SOFTWARE PORTING GUIDE

(DOC No. SLiM-SH430UH-SWPG(USB 2.0_Linux))

SLiM-SH430UH(USB 2.0_Linux)

Structured Light Module System with 3D Sensing IC

Preliminary version 01 April, 2021

Himax Technologies, Inc.
http://www.himax.com.tw

Revision History

April, 2021

Version	Date	Description of changes
01	2021/04/25	New setup.

Himax Confidential
Do Not Copy

List of Contents

April, 2021

Himax Confidential
Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

List of Figures

Himax Confidential
Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

1. System Block Diagram

SLiM-SH430UH USB Dongle would be connected to x86_64 Linux PC with USB **(USB 2.0 High Speed)** micro-B to Type-A cable as the below diagram:

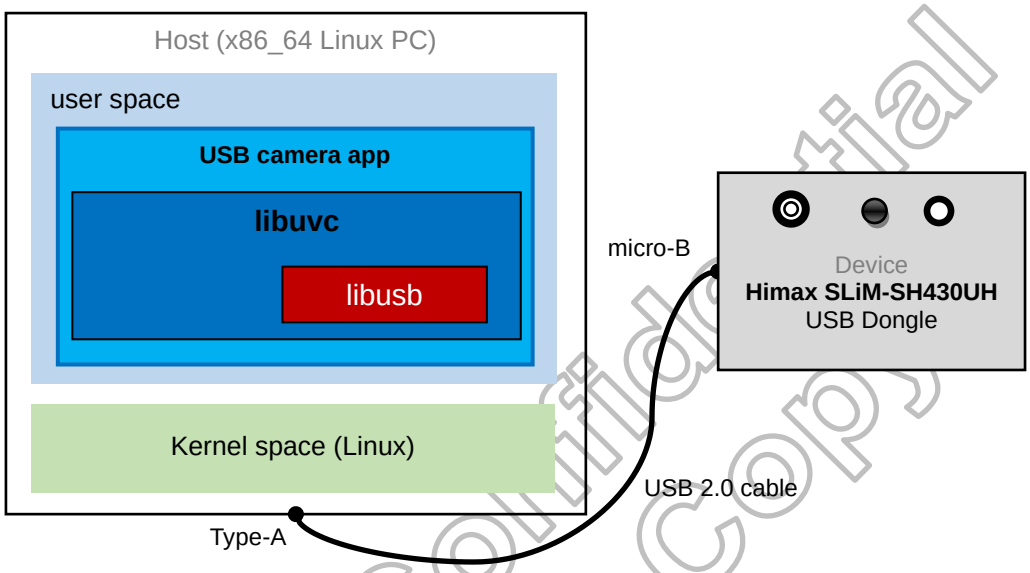

Figure 1.1: System block diagram

Himax Confidential
Do Not Copy

Figure 1.2: SLiM-SH430UH USB Dongle physical pictures

In this case, we will provide a prebuilt shared library “libuvc” and a UVC camera sample code “test.c” for user reference.

More library function call descriptions and porting details will be described in later sections.

Himax Confidential
Do Not Copy

2. Video Streaming Information

2.1. Video frame sequence

SLiM-SH430UH Camera module simultaneously provides NIR **(Near-Infrared Radiation)** frame and Depth frame in the same video streaming, those two types of frames were arranged in interlaced frame sequence and looping, e.g., if 1st frame is NIR frame, then 2nd frame is Depth frame, 3rd frame is NIR frame, 4th frame is Depth frame, ..., and so on. About this mode, we also call it “Alternative Mode”, the following diagram is the visualized representation of the frame sequence:

