

# Sentech USB Camera Driver for Linux

Version 4.0.3

2018/10/19

## Index

1. At first .....	3
2. Driver Configuration .....	4
3. Functions .....	5
3.1 Deveice Driver Functions .....	5
3.1.1 Device Driver Functions .....	6
3.1.2 IO control request .....	13
3.1.3 Camera settings .....	60
3.2 Library functions .....	118

## 1. At first

### **Disclaimer**

The sample programs are for SDK functions/specification confirmation and not guaranteed for working in all environments.

We are not responsible for any possible damage caused by using the sample programs.

### **Notes**

It is necessary to make the error process because the frame drop may occur when the USB bus transferring speed reduces based of the PC (including the CPU) usage.

The following operating conditions may occurred errors during the use of this product:

The USB camera is operating with

- a low performance PC
- a low quality USB cable
- a long USB cable
- an extend USB cable (not recommend)
- a USB HUB
- an early version of USB2.0 interface board
- an USB interface board (recommend chipset is newer than ICH4)
- the multiple USB cameras
- a PC that other USB devices are connected
- the high speed operating clock speed
- a USB host controller other than made by Intel

### **Operation Environment**

Sentech has been tested this driver in following Linux distribution environments. It may not work correctly in other Linux distribution environments.

Ubuntu 16.04(LTS)

Ubuntu 14.04(LTS)

CentOS 7

Sentech has been tested this driver in following environments:

Raspberry Pi 3

JETSON TX1

Intel Edison

## 2. Driver Configuration

This driver is configured with “Video for Linux2 (V4L2) standard compliant device driver (stcam\_dd) and the image processing library (stcam\_lib) for the image acquisition from the camera and change the camera settings.

The image processing by the image processing library is on the host side (computer and/or embedded system). The frame drop may occur if the hardware performance is not good enough for these image processing.

If acquire the RAW image is enough (the image processing is not need it), the RAW image can acquire with the device driver.

### 3. Functions

#### 3.1 Deveice Driver Functions

The API format is the Video for Linux2 (V4L2) standard compliant format. The API may change depending on the version of Linux kernel.

Please check the detail of API of V4L2 standard by following link:

<https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/user-func.html>

### 3.1.1 Device Driver Functions

It makes requesting to the device driver by system call of Linux kernel.  
The function and the way to calling API are following the V4L2 standard.

#### **open**

This function opens the device.

```
int open(  
    const char *device_name,    // Device file  
    int flags                    // Flag  
);
```

#### [Parameters]

device\_name

Set the device file path.

flags

Set [O\_RDWR | O\_NONBLOCK] to open the device.

#### [Return]

If the function was successful, return a file descriptor for using the device.

If the function was failed, return -1.

Please check the error code in errno of Linux for the failure details.

#### [Descriptions]

This function opens the device.

It is necessary to close the opened camera before finish to use the camera.

## **close**

This function closes the device.

```
int close(  
    int fd        // File descriptor  
);
```

### [Parameters]

fd

Set the file descriptor for using device that is obtained by open function.

### [Return]

If the function was successful, return 0.

If the function was failed, return -1.

Please check the error code in errno of Linux for the failure details.

### [Descriptions]

This function closes the device.

It is necessary to close the opened camera before finish to use the camera.

## **mmap**

This function maps the buffer for the image to the user space.

```
void *mmap(  
    void *start,          // Start location  
    size_t length,        // Buffer size  
    int prot,             // Memory protection  
    int flags,            // Flag  
    int fd,               // File descriptor  
    off_t offset           // Offset  
);
```

### [Parameters]

**start**

Set the address space of the application that is NULL pointer.

**length**

Set the size of the buffer for mapping.

**prot**

Set the memory protection that is [PROT\_READ | PROT\_WRITE].

**flags**

Set the mapping flag that is [MAP\_SHARED].

**fd**

Set the file descriptor for using device that is obtained by open function.

**offset**

This function maps the buffer for the image to the user space.

### [Return]

If the function was successful, return the address of the mapped buffer.

If the function was failed, return MAP\_FAILED (-1).

Please check the error code in errno of Linux for the failure details.



[Descriptions]

This function maps the buffer for the image to the user space.

## **munmap**

This function un-mappes the mapped the buffer for the image from the user space.

```
int munmap(  
    void *start,          // Buffer address  
    size_t length,        // Buffer size  
);
```

### [Parameters]

start

Set the address for mapped buffer.

length

Set the size for mapped buffer.

### [Return]

If the function was successful, return 0.

If the function was failed, return -1.

Please check the error code in errno of Linux for the failure details.

### [Descriptions]

This function un-mappes the mapped the buffer for the image from the user space.

## **select**

This function waits until the image becomes readable.

```
int select(  
    int nfd,           // File descriptor  
    fd_set *readfds,   // Read descriptor  
    fd_set *writefds,  // Write descriptor  
    fd_set *exceptfds, // Exception monitoring descriptor  
    struct timeval *timeout // Timeout time  
);
```

### [Parameters]

**nfd**

Set “open function obtained value + 1”.

**readfds**

Set the file descriptor for using device for reading that is obtained by open function.

**writefds**

Set the file descriptor for the target device for writing that is NULL pointer.

**exceptfds**

Set the file descriptor for the exception monitoring device that is NULL pointer..

**timeout**

Set the timeout time.

### [Return]

If the function was successful, return more than 1.

If the time out is occurred, return 0.

If the function was failed, return -1.

Please check the error code in errno of Linux for the failure details.

### [Descriptions]

This function waits until the image becomes readable.

## **ioctl**

This function requests the IO control.

```
int ioctl(  
    int      fd,      // File descriptor  
    int      request, // Request code  
    void     *argp    // Request parameter  
);
```

### [Parameters]

**fd**

Set the file descriptor for using device that is obtained by open function.

**request**

Set the request code.

**argp**

Set the additional information for the request code.

### [Return]

If the function was successful, return 0.

If the function was failed, return -1.

Please check the error code in `errno` of Linux for the failure details.

### [Descriptions]

This function requests the IO control.

Please check “3.1.2 IO control request” for IO control request details.

### 3.1.2 IO control request

The system call (ioctl) of Linux kernel uses for detailed requests for the device drive.

The functions are defined by the V4L2 standard (VIDIOC\_) and additional functions for Sentech cameras (STCIOC\_) are available.

The way to calling API is following the V4L2 standard.

#### **VIDIOC\_QUERYCAP**

This function obtains the device information and functionality of the device.

[Request Parameters]

```
struct v4l2_capability {  
    __u8    driver[16];  
    __u8    card[32];  
    __u8    bus_info[32];  
    __u32    version;  
    __u32    capabilities;  
    __u32    device_caps;  
    __u32    reserved[3];  
};
```

[Parameters]

driver

Obtain the driver name [stcam\_dd].

card

Obtain the camera model name.

bus\_info

Obtain the path information.

version

Obtain the version of the driver.

capabilities

Obtain the available functionality of the driver that is [V4L2\_CAP\_VIDEO\_CAPTURE | V4L2\_CAP\_STREAMING | V4L2\_CAP\_DEVICE\_CAPS].

device\_caps

Obtain the available functionality of the driver that is [V4L2\_CAP\_VIDEO\_CAPTURE | V4L2\_CAP\_STREAMING].

[Descriptions]

This function obtains the device information and functionality of the device.

## VIDIOC\_ENUM\_INPUT

This function obtains the video input information for the specified index.

### [Request Parameters]

```
struct v4l2_input {
    __u32      index;
    __u8       name[32];
    __u32      type;
    __u32      audioset;
    __u32      tuner;
    v4l2_std_id std;
    __u32      status;
    __u32      capabilities;
    __u32      reserved[3];
};
```

### [Parameters]

index

Set the index of the video input.

name

Obtain the name of the video input.

type

Obtain the type of the video input.

status

Obtain the status of the video input.

capabilities

Obtain the functionality of the video input.

[Descriptions]

This function obtains the video input information for the specified index.

If the corresponding video input for the specified index is not available, return EINVAL for errno.

Please check the error code in errno of Linux for the failure details.

## **VIDIOC\_G\_INPUT**

This function obtains the video input information.

[Request Parameters]

int \*argp

Obtain the index of the video input.

[Descriptions]

This function obtains the video input information.

Please check VIDIOC\_ENUM\_INPUT for the corresponding video input information for the specified index.

## **VIDIOC\_S\_INPUT**

This function sets the video input information.

[Request Parameters]

int \*argp

Set the index of the video input.

[Descriptions]

This function sets the video input information.

Please check VIDIOC\_ENUM\_INPUT for the corresponding video input information for specified index.



## VIDIOC\_ENUM\_FMT

This function obtains the color format of the specified index.

[Request parameters]

```
struct v4l2_fmtdesc {
    __u32      index;
    __u32      type;
    __u32      flags;
    __u8       description[32];
    __u32      pixelformat;
    __u32      reserved[4];
};
```

[Parameters]

index

Set the index of the color format.

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

flags

Obtain the flag of the special format.

description

Obtain the descriptions of the color format.

pixelformat

Obtain the image format identifier that is four characters.

[Descriptions]

This function obtains the color format of the specified index.

If the corresponding color format for the specified index is not available, return `EINVAL` for `errno`.

Please check the error code in `errno` of Linux for the failure details.

## VIDIOC\_G\_FMT

This function obtains the format of the image.

[Request parameters]

```
struct v4l2_format {
    __u32    type;
    union {
        struct v4l2_pix_format    pix;
    } fmt;
};
```

```
struct v4l2_pix_format {
    __u32    width;
    __u32    height;
    __u32    pixelformat;
    __u32    field;
    __u32    bytesperline;
    __u32    sizeimage;
    __u32    colorspace;
    __u32    priv;
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

width

Obtain the width of the output image.

height

Obtain the height of the output image.

pixelformat

Obtain the identifier of the image format that is four characters.

`bytesperline`

Obtain the number of the byte for one horizontal line.

`sizeimage`

Obtain the number of the byte for the output image.

`colospace`

Obtain the color space information.

[Descriptions]

This function obtains the format of the image.

## VIDIOC\_S\_FMT

This function sets the format of the image.

[Request parameters]

```
struct v4l2_format {
    __u32    type;
    union {
        struct v4l2_pix_format    pix;
    } fmt;
};
```

```
struct v4l2_pix_format {
    __u32    width;
    __u32    height;
    __u32    pixelformat;
    __u32    field;
    __u32    bytesperline;
    __u32    sizeimage;
    __u32    colorspace;
    __u32    priv;
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

width

Set the width of the output image.

height

Set the height of the output image.

pixelformat

Set the identifier of the image format that is four characters.

`bytesperline`

Set the number of the byte for one horizontal line.

`sizeimage`

Set the number of the byte for the output image.

`colospace`

Set the color space information.

[Descriptions]

This function sets the format of the image.

## VIDIOC\_TRY\_FMT

This function checks the availability of the specified image format.

[Request parameters]

```
struct v4l2_format {
    __u32    type;
    union {
        struct v4l2_pix_format    pix;
    } fmt;
};
```

```
struct v4l2_pix_format {
    __u32    width;
    __u32    height;
    __u32    pixelformat;
    __u32    field;
    __u32    bytesperline;
    __u32    sizeimage;
    __u32    colorspace;
    __u32    priv;
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

width

Set the width of the output image.

height

Set the height of the output image.

pixelformat

Set the identifier of the image format that is four characters.

bytesperline

Set the number of the byte for one horizontal line.

sizeimage

Set the number of the byte for the output image.

colospace

Set the color space information.

[Descriptions]

This function checks the availability of the specified image format.

These settings do not apply for the image data from the camera.

## VIDIOC\_G\_PARM

This function obtains the parameters of the streaming.

[Request parameters]

```
struct v4l2_streamparm {
    __u32    type;
    union {
        struct v4l2_captureparm capture;
    } parm;
};
```

```
struct v4l2_captureparm {
    __u32          capability;
    __u32          capturemode;
    struct v4l2_fract timeperframe;
    __u32          extendedmode;
    __u32          readbuffers;
    __u32          reserved[4];
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

capability

Obtain the functionality of the streaming.

capturemode

Obtain the capture mode.

timeperframe

Obtain the output frame rate.

extendedmode

Obtain the extended parameters.



readbuffers

Obtain the number of the secured output buffer.

[Descriptions]

This function obtains the parameters of the streaming.

## VIDIOC\_S\_PARM

This function sets the parameter of the streaming.

[Request parameters]

```
struct v4l2_streamparm {
    __u32    type;
    union {
        struct v4l2_captureparm capture;
    } parm;
};
```

```
struct v4l2_captureparm {
    __u32          capability;
    __u32          capturemode;
    struct v4l2_fract timeperframe;
    __u32          extendedmode;
    __u32          readbuffers;
    __u32          reserved[4];
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

capability

Set the functionality of the streaming.

capturemode

Set the capture mode.

timeperframe

Set the output frame rate.

extendedmode

Set the extended parameters.

readbuffers

Set the number of the secured output buffer.

[Descriptions]

This function sets the parameter of the streaming.

## VIDIOC\_REQBUFS

This function secures the buffer for the image.

[Request parameters]

```
struct v4l2_requestbuffers {  
    __u32      count;  
    __u32      type;  
    __u32      memory;  
    __u32      reserved[2];  
};
```

[Parameters]

count

Set the number of buffer to secure it.

This will be overwritten by the number of the secured buffer after process it.

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

memory

Set the type of secure buffer that is V4L2\_MEMORY\_MMAP

[Descriptions]

This function secures the buffer for the image.

## VIDIOC\_QUERYBUF

This function obtains the information of the secured buffer for the image.

[Request parameters]

```
struct v4l2_buffer {
    __u32                index;
    __u32                type;
    __u32                bytesused;
    __u32                flags;
    __u32                field;
    struct timeval        timestamp;
    struct v4l2_timecode  timecode;
    __u32                sequence;
    __u32                memory;
    union {
        __u32            offset;
    } m;
    __u32                length;
    __u32                reserved2;
    __u32                reserved;
};
```

[Parameters]

index

Set the index of the secured buffer.

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

flags

Obtain the flag of the buffer status.

field

Obtain the image field.

timestamp

Obtain the time stamp of the image.

memory

Obtain the type of the secured buffer.

offset

Obtain the start location of the secured buffer.

length

Obtain the size of the secured buffer.

[Descriptions]

This function obtains the information of the secured buffer for the image.

## VIDIOC\_QBUF

This function connects the buffer to the streaming queue.

[Request parameters]

```
struct v4l2_buffer {
    __u32                index;
    __u32                type;
    __u32                bytesused;
    __u32                flags;
    __u32                field;
    struct timeval        timestamp;
    struct v4l2_timecode  timecode;
    __u32                sequence;
    __u32                memory;
    union {
        __u32            offset;
    } m;
    __u32                length;
    __u32                reserved2;
    __u32                reserved;
};
```

[Parameters]

index

Set the index of the buffer.

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

flags

Set the flag of the buffer status.

field

Set the image field.

timestamp

Set the time stamp of the image.

memory

Set the type of the secured buffer.

offset

Set the start location of the secured buffer.

length

Set the size of the secured buffer.

[Descriptions]

This function connects the buffer to the streaming queue.

The buffer will be able to the target for the image writing.



## VIDIOC\_DQBUF

The function disconnects the buffer from the streaming queue.

[Request parameters]

```
struct v4l2_buffer {
    __u32          index;
    __u32          type;
    __u32          bytesused;
    __u32          flags;
    __u32          field;
    struct timeval  timestamp;
    struct v4l2_timecode timecode;
    __u32          sequence;
    __u32          memory;
    union {
        __u32      offset;
    } m;
    __u32          length;
    __u32          reserved2;
    __u32          reserved;
};
```

[Parameters]

index

Set the index of the buffer.

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

flags

Set the flag of the buffer status.

field

Set the image field.

timestamp

Set the time stamp of the image.

memory

Set the type of the secured buffer.

offset

Set the start location of the secured buffer.

length

Set the size of the secured buffer.

[Descriptions]

The function disconnects the buffer from the streaming queue.

The buffer will be not able to the target for the image writing.

## **VIDIOC\_STREAMON**

This function starts the streaming.

[Request parameters]

const int \*argp

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

[Descriptions]

This function starts the streaming.

The image is obtained and copy to the buffer.

## **VIDIOC\_STREAMOFF**

This function stops the streaming.

[Request parameters]

const int \*argp

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

[Descriptions]

This function stops the streaming.

The image obtaining is stopped.

## VIDIOC\_CROPCAP

This function obtains the crop image information.

[Request parameters]

```
struct v4l2_cropcap {  
    __u32                type;  
    struct v4l2_rect      bounds;  
    struct v4l2_rect      defrect;  
    struct v4l2_fract     pixelaspect;  
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

bounds

Obtain the maximum size of cropping.

defrect

Obtain the default size of cropping.

pixelaspect

Obtain the aspect of cropping.

[Descriptions]

This function obtains the crop image information.

This is changed when changing the size of the output image from the camera.

## **VIDIOC\_G\_CROP**

This function obtains the cropping area of the image.

