*Sparrow*

# IEEE 1394b Industrial CCD Camera

# IMS-130

# User's Manual

Manual Version: 1.2

Release Date: July 15, 2010

# Table of Contents

# 1. Introduction

Sparrow; a new IEEE 1394 camera product line of ICP DAS--designed for industrial machine vision applications. Sparrow IMS camera are designed within high sensitivity industrial CCD sensor and be able to acquire color image or monochrome image. The IMS-130 offers 1288 x 964 resolutions and acquisition speed up to 31 frames per second.

Sparrow camera also offers industrial digital I/O and screw locked 1394b connector interface designed. Compact housing with industrial level design can be adopted into all kind of inspection system such as inspection by robot arms. Sparrow IMS camera image applications can be implement on industry environments such as semiconductor, electric component, high-speed assembly and manufacture quality control

## 1.1  Features

- Progressive-scan for on-the-fly applications
- VGA to up 1.3 mega-pixel high resolution
- Small and exquisite compact industrial housing design
- Built-in 32MB memory buffer
- 64KB data flash and AES project encryption
- Robust external trigger I/O interface supported
- Free SDK API for VC, VB, BCB and VC.Net 2005
- Supports Windows XP driver only

## 1.2  Applications

- Electronic component inspection
- Alignment AOI system
- Manufacturing quality control
- Food and beverage inspection
- Factory automation inspection

## 1.3  System Requirement

To ensure seamless operation, ICP DAS recommends that your system meets the minimum requirements below:

- Platform: Pentium III 800MHz CPU, 256MB DDRAM or above.
- VGA display: AGP 4X or above.
- Display setting: 800 x 600 resolution or above.

> ➤ **32-bit OS only:**

**If using Windows XP SP2 or above version, please rollback to Service Pack 1.**

● **The 1394b full spec. driver may only supported in Windows XP SP1, therefore ICPDAS has offer 'fixSP2g' program to help customer to roll-back the 1394b driver into Windows XP SP2/SP3 or above system (Please refer Chapter 3.2.3 Windows XP SP1 rollback).**

\*\*Please refer 1.6 Benchmark for system limitation information.\*\*

## 1.4 Function Descriptions

To ensure proper implementation, please carefully review the, limitation parameters and formula calculations, listed below.**AOI (Area of Interest)**

The AOI (Area of Interest) function allows users to select an area of interest, for the camera's CCD array to specifically read, display, and transmit.

Sparrow IMS camera offers include Mono 8-bit, Mono 16-bit and RGB 24-bit in total 3 color formats and each color format will generate difference data structure.

- Mono 8-bit: It's black and white image data and grayscale present in 8-bit depth, each pixel size is one byte.
- Mono 16-bit: It's black and white image data and grayscale present in 16-bit depth, each pixel size is two bytes.
- RGB 24-bit: its color image and construction by red, green and blue individually color data, each pixel size is three bytes.
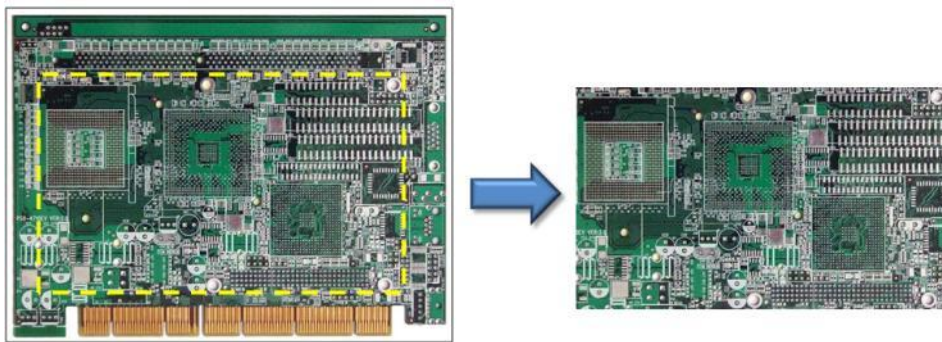


Figure 1-1 Color AOI Acquisition
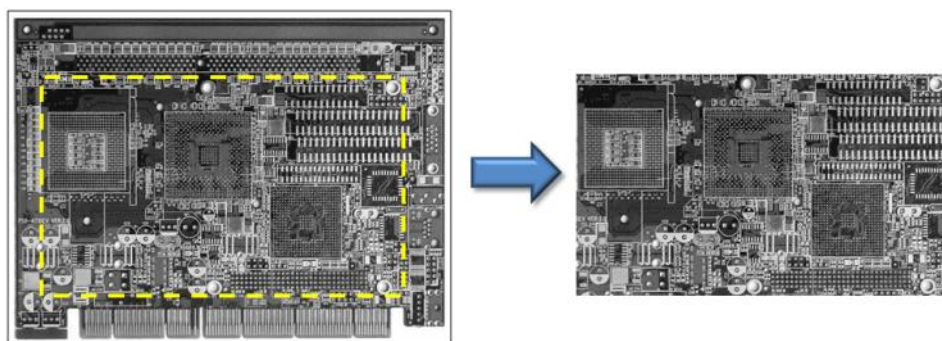


Figure 1-2 Mono AOI Acquisition

Before using AOI function, please make sure AOI mode is selected in Video Mode. NO activated if AOI is not selected.

The AOI function will required the values of Left, Top, Width and Height to specify AOI region. Due the AOI image data structure, so user must to follow the possible values of AOI definition as below.

Full Image

(0,0)

x: AOI Top

(x,y)

AOI Height

AOI Width

y: AOI Left

IMS-30 (648, 488)
IMS-130 (1288, 964)
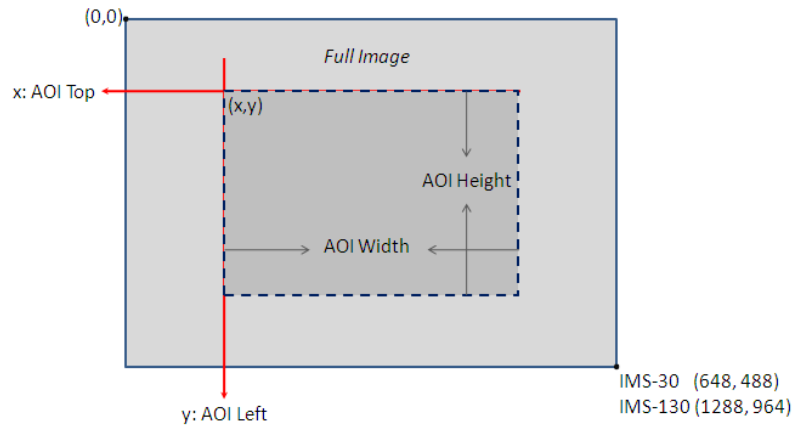
- AOI Left: The value must be 2*N, when N is an integer
- AOI Top: The value must be 2*N, when N is an integer
- AOI Width: The value must be 8*N, when N is an integer
- AOI Height: The value must be 2*N, when N is an integer
- AOI Left + AOI Width must be under the total width of Full Image
- AOI Top + AOI Height must be under the total height of Full Image

Also, frame rate can be changed in AOI function. Frame rate is automatically adjusted into range of camera if frame rate setting over range of camera.

## 1.4.2 Brightness

All image data luminance will be changed by adjusting Brightness. Sparrow IMS camera offers adjustable function of Brightness and absolute value range from 0.0 ~ 6.2439%. Default value of Brightness is 3.997803.



0                                    6.2439

## 1.4.3 Sharpness

All image data contrast will be changed by adjusting Sharpness. Sparrow IMS camera offers adjustable function of Sharpness and range from 0 ~ 4095. Default value of Sharpness is 2048.

## 1.4.4 White Balance

Sparrow IMS CCD camera camera are using the Bayer tiling in the CCD for the color pixel data. White Balance function is able to work well on color correction to deal with differing lighting conditions.

The White Balance scheme outlined in the IIDC specification states that blue and red are adjustable but the green is not. White Balance function will offer gain value of red and blue for adjusting the color balance of the camera and range from 0 ~ 1024. Default value of gain value of red of White Balance is 600, while blue is 840.

## 1.4.5 Hue

All color image space can be changed by adjusting Hue function. Sparrow IMS camera offers adjustable function of Hue and absolute value range from -180.0 ~ 179.912, and default value is 0.



## 1.4.6 Saturation

All image date color purity can be changed by adjusting Saturation function. Sparrow IMS camera offers adjustable function of Saturation and absolute value range from 0.0 ~ 399.902, and the default value is 136.9141.

### 1.4.7 Gamma and LUT

Gamma and LUT (Lookup Table) function will be changes image data converted structure. Once change gamma value then the lookup table data structure will be change immediately. Sparrow IMS offers adjustable function of gamma and absolute value range from 0.50 ~ 3.99902, and the default value is 0.999.234.

Sparrow IMS camera support three 11-bit input lookup table that produce 9-bit outputs. Three channels can be individually for red, green and blue color data output, or equalized value for monochrome color format.

It can be used to apply a non-linear mapping of the image result. User can used this function to get Lookup Table file of the camera. This file is standard text file format and user can open or modify Lookup Table content very easily.

By default, when gamma value =1.0, which yields a linear response then gamma functionality will be OFF and failed to change the Lookup Table content.

```
[ICPDAS IMS-130]
[Channel: 3    BitDepth: 9    Entries: 2048]
0,0,0
0,0,0
0,0,0
1,1,1
1,1,1
1,1,1
1,1,1
2,2,2
2,2,2
2,2,2
2,2,2
3,3,3
3,3,3
3,3,3
3,3,3
4,4,4
4,4,4
4,4,4
4,4,4
5,5,5
5,5,5
5,5,5
5,5,5
6,6,6
.
.
.
506,506,506
506,506,506
506,506,506
506,506,506
507,507,507
507,507,507
507,507,507
507,507,507
508,508,508
508,508,508
508,508,508
508,508,508
509,509,509
509,509,509
509,509,509
509,509,509
510,510,510
510,510,510
510,510,510
510,510,510
511,511,511
511,511,511
```

### 1.4.8 Gain

Gain function will be changes over all image data, and its adjustable level of amplification of the camera signal. Sparrow IMS camera offers adjustable function of Gain and absolute value range from -6.26386 ~ 24.0001. However when gain is increased in an image, one must also take into consideration the addition of significant noise. Therefore keep the Gain value in 0 is strong recommend by default.

### 1.4.9 Shutter (Exposure)

Shutter function is adjustable level of shutter speed for the exposure of image integration time. However the camera shutter integration time must be less than the frame rate. Therefore the Sparrow IMS camera offers adjustable function of Shutter speed and range from 4.65us ~ 133ms.

## 1.4.10    Trigger Control

Sparrow IMS camera offers two Trigger control mode, included Standard Trigger Mode and Bulb Shutter Mode. The trigger control resource could be come from hardware or software.

Sparrow IMS camera designs the pin 1 of GPIO interface which dedicated for physical external hardware trigger resource. It is opto-isolated open collector circuit designed (For detail schematic, please refer 1.4.11) and input voltage range from +3.3VDC to +30VDC in maximum input 8.3mA.

➢ **Standard Trigger Mode**

The Standard Trigger Mode offers the control resource can be come from hardware or software. If user selects the trigger resource from software, and user be able to use camera acquire image by programming software trigger.

The trigger signal can be chosen by rising edge or falling edge when user selected the trigger resource from hardware.
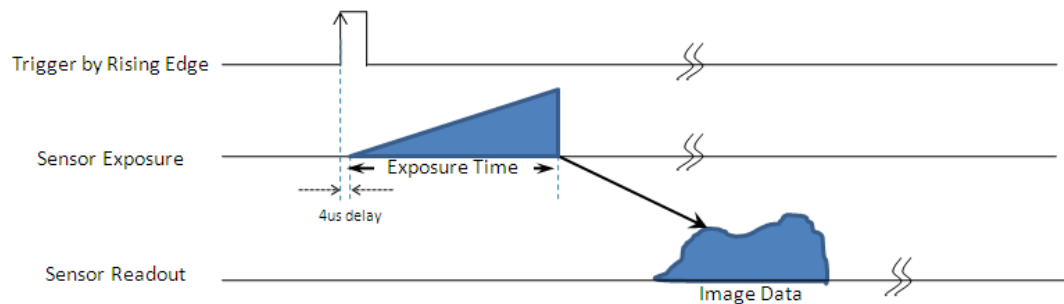


Figure 1.4.10.1 Standard Trigger by Rising Edge
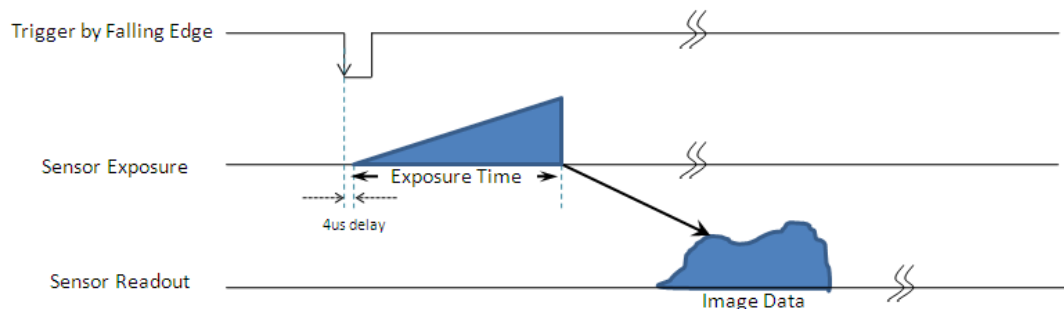


Figure 1.4.10.2 Standard Trigger by Falling Edge

➢ **Bulb Shutter Mode**

The Bulb Shutter Mode offers the control resource can be come from hardware or software. If user selected the trigger resource from software, and then user is able to use camera acquire image by programming software trigger.

The camera exposure time will be base on the trigger signal duration, and the trigger signal can be choose by rising edge or falling edge when user selects the trigger resource from hardware.
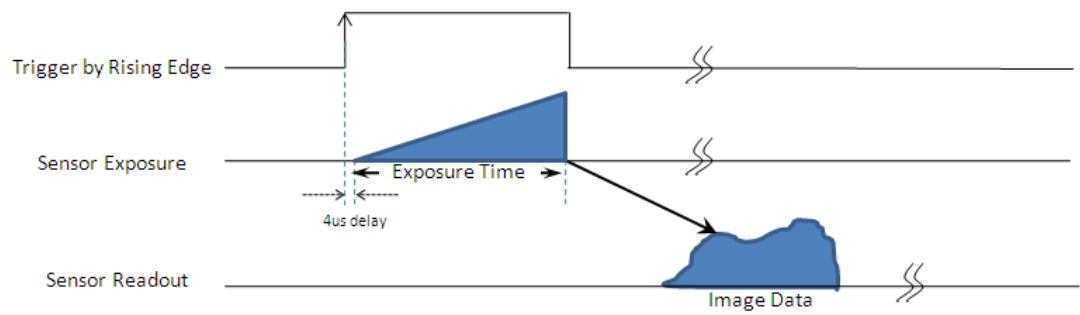
Figure 1.4.10.3 Bulb Shutter Trigger by Rising Edge
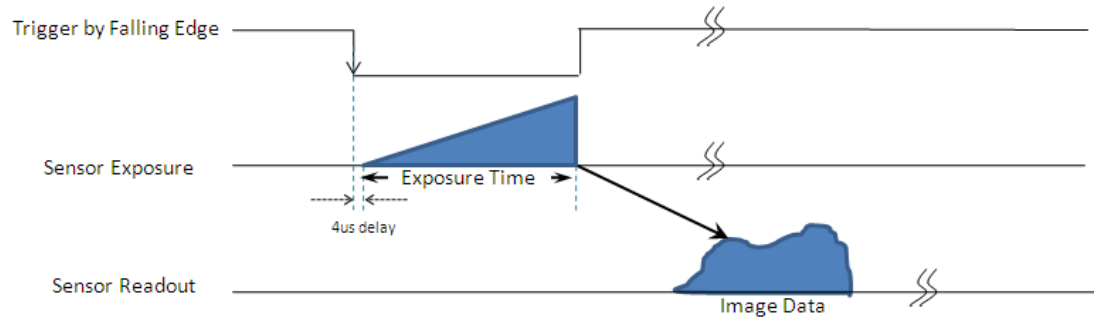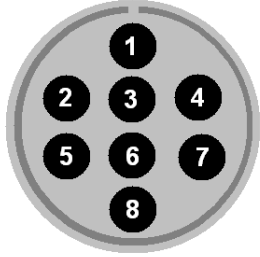


Figure 1.4.10.4 Bulb Shutter Trigger by Falling Edge

## 1.4.11  GPIO (General Purpose Digital Input and Output)

The Sparrow IMS camera offers the GPIO interface and pin define as below.

| 8-Pin Hirose GPIO interface | Pin No. | Define | Description |
|---|---|---|---|
|  | 1 | GPIO0 | Opto-isolated input, Default for Trigger Input (+3.3V to +30V), Input delay time:4us |
| | 2 | GPIO1 | Opto-isolated output, Default for Strobe Output, Drive strength:25mA at 30V |
| | 3 | GPIO2 | Bi-directional Input / Output |
| | 4 | GPIO3 | Bi-directional Input / Output |
| | 5 | IGND | Ground pin for bi-direction IO, Vext, +3.3V |
| | 6 | GND | Ground pin for Opto-isolated IO pins |
| | 7 | VEXT | Allows the camera to be powered externally |
| | 8 | +3.3V | Power external devices up to 150mA |

### 1.1.1.1 GPIO0 (Opto-Isolated Input) Circuit

The figure below shows the schematic for the opto-isolated input circuit.
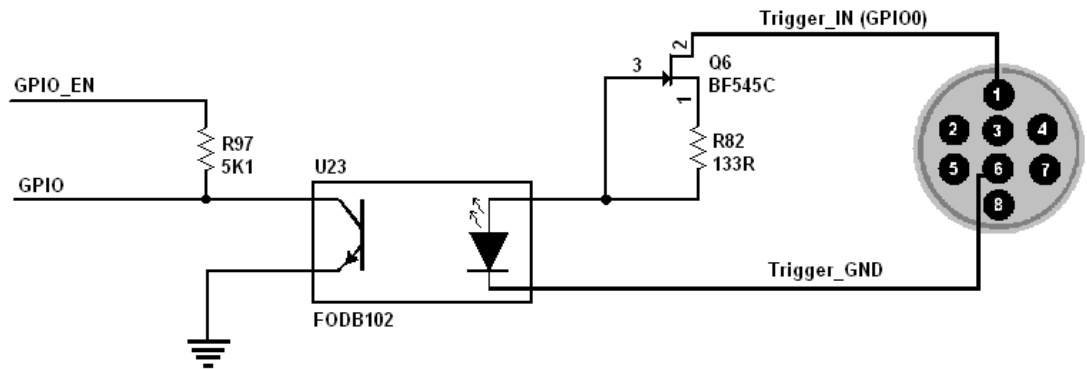


- Logical o input voltage: 0VDC to +1DC (voltage at OPTO_IN)
- Logical 1 input voltage: +1.5VDC to +30VDC(voltage at OPTO_IN)
- Maximum input current:8.3mA
- Behavior between 1VDC and 1.5VDC is undefined and input voltages between those values should be avoided.
- Input delay time:4us

## 1.1.1.2 GPIO1 (Opto-Isolated Output) Circuit

The figure below shows the schematic for the opto-isolated output circuit. The maximum current allowed the opto-isolated output circuit is 25mA (limit by PTC resistor), and the output impedance is 140 Ohm.
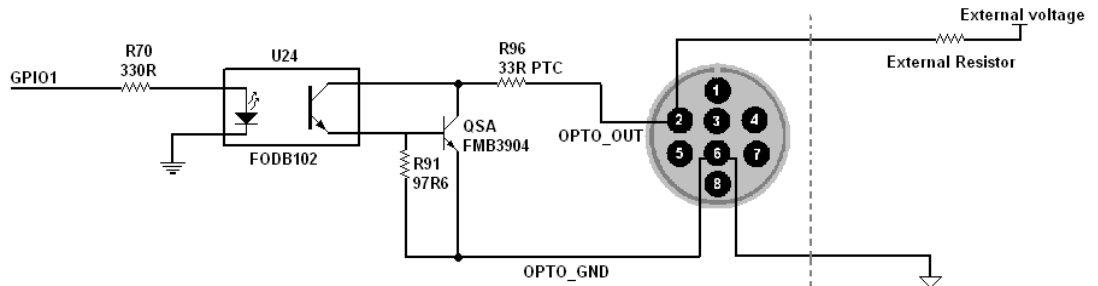


■    **Response Time**

The following table lists the switching times for the opto-isolator in the output pin, assuming an output VCC of 5V and 1K Ohm resistor.

| Parameter | Value |
|---|---|
| Delay Time | 9us |
| Rise Time | 16.8us |
| Storage Time | 0.52us |
| Fall Time | 2.92us |

■    **Sample voltage / Resistor combinations**

The following table lists several external voltage and resistor combinations that have been tested to work with the GPIO1 opto-isolated output.

| External Voltage | External Resistor | OPTO_OUT Voltage | OPTO_OUT Current | Short Current |
|---|---|---|---|---|
| 3.3V | 1K Ohm | 0.56V | 2.7mA | 43mA |
| 5V | 1K Ohm | 0.84V | 4.2mA | 47mA |
| 12V | 2.4K Ohm | 0.91V | 4.6mA | 21.5mA |
| 24V | 4.7K Ohm | 1.07V | 5.1mA | 13.5mA |
| 30V | 4.7K Ohm | 1.51V | 13.3mA | 6.1mA |

### 1.1.1.1 GPIO2 and GPIO3 (Bi-Direction) Input / Output Circuit

The figure below shows the schematic for the GPIO2 and GPIO3 bi-directional input or output circuit.
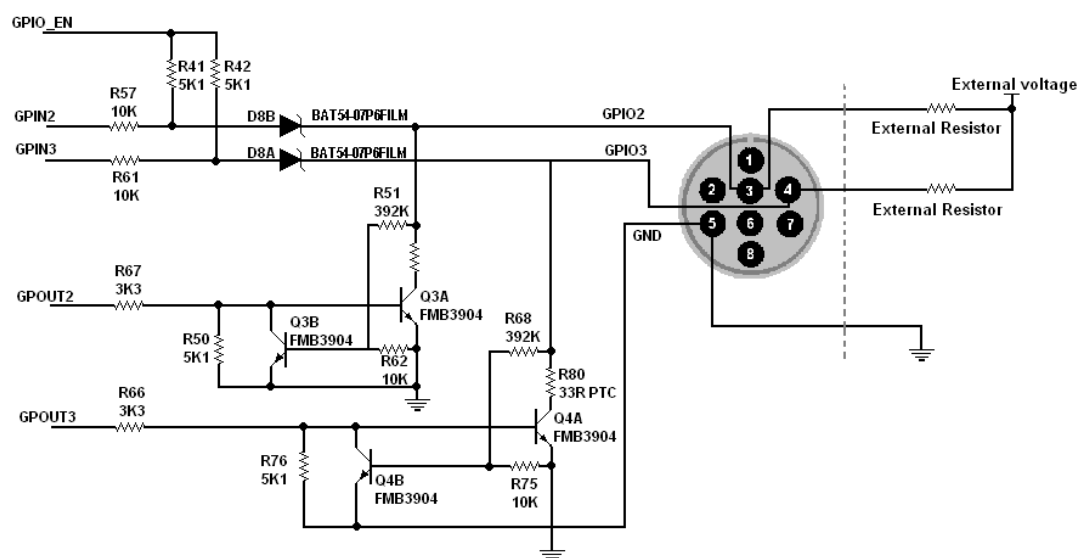


- ■ **Input Side**
  - ➢ Logic 0 input voltage: 0VDC TO +0.5VDC (voltage at GPIO2/GPIO3)
  - ➢ Logic 1 input voltage: +1.5VDC to +30VDC (voltage at GPIO2/GPIO3)
  - ➢ Behavior between 0.5VDC and 1.5VDC is undefined and input voltages between values should be avoided
  - ➢ To avoid damage, connect the GND pin first before applying voltage to the GPIO line.

