# User Manual

**Overview**

In Linux, this camera uses the UVC protocol and can be enabled through the V4L2 module. When compiling the Linux system, the V4L2 module should be added. VMware and HUB are not recommended because data transmission error may occur due to large data volume. However, VMware is applicable for some computers if USB3.0 is available.

**Quick Application on a PC with Linux System**

Install OpenCV 4.x version.

Copy the library file of the corresponding platform in the folder named with the cross-compiler to this file directory.

Run the following command in terminal for demonstration:

g++ main.cpp Usb.cpp -L. -lthermometry -lSimple -lm -lpthread -lpot -lusb-1.0 -o main1 `pkg-config opencv4 --libs --cflags`

export LD\_LIBRARY\_PATH=./

./main1

Input **a** (do not press **Enter**) in the terminal to start the device. An image is displayed and temperature measuring starts. Input **s** to stop the device.

In this example, OpenCV has enabled YUYV-RGB conversion and display functions, which are not mandatory for migration.

**Module Introduction**

libthermometry.so thermometry.h: Temperature measuring module

libSimple.so SimplePictureProcessing.h: Image processing module

libpot.so pot.h Usb.cpp Usb.h: Corona removing module (applicable to 384×288 array. The module outputs YUYV data.)

**Introduction to Main.cpp codes**

main(): Creates the interface and interaction button, listens to the keyboard buttons, and sends data to handlerThread for processing

handlerThread: Performs processing according to input buttons

renderThread: Provides V4L2-based display according to UVC transmission

bulkThread: Provides corona removing (applicable to 384×288 array. The module outputs YUYV data.) and uses bulk for transmission, mutually exclusive with UVC

**Introduction to handlerThread**

Enter key: Shutter calibration. When pressing the key, the device shutter will open and close once, upon which you will hear a click. Timed shutter calibration ensures clear image and accurate temperature measurements.

0: Start the bulk thread and calibrate the corona

1: Collect data before corona calibration

2: Validate corona calibration

3: Invalidate corona calibration

4: Save parameters: Save configured temperature measurement parameters. With this option, parameters will be retained if power outage occurs. Otherwise, the parameters will be rolled back to the previous state upon power outage.

5: Example of setting the first extra point in 8005 mode (see "Introduction to renderThread" for the difference between 8004 and 8005)

6: Example of setting the second extra point in 8005 mode

7: Example of setting the third extra point in 8005 mode

q: Switch to white hot palette in 8004 mode

w: Switch to black hot palette in 8004 mode

e: Switch to iron rainbow palette in 8004 mode

r: Switch to rainbow 1 palette in 8004 mode

t: Switch to rainbow 2 palette in 8004 mode

y: Switch to high-dynamic rainbow palette in 8004 mode

u: Switch to high-contrastive rainbow palette in 8004 mode

i: Switch to lava rainbow palette in 8004 mode

o: Switch to the second iron rainbow palette in 8004 mode

p: Switch displayMode in 8004 mode when computeMethod is set to ComputeDivTempType2 (see ComputeDivTempType2 introduced in "Introduction to renderThread")

a: Create renderThread, start the UVC device, and enable display

b: Shut down the UVC device

d: Switch computeMethod in 8004 mode circularly (see computeMethod introduced in "Introduction to renderThread")

f: Switch to high temperature section for measuring

g: Switch to normal temperature section for measuring

h: Set temperature measurement parameters (see Code Notes for parameter functions)

z: Switch the palette in 8005 mode, which is different from that in 8004 mode

**Introduction to renderThread**

**V4l2 related:**

int init\_v4l2(void);//Initialize v4l2. Note whether the infrared device has been started

int v4l2\_grab(void);//Collect data

int v4l2\_control(int);//Control the camera

int v4l2\_release();//Release

*Tips on length and width:*

The last four (288-292) lines of the image are used to transfer parameters. For example, for the T3 series, the original resolution is 384×288, and must be set to 384×292 for data collection.