[Request parameters]

```
struct v4l2_crop {  
    __u32                type;  
    struct v4l2_rect     c;  
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

c

Obtain the cropping area of the image.

[Descriptions]

This function obtains the cropping area of the image.

## VIDIOC\_S\_CROP

This function sets the cropping area of the image.

[Request parameters]

```
struct v4l2_crop {  
    __u32                type;  
    struct v4l2_rect     c;  
};
```

[Parameters]

type

Set the buffer type that is [V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE].

c

Set the cropping area of the image.

[Descriptions]

This function sets the cropping area of the image.

This is changed to the maximize image size when changing the output image size from the camera.

This function will cropping image from the original image from the camera. Do not change the image size from the camera by this function. Thus, decreasing image size by this function will not increase the frame rate.

Please use STCIIOC\_S\_SCAN\_SIZE or STC\_CID\_SCAN\_MODE functions to change the size of the output from the camera.

## VIDIOC\_QUERYCTRL

This function obtains the type, setting range and other of the camera settings.

[Request parameters]

```
struct v4l2_queryctrl {
    __u32      id;
    __u32      type;
    __u8       name[32];
    __s32      minimum;
    __s32      maximum;
    __s32      step;
    __s32      default_value;
    __u32      flags;
    __u32      reserved[2];
};
```

[Parameters]

id

Set the ID of the camera setting.

type

Obtain the type of the camera setting.

name

Obtain the name of the camera setting.

minimum

Obtain the minimum value of the camera setting.

maximum

Obtain the maximum value of the camera setting.

step

Obtain the adjustable unit of the camera setting.

default\_value

Obtain the default value of the camera setting.

flags

Obtain the status flags (enabled/disabled/read-only) of camera setting.

[Descriptions]

This function obtains the type, setting rang and other of the camera settings (check 3.1.3).

The available setting and function are different for each camera model. Please check availability with this function.



## VIDIOC\_QUERYMENU

This function obtains the menu type of the camera settings.

[Request parameters]

```
struct v4l2_querymenu {  
    __u32      id;  
    __u32      index;  
    union {  
        __u8    name[32];  
    };  
    __u32      reserved;  
};
```

[Parameters]

id

Set the ID of the camera setting.

index

Set the menu index of the camera setting.

name

Obtain the menu name of the camera setting.

[Descriptions]

This function obtains the menu type of the camera settings (check 3.1.3).

The available index is different for each camera model. Please check availability with this function.

## **VIDIOC\_G\_CTRL**

This function obtains the setting of the camera.

[Request parameters]

```
struct v4l2_control {  
    __u32      id;  
    __s32      value;  
};
```

[parameters]

id

Set the ID of the camera setting.

value

Obtain the value of the camera setting.

[Descriptions]

This function obtains the setting of the camera (check 3.1.3).

The available settings are different for each camera model. Please check availability with VIDIOC\_QUERYCTRL.

## **VIDIOC\_S\_CTRL**

This function sets the camera setting.

[Request parameters]

```
struct v4l2_control {  
    __u32      id;  
    __s32      value;  
};
```

[Parameters]

id

Set the ID of the camera setting.

value

Set the value of the camera setting.

[Descriptions]

This function sets the camera setting (check 3.1.3).

The available settings are different for each camera model. Please check availability with VIDIOC\_QUERYCTRL.

## **VIDIOC\_LOG\_STATUS**

This function records the kernel log when the request is received.

[Request parameters]

None.

[Descriptions]

This function records the kernel log when the request is received.

## STCIOC\_G\_CAM\_VERSION

This function obtains the version information of the camera.

[Request parameters]

```
struct stcam_st_cam_version {  
    __u16    vendor_id;  
    __u16    product_id;  
    __u16    camera_type;  
    __u8     product_name[STC_IOC_PARAM_LEN_PRODUCT_NAME];  
    __u16    fpga_version;  
    __u16    firm_version;  
};
```

[Parameters]

vendor\_id

Obtain the USB vender ID.

product\_id

Obtain the USB product ID.

camera\_type

Obtain the camera model code.

product\_name

Obtain the name of the camera.

fpga\_version

Obtain the FPGA version of the camera.

firm\_version

Obtain the firmware version of the camera.

[Descriptions]

This function obtains the version information of the camera.

## **STCIOC\_G\_USER\_DATA**

This function loads the user data from the EEPROM on the camera.

[Request parameters]

```
struct stcam_st_user_data {  
    __u8    data[STC_IOC_PARAM_LEN_USER_DATA];  
    __u16   offset;  
    __u16   length;  
};
```

[Parameters]

data

Set the location to save the user data.

offset

Obtain the location of EEPROM on the camera, to read the user data.

length

Obtain the user data size to read from EEPROM on the camera.

[Descriptions]

This function loads the user data from the EEPROM on the camera.

## **STCIOC\_S\_USER\_DATA**

This function saves the user data to the EEPROM on the camera.

[Request parameters]

```
struct stcam_st_user_data {  
    __u8    data[STC_IOC_PARAM_LEN_USER_DATA];  
    __u16   offset;  
    __u16   length;  
};
```

[Parameters]

data

Set the location of the data to save.

offset

Set the location of the EEPROM on the camera, to save the user data.

length

Set the user data size for save to the EEPROM.

[Descriptions]

This function saves the user data to the EEPROM on the camera.

## **STCIOC\_G\_CAM\_USER\_ID**

This function obtains the camera identify information (camera ID).

[Request parameters]

```
struct stcam_st_cam_user_id {  
    __u32    camera_id;  
    __u8     camera_name[STC_IOC_PARAM_LEN_CAMERA_NAME];  
    __u8     length;  
};
```

[Parameters]

camera\_id

Obtain the camera ID.

camera\_name

Set the buffer for the obtained camera ID.

length

Obtain the size of the the camera ID.

[Descriptions]

This function obtains the camera identify information (camera ID).

## **STCIOC\_S\_CAM\_USER\_ID**

This function sets the camera identify information (camera ID).

[Request parameters]

```
struct stcam_st_cam_user_id {  
    __u32    camera_id;  
    __u8     camera_name[STC_IOC_PARAM_LEN_CAMERA_NAME];  
    __u8     length;  
};
```

[Parameters]

camera\_id

Set the camera ID.

camera\_name

Set the buffer for set the camera ID.

length

Set the size of the camera ID.

[Descriptions]

This function sets the camera identify information (camera ID).



## STCIOC\_G\_MAX\_SCAN\_SIZE

This function obtains the maximum image size that can be obtained from the camera.

[Request parameters]

```
struct stcam_st_max_scan_size {  
    __u16    width;  
    __u16    height;  
};
```

[Parameter]

width

Obtain the maximum width of the image.

height

Obtain the maximum height of the image.

[Descriptions]

This function obtains the maximum image size that can be obtained from the camera.

## STCIOC\_G\_SCAN\_SIZE

This function obtains the size and offse of the output image from the camera.

[Request parameters]

```
struct stcam_st_scan_size {
    union {
        struct stcam_st_scan_normal    normal;
        struct stcam_st_scan_roi       roi;
    };
};
```

```
struct stcam_st_scan_normal {
    __u16  offset_x;
    __u16  offset_y;
    __u16  width;
    __u16  height;
};
```

```
struct stcam_st_scan_roi {
    __u16  offset_x;
    __u16  offset_y;
    __u16  width;
    __u16  height;
    __u8   index;
    __u8   region_mode;
};
```

[parameters]

offset\_x

Obtain the horizontal offset of the image.

offset\_y

Obtain the vertical offset of the image.

width

Obtain the width of the image.

height

Obtain the height of the image.

index

Obtain the index of ROI.

region\_mode

Obtain the enable status of ROI.

[Descriptions]

This function obtains the size and offse of the output image from the camera.

The index and enable status of ROI are obtained if the camera is supported the multi ROI function.

## STCIOC\_S\_SCAN\_SIZE

This function sets the size and offset of the output image from the camera.

[Request parameters]

```
struct stcam_st_scan_size {  
    union {  
        struct stcam_st_scan_normal    normal;  
        struct stcam_st_scan_roi       roi;  
    };  
};
```

```
struct stcam_st_scan_normal {  
    __u16  offset_x;  
    __u16  offset_y;  
    __u16  width;  
    __u16  height;  
};
```

```
struct stcam_st_scan_roi {  
    __u16  offset_x;  
    __u16  offset_y;  
    __u16  width;  
    __u16  height;  
    __u8   index;  
    __u8   region_mode;  
};
```

[Parameters]

offset\_x

Set the horizontal offset of the image.

offset\_y

Set the vertical offset of the image.

width

Set the width of the image.

height

Set the height of the image.

index

Set the index of ROI.

region\_mode

Set enable / disable for ROI.

#### [Descriptions]

This function sets the size and offset of the output image from the camera.

It is necessary to set `STC_MENU_SCAN_MODE_VARIABLE_PARTIAL` at `STC_CID_SCAN_MODE` for USB2.0 cameras before use this function.

It is necessary to set `STC_MENU_SCAN_MODE_ROI` at set `STC_CID_SCAN_MODE` for USB3.0 cameras before use this function.

The vertical setting is only changeable for USB 2.0 cameras.

The index and enabled status of ROI are changeable if the camera is supported the multi ROI function. Index[0] cannot disabled.

## STCIOC\_G\_HDR

This function obtains the HDR settings.

[Request parameters]

```
struct stcam_st_hdr {  
    union {  
        struct stcam_st_hdr_cmos_4m    cmosis_4m;  
    };  
};
```

```
struct stcam_st_hdr_cmos_4m {  
    __u8    mode;  
    __u8    slope;  
    __u8    vlow3;  
    __u8    vlow2;  
    __u8    knee2;  
    __u8    knee1;  
};
```

[Parameters]

mode

Obtain the HDR mode.

slope

Obtain the number of the slope.

vlow3

Obtain the Vlow3 voltage. This is only valid when the number of the slope is 3.

vlow2

Obtain the Vlow2 voltage. This is only valid when the number of the slope is 2 or 3.

knee2

Obtain the Knee2. This is only valid when the number of the slope is 3.

knee1

Obtain the Knee1. This is only valid when the number of the slope is 2 or 3.

[Descriptions]

This function obtains the HDR settings.

Please check HDR function availability with STC\_CID\_HDR\_TYPE.

## STCIOC\_S\_HDR

This function sets the HDR settings.

[Request parameters]

```
struct stcam_st_hdr {  
    union {  
        struct stcam_st_hdr_cmos_4m    cmosis_4m;  
    };  
};
```

```
struct stcam_st_hdr_cmos_4m {  
    __u8    mode;  
    __u8    slope;  
    __u8    vlow3;  
    __u8    vlow2;  
    __u8    knee2;  
    __u8    knee1;  
};
```

[Parameters]

mode

Set the HDR mode.

slope

Set the number of the slope.

vlow3

Set the Vlow3 voltage. This is only valid when the number of the slope is 3.

vlow2

Set the Vlow2 voltage. This is only valid when the number of the slope is 2 or 3.

knee2

Set the Knee2. This is only valid when the number of the slope is 3.



knee1

Set the Knee1. This is only valid when the number of the slope is 2 or 3.

[Descriptions]

This function sets the HDR settings.

Please check HDR function availability with STC\_CID\_HDR\_TYPE.

## STCIOC\_G\_DEF\_PIX\_CORRECT\_POS

This function obtains the location of the tareget pixel for the pixel defect correction.

[Request parameters]

```
struct stcam_st_def_pix_correct_pos {  
    __u8    index;  
    __u16   pos_x;  
    __u16   pos_y;  
};
```

[Parameters]

index

Obtain the index of the target pixel for the pixel defect correction.

pos\_x

Obtain the horizontal position of the target pixel for the pixel defect correction.

pos\_y

Obtain the vertical position of the target pixel for the pixel defect correction.

[Descriptions]

This function obtains the location of the tareget pixel for the pixel defect correction.

Please check the pixel defect correction availability with

STC\_CID\_DEFECT\_PIXEL\_CORRECTION\_COUNT.

## STCIOC\_S\_DEF\_PIX\_CORRECT\_POS

This function sets the location of the tareget pixel for the pixel defect correction.

[Request parameters]

