2024

EHD SC-ITR Series Cooled Cameras Manual

EHD imaging GmbH

25.4.2024

1.1 SC-ITR Series TE-Cooling USB3.0 CMOS Camera

1.1.1 The Basic Characteristic of the SC-ITR Series

SC-ITR series camera adopts SONY Exmor CMOS and GPIXEL GSENSE and GLUX sCMOS sensor as the image-picking device and USB3.0 is used as the transfer interface to increase the frame rate.

With the two-stage peltier cooling sensor chip to 40 degree below ambient temperature. This will greatly increase the signal to noise ratio and decrease the image noise. Smart structure is designed to assure the heat radiation efficiency and avoid the moisture problem. Electric fan is used to increase the heat radiation speed.

SC-ITR series comes with advanced video & image processing application EHDView / EHDLite; Providing Windows/Linux/OSX multiple platform SDK; Native C/C++, C#/VB.NET, DirectShow, Twain Control API;

The SC-ITR can be widely used in low light environment and microscope fluorescence image capture and analysis, as well as the astronomy deep sky application.

SC-ITR Series (Square Housing)

The basic characteristic of SC-ITR series can be summarized as follows:

- Standard camera with SONY Exmor CMOS and GPIXEL GSENSE sensors;
- 8 Bit /12 Bit or 16 Bit switchable (depends on sensor)
- TE-cooling with controllable electric fan;
- Sensor chip cooling up to 40°C below ambient temperature;
- Working temperature can be regulated to specified temperature in 5 minutes;
- Smart structure to assure the heat radiation efficiency and avoid the moisture problem;
- IR-CUT/AR coated windows (Optional);
- C-Mount
- USB3.0 5Gbit/second interface ensuring high speed data transmission;
- Up to 3600 seconds long time exposure;
- Ultra-Fine color engine with perfect color reproduction capability;
- Support the capture of video and image in software / hardware trigger mode
- With advanced video & image processing application EHDView/EHDLite;
- Support both video and trigger modes;
- Providing Windows/Linux/Mac OS multiple platforms SDK;
- Native C/C++, C#/VB.NET, DirectShow, Twain control API;

1.1.2 SC-ITR Series Datasheet

Order Code	Sensor & Size mm	Pixel µm	G Sensitivity Dark Signal	FPS/Resolution ADC	Binning	Exposure
SC492M-ITR	45M/IMX492(M) 4/3" (19.11x13.00)	2.315x2.315	351mV with 1/30s 0.12mV with 1/30s	8.1@8176x5616 30.0@4080x2808 8 bit / 12 bit	1x1 2x2	0.1ms – 3600s
SC571M-ITR	26M/IMX571(M) 1.8' (23.48x15.67)	3.76x3.76	871mv with 1/30s 0.07mV with 1/30s	14@6224x4168 37@3104x2084 110@2064x1386 8 bit / 16 bit	1x1 2x2 3x3	0.1ms – 3600s
SC571C-ITR	26M/IMX571(C) 1.8' (23.48x15.67)	3.76x3.76	485mv with 1/30s 0.1mV with 1/30s	14@6224x4168 37@3104x2084 110@2064x1386 8 bit / 16 bit	1x1 2x2 3x3	0.1ms – 3600s
SC269C-ITR	21M/IMX269(C) 4/3" (17.4x13.1)	3.3x3.3	400mV with 1/30s 0.1mV with 1/30s	17@5280x3954 56@2640x1976 67@1760x1316 192@584x438 8 bit / 12 bit	1x1 2x2 3x3 9x9	0.1 – 3600s
SC183M-ITR	20M/IMX183(M) 1" (13.058x8.755)	2.4x2.4	388mV with 1/30s 0.27mV with 1/30s	19@5440x3648 48@2736x1824 59@1824x1216 8 bit / 12 bit	1x1 2x2 3x3	0.1 – 3600s
SC183C-ITR	20M/IMX183(C) 1" (13.058x8.755)	2.4x2.4	462mV with 1/30s 0.27mV with 1/30s	19@5440x3648 48@2736x1824 59@1824x1216 8 bit / 12 bit	1x1 2x2 3x3	0.1 – 3600s
SC294C-ITR	10.3M/IMX294© 4/3" (19.11x13.0)	4.63x4.63	419mV with 1/30s 0.12mV with 1/30s	30@4128x2808 59@2048x1080 87@1360x720 8 bit / 12 bit	1x1 2x2 3x3	0.1ms – 3600s
SC294C-ITR	10.3M/IMX294© 4/3" (19.11x13.0)	4.63x4.63	419mV with 1/30s 0.12mV with 1/30s	30@4128x2808 59@2048x1080 87@1360x720 8 bit / 12 bit	1x1 2x2 3x3	0.1ms – 3600s
SC294M-ITR	10.3M/IMX294(M) 4/3" (19.11x13.0)	4.63x4.63	701mV with 1/30s 0.12mV with 1/30s	30@4128x2808 59@2048x1080 87@1360x720 8 bit / 14 bit	1x1 2x2 3x3	0.1ms – 3600s
SC533M-ITR	9M/IMX533(M) 1" (11.28x11.28)	3.76x3.76	877mV with 1/30s 0.04mV with 1/30s	40@2992x3000 62@1488x1500 186@992x998 8 bit / 14 bit	1x1 2x2 3x3	0.1ms – 3600s

SC533C-ITR	9M/IMX533(C) 1" (11.28x11.28)	3.76x3.76	534mV with 1/30s 0.04mV with 1/30s	40@2992x3000 62@1488x1500 186@992x998 8 bit / 14 bit	1x1 2x2 3x3	0.1ms – 3600s
SC585C-ITR	8.3M/IMX585(C) 1/1.2" (11.14x6.26)	2.9x2.9	5970mV with 1/30s 0.15mV with 1/30s	45@3840x2160 70@1920x1080 8 bit / 12 bit	1x1 2x2	0.1ms – 3600s
SC428M-ITR	7.0M/IMX428(M) GS 1" (11.28x11.28)	4.5x4.5	3354mV with 1/30s 0.15mV with 1/30s	51@3200x2200 133@1584x1100 8 bit / 12 bit	1x1 2x2	0.1ms – 3600s
SC428C-ITR	7.0M/IMX428(C) GS 1" (11.28x11.28)	4.5x4.5	2058mV with 1/30s 0.15mV with 1/30s	51@3200x2200 133@1584x1100 8 bit / 12 bit	1x1 2x2	0.1ms – 3600s
SC432M-ITR	1.7M/IMX432(M) GS 1" (11.28x11.28)	9.0x9.0	8100mV with 1/30s 0.3mV with 1/30s	98@1600x1100 8 bit / 12 bit	1x1	0.1ms – 3600s
SC432C-ITR	1.7M/IMX432(C) GS 1" (11.28x11.28)	9.0x9.0	4910mV with 1/30s 0.3mV with 1/30s	98@1600x1100 8 bit / 12 bit	1x1	0.1ms – 3600s
SC2020UV-ITR	4.2M/GSENSE2020BSI UV 1.2" (13.31x13.31)	6.5x6.5	1.1x108(e-/((W/m2).s)) 0.21(e-/s/pix)@-20°C	74@2048x2048 8 bit / HDR 16 bit	1x1 2x2	0.012ms–3600s
SC9701UV-ITR	1.3M/GLUX9701BSI UV 1" (11.28x11.28)	9.76x9.76	2.57x108(e-/((W/m2).s)) 0.08(e-/s/pix)@-28°C	30@1280x1024 30@640x512 8 bit / HDR 16 bit	1x1 2x2	0.1ms – 3600s
SC1605UV-ITR Under development	0.5M/GLUX1605BSI UV 1" (12.8x9.6)	16.0x16.0	TBD	60@800x600 60@400x300 8 bit / HDR 16 bit	1x1 2x2	0.1ms – 3600s

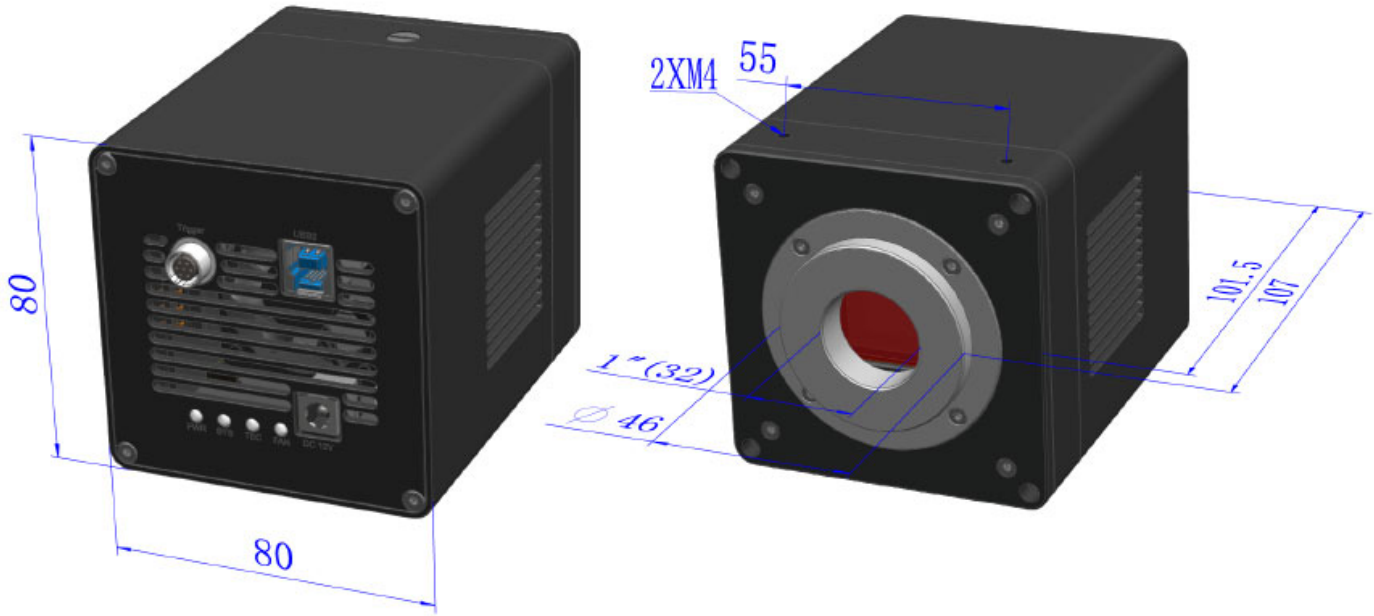
C: Color; M: Monochrome, UV: Ultraviolett, GS: Global Shutter

Other Specification for SC-ITR Series	
Spectral Range	190-1000nm (The spectral response range of each model is different. Please refer to the product manual of each model for detailed parameters)
Protect Windows	IR CUT (AR protection glass is optional)
White Balance	ROI White Balance/ Manual Temp Tint Adjustment/NA for Monochromatic Sensor
Color Technique	Ultra-Fine Color Engine/NA for Monochromatic Sensor
Capture/Control SDK	Windows/Linux/macOS/Android Multiple Platform SDK (Native C/C++, C#/VB.NET, Python, Java, DirectShow, Twain, etc)
Recording System	Still Picture and Movie (Free running mode or trigger mode)
Cooling System*	Two-stage TE-cooling System -40°C below ambient
IO Interface	One optocoupler isolation input, one optocoupler isolation output, two direct connection GPIO
Operating Environment	
Operating Temperature (°C)	-10~ 50
Storage Temperature (°C)	-20~ 60
Operating Humidity	30~80%RH
Storage Humidity	10~60%RH
Power Supply	DC 5V over PC USB Port External Power Adapter for Cooling System, DC12V, 3A
Software Environment	
Operating System	Microsoft® Windows® XP / Vista / 7 / 8 /10 (32 & 64 bit)

	OSx(Mac OS X) Linux
PC Requirements	CPU: Equal to Intel Core2 2.8GHz or Higher
	Memory:2GB or More
	USB Port:USB3.0 High-speed Port
	Display:17” or Larger
	CD-ROM

1.1.3 Dimension of SC-ITR Series and Connection

The SC-ITR series body, made from tough, alloy with CNC technique, ensures a heavy duty, workhorse solution. The camera is designed with a high quality IR-CUT or AR to block the IR light or protect the camera sensor. The fan‘s vibration is minimized to the low level to eliminate the vibration caused imaging blur. This design ensures a rugged, robust solution with an increased lifespan when compared to the other industrial camera solutions.



Dimension of ITR (Square) Series

2 Description of Software Development

2.1 SDK introduction

The SDK download link is as follows:

https://www.ehd.de/products/driver/driver.htm

2.1.1 SDK support platform

- Win32:
  x86: XP SP3 and above versions; CPU needs to support the SSE2 instruction set at least
  x64: Win7 and above versions
  arm: Win10 and above versions
  arm64: Win10 and above versions
- WinRT: x86, x64, arm, arm64□Windows 10 or above versions
- macOS: x86 and x64 bundle□macOS 10.10 or above versions
- Linux: core 2.6.27 and above versions
  x86: CPU needs to support at least SSE3 instruction sets; GLIBC 2.8 and above
- x64: GLIBC 2.14 and above versions
- armel: GLIBC 2.17 and above versions. Compiled by toolchain arm-linux-gnueabi (version 4.9.2)
  armhf: GLIBC 2.17 and above versions; Compiled by toolchain arm-linux-gnueabihf (version 4.9.2)
  arm64: GLIBC 2.17 and above versions; Compiled by toolchain aarch64-linux-gnu (version 4.9.2)
- Android: arm, arm64, x86, x64; Compiled by android-ndk-r18b.

