizontal mirror function	enter

Return Value

Return Value	describe
none	

Example

```
sensor.set_hmirror(True)
```

2.5 sensor.set_vflip

describe

Configure whether the image sensor is flipped vertically.

grammar

```
sensor.set_vflip(enable)
```

parameter

Parameter name	describe	Input/Output
enable	`True` Turn on the vertical flip function `False` Turn off the vertical flip function	enter

Return Value

Return Value	describe
none	

Example

```
sensor.set_vflip(True)
```

2.6 sensor.run

describe

Enables output of the image sensor. **This must be done before calling .** `MediaManager.init()`

grammar

```
sensor.run()
```

Return Value

Return Value	describe
none	

Precautions

- When multiple sensors (up to 3) are used simultaneously, only one of them `run` needs to be executed.

Example

```
# Start the sensor device output data stream
sensor.run()
```

2.7 sensor.stop

describe

Stops image sensor output. **This method must be called before .** `MediaManager.deinit()`

grammar

```
sensor.stop()
```

Return Value

Return Value	describe
none	

Precautions

- If multiple image sensors are used simultaneously (maximum 3), **each sensor must be called separately** `stop` .

Example

```
# Stop the data stream output of sensor device 0
sensor.stop()
```

2.8 sensor.snapshot

describe

Capture a frame of image data from the specified output channel.

grammar

```
sensor.snapshot(chn=CAM_CHN_ID_0)
```

parameter

Parameter name	describe	Input/Output
chn	Sensor output channel number	enter

Return Value

Return Value	describe
image object	Captured image data
other	Capture failed

Example

```
# Capture a frame of image data from channel 0 of sensor device 0
sensor.snapshot()
```