```
struct stcam_st_def_pix_correct_pos {  
    __u8    index;  
    __u16   pos_x;  
    __u16   pos_y;  
};
```

[Parameters]

index

Set the index of the target pixel for the pixel defect correction.

pos\_x

Set the horizontal position of the target pixel for the pixel defect correction.

pos\_y

Set the vertical position of the target pixel for the pixel defect correction.

[Descriptions]

This function sets the location of the tareget pixel for the pixel defect correction.

Please check the pixel defect correction availability with

STC\_CID\_DEFECT\_PIXEL\_CORRECTION\_COUNT.

### 3.1.3 Camera settings

[VIDIOC\_G\_CTRL/S\_CTRL] of system call [ioctl] uses to read and write the camera settings.

The way to calling API is following the V4L2 standard.

The available function and range of the setting are different for each camera model. Please check the return of [VIDIOC\_G\_CTRL/S\_CTRL] of system call [ioctl] for availability for these.

#### **V4L2\_CID\_GAIN**

This function obtains the gain.

This function sets the gain.

[Type]

INTEGER

[Descriptions]

This function obtains the gain.

This function sets the gain.

## **V4L2\_CID\_EXPOSURE**

This function obtains the exposure time.

This function sets the exposure time.

[Type]

INTEGER

[Descriptions]

This function obtains the exposure time.

This function sets the exposure time.

When the exposure time is 0, the shutter is off and the exposure time is nearly same as the reciprocal of FPS [Hz].

When increasing the exposure time value, the exposure time is increasing.

## **V4L2\_CID\_HFLIP**

This function obtains the horizontal image flip mode.

This function sets the horizontal image flip mode.

[Type]

BOOLEAN

[Descriptions]

This function obtains the horizontal image flip mode.

This function sets the horizontal image flip mode.

The color array information may change when changing the horizontal image flip mode.

Please obtain the color array information with STC\_CID\_COLOR\_ARRAY after change the horizontal image flip mode.

**V4L2\_CID\_VFLIP**

This function obtains the vertical image flip mode.

This function sets the vertical image flip mode.

[Type]

BOOLEAN

[Descriptions]

This function obtains the vertical image flip mode.

This function sets the vertical image flip mode.

The color array information may change when changing the horizontal image flip mode.  
Please obtain the color array information with STC\_CID\_COLOR\_ARRAY after change the horizontal image flip mode.

**STC\_CID\_COLOR\_ARRAY**

This function obtains the color array information for the color camera.

[Type]

INTEGER

STC_MENU_COLOR_ARRAY_MONO	0x0001	Monochrome
STC_MENU_COLOR_ARRAY_RGGB	0x0002	Color: Bayer in order RGGB
STC_MENU_COLOR_ARRAY_GRGB	0x0003	Color: Bayer in order GRGB
STC_MENU_COLOR_ARRAY_GBRG	0x0004	Color: Bayer in order GBRG
STC_MENU_COLOR_ARRAY_BGGR	0x0005	Color: Bayer in order BGGR

[Descriptions]

This function obtains the color array information for the color camera.

This is read-only and the color model only..

## STC\_CID\_TRANSFER\_BITS\_PER\_PIXEL

This function obtains the transfer bits per pixel setting.

This function sets the transfer bits per pixel setting.

[Type]

MENU

STC_MENU_TRANSFER_BITS_PER_PIXEL_RAW_08	0	8bits (1 Byte) per one pixel for the transferring image. This is available for all cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_RAW_10	1	10bits (in 2 Bytes) per one pixel for the transferring image. This is only available for specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_RAW_10P	2	10bits per one pixel for the transferring image. 4 pixels are transferring by 5 Bytes. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_RAW_12	3	12bits (in 2 Bytes) per one pixel for the transferring image. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_RAW_12P	4	12bits per one pixel for the transferring image. 2 pixels are transferring by 3 Bytes. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_MONO_08	5	8bits (1 Byte) per one pixel for the transferring image. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_MONO_10	6	10bits (in 2 Bytes) per one pixel for the transferring image. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_MONO_10P	7	10bits per one pixel for the transferring image. 4 pixels are transferring by 5 Bytes. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_MONO_12	8	12bits (in 2 Bytes) per one pixel for the transferring image. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_MONO_12P	9	12bits per one pixel for the transferring image. 2 pixels are transferring by 3 Bytes. This is only available for the specific cameras.
STC_MENU_TRANSFER_BITS_PER_PIXEL_BGR_08	10	8bits RGB (24bits) per one pixel for the transferring image. 1 pixel transferring by 3 Bytes. This is only available for the specific cameras.

STC_MENU_TRANSFER_BITS_PER_PIXEL_BGR_10	11	10bits RGB (30bits) per one pixel for the transferring image. 1 pixel transferring by 4 Bytes.  This is only available for the specific cameras.
---	----	--

#### [Descriptions]

This function obtains the transfer bits per pixel setting.

This function sets the transfer bits per pixel setting.

When selecting other than STC\_MENU\_TRANSFER\_BITS\_PER\_PIXEL\_RAW\_08 and STC\_MENU\_TRANSFER\_BITS\_PER\_PIXEL\_MONO\_08, the bandwidth of USB bus is not enough due to increasing the amount of the image transferring data.

In this case, please select slower camera clock speed.

### **STC\_CID\_TEMPERATURE**

This function obtains the temperature information of the camera.

#### [Type]

INTEGER

#### [Descriptions]

This function obtains the temperature information of the camera.

This is read-only.

### **STC\_CID\_RESET\_COUNTER**

This function resets the inner frame counter of the camera.

#### [Type]

BUTTON

#### [Descriptions]

This function resets the inner frame counter of the camera.

This is read-only.



**STC\_CID\_CAMERA\_SETTING**

This function saves the camera settings to the camera.

This function loads the camera settings from the camera.

[Type]

MENU

STC_MENU_CAMERA_SETTING_INITIALIZE	0	Reset the setting of camera to default.
STC_MENU_CAMERA_SETTING_STANDARD_READ	1	Obtain the standard settings (Exposure time, gain, IO, trigger mode and others) from the camera.
STC_MENU_CAMERA_SETTING_STANDARD_WRITE	2	Set the standard settings (Exposure time, gain, IO, trigger mode and others) to the camera.
STC_MENU_CAMERA_SETTING_DEF_PIX_POS_READ	3	Obtain the location information of the pixel defect correction from the camera.
STC_MENU_CAMERA_SETTING_DEF_PIX_POS_WRITE	4	Set the location information of the pixel defect correction to the camera..

[Descriptions]

This function saves the camera settings to the camera.

This function loads the camera settings from the camera.

This is write-only.

## STC\_CID\_SCAN\_MODE

This function obtains the scan mode.

[Type]

MENU

STC_MENU_SCAN_MODE_NORMAL	0	Normal
STC_MENU_SCAN_MODE_PARTIAL_1	1	1/1 partial scanning mode
STC_MENU_SCAN_MODE_PARTIAL_2	2	1/2 partial scanning mode
STC_MENU_SCAN_MODE_PARTIAL_4	3	1/4 partial scanning mode
STC_MENU_SCAN_MODE_VARIABLE_PARTIAL	4	Variable partial scanning mode
STC_MENU_SCAN_MODE_BINNING	5	Binning mode
STC_MENU_SCAN_MODE_BINNING_PARTIAL_1	6	Binning 1/1 partial scanning mode
STC_MENU_SCAN_MODE_BINNING_PARTIAL_2	7	Binning 1/2 partial scanning mode
STC_MENU_SCAN_MODE_BINNING_PARTIAL_4	8	Binning 1/4 partial scanning mode
STC_MENU_SCAN_MODE_BINNING_VARIABLE_PARTIAL	9	Binning variable partial scanning mode
STC_MENU_SCAN_MODE_ROI	10	ROI mode
The width, height, and the horizontal and vertical offset for the image can be defined.		
This is only available for the specific cameras		

[Descriptions]

This function obtains the scan mode.

## STC\_CID\_MAX\_ROI\_COUNT

This function obtains the maximum number of the ROI.

[Type]

INTEGER

[Descriptions]

This function obtains the maximum number of the ROI.

If the camera does not support multi ROI, return 0.

This is read-only.

## STC\_CID\_V\_DECIMATION\_BINNING

This function obtains the vertical binning / vertical decimation setting.

This function sets the vertical binning / vertical decimation setting.

[Type]

MENU

STC_MENU_DECIMATION_BINNING_0_0	0	Decimation 0 / Binning 0
STC_MENU_DECIMATION_BINNING_1_0	1	Decimation 1 / Binning 0
STC_MENU_DECIMATION_BINNING_1_1	2	Decimation 1 / Binning 1
STC_MENU_DECIMATION_BINNING_1_2	3	Decimation 1 / Binning 2
STC_MENU_DECIMATION_BINNING_1_4	4	Decimation 1 / Binning 4
STC_MENU_DECIMATION_BINNING_2_0	5	Decimation 2 / Binning 0
STC_MENU_DECIMATION_BINNING_2_1	6	Decimation 2 / Binning 1
STC_MENU_DECIMATION_BINNING_2_2	7	Decimation 2 / Binning 2
STC_MENU_DECIMATION_BINNING_3_0	8	Decimation 3 / Binning 0
STC_MENU_DECIMATION_BINNING_3_1	9	Decimation 3 / Binning 1
STC_MENU_DECIMATION_BINNING_3_3	10	Decimation 3 / Binning 3
STC_MENU_DECIMATION_BINNING_4_0	11	Decimation 4 / Binning 0
STC_MENU_DECIMATION_BINNING_4_1	12	Decimation 4 / Binning 1
STC_MENU_DECIMATION_BINNING_4_2	13	Decimation 4 / Binning 2
STC_MENU_DECIMATION_BINNING_5_0	14	Decimation 5 / Binning 0
STC_MENU_DECIMATION_BINNING_5_1	15	Decimation 5 / Binning 1
STC_MENU_DECIMATION_BINNING_6_0	16	Decimation 6 / Binning 0
STC_MENU_DECIMATION_BINNING_7_0	17	Decimation 7 / Binning 0
STC_MENU_DECIMATION_BINNING_7_1	18	Decimation 7 / Binning 1
STC_MENU_DECIMATION_BINNING_7_3	19	Decimation 7 / Binning 3

[Descriptions]

This function obtains the vertical binning / vertical decimation setting.

This function sets the vertical binning / vertical decimation setting.

Selectable combination of the vertical binning and vertical decimation are different for each camera model.

#### STC-MBA5MUSB3 and STC-MCA5MUSB3

Decimation	Binning	Output size (Sensor size ratio)
0	0	1/1
1	0,1	1/2
2	0	1/3
3	0,1,3	1/4
4	0	1/5
5	0,1	1/6
6	0	1/7
7	0,1,3	1/8

#### STC-MBE132U3V and STC-MCE132U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
1	2	1/2
2	1	1/2
2	2	1/4
4	1	1/4
4	2	1/8

#### STC-MBCM401U3V and STC-MBCM200U3

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
1	2	1/2
1	4	1/4
2	1	1/2
4	1	1/4

#### STC-MCCM401U3V and STC-MCCM200U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2
4	1	1/4

STC-MBS241U3V, STC-MBS231U3V, STC-MBS510U3V, STC-MBS500U3V,  
STC-MBS322U3V and STC-MBS312U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2
1	2	1/2

STC-MCS241U3V, STC-MCS231U3V, STC-MCS510U3V, STC-MCS500U3V,  
STC-MCS322U3V and STC-MCS312U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2

## STC\_CID\_H\_DECIMATION\_BINNING

This function obtains the horizontal binning / horizontal decimation setting.

This function sets the horizontal binning / horizontal decimation setting.

[Type]

MENU

STC_MENU_DECIMATION_BINNING_0_0	0	Decimation 0 / Binning 0
STC_MENU_DECIMATION_BINNING_1_0	1	Decimation 1 / Binning 0
STC_MENU_DECIMATION_BINNING_1_1	2	Decimation 1 / Binning 1
STC_MENU_DECIMATION_BINNING_1_2	3	Decimation 1 / Binning 2
STC_MENU_DECIMATION_BINNING_1_4	4	Decimation 1 / Binning 4
STC_MENU_DECIMATION_BINNING_2_0	5	Decimation 2 / Binning 0
STC_MENU_DECIMATION_BINNING_2_1	6	Decimation 2 / Binning 1
STC_MENU_DECIMATION_BINNING_2_2	7	Decimation 2 / Binning 2
STC_MENU_DECIMATION_BINNING_3_0	8	Decimation 3 / Binning 0
STC_MENU_DECIMATION_BINNING_3_1	9	Decimation 3 / Binning 1
STC_MENU_DECIMATION_BINNING_3_3	10	Decimation 3 / Binning 3
STC_MENU_DECIMATION_BINNING_4_0	11	Decimation 4 / Binning 0
STC_MENU_DECIMATION_BINNING_4_1	12	Decimation 4 / Binning 1
STC_MENU_DECIMATION_BINNING_4_2	13	Decimation 4 / Binning 2
STC_MENU_DECIMATION_BINNING_5_0	14	Decimation 5 / Binning 0
STC_MENU_DECIMATION_BINNING_5_1	15	Decimation 5 / Binning 1
STC_MENU_DECIMATION_BINNING_6_0	16	Decimation 6 / Binning 0
STC_MENU_DECIMATION_BINNING_7_0	17	Decimation 7 / Binning 0
STC_MENU_DECIMATION_BINNING_7_1	18	Decimation 7 / Binning 1
STC_MENU_DECIMATION_BINNING_7_3	19	Decimation 7 / Binning 3

[Descriptions]

This function obtains the horizontal binning / horizontal decimation setting.

This function sets the horizontal binning / horizontal decimation setting.

Selectable combination of the horizontal binning and horizontal decimation are different for each camera model.

STC-MBA5MUSB3 and STC-MCA5MUSB3

Decimation	Binning	Output size (Sensor size ratio)
0	0	1/1
1	0,1	1/2
2	0	1/3
3	0,1,3	1/4
4	0	1/5
5	0,1	1/6
6	0	1/7

STC-MBE132U3V and STC-MCE132U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
1	2	1/2
2	1	1/2
2	2	1/4
4	1	1/4
4	2	1/8

STC-MBCM401U3V and STC-MBCM200U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
1	2	1/2
1	4	1/4
2	1	1/2
4	1	1/4

STC-MCCM401U3V and STC-MCCM200U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2
4	1	1/4

STC-MBS241U3V, STC-MBS231U3V, STC-MBS510U3V, STC-MBS500U3V,  
STC-MBS322U3V and STC-MBS312U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2
1	2	1/2

STC-MCS241U3V, STC-MCS231U3V, STC-MCS510U3V, STC-MCS500U3V,  
STC-MCS322U3V and STC-MCS312U3V

Decimation	Binning	Output size (Sensor size ratio)
1	1	1/1
2	1	1/2



**STC\_CID\_BINNINGSUM\_MODE**

This function obtains the pixel mixture mode for the binning mode.

This function sets the pixel mixture mode for the binning mode.

[Type]

MENU

STC_MENU_BINNINGSUM_MODE_OFF	0	No pixel mixture for the binning mode
STC_MENU_BINNINGSUM_MODE_H_ON	1	Horizontal pixel mixture for the binning mode
STC_MENU_BINNINGSUM_MODE_V_ON	2	Vertical pixel mixture for the binning mode

[Descriptions]

This function obtains the pixel mixture mode for the binning mode.

This function sets the pixel mixture mode for the binning mode.

**STC\_CID\_OUTPUT\_FPS**

This function obtains the frame rate (FPS) of the output image.

[Type]

INTEGER

[Descriptions]

This function obtains the frame rate (FPS) of the output image.

This is read-only.

## STC\_CID\_CLOCK

This function obtains the clock speed of the camera.

This function sets the clock speed of the camera.

[Type]

MENU

STC_MENU_CLOCK_NORMAL	0	Normal clock speed
STC_MENU_CLOCK_DIV_2	1	1/2 clock speed
STC_MENU_CLOCK_DIV_4	2	1/4 clock speed
STC_MENU_CLOCK_VGA_90FPS	3	90fps frame rate for VGA
This is only available for the specific cameras,		

[Descriptions]

This function obtains the clock speed of the camera.

This function sets the clock speed of the camera.

The frame rate increases when increasing the clock speed.

The frame rate decreases but the exposure time is extended when decreasing the clock speed. Also, it is possible to reduce the image drop off due the amount of the data per period of time on the USB bus is reduced.

## STC\_CID\_V\_BLANK

This function obtains the vertical blanking to adjust the frame rate.

This function sets the vertical blanking to adjust the frame rate.

[Type]

INTEGER

[Descriptions]

This function obtains the vertical blanking to adjust the frame rate.

This function sets the vertical blanking to adjust the frame rate.

The CPU usage is reduced when the frame rate is slowing down due to the vertical blanking is increased.

### **STC\_CID\_MAX\_SHORT\_EXPOSURE**

This function obtains the maximum exposure time setting can be set to the camera that keeps the frame rate.

[Type]

INTEGER

[Descriptions]

This function obtains the maximum exposure time setting can be set to the camera that keeps the frame rate.

This is read-only.

### **STC\_CID\_DIGITAL\_GAIN**

This function obtains the digital gain.

This function sets the digital gain.

[Type]

INTEGER

[Descriptions]

This function obtains the digital gain.

This function sets the digital gain.

**STC\_CID\_EXPOSURE\_MODE**

This function obtains the exposure mode.

This function sets the exposure mode.

[Type]

MENU

STC_MENU_EXPOSURE_MODE_OFF	0	OFF	Electronic shutter function is disabled.
STC_MENU_EXPOSURE_MODE_TIMED	1	Timed	The exposure time sets by V4L2_CID_EXPOSURE
STC_MENU_EXPOSURE_MODE_TRIGGER_WIDTH	2	Trigger width	The exposure time is the duration of the active trigger signal.
STC_MENU_EXPOSURE_MODE_TRIGGER_CONTROLLED	3	Trigger controlled	The exposure time is the period of time between the ExposureStart trigger signal and ExposureEnd trigger signal.

[Descriptions]

This function obtains the exposure mode.

This function sets the exposure mode.

## STC\_CID\_SENSOR\_SHUTTER\_MODE

This function obtains the sensor shutter mode.

This function sets the sensor shutter mode.

[Type]

MENU

STC_MENU_SENSOR_SHUTTER_MODE_ROLLING	0	Rolling
STC_MENU_SENSOR_SHUTTER_MODE_GLOBAL_RESET	1	Global reset
STC_MENU_SENSOR_SHUTTER_MODE_GLOBAL	2	Global

[Descriptions]

This function obtains the sensor shutter mode.

This function sets the sensor shutter mode.

## STC\_CID\_ALC\_MODE

This function obtains the ALC mode.

This function sets the ALC mode.

[Type]

MENU

STC_MENU_ALC_MODE_OFF	0	Disables the auto shutter and the AGC on the camera
STC_MENU_ALC_MODE_AE_ON	1	Enables the auto shutter on the camera
STC_MENU_ALC_MODE_AGC_ON	2	Enables the AGC on the camera
STC_MENU_ALC_MODE_AE_AGC_ON	3	Enables the auto shutter and the AGC on the camera

[Descriptions]

This function obtains the ALC mode.

This function sets the ALC mode.

### **STC\_CID\_ALC\_TARGET\_LEVEL**

This function obtains the target level for the ALC function.

This function sets the target level for the ALC function.

[Type]

INTEGER

[Descriptions]

This function obtains the target level for the ALC function.

This function sets the target level for the ALC function.

### **STC\_CID\_AGC\_MIN\_GAIN**

This function obtains the minimum gain for the AGC.

This function sets the minimum gain for the AGC.

[Type]

INTEGER

[Descriptions]

This function obtains the minimum gain for the AGC.

This function sets the minimum gain for the AGC.

### **STC\_CID\_AGC\_MAX\_GAIN**

This function obtains the maximum gain for the AGC.

This function sets the maximum gain for the AGC.

[Type]

INTEGER

[Descriptions]

This function obtains the maximum gain for the AGC.

This function sets the maximum gain for the AGC.

### **STC\_CID\_AE\_MIN\_EXPOSURE\_CLOCK**

This function obtains the minimum exposure time for the auto shutter.

This function sets the minimum exposure time for the auto shutter.

[Type]

INTEGER

[Descriptions]

This function obtains the minimum exposure time for the auto shutter.

This function sets the minimum exposure time for the auto shutter.

### **STC\_CID\_AE\_MAX\_EXPOSURE\_CLOCK**

This function obtains the maximum exposure time for the auto shutter.

This function sets the maximum exposure time for the auto shutter.

[Type]

INTEGER

[Descriptions]

This function obtains the maximum exposure time for the auto shutter.

This function sets the maximum exposure time for the auto shutter.

## **STC\_CID\_FRAME\_START\_TRIGGER\_MODE**

This function obtains enable / disable for the frame start trigger.

This function sets enable / disable for the frame start trigger.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the frame start trigger.

This function sets enable / disable for the frame start trigger.

## **STC\_CID\_FRAME\_START\_TRIGGER\_SOURCE**

This function obtains the trigger source for the frame start trigger.

This function sets the trigger source for the frame start trigger.

[Type]

MENU

STC_MENU_TRIGGER_SOURCE_DISABLED	0	Disabled
STC_MENU_TRIGGER_SOURCE_SOFTWARE	1	Software
STC_MENU_TRIGGER_SOURCE_HARDWARE	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE0	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE1	3	Line1
STC_MENU_TRIGGER_SOURCE_LINE2	4	Line2
STC_MENU_TRIGGER_SOURCE_LINE3	5	Line3

[Descriptions]

This function obtains the trigger source for the frame start trigger.

This function sets the trigger source for the frame start trigger.



### **STC\_CID\_FRAME\_START\_TRIGGER\_DELAY**

This function obtains the trigger delay for the frame start trigger.

This function sets the trigger delay for the frame start trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the trigger delay for the frame start trigger.

This function sets the trigger delay for the frame start trigger.

### **STC\_CID\_FRAME\_START\_TRIGGER\_OVERLAP**

This function obtains the trigger overlap for the frame start trigger.

This function sets the trigger overlap for the frame start trigger.

[Type]

MENU

STC_MENU_TRIGGER_OVERLAP_OFF	0	Trigger input is invalid until the read out finishes
STC_MENU_TRIGGER_OVERLAP_READ_OUT	1	Trigger input immediate after exposure end is valid
STC_MENU_TRIGGER_OVERLAP_PREVIOUS_FRAME	2	Trigger input is valid while the previous frame image transferring

[Descriptions]

This function obtains the trigger overlap for the frame start trigger.

This function sets the trigger overlap for the frame start trigger.

**STC\_CID\_FRAME\_BURST\_START\_TRIGGER\_MODE**

This function obtains enale / disable for the frame burst start trigger.

This function sets enale / disable for the frame burst start trigger.

[Type]

BOOLEAN

[Descriptions]

This function obtains enale / disable for the frame burst start trigger.

This function sets enale / disable for the frame burst start trigger.

**STC\_CID\_FRAME\_BURST\_START\_TRIGGER\_SOURCE**

This function obtains the trigger souce for the frame burst start trigger.

This function sets the trigger souce for the frame busrt start trigger.

[Type]

MENU

STC_MENU_TRIGGER_SOURCE_DISABLED	0	Disabled
STC_MENU_TRIGGER_SOURCE_SOFTWARE	1	Software
STC_MENU_TRIGGER_SOURCE_HARDWARE	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE0	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE1	3	Line1
STC_MENU_TRIGGER_SOURCE_LINE2	4	Line2
STC_MENU_TRIGGER_SOURCE_LINE3	5	Line3

[Descriptions]

This function obtains the trigger souce for the frame burst start trigger.

This function sets the trigger souce for the frame busrt start trigger.

### **STC\_CID\_FRAME\_BURST\_START\_TRIGGER\_DELAY**

This function obtains the trigger delay for the frame burst start trigger.

This function sets the trigger delay for the frame burst start trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the trigger delay for the frame burst start trigger.

This function sets the trigger delay for the frame burst start trigger.

### **STC\_CID\_FRAME\_BURST\_START\_TRIGGER\_OVERLAP**

This function obtains the trigger overlap for the frame burst start trigger.

This function sets the trigger overlap for the frame burst start trigger.

[Type]

MENU

STC_MENU_TRIGGER_OVERLAP_OFF	0	Trigger input is invalid until the read out finishes
STC_MENU_TRIGGER_OVERLAP_READ_OUT	1	Trigger input immediate after exposure end is valid
STC_MENU_TRIGGER_OVERLAP_PREVIOUS_FRAME	2	Trigger input is valid while the previous frame image transferring

[Descriptions]

This function obtains the trigger overlap for the frame burst start trigger.

This function sets the trigger overlap for the frame burst start trigger.

## STC\_CID\_EXPOSURE\_START\_TRIGGER\_MODE

This function obtains enable / disable for the exposure start trigger.

This function sets enable / disable for the exposure start trigger.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the exposure start trigger.

This function sets enable / disable for the exposure start trigger.

## STC\_CID\_EXPOSURE\_START\_TRIGGER\_SOURCE

This function obtains the trigger source for the exposure start trigger.

This function sets the trigger source for the exposure start trigger.

[Type]

MENU

STC_MENU_TRIGGER_SOURCE_DISABLED	0	Disabled
STC_MENU_TRIGGER_SOURCE_SOFTWARE	1	Software
STC_MENU_TRIGGER_SOURCE_HARDWARE	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE0	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE1	3	Line1
STC_MENU_TRIGGER_SOURCE_LINE2	4	Line2
STC_MENU_TRIGGER_SOURCE_LINE3	5	Line3

[Descriptions]

This function obtains the trigger source for the exposure start trigger.

This function sets the trigger source for the exposure start trigger.

### **STC\_CID\_EXPOSURE\_START\_TRIGGER\_DELAY**

This function obtains the trigger delay for the exposure start trigger.

This function sets the trigger delay for the exposure start trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the trigger delay for the exposure start trigger.

This function sets the trigger delay for the exposure start trigger.

### **STC\_CID\_EXPOSURE\_START\_TRIGGER\_OVERLAP**

This function obtains the trigger overlap for the exposure start trigger.

This function sets the trigger overlap for the exposure start trigger.

[Type]

MENU

STC_MENU_TRIGGER_OVERLAP_OFF	0	Trigger input is invalid until the read out finishes
STC_MENU_TRIGGER_OVERLAP_READ_OUT	1	Trigger input immediate after exposure end is valid
STC_MENU_TRIGGER_OVERLAP_PREVIOUS_FRAME	2	Trigger input is valid while the previous frame image transferring

[Descriptions]

This function obtains the trigger overlap for the exposure start trigger.

This function sets the trigger overlap for the exposure start trigger.

## STC\_CID\_EXPOSURE\_END\_TRIGGER\_MODE

This function obtains enable / disable for the exposure end trigger.

This function sets enable / disable for the exposure end trigger.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the exposure end trigger.

This function sets enable / disable for the exposure end trigger.

## STC\_CID\_EXPOSURE\_END\_TRIGGER\_SOURCE

This function obtains the trigger source for the exposure end trigger.

This function sets the trigger source for the exposure end trigger.

[Type]

MENU

STC_MENU_TRIGGER_SOURCE_DISABLED	0	Disabled
STC_MENU_TRIGGER_SOURCE_SOFTWARE	1	Software
STC_MENU_TRIGGER_SOURCE_HARDWARE	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE0	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE1	3	Line1
STC_MENU_TRIGGER_SOURCE_LINE2	4	Line2
STC_MENU_TRIGGER_SOURCE_LINE3	5	Line3

[Descriptions]

This function obtains the trigger source for the exposure end trigger.

This function sets the trigger source for the exposure end trigger.

### **STC\_CID\_EXPOSURE\_END\_TRIGGER\_DELAY**

This function obtains the trigger delay for the exposure end trigger.

This function sets the trigger delay for the exposure end trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the trigger delay for the exposure end trigger.

This function sets the trigger delay for the exposure end trigger.

### **STC\_CID\_EXPOSURE\_END\_TRIGGER\_OVERLAP**

This function obtains the trigger overlap for the exposure end trigger.

This function sets the trigger overlap for the exposure end trigger.

[Type]

MENU

STC_MENU_TRIGGER_OVERLAP_OFF	0	Trigger input is invalid until the read out finishes
STC_MENU_TRIGGER_OVERLAP_READ_OUT	1	Trigger input immediate after exposure end is valid
STC_MENU_TRIGGER_OVERLAP_PREVIOUS_FRAME	2	Trigger input is valid while the previous frame image transferring

[Descriptions]

This function obtains the trigger overlap for the exposure end trigger.

This function sets the trigger overlap for the exposure end trigger.

## **STC\_CID\_READ\_OUT\_START\_TRIGGER\_MODE**

This function obtains enable / disable for the readout start trigger.

This function sets enable / disable for the readout start trigger.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the readout start trigger.

This function sets enable / disable for the readout start trigger.

## **STC\_CID\_READ\_OUT\_START\_TRIGGER\_SOURCE**

This function obtains the trigger source for the readout start trigger.

This function sets the trigger source for the readout start trigger.

[Type]

MENU

STC_MENU_TRIGGER_SOURCE_DISABLED	0	Disabled
STC_MENU_TRIGGER_SOURCE_SOFTWARE	1	Software