2.1.2 SDK content brief introduction

EHD SC-ITR series cameras support a variety of API, including: Native C/C++, NET / C#VB.NET, DirectShow, Twain, LabView and so on. Compared with the other API, as the low-level API, native C/C++ API is characterized by using pure C/C++ development, independent of other runtime libraries, having simple interface and flexible control. This SDK package contains all the resources and information you need to use, as follows:

- inc:
  nncam.h, C/C++ Header file.
- win: Microsoft Windows Platform file
  - ✧ *dotnet:*

  nncam.cs, Support for C#. nncam.cs, use P/Invoke to call to nncam.dll. Please copy nncam.cs to your C # project.
  nncam.vb, Support for VB.NET. nncam.vb uses P/Invoke to call to nncam.dll. Please copy nncam.vb to your VB.NET project
  - ✧ *x86:*

  nncam.lib, x86 lib file.
  nncam.dll, x86 dynamic library files.
  democpp.exe, x86 C++ demo execute the procedure.
  - ✧ *x64:*

  nncam.lib, x64 lib file.
  nncam.dll, x64 dynamic library files.
  democpp.exe, x64 C++ demo execute the procedure.
  - ✧ *arm:*

  nncam.lib, arm lib file.

nncam.dll, arm dynamic library files.

✧ *arm64:*

nncam.lib, arm64 lib file.

nncam.dll, arm64 dynamic library files.

✧ *winrt:*

They can be applied for Dynamic library files of WinRT/ UWP (Universal Windows Platform)/ Windows Store App. They are compatible with Windows Runtime and can be referenced by the Universal Windows Platform app. If you use C # to develop UWP, you can use the nncam.cs to wrap class.

✧ *Please pay attention to the DeviceCapability of uwp. Refer to How to add USB device capabilities to the app manifest. (Microsoft seems to limit the Device entry under DeviceCapability to no more than 100) demouwp.zip is a simple example of uwp. Please modify vid and pid. under DeviceCapability in the file Package.appxmanifest before compiling the run example.*

✧ *drivers: (Cameras produced after 2017.1.1 support WinUSB. You no longer need to install drivers on Windows8 and above)*

The x86 folder contains the kernel state driver file for x86, including nncam.cat, nncam.inf and nncam.sys.

The x64 folder contains the kernel state driver file for x64, including nncam.cat, nncam.inf and nncam.sys

✧ *samples:*

1.democpp, take C++ for example. This example shows an enumeration device, an open device, a preview video, a snap image, a set resolution, a trigger and a wide variety of picture formats (bmp, jpg, png etc.) save the image to the file, wmv format video, trigger mode, I/O control, etc. This example uses the Pull Mode mechanism. In order to keep the code clean, the WTL library used by the example can be downloaded from this link http: //sourceforge.net/projects/wtl/

2.demopush, take C++ for example, using the Push Mode mechanism, StartPushModeV3

3.demomfc, A simple C++ example. it uses MFC as the GUI library, supports opening devices, previews video, captures images, sets resolution and saves images to files in a variety of image formats (.bmp, .jpg, .png, etc.). This example uses the Pull Mode mechanism.

4.demowinformcs1, take C# winform for example. It supports to open the device, preview video, capture images, save pictures to files and set white balance. This example uses the PullMode mechanism, called StartPullModeWithWndMsg.

5.demowinformcs2, take C# winform for example. It supports to open the device, preview video, capture images, save pictures to files, set white balance. This example uses the Pull Mode mechanism called StartPullModeWithCallback

6.demowinformcs3, take C# winform for example. It supports to open the device, preview video, capture images, save pictures to files, set white balance. This example uses the Push Mode mechanism called StartPushMode

7.demowinformvb, take VB.NET winform for example. It supports to open the device, preview video, capture images, save pictures to files and set white balance. This example uses the Pull Mode mechanism.

- linux: Linux platform file

  Udev: 99-nnpcam.rules, udev rule file

  Please refer to: http: //reactivated.net/writing_udev_rules.html

  ✧ *c#: nncam.cs, Support. Net Core C#. nncam.cs calls to libnncam.so. using P/Invoke Please copy nncam.cs to your C # project.*

  ✧ *x86: libnncam.so, X86 version of so file.*

  ✧ *x64: libnncam.so, x64 version of so file.*

  ✧ *armel: libnncam.so, armel version so file, toolchain is arm-linux-gnueabi*

  ✧ *armhf: libnncam.so, armhf version so file, toolchain isarm-linux-gnueabihf*

  ✧ *arm64: libnncam.so, Arm64 version so file, toolchain is aarch64-linux-gnu*

- android: Android platform. libnncam.so. for the four architectures of arm, arm64, x86, x64
- mac: macOS platform file
- python: nncam.py and example code.
- java: nncam.java and example code(Console and Swing)
- doc: SDK uses documentation, simplified Chinese, English.
- sample:

  ✧ *demosimplest, the simplest example is about 60 lines of code.*

  ✧ *demoraw, RAW data and static capture, about 120 lines of code.*

- extras:
  - *directshow: DirectShow SDK and demo programs.*
  - *twain: TWAIN SDK*
  - *labview: Labview SDK and demo programs.*
  - *MATLAB: MATLAB demo programs.*

2.2 Client democpp description

As shown in Figure 2-1, “1” is the control menu area and "2" is the video display area.

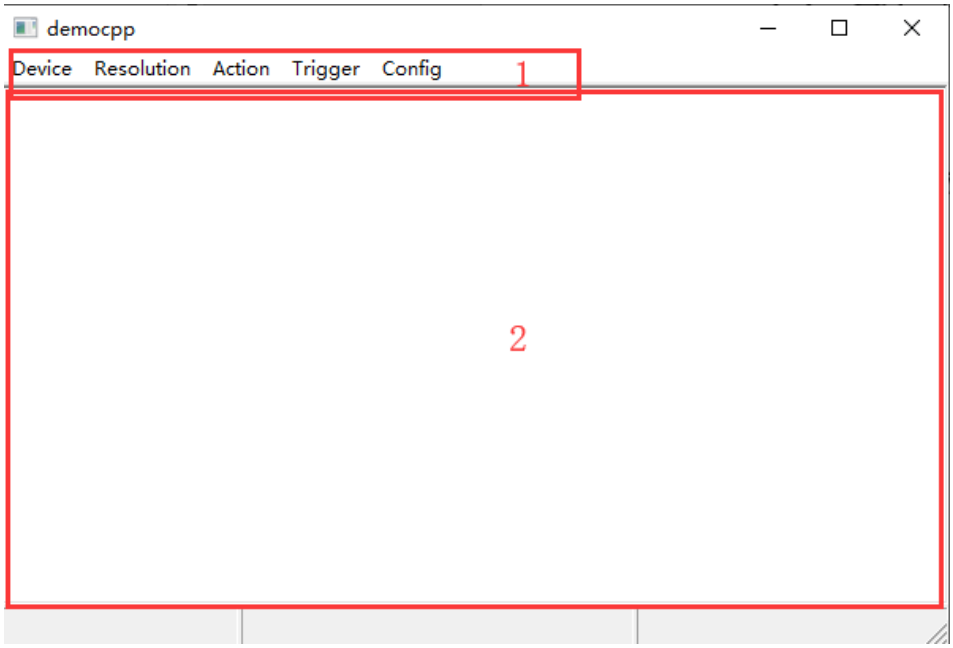


Figure 2-1 democpp interface

The main features of the control menu are:

Device: camera names installed are listed under this menu. Clicking the camera name to open the camera;
Resolution: switch the resolution and capture the image;
Action: Pause, ROI setting, test image, read firmware version number, hardware version number, production date, etc.
Trigger: Defining the trigger mode, the I/O port setting and the like;
Config: Set exposure, gain, white balance, frame rate, etc.

2.3 EHDView UI description

EHDView software fully controls all camera features and streams high-speed videos by USB port using Ultra Fine™ color engine. Ultra Fine™ color engine contains excellent procedure of processing RAW data and thus realizes the conversion of sensor detected data to image. Furthermore, EHDView also provides many advanced video and image processing features, such as image gray level correction, 2D measurement, stitching, depth of field extension, video watermarking, color synthesis, image segmentation and counting and so on. EHDView’s multilingual environment can support any language and currently includes, but is not limited to, English, simplified Chinese, traditional Chinese, German, Japanese, Russian, French, Italian, Polish, Turkish, etc. The UI of EHDView is shown in Figure 2-2.

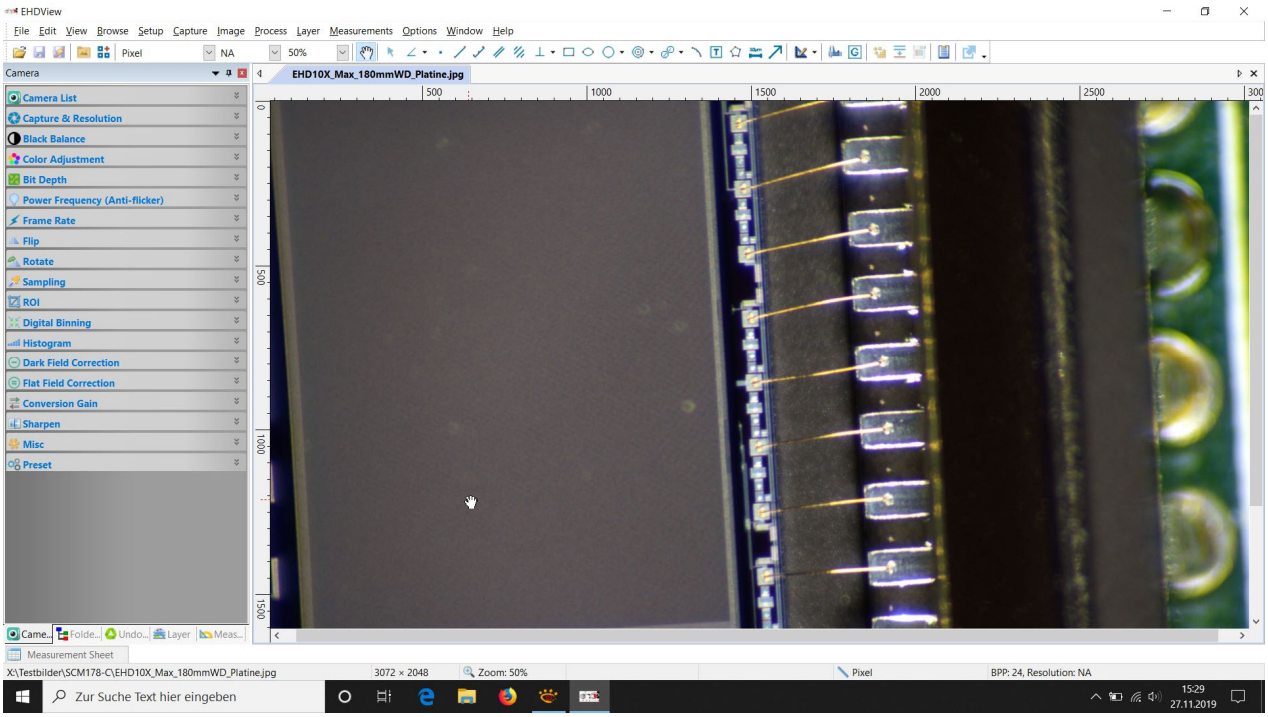


Figure 2-2 EHDView UI

The main features of EHDView are

Exposure and gain	Automatic exposure, manual exposure, gain up to 5x;
White balance	Automatic white balance; can be adjusted by manually setting the color temperature and color;
Color adjustment	Color, saturation, brightness, contrast, gamma value adjustment feature;
Frame rate control	According to the performance of different computers and USB, the compatibility of the camera can be realized by adjusting the frame rate;
Light source frequency control	Natural light / DC, AC50HZ, AC60HZ selection button eliminates video flicker;
Acoustic image	The direction of the adjustable sample can be adjusted by "horizontal" or "vertical" and is consistent with the direction of the visual system;
Sampling and neighborhood averaging and other features	Neighborhood average can improve the signal-to-noise ratio of video stream and sampling extraction mode can ensure the sharpness of video stream. Support video stream histogram expansion, image negative and positive film switching, gray calibration, clarity factor calculation to facilitate video focusing;
Parameter saving	Load, save, overwrite, load, export custom camera panel control;
Video feature	video broadcasting, timing capture, video recording, video watermarking, watermark movement alignment, watermark rotation alignment, video grid overlay, video measurement, video scaling, grayscale scaling calibration, video high dynamic (HDR), video depth of field expansion, video image stitching, video scale, the date and the like are superimposed;
Image processing and enhancement	Image contrast control and adjustment, image de-noising, various image filter algorithms, image mathematical morphology algorithm, image rotation, image scaling and image printing;
2D measurement	Convenient and practical video and image size calibration, a variety of video and image two-dimensional geometric measurements such as length, area, perimeter and angle, etc., the measurement results can be controlled according to image characteristics or preferences;
Image mosaicking	The image stitching can automatically splice the sequence image into a mosaic image. a video window, an image window and a browsing window splicing operation are supported;
EDF (Expansion of depth of field)	Depth of field expansion can get ultra-clear images beyond the conventional depth of field by focusing on different layers of images. EHDView supports EDF depth expansion in three windows: video window, image window and browsing window. For different images, EHDView also provides three different depth of field extension algorithms, such as maximum contrast, weighted average and FFDSSD. In addition, the translation, rotation and automatic depth of field expansion between different focus images are considered to ensure the accuracy and rapidity of EDF;
Professional segmentation and counting	The segmentation and counting of EHDView provides six image segmentation methods for users to call according to different image characteristics. The six segmentation methods are watershed, dark OTSU, bright OTSU, RGB histogram, HSV histogram, color segmentation and so on. Users can select any of the six segmentation methods for segmentation, but after selecting any segmentation method, other segmentation methods will be disabled. After the segmentation is completed, there may be the adhesion of the counting object, which can be manually segmented by manual segmentation and the counting results can be counted and analyzed by selecting the counting result menu after confirming the expected results;
Image superposition denoising	EHDView image superposition denoising feature introduces advanced image matching technology, users only need to record a small video of their own image to be superimposed, then they can superimpose and output high fidelity images under the condition of displacement, rotation and magnification of frame burst images, which is simple and easy to use;
Color synthesis	Color synthesis can use black and white fluorescent light source images to create and configure color composite images. Fluorescent probes and colors can be selected directly from predefined data. The dye database of special probe can also be built by the user himself;

3 Camera Installation and Operation

3.1 Installation steps

1. Fix the camera onto the installation position and attach appropriate C-mount lens to the camera.

2. Verify that the camera is properly connected to the industrial computer or PC using the Micro USB3.0 cable that comes with the camera. Tighten the cable by fastening screw at the camera side.