- ■ **Output Side**

The maximum output current through the bi-directional circuit is 25mA (limit by PTC resistor) and the output impedance is 40 Ohm. The following table lists several external voltage and resistor combinations that have been tested to work with the bi-directional GPIO when configure as output.

| External Voltage | External Resistor | GPIO2/3 Voltage | Short Current |
|---|---|---|---|
| 3.3V | 1K Ohm | 0.157V | 62mA |
| 5V | 1K Ohm | 0.218V | 45mA |
| 12V | 1K Ohm | 0.46V | 23mA |
| 24V | 1K Ohm | 0.86V | 7.5mA |
| 30V | 1K Ohm | 0.966V | 0.1mA |

The following table lists the switching times for a standard GPIO pin, assuming an output VCC of 5V and 1K Ohm resistor.

| Parameter | Value |
|---|---|
| Delay Time | 0.28us |
| Rise Time | 0.06us |
| Storage Time | 0.03us |
| Fall Time | 0.016us |

### 1.4.12 Strobe Control

This feature allows users to enable and parameterize up to three strobe light control output signals. The signals are designated as Strobe 1, Strobe 2, and Strobe3. Each strobe signal can be set to active high or active low by logical value, please refer to section 6, "Function Library", for command definitions.

The strobe is determined by a combination of two values; Delay Value and Strobe Duration-- both range from 0 sec. ~ 10 sec.

The Strobe delay will determine the time between the start of image exposure and when the strobe signal changes state as show in Figure 1-13.
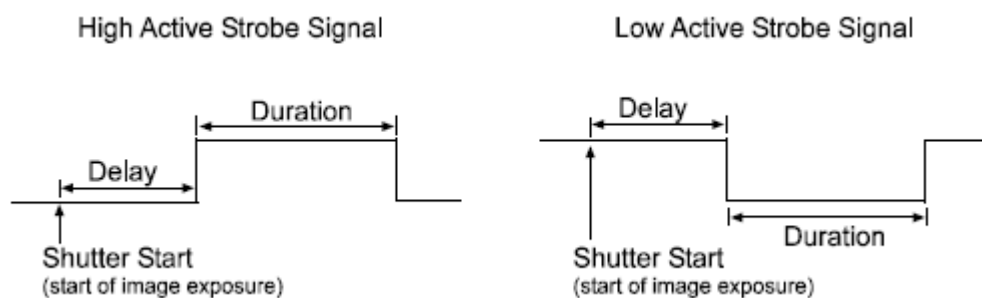


**Figure 1-13: Strobe Signal**

### 1.4.13 Save Camera Configuration

Sparrow IMS camera provides a function on save camera configuration to help customer to setting and maintain camera parameters easily. User can use EzView Pro Utility to real-time adjust camera parameters and save the configure setting into the camera or local computer. The configure file can be upload to any Sparrow IMS camera.

### 1.4.14 Data Flash Control

Sparrow IMS camera offers 64K bytes Data Flash memory for customer to save or restore camera settings and AES project encryption register code. The address of data flash can be read and write, range from 0x0 ~ 0xFFFF. Keep the camera power sustained when written data into the Data Flash; otherwise the data may loose due the camera power off.

### 1.4.15    Software AES Project Encryption

Sparrow IMS camera offer an software AES project encryption function. The system designer can use this function to generated unique security key to protected the copy right of their system. The IMS_Get_CameraInform function can get the unique hardware information of the camera, and call AES_GET_REG function to set 16bytes AES key then will get 16bytes AES project encryption register code.
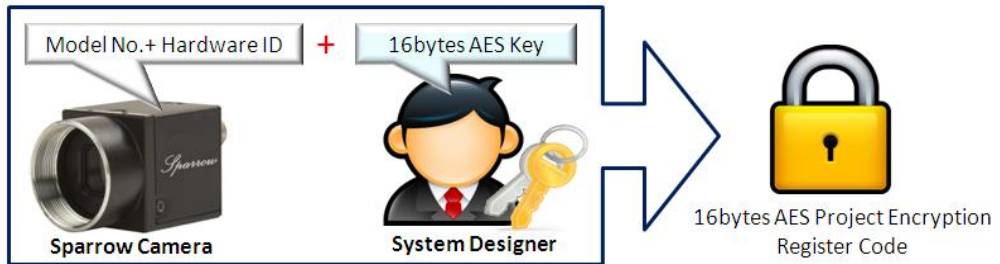


Normally the AES project encryption register code can be save into any device as customer need. Also user can write the register code into the Data Flash of the camera via IMS_Write_DataFlash function. The system designer can use IMS_Read_DataFlash to get this code to check whole system condition before system operation.

### 1.4.16    Test Pattern

Sparrow IMS cameras offer an internal generated test pattern for testing camera transmission. The test pattern will show a gray diamond pattern image when user enable the Test Pattern function and start acquired images.
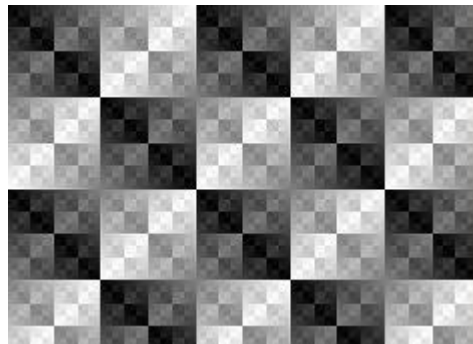
# 1.5  Spectral Response

Below is the spectral response curve of Sparrow IMS camera CCD sensor and excludes the Lens and lighting source characteristics.

IMS-130 Monochrome Mode

Spectral Sensitivity
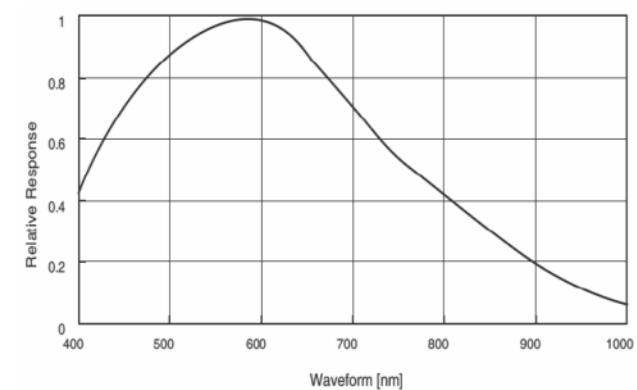


IMS-130 Color mode

Spectral Sensitivity
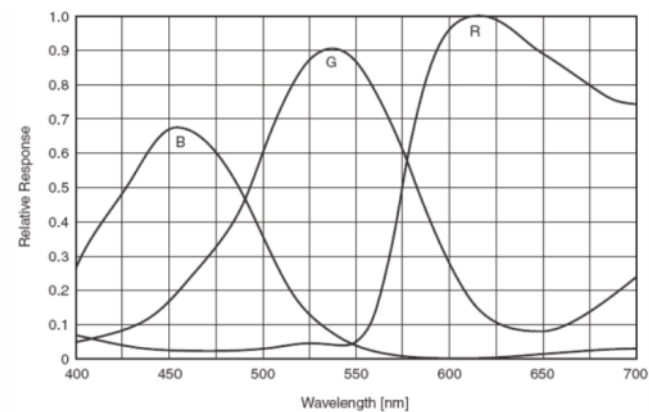
## 1.6 Benchmarks

Sparrow IMS camera is able to connect to IEEE 1394a (400Mbps) or IEEE 1394b (800Mbps) IEEE interface port. The bandwidth of IEEE interface port may decide the frame rate of camera acquisition, and also the sharing connected structure may reduce camera acquisition speed.

The following benchmark information is based on the IEEE 1394 port bandwidth condition and user may get the reference for system performance consideration.

### 1.6.1 Standard Video Mode

| Resolution | Video Mode | 1394a | 1394b |
|---|---|:---:|:---:|
| 640 x 480 | MONO8 - 30FPS | ○ | ○ |
| | MONO8 - 15FPS | ○ | ○ |
| | MONO8 - 7.5FPS | ○ | ○ |
| | MONO16 - 30FPS | ○ | ○ |
| | MONO16 - 15FPS | ○ | ○ |
| | MONO16 - 7.5FPS | ○ | ○ |
| | RGB - 30FPS | ○ | ○ |
| | RGB - 15FPS | ○ | ○ |
| | RGB - 7.5FPS | ○ | ○ |
| 800 x 600 | MONO8 - 30FPS | ○ | ○ |
| | MONO8 - 15FPS | ○ | ○ |
| | MONO8 - 7.5FPS | ○ | ○ |
| | MONO16 - 30FPS | ○ | ○ |
| | MONO16 - 15FPS | ○ | ○ |
| | MONO16 - 7.5FPS | ○ | ○ |
| | RGB - 30FPS | ✕ | ○ |
| | RGB - 15FPS | ○ | ○ |
| | RGB - 7.5FPS | ○ | ○ |
| 1024 x 768 | MONO8 - 30FPS | ○ | ○ |
| | MONO8 - 15FPS | ○ | ○ |
| | MONO8 - 7.5FPS | ○ | ○ |
| | MONO16 - 30FPS | ✕ | ○ |
| | MONO16 - 15FPS | ○ | ○ |
| | MONO16 - 7.5FPS | ○ | ○ |
| | RGB - 15FPS | ✕ | ○ |
| | RGB - 7.5FPS | ○ | ○ |

| Resolution | Video Mode | 1394a | 1394b |
|---|---|---|---|
| 1280 x 960 | MONO8 - 30FPS | ✕ | ○ |
| | MONO8 - 15FPS | ○ | ○ |
| | MONO8 - 7.5FPS | ○ | ○ |
| | MONO16 - 30FPS | ✕ | ✕ |
| | MONO16 - 15FPS | ✕ | ○ |
| | MONO16 - 7.5FPS | ○ | ○ |
| | RGB - 15FPS | ✕ | ○ |
| | RGB - 7.5FPS | ○ | ○ |

## 1.6.2 AOI Video Mode

When set the camera Video Mode in VM_AOI, the real acquisition speed will base on the IEEE 1394 bandwidth, video format and resolution for the frame rate result. Normally IEEE 1394a host chipset bandwidth can reach about 30MB/s, and IEEE 1394b host chipset can reach bandwidth around 60MB/s.

By using more Sparrow cameras in the system, user must consider if here is enough bandwidth of IEEE 1394, or choose multiple host chipsets interface card for it.

| IEEE 1394 Bandwidth | 1394a S400 | 1394b S800 |
|---|---|---|
| Model No. | IMS-130 | IMS-130 |
| 640 x 480_Mono8 | 52fps | 52fps |
| 640 x 480_Mono16 | 50fps | 52fps |
| 640 x 480_RGB | 34fps | 52fps |
| 800 x 600_Mono8 | 44fps | 44fps |
| 800 x 600_Mono16 | 32fps | 44fps |
| 800 x 600_RGB | 22fps | 36fps |
| 1024 x 768_Mono8 | 36fps | 36fps |
| 1024 x 768_Mono16 | 20fps | 36fps |
| 1024 x 768_RGB | 13fps | 24fps |
| 1280 x 960_Mono8 | 26fps | 30fps |
| 1280 x 960_Mono16 | 13fps | 31fps |
| 1280 x 960_RGB | 8.5fps | 18fps |

## 1.6.3 ICP DAS IEEE 1394 products

| Model No. | IEEE 1394 Bandwidth |
|---|---|
| VISION BOX Series (VB-115/VB-216) | 400Mbps x 2 ports |
| IOI-4601-21 2-CH IEEE 1394a interface card | 400Mbps x 2 ports |

# 2. Hardware Reference

## 2.1 Sparrow IMS-130

### 2.1.1 Camera Specification

| | |
|---|---|
| Model No. | IMS-130 |
| Sensor Size | 1/3" Sony CCD |
| Pixel Size | 3.75um x 3.75um |
| Resolution | 1288 x 964 |
| Max. Frame Rate at 1394b S800 | Mono mode up to 31fps<br>Color mode up to 16fps |
| A/D Converter | 12-bit Analog to Digital converter |
| Image Data Format | Mono 8-bit, Mono 16-bit, RGB 24-bit |
| Data Flash | 64K bytes |
| Interfaces | 9-pin IEEE 1394b for camera control and video data transmission,<br>8-pin Hirose for four general purpose digital input and output supported |
| I/O | Opto-isolated Trigger Input x 1, Opto-isolated Output x 1<br>Bi-directional TTL GPIO x 2 |
| Power Consumption | 2.5W, max at 12V |
| S/N Ratio | 59dB |
| Dimensions | 29 x 29 x 30mm(without Lens) |
| Mass | 58g (without Lens) |
| Lens Mount | C-Mount |
| 1394 Compliance | IIDC 1394 based digital camera specification V1.31 |
| Operating Temperature | 0°C ~45°C |
| Storage Temperature | -30°C ~60°C |
| Operating Humidity | 20% ~ 80% (no condensation) |
| Storage Humidity | 20% ~ 95% (no condensation) |
| Certificated | CE, FCC |

## 2.1.2   Camera Interface
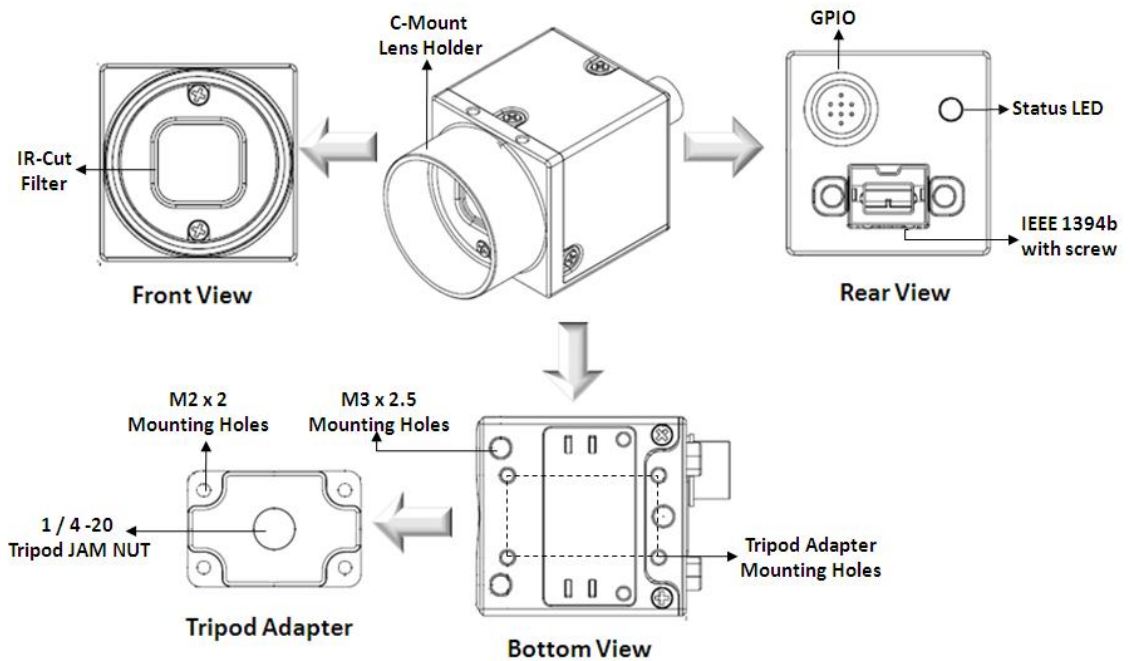


## 2.1.3   Standard Package Contents

| Camera with Lens-cap | Driver CD & Installation Guide |
|---|---|
|  |  |
| | Tripod Adapter with screws |
| |  |
| | GPIO Wiring Connector |
| |  |

## 2.2 Optional Accessories

To optimizing system performance and reliability, 4 optional accessories are suggested.

### 2.2.1   VISION BOX platform

| Model No. | VB-115C | VB-115H | VB-216C | VB-216H |
|---|---|---|---|---|
| Appearance | | | | |
| CPU | Intel Mobile Celeron M 1.5GHz | | Intel Core Duo T2300E 1.66GHz | |
| Chipset | Intel 910GM | | Intel 945GME | |
| L2 Cache | 1 MB | | 2 MB | |
| Memory | 2 GB DDR2 533 | | | |
| OS Storage | 4GB 233x Compact Flash Card | 2.5" 160GB SATA Hard Disk | 2GB 266x Compact Flash Card | 2.5" 80GB SATA Hard Disk |
| 1394 ports | 1394a ports (400Mbps bandwidth x 2 ports) | | | |
| PCI Bus | One 32-bit/33MHz PCI Slot, Not support PCI long card. | | | |
| Front I/O | 2 x USB 2.0 Ports, ATX power on/off switch, Status LEDs (HDD Access, Power, LAN) | | | |
| Rear I/O | PS/2 connector, VGA connector, 6 x USB 2.0 ports, 2 x GbE LAN ports, 4 x Serial Ports via one DB44 connector (3 x RS-232, 1 x RS/232/422/485) , DVI Interface, 1 x MIC-in, 1 x Speaker-out, 2-pin connector output for remote power on/off switch, DC-in power connector for +12V ~ +30V DC power input | | | |
| Power | DC to DC power designed for on-board, supporting from 12 to 30VDC One External 120W AC adapter (Input: 100~240VAC, 2A, 50/60Hz; Output:19VDC) | | | |
| Dimensions | 195 mm (W) x 268 mm (D) x 107 mm (H) | | | |
| Construction | Fan-less design with aluminum cooling chassis | | | |
| Environment | Operating temperature: Ambient air temperature : 0°C to 40°C VISION BOX case ( Surface Temperature of Chassis) ➢ 5°C to 45°C (W/HDD); -10°C to 50°C (W/CF card only) Storage temperature: -20°C to 80°C Relative humidity: 10% to 90% (Non-condensing) | | | |
| Certification | CE, FCC A | | | |

## 2.2.2  1394 Interface Cards

Sparrow camera support to with both 1394a / 1394b interface card.

### 2.2.2.1 Dual-port Card: IOI-4601-21

The IOI-4601-21 is dual IEEE 1394 host controller PCI interface card. It supported simultaneous dual ports data transmission at rates of 400Mbps.

| | |
|---|---|
| Host Bus | 32-bit PCI local bus complies with PCI 2.1 and 2.2 specification<br>Use only one IRQ for both OHCI 1.1 channels |
| Interface Protocol | Bus Master DMA |
| 1394 Bus Transfer Rate | 100/200/400 Mbps |
| Host Bus Burst Data Rate | Up to 133 MB/s burst rate |
| IEEE-1394 to PCI Chip | 2x Ti TSB43AB21 |
| 1394 Connector | External X 2 (FW-6pin X 2), OHCI 1 (FW-6pin X 1), OHCI 2 (FW-6pin X 1) |
| 1394 Bus Power Connector | Mini 4-pin DC +12V power connector |
| Bus Power Connector | with mini 4-pin DC + 12V Power Connector |
| Performance | Maximum 1394 Bus Transfer is 800 Mbps (400 Mbps per channel) |

## 2.2.3  1394 Cable

ICPDAS prepare two type 1394 cables and both with high flex reliability wiring design. The CA-1394AB-45 is IEEE 1394 bilingual cable and it's ideally for 1394a interface to 1394b connection. The CA-1394B-45 is IEEE 1394b cable and it's ideally for standard 1394b point to point connection.