STC_MENU_TRIGGER_SOURCE_HARDWARE	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE0	2	Hardware/Line0
STC_MENU_TRIGGER_SOURCE_LINE1	3	Line1
STC_MENU_TRIGGER_SOURCE_LINE2	4	Line2
STC_MENU_TRIGGER_SOURCE_LINE3	5	Line3

[Descriptions]

This function obtains the trigger source for the readout start trigger.

This function sets the trigger source for the readout start trigger.



### **STC\_CID\_READ\_OUT\_START\_TRIGGER\_DELAY**

This function obtains the trigger delay for the readout start trigger.

This function sets the trigger delay for the readout start trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the trigger delay for the readout start trigger.

This function sets the trigger delay for the readout start trigger.

### **STC\_CID\_READ\_OUT\_START\_TRIGGER\_OVERLAP**

This function obtains the trigger overlap for the readout start trigger.

This function sets the trigger overlap for the readout start trigger.

[Type]

MENU

STC_MENU_TRIGGER_OVERLAP_OFF	0	Trigger input is invalid until the read out finishes
STC_MENU_TRIGGER_OVERLAP_READ_OUT	1	Trigger input immediate after exposure end is valid
STC_MENU_TRIGGER_OVERLAP_PREVIOUS_FRAME	2	Trigger input is valid while the previous frame image transferring

[Descriptions]

This function obtains the trigger overlap for the readout start trigger.

This function sets the trigger overlap for the readout start trigger.

## STC\_CID\_TRIGGER\_SOFTWARE

This function sends the software trigger signal to the camera.

[Type]

MENU

STC_MENU_TRIGGER_SOFTWARE_FRAME_START	0	Frame start
STC_MENU_TRIGGER_SOFTWARE_FRAME_BURST_START	1	Frame burst start
STC_MENU_TRIGGER_SOFTWARE_EXPOSURE_START	2	Exposure start
STC_MENU_TRIGGER_SOFTWARE_EXPOSURE_END	3	Exposure end
STC_MENU_TRIGGER_SOFTWARE_READ_OUT_START	4	Read out start

[Descriptions]

This function sends the software trigger signal to the camera.

This is write-only.

## STC\_CID\_TRIGGER\_WAIT\_HD

This function obtains the exposure starts with the next HD timing after the trigger signal input or without.

This function sets the exposure starts with the next HD timing after the trigger signal input or without.

[Type]

BOOLEAN

[Descriptions]

This function obtains the exposure starts with the next HD timing after the trigger signal input or without.

This function sets the exposure starts with the next HD timing after the trigger signal input or without.

### **STC\_CID\_TRIGGER\_WAIT\_READOUT**

This function obtains the exposure starts adjusting if the exposure does not finish when the image out for the previous trigger signal is finished or no start timing adjustment.

This function sets the exposure starts adjusting if the exposure does not finish when the image out for the previous trigger signal is finished or no start timing adjustment.

[Type]

BOOLEAN

[Descriptions]

This function obtains the exposure starts adjusting if the exposure does not finish when the image out for the previous trigger signal is finished or no start timing adjustment.

This function sets the exposure starts adjusting if the exposure does not finish when the image out for the previous trigger signal is finished or no start timing adjustment.

### **STC\_CID\_EXPOSURE\_DELAY**

This function obtains the delay of the exposure start.

This function sets the delay of the exposure start.

[Type]

INTEGER

[Descriptions]

This function obtains the delay of the exposure start.

This function sets the delay of the exposure start.

### **STC\_CID\_STROBE\_START\_DELAY**

This function obtains the output delay of the strobe control signal.

This function sets the output delay of the strobe control signal.

[Type]

INTEGER

[Descriptions]

This function obtains the output delay of the strobe control signal.

This function sets the output delay of the strobe control signal.

### **STC\_CID\_STROBE\_END\_DELAY**

This function obtains the pulse duration of the strobe control signal.

This function sets the pulse duration of the strobe control signal.

[Type]

INTEGER

[Descriptions]

This function obtains the pulse duration of the strobe control signal.

This function sets the pulse duration of the strobe control signal.

### **STC\_CID\_OUTPUT\_PULSE\_DELAY**

This function obtains the delay of the output signal.

This function sets the delay of the output signal.

[Type]

INTEGER

[Descriptions]

This function obtains the delay of the output signal.

This function sets the delay of the output signal.

### **STC\_CID\_OUTPUT\_PULSE\_DURATION**

This function obtains the pulse duration of the output signal.

This function sets the pulse duration of the output signal.

[Type]

INTEGER

[Descriptions]

This function obtains the pulse duration of the output signal.

This function sets the pulse duration of the output signal.

### **STC\_CID\_READOUT\_DELAY**

This function obtains the image readout delay.

This function sets the image readout delay.

[Type]

INTEGER

[Descriptions]

This function obtains the image readout delay.

This function sets the image readout delay.

### **STC\_CID\_LINE\_DEBOUNCE\_TIME**

This function obtains the debounce filter for the input signal through Line.

This function sets the debounce filter for the input signal through Line.

[Type]

INTEGER

[Descriptions]

This function obtains the debounce filter for the input signal through Line.

This function sets the debounce filter for the input signal through Line.

### STC\_CID\_IO0\_DIRECTION

This function obtains the signal direction (input/output) for the IO0.

This function sets the signal direction (input/output) for the IO0.

[Type]

MENU

STC_MENU_IO_DIRECTION_INPUT	0	Input
STC_MENU_IO_DIRECTION_OUTPUT	1	Output

[Descriptions]

This function obtains the signal direction (input/output) for the IO0.

This function sets the signal direction (input/output) for the IO0.

### STC\_CID\_IO0\_POLARITY

This function obtains the polarity of the signal through IO0.

This function sets the polarity of the signal through IO0.

[Type]

MENU

STC_MENU_IO_POLARITY_POSI	0	Positive
STC_MENU_IO_POLARITY_NEGA	1	Negative

[Descriptions]

This function obtains the polarity of the signal through IO0.

This function sets the polarity of the signal through IO0.

## STC\_CID\_IO0\_MODE

This function obtains the signal mode for the IO0.

This function sets the signal mode for the IO0.

[Type]

MENU

STC_MENU_IO_MODE_OUT_DISABLE	0	Disables the output port
STC_MENU_IO_MODE_OUT_GENERAL_OUTPUT	1	Enables the output port for the general signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_PROGRAMMABLE	2	Enables the output port for the trigger signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_LOOP_THROUGH	3	Enables the output port for the trigger signal output that is the input trigger signal loop through the output
STC_MENU_IO_MODE_OUT_EXPOSURE_END	4	Enables the output port for the exposure end signal output
STC_MENU_IO_MODE_OUT_CCD_READ_END_OUTPUT	5	Enables the output port for the image readout end signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_PROGRAMMABLE	6	Enables the output port for the strobe control signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_EXPOSURE	7	Enables the output port for the strobe control signal output that is the same as the exposure time and timing
STC_MENU_IO_MODE_OUT_TRIGGER_VALID_OUT	8	Enables the output port for the trigger signal invalid period of time
STC_MENU_IO_MODE_OUT_TRANSFER_END	9	Enables the output port for the transfer end signal output
STC_MENU_IO_MODE_IN_DISABLE	10	Disables the input port
STC_MENU_IO_MODE_IN_GENERAL_INPUT	11	Enables the input port for the general signal input
STC_MENU_IO_MODE_IN_TRIGGER_INPUT	12	Enables the input port for the trigger signal input
STC_MENU_IO_MODE_IN_READOUT_INPUT	13	Enables the input port for the image readout signal input

STC_MENU_IO_MODE_IN_SUB_TRIGGER_INPUT	14	Enables the input port for the second trigger signal for the start and stop trigger mode
---------------------------------------	----	--

#### [Descriptions]

This function obtains the signal mode for the IO0.

This function sets the signal mode for the IO0.

### STC\_CID\_IO0\_STATUS

This function obtains the status of the general input signal of the IO0.

This function sets the status of the general output signal of the IO0.

#### [Type]

MENU

STC_MENU_IO_STATUS_LOW	0	Low
STC_MENU_IO_STATUS_HIGH	1	High

#### [Descriptions]

This function obtains the status of the general input signal of the IO0.

This function sets the status of the general output signal of the IO0.



## STC\_CID\_IO1\_DIRECTION

This function obtains the signal direction (input/output) for the IO1.

This function sets the signal direction (input/output) for the IO1.

[Type]

MENU

STC_MENU_IO_DIRECTION_INPUT	0	Input
STC_MENU_IO_DIRECTION_OUTPUT	1	Output

[Descriptions]

This function obtains the signal direction (input/output) for the IO1.

This function sets the signal direction (input/output) for the IO1.

## STC\_CID\_IO1\_POLARITY

This function obtains the polarity of the signal through IO1.

This function sets the polarity of the signal through IO1.

[Type]

MENU

STC_MENU_IO_POLARITY_POSI	0	Positive
STC_MENU_IO_POLARITY_NEGA	1	Negative

[Descriptions]

This function obtains the polarity of the signal through IO1.

This function sets the polarity of the signal through IO1.

## STC\_CID\_IO1\_MODE

This function obtains the signal mode for the IO1.

This function sets the signal mode for the IO1.

[Type]

MENU

STC_MENU_IO_MODE_OUT_DISABLE	0	Disables the output port
STC_MENU_IO_MODE_OUT_GENERAL_OUTPUT	1	Enables the output port for the general signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_PROGRAMMABLE	2	Enables the output port for the trigger signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_LOOP_THROUGH	3	Enables the output port for the trigger signal output that is the input trigger signal loop through the output
STC_MENU_IO_MODE_OUT_EXPOSURE_END	4	Enables the output port for the exposure end signal output
STC_MENU_IO_MODE_OUT_CCD_READ_END_OUTPUT	5	Enables the output port for the image readout end signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_PROGRAMMABLE	6	Enables the output port for the strobe control signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_EXPOSURE	7	Enables the output port for the strobe control signal output that is the same as the exposure time and timing
STC_MENU_IO_MODE_OUT_TRIGGER_VALID_OUT	8	Enables the output port for the trigger signal invalid period of time
STC_MENU_IO_MODE_OUT_TRANSFER_END	9	Enables the output port for the transfer end signal output
STC_MENU_IO_MODE_IN_DISABLE	10	Disables the input port
STC_MENU_IO_MODE_IN_GENERAL_INPUT	11	Enables the input port for the general signal input
STC_MENU_IO_MODE_IN_TRIGGER_INPUT	12	Enables the input port for the trigger signal input
STC_MENU_IO_MODE_IN_READOUT_INPUT	13	Enables the input port for the image readout signal input

STC_MENU_IO_MODE_IN_SUB_TRIGGER_INPUT	14	Enables the input port for the second trigger signal for the start and stop trigger mode
---------------------------------------	----	--

#### [Descriptions]

This function obtains the signal mode for the IO1.

This function sets the signal mode for the IO1.

### STC\_CID\_IO1\_STATUS

This function obtains the status of the general input signal of the IO1.

This function sets the status of the general output signal of the IO1.

#### [Type]

MENU

STC_MENU_IO_STATUS_LOW	0	Low
STC_MENU_IO_STATUS_HIGH	1	High

#### [Descriptions]

This function obtains the status of the general input signal of the IO1.

This function sets the status of the general output signal of the IO1.

## STC\_CID\_IO2\_DIRECTION

This function obtains the signal direction (input/output) for the IO2.

This function sets the signal direction (input/output) for the IO2.

[Type]

MENU

STC_MENU_IO_DIRECTION_INPUT	0	Input
STC_MENU_IO_DIRECTION_OUTPUT	1	Output

[Descriptions]

This function obtains the signal direction (input/output) for the IO2.

This function sets the signal direction (input/output) for the IO2.

## STC\_CID\_IO2\_POLARITY

This function obtains the polarity of the signal through IO2.

This function sets the polarity of the signal through IO2.

[Type]

MENU

STC_MENU_IO_POLARITY_POSI	0	Positive
STC_MENU_IO_POLARITY_NEGA	1	Negative

[Descriptions]

This function obtains the polarity of the signal through IO2.

This function sets the polarity of the signal through IO2.

## STC\_CID\_IO2\_MODE

This function obtains the signal mode for the IO2.

This function sets the signal mode for the IO2.

[Type]

MENU

STC_MENU_IO_MODE_OUT_DISABLE	0	Disables the output port
STC_MENU_IO_MODE_OUT_GENERAL_OUTPUT	1	Enables the output port for the general signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_PROGRAMMABLE	2	Enables the output port for the trigger signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_LOOP_THROUGH	3	Enables the output port for the trigger signal output that is the input trigger signal loop through the output
STC_MENU_IO_MODE_OUT_EXPOSURE_END	4	Enables the output port for the exposure end signal output
STC_MENU_IO_MODE_OUT_CCD_READ_END_OUTPUT	5	Enables the output port for the image readout end signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_PROGRAMMABLE	6	Enables the output port for the strobe control signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_EXPOSURE	7	Enables the output port for the strobe control signal output that is the same as the exposure time and timing
STC_MENU_IO_MODE_OUT_TRIGGER_VALID_OUT	8	Enables the output port for the trigger signal invalid period of time
STC_MENU_IO_MODE_OUT_TRANSFER_END	9	Enables the output port for the transfer end signal output
STC_MENU_IO_MODE_IN_DISABLE	10	Disables the input port
STC_MENU_IO_MODE_IN_GENERAL_INPUT	11	Enables the input port for the general signal input
STC_MENU_IO_MODE_IN_TRIGGER_INPUT	12	Enables the input port for the trigger signal input
STC_MENU_IO_MODE_IN_READOUT_INPUT	13	Enables the input port for the image readout signal input

STC_MENU_IO_MODE_IN_SUB_TRIGGER_INPUT	14	Enables the input port for the second trigger signal for the start and stop trigger mode
---------------------------------------	----	--

#### [Descriptions]

This function obtains the signal mode for the IO2.

This function sets the signal mode for the IO2.

### STC\_CID\_IO2\_STATUS

This function obtains the status of the general input signal of the IO2.

This function sets the status of the general output signal of the IO2.

#### [Type]

MENU

STC_MENU_IO_STATUS_LOW	0	Low
STC_MENU_IO_STATUS_HIGH	1	High

#### [Descriptions]

This function obtains the status of the general input signal of the IO2.

This function sets the status of the general output signal of the IO2.

### STC\_CID\_IO3\_DIRECTION

This function obtains the signal direction (input/output) for the IO3.

This function sets the signal direction (input/output) for the IO3.

[Type]

MENU

STC_MENU_IO_DIRECTION_INPUT	0	Input
STC_MENU_IO_DIRECTION_OUTPUT	1	Output

[Descriptions]

This function obtains the signal direction (input/output) for the IO3.

This function sets the signal direction (input/output) for the IO3.

### STC\_CID\_IO3\_POLARITY

This function obtains the polarity of the signal through IO3.

This function sets the polarity of the signal through IO3.

[Type]

MENU

STC_MENU_IO_POLARITY_POSI	0	Positive
STC_MENU_IO_POLARITY_NEGA	1	Negative

[Descriptions]

This function obtains the polarity of the signal through IO3.

This function sets the polarity of the signal through IO3.

## STC\_CID\_IO3\_MODE

This function obtains the signal mode for the IO3.

This function sets the signal mode for the IO3.

[Type]

MENU

STC_MENU_IO_MODE_OUT_DISABLE	0	Disables the output port
STC_MENU_IO_MODE_OUT_GENERAL_OUTPUT	1	Enables the output port for the general signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_PROGRAMMABLE	2	Enables the output port for the trigger signal output
STC_MENU_IO_MODE_OUT_TRIGGER_OUTPUT_LOOP_THROUGH	3	Enables the output port for the trigger signal output that is the input trigger signal loop through the output
STC_MENU_IO_MODE_OUT_EXPOSURE_END	4	Enables the output port for the exposure end signal output
STC_MENU_IO_MODE_OUT_CCD_READ_END_OUTPUT	5	Enables the output port for the image readout end signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_PROGRAMMABLE	6	Enables the output port for the strobe control signal output
STC_MENU_IO_MODE_OUT_STROBE_OUTPUT_EXPOSURE	7	Enables the output port for the strobe control signal output that is the same as the exposure time and timing
STC_MENU_IO_MODE_OUT_TRIGGER_VALID_OUT	8	Enables the output port for the trigger signal invalid period of time
STC_MENU_IO_MODE_OUT_TRANSFER_END	9	Enables the output port for the transfer end signal output
STC_MENU_IO_MODE_IN_DISABLE	10	Disables the input port
STC_MENU_IO_MODE_IN_GENERAL_INPUT	11	Enables the input port for the general signal input
STC_MENU_IO_MODE_IN_TRIGGER_INPUT	12	Enables the input port for the trigger signal input
STC_MENU_IO_MODE_IN_READOUT_INPUT	13	Enables the input port for the image readout signal input



STC_MENU_IO_MODE_IN_SUB_TRIGGER_INPUT	14	Enables the input port for the second trigger signal for the start and stop trigger mode
---------------------------------------	----	--

[Descriptions]

This function obtains the signal mode for the IO3

This function sets the signal mode for the IO3.

**STC\_CID\_IO3\_STATUS**

This function obtains the status of the general input signal of the IO3.

This function sets the status of the general output signal of the IO3.

[Type]

MENU

STC_MENU_IO_STATUS_LOW	0	Low
STC_MENU_IO_STATUS_HIGH	1	High

[Descriptions]

This function obtains the status of the general input signal of the IO3.

This function sets the status of the general output signal of the IO3.

## **STC\_CID\_LED\_STATUS**

This function obtains the status of the LED.

This function sets the status of the LED.

[Type]

MENU

STC_MENU_LED_STATUS_OFF	0	Off
STC_MENU_LED_STATUS_GREEN_ON	1	Green LED is On.
STC_MENU_LED_STATUS_RED_ON	2	Red LED is On.
STC_MENU_LED_STATUS_GREEN_RED_ON	3	Green and Red LEDs are On

[Descriptions]

This function obtains the status of the LED.

This function sets the status of the LED.

## **STC\_CID\_RESET\_SWITCH**

This function obtains the enables / disables setting for the camera reset switch.

This function sets the enables / disables setting for the camera reset switch.

[Type]

BOOLEAN

[Descriptions]

This function obtains the enables / disables setting for the camera reset switch.

This function sets the enables / disables setting for the camera reset switch.

### **STC\_CID\_SW\_STATUS**

This function obtains the status of the DIP switch of the camera.

[Type]

INTEGER

[Descriptions]

This function obtains the status of the DIP switch of the camera.

Bit0 represents the SW1, bit1 represents the SW2, bit2 represents the SW3 and bit3 represents the SW4.

The DIP Switch is OFF when obtaining "0". The DIP Switch is ON when obtaining "1".

This is read-only.

### **STC\_CID\_TIMEOUT\_ST2EE**

This function obtains the time out time for the software trigger.

This function sets the time out time for the software trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the time out time for the software trigger.

This function sets the time out time for the software trigger.

Set the time out time from call STC\_CID\_TRIGGER\_SOFTWARE to detect the exposure end for the software trigger.

### **STC\_CID\_TIMEOUT\_TE2EE**

This function obtains the time out time for the hardware trigger.

This function sets the time out time for the hardware trigger.

[Type]

INTEGER

[Descriptions]

This function obtains the time out time for the hardware trigger.

This function sets the time out time for the hardware trigger.

Set the timeout time from the image out end for the previous image to detect the exposure end for the hardware trigger.

### **STC\_CID\_TIMEOUT\_EE2TE**

This function obtains the time out time for the exposure time.

This function sets the time out time for the exposure time.

[Type]

INTEGER

[Descriptions]

This function obtains the time out time for the exposure time.

This function sets the time out time for the exposure time.

Set the time out time from detect the exposure end to the image out end whe the exposure end callback function is enabled.

### **STC\_CID\_TIMEOUT\_RO2TE**

This function obtains the time out time for the readout.

This function sets the time out time for the readout.

[Type]

INTEGER

[Descriptions]

This function obtains the time out time for the readout.

This function sets the time out time for the readout.

Set the time out time from call STC\_CID\_TRIGGER\_SOFTWARE to the image out end for the software readout trigger mode.

### **STC\_CID\_EXPOSURE\_END\_CALLBACK**

This function obtains enable / disable for the callback function that calls after the exposure is finished.

This function sets enable / disable for the callback function that calls after the exposure is finished.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the callback function that calls after the exposure is finished.

This function sets enable / disable for the callback function that calls after the exposure is finished.

The signal with below ID of Linux is calling when the exposure is finished, if this function is enabled.

STC\_SIGNAL\_CALLBACK\_EXPOSURE\_END      34

## **STC\_CID\_TRANSFER\_END\_CALLBACK**

This function obtains enable / disable for the callback function that calls after the image out is finished.

This function sets enable / disable for the callback function that calls after the image out is finished.

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the callback function that calls after the image out is finished.

This function sets enable / disable for the callback function that calls after the image out is finished.

The signal with below ID of Linux is calling when the image out is finished, if this function is enabled.

STC\_SIGNAL\_CALLBACK\_TRANSFER\_END      35

## **STC\_CID\_RCV\_ERROR\_CALLBACK**

This function obtains enable / disable for the callback function that calls when the error is occurred

This function sets enable / disable for the callback function that calls when the error is occurred

[Type]

BOOLEAN

[Descriptions]

This function obtains enable / disable for the callback function that calls when the error is occurred

This function sets enable / disable for the callback function that calls when the error is occurred

The signal with below ID of Linux is calling when the error is occurred, if this function is enabled.

This signal also call if set the time out time.

STC\_SIGNAL\_CALLBACK\_ERROR            36

The signal parameter contains the error code of Linux or below timeout error code.

STC\_ERROR\_TIMEOUT\_ST2EE 1

STC\_ERROR\_TIMEOUT\_TE2EE 2

STC\_ERROR\_TIMEOUT\_EE2TE 3

STC\_ERROR\_TIMEOUT\_RO2TE 4

**STC\_CID\_HDR\_TYPE**

This function obtains the supported HDR type.

[Type]

INTEGER

STC_MENU_HDR_TYPE_NONE	0x0000	HDR function does not support
STC_MENU_HDR_TYPE_CMOSIS_4M	0x0001	HDR function supported by the CMOSIS sensor.

[Descriptions]

This function obtains the supported HDR type.

This is read-only.

**STC\_CID\_WHITE\_BALANCE\_MODE**

This function obtains the white balance mode.

This function sets the white balance mode.

[Type]

MENU

STC_MENU_WHITE_BALANCE_MODE_OFF	0	No white balance process
STC_MENU_WHITE_BALANCE_MODE_MANUAL	1	Manual white balance process with the preset gain settings
STC_MENU_WHITE_BALANCE_MODE_FULLAUTO	2	Auto white balance process
2		
STC_MENU_WHITE_BALANCE_MODE_ONESHOT	3	Auto white balance process once to defaming the white balance then change to the manual white balance
		"
		STC_MENU_WHITE_BALANCE_MODE_MANUAL
		" automatically

[Descriptions]

This function obtains the white balance mode.

This function sets the white balance mode.



### **STC\_CID\_WHITE\_BALANCE\_GAIN\_R**

This function obtains the white balance R gain.

This function sets the white balance R gain.

[Type]

INTEGER

[Descriptions]

This function obtains the white balance R gain.

This function sets the white balance R gain.

### **STC\_CID\_WHITE\_BALANCE\_GAIN\_GR**

This function obtains the white balance Gr gain.

This function sets the white balance Gr gain.

[Type]

INTEGER

[Descriptions]

This function obtains the white balance Gr gain.

This function sets the white balance Gr gain.

### **STC\_CID\_WHITE\_BALANCE\_GAIN\_GB**

This function obtains the white balance Gb gain.

This function sets the white balance Gb gain.

[Type]

INTEGER

[Descriptions]

This function obtains the white balance Gb gain.

This function sets the white balance Gb gain.

### **STC\_CID\_WHITE\_BALANCE\_GAIN\_B**

This function obtains the white balance B gain.

This function sets the white balance B gain.

[Type]

INTEGER

[Descriptions]

This function obtains the white balance B gain.

This function sets the white balance B gain.

### **STC\_CID\_CAMERA\_GAMMA**

This function obtains the gamma setting for the gamma process on the camera.

This function sets the gamma setting for the gamma process on the camera.

[Type]

INTEGER

[Descriptions]

This function obtains the gamma setting for the gamma process on the camera.

This function sets the gamma setting for the gamma process on the camera.

### **STC\_CID\_DEFECT\_PIXEL\_CORRECTION\_COUNT**

This function obtains the maximum number of the pixel defect can be correct for this camera.

[Type]

INTEGER

[Descriptions]

This function obtains the maximum number of the pixel defect can be correct for this camera.

Obtain the more than 1 if the camera supports this function.

Obtain 0 if the camera does not support this function.

This is read-only.

### **STC\_CID\_DEFECT\_PIXEL\_CORRECTION\_MODE**

This function obtains the pixel defect correction mode.

[Type]

BOOLEAN

[Descriptions]

This function obtains the pixel defect correction mode.

### **STC\_CID\_DIGITAL\_CLAMP**

This function obtains the digital clamp.

This function sets the digital clamp.

[Type]

INTEGER

[Descriptions]

This function obtains the digital clamp.

This function sets the digital clamp.

### **STC\_CID\_ANALOG\_BLACK\_LEVEL**

This function obtains the analog black level.

This function sets the analog black level.

[Type]

INTEGER

[Descriptions]

This function obtains the analog black level.

This function sets the analog black level.

### **STC\_CID\_ADJUSTMENT\_MODE**

This function obtains the factory adjustment mode.

This function sets the factory adjustment mode.

[Type]

BOOLEAN

[Descriptions]

This function obtains the factory adjustment mode.

This function sets the factory adjustment mode.

This is only available for the specific cameras.

## **STC\_CID\_CLEAR\_BUFFER**

This function clears the buffer in the camera.

[Type]

BUTTON

[Descriptions]

This function clears the buffer in the camera.

The incorrect procedure (like "send the trigger signal before the image out is finished" or "send the image out signal before the exposure is finished") is occurred while the trigger function, the wrong data remains on the buffer of the camera. In this case, the buffer of the camera can clean by this command after stops the transferring by this function.

This is write-only.

## 3.2 Library functions

### **StCam\_Open**

This function initializes the libraries then obtains the library handle.