3.2 Driver check

The Operation Systems under Windows 7(Including WIN7) require a normal installation of the drive before the camera is used and if the drive installation fails, the camera will not be found by the client software.

After the installation is complete, in the **Device Manager**, you can see the new device type, such as IUB4200KMB and then right-click the mouse button to see if the device drive is properly installed or not, as shown in Figure 3-1.

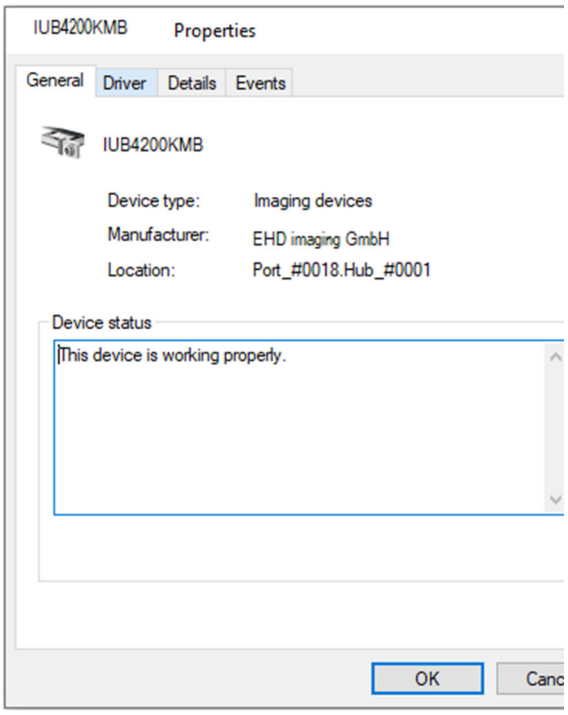


Figure 3-1 EHD driver attribute

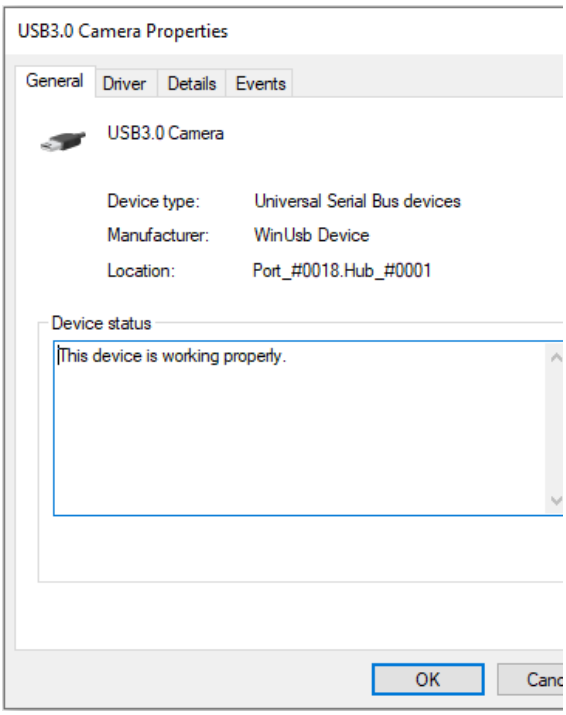


Figure 3-2 Win USB driver attribute

The operating systems above Windows 8 (Including Windows 8) will install the driver automatically after the camera is connected and the driver name is USB3.0 Camera, as shown in Figure 3-2.

3.3 Setup and operation

As shown in Figure 3-3, open democpp.exe and click "Device" in the top control menu, where all connected cameras are displayed and click on the corresponding camera name to run the camera.

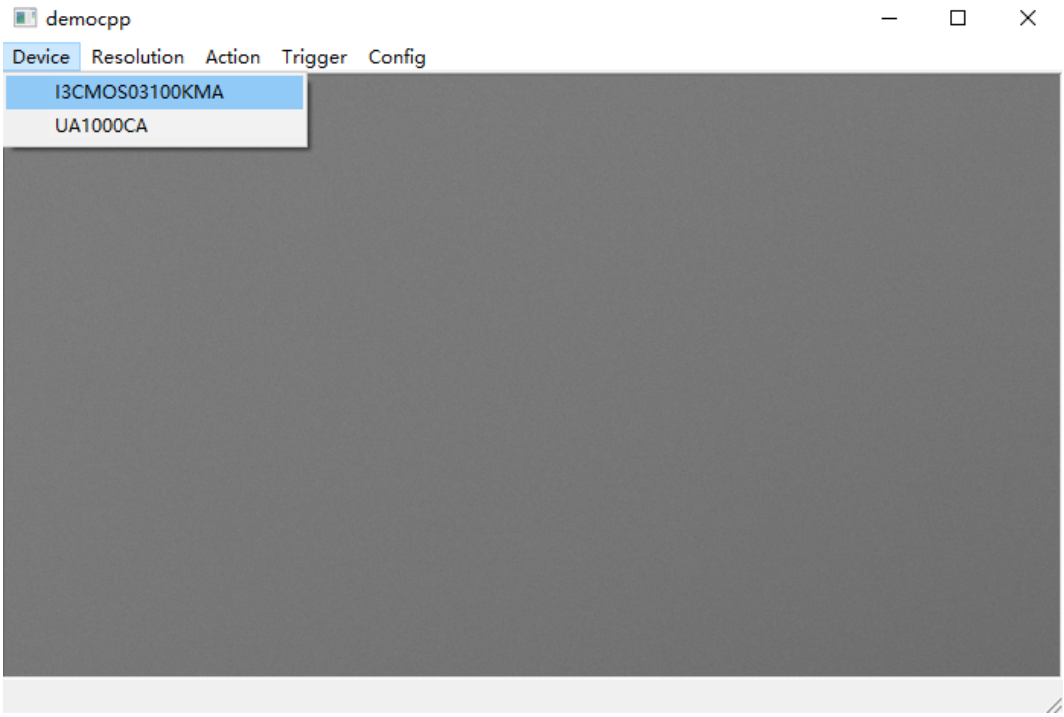


Figure 3-3 democpp UI

4 Main Features of democpp

4.1 Description of main features

As shown in Figure 4-1, in the **democpp**, click the "Resolution-> Preview" in the top control menu to select the resolution of the camera; "Resolution->Snap" captures image at current resolution; "Resolution->Snap Multiple" captures multiple images at the specified resolution.

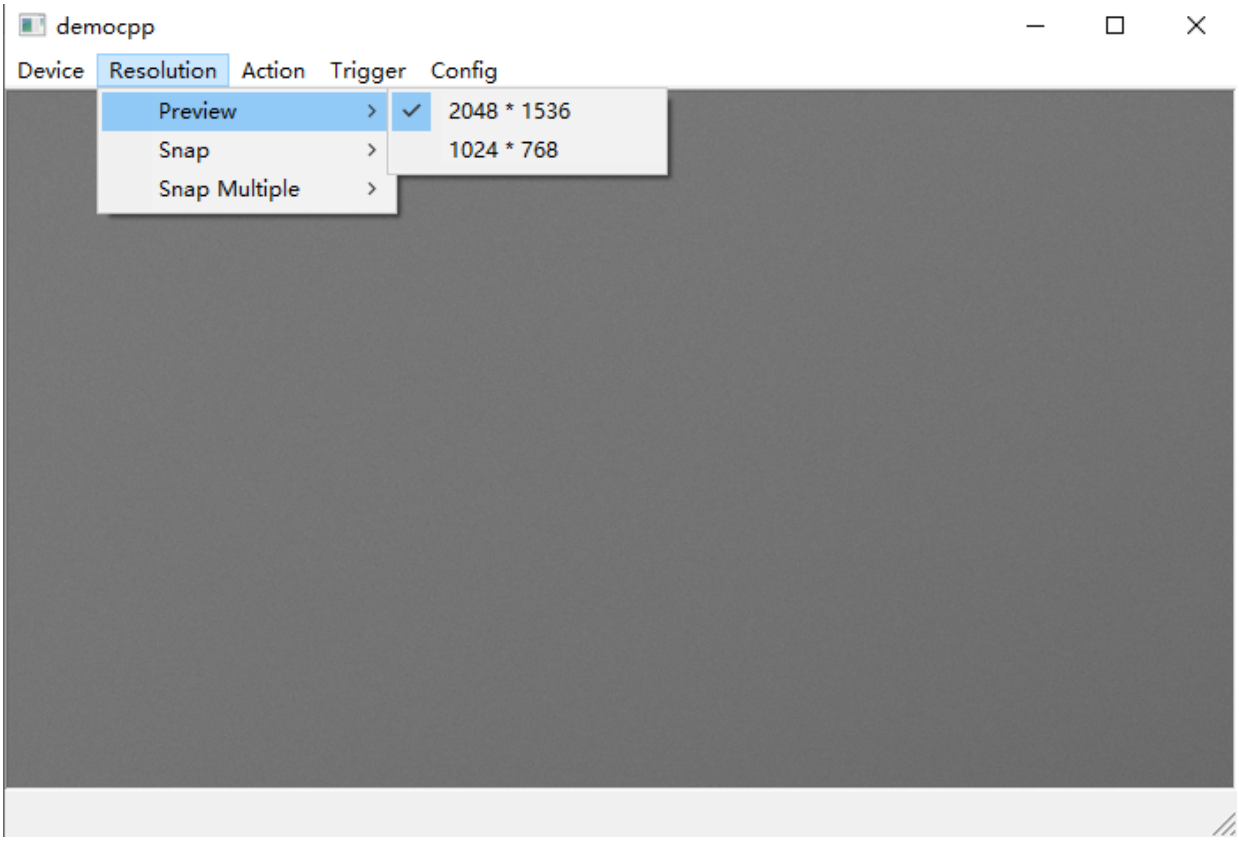


Figure 4-1 Acquisition and resolution

The following is the API code for the image capture operation:

```
//still image snap
nncam_Snap(HnnCam h, unsigned nResolutionIndex);
//multiple still image snap
nncam_SnapN (HnnCam h, unsigned nResolutionIndex, unsigned nNumber);
//The following is the API code that sets the resolution.:
nncam_put_Size(HnnCam h, int nWidth, int nHeight);
nncam_put_eSize(HnnCam h, unsigned nResolutionIndex);
```

4.2 Image format and frame rate

The camera supports a variety of image file formats and the setup of image region of interest. The smaller image ROI will have higher frame rate.

4.2.1 Frame rate

The maximum frame rate that the camera can achieve is determined by the following three factors:

- Frame readout time, the smaller the image height, the shorter the time required to read out, the higher the frame rate.
- Exposure time, the shorter the exposure time, the higher the frame rate.
- Bandwidth, the larger the bandwidth, the higher the frame rate that supports transmission.

As shown in Figure 4-2, in **democpp**, click "Config->Speed" in the top control menu and drag the slider bar in "Speed" dialog to set the frame rate.