### 2.2.3.1 CA-1394AB-45: 4.5 meters high flex 1394a to 1394b bilingual cable
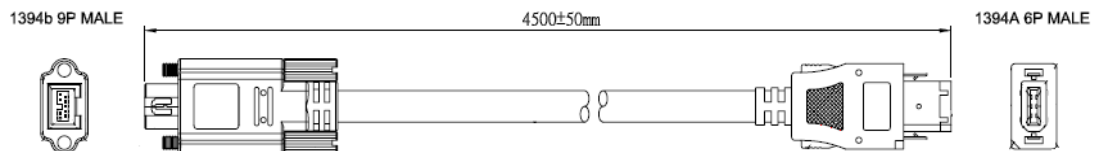
| | |
|---|---|
| 1394 connector | 9-pin male connector with screw nickel plated, PVC molding<br>6-pin male connector with spring latch, PVC molding |
| Cable wiring gauge | UL-20279 cable, (26AWG x 1pair + 2 x DRAIN + AM) x 2pcs, 22AWG x 2 conductors. Double shielded. |
| Length | 450mm |

### 2.2.3.2 CA-1394B-45: 4.5 meters high flex 1394b cable

1394b 9P MALE                     4500±50mm                     1394b 9P MALE
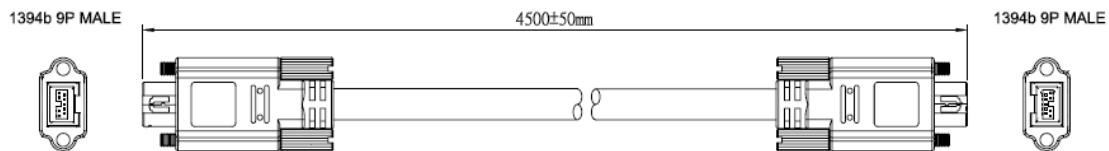
| 1394 connector | 9-pin male connector with screw nickel plated, PVC molding |
|---|---|
| Cable wiring gauge | UL-20279 cable, (26AWG x 1pair + 2 x DRAIN + AM) x 2pcs, 22AWG x 2 conductors. Double shielded. |
| Length | 450mm |

### 2.2.4  1394 Repeater: 1394R3B

1394R3B repeater offers 1port to 2 ports IEEE 1394a signal repeat and cable extension need. The 1394R3B allow convert 1394-1995 to 1394a.



| Top Side View | Left Side View | Right Side Vide |



| Chip | PHY: TI. TSB41AB3 |
|---|---|
| 1394 Bus Transfer Rate | 100 / 200 / 400 Mbps |
| Device Interface | A 400-Mbps, 2-port, 3.3V PHY |
| Power Input Range | DC 12V ~ 30V, Max. 1.35A |
| Connector | 6 Contact Male x3, DC - Walkman-type 2.0mm DC Jack x1 |
| Dimension | 72mm(W) x 58mm(H) x 20mm(D) |

|  | **1394R3B DC power input ONLY for using Notebook 1394 port or 1394 PCMCIA interface card.** |
|---|---|

# 3. Installation Guide

Sparrow IMS-130 IEEE 1394b cameras support operation in VISION BOX, IPC, PC and Notebook system. The installation guide shows how user should use this camera correctly and safely by steps. Please read through this installation guide carefully and follow the installation steps of your specified system platform.

| CAUTION | |
|---|---|
| 🚫 | **DO NOT open the camera housing in any situation.** Touching internal components may damage camera function. Meanwhile when customer to open camera housing then warranty will be void immediately. |
| ⚠️ | Be careful not to allow liquids, flammable or metallic material inside of the camera. |

## 3.1 Hardware Installation

### 3.1.1 VISION BOX

The VISION BOX has built-in two 1394a ports with 6-pin standard 1394a connector. User can use 1394a to 1394b bilingual cable connected to VISION BOX directly.

### 3.1.2 IPC/PC Platform

- With built-in 6 pin1394 port: simply plug-in 1394a to 1394b bilingual cable to 1394 port .

- Without built-in 1394 port: Please following steps as below to install your 1394 interface card on a PCI or PCI-E bus:

  (1) Remove the computer cover using the instructions from the computer manual.

  (2) Check that there is an empty PCI or PCI-E slot to accommodate the card.

  (3) Remove the blank metal plate located at the back of the selected slot (if any). Keep the removed screw to fasten the 1394 card after installation.

  (4) Carefully plug in the 1394 card in the selected PCI or PCI-E slot as illustrated below. If using a tower computer, orient the board to suit the board slots.
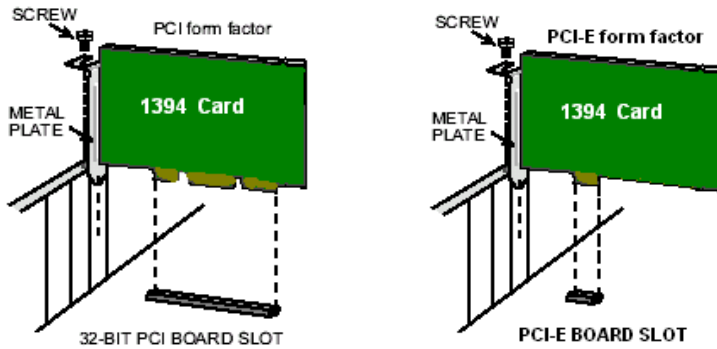
(5) Once perfectly aligned with an empty slot, press the card firmly but carefully in to the connector.

(6) Anchor the board by replacing the screw.

(7) Using 1394 cable (1394b interface card uses 1394b cable; 1394 interface card uses 1394a to 1394b bilingual cable) to connect Sparrow IMS 1394b camera to 1394 card and GPIO cable wiring if necessary. For image acquisition test please refer to the "EZView Pro Utility".



(8) Turn on the system and you will be able to find the 1394 Host controller device with Device Manager.

(9) If you need to extend your working distance, please using 1394 repeater directly.



| ⚠ | DO NOT input DC power to 1394 repeater when using IPC/PC platform. The DC power input may damage your 1394 card or 1394 host controller circuit. |
|---|---|

### 3.1.3 Notebook PC / PCMCIA Socket

- If your notebook PC has an iLink/S400 interface port then your notebook PC has a built-in 1394 host controller. So please follow the steps                    for installation.

    (1) Please prepare one IEEE 1394 repeater, one 4-pin to 6-pin IEEE 1394 cable, one 9-pin to 6-pin IEEE 1394a to 1394b bilingual cable and one walkman type DC power adapter first.

    (2) Please use 4-pin to 6-pin IEEE 1394 cable and 4-pin connector to iLink/S400 interface port and 6-pin connector to 1394 repeater.

    (3) When 6-pin connection to 1394 repeater, then please plug in DC power adaptor and another 6-pin to 6-pin 1394 cable.

    (4) 9-pin to 6-pin 1394 connected to Sparrow IMS Camera IEEE 1394 port and GPIO wiring connection if necessary.

● If your notebook PC does not have a 1394a interface port, then please plug your 1394b PCMCIA card into the socket, and follow the steps for installation.



(1) Please prepare one IEEE 1394b PCMCIA card, one IEEE 1394b cable, one GPIO cable and DC power supply unit.



(2) Plug IEEE 1394b PCMCIA card to Notebook PCMCIA socket as below.



(3) Use 1394b cable connected to IEEE 1394b PCMCIA card and Sparrow IMS Camera IEEE 1394b port directly.

(4) Use GPIO connector and wiring out with DC power supply unit. The GPIO pin-7 for external DC power and the GPIO pin-5 for the power GND.

**(5)** Plug GPIO cable into Sparrow IMS Camera GPIO port and wiring the I/O lines connection if necessary.

## 3.2 XP Driver Installation

Do not plug in any Sparrow cameras before driver is installed completely. Please refer to the following installation for different programming environment installations.

### 3.2.1 Visual Studio (VC/VB/BCB/VC.NET) Users

(1) Insert the Sparrow Driver CD to CD-ROM/DVD-ROM drive.

(2) The Sparrow Driver CD will start to prepare driver installation as below.
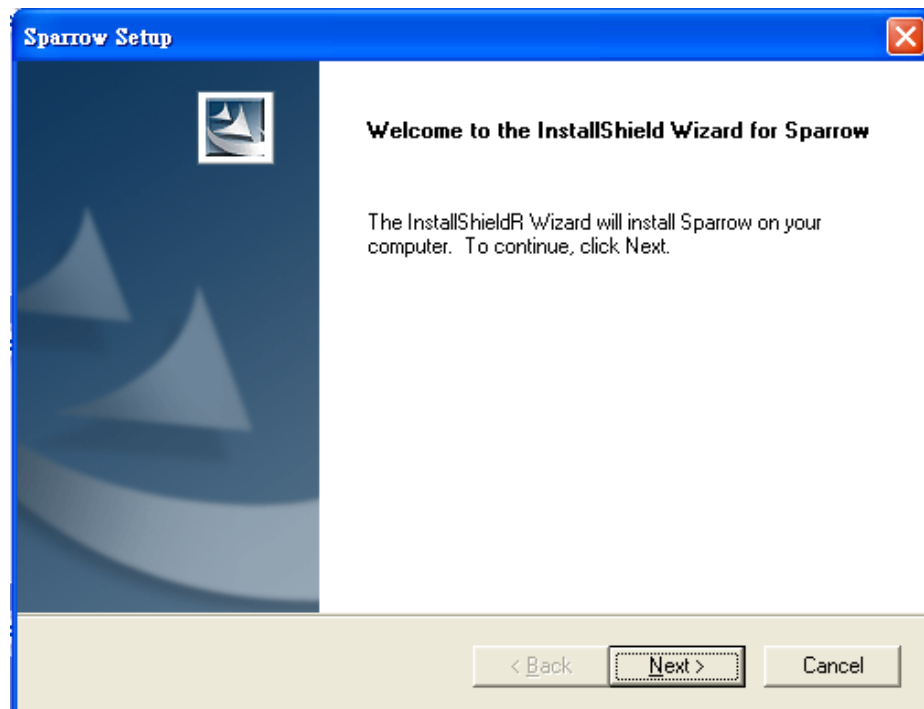


(3) Please click "Next" button for driver installation.
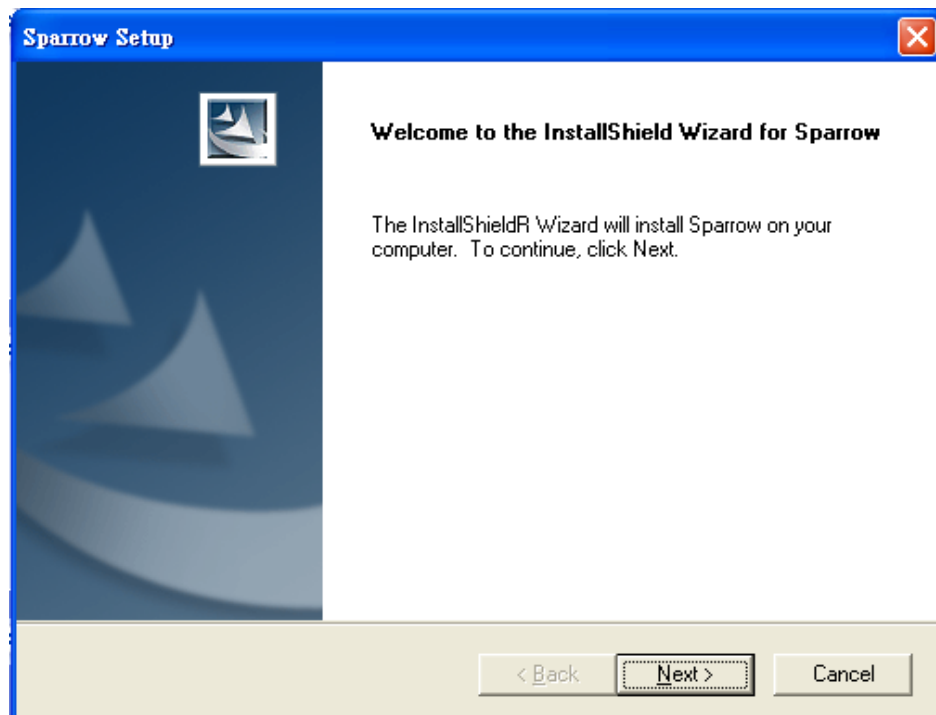
(4) The Sparrow IMS camera default path located at C:\ICPDAS\Sparrow\, and you can click "Change.." button to change driver installation path, otherwise please click "Next" button for continue driver installation.
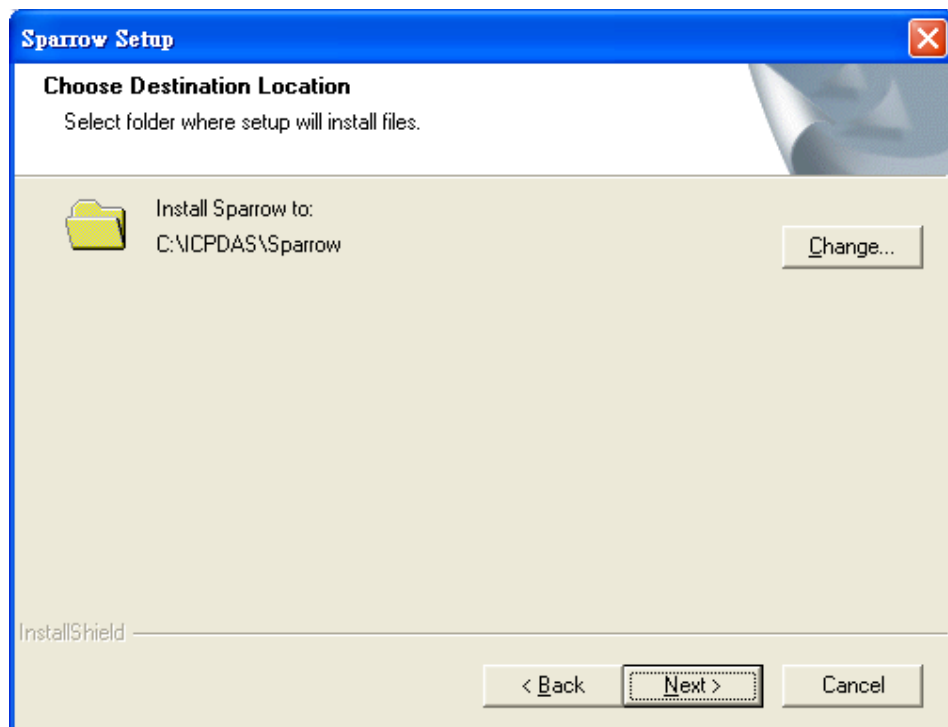


(5) Please click the "Install" button
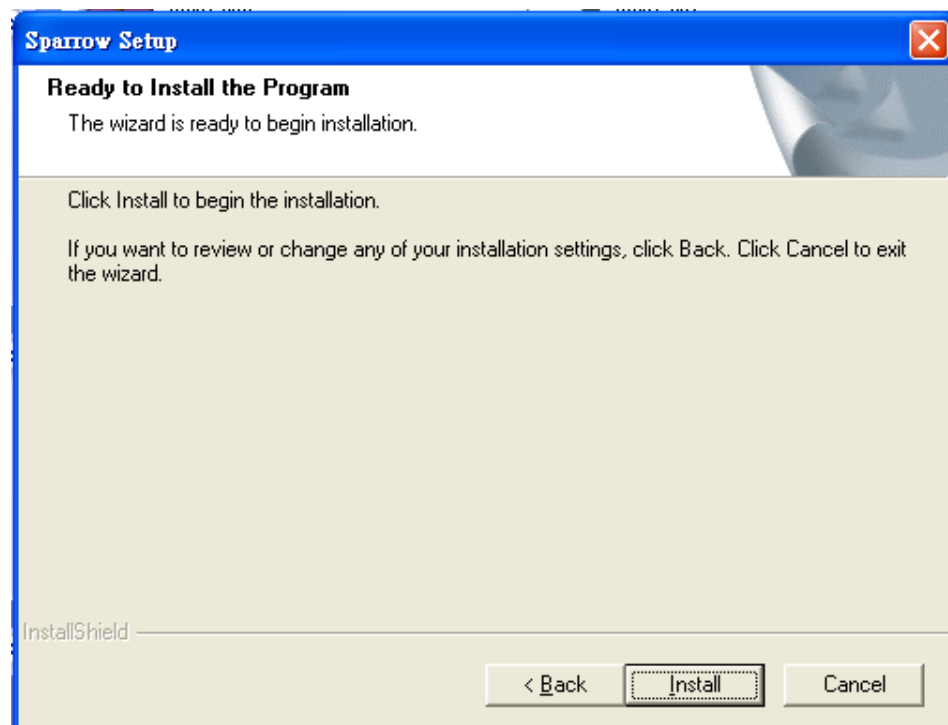
(6) Driver installing
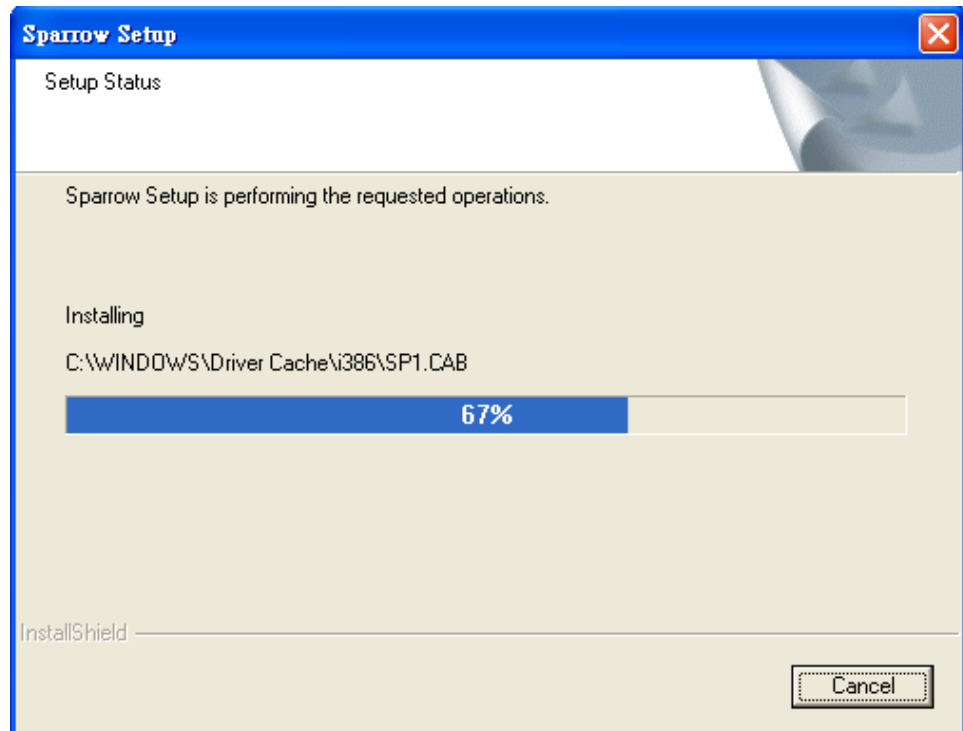


(7) When driver install is completed, please click the "Finish" button and restart your
    system.
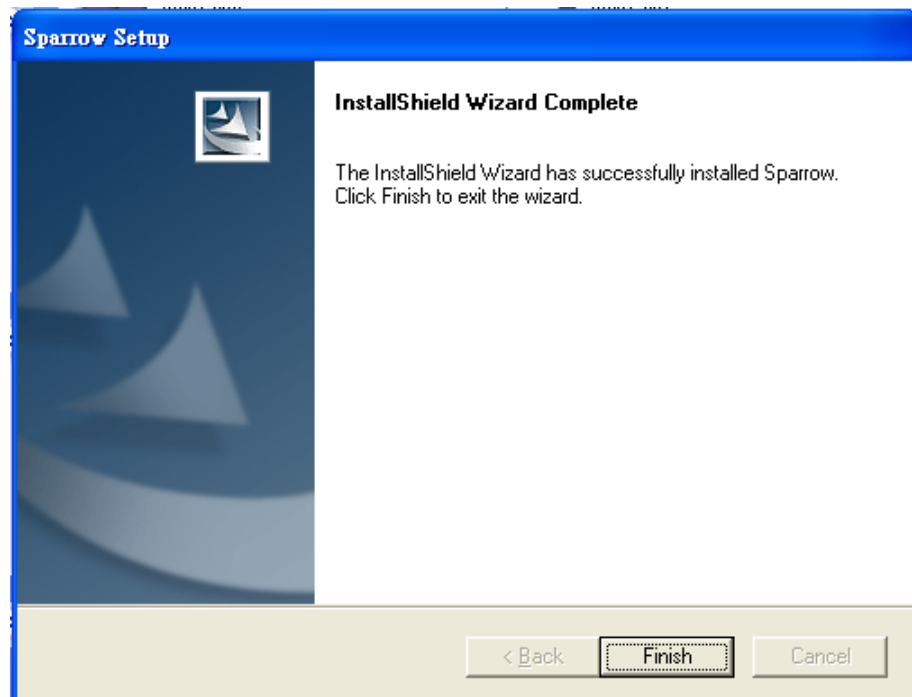
(8) After system re-boot, please plug in the Sparrow IMS cameras and go to "Device Manager" and make sure you see the "ICPDAS IEEE-1394 digital camera Device" in the list of "ICPDAS Sparrow".
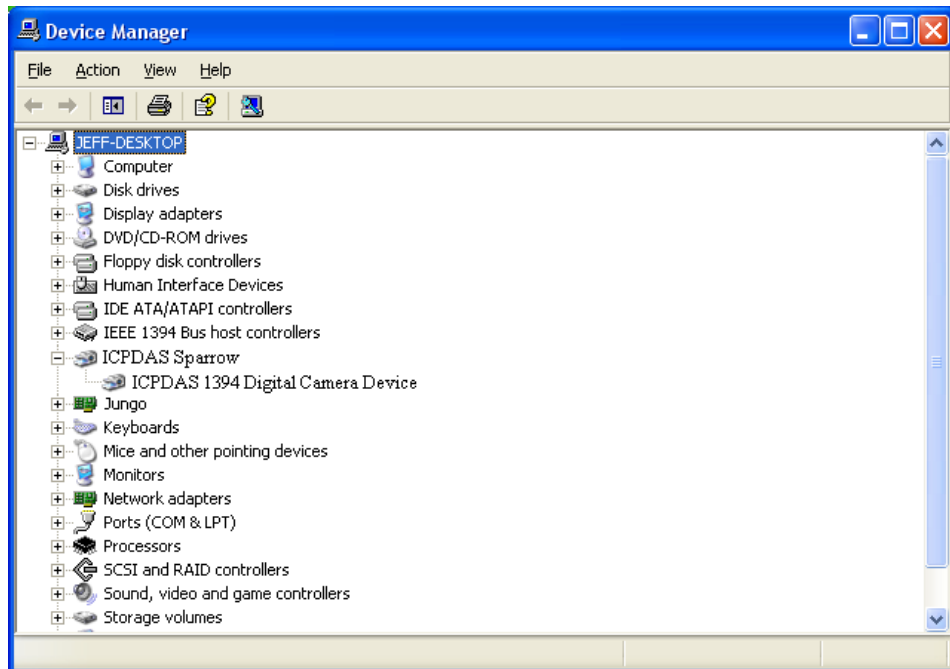


(9) If here is another vendor's 1394 camera driver in system, it may need to remove the other's driver and install Sparrow driver by manually.