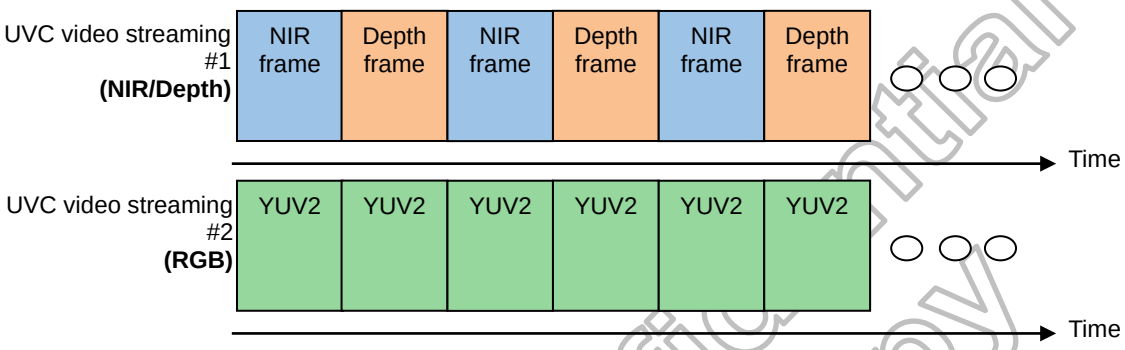

Figure 2.1: Video frame sequence

Himax Confidential
Do Not Copy

- **First** UVC video streaming interface, which contains the interlaced **NIR/Depth** frames, supports two resolutions, **1280x800@30fps** and **640x400@30fps**.

  - NIR pixel format is raw8x2 with 10-bit effective value presents the relative luminance.

Pixel format of 2D NIR [16-bit data / 10-bit valid]															
MSByte								LSByte							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	x	x	x	x	x	x

Figure 2.2: NIR pixel format representation

    - Each NIR 10-bit pixel data would be received as 2 bytes **(16-bit)** in little endian byte order with zero-padding in least-significant 6-bit. So, the user may need to do right-shifting 6-bit on every pixel to get the correct range of NIR luminance they want.

  - Depth pixel format is raw8x2 with 16-bit **(Q12.4)** effective value presents the absolute distance **(Unit: 1/16 mm)**.

Pixel format of 3D Depth [16-bit data / Q12.4]															
MSByte								LSByte							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	d3	d2	d1	d0
12-bit integer												4-bit fraction			

Figure 2.3: Depth pixel format representation

    - Each depth 16-bit **(Q12.4)** pixel data would be received as 2 bytes **(16-bit)** in little endian byte order which contains most-significant 12-bit binary integer and least-significant 4-bit binary fraction. So, the user may need to do right-shifting 4-bit on every pixel if user want the integer part only.

- **Second** UVC video streaming interface on the same UVC video device as NIR/Depth, which contains **RGB** frame, supports two resolutions, **1280x800@15fps** and **640x480@30fps**. And, the RGB pixel format is **YUYV (YUV2 422 packed)**.

  - Beware of the **RGB** frame image may be **rotated 180˚** relative to NIR/Depth frame image.

Himax Confidential Do Not Copy

2.2. Video streaming settings for opening UVC device

A. User must open **NIR/Depth sensor** on the first video streaming interface on the UVC device by requiring the target video format/resolution as "**1280x801@30fps,YUYV**" or "**640x401@30fps,YUYV**" on the UVC video streaming **interface #1**. Please, be aware that the captured frame size is different from image presentation size. Open camera with this video setting is the criterion to access the correct video streaming of this UVC device.

B. User must open **RGB sensor** on the second video streaming interface on the same UVC device as NIR/Depth by requiring the target video format/resolution as "**1280x801@15fps,YUYV**" or "**640x481@30fps,YUYV**" on the UVC video streaming **interface #2**. Please, be aware that the captured frame size is different from image presentation size. Open camera with this video setting is the criterion to access the correct video streaming of this UVC device.

Note: (1) If opened video streaming on interface #1 or #2, with HD **(1280x801)** resolution, the real frame rate will be about 6~7 fps. This situation is caused by the limitation of USB 2.0 bandwidth.

Himax Confidential
Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

3. Himax UVC Library API Call Flow and Usage

In this section, we will see what the call flow is and how to use the UVC library. In addition to the vanilla libuvc, Himax added some custom APIs for frame decoding, sensor info reading, sensor-related controls, and, some utility functions.

3.1. UVC library API call flow

Generally, the API call flow of Himax-customized libuvc is similar to the vanilla libuvc.