**Camera control and related configuration:**

The module control command adopts the Zoom (Absolute) control channel of the UVC protocol. The control command contains 2 bytes, the low byte being the parameter value and the high one being the register address.

For example, call v4l2\_control(0x8000) to send the shutter calibration command 0x8000.

The following table lists the detailed command numbers (for more parameters, see the product specification):

|  |  |  |  |
| --- | --- | --- | --- |
| S/N | Register Address | Parameter | Description |
| 1 | 0x80 | 0x00 | Shutter calibration |
| 2 | 0x01 | Background calibration |
| 3 | 0x02 | Original detector output |
| 4 | 0x04 | 16bit NUC raw data |
| 5 | 0x05 | YUYV data output |
| 6 | 0x20 | Normal temperature measurement |
| 7 | 0x21 | High temperature measurement |
| 8 | 0xFE | Save configuration parameters (data source switching, module palette) |
| 9 | 0xFF | Save temperature measurement parameters |
| 10 | 0x88 | 00-07 | Palette |

1. Shutter calibration: Perform non-uniformity calibration. The shutter will be calibrated after a command is sent. This operation can make the image uniform and the temperature measurement accurate. Shutter calibration is performed once every three minutes for industrial temperature measurement and once a minute for body temperature measurement. Thermometry calculation (see the temperature measurement below) is performed one second after shutter calibration (the frame rate of infrared camera is 25 frames).
2. Background calibration: The camera performs shutter calibration in a uniform scene.

3． Difference between 0x8004 and 0x8005:

(1) In 0x8004 mode, the output format is YUYV but 16bit NUC (RAW) data is yielded in fact. The wide dynamic grayscale value of a point is stored in 14 bits in each continuous 16 bits. In 0x8005 mode, data is output in YUYV format.

(2) In 0x8004 mode, the raw data is processed by libthermometry.so and is converted into global temperature data. The data is then further processed by libSimple.so or linear algorithm (see the main code) and converted into rgba data. In 0x8005 mode, YUYV data is provided. The following information is obtained after libthermometry.so processing on the last four lines of parameters: Temperature at the central point, highest temperature point and position, lowest temperature point and position. Temperature of three customized points is provided but not global temperature data.

**Temperature measurement library related:**

Temperature measurement consists of two steps:

Calculate the thermometer, and then check the data table with 16bit raw data. thermometryT4Line is used to operate the thermometer and thermometrySearchXXX is used to perform table query. For details, see the note of thermometry.h.

Set temperature measurement parameters:

Set the value of h in handlerThread. After setting, recalculate the thermometer. Confirm and save the parameters. Saved parameters will not lose if power outage occurs. Otherwise, parameter loss will occur.

Normal and high temperature sections:

For industrial temperature measurement, the normal temperature range is -20°C to 120°C, and the high temperature range is 120°C to 400°C (slightly different among products). Some products do not support the high temperature measurement. The product specification prevails. It takes about 10 minutes for system stabilizing after switching to the high-temperature section for temperature measurement (depending on products). To obtain clear images and temperatures quickly, in addition to press g for shutter calibration and thermometry calculation of handlerThread, you can choose to perform shutter calibration and thermometry calculation once a minute in the first three minutes after switching and then perform shutter calibration and thermometry calculation every three minutes. The high temperature section switching operation can not be saved in case of power failure, and you need to switch after every startup.

**Image processing related:**

Before using the image algorithm in libSimple.so, initialize the high-performance algorithm first, and then release resources.