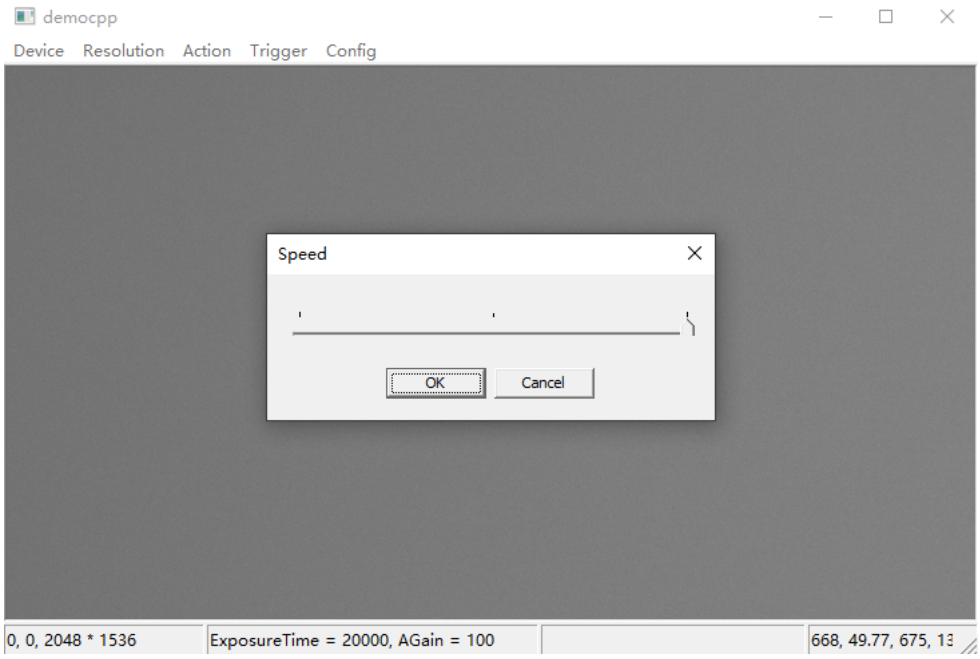


Figure 4-2 Frame rate setup

The following is the API code for the setup of the frame rate:

nncam_put_Speed(HnnCam h, unsigned short nSpeed);

4.2.2 Area of interest setup

When the user is only interested in some details of the image, the camera can output the image ROI according to the requirement. Setup the image ROI can reduce the transmission data bandwidth and improve the camera frame rate to a certain extent.

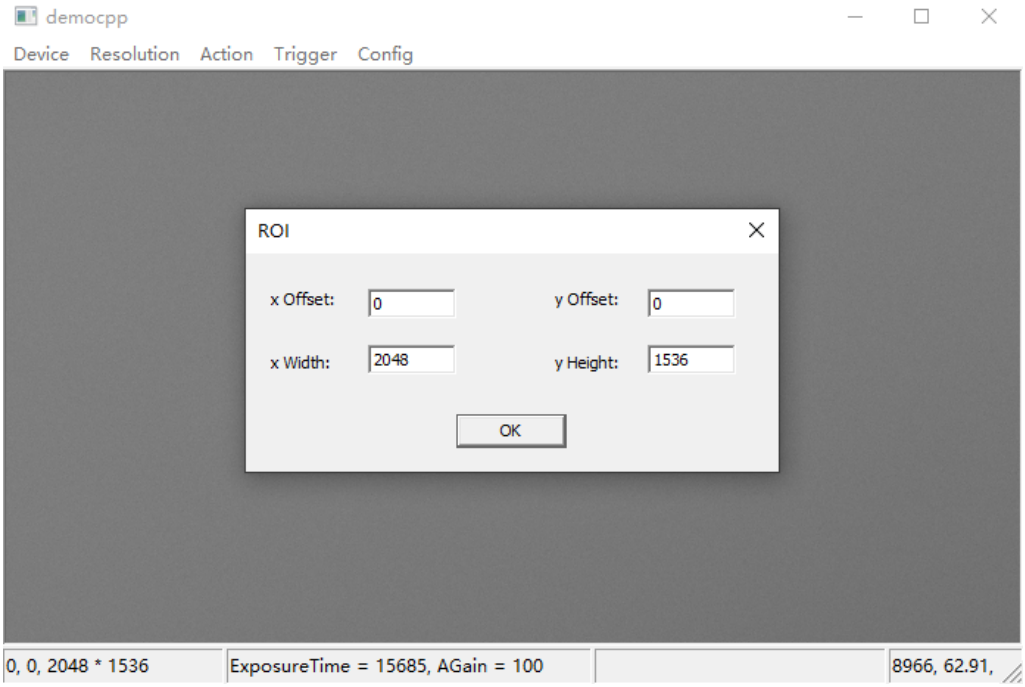


Figure 4-3 Area of interest setup

As shown in Figure 4-3, in **democpp**, click "Action->ROI" in the upper control menu and in "ROI" dialog, fill x Offset, y Offset, x Width, y Width to adjust the ROI, where the values in x Offset and y Offset represent the starting point of the ROI up left corner.

The following is the API code for the setup of the image ROI:

nncam_put_Roi(HnnCam h, unsigned xOffset, unsigned yOffset, unsigned xWidth, unsigned yHeight);

4.3 Global Shutter and Rolling Shutter

4.3.1 Global Shutter

For cameras that support global shutter, exposure starts in each line simultaneously. After the exposure, data is read out line by line, as shown in Figure 4-4.

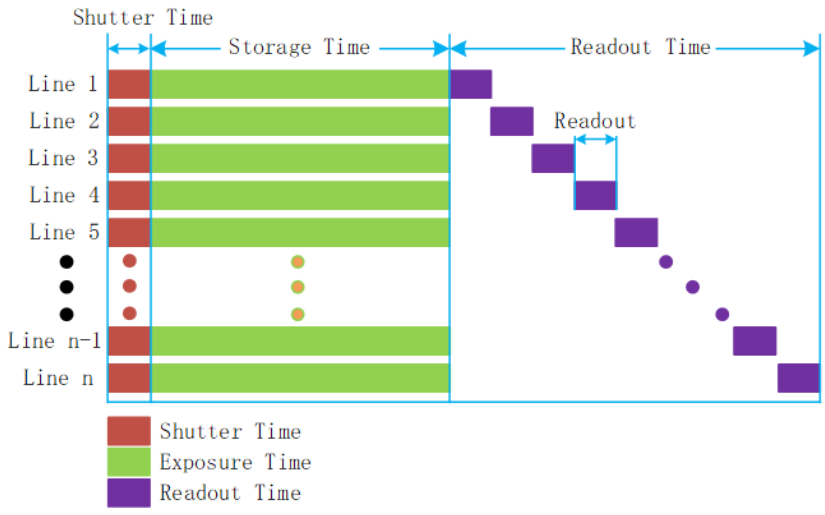


Figure 4-4 global shutter exposure principle

4.3.2 Rolling Shutter

For cameras that support rolling shutter, after the first line exposure, the next line begins to exposure, repeat in this way. Sensor receive exposure and data read the time length to be consistent, but the time of begin to receive exposure is inconsistent, as shown in Figure 4-5.

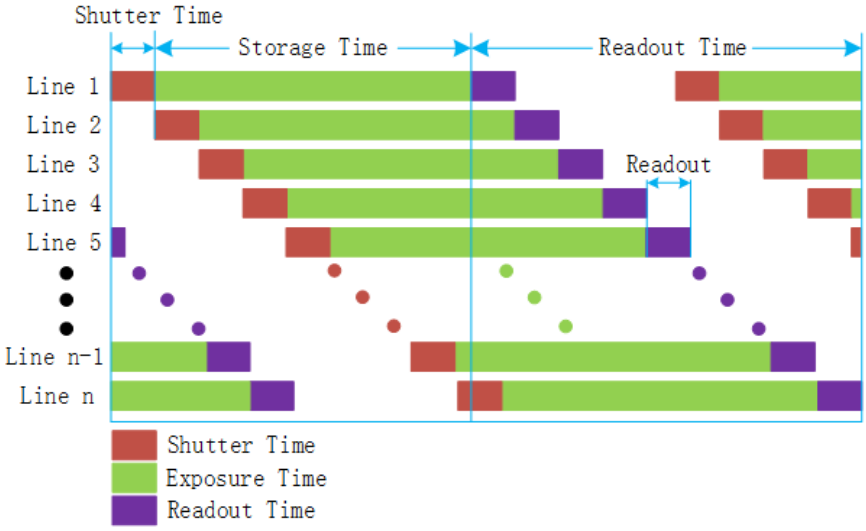


Figure 4-5 Rolling shutter exposure principle

4.4 Image acquisition and transmission

There are two image acquisition modes, free run mode and trigger mode. Among them, the free run mode is continuous acquisition mode and the trigger mode captures one or more frames of images according to the trigger signal. The trigger sources include software trigger and external trigger.

As shown in Figure 4-6, in democpp, the free run mode and trigger mode are switched by clicking on "Trigger" menu and choosing “Enter Trigger Mode” command. The "✔" checkmark indicates that the current mode is in trigger mode, otherwise free run mode.

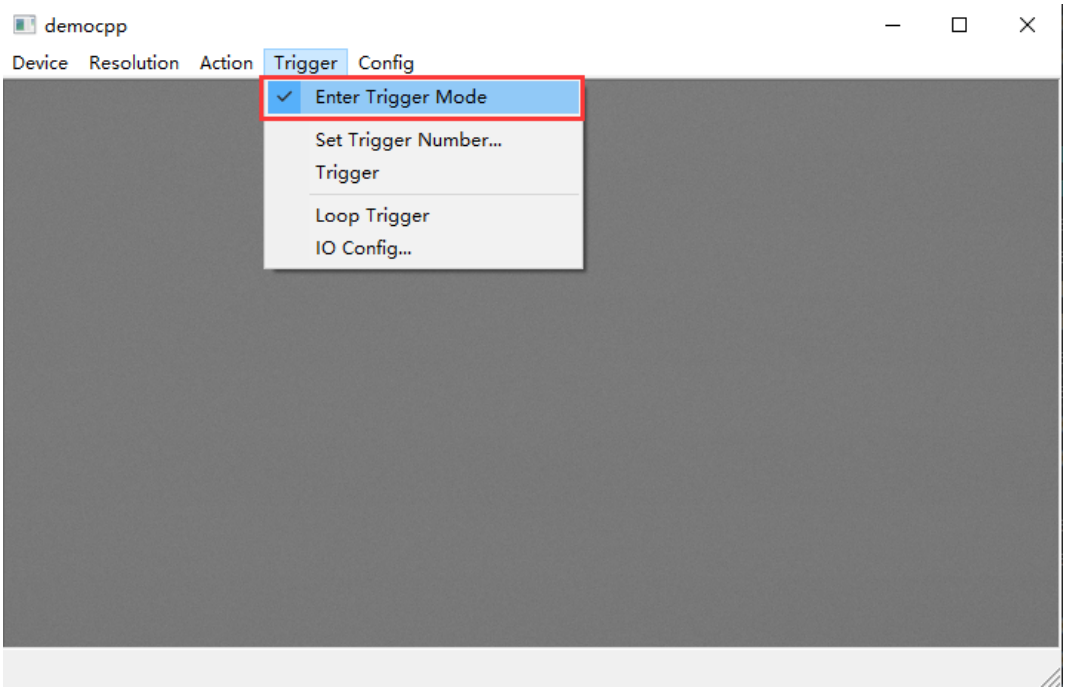


Figure 4-6 Image acquisition mode switching

The following are the API codes for the switching of the free run mode to trigger mode:

```
// 0 = video mode, 1 = software or simulated trigger mode, 2 = external trigger mode
nncam_put_Option(m_hCam, NNCAM_OPTION_TRIGGER, val);
```

4.4.1 Free run mode

Under free run mode, the user can control the camera to continuously output images. Starting the **democpp** software, connecting the camera and clicking run, the camera will run under free run mode by default. The camera continuously outputs the image according to the current setting.

4.4.2 Trigger mode

After the camera enters the trigger mode, it enters the waiting trigger state automatically. After receiving a trigger signal, the camera begins to expose and after the exposure is finished, the image data will be flashed out. Under trigger mode, image acquisition methods have single frame trigger, multi frame trigger, counter trigger and PWM trigger mode.

4.4.3 Trigger signal source selection

Under the trigger mode, trigger signal source is either from software trigger, or from external trigger. The external trigger signal is from either the pin isolated by opto-coupler or the non-isolated pin.

The following is the API code for the setup of the trigger source:

```
// Trigger Source:   0-> line0 , 1-> line2 , 2-> line3 , 3-> Counter , 4-> PWM , 5-> Software
nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_TRIGGERSOURCE, val, NULL);
```

- Software trigger

The camera supports software trigger mode. When a software trigger is executed, the client software will send the command through USB3.0 to activate the camera to acquire and transmit images.

As shown in Figure 4-7, in **democpp**, first click "Enter Trigger Mode" to enter trigger mode. Click "Set trigger Number" to define the number of triggers and finally click "Trigger" and the software will receive the number of triggers. If you click "Loop Trigger", you will enter a continuous trigger mode and clicking it again will exit the current trigger mode.