```
void* StCam_Open(  
    int devHandle    // Device handle  
);
```

#### [Parameters]

devHandle

Set the library handle that obtained by the StCam\_Open function.

#### [Return]

If the function was successful, return the library handle.

If the function failed, return NULL.

#### [Descriptions]

This function initializes the libraries then obtains the library handle.

The library handle use for call the library functions.

It is necessary to close the opened libraries with StCam\_Close function before finish to use the camera.

## **StCam\_Close**

This function closes the libraries.

```
void StCam_Close(  
    void* hCamera    // Library handle  
);
```

### [Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

### [Return]

None.

### [Descriptions]

This function closes the libraries.

It is necessary to close the opened libraries with StCam\_Close function before finish to use the camera.

### **StCam\_GetlibVersion**

This function obtains the version information of the libraries.

```
int StCam_GetlibVersion(  
    void* hCamera,                // Library handle  
    unsigned long* pdwLibVersion  // Library version  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pdwLibVersion**

Set the pointer that obtains the library version.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the version information of the libraries.



### **StCam\_WriteSettingFile**

This function saves the current library settings to the file.

```
int StCam_WriteSettingFile(  
    void* hCamera,      // Library handle  
    char* pszFileName  // Setting file name  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pszFileName**

Set the pointer of the file name that end with “NULL” for save the library setting file.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function saves the current library settings to the file.

### **StCam\_ReadSettingFile**

This function loads the library settings from the file.

```
int StCam_ReadSettingFile(  
    void* hCamera,      // Library handle  
    char* pszFileName  // Setting file name  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pszFileName**

Set the pointer of the file name that end with “NULL” for load the library setting file.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function loads the library settings from the file.

### StCam\_ConvTo8BitsImage

This function converts to the image that is 8btis represented in each pixel image.

```
int StCam_ConvTo8BitsImage(  
    void* hCamera,                // Library handle  
    unsigned long dwWidth,        // Width  
    unsigned long dwHeight,       // Height  
    unsigned long dwSrcLinePitch, // Number of byte for 1 line  
    unsigned long dwTransferBitsPerPixel, // Number of bits per pixel for  
transferring image  
    unsigned short* pwRaw,        // Input image  
    unsigned long* pdwDestLinePitch, // Number of byte for 1 line  
    unsigned char* pbyteRaw       // Output image  
);
```

#### [Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

dwWidth

Set the with of the image.

dwHeight

Set the height of the image.

dwSrcLinePitch

Set the number of the bytes for 1 line of the image.

dwTransferBitsPerPixel

Set the number of the bits represented in each pixel for transferring image that obtains by STC\_CID\_TRANSFER\_BITS\_PER\_PIXEL.

pwRaw

Set the pointer of the input image.

pdwDestLinePitch

Set the pointer of the number of the bytes for 1 line of the output image.

pbyteRaw

Set the pointer of the output image.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function converts to the image that is 8bits represented in each pixel image.

It is necessary to convert from 10bits or 12bits RAW image data to 8bits image data with this command before use the image process function of this library.

### **StCam\_ConvYUVOrBGRToBGRImage**

Convert from the YUV or BGR image to the BGR image.

```
int StCam_ConvYUVOrBGRToBGRImage(
    void* hCamera,                // Library handle
    unsigned long dwWidth,        // Width
    unsigned long dwHeight,       // Height
    unsigned long dwTransferBitsPerPixel, // Number of bits represented in
each pixel for transferring image
    unsigned char* pbyteYUVOrBGR, // YUV (or BGR) image buffer
    unsigned long dwPixelFormat,   // Pixel format
    unsigned char* pbyteBGR        // BGR image buffer
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**dwTransferBitsPerPixel**

Set the number of the bits represented in each pixel for transferring image that obtains by STC\_CID\_TRANSFER\_BITS\_PER\_PIXEL.

**pbyteYUVOrBGR**

Set the pointer of the buffer for the YUV or BGR image.

dwPixelFormat

Set the pixel format for the BGR converted image that is the one of the pixel format listed in the below chart.

Pixel format	Value	Descriptions
STCAM_PIXEL_FORMAT_08_MONO_OR_RAW	0x00000001	8bits for one pixel for monochrome
STCAM_PIXEL_FORMAT_24_BGR	0x00000004	24bits (in order B, G, R) for one pixel
STCAM_PIXEL_FORMAT_32_BGR	0x00000008	32bits (in order B, G, R, Dummy) for one pixel

pbyteBGR

Set the pointer that obtains the BGR image.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function converts from the YUV or BGR image to the BGR image.

The CPU usage is increasing while image converting. The RAW or BGR output image format is recommending output image format to reduce the CPU usage.

**StCam\_DecodingCombinedMultiROI**

Decode the image from the RAW image in the multiple ROIs configuration.