In 8004 mode:

In case of computeMethod == 1, use the best-effect image algorithm Compute, and main frequency higher than 1.8ghz is recommended. Recommend Setparameter(100,0.5f,0.1f,0.1f,1.0f,3.5f) for parameter settings and no modification;

In case of computeMethod==2, use general-effect image algorithm ComputeMethodTwo. There is no mandatory requirement for the main frequency. Recommend SetMethodTwoParameter(13,0,3,3,50,50) for parameter settings;

In case of computeMethod==3, ComputeDivTemp is used by combining ComputeMethodTwo and SetMethodTwoParameter. Based on SetDivTemp(divTmpNuc1,divTmpNuc2), black-white image is displayed under divTmp1 Celsius, yellow-red image is displayed between divTmp1 Celsius and divTmp2 Celsius, and red image is displayed over divTmp2 Celsius. Reset SetDivTemp each time you perform thermometry calculation with thermometryT4Line because the operation is temperature related.

In case of computeMethod==4, ComputeDivTempType2 is used based on optimization of ComputeDivTemp. Select divTmp1 Celsius and lower, the lowest temperature is displayed. Select divTmp2 Celsius and higher, the highest temperature is displayed. Between divTmp1 Celsius and divTmp2 Celsius, the palette is displayed in transition. With lowTmpPaletteIndex, you can set the lowest temperature on the palette. In case of displayMode==1, data is displayed in binarization of divTmp1. In case of displayMode==2, data is displayed in ternary of divTmp1 and divTmp2.

In case of computeMethod==0, the linear algorithm is used for display.

About the palette:

(1) In 0x8005 mode, you can switch palette by sending 0x88 (00-07). Eight palettes are available: White hot, black hot, iron rainbow, lava, rainbow, iron gray, red hot, and rainbow 2.

(2) In 0x8004 mode, you can switch the palette by referring to handlerThread.

**Introduction to bulkThread**

This thread provides corona removing and is applicable to 384×288 array and the module outputs YUYV data. It uses bulk for transmission and is mutually exclusive with UVC.

1. Obtain access permission to USB

sudo vi /etc/udev/rules.d/50-myusb.rules

SUBSYSTEMS=="usb", ATTRS{idVendor}=="1514", ATTRS{idProduct}=="ffff", GROUP="jun", MODE="0666"

sudo udevadm control --reload

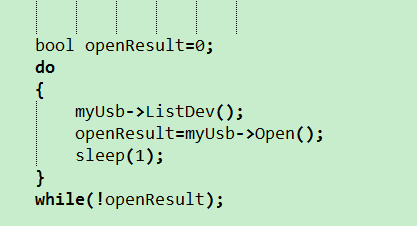
If subsequent prompt indicates that no permission is obtained and the device can not start, plug in or out the device or power off and restart.

Fill in Group in according to actual conditions.

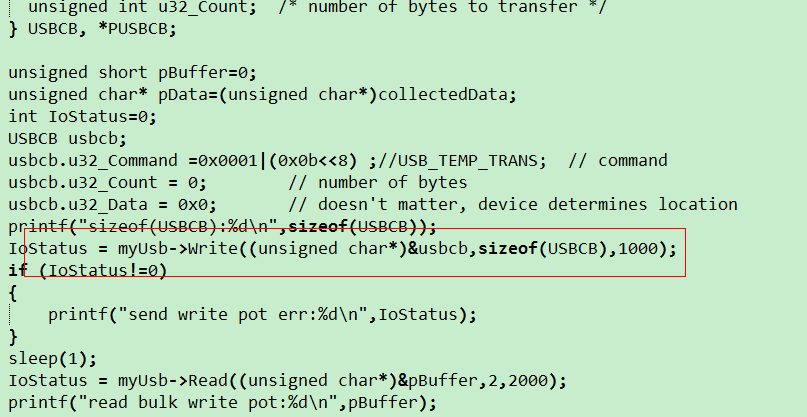
In UVC state, vid is 1514 and pid is 0001.

In bulk state, vid is 1514 and pid is ffff.