- Open & Start device procedure: Init → find device → open device → configure format/size → start streaming...
- Stop & Close device procedure: Stop streaming → close → unref device → exit.

Due to two UVC video streaming interfaces - NIR/Depth and RGB - are on the same UVC device, user only need to call the uvc_open() API once to get only one UVC device handle, instead of two. And, then user can get two UVC stream control blocks for NIR/Depth and RGB video streamings individually by calling the uvc_get_stream_ctrl_format_size_ifno() API twice with different arguments, which is Himax-customized variant API of the vanilla uvc_get_stream_ctrl_format_size() API plus specifying the video streaming interface number.

- For NIR/Depth video streaming: uvc_get_stream_ctrl_format_size_ifno(devh, &nir_ctrl, UVC_FRAME_FORMAT_YUYV, 640, 401, 30, 1);
- For RGB video streaming: uvc_get_stream_ctrl_format_size_ifno(devh, &rgb_ctrl, UVC_FRAME_FORMAT_YUYV, 640, 481, 30, 2);
  - The last argument 1 or 2 is used to specify the video streaming interface number we wanted.

Additionally, Himax already pre-parsed the NIR/Depth frames in the first UVC video streaming, and registered in different frame callback functions by calling uvc_start_streaming_multi_cb() API, which is Himax-customized multi-callback variant API of the vanilla uvc_start_streaming() API. In consequence, NIR frame and depth frame will be placed in different frame buffer of dedicated user frame callback functions.

- For NIR/Depth video streaming: uvc_start_streaming_multi_cb(devh, &nir_ctrl, UVC_FRAME_FORMAT_YUYV, nir_cb, NULL, depth_cb, NULL, dot_cb, NULL, 0);
  - nir_cb is the user frame callback function for NIR frame, and so on.

Furthermore, Himax SLiM-SH430UH camera device supports dynamically changing two resolution settings between stop-streaming and start-streaming step, no need to close and open device again.

- Stop streaming → configure another format/size → start streaming.

Please check the example code test.c source file for more details.

Himax Confidential Do Not Copy

3.2. UVC library API usage

A. uvc_init(&ctx, NULL);

Initial a UVC context.

- If it initialized successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to initialize a USB context for UVC.
- 2nd argument usb_ctx is set to NULL for using UVC library owned USB context.

B. uvc_find_device(ctx, &dev, 0x2AAD, 0x6373, NULL);

Find the first UVC device which matches the VID:PID specified in third and fourth arguments.

- This function need an available UVC context, so it must be called after uvc_init() function called successfully.
- If it found successfully, then it'll return UVC_SUCCESS (0), else if device not found, then it'll return UVC_ERROR_NO_DEVICE (-4), otherwise it's failed to get any UVC device from USB device list.
- 1st argument ctx is the UVC context initialized by uvc_init() function.
- 2nd argument dev will output the UVC device reference if found, or NULL if not found.
- 3rd and 4th arguments are device's vid and pid specified by user.
  - SLiM-SH430UH camera device have only one dedicated UVC device ID, which its VID:PID is **2AAD**:**6373** **(in hexadecimal)**, please do not change this value. Both NIR/Depth and RGB video streaming interfaces are on the same UVC device.
- 5th argument is device's sn, currently we set to NULL for ignore this condition as us propose.

C. uvc_open(dev, &devh);

Open the specified UVC device and return the UVC device handle.

- This function need an available UVC device, so it must be called after uvc_find_device() function called successfully.
- If it opened successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to open the UVC device.
- 1st argument dev is the UVC device to be opened.
- 2nd argument devh will output the UVC device handle reference once it opened successfully.

Himax Confidential Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

D. uvc_hx_set_frame_mode(devh, mode_id);

Select Himax frame mode function for changing the frame sequence of the first UVC video streaming interface for NIR/depth video streaming **(Not applied to the second UVC video streaming interface for RGB video streaming)**, instead of the default 2D(NIR)/(3D) Depth alternative mode.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it set successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to set.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument mode_id value is defined in the enum hx_frame_mode_select in the libuvc_hx.h header file, currently only support following modes.
  - HX_FRAME_MODE_SELECT_ONLY_NIR **(NIR only)**
  - HX_FRAME_MODE_SELECT_ONLY_DEPTH **(Depth only)**
  - HX_FRAME_MODE_SELECT_ALT_NIR_DEPTH **(Default 2D(NIR)/(3D) Depth alternative mode)**

E. uvc_hx_get_frame_mode(devh, mode_id);

Get NIR/Depth video streaming after selecting frame mode via uvc_hx_set_frame_mode().