```
int StCam_DecodingCombinedMultiROI(
    void* hCamera,                // Library handle
    unsigned long dwDecodeMode,    // Decode mode
    unsigned char* pbyteRaw,       // Input RAW image
    unsigned char* *ppbyteDecodedRaw, // Output RAW image
    unsigned long* pdwWidth,       // Width
    unsigned long* pdwHeight,      // Height
    unsigned long* pdwLinePitch    // Number of byte for 1 line
);
```

[Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

dwDecodeMode

Set the decode mode for the RAW image that is one of the decode mode lited in the chart below.

Decode mode	Value	Descriptions
STCAM_DECODING_COMBINED_MULTI_ROI_FIRST_ROI	0x00000000	Obtains the address and the size of the first region 0x00000001: second region 0x00000002: third region.
STCAM_DECODING_COMBINED_MULTI_ROI_EXCEPT_BLANK_ROW_AND_COL	0x80000000	Obtains the address and the size of the image data that merges all regions. The image only includes the specified

region of the image.

pbyteRaw

Set the pointer of the buffer for the input RAW image

ppbyteDecodedRaw

Set the pointer that obtains decoded RAW image.

pdwWidth

Set the pointer that obtains the width of the image.

pdwHeight

Set the pointer that obtains the height of the image.

pdwLinePitch

Set the pointer that obtains the number of the byte for 1 line of the image

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function decodes the image from the RAW image in the multiple ROIs configuration.

The input image buffer offsets and returns (the data is valid while the input image buffer is valid) if the image data copy is not necessary.

A new image buffer is secured in the library and returns its address (the data is valid until this function is called again) if the image data copy is necessary.



### **StCam\_GetVBlankFromFPS**

This function obtains the vertical blanking to adjust the frame rate.

```
int StCam_GetVBlankFromFPS(  
    void* hCamera,          // Library handle  
    float fFPS,             // FPS (Frame rate)  
    unsigned long* pdwVBlank // Expand number of line for vertical blanking  
);
```

#### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**fFPS**

Set the FPS (frame rate).

**pdwVLines**

Set the pointer that obtains the expand number of the line for the vertical blanking.

#### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### [Descriptions]

This function obtains the vertical blanking to adjust the frame rate.

This function is only available for the specific cameras that is STC\_CID\_V\_BLANK valid camera.

### **StCam\_GetGainDBFromSettingValue**

This function converts from gain to gain (unit is dB).

```
int StCam_GetGainDBFromSettingValue(  
    void* hCamera,          // Library handle  
    unsigned short wGain,   // Gain  
    float *pfGaindB         // Gain (unit is dB)  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wGain**

Set the gain.

**pfGaindB**

Set the pointer that obtains the gain (unit is dB).

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function converts from gain to gain (unit is dB).

### **StCam\_GetExposureClockFromTime**

This function converts from the exposure time (unit is second) to the exposure time setting.

```
int StCam_GetExposureClockFromTime(  
    void* hCamera,                // Library handle  
    float fExpTime,               // Exposure time (unit is second)  
    unsigned long* pdwExposureClock // Exposure time setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**fExpTime**

Set the exposeure time (unit is second).

**pdwExposureClock**

Set the pointer that obtains the exposure time setting.

#### **[Return]**

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function converts from the exposure time (unit is second) to the exposure time setting.

The exposure time (unit is second) converts to the approximate exposure time setting.

### **StCam\_GetExposureTimeFromClock**

This function converts from the exposure time setting to the exposure time (unit is second).

```
int StCam_GetExposureTimeFromClock(  
    void* hCamera,                // Library handle  
    unsigned long dwExposureClock, // Expoure time setting  
    float *pfExpTime              // Expoure time (unit is second)  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwExposureClock**

Set the exposure time setting.

**pfExpTime**

Set the pointer that obtains the exposure time (unit is second).

#### **[Return]**

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function converts from the exposure time setting to the exposure time (unit is second).

The exposure time setting converts to the approximate exposure time (unit is second)

### **StCam\_GetDigitalGainSettingValueFromGainTimes**

This function converts from the digital gain magnification (x times magnifications) to the digital gain setting.

```
int StCam_GetDigitalGainSettingValueFromGainTimes(  
    void* hCamera,                // Library handle  
    float fDigitalGainTimes,      // Digital gain magnifications  
    unsigned short* pwDigitalGain // Digital gain setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**fDigitalGainTimes**

Set the digital gain (unit is x times magnifications).

**pwDigitalGain**

Set the pointer that obtains the digital gain setting.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function converts from the digital gain magnification (x times magnifications) to the digital gain setting. The maximum digital gain setting is returned if the digital gain magnifications is the greater than the maximum digital gain magnifications.

### **StCam\_GetDigitalGainTimesFromSettingValue**

This function converts from the digital gain setting to the digital gain magnification (x times magnification).

```
int StCam_GetDigitalGainTimesFromSettingValue(  
    void* hCamera,          // Library handle  
    unsigned short wDigitalGain, // Digital gain setting  
    float *pfDigitalGainTimes    // Digital gain magnifications  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wDigitalGain**

Set the digital gain setting.

**pfDigitalGainTimes**

Set the pointer that obtains the digital gain (unit is x times magnifications).

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function converts from the digital gain setting to the digital gain magnifications (x times magnifications). The maximum digital gain magnifications is returned if the digital gain setting is the greater than the maximum digital gain setting.

## StCam\_ALC

This function processes the ALC function on the PC.

```
int StCam_ALC(  
    void* hCamera,                // Library handle  
    unsigned short wCurrentBrightness, // Average brightness of the image  
    unsigned long* pdwALCStatus      // ALC status  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wCurrentBrightness**

Set the average brightness of the image from 0 to 255.

**pdwALCStatus**

Set the pointer that obtains the ALC status.

Bit0 changes to 1 when the exposure time is maintained.

Bit1 changes to 1 when the gain is maintained.

### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

### [Descriptions]

This function processes the ALC function on the PC.

Please use ALC function on the camera if STC\_CID\_ALC\_MODE is enabled camera.

The exposure time or gain is maintained the brightness to match the target brightness from the current brightness with this function.

Please obtain the average brightness with StCam\_GetAveragePixelValue.

It is necessary to use StCam\_GetAveragePixelValue multiple times when using weighted ALC function. It is necessary to set the average brightness after weight.

The ALC process becomes steady when using the average brightness of the few frames if the average brightness of the object is not steady.

It will take a few frames to reflect the maintained the exposure time or the gain for the free run mode. It will take another a few frames to reflect maintained the exposure time or the gain depending on the USB bus speed or the process speed of the PC.

Please use this function with enough interval time to avoid the image hunting.



### StCam\_GetAveragePixelValue

This function obtains the average brightness for the ALC function on the PC

```
int StCam_GetAveragePixelValue(  
    void* hCamera,                // Library handle  
    unsigned long dwImageWidth,    // Width  
    unsigned long dwImageHeight,   // Height  
    unsigned long dwImageLinePitch, // Number of byte for 1 line  
    unsigned short wColorArray,    // Color array  
    unsigned long dwTransferBitsPerPixel, // Number of bits represented in  
    each pixel for transferring image  
    unsigned char* pbyteRaw,       // RAW image buffer  
    unsigned long dwROIOffsetX,    // Horizontal offset for ROI  
    unsigned long dwROIOffsetY,    // Vertical offset for ROI  
    unsigned long dwROIWidth,      // Width for ROI  
    unsigned long dwROIHeight,     // Height for ROI  
    float *pfAverage              // Average brightness  
);
```

#### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwImageWidth**

Set the width of the RAW image.

**dwImageHeight**

Set the height of the RAW image.

**dwLinePitch**

Set the number of the byte for 1 line of the image

**wColorArray**

Only copy the data based on dwPixelFormat when setting  
"STCAM\_COLOR\_ARRAY\_MONO" at wColorArray.

**dwTransferBitsPerPixel**

Set the number of the bits represented in each pixel for transferring image that obtains by STC\_CID\_TRANSFER\_BITS\_PER\_PIXEL.

**pbyteRaw**

Set the pointer of the buffer for the RAW image.

**dwROIOffsetX**

Set the horizontal offset of the average brightness calculation image area.

**dwROIOffsetY**

Set the vertical offset of the average brightness calculation image area.

**dwROIWidth**

Set the width of the average brightness calculation image area.

**dwROIHeight**

Set the height offset of the average brightness calculation image area.

**pfAverage**

Set the pointer that is the float type array for the average brightness.

[0]: Monochrome average or R average, [1]: Gr average, [2]: Gb average, [3]: B average

**[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

**[Descriptions]**

This function obtains the average brightness for the ALC function on the PC

Please use ALC function on the camera if STC\_CID\_ALC\_MODE is enabled camera.

It is necessary to use StCam\_GetAveragePixelValue multiple times when using weighted ALC function. It is necessary to set the average brightness after weight.

## StCam\_SetALCMode

This function sets the ALC mode on the PC.

```
int StCam_SetALCMode(  
    void* hCamera,          // Library handle  
    unsigned char byteALCMode // ALC mode  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteALCMode**

Set the ALC mode at the PC that is one of the ALC mode listed in the chart below.

ALC mode	Value	Descriptions
STCAM_ALCMODE_OFF	0	Disables the auto shutter and the AGC
STCAM_ALCMODE_PC_AE_AGC_ON	1	Enables the auto shutter and the AGC on the PC
STCAM_ALCMODE_PC_AE_ON	2	Enables the auto shutter on the PC
STCAM_ALCMODE_PC_AGC_ON	3	Enables the AGC on the PC
STCAM_ALCMODE_PC_AE_AGC_ONESHOT	4	Enables the one shot auto shutter and the AGC on the PC  Disables the one shot auto shutter and the AGC on the PC when define the setting
STCAM_ALCMODE_PC_AE_ONESHOT	5	Enables the one shot auto shutter on the PC  Disables the one shot auto shutter on the PC when define the setting
STCAM_ALCMODE_PC_AGC_ONESHOT	6	Enables the one shot AGC on the PC  Disables the one shot AGC on the PC when define the setting

### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function sets the ALC mode on the camera.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

**StCam\_GetALCMode**

This function obtains the ALC mode at the PC.

```
int StCam_GetALCMode(  
    void* hCamera,           // Library handle  
    unsigned char* pbyteALCMode // ALC mode  
);
```

[Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

pbyteALCMode

Obtain the ALC mode at the PC that is one of the ALC mode listed in the chart below.

ALC mode	Value	Descriptions
STCAM_ALCMODE_OFF	0	Disables the auto shutter and the AGC
STCAM_ALCMODE_PC_AE_AGC_ON	1	Enables the auto shutter and the AGC on the PC
STCAM_ALCMODE_PC_AE_ON	2	Enables the auto shutter on the PC
STCAM_ALCMODE_PC_AGC_ON	3	Enables the AGC on the PC
STCAM_ALCMODE_PC_AE_AGC_ONESHOT	4	Enables the one shot auto shutter and the AGC on the PC  Disables the one shot auto shutter and the AGC on the PC when define the setting
STCAM_ALCMODE_PC_AE_ONESHOT	5	Enables the one shot auto shutter on the PC  Disables the one shot auto shutter on the PC when define the setting
STCAM_ALCMODE_PC_AGC_ONESHOT	6	Enables the one shot AGC on the PC  Disables the one shot AGC on the PC when define the setting

[Return]

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function obtains the ALC mode at the PC.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_SetALCTargetLevel**

This function obtains the target level for the ALC function.

```
int StCam_SetALCTargetLevel(  
    void* hCamera,          // Library handle  
    unsigned short wLevel   // ALC target level  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wLevel**

Set the target level for the ALC function.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the target level for the ALC function.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_GetALCTargetLevel**

This function sets the target level for the ALC function.

```
int StCam_GetALCTargetLevel(  
    void* hCamera,          // Library handle  
    unsigned short* pwLevel // ALC target level  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwLevel**

Set the pointer that obtains the target level for the ALC function.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the target level for the ALC function.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.



### **StCam\_GetAGCMinGain**

This function obtains the minimum gain for the AGC

```
int StCam_GetAGCMinGain(  
    void* hCamera,          // Library handle  
    unsigned short* pwValue // AGC minimum gain  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwMinGain**

Set the pointer that obtains the minimum gain for the AGC.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the minimum gain for the AGC.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_SetAGCMinGain**

This function sets the minimum gain for the AGC

```
int StCam_SetAGCMinGain(  
    void* hCamera,          // Library handle  
    unsigned short wValue    // AGC minimum gain  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wMinGain**

Set the minimum gain for the AGC.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the minimum gain for the AGC.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_GetAGCMaxGain**

This function obtains the maximum gain for the AGC.

```
int StCam_GetAGCMaxGain(  
    void* hCamera,          // Library handle  
    unsigned short* pwValue // AGC maximum gain  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwMaxGain**

Set the pointer that obtains the maximum gain for AGC.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the maximum gain for the AGC.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_SetAGCMaxGain**

This function sets the maximum gain for the AGC.

```
int StCam_SetAGCMaxGain(  
    void* hCamera,          // Library handle  
    unsigned short wValue    // AGC maximum gain  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wMaxGain**

Set the maximum gain for the AGC.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the maximum gain for the AGC.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_GetAEMinExposureClock**

This function obtains the minimum exposure time for the AE.

```
int StCam_GetAEMinExposureClock(  
    void* hCamera,                // Library handle  
    unsigned long* pdwMinExposureClock // AE minimum exposure time  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pdwMinExposureClock**

Set the pointer that obtains the minimum exposure time for AE.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the minimum exposure time for the AE.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_SetAEMinExposureClock**

This function sets the minimum exposure time for the AE.

```
int StCam_SetAEMinExposureClock(  
    void* hCamera,                // Library handle  
    unsigned long dwMinExposureClock // AE minimum exposure time  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwMinExposureClock**

Set the minimum exposure time for the AE.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the minimum exposure time for the AE.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_GetAEMaxExposureClock**

This function obtains the maximum exposure time for the AE.

```
int StCam_GetAEMaxExposureClock(  
    void* hCamera,                // Library handle  
    unsigned long* pdwMaxExposureClock // AE maximum exposure time  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pdwMaxExposureClock**

Set the pointer that obtains the minimum exposure time for the AE.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the maximum exposure time for the AE.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.

### **StCam\_SetAEMaxExposureClock**

This function sets the maximum exposure time for the AE.

```
int StCam_SetAEMaxExposureClock(  
    void* hCamera,                // Library handle  
    unsigned long dwMaxExposureClock // AE maximum exposure time  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwMaxExposureClock**

Set the maximum exposure time for the AE.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the maximum exposure time for the AE.

Please use ALC function on the camera for STC\_CID\_ALC\_MODE valid camera.



**StCam\_NoiseReduction**

This function processes the noise reduction of the image for the long exposure.

```
int StCam_NoiseReduction(
    void* hCamera,                // Library handle
    unsigned long dwReductionMode, // Noise reduction mode
    unsigned long dwWidth,        // Width
    unsigned long dwHeight,       // Height
    unsigned long dwLinePitch,    // Number of byte for 1 line
    unsigned short wColorArray,   // Color array
    unsigned char* pbyteRaw,      // RAW image buffer
    unsigned short wRawBitsPerPixel // Number of bits represented in each pixel
    for RAW image
);
```

[Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwNoiseReductionMode**

Set the noise reduction mode that is one of the noise reduction mode listed in the chart below.

Noise reduction mode	Value	Descriptions	
STCAM_NR_OFF	0x00000000	Noise reduction process with the current image	
STCAM_NR_EASY	0x00000001	Noise reduction process with the current image	
		Advantage	It is not necessary to calibrate with the shade image
		The random noises can be reduced	
		Disadvantage	The resolution of the image is reduced
STCAM_NR_COMPLEX	0x00000002	Does not reduce the noise in the high brightness and the adjoining noises	
		Noise reduction process with the shade image	
		It is necessary to calibrate the shade image with	

"STCAM\_NR\_DARK\_CL" mode before use the noise reduction process

Advantage The resolution of the image may not reduce

The noise the high brightness and the adjoining noises can be reduced

Disadvantage It is necessary to calibrate with the shade image

The random noises do not reduce

STCAM_NR_DARK_CL	0x80000000	Calibrates the shade image for the noise reduction process The shade image obtains with the same condition of the camera (the gain and the exposure time) for taking the image
------------------	------------	---

dwWidth

Set the width of the image.

dwHeight

Set the height of the image.

dwLinePitch

Set the number of the byte for 1 line.

wColorArray

Set the color array information for RAW image that obtained by STC\_CID\_COLOR\_ARRAY.

pbyteRaw

Set the pointer of the buffer for the RAW image.

This image is overwritten after the process.

wRawBitsPerPixel

Set the number of the bits represented in each pixel for the RAW image from 8, 10 or 12 bits.

#### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### [Descriptions]

This function processes the noise reduction of the image for the long exposure.

The stars pixels that are red, blue or green dots with the color camera (monochrome dots with the monochrome camera) are appeared on the image when using the long exposure function with the start and stop trigger. This is so call "dark current noise", do not the issue of the camera. This noise can be reducing with this function.

### **StCam\_SetShadingCorrectionTarget**

This function sets the target pixel level for the shading correction.

```
int StCam_SetShadingCorrectionTarget(  
    void* hCamera,          // Library handle  
    unsigned short wTarget  // Target pixel level for shading correction  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wTarget**

Set the target pixel level for the shading correction.

The average pixel level is using for this function when setting 0.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the target pixel level for the shading correction.

The shading correction value calculates automatically to match the set pixel level when the pixel level sets before calibration,

### **StCam\_GetShadingCorrectionTarget**

This function obtains the target pixel level for the shading correction.

