Himax Technologies, Inc.

Software Package (SDK)

Himax SH430UH

Linux UVC Extension Controls
User Manual

Preliminary version 1.1. Jun 7, 2020

Himax Technologies, Inc.

Revision History

Jun 7, 2020

Version	Date	Description of changes
1.0	2020/12/15	First release.
1.1	2021/06/07	Add Liveness check command.

List of Contents

1. Linux UVC Extension Controls API usage

Himax added custom Linux UVC Extension Controls APIs as following sections.

1.1 uvc_error_t uvc_hx_get_ci1_fw_version(uvc_device_handle_t *devh, char *ci1_fw_version, uint16_t ci1_fw_version_len)

Read the Ci1 FW version from Himax UVC camera device.

※ This function needs 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 `ci1_fw_version` is a pointer pointed to a `char` array with length of `32`.

※ 3rd argument `ci1_fw_version_len` should be fixed `32`.

1.2 uvc_error_t uvc_hx_set_frame_mode(uvc_device_handle_t *devh, enum hx_frame_mode_select 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 needs 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.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.)*

1.3 uvc_error_t uvc_hx_roi_set_feature(uvc_device_handle_t *devh, struct hx_roi_table_feature *feature)

Sent the **ROI set feature** command to Himax UVC camera device.

※ This function needs 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 `feature` is a pointer pointed to a `hx_roi_table_feature` structure. This `feature` contains six bit-fields, total 8-bits:

bit[7]	bit[6]	bit[5]	bit[4]	bit[3:2]	bit[1:0]
sync	update	clear	reset	type	mode

※ There are 3 cases to use this API, please clear all fields first before set any field bit:

- Before calling `uvc_hx_roi_set_coordinate()`, set `mode` = `1` and `clear` = `1`.
- After calling `uvc_hx_roi_set_coordinate()`, set `update` = `1` and `sync` = `1`.
- Reset ROI to default state once you want to stop ROI: set `reset` = `1`.

1.4 uvc_error_t uvc_hx_roi_set_coordinate(uvc_device_handle_t *devh, uint8_t weighting, int *box, int size)

Sent the **ROI set coordinate** command to Himax UVC camera device.

※ This function needs 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 `weighting` is currently set to `0x05`.

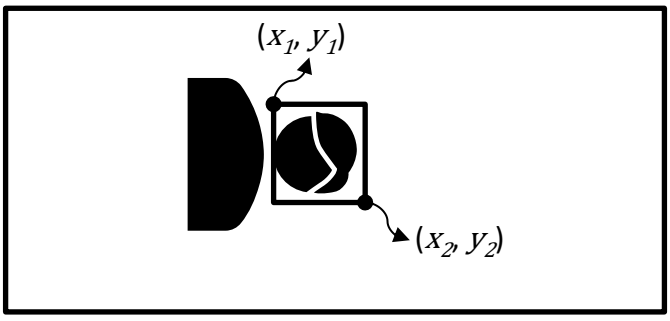
※ 3rd argument `box` is a pointer pointed to an `int` array with length of `4`.

- This `box` contains two landscape coordinates (x_1, y_1) and (x_2, y_2), and store as following integer ordering:

[0]	[1]	[2]	[3]
x_1	y_1	x_2	y_2

- And, these two landscape coordinates are presented as below diagram:

(0, 0)



(1279, 799)

※ 4th argument `size` should be fixed `4`.

1.5 uvc_error_t uvc_hx_set_2d_target_fmean(uvc_device_handle_t *devh, uint16_t fmean, uint16_t bound)

Set the target fmean and bound values of the 2D frame of Himax UVC camera device.

※ This function needs 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 `fmean` is a 16-bit integer, `fmean` + `bound` should be in `0`...`1023`.

※ 3rd argument `bound` is a 16-bit integer, `fmean` + `bound` should be in `0`...`1023`.

1.6 uvc_error_t uvc_hx_get_2d_target_fmean(uvc_device_handle_t *devh, uint16_t *fmean, uint16_t *bound)

Get the target fmean and bound values of the 2D frame of Himax UVC camera device.

※ This function needs 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 `fmean` is a pointer pointed to a 16-bit integer to be read.

※ 3rd argument `bound` is a pointer pointed to a 16-bit integer to be read.

1.7 uvc_error_t uvc_hx_set_again(uvc_device_handle_t *devh, uint8_t again)

Set the A-gain value of Himax UVC camera device.

※ This function needs 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 `again` is a 8-bit integer.

1.8 uvc_error_t uvc_hx_get_hv2_fw_version(uvc_device_handle_t *devh, hx_hv2_fw_version_t *hv2_fw_version)

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

※ This function needs 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.h` header file for more details.

1.9 uvc_error_t uvc_hx_get_intrinsic_extrinsic_matrices(uvc_device_handle_t *devh, hx_cam_intrinsic_extrinsic_matrices_t *cam_matrices, int require_scaling)

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

※ This function needs 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.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.

1.10 uvc_error_t uvc_hx_get_build_type(uvc_device_handle_t *devh, uint16_t *build_type)

Read the build type from Himax UVC camera device.

※ This function needs 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.

1.11 uvc_error_t uvc_hx_get_serial_number(uvc_device_handle_t *devh, uint32_t *serial_number)

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

※ This function needs 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.

1.12 uvc_error_t uvc_hx_set_face_recognize(uvc_device_handle_t *devh, uint8_t mode, uint16_t face_id, uint16_t *result)

Trigger one action of face recognition/registration/deletion on Himax UVC camera device.

※ This function needs 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` is an 8-bit integer, set "`0`" to do face recognition, set "`1`" to do face registration, set "`'d'`" to delete registered faces in face database with specified face id.

※ 3rd argument `face_id` is a 16-bit integer, only used in face deletion action, presents as the face id to be deleted; if set value as "`0xFFFF`", then it will delete all registered faces in face database (clear all faces).

※ 4th argument `result` is a pointer pointed to a 16-bit integer, presents as the result returned from face SDK, if it returned `result` has "`0xE0XX`" prefix value means there is error occurred, otherwise the required action is finished successfully with returned face id in this `result` value.

1.13 uvc_error_t uvc_hx_set_liveness_check(uvc_device_handle_t *devh, uint8_t mode, uint16_t *result)

Trigger one action of face liveness check on Himax UVC camera device.

※ This function needs 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` is an 8-bit integer, currently only support mode "`0`".

※ 3rd argument `result` is a pointer pointed to a 16-bit integer, presents as the result returned from face SDK, if it returned `result` has "`0xE0XX`" prefix value means there is error occurred, otherwise the required action is finished successfully.