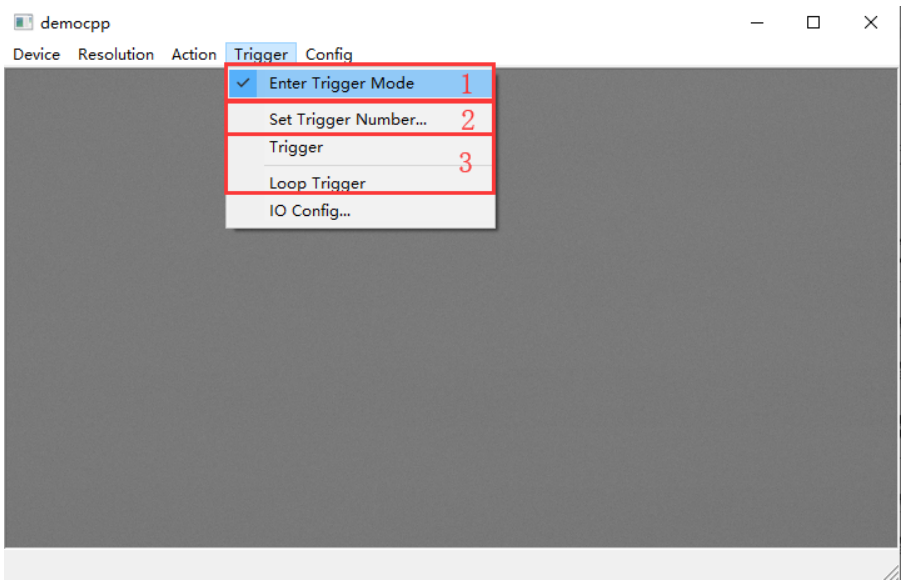


Figure 4-7 Software trigger setup

- External trigger

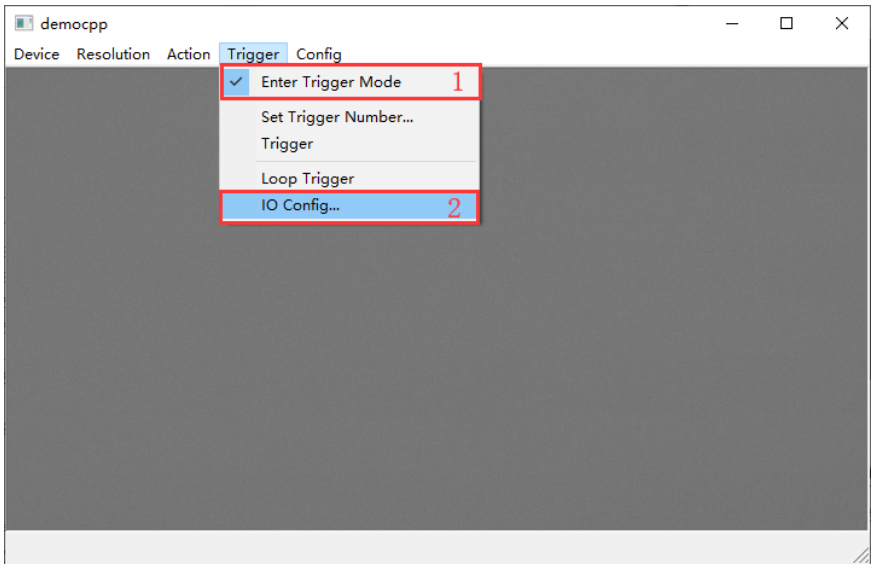


Figure 4-8 Set the external trigger source

Figure 4-8 shows the configuration of external trigger. First select "Enter Trigger Mode" to enter trigger mode. Then click "IO Config" and the I/O Control dialog will pop up as shown in Figure 4-9. The trigger source is selected in the "Trigger Source" and then click "OK". At this time a high pulse will trigger the camera on the corresponding line.

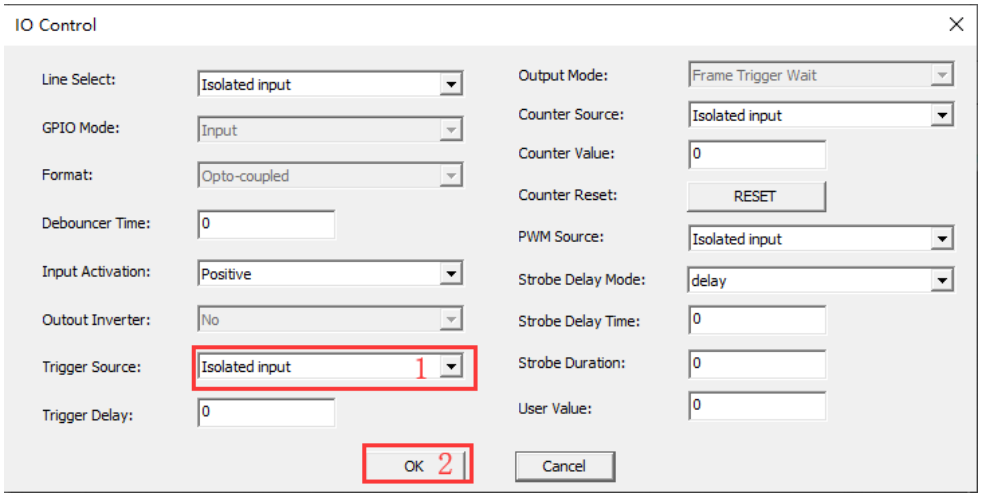


Figure 4-9 I/O control dialog

4.4.4 Frame burst mode

The camera provides a frame burst mode, that is, receiving one trigger signal and producing multiple burst images. The trigger frame value can range from 1 to 1023. “Burst Count = 1” means a one-frame image output, as shown in Figure 4-10, “Burst Count = 3” means a three-frame image output.

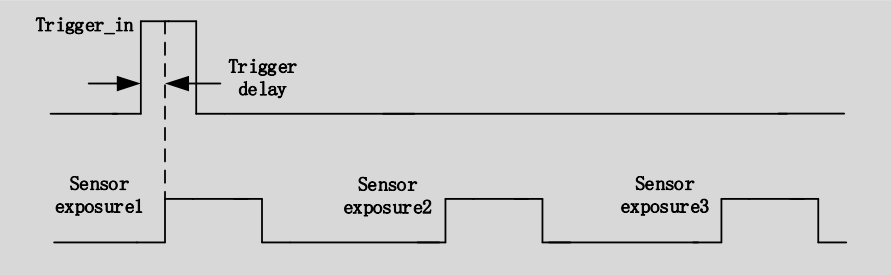


Figure 4-10 Frame burst trigger timing

Here is the API code for the setup of the trigger frame value

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_BURSTCOUNTER, val, NULL);

4.4.5 Counter trigger mode

Under this mode, trigger signal number is divided by user-defined counter value. For example, when you set the counter to 3, the camera needs to receive three trigger signals before it can begin exposure, as shown in Figure 4-11.

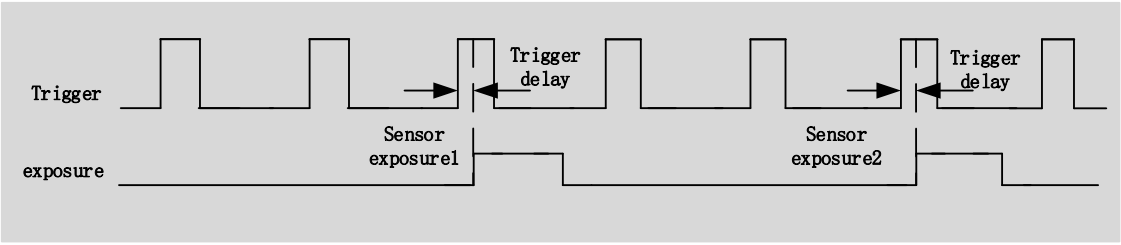


Figure 4-11 Counter trigger mode

The specific operation in **democpp** are shown in Figure 4-12. First, under "Trigger Source", select the trigger source as Counter, then click "Counter Source" to select the external trigger source that needs to be divided and configure the frequency division coefficient in "Counter Value" in the range of 1-1023. "Counter Reset" can clear the current frequency division counter to zero.

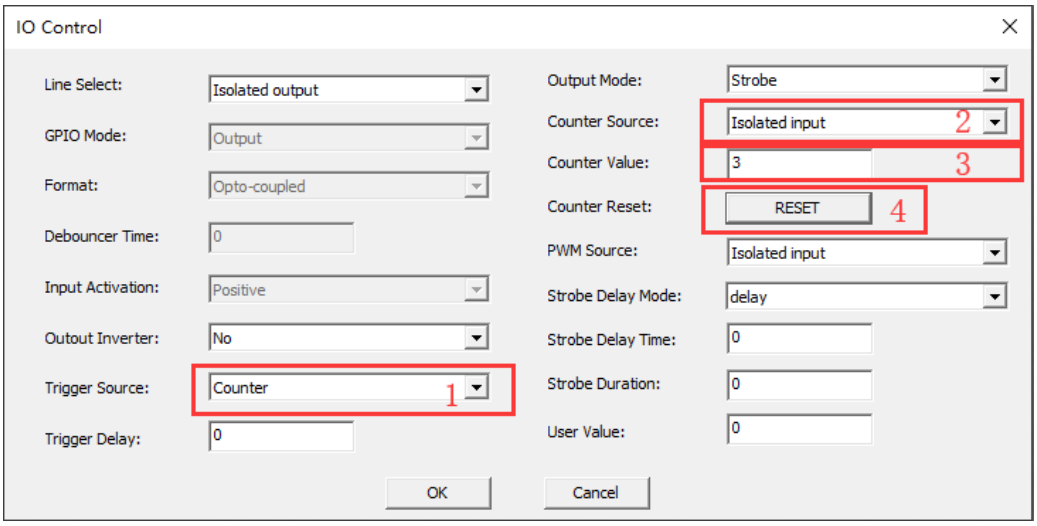


Figure 4-12 Counter trigger mode setup

The following is the API code for the setup of the counter trigger mode:

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_TRIGGERSOURCE, 3, NULL);

//Counter Source: 0-> line0 , 1-> line2 , 2-> line3

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_COUNTERSOURCE, val, NULL);

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_COUNTERVALUE, val, NULL);

4.4.6 PWM trigger mode

The camera provides Pulse Width Modulation (PWM) trigger mode, which controls exposure time by pulse width. The main difference between this mode and the standard single frame trigger mode is the exposure method. The exposure time per frame is determined by the trigger pulse width, as shown in Figure 4-13.

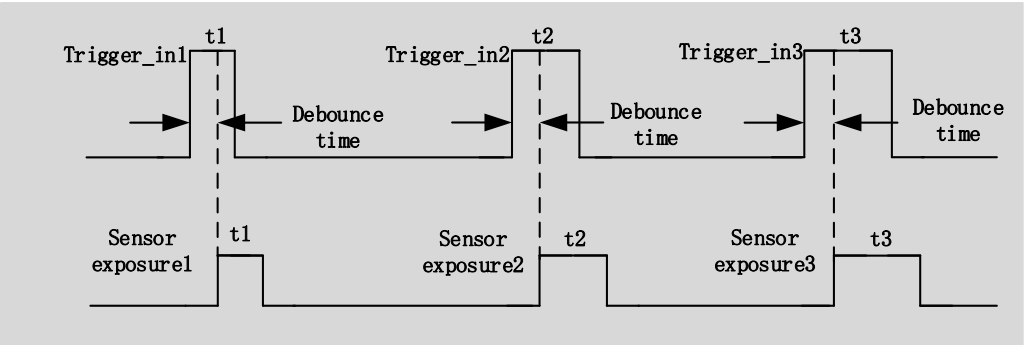


Figure 4-13 PWM mode timing

The following cameras with rolling shutter sensors do not support PWM trigger mode：

MaxCam-455M-TE, MaxCam-410C-TE, MaxCam2020e-TE, MaxCam-2020UV-TE, MaxCam-400UV-TE

As shown in Figure 4-14 in **democpp**, select the trigger source as PWM, under "Trigger Source" and click "PWM Source" to select the external trigger source for input.

The following is the API code for the setup of the counter trigger mode:

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_TRIGGERSOURCE, 4, NULL);

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_PWMSOURCE, val, NULL);

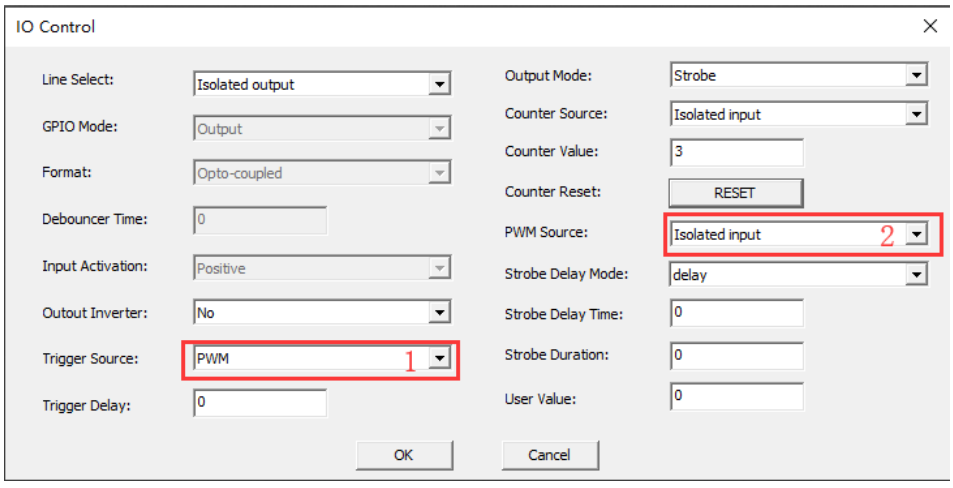


Figure 4-14 PWM mode parameter control

4.5 General Purpose I/O configuration

(This is for I3 series only)The camera with the hardware version number V2.0 and above has a configurable GPIO port, as shown in Figure 4-15, in the demo, enter the "IO Config" dialog, select the "Line Select" to be GPIO0 and then click the "GPIO Mode" to configure the input or output.