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it set successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to set.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument mode_id value is a pointer pointed to 8-bit integer, presents as a positive integer, likes "4" or similar values. The mode_id value is defined in the enum hx_frame_mode_select in the libuvc_hx.h header file, currently only support following modes.

F. uvc_hx_get_hv2_fw_version(devh, &hv2_fw_version);

Read the HV2 FW Version data from Himax UVC camera device.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it read successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to read.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument hv2_fw_version is a pointer pointed to a Himax custom data structure hx_hv2_fw_version_t, which contains the chip_id/version_major /data**(YYMMDD)**/version_minor/customer_id fields, please check the libuvc_hx.h header file for more details.

Himax Confidential Do Not Copy

G. uvc_hx_get_build_type(devh, &build_type);

Read the build type from Himax UVC camera device.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it read successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to read.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument build_type is a pointer pointed to a 16-bit integer, presents as four hexadecimals string, likes "D1C1" or similar values; otherwise "0xFFFF" **(16-bit with all "ones")** means the UVC device's build type hasn't been provisioned yet.

H. uvc_hx_get_serial_number(devh, &serial_number);

Read the serial number **(SN)** from Himax UVC camera device.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it read successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to read.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument serial_number is a pointer pointed to a 32-bit integer, presents as a positive integer, likes "14" or similar values; otherwise "0xFFFFFFFF" **(32-bit with all "ones")** means the UVC device's serial number hasn't been provisioned yet.

Himax Confidential
Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

I.uvc_hx_get_intrinsic_extrinsic_matrices(devh, &cam_matrices, 0-or-1);

Read the camera's intrinsic and extrinsic parameter matrices from Himax UVC camera device.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it read successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to read.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument cam_matrices is a pointer pointed to a Himax custom data structure hx_cam_intrinsic_extrinsic_matrices_t, which contains the target NIR sensor width/height w_d/h_d, target RGB sensor width/height w_r/h_r, NIR sensor's intrinsic matrix Kd[3][3], and it's inverse matrix Kd_[3][3], RGB sensor's intrinsic matrix Kr[3][3], rotation component R[3][3] and translation component T[3] of the extrinsic matrix translated from NIR to RGB, please check the libuvc_hx.h header file for more details.
- 3rd argument require_scaling is used to select right camera intrinsic/extrinsic parameter matrices for different target video resolution:
  0: Output camera matrices of **HD** resolution,
  (NIR/Depth: 1280x800, RGB:1280x800),
  and, set NIR/Depth frame size w_d/h_d to 1280/800,
  and, set RGB frame size w_r/h_r to 1280/800.
  1: Output camera matrices of **VGA** resolution,
  (NIR/Depth: 640x400, RGB: 640x480) which is **½ downscaling**,
  and, set NIR/Depth frame size w_d/h_d to 640/400,
  and, set RGB frame size w_r/h_r to 640/480.
- Note: This function will also do pre-calculating the inverse matrix of NIR sensor's intrinsic matrix (Kd_[3][3]), that can be used in another Himax utility function uvc_hx_rectified_coordinate() API, which is used for converting NIR to RGB coordinates.

Himax Confidential
Do Not Copy

J. uvc_hx_rectified_coordinate(&cam_matrices, depth, x_nir, y_nir, &x_rgb, &y_rgb);

Himax utility function for converting a coordinate of portrait image from NIR to RGB image coordinate, input the NIR sensor's image coordinate (x_{nir}, y_{nir}) and depth value Z_{nir}, and output the RGB sensor's image coordinate (x_{rgb}, y_{rgb}).