(10) When the installation has been completed, open "EZView Pro" utility for image acquisition test, and please refer to "4. EZView Pro Utility" for details.

### 3.2.2 NI LabVIEW Users

If you prefer to use LabVIEW from NI (National Instruments) for your system development, then we'll suggest you to use NI-IMAQ directly.

Sparrow IMS camera are fully compatible with NI-IMAQ-1394. Please just select "NI-IMAQ IEEE 1394 IIDC Digital Camera" driver for your installation and DO NOT install the Sparrow driver.

After installation is completed, then you can use "Measurement & Automation Explorer" of NI for configuration of the camera and the image grab test.
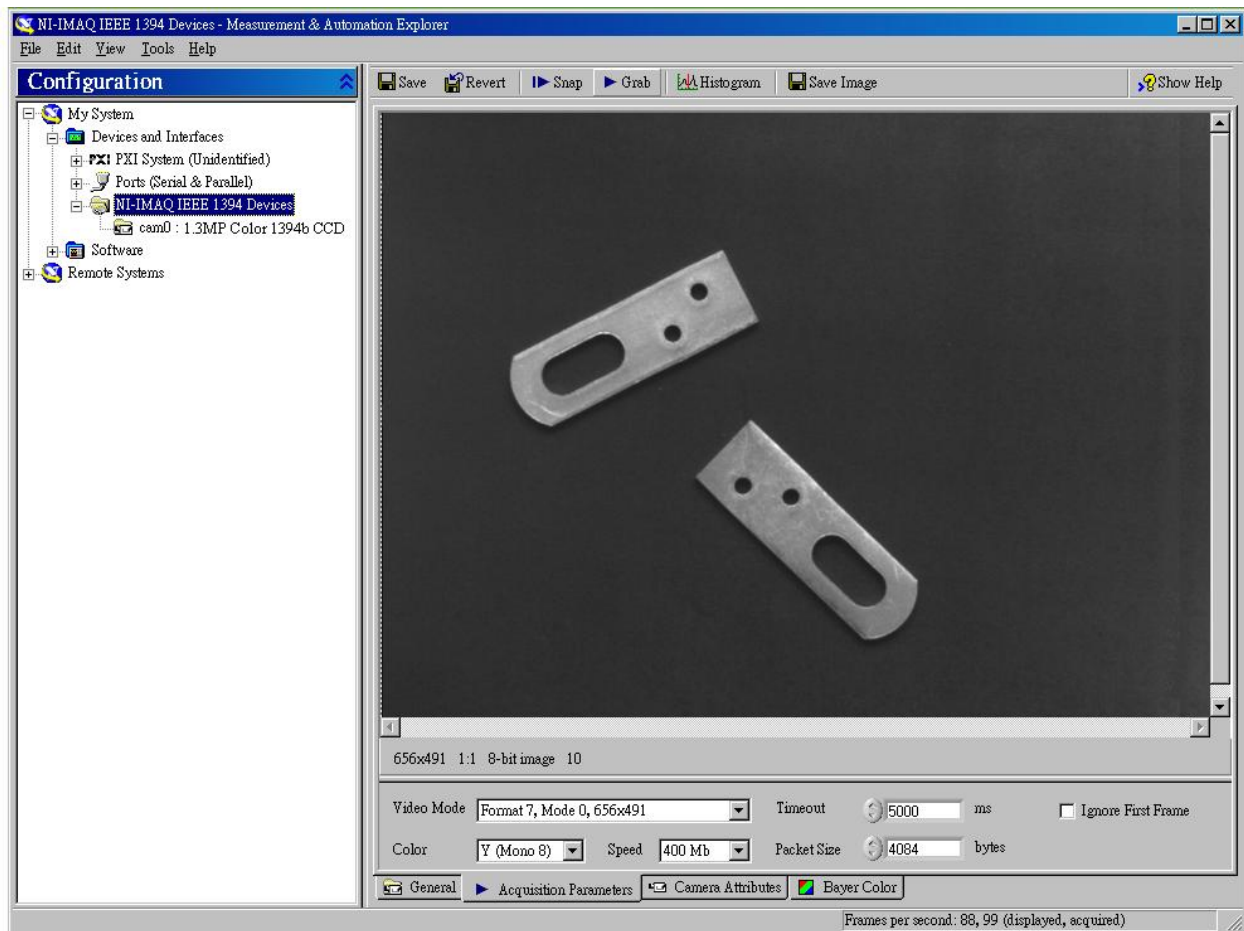


| ⚠ | When you install the NI-IMAQ-1394 driver, all Sparrow official drivers, and API and EZView Pro utilities would be fail to work. |
|---|---|

### 3.2.3 Windows XP SP1 rollback

Windows XP Service Pack 2 or above version may decrease the performance of IEEE 1394 device due to XP SP2 compatibility. To optimize IEEE 1394 performance, MS XP SP1 is suggested

There have two solutions for get the fix SP2 bug installation file.

(1) Go to the Microsoft download center for download the file at : http://www.microsoft.com/downloads/details.aspx?FamilyId=CA0F2007-18B5-4112-8BD6-8BF4BD3130B9&displaylang=en

(2) Or after Sparrow driver install completed then user will find a 'FixSP2g' utility under the list menu of Sparrow, then user can just click it and follow the steps until get the rollback to SP1 successfully information.

After you install this update, if you add a host controller for a new 1394 device to your computer, you must add or modify the SidSpeed entry in the Windows registry for the new 1394 host controller. To do this, follow these steps:

(1) Click **Start**, click **Run**, type 'regedit', and then click **OK**.

(2) Locate and then click the following registry subkey:

   **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\PCI\\*1394_hc_hw_id*\\*1394_instance_id*\Device Parameters**

(3) If the SidSpeed value does not exist in the right pane after you click the **Device Parameters** subkey in step 2, create it. To do this, point to **New** on the **Edit** menu, click **DWORD Value**, type SidSpeed, and then press ENTER.

(4) Right-click **SidSpeed**, and then click **Modify**.

(5) In the **Value data** box, type one of the values that are shown in the following table, and then click **OK**.

   Collapse this tableExpand this table

   **Value Speed**

   0      S100 speed

   1      S200 speed

   2      S400 speed (default value)

   3      S400/S800 speed (Windows XP Service Pack 1 [SP1] value)

   **Note** If you try to use a value that is larger than 3, SidSpeed will use a value of S100.

(6) Quit Registry Editor.

# 4. EzView Pro Utility

Once hardware installation is completed, please make sure the cameras are configured correctly in Device Manager before running the EZView Pro utility. This chapter outlines how to establish a vision system and how to manually control the Sparrow camera to verify correct operation. EZView Pro provides an easy way on setting, configuring, testing, and debugging the vision system.

**Note**: EZView Pro is only available for Windows XP with a recommended screen resolution higher than 1024 x 768.

## 4.1 Overview
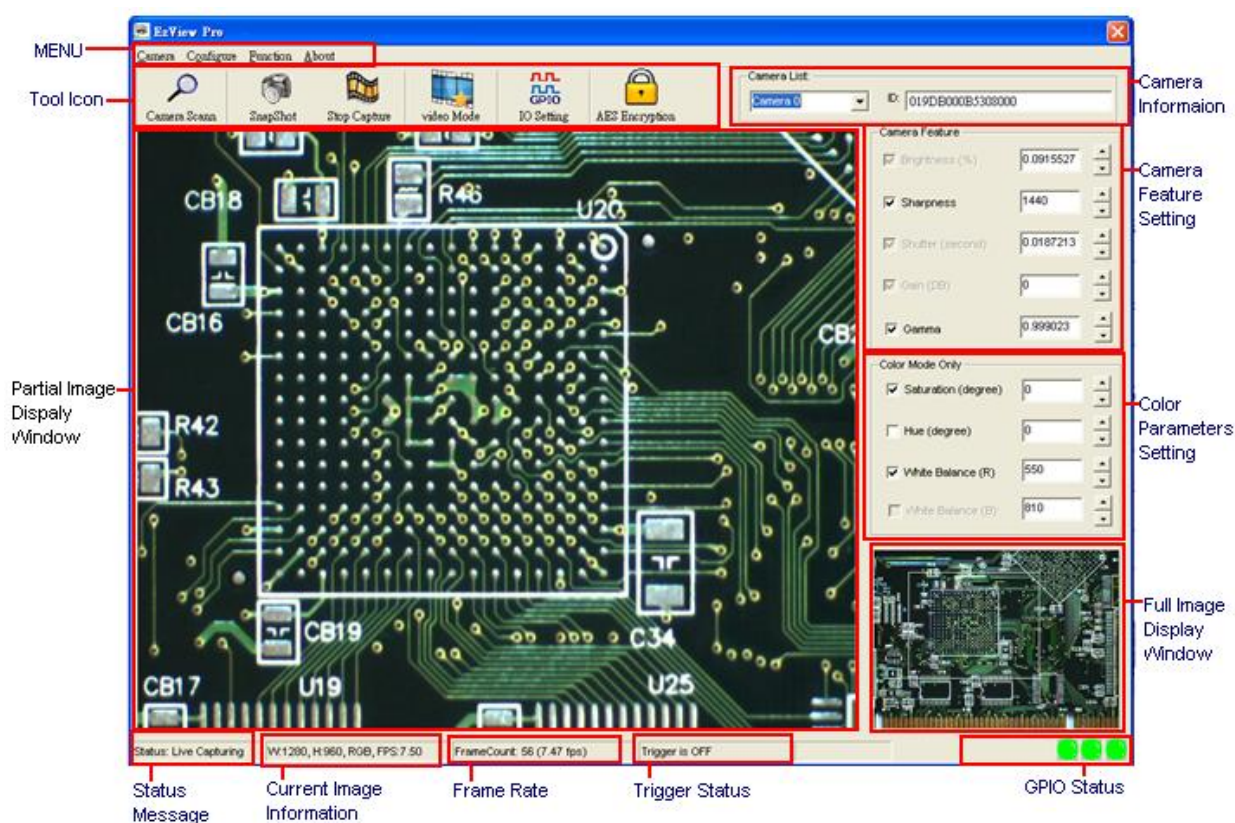


Figure 4-1 EzView_Pro utility layout

## 4.2 Menu

The most functionalities control of Sparrow camera is under EZView Pro utility menu which defines as, Camera, Configure, Function and About selections.

### 4.2.1 Camera

The Camera function menu will show up function control included: Scan Camera, Select Camera, Video Mode, Camera Info, Check Bus Rate and Exit.
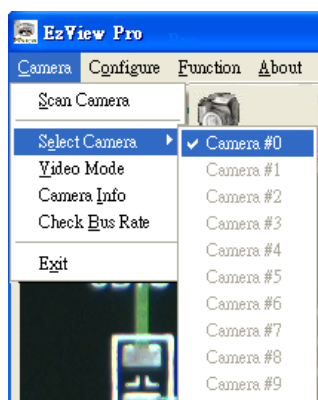


■ **Scan Camera**

This function is same with Scan Camera icon, and EzView Pro utility will scan and list out the Sparrow 1394 cameras available at the local computer.

■ **Select Camera**

This function is same with Camera Information combo list, and user can use this function to select specific available camera.

■ **Video Mode**

■ **General Video Mode Setting**

The Sparrow camera General Video Mode support 4 standard resolutions and 3 difference color modes, and the maximum possibility frame rate for difference resolution and color mode combination. Limited by Windows display component, Gray 16 bit video mode cannot be displayed properly. It actually works as Gray 8 bit mode if using Gray 16 bit mode.
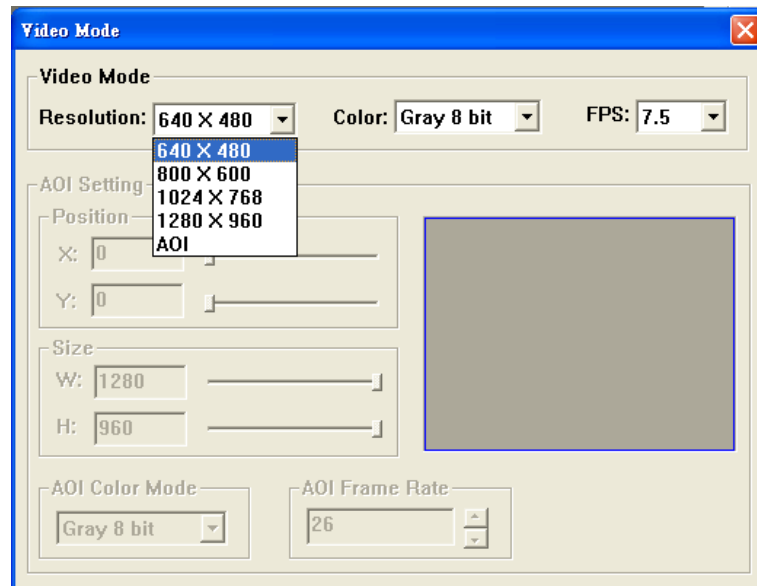
■ **AOI Setting**

The AOI Setting only workable when user select the Video Mode in AOI. User can set the AOI size and start position for the area of interest field. Also, Sparrow camera will be calculated for maximum possible AOI frame rate for difference AOI size and color mode combination.
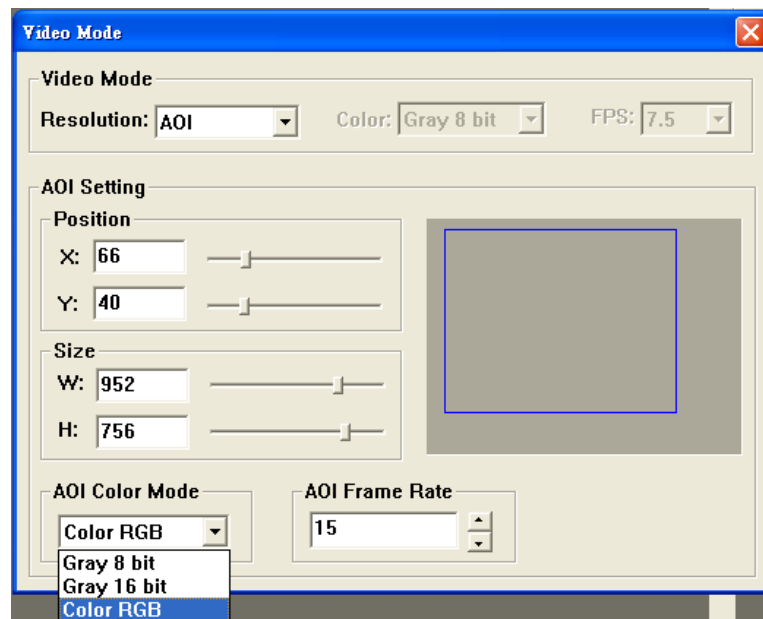


■ **Camera Info**

When click Camera Info function will show up a message box and list current camera model name and serial number information.
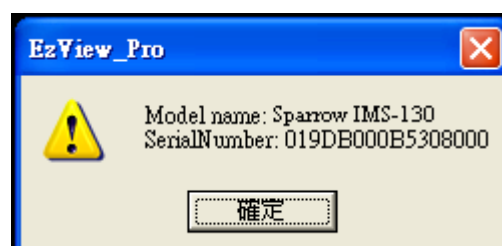
■ **Check Bus Rate**

Click Check Bus Rate function will show up the current camera connected 1394 bus bandwidth in message box
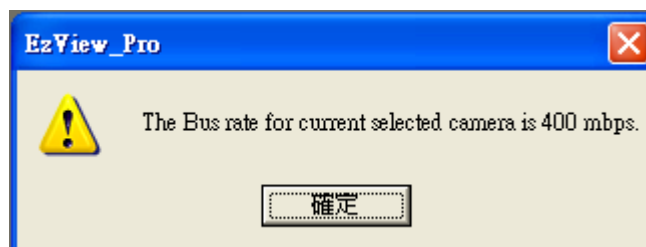


## 4.2.2  Configure

The Configure function is able to save current camera operation parameters setting into Sparrow camera configure memory or into a specific file location. Also, user can use this function to get the camera configuration information from Sparrow camera configures memory or specific file location.
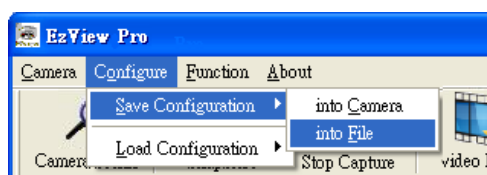


## 4.2.3  Function

The Function menu function control includes following Ext. Trigger, GPIO & Strobe, Enable Test Image, LUT and AES encrypted SN code.
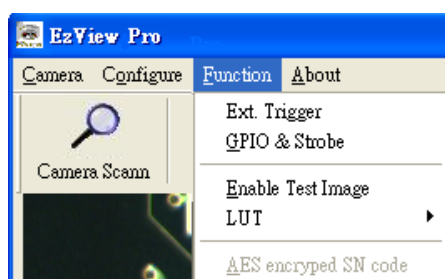


■ **Ext. Trigger**

Click Ext. Trigger function Ext. Trigger Setting dialog will be pop-out. User can select the trigger source and trigger mode for it. The trigger function is activated when trigger is enabling
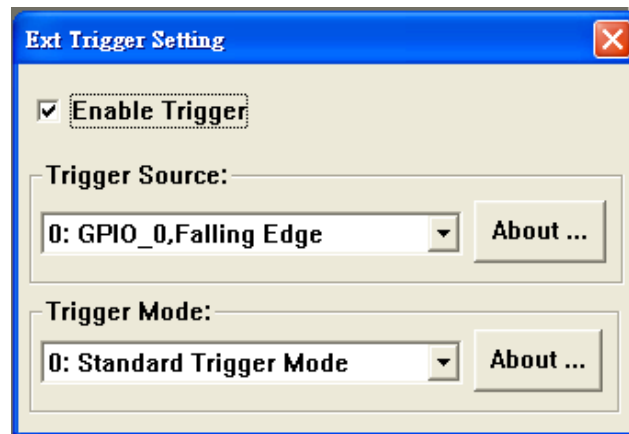
■ **GPIO & Strobe**

Click GPIO & Strobe GPIO dialog will be pop-out. User can select the GPIO channel and set the IO or strobe function active by high or low. Also, user can set the strobe delay time and duration time for strobe output timing.
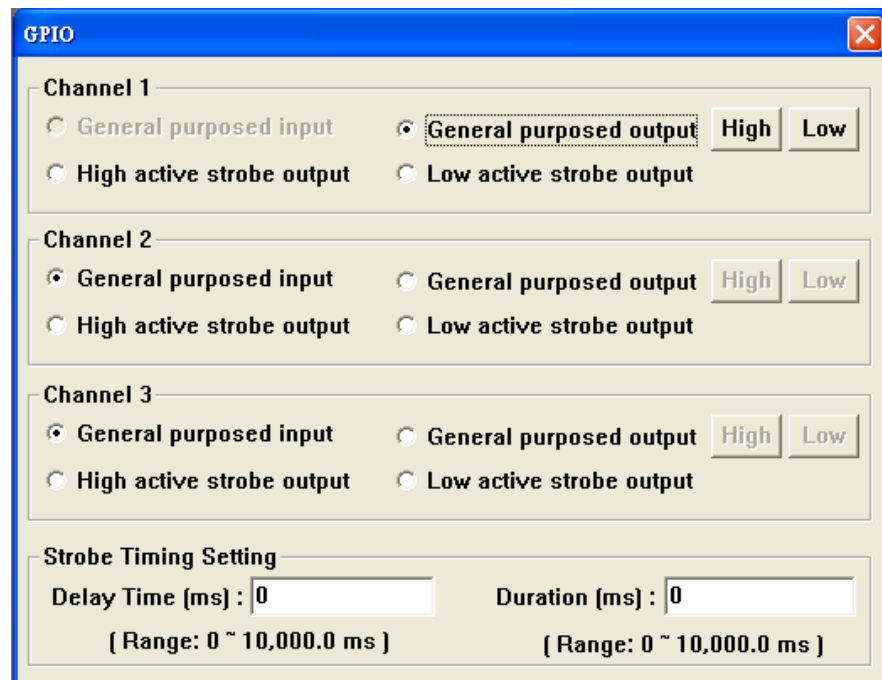


■ **Enable Test Image**

Click Enable Test Image and click Snapshot or Live Capture function icon. Sparrow camera will generate internal testing pattern and transmits out the image data as below. Sparrow camera will not be able to acquire any live image before user to disable the test image function.
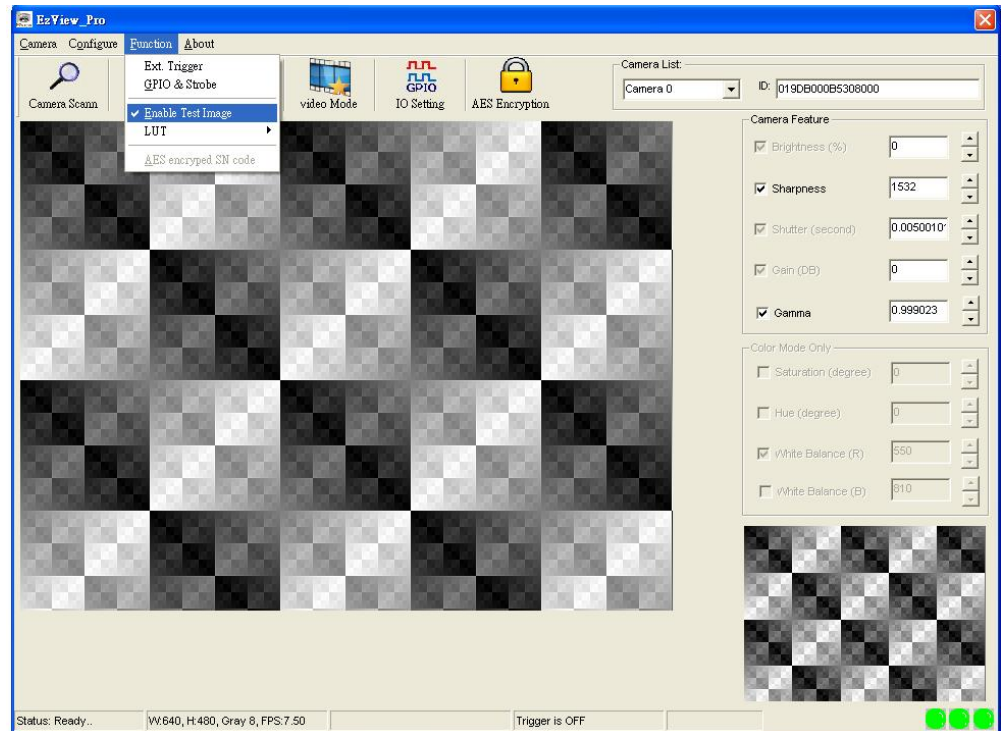
■ **LUT**

User is able to download or upload the camera look up table information via LUT function. The LUT information present by txt file format and user can modify the LUT information then upload to Sparrow camera for modifying the image data analog to digital convert structure.