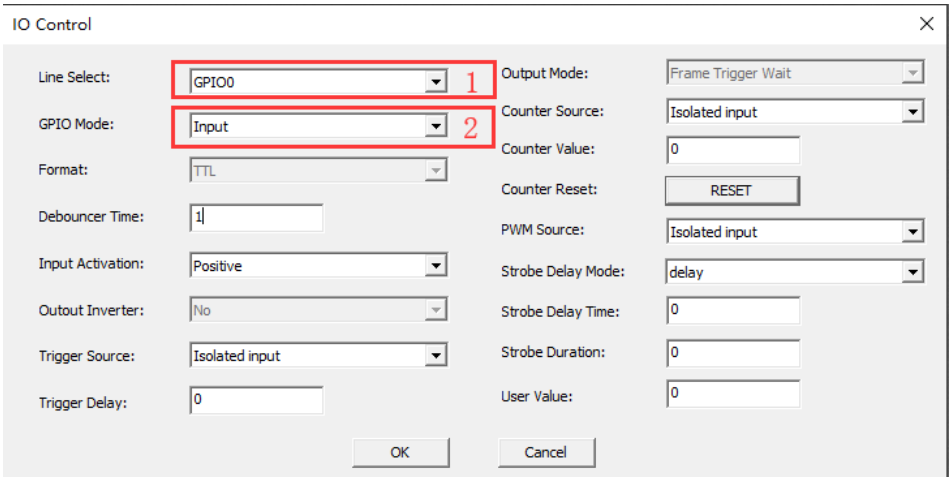


Figure 4-15 GPIO configuration

The following is the API code to configure the GPIO input and output direction:

nncam_IoControl(m_hCam, index, NNCAM_IOCONTROLTYPE_SET_GPIODIR, val, NULL);

4.6 Input signal

4.6.1 Signal debouncer

Because the external trigger input signal of the camera may have burr, if it goes directly into the internal logic of the camera, it will cause false trigger. The input trigger signal should be debounced. In addition, the effective pulse width of the trigger signal inputted by the user should be greater than the debouncer time, otherwise the trigger signal will be ignored. The timing is shown in Figure 4-16. If the effective pulse width of Trigger_in1 is less than the debouncer time, the trigger signal will be ignored.

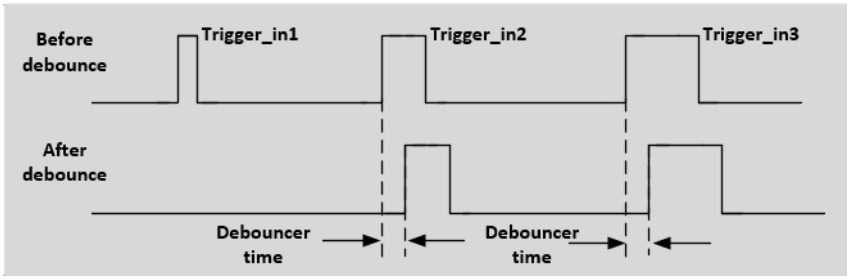


Figure 4-16 Signal debounce timing

As shown in Figure 4-17, in **democpp**, enter the "IO Config" dialog, click "Line Select" to select the input line and then set the debouncer time at "Debounce Time" in the range of 0≤20000 in microseconds.

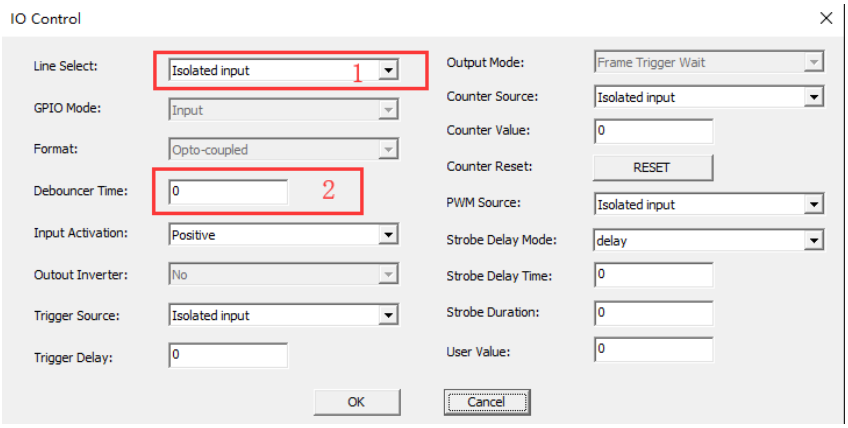


Figure 4-17 Signal Debounce setup

The following is the API code for the setup of the debouncer time:

nncam_IoControl(m_hCam, index, NNCAM_IOCONTROLTYPE_SET_DEBOUNCERTIME, val, NULL);

4.7 Output signal

The camera provides 4 output signal modes: Frame Trigger Wait, Exposure Active, Strobe and User Output.

As shown in Figure 4-18, in the "IO Config" dialog, first select the “Isolated output” in the "Line Select" combobox, then select the output signal mode in the "Output Mode" combobox, click "Output Inverter" to reverse the output signal.

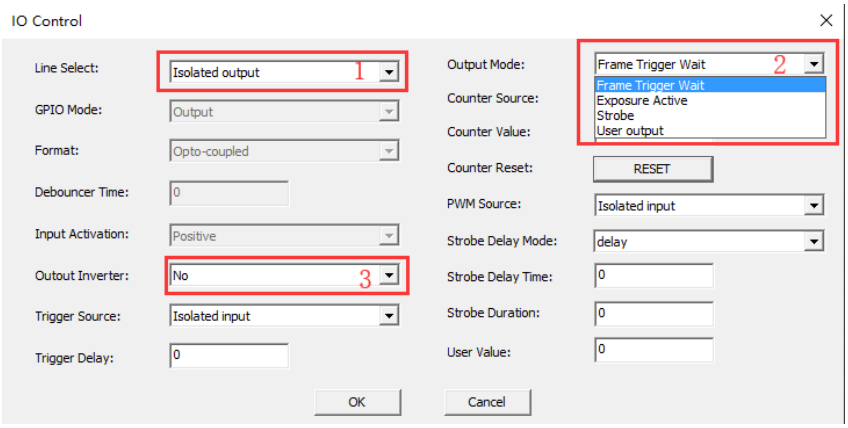


Figure 4-18 Output signal mode setup

The following is the API code for the setup of the output signal mode:

nncam_IoControl(m_hCam, index, NNCAM_IOCONTROLTYPE_SET_OUTPUTMODE, val, NULL);

// Output Mode: 0-> Frame Trigger Wait , 1-> Exposure Active , 2-> Strobe , 3-> User output

// index: 0-> line0 , 1-> line1, 2-> line2 , 3-> line3

4.7.1 Frame Trigger Wait

The “Frame Trigger Wait” signal is pulled low at the start of the exposure and is pulled high when the last frame of data is read out. The trigger signal inputted by the user should be in the valid period of the signal. If the user inputs a trigger signal when the signal is low, the trigger signal input at this time will be ignored. The following example is the case when Burst Count = 2, as shown in Figure 4-19.

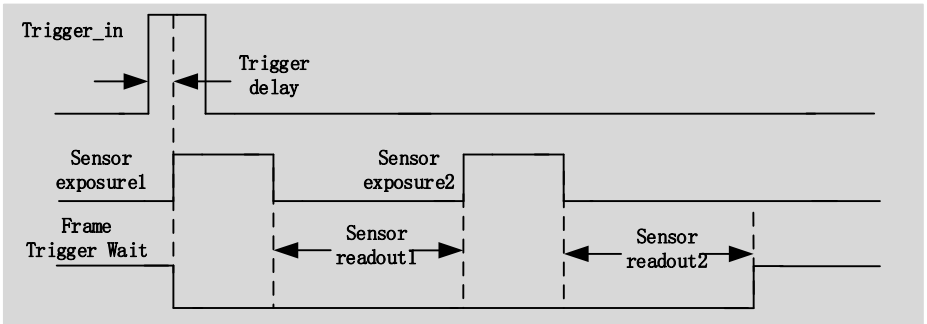


Figure 4-19 Frame Trigger Wait signal timing

4.7.2 Exposure Active

When “Exposure Active” signal is high, it indicates that the sensor is in the exposure process. This signal can be used as a flash trigger and is also useful when you are operating a system where either the camera or the object being imaged is movable. For example, assume that the camera is mounted on an arm mechanism and that the mechanism can move the camera to view different portions of a product assembly.

4.7.3 Strobe

Strobe can be used to control flash and other external devices. User can set the effective level duration, delay time and pre-delay time.

As shown in Figure 4-20, in the "IO Config" dialog of **democpp**, select the Output Mode as “Strobe”, click "Strobe Delay Mode" to select the “delay” or “pre-delay” and set the time in "Strobe Delay Time". "Strobe Duration" can set the effective level duration of Strobe.

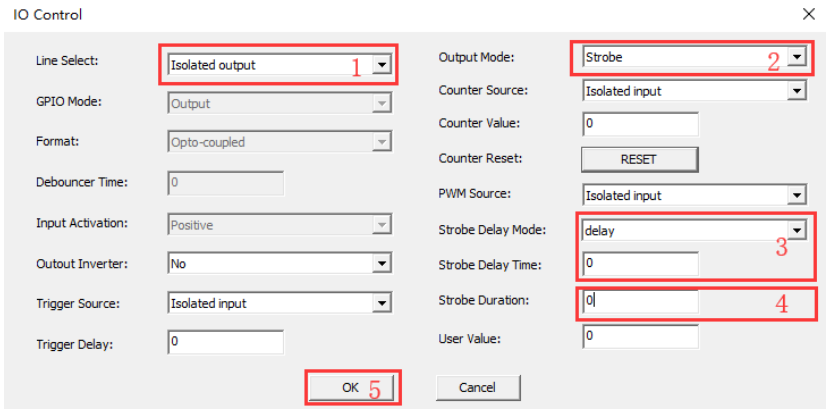


Figure 4-20 Strobe settings

The following is the API code for the setup of the output Strobe signal:

nncam_IoControl(m_hCam, index, NNCAM_IOCONTROLTYPE_SET_OUTPUTMODE, 2, NULL);

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_STROBEDURATION, val, NULL);

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_STROBEDELAYMODE, val, NULL);

nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_STROBEDELAYTIME, val, NULL);

- Strobe effective electrical level duration

As shown in Figure 4-21, the Strobe signal is activated at a high level. When the shutter starts to expose, the Strobe signal’s high duration is determined by the "Strobe Duration" value: when the "Strobe Duration" value is 0, the high level duration of the Strobe signal is equal to the exposure time; if the "Strobe Duration" value is not 0, The Strobe signal’s high continuity time is equal to the "Strobe Duration" value.

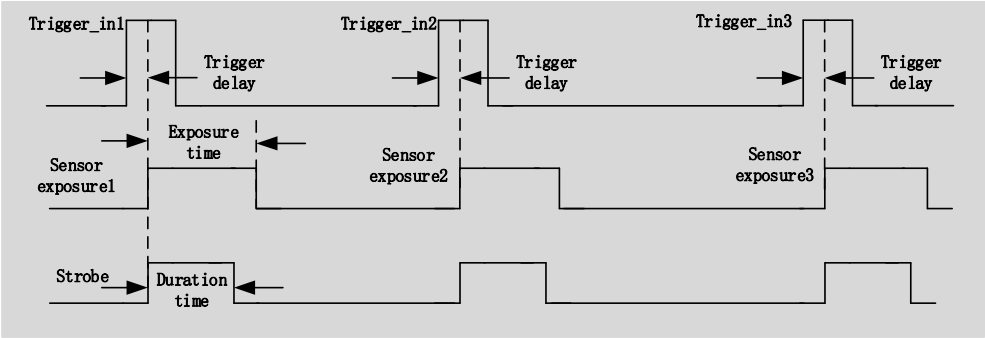


Figure 4-21 Strobe effective level duration

- Strobe output delay

The camera provides the feature of output delay to strobe signal to meet the special usage of users. When the exposure begins, the Strobe signal output does not immediately take effect, delayed according to the value set by "Strobe Delay Time". As shown in Figure 4-22.

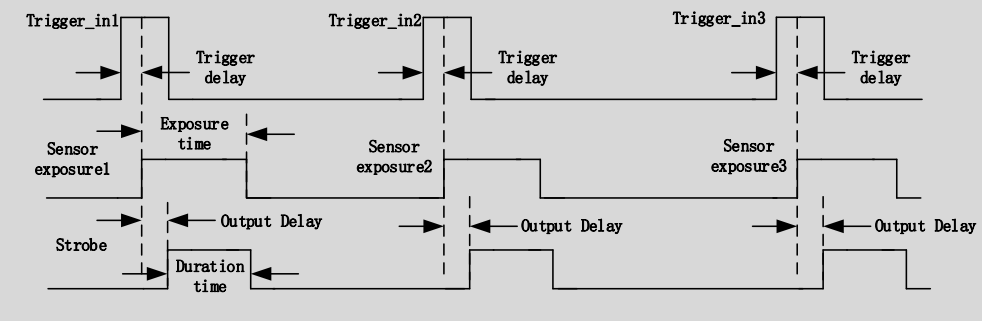


Figure 4-22 Strobe output delay

- Strobe Pre-output

The camera also provides a pre-output feature of the strobe signal, that is, the strobe signal takes effect earlier than the exposure begins.

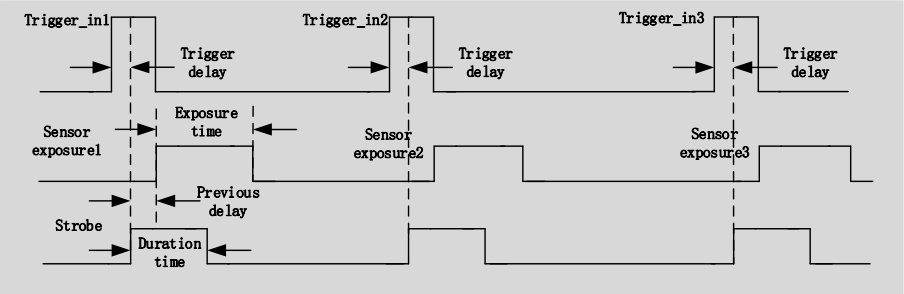


Figure 4-23 Strobe Pre-output

This feature can be applied to flash lamps with slow response. The pre-output time is set by "Strobe Delay Time". The timing is shown in Figure 4-23.