- This function required SLiM-SH430UH NIR & RGB camera sensors' intrinsic and extrinsic matrices (Kd/Kr/R/T) and the pre-calculated inverse matrix of NIR sensor's intrinsic matrix (Kd_), so it must be called after uvc_get_hx_cam_intrinsic_extrinsic_matrices() function called successfully.
- Since, each Himax SLiM-SH430UH camera device has stored different calibrated intrinsic/extrinsic matrices data to each other. Do NOT use the same camera matrices data on the mismatch camera device.
- If it computed successfully, then it'll return 0, otherwise it's failed to compute.
- 1st argument cam_matrices is a pointer pointed to a Himax custom data structure hx_cam_intrinsic_extrinsic_matrices_t, which contains the target NIR sensor width/height w_d/h_d, target RGB sensor width/height w_r/h_r, NIR sensor's intrinsic matrix Kd[3][3], and it's inverse matrix Kd_[3][3], RGB sensor's intrinsic matrix Kr[3][3], rotation component R[3][3] and translation component T[3] of the extrinsic matrix translated from NIR to RGB, please check the libuvc_hx.h header file for more details.
- 2nd argument depth is input the NIR sensor's depth value Z_{nir}.
- 3rd and 4th arguments x_nir and x_nir are input the NIR sensor's image coordinate (x_{nir}, y_{nir}).
- 5th and 6th arguments x_rgb and y_rgb are output the RGB sensor's image coordinate (x_{rgb}, y_{rgb}).

Himax Confidential
Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

K. **// for NIR/Depth**
uvc_get_stream_ctrl_format_size_ifno(devh, &nir_ctrl,
UVC_FRAME_FORMAT_YUYV, 1280-or-640, 801-or-401, 30, 1);
// for RGB
uvc_get_stream_ctrl_format_size_ifno(devh, &rgb_ctrl,
UVC_FRAME_FORMAT_YUYV, 1280-or-640, 801-or-481, 15-or-30, 2);

Get a negotiated video streaming control block with some common parameter user specified, such as, frame format, frame width/height, frame rate (fps), plus video streaming interface number. This is a Himax-customized variant API of the vanilla uvc_get_stream_ctrl_format_size() API plus specifying the video streaming interface number.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- If it configured successfully, then it'll return UVC_SUCCESS (0), otherwise it'll return UVC_ERROR_INVALID_MODE (-51) means it's failed to configure.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument ctrl is a pointer pointed to the output result UVC stream control block.
- 3rd argument format is the frame format.
- 4th and 5th arguments width and height are the frame size.
- 6th argument fps is the frame rate per second.
- 7th argument ifno is the UVC video streaming interface number.

L. **uvc_start_streaming(devh, &rgb_ctrl, rgb_cb, NULL, 0);**

Start video streaming with setting up a callback function rgb_cb for RGB frame.

- This function need an available UVC device handle and UVC stream control block, so it must be called after uvc_get_stream_ctrl_format_size() or uvc_get_stream_ctrl_format_size_ifno() function called successfully.
- If it started successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to start.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument ctrl is a pointer pointed to the UVC stream control block.
- 3rd argument cb is the user frame callback function to be registered, and the callback frame will be placed in the 1st argument (uvc_frame_t *frame).
- 4th argument user_ptr is a pointer that will be placed in the 2nd argument (void *ptr) of the user callback function cb(). Set this to NULL if unused.
- 5th argument flags is unused and undefined. Set this to zero.

Himax Confidential
Do Not Copy

M. uvc_start_streaming_multi_cb(devh, &ctrl, nir_cb, NULL, depth_cb, NULL, dot_cb, NULL, 0);

Start video streaming with setting up three callback functions nir_cb/depth_cb/dot_cb for NIR/Depth/Dot frames, respectively. This is a Himax-customized variant API of the vanilla uvc_start_streaming() API with multiple user frame callback functions registration. User does not need to figure out which frame type is at this frame, it has already been parsed.