■ **AES encrypted SN code**

Click AES encrypted SN code AES encryption dialog will be pop-out. User can input 16byte key word for combine with current Sparrow unique hardware ID, and will generate an AES encrypted register code. Normally user can save this AES code for protect the system copy right.

### 4.2.4  About

Click About copyright and version information of EzView_Pro utility is shown up in message box
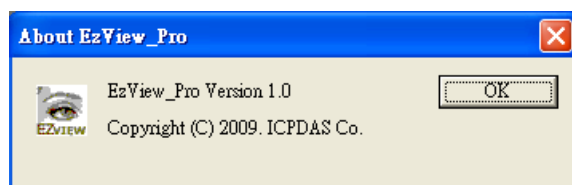
## 4.3 Tool Icons

| | |
|---|---|
| Camera Scann | **Camera Scan**<br>Click the Camera Scan icon to list the Sparrow 1394 cameras available at the local computer. |
| SnapShot | **Snap Shot**<br>Click the Snapshot icon. A single image will appear in the Full Image and Partial Image display window. |
| Stop Capture | **Live/Stop Capture**<br>Click the Video Capture icon. A video frame will appear in the Full Image and Partial Image display window. |
| video Mode | **Video Mode**<br>Click the Video Mode icon. The Video Mode dialog will show up for the selection of video mode parameter setting. |
| GPIO<br>IO Setting | **IO Setting**<br>Click the IO Setting icon. The GPIO dialog will show up for the selection of GPIO and strobe control setting. |
| AES Encryption | **AES Encryption**<br>Click AES Encryption icon |

## 4.4 Camera List

The Camera list will show up current available camera in local computer and the ID box will show the camera ID information once the camera has been selected.

Camera List:
Camera 0     ID: 019DB000B5308000

## 4.5 Display Window

The EzView_Pro utility supports two display windows for real-time image display. The full image data will be reduction show in the bottom right display window, and there have a ROI (Regional of Interest) box to allow user to move the position to see the ROI raw data image show in the partial image display window.
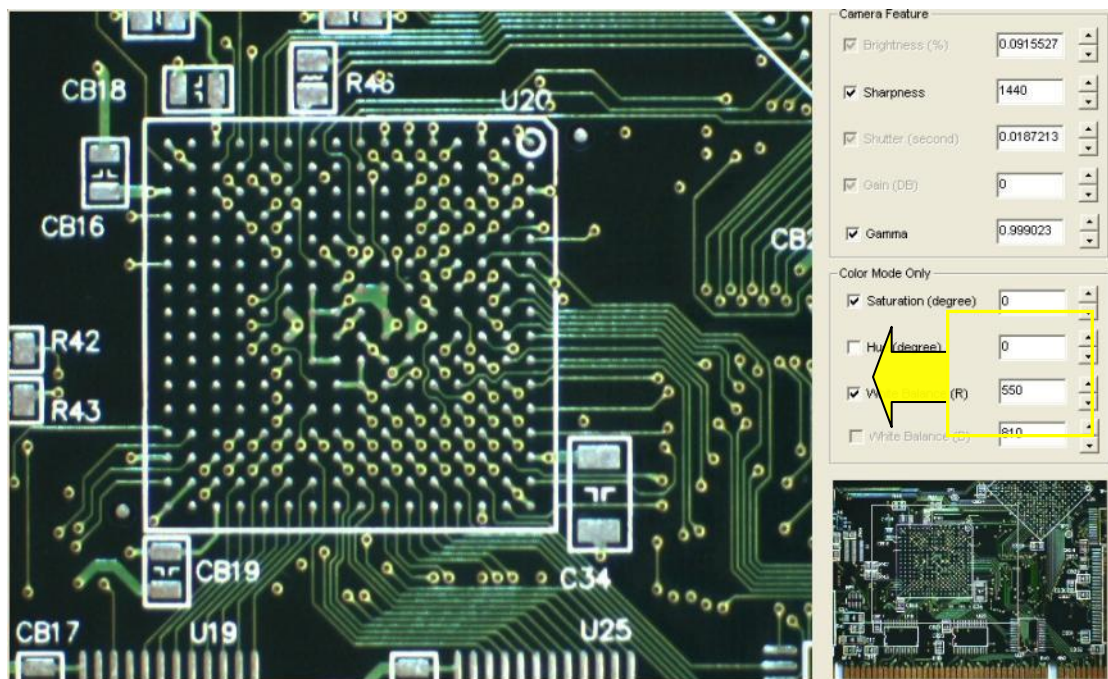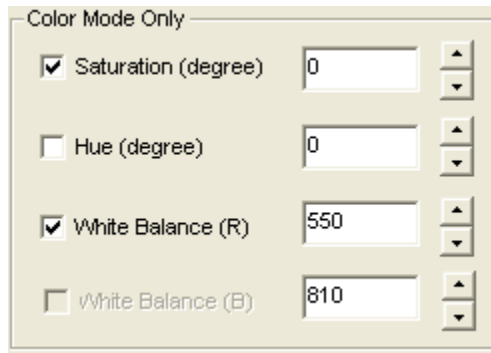


## 4.6 Camera Feature

The Camera Feature allow user real-time to adjust the camera parameter while camera during the image acquisition. Each parameter range and functionality description please refers Chapter 1.4 for detail information.
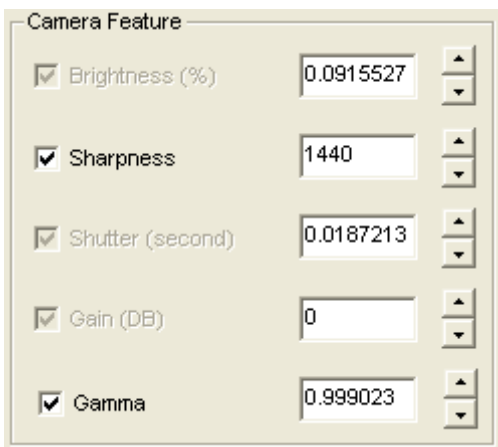
## 4.7 Color Parameter

The Color Parameter function is only work on camera Video Mode RGB selected. User can adjust the color parameter during the image acquisition. Each parameter range and functionality description please refers Chapter 1.4 for detail information.

# 5. Function Library

Users can use these functions to develop application programs under Visual C++ 6.0, Visual Basic 6.0, Boland C++ Builder 6.0, and VC.NET 2005.

The Sparrow DLL file (IMSCamera.dll) is in common use on Visual C++, Visual Basic, and Boland C++ Builder and VC.NET 2005 development language.

For Visual C++, Boland C++ Builder and VC.NET 2005, please just follow standard Syntax description to use.

For Visual Basic, here is "IMSCamera.bas" module file in the sample program and user also can define or modify function module file as you need.

Please refer to Table 5-1 List of Functions for functions by category. All the data types follow Microsoft standard definitions.

# 5.1 List of Functions

| Category | Section | Function |
|---|---|---|
| Camera Management | 5.3 | IMS_ScanCamera |
| | | IMS_InitCamera |
| | | IMS_InitCamera_SC |
| | | IMS_GetLastError |
| | | IMS_CloseCamera |
| | | IMS_Check_BusRate |
| Callback Event | 5.4 | IMS_Register_Callback |
| | | CallBackRoutine |
| Camera Acquisition | 5.5 | IMS_Start_Acquire |
| | | IMS_Stop_Acquire |
| Camera Configuration | 5.6 | IMS_Set_VideoMode |
| | | IMS_Get_VideoResolution |
| | | IMS_Get_ColorFormat |
| | | IMS_Set_AOI_Parameter |
| | | IMS_Get_AOI_Parameter |
| | | IMS_Set_Feature |
| | | IMS_Get_Feature |
| Trigger Setting | 5.7 | IMS_Set_Trigger |
| | | IMS_Get_Trigger |
| | | IMS_Do_SWTrigger |
| GPIO Control | 5.8 | IMS_Set_GPIOMode |
| | | IMS_Get_GPIOMode |
| | | IMS_GPIO_Read |
| | | IMS_GPIO_Write |
| Strobe Control | 5.9 | IMS_Set_Strobe |
| | | IMS_Get_Strobe |
| Look Up Table | 5.10 | IMS_Set_LUT |
| | | IMS_Get_LUT |
| Data Flash Control | 5.11 | IMS_Read_DataFlash |
| | | IMS_Write_DataFlash |
| AES Project Encryption | 5.12 | IMS_Get_CameraInform |
| | | AES_GET_REG |
| Test Pattern | 5.13 | IMS_Enable_TestImage |

**Table 5-1 List of Functions**

# 5.2 Programming Flowchart

● *Camera scan & initial*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───► │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
```

● *Camera close*

```
┌─────────────────┐
│ IMS_CloseCamera │
└─────────────────┘
```

● *Check 1394 Bus Rate*
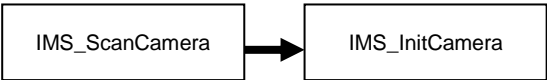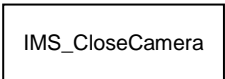
```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───► │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                │
                                ┊
                                ▼
                         ┌────────────────────┐
                         │ IMS_Check_BusRate  │
                         └────────────────────┘
```

● *Camera feature setting*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───► │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                │
                                ┊
                                ▼
                         ┌────────────────────┐
                         │ IMS_Set_Feature    │
                         └────────────────────┘
```

● *Check camera feature setting*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───► │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                │
                                ┊
                                ▼
                         ┌────────────────────┐
                         │ IMS_Get_Feature    │
                         └────────────────────┘
```
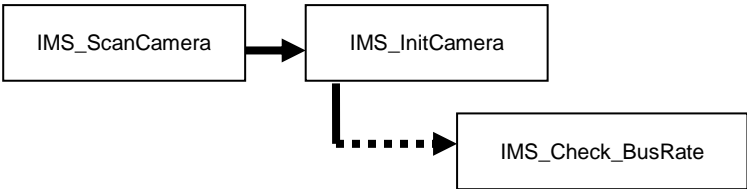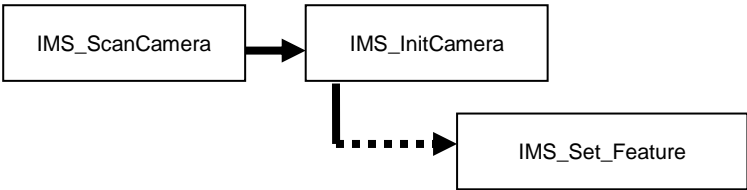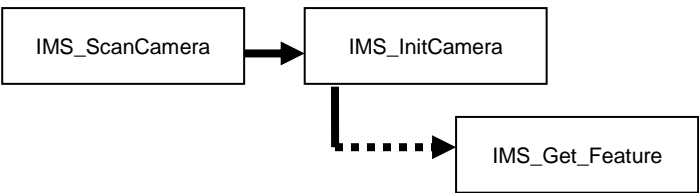
● *Camera video mode setting*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───► │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                │
                                ┊
                                ▼
                         ┌────────────────────┐
                         │ IMS_Set_VideoMode  │
                         └────────────────────┘
```
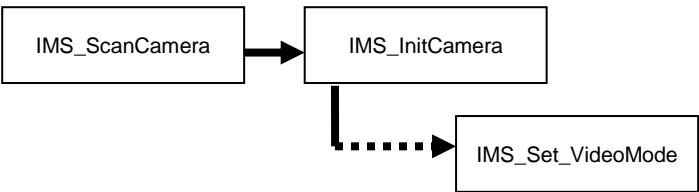
● *Check camera video resolution*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───> │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  ┊
                                  ┊    ┌──────────────────────────┐
                                  ┊┄┄> │ IMS_Get_VideoResolution  │
                                       └──────────────────────────┘
```
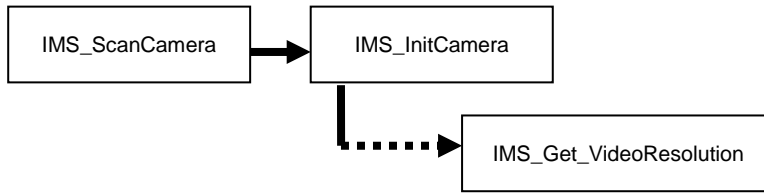
● *Check camera color format*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───> │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  ┊
                                  ┊    ┌──────────────────────┐
                                  ┊┄┄> │ IMS_Get_ColorFormat  │
                                       └──────────────────────┘
```

● *Camera Acquisition*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───> │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  │
                                  ▼
┌──────────────────────┐    ┌──────────────────┐    ┌──────────────────┐
│ IMS_Register_Callback│──> │ IMS_Start_Acquire│──> │ IMS_Stop_Acquire │
└──────────────────────┘    └──────────────────┘    └──────────────────┘
```
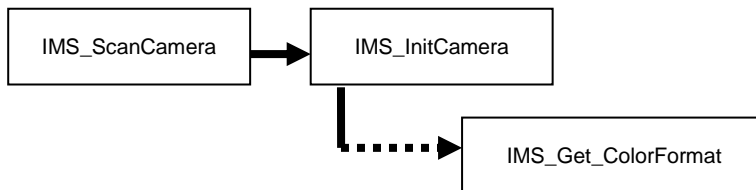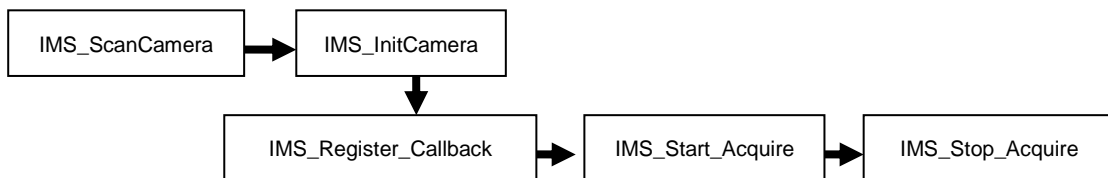
● *Camera AOI setting and acquisition*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───> │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  │
                                  ▼
                    ┌────────────────────────┐
                    │ IMS_Set_AOI_Parameter  │
                    └────────────────────────┘
                                  │
                                  ▼
┌──────────────────────┐    ┌──────────────────┐    ┌──────────────────┐
│ IMS_Register_Callback│──> │ IMS_Start_Acquire│──> │ IMS_Stop_Acquire │
└──────────────────────┘    └──────────────────┘    └──────────────────┘
```

● *Check Camera AOI setting*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───> │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  │
                                  ▼
                    ┌────────────────────────┐
                    │ IMS_Get_AOI_Parameter  │
                    └────────────────────────┘
```
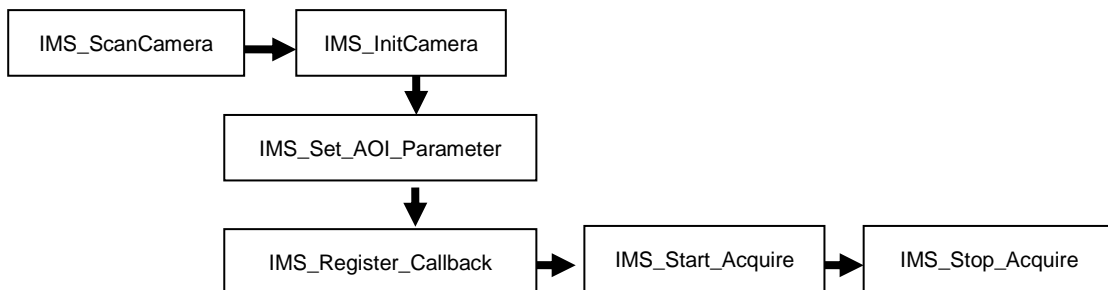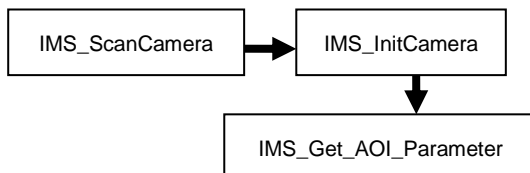
● *External trigger setting for camera acquisition*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───▶ │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  │
                                  ▼
                         ┌─────────────────┐
                         │  IMS_Set_Trigger │
                         └─────────────────┘
                                  │
                                  ▼
┌──────────────────────┐   ┌──────────────────┐   ┌──────────────────┐
│ IMS_Register_Callback│──▶│ IMS_Start_Acquire│──▶│ IMS_Stop_Acquire │
└──────────────────────┘   └──────────────────┘   └──────────────────┘
```
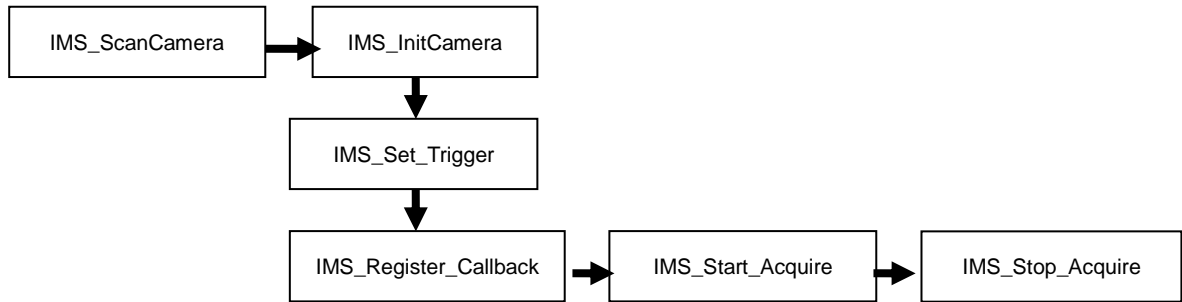
● *Camera acquisition by software trigger*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───▶ │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  │
                                  ▼
                         ┌──────────────────────┐
                         │ IMS_Register_Callback │
                         └──────────────────────┘
                                  │
                                  ▼
                         ┌─────────────────┐
                         │  IMS_Set_Trigger │
                         └─────────────────┘
                                  │
                                  ▼
┌──────────────────┐   ┌──────────────────┐   ┌──────────────────┐
│ IMS_Start_Acquire│──▶│ IMS_Do_SWTrigger │──▶│ IMS_Stop_Acquire │
└──────────────────┘   └──────────────────┘   └──────────────────┘
```
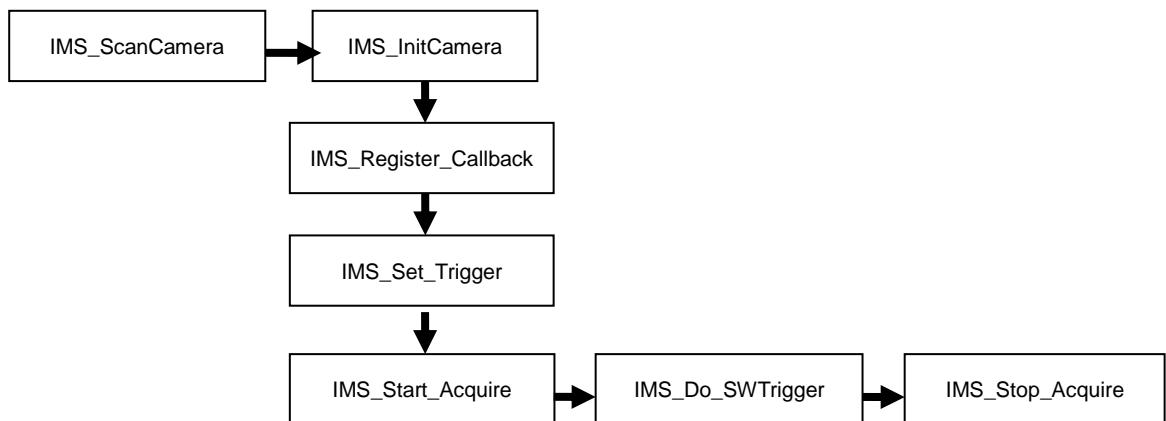
● *Enable camera test image*
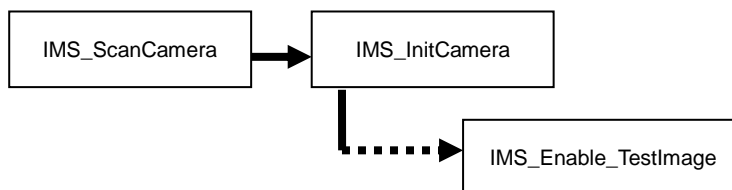
```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───▶ │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  ┊
                                  ┊┄┄┄▶ ┌──────────────────────┐
                                        │ IMS_Enable_TestImage │
                                        └──────────────────────┘
```

● *Camera digital input/output setting*