4.7.4 User Output

When choosing the "User Output" output mode, the user can enter a value after the "User Value" control to set the corresponding line output 0 or 1. The value here is only the low three bits of binary, for example when line1, line3 is set to the "User Output" output mode and the "User Value" is set to 4 (undefinedb100), then line3 outputs 1, line1 output 0. as shown in Figure 4-24

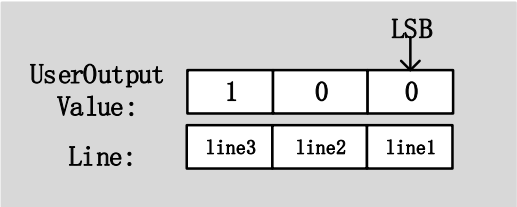


Figure 4-24 User-defined output schematic diagram

The following is the API code to set User Output:

```
nncam_IoControl(m_hCam, index, NNCAM_IOCONTROLTYPE_SET_OUTPUTMODE, 3, NULL);
nncam_IoControl(m_hCam, 0, NNCAM_IOCONTROLTYPE_SET_USERVALUE, val, NULL);
```

4.8 Camera control parameter configuration

4.8.1 Exposure time

The exposure time range is specified in the camera technical specifications section(Sec.**Fehler! Verweisquelle konnte nicht gefunden werden.**). Exposure time control supports manual control and automatic exposure control. When the camera is in trigger mode, the automatic control exposure feature will be disabled.

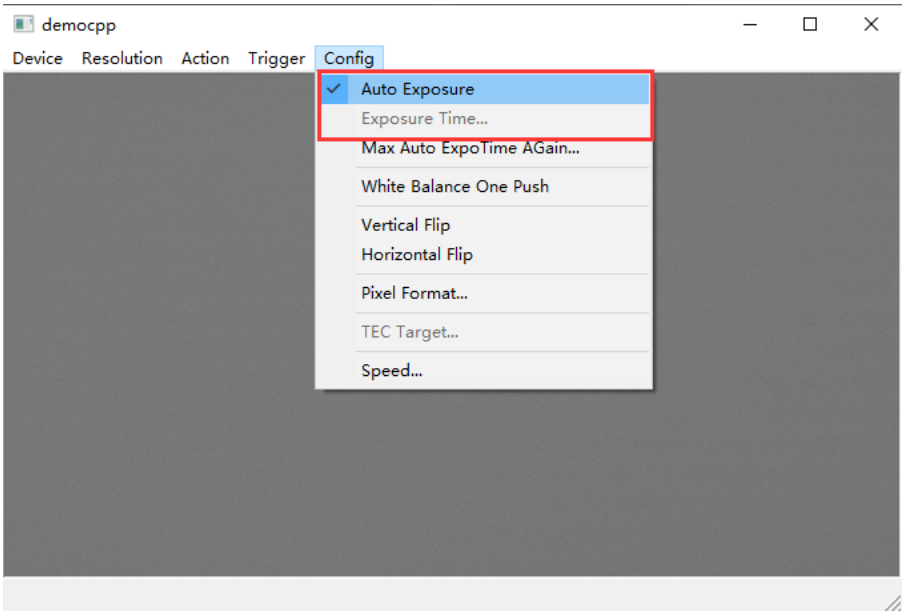


Figure 4-25 Exposure time setup

As shown in Figure 4-25, in **democpp**, click "Config" in the top control menu. Click "Auto Exposure" and the "✔" checkmark indicates that automatic control exposure mode is on. Click again to exit the current mode. Click "Exposure Time" menu and a dialog called Exposure Time will pop up and drag the slider in the new dialog to manually set the exposure time control.

Here is the API code for the setup the exposure time:

```
nncam_put_AutoExpoEnable(HnnCam h, int bAutoExposure);
nncam_put_AutoExpoTarget(HnnCam h, unsigned short Target);
nncam_put_ExpoTime(HnnCam h, unsigned Time); /* in microseconds */
```

4.8.2 Gain control

The gain value range is specified in the camera technical specifications section. When the gain increases, the image noise increases.

The following is the API code for the setup of the gain control:

```
nncam_put_ExpoAGain(HnnCam h, unsigned short AGain); /* percent */
```

4.8.3 White balance

White balance means that the camera performs color adjustment under different light source. The user can make the white area always white at different color temperatures by adjusting the “R”, “B” component’s gain on the image. Ideally, the ratio of R, G and B components in the white region is 1:1:1.

The white balance setting is shown in Figure 4-26. Click "Config" on the control menu at the top of democpp and click "White Balance One Push" to automatically balance white once.

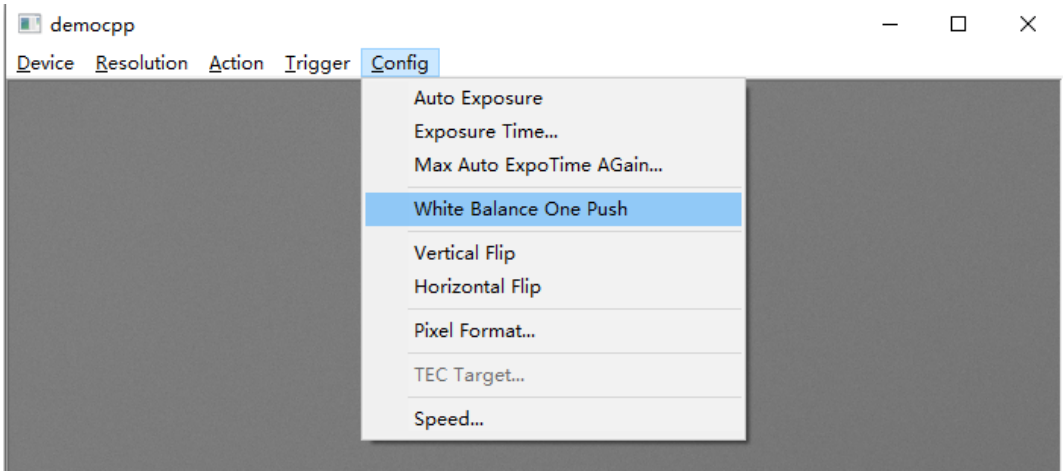


Figure 4-26 White balance setup

The following is the API code for the setup of the white balance one push:

```
//auto white balance "one push". This feature must be called AFTER nncam_StartXXXX
nncam_AwbOnePush(HnnCam h,PINNCAM_TEMPTINT_CALLBACK fnTTProc, void* pTTCtx);
```

4.8.4 Color adjustment

User can adjust hue, saturation, brightness, contrast and gamma value.

The following is the API code for the color adjustment:

```
nncam_put_Hue(HnnCam h, int Hue);
nncam_put_Saturation(HnnCam h, int Saturation);
nncam_put_Brightness(HnnCam h, int Brightness);
nncam_put_Contrast(HnnCam h, int Contrast);
nncam_put_Gamma(HnnCam h, int Gamma);    /* percent */
```

4.8.5 Image flip

As shown in Figure 4-27, in **democpp**, click "Config" in the control menu. Click "Vertical Flip" to flip the image vertically and "Horizontal Flip" to flip horizontally.

The following is the API code to flip the image:

```
nncam_put_VFlip(HnnCam h, int bVFlip); /* vertical flip */
nncam_put_HFlip(HnnCam h, int bHFlip); /* horizontal flip */
```

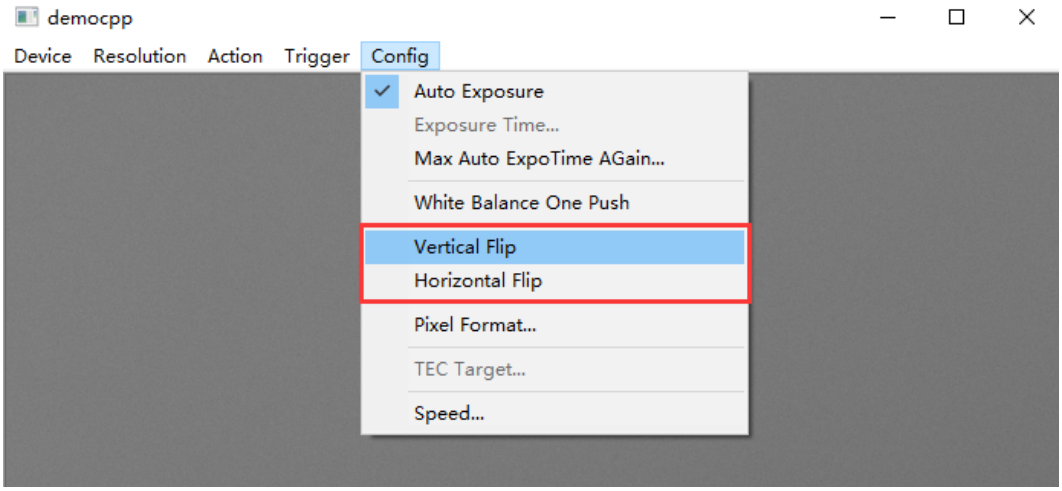


Figure 4-27 Image flip setup

4.8.6 Test pattern

In **democpp**, as shown in Figure 4-28, clicking "Action" in the control menu, then clicking “Test Pattern”. selecting "Normal" to display unaltered sensor captured image, "Test Pattern 1" a gray scale gradient oblique stripe showing the movement, "Test Pattern 2" a gray scale gradient vertical stripe showing the movement and "Test Pattern 3" a grayscale gradient horizontal stripe that shows the movement. The color camera can output a corresponding test pattern.

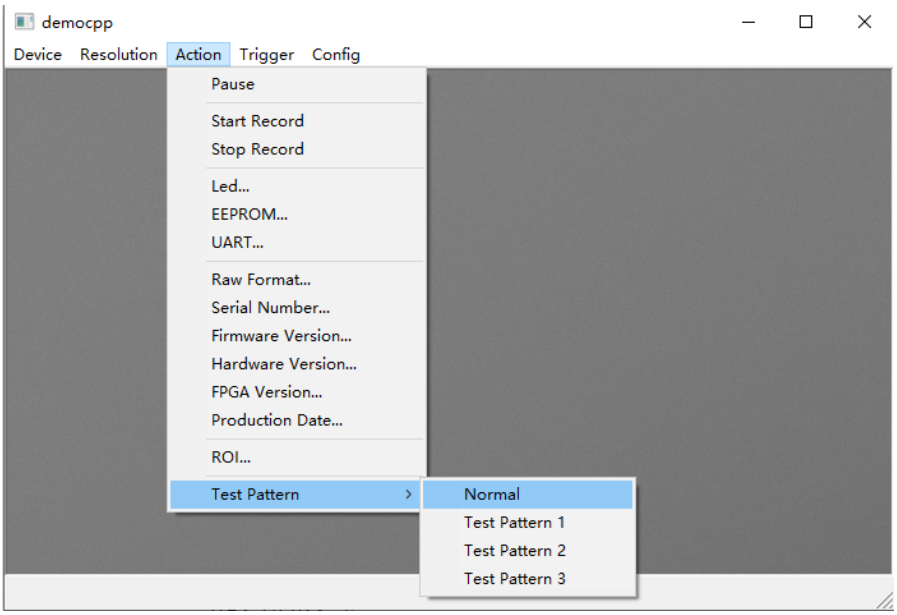


Figure 4-28 Set the test pattern

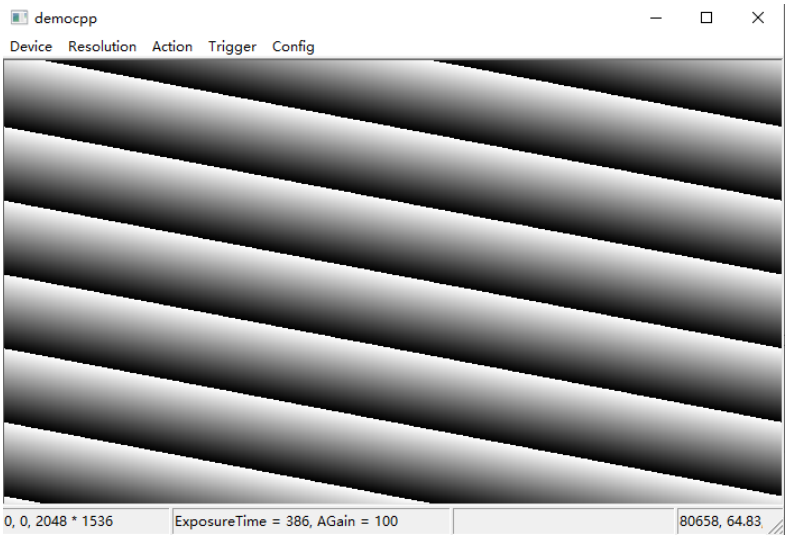


Figure 4-29 Grayscale gradient oblique stripes

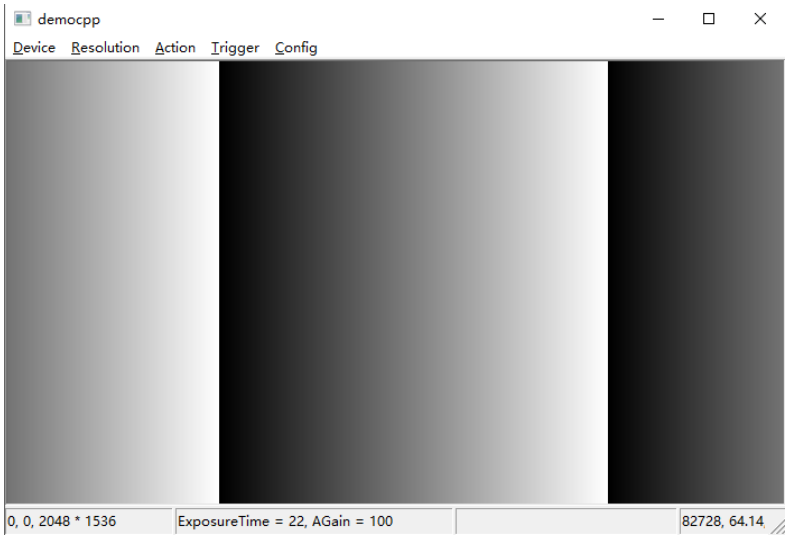


Figure 4-30 Grayscale gradient vertical stripes

Figure 4-31 Grayscale gradient horizontal stripes

The following is the API code for the setup of the test pattern:

// TestPattern: 0-> TestPattern Off , 3-> Moving Diagonal Gray Gradient , 5-> Moving Vertical Gray Gradient , 7-> Moving Horizontal Gray Gradient ,

9-> Moving Diagonal Chromatic Gradient

nncam_put_Option(m_hCam, NNCAM_OPTION_TESTPATTERN, val);

4.9 SC-ITR series camera's I/ O electrical properties

4.9.1 ITR series camera's opto-isolated input circuit (line0)

In the camera I/O control, opto-isolated input circuit is shown in 1.