- This function need an available UVC device handle and UVC stream control block, so it must be called after uvc_get_stream_ctrl_format_size() or uvc_get_stream_ctrl_format_size_ifno() function called successfully.
- If it started successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to start.
- 1st argument devh is the UVC device handle to be read.
- 2nd argument ctrl is a pointer pointed to the UVC stream control block.
- 3rd/5th/7th arguments nir_cb/depth_cb/dot_cb are the user frame callback functions to be registered, and the callback frame will be placed in the 1st argument (uvc_frame_t *frame).
- 4th/6th/8th arguments nir_user_ptr/depth_user_ptr/dot_user_ptr are a pointer that will be placed in the 2nd argument (void *ptr) of the corresponding user callback function xxx_cb(). Set this to NULL if unused.
- 5th argument flags is unused and undefined. Set this to zero.

N. xxx_cb(uvc_frame_t *frame, void *ptr);

User frame callback function of nir_cb/depth_cb/dot_cb/rgb_cb.

- 1st argument frame is the UVC frame data (uvc_frame_t *) received.
- 2nd argument ptr is a user pointer registered at uvc_start_streaming() or uvc_start_streaming_multi_cb() function called.
  - frame → data is the pointer pointed to the video frame data buffer.
  - frame → data_bytes is the frame size in byte.
  - frame → width and frame → height are the frame resolution.
  - frame → step is the length of one line in bytes, it's related to width and format.
- Please check the example code "test.c" source file for more details.

O. uvc_y16_shr(in, out, shr);

Right-shifting N-bits on every single pixel of the input Y16 frame and output the result in output Y16 frame.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in/out are pointers pointed to input/output Y16 frames.
- 3rd argument shr specify the number of bits to be shifted.

Himax Confidential Do Not Copy

P. uvc_y16_shl(in, out, shl);

Left-shifting N-bits on every single pixel of the input Y16 frame and output the result in output Y16 frame.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in/out are pointers pointed to input/output Y16 frames.
- 3rd argument shl specify the number of bits to be shifted.

Q. uvc_y16_to_y8(in, out, effective_bit_alignment);

Convert from Y16 frame to Y8 frame (dropped the least-significant bits) with specified zero-padding alignment for NIR frame.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in and out are the pointers pointed to the input Y16 frame and output Y8 frame, respectively.
- 3rd argument effective_bit_alignment indicates NIR effective 10-bit value is left- or right-aligned, the available value is defined as below.
  - HX_EFFECTIVE_BIT_ALIGNMENT_LEFT (0): NIR effective 10-bit value is left-aligned (zero-padding on least-significant 6-bit).
  - HX_EFFECTIVE_BIT_ALIGNMENT_RIGHT (1): NIR effective 10-bit value is right-aligned (zero-padding on most-significant 6-bit).

R. uvc_y8_to_bgr(in, out);
uvc_y8_to_rgb(in, out);

Convert from Y8 frame to gray BGR/RGB frame for NIR frame.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in and out are the pointers pointed to the input Y8 frame and output BGR/RGB frame, respectively.

Himax Confidential
Do Not Copy

S. uvc_any2bgr(in, out);
uvc_any2rgb(in, out);

Convert from any type frame to BGR/RGB frame for NIR and RGB (YUV) frame.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in and out are the pointers pointed to the input frame and output BGR/RGB frame, respectively.

Note: (1) This function implied uvc_y8_to_bgr() or uvc_y8_to_rgb() function if input is Y8 frame.

T. uvc_y16_to_depth_colormap_bgr(in, out, shr);
uvc_y16_to_depth_colormap_rgb(in, out, shr);

Convert from RAW16 depth frame to BGR/RGB color map frame for depth frame used only.

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in and out are the pointers pointed to the input Y16 frame and output BGR/RGB frame, respectively.
- 3rd argument shr specify the number of bits to be right-shifted on every depth pixel.

Note: (1) Since the internal color map conversion only accepted input depth range from 0 ~ 1023, so user must tune this value and specify it to get correct depth color map.

U. uvc_hx_y16_rotate(in, out, mode);
uvc_hx_bgr_rotate(in, out, mode);
uvc_hx_rgb_rotate(in, out, mode);

Rotate Y16/BGR/RGB frame by 0/90/180/270 degrees

- This function is typically been called in user frame callback function cb(), and used to process the pixel data.
- If it run successfully, then it'll return UVC_SUCCESS (0), otherwise it's failed to run.
- 1st and 2nd arguments in and out are the pointers pointed to the input Y16 frame and output BGR/RGB frame, respectively.
- 3rd argument mode value is defined in the enum uvc_hx_rotation_mode in the libuvc_hx.h header file, currently only support following modes.