```
int StCam_GetShadingCorrectionTarget(  
    void* hCamera,           // Library handle  
    unsigned short* pwTarget // Target pixel level for shading correction  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwTarget**

Set the pointer that obtains the target level for the shading correction.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the target pixel level for the shading correction.

## StCam\_SetShadingCorrectionMode

This function sets the shading mode for the shading correction.

```
int StCam_SetShadingCorrectionMode(  
    void* hCamera,          // Library handle  
    unsigned long dwMode    // Shading mode  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwMode**

Set the shading mode that is one of the shading mode listed in the chart below.

Shading mode	Value	Descriptions
STCAM_SHADING_CORRECTION_MODE_OFF	0x0000	Noise reduction process with the current image
STCAM_SHADING_CORRECTION_MODE_CALIBRATION_MULTIPLICATION	0x0001	Does not use shading correction.
STCAM_SHADING_CORRECTION_MODE_ON_MULTIPLICATION	0x0002	The calibration for shading correction by multiplication. Selects this with a uniform subject under the specified light source.
STCAM_SHADING_CORRECTION_MODE_CALIBRATION_ADDITION	0x0003	The calibration for shading correction by addition and subtraction. Selects this with a uniform subject under the specified light source.
STCAM_SHADING_CORRECTION_MODE_ON_ADDITION	0x0004	The shading correction by addition and subtraction. It is necessary to select "Calibration (Addition)" beforehand to process the

shading correction.

In the Multi-ROI mode, the shading correction is also applied to disabled areas.

#### [Return]

If the function was successful, return the 0.

If the function failed, return an `ErrorCode` of `Linux`.

#### [Descriptions]

This function sets the shading mode for the shading correction.

## StCam\_GetShadingCorrectionMode

This function obtains the shading mode for the shading correction.

```
int StCam_GetShadingCorrectionMode(  
    void* hCamera,          // Library handle  
    unsigned long* pdwMode  // Shading mode  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pdwMode**

Set the pointer that obtains the shading mode that is one of the shading mode listed in the chart below.

Shading mode	Value	Descriptions
STCAM_SHADING_CORRECTION_MODE_OFF	0x0000	Noise reduction process with the current image
STCAM_SHADING_CORRECTION_MODE_CALIBRATION_MULTIPLICATION	0x0001	Does not use shading correction.
STCAM_SHADING_CORRECTION_MODE_ON_MULTIPLICATION	0x0002	The calibration for shading correction by multiplication. Selects this with a uniform subject under the specified light source.
STCAM_SHADING_CORRECTION_MODE_CALIBRATION_ADDITION	0x0003	The calibration for shading correction by addition and subtraction. Selects this with a uniform subject under the specified light source.
STCAM_SHADING_CORRECTION_MODE_ON_ADDITION	0x0004	The shading correction by addition and subtraction. It is necessary to select "Calibration (Addition)"



beforehand to process the shading correction.

In the Multi-ROI mode, the shading correction is also applied to disabled areas.

#### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### [Descriptions]

This function obtains the shading mode for the shading correction.

## StCam\_ShadingCorrection

This function processes the shading correction.

```
int StCam_ShadingCorrection(  
    void* hCamera,                // Library handle  
    unsigned long dwWidth,         // Width  
    unsigned long dwHeight,        // Height  
    unsigned long dwLinePitch,     // Number of byte for 1 line  
    unsigned char* pbyteRaw,       // RAW image buffer  
    unsigned short wRawBitsPerPixel // Number of bits represented in each pixel  
    for RAW image  
);
```

### [Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

dwWidth

Set the width of the image.

dwHeight

Set the height of the image.

dwLinePitch

Set the number of the byte for 1 line.

pbyteRaw

Set the pointer of the buffer for the RAW image.

wRawBitsPerPixel

Set the number of the bits represented in each pixel for the RAW image from 8, 10 or 12 bits.

#### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### [Descriptions]

This function processes the shading correction.

Please follow below procedure for the shading correction.

Set "OFF" at StCam\_SetShadingCorrectionMode

Set the target level at StCam\_SetShadingCorrectionTarget

Set "Calibration" at StCam\_SetShadingCorrectionMode

Process calibration by StCam\_ShadingCorrection

Set "ON" at StCam\_SetShadingCorrectionMode

Shading correction process by StCam\_ShadingCorrection

## StCam\_GetWhiteBalanceMode

This function obtains the white balance mode.

```
int StCam_GetWhiteBalanceMode(  
    void* hCamera,                // Library handle  
    unsigned char* pbyteWBMode    // White balance mode  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pbyteWBMode**

Set the pointer that obtains the white balance mode that is one of the white balance mode listed in the chart below.

White balance mode	Value	Descriptions
STCAM_WB_OFF	0	No white balance process
STCAM_WB_MANUAL	1	Manual white balance process with the preset gain settings that sets by StTrg_SetWhiteBalanceGain and obtains by StTrg_GetWhiteBalanceGain
STCAM_WB_FULLAUTO	2	Auto white balance process
STCAM_WB_ONESHOT	3	Auto white balance process once to defaming the white balance then change to the manual white balance "STCAM_WB_MANUAL" automatically

### [Return]

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

### [Descriptions]

This function obtains the white balance mode.

Please use White balance function on the camera for STC\_CID\_WHITE\_BALANCE\_MODE valid camera.

**StCam\_SetWhiteBalanceMode**

This function sets the white balance mode.

```
int StCam_SetWhiteBalanceMode(  
    void* hCamera,          // Library handle  
    unsigned char byteWBMode // White balance mode  
);
```

[Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

byteWBMode

Set the white balance mode that is one of the white balance mode listed in the chart below.

White balance mode	Value	Descriptions
STCAM_WB_OFF	0	No white balance process
STCAM_WB_MANUAL	1	Manual white balance process with the preset gain settings that sets by StTrg_SetWhiteBalanceGain and obtains by StTrg_GetWhiteBalanceGain
STCAM_WB_FULLAUTO	2	Auto white balance process
STCAM_WB_ONESHOT	3	Auto white balance process once to defaming the white balance then change to the manual white balance "STCAM_WB_MANUAL" automatically

[Return]

If the function was scusessful, return the 0.  
If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function sets the white balance mode.  
Please use White balance function on the camera for STC\_CID\_WHITE\_BALANCE\_MODE valid camera.

### **StCam\_GetWhiteBalanceGain**

This function obtains the gain for the white balance.

```
int StCam_GetWhiteBalanceGain(  
    void* hCamera,                // Library handle  
    unsigned short* pwWBGainR,    // R gain for white balance  
    unsigned short* pwWBGainGr,   // Gr gain for white balance  
    unsigned short* pwWBGainGb,   // Gb gain for white balance  
    unsigned short* pwWBGainB     // B gain for white balance  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwWBGainR**

Set the pointer that obtains the R gain for the white balance.

**pwWBGainGr**

Set the pointer that obtains the Gr gain for the white balance.

**pwWBGainGb**

Set the pointer that obtains the Gb gain for the white balance.

**pwWBGainB**

Set the pointer that obtains the B gain for the white balance.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the gain for the white balance.

Please use White balance function on the camera for  
STC\_CID\_WHITE\_BALANCE\_MODE valid camera.

## StCam\_SetWhiteBalanceGain

This function sets the gain for the white balance.

```
int StCam_SetWhiteBalanceGain(  
    void* hCamera,          // Library handle  
    unsigned short wWBGainR, // R gain for white balance  
    unsigned short wWBGainGr, // Gr gain for white balance  
    unsigned short wWBGainGb, // Gb gain for white balance  
    unsigned short wWBGainB   // B gain for white balance  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wWBGainR**

Set the R gain for the white balance.

The multiplier for R is "sets value" / 128 (The gain is x1 time when setting 128).

Setting range is 128 to "obtained by StCam\_GetWhiteBalanceMaxGain".

**wWBGainGr**

Set the Gr gain for the white balance.

The multiplier for Gr is "sets value" / 128 (The gain is x1 time when setting 128).

Setting range is 128 to "obtained by StCam\_GetWhiteBalanceMaxGain".

**wWBGainGb**

Set the Gb gain for the white balance.

The multiplier for Gb is "sets value" / 128 (The gain is x1 time when setting 128).

Setting range is 128 to "obtained by StCam\_GetWhiteBalanceMaxGain".

**wWBGainB**

Set the B gain for the white balance.

The multiplier for B is "sets value" / 128 (The gain is x1 time when setting 128).

Setting range is 128 to "obtained by StCam\_GetWhiteBalanceMaxGain".

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function sets the gain for the white balance.

Please use White balance function on the camera for  
STC\_CID\_WHITE\_BALANCE\_MODE valid camera.



### **StCam\_GetWhiteBalanceMaxGain**

This function obtains the maximum gain for the white balance.

```
int StCam_GetWhiteBalanceMaxGain(  
    void* hCamera,                // Library handle  
    unsigned short* pwWBGainR,    // Maximum R gain for white balance  
    unsigned short* pwWBGainGr,   // Maximum Gr gain for white balance  
    unsigned short* pwWBGainGb,   // Maximum Gb gain for white balance  
    unsigned short* pwWBGainB     // Maximum B gain for white balance  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwWBGainR**

Set the pointer that obtains the maximum R gain for the white balance.

**pwWBGainGr**

Set the pointer that obtains the maximum Gr gain for the white balance.

**pwWBGainGb**

Set the pointer that obtains the maximum Gb gain for the white balance.

**pwWBGainB**

Set the pointer that obtains the maximum B gain for the white balance.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the maximum gain for the white balance.

Please use White balance function on the camera for

STC\_CID\_WHITE\_BALANCE\_MODE valid camera.

## StCam\_WhiteBalance

This function processes the white balance to the RAW image.

```
int StCam_WhiteBalance(  
    void* hCamera,           // Library handle  
    unsigned long dwWidth,    // Width  
    unsigned long dwHeight,   // Height  
    unsigned long dwLinePitch, // Number of byte for 1 line  
    unsigned short wColorArray, // Color array  
    unsigned char* pbyteRaw,   // RAW image buffer  
    unsigned short wRawBitsPerPixel // Number of bits represented in each pixel  
    for RAW image  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**dwLinePitch**

Set the number of the byte for 1 line.

**wColorArray**

Set the color array information for the RAW image that obtained by STC\_CID\_COLOR\_ARRAY.

**pbyteRaw**

Set the pointer of the buffer for the RAW image.

The image is overwritten by the white balance processed image.

wRawBitsPerPixel

Set the number of the bits represented in each pixel for the RAW image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the white balance to the raw image.

Please use White balance function on the camera for  
STC\_CID\_WHITE\_BALANCE\_MODE valid camera.

The white balance process is not processed for the monochrome model and set “Off” at the white balance mode for the color model.

Please set white balance mode with StCam\_SetWhiteBalanceMode and gain with StCam\_SetWhiteBalanceGain before use this function.

## StCam\_GetGammaMode

This function obtains the gamma mode.

```
int StCam_GetGammaMode(  
    void* hCamera,           // Library handle  
    unsigned char byteGammaTarget, // Gamma target  
    unsigned char* pbyteGammaMode, // Gamma mode  
    unsigned short* pwGamma,      // Gamma  
    short* pshtBrightness,       // Brightness offset  
    unsigned short* pwContrast,   // Contrast  
    unsigned short* pwGammaTable, // Gamma table  
    unsigned short* pwBitsPerEachColor // Number of bits represented in each color  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteGammaTarget**

Set the gamma target that is one of the gamma target listed in the chart below.

Gamma target	Value	Descriptions
STCAM_GAMMA_TARGET_Y	0	Gamma adjusts for the luminance
STCAM_GAMMA_TARGET_R	1	Gamma adjusts for R
STCAM_GAMMA_TARGET_GR	2	Gamma adjusts for Gr
STCAM_GAMMA_TARGET_GB	3	Gamma adjusts for Gb
STCAM_GAMMA_TARGET_B	4	Gamma adjusts for B

pbyteGammaMode

Set the pointer that obtains the gamma mode that is one of the gamma mode listed in the chart below.

Gamma mode	Value	Descriptions
STCAM_GAMMA_OFF	0	No gamma adjustment
STCAM_GAMMA_ON	1	Gamma adjusts with the gamma setting
STCAM_GAMMA_REVERSE	2	Gamma adjusts with the gamma setting (Negative)
STCAM_GAMMA_TABLE	255	Gamma adjusts with the gamma table

pwGamma

Set pointer that obtains the gamma setting from 1 to 500 that is specified as "actual gamma value multiplied by 100" (default is 100).

The invalid gamma setting returns when selecting "STCAM\_GAMM\_TABLE" for the gamma mode.

pshtBrightness

Set the pointer that obtains the contrast setting from -255 to 255..

pbyteContrast

Set pointer that obtains the contrast setting from 0 to 127.

pbyteGammaTable

Set the pointer that obtains the gamma table (BYTE pbyteGammaTable[256]) for the gamma table mode.

pwBitsPerEachColor

Set the pointer that obtains the number of the bits represented in each color (B, G and R) of the BGR image from 8, 10 or 12 bits.

[Return]

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function obtains the gamma mode.

**StCam\_SetGammaMode**

This function sets the gamma mode.

```
int StCam_SetGammaMode(  
    void* hCamera,           // Library handle  
    unsigned char byteGammaTarget, // Gama target  
    unsigned char byteGammaMode,  // Gamma mode  
    unsigned short wGamma,        // Gamma  
    short shtBrightness,         // Offset  
    unsigned short wContrast,     // Contrast  
    unsigned short* pwGammaTable, // Gamma table  
    unsigned short wBitsPerEachColor // Number of bits represented in each color  
);
```

[Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteGammaTarget**

Set the gamma target that is one of the target gamma listed in the chart below.

Gamma target	Value	Descriptions
STCAM_GAMMA_TARGET_Y	0	Gamma adjusts for the luminance
STCAM_GAMMA_TARGET_R	1	Gamma adjusts for R
STCAM_GAMMA_TARGET_GR	2	Gamma adjusts for Gr
STCAM_GAMMA_TARGET_GB	3	Gamma adjusts for Gb
STCAM_GAMMA_TARGET_B	4	Gamma adjusts for B

### byteGammaMode

Set the gamma mode that is one of the gamma mode listed in the chart below.

Gamma mode	Value	Descriptions
STCAM_GAMMA_OFF	0	No gamma adjustment
STCAM_GAMMA_ON	1	Gamma adjusts with the gamma setting
STCAM_GAMMA_REVERSE	2	Gamma adjusts with the gamma setting (Negative)
STCAM_GAMMA_TABLE	255	Gamma adjusts with the gamma table

### wGamma

Set the gamma setting from 1 to 500 that is specified as "actual gamma value multiplied by 100" (default is 100).

This gamma setting is only valid when selecting "STCAM\_GAMM\_ON" or "STCAM\_GAMMA\_REVERSE" for the gamma mode.

### shtBrightness

Set the offset setting from -255 to 255.

This offset setting is only valid when selecting "STCAM\_GAMM\_ON" or "STCAM\_GAMMA\_REVERSE" for the gamma mode.

### byteContrast

Set the contrast setting from 0 to 127.

This contast setting is only valid when selecting "STCAM\_GAMM\_ON" or "STCAM\_GAMMA\_REVERSE" for the gamma mode.

### pbyteGammaTable

Set the pointer of the gamma table (BYTE pbyteGammaTable[256]) for the gamma table mode.

This Gamma table setting is only valid when selecting STCAM\_GAMMA\_TABLE for the gamma mode.

NULL returns when selecting other than STCAM\_GAMMA\_TABLE for the gamma mode.



pwBitsPerEachColor

Set the pointer that obtains the number of the bits represented in each color (B, G and R) of the BGR image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function sets the gamma mode.

## StCam\_RawColorGamma

This function processes the gamma adjustment for the RAW image.

```
int StCam_RawColorGamma(  
    void* hCamera,           // Library handle  
    unsigned long dwWidth,    // Width  
    unsigned long dwHeight,   // Height  
    unsigned short wColorArray, // Color Array  
    unsigned char* pbyteRaw,   // RAW image buffer  
    unsigned short wRawBitsPerPixel // Number of bits represented in each pixel  
of the RAW image  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**wColorArray**

Set the color array information for the RAW image that obtained by STC\_CID\_COLOR\_ARRAY.

**pbyteRaw**

Set the pointer of the buffer for the RAW image.

The image is overwritten after process.

**wRawBitsPerPixel**

Set the number of the bits represented in each pixel of the RAW image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the gamma adjustment for the RAW image.

It is necessary to set the gamma table with StCam\_SetGammaMode.

Please use StCam\_BGRGamma for the monochrome model.

## StCam\_BRRGamma

This function processes the gamma adjustment for the BGR image.

```
int StCam_BRRGamma(  
    void* hCamera,           // Library handle  
    unsigned long dwWidth,   // Width  
    unsigned long dwHeight,  // Height  
    unsigned long dwPixelFormat, // Pixel format  
    unsigned char* pbyteBGR, // BGR (or monochrome) image buffer  
    unsigned short wBitsPerEachColor // Number of bits represented in each color  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**dwPixelFormat**

Set the pixel format of the image that is one of the pixel format listed in the chart below.

Pixel format	Value	Descriptions
STCAM_PIXEL_FORMAT_08_MONO_OR_RAW	0x00000001	8bits for 1 pixel for monochrome
STCAM_PIXEL_FORMAT_24_BGR	0x00000004	24bits (in order B, G, R) for 1pixel
STCAM_PIXEL_FORMAT_32_BGR	0x00000008	32bits (in order B, G, R, Dummy) for 1pixel

**pbyteBGR**

Set the pointer of the buffer for the gamma processed BGR (or monochrome) image.

pwBitsPerEachColor

Set the pointer that obtains the number of the bits represented in each color (B, G and R) of the BGR image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the gamma adjustment for the BGR image.

It is necessary to set the gamma table with StCam\_SetGammaMode.

For the color model, the CPU usage increases because convert from RGB to Y before the gamma process then convert from Y to RGB.