```
┌─────────────────┐      ┌─────────────────┐
│ IMS_ScanCamera  │ ───▶ │ IMS_InitCamera  │
└─────────────────┘      └─────────────────┘
                                  ┊
                                  ┊┄┄┄▶ ┌──────────────────────┐
                                        │  IMS_Set_GPIOMode    │
                                        └──────────────────────┘
```

● *Check Camera digital input/output setting*

```
┌──────────────────┐      ┌──────────────────┐
│  IMS_ScanCamera  │─────▶│  IMS_InitCamera  │
└──────────────────┘      └──────────────────┘
                                   ┊
                                   ┊      ┌──────────────────┐
                                   └┄┄┄┄▶│ IMS_Get_GPIOMode │
                                          └──────────────────┘
```
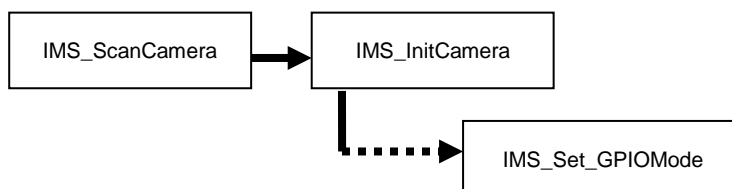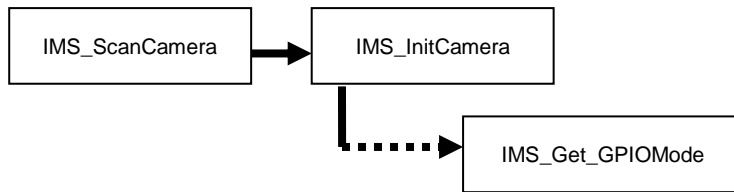
● *Write digital input/output control into camera*

```
┌──────────────────┐      ┌──────────────────┐
│  IMS_ScanCamera  │─────▶│  IMS_InitCamera  │
└──────────────────┘      └──────────────────┘
                                   ┊
                                   ┊      ┌──────────────────┐
                                   └┄┄┄┄▶│  IMS_GPIO_Write  │
                                          └──────────────────┘
```
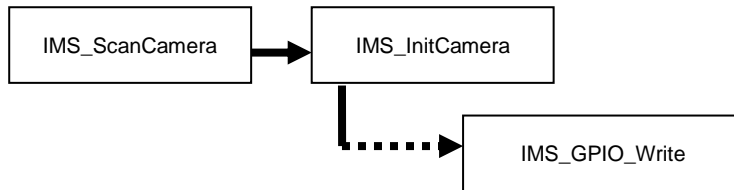
● *Read digital input/output control status from camera*

```
┌──────────────────┐      ┌──────────────────┐
│  IMS_ScanCamera  │─────▶│  IMS_InitCamera  │
└──────────────────┘      └──────────────────┘
                                   ┊
                                   ┊      ┌──────────────────┐
                                   └┄┄┄┄▶│  IMS_GPIO_Read   │
                                          └──────────────────┘
```

● *Set strobe control*

```
┌──────────────────┐      ┌──────────────────┐
│  IMS_ScanCamera  │─────▶│  IMS_InitCamera  │
└──────────────────┘      └──────────────────┘
                                   ┊
                                   ┊      ┌──────────────────┐
                                   └┄┄┄┄▶│  IMS_Set_Strobe  │
                                          └──────────────────┘
```

● *Get strobe control*

```
┌──────────────────┐      ┌──────────────────┐
│  IMS_ScanCamera  │─────▶│  IMS_InitCamera  │
└──────────────────┘      └──────────────────┘
                                   ┊
                                   ┊      ┌──────────────────┐
                                   └┄┄┄┄▶│  IMS_Get_Strobe  │
                                          └──────────────────┘
```
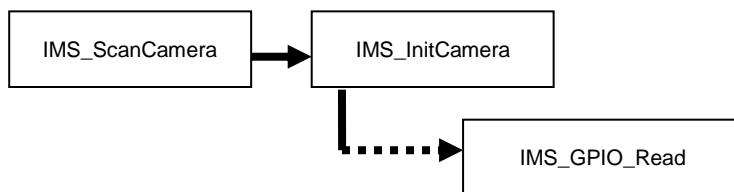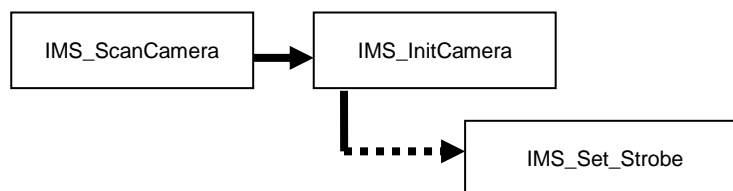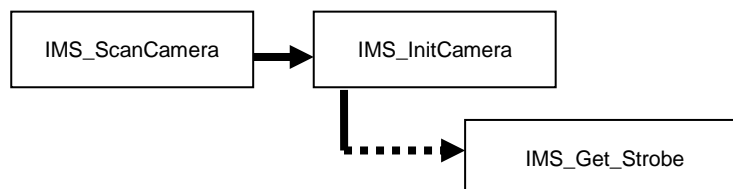
● *Set LUT (Lookup Table) into camera*

```
┌─────────────────┐        ┌─────────────────┐
│ IMS_ScanCamera  │ ─────▶ │ IMS_InitCamera  │
└─────────────────┘        └─────────────────┘
                                   ┊
                                   ┊        ┌─────────────────┐
                                   ┊┈┈┈┈┈▶  │  IMS_Set_LUT    │
                                            └─────────────────┘
```
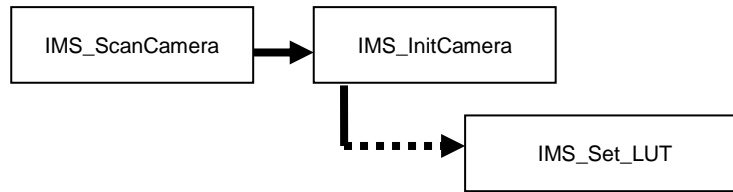
● *Get LUT(Lookup Table) from camera*

```
┌─────────────────┐        ┌─────────────────┐
│ IMS_ScanCamera  │ ─────▶ │ IMS_InitCamera  │
└─────────────────┘        └─────────────────┘
                                   ┊
                                   ┊        ┌─────────────────┐
                                   ┊┈┈┈┈┈▶  │  IMS_Get_LUT    │
                                            └─────────────────┘
```
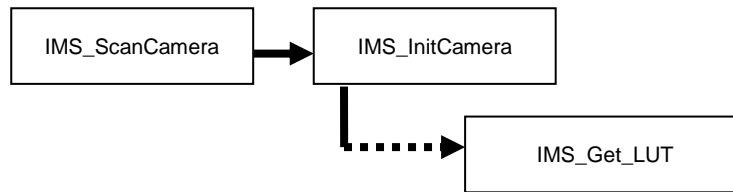
● *Write data into camera data flash (ex. Camera Configure file)*

```
┌─────────────────┐        ┌─────────────────┐
│ IMS_ScanCamera  │ ─────▶ │ IMS_InitCamera  │
└─────────────────┘        └─────────────────┘
                                   ┊
                                   ┊        ┌──────────────────────┐
                                   ┊┈┈┈┈┈▶  │  IMS_Write_DataFlash  │
                                            └──────────────────────┘
```
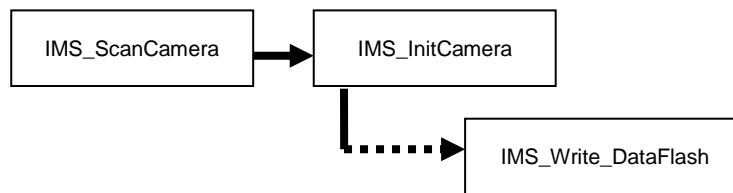
● *Read data from camera data flash*

```
┌─────────────────┐        ┌─────────────────┐
│ IMS_ScanCamera  │ ─────▶ │ IMS_InitCamera  │
└─────────────────┘        └─────────────────┘
                                   ┊
                                   ┊        ┌──────────────────────┐
                                   ┊┈┈┈┈┈▶  │  IMS_Read_DataFlash   │
                                            └──────────────────────┘
```

● *Get camera information*

```
┌─────────────────┐        ┌─────────────────┐
│ IMS_ScanCamera  │ ─────▶ │ IMS_InitCamera  │
└─────────────────┘        └─────────────────┘
                                   ┊
                                   ┊        ┌──────────────────────┐
                                   ┊┈┈┈┈┈▶  │ IMS_Get_CameraInform  │
                                            └──────────────────────┘
```
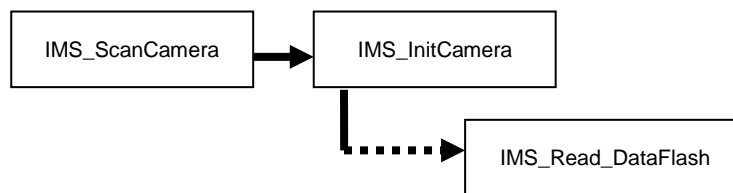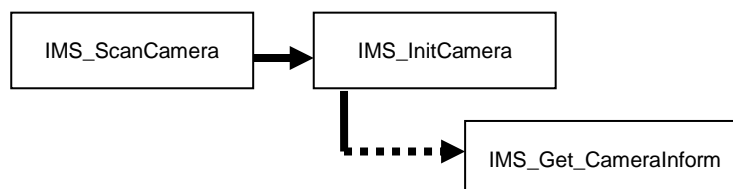
● *Get the last error of camera*

```
┌──────────────────┐        ┌──────────────────┐
│  IMS_ScanCamera  │───────▶│  IMS_InitCamera  │
└──────────────────┘        └──────────────────┘
                                     ┊
                                     ┊
                                     ▼
                            ┌──────────────────┐
                            │ IMS_GetLastError │
                            └──────────────────┘
```
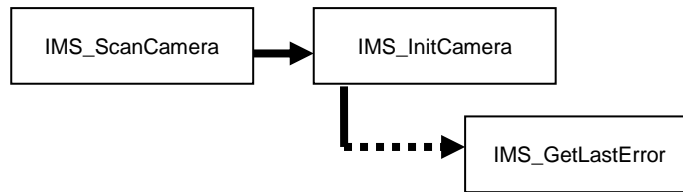
● *AES project encrypted code write into the camera*