Himax Confidential
Do Not Copy

V. uvc_stop_streaming(devh);

Once user done the video streaming, use this function to stop the UVC video streaming.

- This function need an available UVC device handle and its video streaming had started, so it must be called after uvc_start_streaming() or uvc_start_streaming_multi_cb() function called successfully.
- It has no return value.
- 1st argument devh is the UVC device handle to be stopped.

W. uvc_close(devh);

Close the UVC device.

- This function need an available UVC device handle, so it must be called after uvc_open() function called successfully.
- It has no return value.
- 1st argument devh is the UVC device handle to be closed.

X. uvc_unref_device(dev);

Decrease the reference count for a UVC device which was increased after found or opened device. This should be called after close UVC device.

- This function need an available UVC device, so it must be called after uvc_find_device() function called successfully.
- It has no return value.
- 1st argument dev is the UVC device to be unreferred.

Y. uvc_exit(ctx);

Close the initialized UVC context and shutting down any active UVC device.

- This function need an available UVC context, so it must be called after uvc_init() function called successfully.
  - It has no return value.
  - 1st argument ctx is the UVC context to be uninitialized.

Himax Confidential Do Not Copy

This information contained herein is the exclusive property of Himax and shall not be distributed, reproduced, or disclosed in whole or in part without prior written permission of Himax.

4. Himax Camera Test App Example Code

4.1. Himax camera test app example code

The example code test.c app for the libuvc used OpenCV to draw the window for video frame.

About the more details of function call usage, please refer to the example code test.c source file.

4.2. Screenshot of example code

Running on the x86_64 PC with Ubuntu Linux 18.04 operating system.

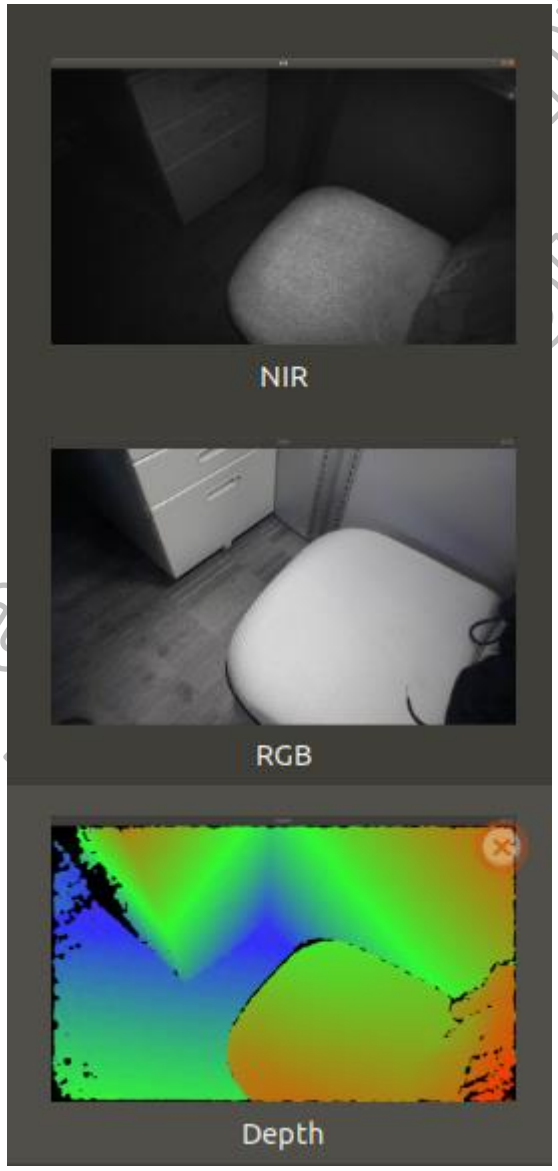

Figure 4.1: Screenshot of example code

Himax Confidential