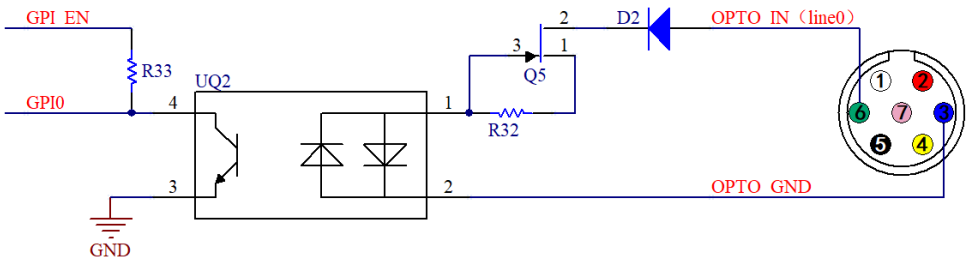


Figure -32 Opto-isolated input circuit

Logic 0 input level: 0~2.2VDC (OPTO_IN pin)

Logic 1 input level: 3.3~24VDC (OPTO_IN pin)

Maximum input current: 30mA

The input level is between 2.2V and 3.2V, the circuit action state is uncertain, please avoid the input voltage working in this range.

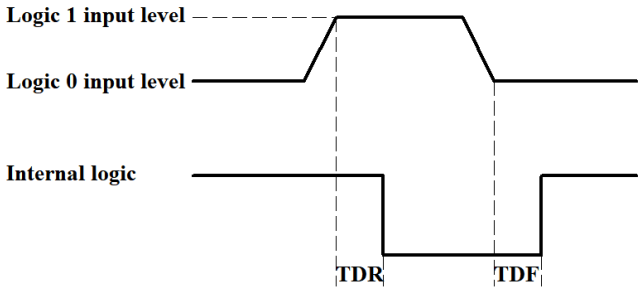


Figure -33 Input logic level

Input rise delay (TDR): 6us

Input drop delay (TDF): 6us

4.9.2 ITR series camera's opto-isolated output circuit (line1)

In camera I/O control, opto-isolated output circuit is shown in 3.

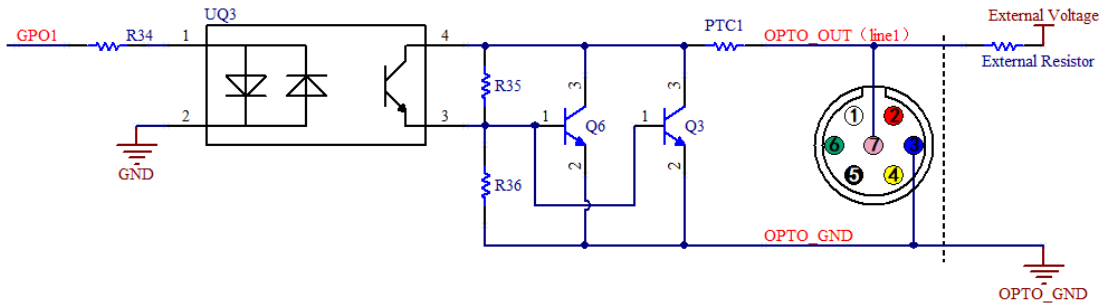


Figure -34 Opto-isolated output circuit

Opto-isolated output maximum current: 30mA

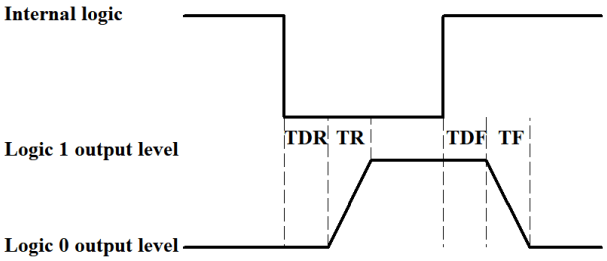


Figure -35 Output logic level

The electrical characteristics of the opto-isolated output signal (external voltage 5V, external resistor 1K) are shown in Table -1.

Parameter name	Parameter symbol	Parameter values
Output logic low level	VL	742mV
Output logic high	VH	4.134V
output rise time	TR	4us
Output downtime	TF	1.8us
Output rising delay	TDR	12us
Output drop delay	TDF	2us

Table -1 Opto-isolated output signal's electrical characteristics

The corresponding current and output logic low level parameters are shown in Table -2 when different voltage and resistors are used in external circuit.

External voltage	Non-essential resistance	VL	Output current
3.3V	1KΩ	510mV	2.82mA
5V	1KΩ	742mV	4.31mA
12V	2.4KΩ	795mV	4.68mA
24V	4.7KΩ	850mV	4.97mA

Table -2 Opto-isolated output logic's low level parameters

4.9.3 ITR series camera's Input and output I/O circuit (line2/line3)

Non-isolated configurable input, output I/O circuit is shown in, Figure -37.

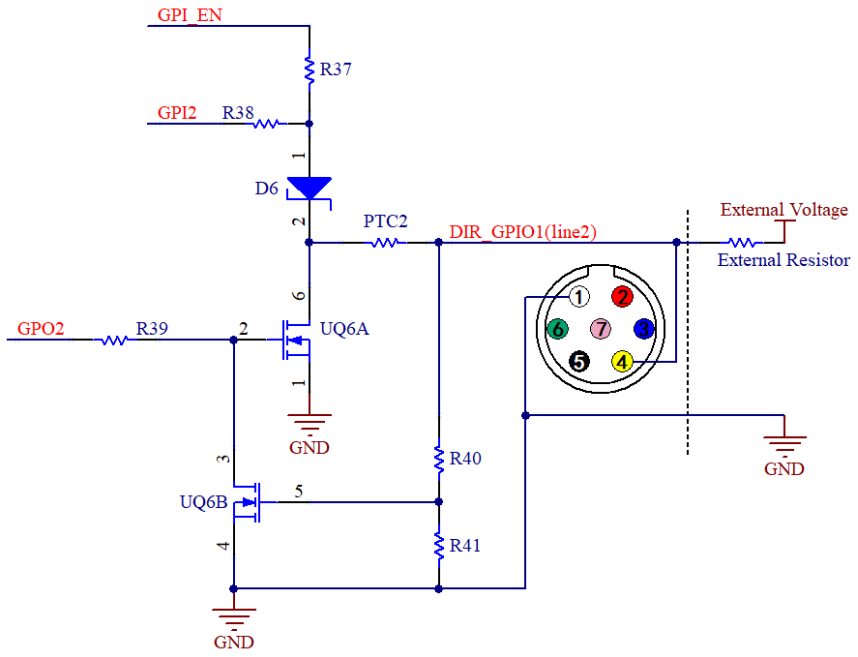


Figure -36 Non-isolated configurable input, output I/ O circuit (line2)

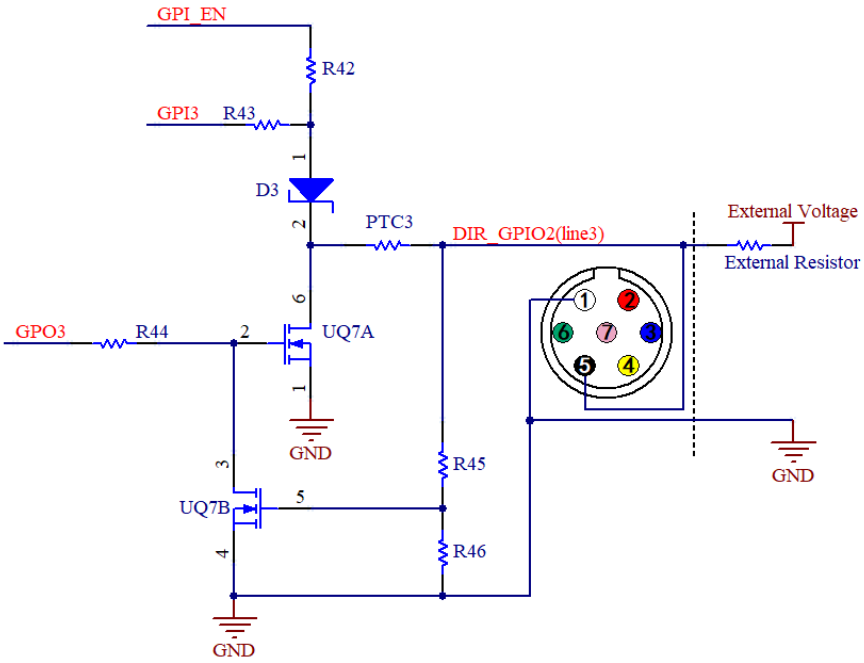


Figure -37 Non-isolated configurable input, output I/ O circuit (line3)

1, Line2/line3 set as input pin:

Logic 0 input level: 0-0.6 VDC (DIR_GPIO1/DIR_GPIO2 pin)

Logic 1 input level: 2.0~24VDC (DIR_GPIO1/DIR_GPIO2 pin)

Maximum input current: 25mA

The input level is between 0.6V and 2.0V, the circuit action state is uncertain. Please avoid the input voltage working in this range.

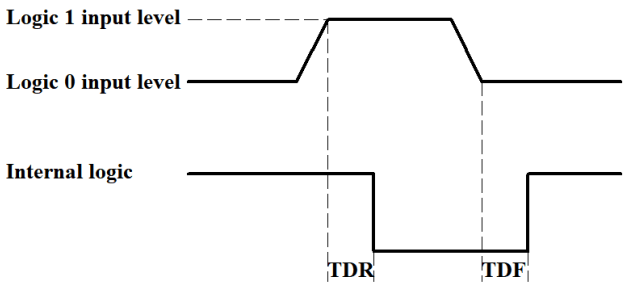


Figure -38 Input logic level

To prevent damage to the GPIO pin, connect the GND pin before entering voltage to the Line2 pin.

Input rise delay (TDR): 0.02us

Input drop delay (TDF): 0.02us

2, Line2/line3 set as output pin

The maximum current allowed through this pin is 25 mA.

When the ambient temperature is 25 degrees Celsius, the relationships between the external voltage, resistance and output low level are shown in Table -3.

External voltage	Non-essential resistance	VL (GPIO)
3.3V	1KΩ	0.11V
5V	1KΩ	0.167V
12V	2.4KΩ	0.184V
24V	4.7KΩ	0.385V

Table -3 Non-isolated output logic's low level parameters

The external pull-up voltage 5V pull-up resistance 1KΩ, GPIO output logic level, electrical characteristics are shown in 3.

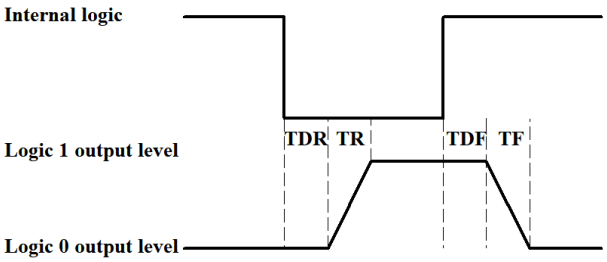


Figure -39 Output logic level

Parameter name	Parameter symbol	Parameter values
Output rise time	TR	0.08us
Output downtime	TF	0.02us
Output rising delay	TDR	0.1us
Output drop delay	TDF	0.04us

Table -4 Non-isolated output's electrical characteristics

5.0 Support

You can also get support in the following ways:

- Web site support: www.ehd.de for relevant documentation and on-line technical support.
- E-mail support: info@ehdimaging.de
- Other support: please contact the sales representative.

EHD imaging GmbH
Zum Rennplatz 15
D-49401 Damme (Germany)
www.ehd.de