```
┌──────────────────┐    ┌──────────────────┐
│  IMS_ScanCamera  │───▶│  IMS_InitCamera  │
└──────────────────┘    └──────────────────┘
                                 │
                                 ▼
        ┌───────────────────┐    ┌───────────────┐    ┌────────────────────┐
        │ IMS_Get_CameraInform │─▶│ AES_GET_REG  │───▶│ IMS_Write_DataFlash │
        └───────────────────┘    └───────────────┘    └────────────────────┘
```
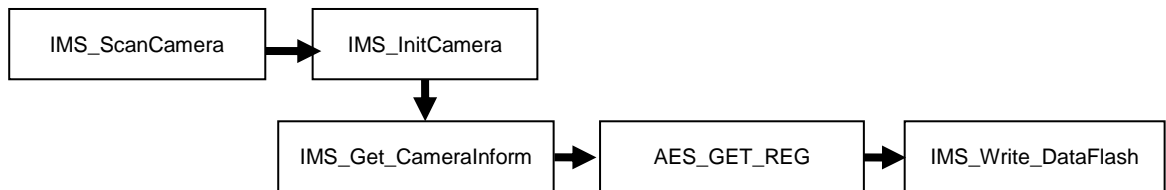
## 5.3 Camera Management

*IMS_ScanCamera*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_ScanCamera(U16\* numCam)*

**Visual Basic 6.0**

    *IMS_ScanCamera (ByRef numCam As Integer) As Integer*

**VC.NET 2005**

    *short IMS_ScanCamera(IntPtr numCam)*

**Description:**

This function scans all ICP DAS cameras in system. After scanning, the total number of camera will be stored in $numCam$.

**Parameters:**

$numCam$    The pointer to the variable that will be filled with number of ICP DAS camera scanned.

**Return:**

*RETURN_SUCCESSFUL*          Successfully

**Note:**

This function shall always return *RETURN_SUCCESSFUL*

***IMS_InitCamera***

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_InitCamera(I16 CamID)*

**Visual Basic 6.0**

*IMS_InitCamera (ByVal CamID As Integer) As Integer*

**VC.NET 2005**

*short IMS_InitCamera_SC(Int16 CamID)*

**Description:**

This is Camera initialization function. It is required before configuration setting and image acquiring. Before calling this, You should call ***IMS_ScanCamera( )*** first, to determine how many available cameras.

* Call ***IMS_CloseCamera( )*** to close camera

**Parameters:**

*CamID*      The ID of Camera to be initialized. If ***IMS_ScanCamera( )*** reports 3 available cameras, then valid camera ID are 0,1, and 2

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State.** |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call ***IMS_GetLastError( )*** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_InitCamera_SC

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

> *I16 __stdcall IMS_InitCamera_SC(I16 CamID)*

**Visual Basic 6.0**

> *IMS_InitCamera (ByVal CamID As Integer) As Integer*

**VC.NET 2005**

> *short IMS_InitCamera(Int16 CamID)*

**Description:**

This is Camera initial function and after successfully initialized the camera, the configurations stored inside the data flash will be set into the Camera. The effect is the same as calling IMS_InitCamera() and then following configuration setting functions:

- IMS_Set_VideoMode()
- IMS_Set_AOI_Parameter()
- IMS_Set_Feature() , for all features
- IMS_Set_Trigger()
- IMS_Set_GPIOMode()
- IMS_Set_Strobe()

To modify configurations stored inside data flash, please use **EzView Pro** utility.

**Parameters:**

*CamID*      The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State.** |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|

| | |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### *GetLastError*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_GetLastError(I16 CamID)*

**Visual Basic 6.0**

    *IMS_GetLastError (ByVal CamID As Integer) As Integer*

**VC.NET 2005**

    *short IMS_GetLastError(Int16 CamID)*

**Description:**

When ever you get a **RETURN_CHECK_LASTERROR** in previous function call, use this to retrieve detail error information!

**Parameters:**

*CamID*    The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

**Return:**

The Error code definition as below

| Code | Error Messages | Error Description |
|------|----------------|-------------------|
| **0** | CAM SUCCESS | No Error |
| **-1** | CAM ERROR | Generic Error<br>This error typically indicates some problem from the Windows I/O subsystem. A call to the win32 **GetLastError()** should prove enlightening. |
| **-10** | CAM_ERROR_UNSUPPORTED | The feature or function is not supported. |
| **-11** | CAM_ERROR_NOT_INITIALIZED | The camera is not properly initialized. |
| **-12** | CAM_ERROR_INVALID_VIDEO_SETTINGS | The selected video settings are not supported. |
| **-13** | CAM_ERROR_BUSY | Many functions are NOT allowed while acquiring images, you must stop it first. |
| **-14** | CAM_ERROR_INSUFFICIENT_RESOURCES | Insufficient memory or bus bandwidth is not available to complete the request. |

| -15 | CAM_ERROR_PARAM_OUT_OF_RANGE | Many parameters have been bounds, one of them has been exceeded. |
|-----|------------------------------|------------------------------------------------------------------|
| -16 | CAM_ERROR_FRAME_TIMEOUT | This error indicated that the timeout has expired and no frame is ready. |
| -17 | CAM_ERROR_FEATURE_UNAVAILABLE | This error indicated that feature is not available. |
| -18 | CAM_ERROR_SW_TRIGGER_NOT_SET | This error indicated that the software trigger function is not correctly set. |
| -19 | CAM_ERROR_CANNOT_OPEN_FILE | This error tells the specified file name or path is not correct. |
| -20 | CAM_ERROR_LUT_FILE_DATA_ERROR | This error tells the data inside LUT file are not correct. |

***IMS_CloseCamera***

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_CloseCamera(I16 CamID)*

**Visual Basic 6.0**

    *IMS_CloseCamera (ByVal CamID As Integer) As Integer*

**VC.NET 2005**

    *short IMS_CloseCamera(Int16 CamID)*

**Description:**

This is Camera closing function. Before calling this function, you must stop the image acquiring first. After calling this, you may remove the *CamID* specified camera from 1394 bus. If you plug it again into 1394bus, be sure to call ***IMS_ScanCamera( ).*** And, call ***IMS_InitCamera( )*** again, if you want to operate the same camera.

**Parameters:**

*CamID*    The ID of Camera to be initialized. If ***IMS_ScanCamera( )*** reports 3 available cameras, then valid camera ID are 0,1, and 2

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Check_BusRate

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Check_BusRate(I16 CamID, U16* BusRate)*

**Visual Basic 6.0**

*IMS_Check_BusRate (ByVal CamID As Integer, ByRef BusRate As Integer) As Integer*

**VC.NET 2005**

*short IMS_Check_BusRate(Int16 CamID, IntPtr BusRate)*

**Description:**

This function is used to check current IEEE 1394 bus rate.

**Parameters:**

*CamID*   The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

*BusRate*  Pointer to the variable that will be filled the bus rate information when function successfully returned. If **RETURN_UNAVAILABLE** or **RETURN_CHECK_LASTERROR** returned, *BusRate remains unchanged.

BusRate is in unit of Mbps. So, if *busRate = 400, it means 400 Mbps

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.4 Callback Event

### IMS_Register_Callback

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Register_CallBack(I16 CamID, void\* UserData, void (\_\_stdcall \*CallBackRoutine)(I16 CamID, U8\* pBuffer, U32 size, void\* UserData, U8 TimeOutFlag))*

**Visual Basic 6.0**

*IMS_Register_CallBack (ByVal CamID As Integer, ByVal UserData As Long, ByVal CallBackRoutine As Long) As Integer*

**VC.NET 2005**

*short IMS_Register_CallBack(Int16 CamID, IntPtr UserData, Callback CallBackRoutine)*

**Description:**

This function is used to register callback routine.

**Note:** If you successfully call **IMS_Register_CallBack( )** twice or more, the last call will over-write previous registration.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *CallBackRoutine* | The call back routine. Proto type is as below: |
| | **- void \_\_stdcall CallBackRoutine(I16 CamID, U8\* pBuffer, U32 size, void\* UserData, U8 TimeOutFlag)** |
| | More information about **CallBackRoutine( )**; |
| *UserData* | Additional data that user can specify. This data will be passed to callback routine, when it is called. |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |

| | |
|---|---|
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### *CallbackRoutine*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall CallBackRoutine(I16 CamID, U8\* pBuffer, U32 size, void\* UserData, U8 TimeOutFlag)*

**Visual Basic 6.0 (Callback.bas)**

*CallBackRoutine(ByVal CamID As Integer, ByVal pFrameBuffer As Long, ByVal size As Long, ByVal UserData As Long, ByVal TimeOutFlag As Byte)*

**VC.NET 2005**

*\*Please refer to C# sample*

**Description:**

The callback routine is user defined code that will be called when a frame is ready, or *TimeOut_ms* specified in **IMS_Start_Acquire( )** is expired.

If callback routine is called due to time out, the *TimeOutFlag* will be set 1.

If callback routine is called due to frame ready, the *TimeOutFlag* will be set 0.

Callback routine is registered by **IMS_Register_CallBack()**.

**Parameters:**

*CamID*        The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

*pBuffer*       The starting address of memory buffer where frame data is stored. If *TimeOutFlag* = 1, *pBuffer* will be NULL.

*size*         The size of frame buffer. If *TimeOutFlag* = 1, size will be 0.

*UserData*     Additional user data specified in parameter of **IMS_Register_CallBack( )**.

*TimeOutFlag*  If *TimeOutFlag* = 0, a frame is ready.

            If *TimeOutFlag* = 1, Time out occurs.

            The time out value is specified by *TimeOut_ms* of **IMS_Start_Acquire( ).**

**Return:**

None

## 5.5 Camera Acquisition

*IMS_Start_Acquire*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Start_Acquire(I16 CamID, U32 NumFrames, U32 TimeOut_ms)*

**Visual Basic 6.0**

*IMS_Start_Acquire (ByVal CamID As Integer, ByVal NumFrames As Long, ByVal TimeOut_ms As Long) As Integer*

**VC.NET 2005**

*short IMS_Start_Acquire(Int16 CamID, UInt32 NumFrames, UInt32 TimeOut_ms)*

**Description:**

This function starts image acquiring process. After successfully starting acquisition, the **CallBackRoutine( )** specified by **IMS_Register_CallBack( )** could be triggered.

**The acquiring process stops when:**

- Captured *NumFrames* Frames.
- Time out occurs. In this case, **CallBackRoutine** will be called with *TimeOutFlag* = 1.
- **IMS_Stop_Acquire( )** is successfully called

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *NumFrames* | Specifying the total number of frames that you want to capture. After capturing enough frames, the acquiring process stops automatically. |
| | If *NumFrames* is set 0, it will acquire is unlimited number of frames. Only **IMS_Stop_Acquire( )** can stop it, or time out occurs. |
| *TimeOut_ms* | Specifying the time out value, in milliseconds. If *TimeOut_ms* = 0, the acquiring process will never time out. |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

***IMS_Stop_Acquire***

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Stop_Acquire(I16 CamID)*

**Visual Basic 6.0**

*IMS_Stop_Acquire (ByVal CamID As Integer) As Integer*

**VC.NET 2005**

*short IMS_Stop_Acquire(Int16 CamID)*

**Description:**

This function will stop image acquiring process.

After successfully stop Acquisition, the ***CallBackRoutine( )*** won't be triggered any more.

**Parameters:**

*CamID*    The ID of Camera to be initialized. If ***IMS_ScanCamera( )*** reports 3 available cameras, then valid camera ID are 0,1, and 2

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call ***IMS_GetLastError( )*** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.6 Camera Configuration

*IMS_Set_VideoMode*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Set_VideoMode(I16 CamID, U32 VideoMode)*

**Visual Basic 6.0**

*IMS_Set_VideoMode (ByVal CamID As Integer, ByVal VideoMode As Long) As Integer*

**VC.NET 2005**

*short IMS_Set_VideoMode(Int16 CamID, UInt32 VideoMode)*

**Description:**

This function is used to set the video mode.

**Available Video Modes**

➢ **640 x 480**

VM_640X480_MONO8_30FPS, VM_640X480_MONO16_30FPS, VM_640X480_RGB_30FPS

VM_640X480_MONO8_15FPS, VM_640X480_MONO16_15FPS, VM_640X480_RGB_15FPS

VM_640X480_MONO8_7.5FPS, VM_640X480_MONO16_7.5FPS, VM_640X480_RGB_7.5FPS

➢ **800 x 600**

VM_800X600_MONO8_30FPS, VM_800X600_MONO16_30FPS, VM_800X600_RGB_30FPS

VM_800X600_MONO8_15FPS, VM_800X600_MONO16_15FPS, VM_800X600_RGB_15FPS

VM_800X600_MONO8_7.5FPS, VM_800X600_MONO16_7.5FPS, VM_800X600_RGB_7.5FPS

➢ **1024 x 768**

VM_1024X768_MONO8_30FPS, VM_1024X768_MONO16_30FPS,

VM_1024X768_MONO8_15FPS, VM_1024X768_MONO16_15FPS, VM_1024X768_RGB_15FPS

VM_1024X768_MONO8_7.5FPS, VM_1024X768_MONO16_7.5FPS, VM_1024X768_RGB_7.5FPS

➢ **1280 x 960**

VM_1280X960_MONO8_30FPS,

VM_1280X960_MONO8_15FPS, VM_1280X960_MONO16_15FPS, VM_1280X960_RGB_15FPS

VM_1280X960_MONO8_7.5FPS, VM_1280X960_MONO16_7.5FPS, VM_1280X960_RGB_7.5FPS

**VM_AOI**

**Parameters:**

*CamID*       The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras,
              then valid camera ID are 0,1, and 2

*VideoMode*  The target video mode to set


**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### *IMS_Get_VideoResolution*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

  *stdcall IMS_Get_VideoResolution(I16 CamID, U32 *Width, U32 *Height)*

**Visual Basic 6.0**

  *IMS_Get_VideoResolution (ByVal CamID As Integer, ByRef Width As Long, ByRef Height As Long) As Integer*

**VC.NET 2005**

  *short IMS_Get_VideoResolution(Int16 CamID, IntPtr Width, IntPtr Height)*

**Description:**

This function is used to get the video dimension, width and height

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Width* | Pointer to the variable that will be filled with video frame width information, in pixel |
| *Height* | Pointer to the variable that will be filled with video frame height information, in pixel |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Get_ColorFormat

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Get_ColorFormat(I16 CamID, U32 \*Color)*

**Visual Basic 6.0**

*IMS_Get_ColorFormat (ByVal CamID As Integer, ByRef Color As Long) As Integer*

**VC.NET 2005**

*short IMS_Get_ColorFormat(Int16 CamID, IntPtr Color)*

**Description:**

This function is used to get the video color format

If successfully returned, the Color will be filled the color format information.

**Color Format**

- VM_MONO8: Black & white 8-bit mode. Every pixel is 1 byte.
- VM_MONO16: Black & white 16-bit mode. Every pixel is 2 bytes.
- VM_RGB: Color 24-bit mode. Every pixel has 3 byte, for Red, Green & Blue individually

**Parameters:**

*CamID*      The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

*Color*      Pointer to the variable that will be filled with video Color information: VM_MONO8, VM_MONO16 or VM_RGB

**Return:**

*RETURN_SUCCESSFUL*      Successfully

*RETURN_UNAVAILABLE*      The *CamID* is invalid or this function is not available in **Current Camera State**.

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Set_AOI_Parameter

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Set_AOI_Parameter(I16 CamID, U32 Color, F32 FrameRate, U16 Left, U16 Top, U16 Width, U16 Height)*

**Visual Basic 6.0**

*IMS_Set_AOI_Parameter (ByVal CamID As Integer, ByVal Color As Long, ByVal FrameRate As Single, ByVal Left As Integer, ByVal Top As Integer, ByVal Width As Integer, ByVal Height As Integer) As Integer*

**VC.NET 2005**

*short IMS_Set_AOI_Parameter(Int16 CamID, UInt32 Color, Single FrameRate, UInt16 Left, UInt16 Top, UInt16 Width, UInt16 Height)*

**Description:**

Before calling this, you should set **VM_AOI** in ***IMS_Set_VideoMode*** first,

This function is used to set parameters for AOI, Area of Interest function.

The *Color* parameter specify for the color format.

- VM_MONO8: Black & white 8-bit mode. Every pixel is 1 byte.
- VM_MONO16: Black & white 16-bit mode. Every pixel is 2 bytes.
- VM_RGB: Color 24-bit mode. Every pixel has 3 byte, for Red, Green & Blue individually.

The *FrameRate* parameter specify for the target frame rate. For example: *FrameRate* = 15.0 means 15.0 fps.

**NOTE: If target *FrameRate* is out of possible range, it will be automatically converted to the closest allowable frame rate.**

Use ***IMS_Get_AOI_Parameter( )*** to check actual frame rate.

The *Left, Top, Width, Height* parameters are specify AOI region. Please keep in mind that the possible values have some limitations

- Left value must be 2*N, where N is an integer.
- Top value must be 2*N, where N is an integer.
- Width value must be 8*N, where N is an integer.
- Height value must be 2*N, where N is an integer.
- Left + Width <= Full image width
- Top + Height <= Full image height

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Color* | The video Color format: VM_MONO8, VM_MONO16 or VM_RGB. |
| *FrameRate* | Target Frame rate in FPS, for example: 15.0 |
| *Left* | AOI starting X position |
| *Top* | AOI starting Y position |
| *Width* | AOI horizontal size |
| *Height* | AOI vertical size |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Get_AOI_Parameter

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Get_AOI_Parameter(I16 CamID, U32 \*Color, F32 \*FrameRate, U16 \*Left, U16 \*Top, U16 \*Width, U16 \*Height)*

**Visual Basic 6.0**

*IMS_Get_AOI_Parameter (ByVal CamID As Integer, ByRef Color As Long, ByRef FrameRate As Single, ByRef Left As Integer, ByRef Top As Integer, ByRef Width As Integer, ByRef Height As Integer) As Integer*

**VC.NET 2005**

*short IMS_Get_AOI_Parameter(Int16 CamID, IntPtr Color, IntPtr FrameRate, IntPtr Left, IntPtr Top, IntPtr Width, IntPtr Height)*

**Description:**

Before calling this, you should set **VM_AOI** in *IMS_Set_VideoMode* first,

This function is used to read back current parameter values for AOI, Area Of Interest, function.

Please refer to *IMS_Set_AOI_Parameter( )*.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Color* | Pointer to the variable that will be filled with video Color information: VM_MONO8, VM_MONO16 or VM_RGB |
| *FrameRate* | Pointer to the variable that will be filled with current Frame rate in FPS |
| *Left* | Pointer to the variable that will be filled with current AOI starting X position |
| *Top* | Pointer to the variable that will be filled with current AOI starting Y position |
| *Width* | Pointer to the variable that will be filled with current AOI horizontal size |
| *Height* | Pointer to the variable that will be filled with current AOI vertical size |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |

| | |
|---|---|
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## *IMS_Set_Feature*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Set_Feature(I16 CamID, U32 FeatureID, F32 FeatureValue, bool FeatureOnOff)*

**Visual Basic 6.0**

*IMS_Set_Feature (ByVal CamID As Integer, ByVal FeatureID As Long, ByVal FeatureValue As Single, ByVal FeatureOnOff As Boolean) As Integer*

**VC.NET 2005**

*short IMS_Set_Feature(Int16 CamID, UInt32 FeatureID, Single FeatureValue, bool FeatureOnOff)*

**Description:**

This function is used to set Features of ICPDAS Cameras.

The *FeatureID* parameter specify for which feature to be set a new value. Available features including:

| *FeatureID* | Value range | unit |
|---|---|---|
| **CAMERA_FEATURE_BRIGHTNESS** | 0.0 ~ 6.2439 (default 3.997803) | % |
| **CAMERA_FEATURE_SHARPNESS** | 0 ~ 4095 (default 2048) | - |
| **CAMERA_FEATURE_WHITE_BALANCE_R** | 0 ~ 1023 (default 600) | - |
| **CAMERA_FEATURE_WHITE_BALANCE_B** | 0 ~ 1023 (default 840) | - |
| **CAMERA_FEATURE_HUE** | -180.0 ~ 179.912 (default 0) | Degree |
| **CAMERA_FEATURE_SATURATION** | 0.0 ~ 399.902 (default 136.9141) | % |
| **CAMERA_FEATURE_GAMMA** | 0.50 ~ 3.99902 (default 0.9990234) | - |
| **CAMERA_FEATURE_SHUTTER** | 4.65e-6 ~ 0.133263 or 1/fps | us / ms |
| **CAMERA_FEATURE_GAIN** | -6.26386 ~ 24.0001 (default 0) | dB |

The *FeatureValue* parameter specify for the feature value. Its may call failed when user set the value out of range.

The *FeatureOnOff* parameter is used to turn on or off the feature.

Note- **CAMERA_FEATURE_BRIGHTNESS** feature is always ON, so, you can't turn it off.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If ***IMS_ScanCamera( )*** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *FeatureID* | The camera features |
| *FeatureValue* | The value of feature to be set |
| *FeatureOnOff* | True: turn on this feature |
| | False: turn off this feature |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call ***IMS_GetLastError( )*** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Get_Feature

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Get_Feature(I16 CamID, U32 FeatureID, F32 *FeatureValue, bool *FeatureOnOff)*

**Visual Basic 6.0**

*IMS_Get_Feature (ByVal CamID As Integer, ByVal FeatureID As Long, ByRef FeatureValue As Single, ByRef FeatureOnOff As Boolean) As Integer*

**VC.NET 2005**

*short IMS_Get_Feature(Int16 CamID, UInt32 FeatureID, IntPtr FeatureValue, IntPtr FeatureOnOff)*

**Description:**

This function is used to get current value of specified features.

The *FeatureID* parameter specify for which feature value to read. Available features including:

| FeatureID | Value range | unit |
|---|---|---|
| **CAMERA_FEATURE_BRIGHTNESS** | 0.0 ~ 6.2439 | % |
| **CAMERA_FEATURE_SHARPNESS** | 0 ~ 4095 | - |
| **CAMERA_FEATURE_WHITE_BALANCE_R** | 0 ~ 1023 | - |
| **CAMERA_FEATURE_WHITE_BALANCE_B** | 0 ~ 1023 | - |
| **CAMERA_FEATURE_HUE** | -180.0 ~ 179.912 | Degree |
| **CAMERA_FEATURE_SATURATION** | 0.0 ~ 399.902 | % |
| **CAMERA_FEATURE_GAMMA** | 0.50 ~ 3.99902 | - |
| **CAMERA_FEATURE_SHUTTER** | 4.65e-6 ~ 0.133263 or 1/fps | us / ms |
| **CAMERA_FEATURE_GAIN** | -6.26386 ~ 24.0001 | dB |

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *FeatureID* | The camera features |
| *FeatureValue* | Pointer to the variable that will be filled with current feature value |
| *FeatureOnOff* | Pointer to the variable that will be filled with current feature on/off state |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

*RETURN_CHECK_LASTERROR*      Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code

**Current Camera State:**

| Camera State | Description |
| --- | --- |
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.7 Trigger Setting

*IMS_Set_Trigger*

### Syntax:

### Visual C++ 6.0 / Boland C++ Builder 6.0:

*stdcall IMS_Set_Trigger(I16 CamID, bool On_Off, U16 TriggerSource, U16 TriggerMode)*

### Visual Basic 6.0

*IMS_Set_Trigger (ByVal CamID As Integer, ByVal on_off As Boolean, ByVal TriggerSource As Integer, ByVal TriggerMode As Integer) As Integer*