Please use StCam\_RawColorGamma to reduce the CPU usage if the image quality is good enough.

## Stcam\_ColorInterpolation

This function processes the color interpolation

```
int StCam_ColorInterpolation(  
    void* hCamera,                // Library handle  
    unsigned long dwWidth,        // Width  
    unsigned long dwHeight,      // Height  
    unsigned short wColorArray,   // Color array  
    unsigned char* pbyteRaw,      // RAW image buffer  
    unsigned char* pbyteBGR,      // BGR (or monochrome) image  
    buffer  
    unsigned char byteColorInterpolationMethod, // Color interpolation method  
    unsigned long dwPixelFormat,    // Pixel formwat  
    unsigned short wRawBitsPerPixel // Number of bits represented in  
    each pixel for RAW image  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**wColorArray**

Set the color array information for the RAW image that obtained by  
STC\_CID\_COLOR\_ARRAY.

**pbyteRaw**

Set the pointer of the buffer for the RAW image before the color interpolation.

pbyteBGR

Set the pointer of the BGR (or monochrome) image after the color interpolation.

byteColorInterpolationMethod

Set the color interpolation method that is one of the color interpolation method listed in the chart below.

This is invalid when selecting STCAM\_COLOR\_ARRAY\_MONO at wColorArray.

Color interpolation method	Value	Descriptions
STCAM_COLOR_INTERPOLATION_NONE_MONO	0	Monochrome image without the color interpolation process
STCAM_COLOR_INTERPOLATION_NONE_COLOR	1	Color image without the color interpolation process
STCAM_COLOR_INTERPOLATION_NEAREST_NEIGHBOR	2	Color image with "Nearest neighbor color interpolation" process Copy the nearest pixel's color information
STCAM_COLOR_INTERPOLATION_BILINEAR	3	Color image with "Bilinear color interpolation" process Color interpolation with 4 surrounding pixel's color information
STCAM_COLOR_INTERPOLATION_BICUBIC	4	Color image with "Bicubic color interpolation" process Color interpolation with 16 surrounding pixel's color information
STCAM_COLOR_INTERPOLATION_BILINEAR_FALSE_COLOR_REDUCTION	5	Color image with "Bilinear color interpolation (reduce false color)" process Color interpolation with 4 surrounding pixel's color information Less false color at the edge, but

		this requires the more CPU usage
STCAM_COLOR_INTERPOLATION_NEAREST_NEIGHBOR2	6	Color image with "Nearest neighbor color interpolation" process Copy the nearest pixel's color information

### dwPixelFormat

Set the pixel format of the image that is after color interpolation that is one of the pixel format listed in the chart below.

Pixel format	Value	Descriptions
STCAM_PIXEL_FORMAT_08_MONO_OR_RAW	0x00000001	8bits for 1 pixel for monochrome
STCAM_PIXEL_FORMAT_24_BGR	0x00000004	24bits (in order B, G, R) for 1pixel
STCAM_PIXEL_FORMAT_32_BGR	0x00000008	32bits (in order B, G, R, Dummy) for 1pixel

### wRawBitsPerPixel

Set the number of the bits represented in each pixel for the RAW image from 8, 10 or 12 bits.

#### [Return]

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### [Descriptions]

This function processes the color interpolation

Each pixel of the RAW image has only R, G or B information. It is necessary to do the color interpolation process to make RGB information for each pixel.

Only copy the data based on dwPixelFormat when sets

"STCAM\_COLOR\_ARRAY\_MONO" at wColorArray



**StCam\_MirrorRotation**

This function processes the mirror image and the rotation of the image.

```
int StCam_MirrorRotation(  
    void* hCamera,           // Library handle  
    unsigned char byteMirrorMode, // Mirror mode  
    unsigned char byteRotationMode, // Rotation mode  
    unsigned long* pdwWidth,    // Width  
    unsigned long* pdwHeight,   // Height  
    unsigned short* pwColorArray, // Color array  
    unsigned char* pbyteRaw     // RAW image buffer  
);
```

[Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteMirrorMode**

Set the mirror mode that is one of the mirror mode listed in the chart below.

Mirror mode	Value	Descriptions
STCAM_MIRROR_OFF	0	Normal image
STCAM_MIRROR_HORIZONTAL	1	Horizontally flipped image
STCAM_MIRROR_VERTICAL	2	Vertical flipped image
STCAM_MIRROR_HORIZONTAL_VERTICAL	3	Horizontally and vertically flipped image (180 degrees rotated image from the normal image)

### byteRotationMode

Set the rotation mode that is one of the rotation mode listed in the chart below.

Rotation mode	Value	Descriptions
STCAM_ROTATION_OFF	0	Normal image
STCAM_ROTATION_CLOCKWISE_90	1	90 degrees clockwise rotated image from the normal image
STCAM_ROTATION_COUNTERCLOCKWISE_90	2	90 degrees counter clockwise rotated image from the normal image

### pdwWidth

Set the width of the image.

The width is overwritten with new width of the image when rotating the image.

### pdwHeight

Set the height of the image.

The height is overwritten with new height of the image when rotating the image.

### wColorArray

Only copy the data based on dwPixelFormat when sets "STCAM\_COLOR\_ARRAY\_MONO" at wColorArray.

### pbyteRaw

Set the pointer of the buffer for the RAW image.

The image is overwritten after process.

### [Return]

If the function was scusessful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the mirror image and the rotation of the image.

The image size and color array information will be change after the mirror / rotation process.

Please use mirror function on the camera for V4L2\_CID\_HFLIP/V4L2\_CID\_VFLIP valid camera.

### StCam\_GetHueSaturationMode

This function obtains the hue and saturation mode.

```
int StCam_GetHueSaturationMode(  
    void* hCamera,                // Library handle  
    unsigned char* pbyteHueSaturationMode, // Hue / saturation mode  
    short* pshtHue,                // Hue  
    unsigned short* pwSaturation    // Saturation  
);
```

[Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pbyteHueSaturationMode**

Set the pointer that obtains the hue and saturation mode that is one of the hue and saturation mode listed in the chart below.

Hue and saturation mode	Value	Descriptions
STCAM_HUE_SATURATION_OFF	0	No hue and saturation adjustment
STCAM_HUE_SATURATION_ON	1	Hue and saturation adjusts with the hue gain and the saturation gain

**pshtHue**

Set the pointer that obtains the hue.

**pwSaturation**

Set the pointer that obtains the saturation.

[Return]

If the function was scuessful, return the 0.  
If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function obtains the hue and saturation mode.

**StCam\_SetHueSaturationMode**

This function sets the hue and saturation mode.

```
int StCam_SetHueSaturationMode(
    void* hCamera,                // Library handle
    unsigned char byteHueSaturationMode, // Hue / saturation mode
    short shtHue,                 // Hue
    unsigned short wSaturation    // Saturation
);
```

[Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

byteHueSaturationMode

Set the hue and saturation mode that is one of hue and saturation mode listed in the chart below.

Hue and saturation mode	Value	Descriptions
STCAM_HUE_SATURATION_OFF	0	No hue and saturation adjustment
STCAM_HUE_SATURATION_ON	1	Hue and saturation adjusts with the hue gain and the saturation gain

shtHue

Set the hue that is amount in “number of the degree x10”.  
The hue range is -1800 to 1800 and set “0” if there is no change for the hue phase.

wSaturation

Set the saturation from 0 to 200.  
Set “0” if there is no change for the color saturation.

[Return]

If the function was scusessful, return the 0.  
If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function sets the hue and saturation mode.

**StCam\_GetColorMatrix**

This function obtains the color matrix.

```
int StCam_GetColorMatrix(  
    void* hCamera,                // Library handle  
    unsigned char* pbyteColorMatrixMode, // Color matrix mode  
    short* pshtColorMatrix        // Color matrix  
);
```

[Parameters]

hCamera

Set the library handle that obtained by the StCam\_Open function.

pbyteColorMatrixMode

Set the pointer that obtains the color matrix mode that is one of the color matrix mode listed in the chart below.

Color matrix mode	Value	Descriptions
STCAM_COLOR_MATRIX_OFF	0x00	No color matrix adjustment
STCAM_COLOR_MATRIX_CUSTOM	0xFF	Color matrix adjusts with customer's table

pshtColorMatrix

Set the pointer that obtains the color matrix (SHORT pshtColorMatrix[12]).

[Return]

If the function was scusessful, return the 0.  
If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function obtains the color matrix.

## StCam\_SetColorMatrix

This function sets the color matrix.

```
int StCam_SetColorMatrix(  
    void* hCamera,                // Library handle  
    unsigned char byteColorMatrixMode, // Color matrix mode  
    short*      pshtColorMatrix    // Color matrix  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteColorMatrixMode**

Set the color matrix mode that is one of the color matrix mode listed in the chart below.

Color matrix mode	Value	Descriptions
STCAM_COLOR_MATRIX_OFF	0x00	No color matrix adjustment
STCAM_COLOR_MATRIX_CUSTOM	0xFF	Color matrix adjusts with customer's table

**pshtColorMatrix**

Set the pointer that obtains the color matrix (SHORT pshtColorMatrix[12]).

### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

### [Descriptions]

This function obtains the color matrix.

The color matrix formulas are as follows:

$$R' = (R_{xm}[0] + G_{xm}[1] + B_{xm}[2] + m[3]) / 100$$

$$G' = (R_{xm}[4] + G_{xm}[5] + B_{xm}[6] + m[7]) / 100$$

$$B' = (R_{xm}[8] + G_{xm}[9] + B_{xm}[10] + m[11]) / 100$$



### **StCam\_SetHighChromaSuppression**

This function sets the high brightness chroma suppression settings

```
int StCam_SetHighChromaSuppression(  
    void* hCamera,           // Library handle  
    unsigned short wStartLevel, // Chroma suppression control threshold  
    unsigned short wSuppression // Chroma suppression setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wStartLevel**

Set the chroma suppression control threshold brightness level from 0 (dark) to 255 (bright).

The chroma suppression target pixel is the higher brightness level of the pixel than this threshold brightness level.

**wSuppression**

Set the chroma suppression setting from 0 (minimum) to 255 (maximum).

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the high brightness chroma suppression settings

### **StCam\_SetLowChromaSuppression**

This function sets the low brightness chroma suppression settings

```
int StCam_SetLowChromaSuppression(  
    void* hCamera,           // Library handle  
    unsigned short wStartLevel, // Chroma suppression control threshold  
    unsigned short wSuppression // Chroma suppression setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**wStartLevel**

Set the chroma suppression control threshold brightness level from 0 (dark) to 255 (bright).

The chroma suppression target pixel is the lower brightness level of the pixel than this threshold brightness level.

**wSuppression**

Set the chroma suppression setting from 0 (minimum) to 255 (maximum).

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function sets the low brightness chroma suppression settings

### **StCam\_GetHighChromaSuppression**

This function obtains the high brightness chroma suppression settings

```
int StCam_GetHighChromaSuppression(  
    void* hCamera,                // Library handle  
    unsigned short* pwStartLevel,  // Chroma suppression control threshold  
    unsigned short* pwSuppression // Chroma suppression setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwStartLevel**

Set the pointer that obtains the chroma suppression control threshold brightness level for the high brightness chroma suppression.

**pwSuppression**

Set the pointer that obtains the chroma suppression setting.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the high brightness chroma suppression settings

### **StCam\_GetLowChromaSuppression**

This function obtains the low brightness chroma suppression settings

```
int StCam_GetLowChromaSuppression(  
    void* hCamera,                // Library handle  
    unsigned short* pwStartLevel,  // Chroma suppression control threshold  
    unsigned short* pwSuppression // Chroma suppression setting  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pwStartLevel**

Set the pointer that obtains the chroma suppression control threshold brightness level for the low brightness chroma suppression.

**pwSuppression**

Set the pointer that obtains the chroma suppression setting.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

#### **[Descriptions]**

This function obtains the low brightness chroma suppression settings

## StCam\_HueSaturationColorMatrix

This function processes the hue, the saturation, chroma suppress and the color matrix of the image

```
int StCam_HueSaturationColorMatrix(  
    void* hCamera,                // Library handle  
    unsigned long dwWidth,         // Width  
    unsigned long dwHeight,       // Height  
    unsigned long dwPixelFormat,   // Pixel format  
    unsigned char* pbyteBGR,      // BGR image buffer  
    unsigned short wBitsPerEachColor // Number of bits represented in each color  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**dwPixelFormat**

Set the pixel format that is one of the pixel format listed in the chart below.

Pixel format	Value	Descriptions
STCAM_PIXEL_FORMAT_08_MONO_OR_RAW	0x00000001	8bits for 1 pixel for monochrome
STCAM_PIXEL_FORMAT_24_BGR	0x00000004	24bits (in order B, G, R) for 1pixel
STCAM_PIXEL_FORMAT_32_BGR	0x00000008	32bits (in order B, G, R, Dummy) for 1pixel

**pbyteBGR**

Set the pointer of the buffer for the BGR image.

The image is overwritten afater process.

wBitsPerEachColor

Set the pointer that obtains the number of the bits represented in each color (B, G and R) of the BGR image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the hue, the saturation, chroma suppress and the color matrix of the image

Please set the hue and saturation setting with StCam\_SetHueSaturationMode and color matrix setting with StCam\_SetColorMatrix before use this function.

## StCam\_GetSharpnessMode

This function obtains the edge enhancement (sharpness) mode of the image.

```
int StCam_GetSharpnessMode(  
    void* hCamera,                // Library handle  
    unsigned char* pbyteSharpnessMode, // Sharpness mode  
    unsigned short* pwSharpnessGain,   // Sharpness gain  
    unsigned char* pbyteSharpnessCoring // Sharpness coring  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**pbyteSharpnessMode**

Set the pointer that obtains the sharpness mode that is one of the sharpness mode listed in the chart below.

Sharpness mode	Value	Descriptions
STCAM_SHARPNESS_OFF	0	No edge enhancement (sharpness) adjustment
STCAM_SHARPNESS_ON	1	Edge enhancement (sharpness) adjusts with the sharpness gain and the coring

**pwSharpnessGain**

Set the pointer that obtains the sharpness gain.

**pbyteSharpnessCoring**

Set the pointer that obtains the sharpness coring.

### [Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

### [Descriptions]

This function obtains the edge enhancement (sharpness) mode of the image.

## StCam\_SetSharpnessMode

This function sets the edge enhancement (sharpness) mode of the image.

```
int StCam_SetSharpnessMode(  
    void* hCamera,                // Library handle  
    unsigned char byteSharpnessMode, // Sharpness mode  
    unsigned short wSharpnessGain,   // Sharpness gain  
    unsigned char byteSharpnessCoring / Sharpness coring  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**byteSharpnessMode**

Set the sharpness mode that is one of the sharpness mode listed in the chart below.

Sharpness mode	Value	Descriptions
STCAM_SHARPNESS_OFF	0	No edge enhancement (sharpness) adjustment
STCAM_SHARPNESS_ON	1	Edge enhancement (sharpness) adjusts with the sharpness gain and the coring

**wSharpnessGain**

Set the sharpness gain of the image from 0 to 500 (default is 0).

The edge enhancement becomes stronger when increasing this.

**byteSharpnessCoring**

Set the sharpness coring of the image from 0 to 255 (default is 0).

The small edge (including the noise) enhancement becomes suppress when increasing this.

### [Return]

If the function was scuessful, return the 0.

If the function failed, return an ErrorCode of Linux.



[Descriptions]

This function sets the edge enhancement (sharpness) mode of the image.

## StCam\_Sharpness

This function processes the edge enhancement (sharpness) of the image

```
int StCam_Sharpness(  
    void* hCamera,           // Library handle  
    unsigned long dwWidth,   // Width  
    unsigned long dwHeight,  // Height  
    unsigned long dwPixelFormat, // Pixel format  
    unsigned char* pbyteBGR, // BGR (or monochrome) image buffer  
    unsigned short wBitsPerEachColor // Number of bits represented in each color  
);
```

### [Parameters]

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**dwPixelFormat**

Set the pixel format that is one of the pixel format listed in the chart below.

Pixel format	Value	Descriptions
STCAM_PIXEL_FORMAT_08_MONO_OR_RAW	0x00000001	8bits for 1 pixel for monochrome
STCAM_PIXEL_FORMAT_24_BGR	0x00000004	24bits (in order B, G, R) for 1pixel
STCAM_PIXEL_FORMAT_32_BGR	0x00000008	32bits (in order B, G, R, Dummy) for 1pixel

**pbyteBGR**

Set the pointer of the BGR (or monochrome) image.

The image is overwritten afater process.

wBitsPerEachColor

Set the pointer that obtains the number of the bits represented in each color (B, G and R) of the BGR image from 8, 10 or 12 bits.

[Return]

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function processes the edge enhancement (sharpness) of the image

Please set the sharpness setting with StCam\_SetSharpnessMode before use this function.

### **StCam\_DetectDefectPixel**

This function registers the pixel defect position based on the RAW image

```
int StCam_DetectDefectPixel(  
    void* hCamera,          // Library handle  
    unsigned long dwWidth,   // Width  
    unsigned long dwHeight,  // Height  
    unsigned char* pbyteRaw, // RAW image buffer  
    unsigned short wThreshold // Threshold for pixel defect  
);
```

#### **[Parameters]**

**hCamera**

Set the library handle that obtained by the StCam\_Open function.

**dwWidth**

Set the width of the image.

**dwHeight**

Set the height of the image.

**pbyteRaw**

Set the pointer of the buffer for the RAW image that is necessary to select the light shading image.

**wThreshold**

Set the threshold of the defect pixel.

If the pixel level is greater than this threshold, the pixel is defect pixel.

#### **[Return]**

If the function was successful, return the 0.

If the function failed, return an ErrorCode of Linux.

[Descriptions]

This function registers the pixel defect position based on the RAW image

This function is only available STC\_CID\_DEFECT\_PIXEL\_CORRECTION\_MODE  
valid camera.