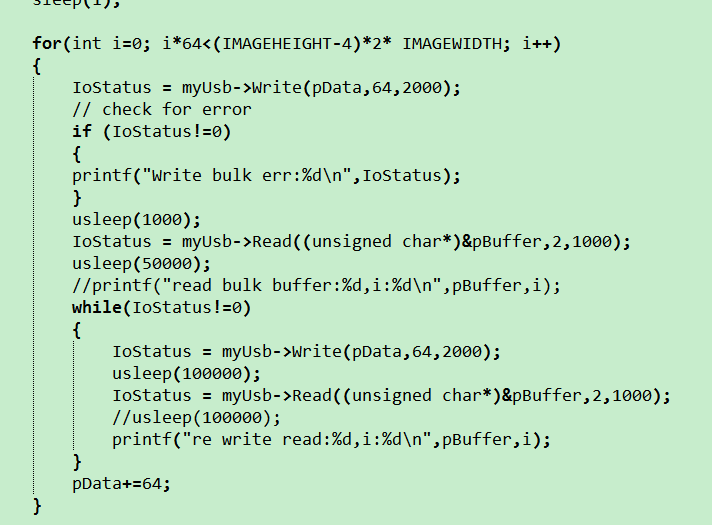
2. In UVC 8004 mode, input **1** (do not press **Enter**) in the terminal, collect data, and process 16-frame data (colorData function).

Input **0** to switch to bulk mode and exit to release uvc resources.

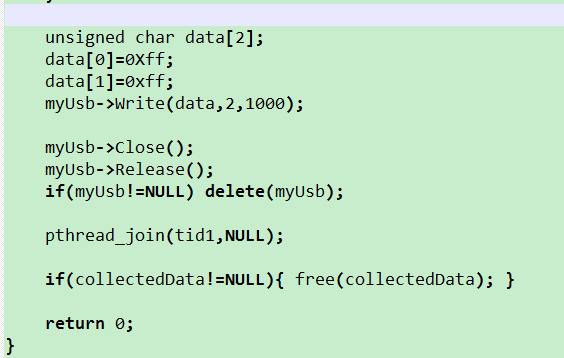
It takes time to switch to bulk mode and you need to continue the cycle until switching finishes. (There is slight probability that the device can not be mounted. If the 1514 device can not be found for a long time, plug or unplug or restart after power-off. A counter can be added in this cycle, and set the count of failures. If the count is greater than a certain value, mounting is not successful).



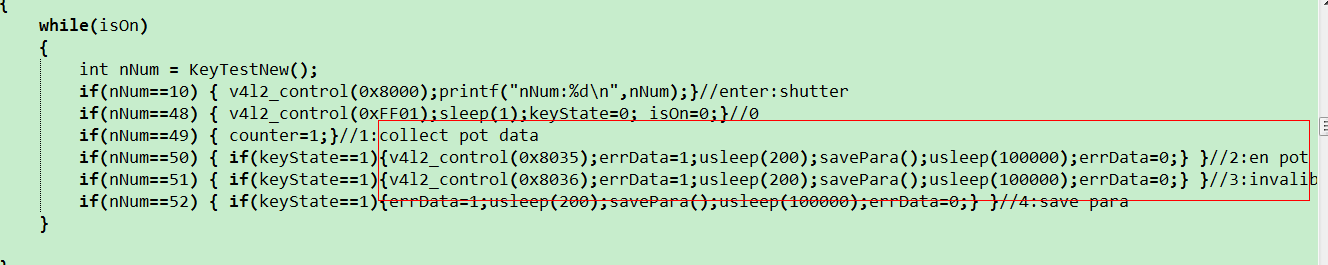
Bulk sends a command to request the device to write corona data.



The value returned by pBuffer is the same as i.



Bulk sends the command, notifying to switch back to UVC and release all bulk resources.



Power off and restart and enable the display in UVC 8005 mode. Input **2** to enable the function of removing corona. Save the setting. The saved setting will not lose upon power failure.

Input **3** to disable the function of removing corona and save the setting.