### VC.NET 2005

*short IMS_Set_Trigger(Int16 CamID, bool On_Off, UInt16 TriggerSource, UInt16 TriggerMode)*

### Description:

This function is to configure the trigger function.

The *On_Off* parameter decides that the trigger function is to be enabled (on_off = true) or disabled (on_off = false).

*TriggerSouce* specify the triggering source. Possible triggering source are:

- *TriggerSource* = 0, falling edge (change from 3.3V or 5V to 0V) of GPIO pin 0
- *TriggerSource* = 1, rising edge (change from 0V to 3.3V or 5V) of GPIO pin 0
- *TriggerSource* = 15, Software controlled trigger. Refer to **IMS_Do_SWTrigger( )**

There are 2 triggering modes specified by *TriggerMode* = 0 or 1.

- *TriggerMode* 0: Standard External Trigger Mode.

    The camera starts integration of the incoming light from external trigger input falling/rising edge.

    The **CAMERA_FEATURE_SHUTTER** feature controls the integration time.

- *TriggerMode* 1: Bulb Shutter mode.

    The camera integration time is equal to low/high state time of the external trigger input.

- If trigger source is set to software (*TriggerSource* =15), only standard External Trigger Mode is available, even *TriggerMode* is set 1

### Parameters:

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *On_Off* | True: Enable trigger function |
| | False: disable trigger function |
| *TriggerSouce* | The source of trigger input. Also specify rising or falling edge. |

*TriggerMode*        Specify the trigger mode. 0 for Standard External Trigger Mode, 1 for Bulb Shutter mode.

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call ***IMS_GetLastError( )*** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### *IMS_Get_Trigger*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_Get_Trigger(I16 CamID, bool \*On_Off, U16 \*TriggerSource, U16 \*TriggerMode)*

**Visual Basic 6.0**

    *IMS_Get_Trigger (ByVal CamID As Integer, ByRef on_off As Boolean, ByRef TriggerSource As Integer, ByRef TriggerMode As Integer) As Integer*

**VC.NET 2005**

    *short IMS_Get_Trigger(Int16 CamID, IntPtr On_Off, IntPtr TriggerSource, IntPtr TriggerMode)*

**Description:**

This function is to get current configure of trigger function. Please refer to **IMS_Set_Trigger( )**

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *On_Off* | Pointer to the variable that will be filled with current on/off status |
| *TriggerSouce* | Pointer to the variable that will be filled with current source of trigger input. Also specify rising or falling edge |
| *TriggerMode* | Pointer to the variable that will be filled with current trigger mode. |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

**IMS_Do_SWTrigger**


**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

   *stdcall IMS_Do_SWTrigger(I16 CamID)*

**Visual Basic 6.0**

   *IMS_Do_SWTrigger (ByVal CamID As Integer) As Integer*

**VC.NET 2005**
   *short IMS_Do_SWTrigger(Int16 CamID)*


**Description:**

This function is used to generate a software asynchronous trigger.

Please refer to **IMS_Set_Trigger( )**. This call only works when TriggerSource is set to 15. You still need to successfully call **IMS_Start_Acquire( )** before I**MS_Do_SWTrigger( )**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.8 GPIO Control

### IMS_Set_GPIOMode

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Set_GPIOMode(I16 CamID, U16 Channel, U16 Mode)*

**Visual Basic 6.0**

*IMS_Set_GPIOMode (ByVal CamID As Integer, ByVal Channel As Integer, ByVal Mode As Integer) As Integer*

**VC.NET 2005**

*short IMS_Set_GPIOMode(Int16 CamID, UInt16 Channel, UInt16 Mode)*

**Description:**

This function is used to set the GPIO mode. Every GPIO channel can be independently chosen to work and definition as below:

- General purposed input (*Mode* = 0)
- General purposed output (*Mode* = 1)
- Strobe Output with High active output (*Mode* = 2)
- Strobe Output with Low active output (*Mode* = 3)

The *Channel* parameter specify for which GPIO channel to be set.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Channel* | The Channel ID, value = 1 ~ 3 for GPIO pin 1 ~ 3 individually |
| *Mode* | The Mode of GPIO |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |

| | |
|---|---|
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Get_GPIOMode

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Get_GPIOMode(I16 CamID, U16 Channel, U16 *Mode)*

**Visual Basic 6.0**

*IMS_Get_GPIOMode (ByVal CamID As Integer, ByVal Channel As Integer, ByRef Mode As Integer) As Integer*

**VC.NET 2005**

*short IMS_Get_GPIOMode(Int16 CamID, UInt16 Channel, IntPtr Mode)*

**Description:**

This function is used to get current GPIO mode setting for specified *Channel*. Please refer to **IMS_Set_GPIOMode( )**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Channel* | The Channel ID, value = 1 ~ 3 for GPIO pin 1 ~ 3 individually |
| *Mode* | Pointer to the variable that will be filled with current mode setting |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_GPIO_Read

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_GPIO_Read(I16 CamID, U16 Channel, bool \*Value)*

**Visual Basic 6.0**

*IMS_GPIO_Read (ByVal CamID As Integer, ByVal Channel As Integer, ByRef value As Boolean) As Integer*

**VC.NET 2005**

*short IMS_GPIO_Read(Int16 CamID, UInt16 Channel, IntPtr Value)*

**Description:**

When GPIO 1~3 is set as general purposed input or output mode, you can use this function to read its value.

For 3.3V or 5V input, *Value* will be true. For 0 V input, *Value* will be false.

For GPIO mode setting, please refers to **IMS_Set_GPIOMode()**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Channel* | The Channel ID, value = 1 ~ 3 for GPIO pin 1 ~ 3 individually |
| *Value* | Pointer to the variable that will be filled with current value of GPIO pin. |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_GPIO_Write

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_GPIO_Write(I16 CamID, U16 Channel, bool Value)*

**Visual Basic 6.0**

*IMS_GPIO_Write (ByVal CamID As Integer, ByVal Channel As Integer, ByVal value As Boolean) As Integer*

**VC.NET 2005**

*short IMS_GPIO_Write(Int16 CamID, UInt16 Channel, bool Value)*

**Description:**

When GPIO 1~3 is set as general purposed output mode, you can use this function to set its value.

For 3.3V output, set *Value* true. For 0 V output, set *Value* false.

For GPIO mode setting, please refers to **IMS_Set_GPIOMode( )**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Channel* | The Channel ID, value = 1 ~ 3 for GPIO pin 1 ~ 3 individually |
| *Value* | Output value of GPIO pin. false for 0V output |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.9 Strobe Control

### IMS_Set_Strobe

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Set_Strobe(I16 CamID, F32 Delay_ms, F32 Duration_ms)*

**Visual Basic 6.0**

*IMS_Set_Strobe (ByVal CamID As Integer, ByVal Delay_ms As Single, ByVal Duration_ms As Single) As Integer*

**VC.NET 2005**

*short IMS_Set_Strobe(Int16 CamID, Single Delay_ms, Single Duration_ms)*

**Description:**

This function is used to set the Strobe parameter.

The *Delay_ms* define the time period in ms between beginning of exposure and strobe signal output.

The *Duration_ms* define the length of strobe signal in mini second.

You can select which GPIO pin(s) to output strobe signal by using **IMS_Set_GPIOMode( )**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Delay_ms* | The delay time between beginning of exposure and strobe signal output. Range: 0 ~ 10,000.0 ms |
| *Duration_ms* | The duration of strobe signal. Range: 0 ~ 10,000.0 ms |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |

CAM_State_2_INIT_NOT_ACQUIRING          Camera was initialized, but not in acquiring

CAM_State_3_ACQUIRING_IMAGE             Camera is acquiring image

### IMS_Get_Strobe

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Get_Strobe(I16 CamID, F32 *Delay_ms, F32 *Duration_ms)*

**Visual Basic 6.0**

*IMS_Get_Strobe (ByVal CamID As Integer, ByRef Delay_ms As Single, ByRef Duration_ms As Single)*
*As Integer*

**VC.NET 2005**

*short IMS_Get_Strobe(Int16 CamID, IntPtr Delay_ms, IntPtr Duration_ms)*

**Description:**

This function is used to get current Strobe parameter. Please refer to **IMS_Set_Strobe( )**.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Delay_ms* | Pointer to the variable that will be filled with delay time value |
| *Duration_ms* | Pointer to the variable that will be filled with duration value |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.10     Lookup Table Control

*IMS_Set_LUT*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

> *stdcall IMS_Set_LUT(I16 CamID, char\* LUT_FileName)*

**Visual Basic 6.0**

> *IMS_Set_LUT (ByVal CamID As Integer, ByVal LUT_FileName As String) As Integer*

**VC.NET 2005**

> *short IMS_Set_LUT(Int16 CamID, string LUT_FileName)*

**Description:**

This function is used to set LUT (Look up Table).

The *LUT_FileName* specify the LUT file name, including path. If not specify the path, current directory is searched.

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *LUT_FileName* | File path and name of LUT file |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### *IMS_Get_LUT*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_Get_LUT(I16 CamID, char* LUT_FileName)*

**Visual Basic 6.0**

    *IMS_Get_LUT (ByVal CamID As Integer, ByVal LUT_FileName As String) As Integer*

**VC.NET 2005**

    *short IMS_Get_LUT(Int16 CamID, string LUT_FileName)*

**Description:**

This function is used to get current contents of LUT (Look up Table).

The *LUT_FileName* specify the LUT file name, including path. If not specify the path, current directory is used

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *LUT_FileName* | File path and name of LUT file |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.11 Data Flash Control

*IMS_Read_DataFlash*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Read_DataFlash(I16 CamID, U32 Address, U8 *Data)*

**Visual Basic 6.0**

*IMS_Read_DataFlash (ByVal CamID As Integer, ByVal Address As Long, ByRef Data As Byte) As Integer*

**VC.NET 2005**

*short IMS_Read_DataFlash(Int16 CamID, UInt32 Address, IntPtr Data)*

**Description:**

This function is used to read data flash value of Sparrow IMS camera.

The *Address* specify the byte address of data flash to be read, range from 0x0 ~ 0xFFFF (total 64K)

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Address* | Byte address |
| *Data* | Pointer to the variable that will be filled with read value |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

### IMS_Write_DataFlash

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall IMS_Write_DataFlash(I16 CamID, U32 Address, U8 Data)*

**Visual Basic 6.0**

*IMS_Write_DataFlash (ByVal CamID As Integer, ByVal Address As Long, ByVal Data As Byte) As Integer*

**VC.NET 2005**

*short IMS_Write_DataFlash(Int16 CamID, UInt32 Address, Byte Data)*

**Description:**

This function is used to write a byte value to data flash of Sparrow IMS camera.

The *Address* specify the byte address of data flash to write, range from 0x0 ~ 0xFFFF (total 64K)

**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *Address* | Byte address |
| *Data* | The Data value to write to flash |

**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |

**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

## 5.12    AES Project Encryption

*IMS_Get_CameraInform*


**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

    *stdcall IMS_Get_CameraInform(I16 CamID, PCAM_INFO CamInfo)*

**Visual Basic 6.0**

    *IMS_Get_CameraInform (ByVal CamID As Integer, CamInfo As CAM_INFO) As Integer*

**VC.NET 2005**

    *short IMS_Get_CameraInform(Int16 CamID, ref CAM_INFO CamInfo)*


**Description:**

This function is used to get camera model name and unique serial number.

This information is retrieved by CAM_INFO struct, which is composed of 2 members.

1st member: char ModelName[32], store the Camera Model name

2nd member: U8 SerialNumber[16], store the unique Serial Number


**Parameters:**

| | |
|---|---|
| *CamID* | The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2 |
| *CamInfo* | Pointer to the CAM_INFO struct that will be filled with camera information, the model name & unique serial number |


**Return:**

| | |
|---|---|
| *RETURN_SUCCESSFUL* | Successfully |
| *RETURN_UNAVAILABLE* | The *CamID* is invalid or this function is not available in **Current Camera State**. |
| *RETURN_CHECK_LASTERROR* | Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code |


**Current Camera State:**

| Camera State | Description |
|---|---|
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

***AES_GET_REG***

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

*stdcall AES_GET_REG(char raw[16],char AES_Key[16],char Encrypted[16])*

**Visual Basic 6.0**

*AES_GET_REG (ByVal raw As Variant, ByVal AES_Key As Variant, ByVal Encrypted As Variant) As Long*

**VC.NET 2005**

*long AES_GET_REG(Byte[] raw, Byte[] AES_Key, IntPtr Encrypted)*

**Description:**

**This function is used compute AES encryption SN code.**

**Parameters:**

*Raw*  The raw data to be encrypted. Size of raw is 16 bytes

*AES_Key*  The AES Key. Size of AES_Key is 16 bytes

*Encrypted*  The AES encrypted result. Size of Encrypted is 16 bytes

**Return:**

*RETURN_SUCCESSFUL*  Successfully

## 5.13　Test Pattern

*IMS_Enable_TestImage*

**Syntax:**

**Visual C++ 6.0 / Boland C++ Builder 6.0:**

   *stdcall IMS_Enable_TestImage(I16 CamID, bool On_Off)*

**Visual Basic 6.0**

   *IMS_Enable_TestImage (ByVal CamID As Integer, ByVal on_off As Boolean) As Integer*

**VC.NET 2005**

   *short IMS_Enable_TestImage(Int16 CamID, bool On_Off)*

**Description:**

This function is to enable or disable the test pattern output

**Parameters:**

*CamID*　　　　　　The ID of Camera to be initialized. If **IMS_ScanCamera( )** reports 3 available cameras, then valid camera ID are 0,1, and 2

*On_Off*　　　　　　true: Enable test pattern output, false: disable pattern output

**Return:**

*RETURN_SUCCESSFUL*　　　　　　Successfully

*RETURN_UNAVAILABLE*　　　　　　The *CamID* is invalid or this function is not available in **Current Camera State**.

*RETURN_CHECK_LASTERROR*　　Some error occurs, please call **IMS_GetLastError( )** to retrieve the error code
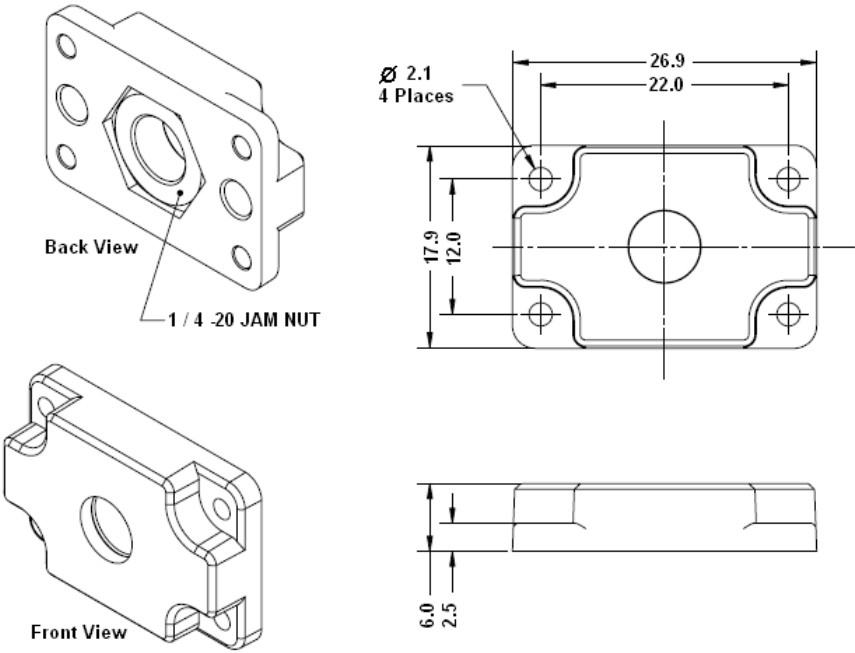
**Current Camera State:**

| Camera State | Description |
| --- | --- |
| CAM_State_0_NOT_SCANNED | Camera not scanned. |
| CAM_State_1_NOT_INIT | Camera was scanned, but not initialized yet. |
| CAM_State_2_INIT_NOT_ACQUIRING | Camera was initialized, but not in acquiring |
| CAM_State_3_ACQUIRING_IMAGE | Camera is acquiring image |

# 6. Mechanical

## 6.1 Sparrow IMS Camera



4−M2x2mm DEEP
3−M3x2.5mm DEEP

8−PIN GPIO CONNECTOR

STATUS LED

IEEE−1394B CONNECTOR WITH JACKSCREWS

## 6.2 Tripod Adapter

Back View

1/4 -20 JAM NUT

Front View

Ø 2.1
4 Places

26.9

22.0

17.9

12.0

6.0

2.5
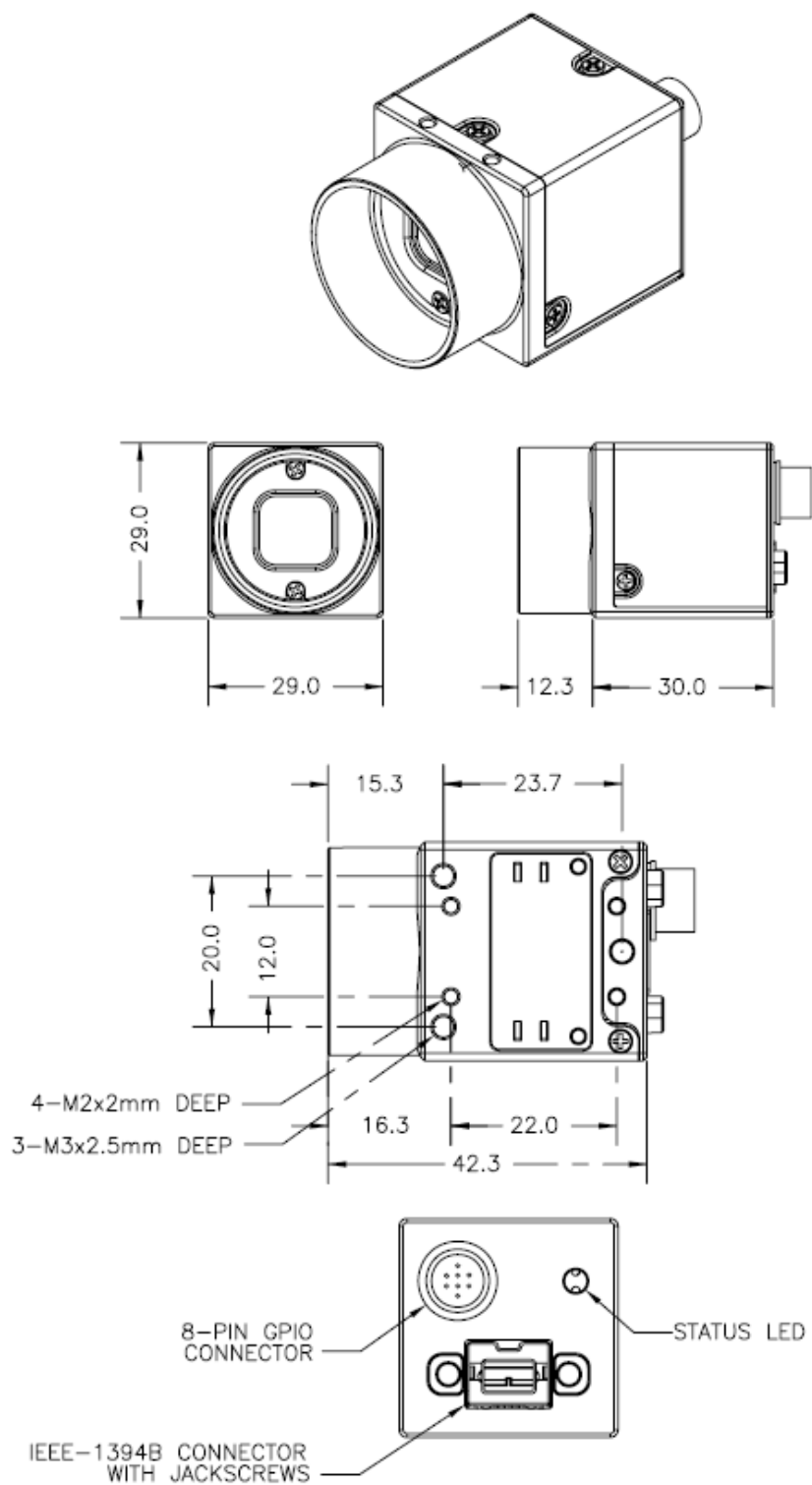
# 7. Appendix

## 7.1 Standards Compliance

FCC

**For customers in the U.S.A.**

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at own expense. You are cautioned that any changes or modifications not expressly approved in this manual could void your authority to operate this equipment. The shielded interface cable recommended in this manual must be used with this equipment in order to comply with the limits for a computing device pursuant to Subpart J of Part 15 of FCC Rules.

**For customers in Canada**

This apparatus complies with the Class A limits for radio noise emissions set out in the Radio Interference Regulations.

**Pour utilisateurs au Canada**

Cet appareil est conforme aux normes classe A pour bruits radioélectriques, spécifiées dans le Règlement sur le brouillage radioélectrique.

**Life support applications**

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Allied customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Allied for any damages resulting from such improper use or sale.

CE

The equipment was passed the test performed according to:

European Standard EN 55022:1998/A1:2000/A2:2003 Class A, EN 61000-3-2:2000, EN 61000-3-3:1995/A1:2001, EN 55024:1998/A1:2001/A2:2003(IEC 61000-4-2:1995/A2:2000, IEC 61000-4-3:2002, IEC 61000-4-4:1995/A2:2001, IEC 61000-4-5:1995/A1:2000, IEC 61000-4-6:1996/A1:2000, IEC 61000-4-8:1993/A1:2000, IEC 61000-4-11:1994/A1:2000)

## 7.2 Glossary

**AOI** is means the area of interest. A user-defined, rectangular area (a square is common) on a CCD that is exposed and processed as an image. For image processing field, the AOI also means a user-defined area for inspection or measurement application for saving system images processing time.
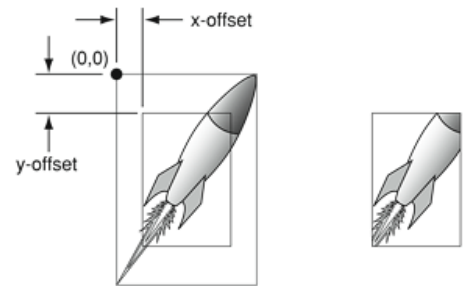


**Bit Depth -**The number of bits used to code a value (such as a pixel component) into an integer value. This is directly related to the number of levels that the value might have, such as 256 with an 8-bit depth or 1,024 with a 10-bit depth.
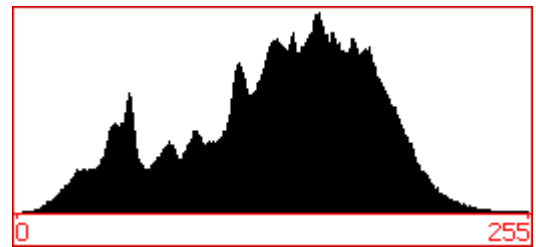


**CMOS (Complementary Metal-Oxide Semiconductor**) – CMOS is a widely used type of semiconductor. CMOS semiconductors use both NMOS (negative polarity) and PMOS (positive polarity) circuits. Since only one of the circuit types is on at any given time, CMOS chips require less power than chips using just one type of transistor. CMOS traditionally consumes little power and can be fabricated on just about any standard silicon production line, so they tend to be extremely inexpensive compared to CCD sensors.
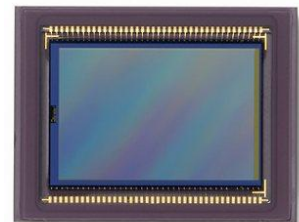


**FireWire/1394** - The 1394 digital link standard was conceived in 1986 by technologists at Apple Computer, who chose the trademark 'FireWire', in reference to its speeds of operation. In 1995, the IEEE (Institute of Electrical and Electronic Engineers) defined this standard is IEEE 1394. There are two IEEE 1394 standards in current market; 1394a for data transmission rates up to 400Mbps, and the other is IEEE 1394b; for data transmission rates up to 800Mbps.
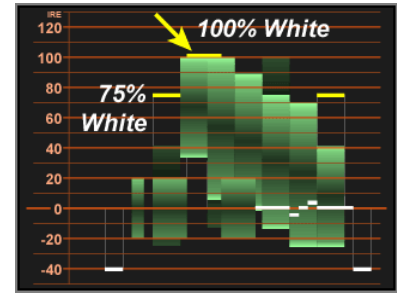
**Gain** is the level of amplification of a signal. The Gain affects image sharpness sensed by the naked eye. When gain is increased in an image, one must also take into consideration the addition of significant "noise".



**Pixel Size -** Most CCD and CMOS imagers consist of picture elements dubbed "pixels". Each pixel is one sensor within the array and has a definite size, which should be available by the manufacturer. Sizes typically range from 8-20 microns. The pixel size is a technical parameter that relates to resolution, process feature dimensions and pixel architecture. For a given die size, a high resolution requires a small pixel
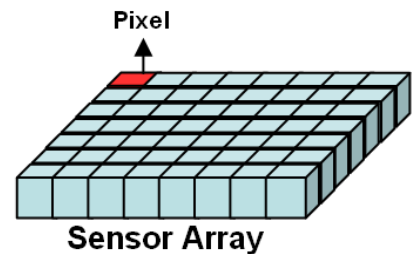


**Progressive-Scan -** A system of video scanning whereby lines of a picture are transmitted consecutively, such as in the computer world. This method is often used in DVD video encoding where the video is produced by scanning the film. It is also used in enhanced and high definition television systems as it is supposed to produce less visual artifacts than the interlaced mode but requires a higher refreshing rate.
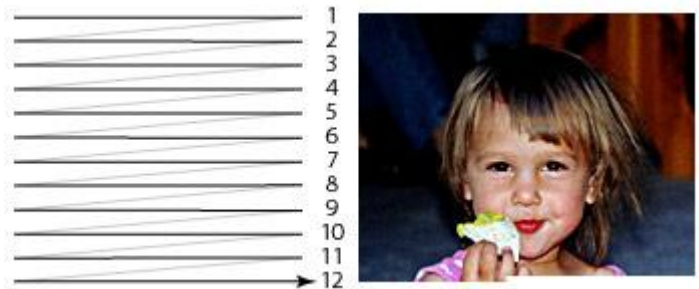


**Sensor Size** - The "Sensor Size" is the dimensions of CCD/CMOS sensor area, which is responsible for transforming light into electrical signals. Typically, the sensor size from 1/4" to 1" and measured by diagonal size.
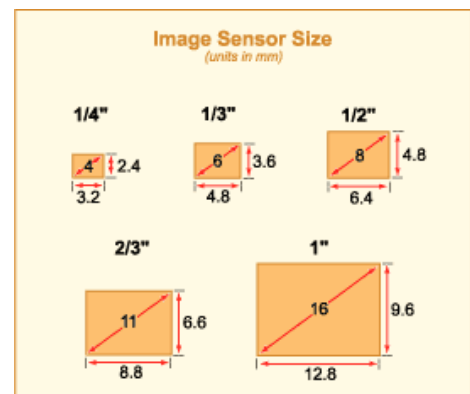
## 7.3 Revision History

| Revision | Revision Date | Change Description |
|----------|---------------|--------------------|
| 1.2 | 2010/7/15 | Add default values of camera features. Add an additional remark for EzView_Pro. |
| | | |
| | | |

# 8. Warranty Policy

ICP DAS supplies a one year warranty period for the Sparrow IMS IEEE 1394 camera, however there certain instances of limited of warranty situations, whereby ICP DAS will not take any responsibility in the following cases:

1. When user open camera housing then warranty will void immediately.

2. In case damages or losses are caused by fire, earthquake, or other acts of the Gods, the act by third party, misuse by the user deliberately or erroneously, use under extreme operating conditions.

3. In case damages or losses are caused by malfunction resulting from bad connection with other equipment.

4. In case damages or losses are caused by incorrect use which is not in line with instruction in user's manual.

5. In case indirect, additional, consequential damages (loss of expected interest, suspension of business activities) are incurred as results of malfunction or non-function of the equipment, we shall be exempted from assuming responsibility for such damages.

# 9. ICP DAS Worldwide

## Headquarters

**ICP DAS CO., LTD.**

No.111, Kuangfu N. Rd., Hukou Shiang,

Hsinchu Hsien, Taiwan 303, R.O.C

TEL: +886-3-597-3366

FAX: +886-3-597-3733

service@icpdas.com

## Taiwan Branch Office

**Ban-Ciao**

16F-1, No.33, Sec. 1, Minson Road, Banciao

City, Taipei Hsien, Taiwan 220, R.O.C

TEL: +886-2-2950-0655

FAX:+886-2-2950-0807

banciao@icpdas.com

**Hsin-Tien**

7F-2, No. 137, Lane 235, Bao-Chiao R., Hsin-

Tien City, Taipei Hsien, Taiwan 231, R.O.C

TEL : (02)8919-2216

FAX : (02)8919-2221

hsintien@icpdas.com

**Tai-Chung**

9F-6, No.123, Sec. 3, Zhong-Gang Road,

Tai-Chung City, Taiwan 407, R.O.C

TEL : (04)2358-2815

FAX : (04)2358-9114

taichung@icpdas.com

**Kao-Hsiung**

3F, No. 505, Zhong-Shan second Road,

Kao-Hsiung City, Taiwan 801, R.O.C

TEL : (07)215-7688

FAX : (07)216-2602

kaoshiung@icpdas.com

## USA Branch Office

**ICP DAS USA, Inc.**

2531 West 237th Street, Suite 121

Torrance, CA 90505, USA

TEL: 1-310-517-9888

FAX: 1-310-517-0998

Sales@icpdas-usa.com

## Europe Branch Office

**ICPDAS-EUROPE GmbH**

Humboldtstrasse 36

70771 Leinfelden-Echterdingen

Germany

TEL: 0049-711-9 97 37 75

FAX: 0049-711-9 97 37 84

info@icpdas-europe.com

## China Branch Office

**Beijing**

TEL : 86-10-6298-0924

FAX : 86-10-6296-2890

beijing@icpdas.com.cn

**Shanghai**

TEL : 86-21-6247-1722

FAX : 86-21-6247-1725

shanghai@icpdas.com.cn

**Wuhan**

TEL : 86-27-8548-3302

**Kunming**

TEL : 86-13113689519

　　　86-87-1294-5396