# SMX-150 Camera Visual Basic 6.0 API Reference

SMX-150 Camera Visual Basic 6.0 API Reference



3403 Southwood Dr., Oceanside, CA 92054

Email: info@sumix.com

www.sumix.com

# Contents

# Tables

# SMX-150 camera Visual Basic 6.0 API reference.

## *Introduction*

SMX-150 camera is supported with Visual Basic 6.0 by employing ActiveX technology. Represented ActiveX component allows developers to control various camera parameters, to grab frames in video and snapshot mode, etc.

## *SMX-150 Camera ActiveX Properties*

**Table 1. ActiveX Properties**

| Name | Type | Access | Description |
|---|---|---|---|
| Connected | Boolean | r | Allows determining physical camera presence in the system. This property will have TRUE value if camera connected and recognized by ActiveX component and FALSE otherwise |
| CameraType | Integer | r | Allows determining type of the camera connected:<br>0 – SMX-150M<br>1 – SMX-150C |
| CameraColorDeep | Integer | r | This property shows the color depth of the stream currently available from camera. For this version of ActiveX component this property will have value 8 for SMX-150M monochrome camera or 24 for SMX-150C color camera |
| CameraSerialNo | Integer | r | Serial Number of the camera |
| FrameMaxWidth | Integer | r | Max width in pixels of video frame |
| FrameMaxHeight | Integer | r | Max height in pixels of video frame |
| Enabled | Boolean | r/w | Enables the ActiveX component. ActiveX component will not communicate to camera unless this property is set to TRUE |
| Visible | Boolean | r/w | Allows ActiveX component to display its contents on the Visual Basic form |
| DeviceID | Integer | r/w | Selected number of camera (0..9). The first camera connected to the system will have number 0, next one will have number 1 and so on. Depending on what camera ActiveX component needs to communicate to it is possible to set different values for this property. If DeviceID is set to value corresponding to camera not physically present in system, than Connected property will be set to |

| Name | Type | Access | Description |
|---|---|---|---|
| | | | FALSE automatically |
| Started | Integer | r/w | Setting this property to any non-zero value will instruct ActiveX component to configure camera for video streaming operations. Setting this property to zero will stop the video streaming operations |
| ShowVideo | Integer | r/w | Setting this property to any non-zero value will instruct ActiveX component to display video contents on the form, if form contains visible ActiveX region |
| FrameStartX | Integer | r/w | Frame start position X. This property may be set to any value in range 0..1272 |
| FrameStartY | Integer | r/w | Frame start position Y. This property may be set to any even value in range 0..1016 |
| FrameWidth | Integer | r/w | Frame width – the sum of FrameStartX and FrameWidth must always be less or equal to 1280. Minimal value for FrameWidth is 8 |
| FrameHeight | Integer | r/w | Frame height – the sum of FrameStartY and FrameHeight must always be less or equal to 1024. Minimal value for FrameHeight is 8 |
| FrameDecimation | Integer | r/w | Decimation – the factor by which the image will be decreased in size. Valid values for this property are 1 and 2. Setting decimation to 2 will result in getting two times smaller image from sensor, in other words, every square four pixels from camera will produce only one pixel in resulting image |
| FrameMirrorV | Boolean | r/w | This property should be set to TRUE if vertical flip operation desired |
| FrameMirrorH | Boolean | r/w | This property should be set to TRUE if horizontal mirror operation desired |
| Frequency | Integer | r/w | This property controls working frequency of the camera. Decreasing working frequency results in maximum exposure time growth, in the same time it leads to frame-readout time extension. Care should be taken when choosing working frequency for the camera. The frequency codes and corresponding frequency values can be found in the table below |
| MinExposure | Integer | r | Min exposure in sensor units |
| MaxExposure | Integer | r | Max exposure in sensor units |

| Name | Type | Access | Description |
|---|---|---|---|
| Exposure | Integer | r/w | Working exposure in sensor units |
| MinExposureMs | Float | r | Min exposure in milliseconds |
| MaxExposureMs | Float | r | Max exposure in milliseconds |
| ExposureMs | Float | r/w | Working exposure in milliseconds |
| MinSnapshotExposure | Integer | r | Min snapshot exposure in sensor units |
| MaxSnapshotExposure | Integer | r | Max snapshot exposure in sensor units |
| SnapshotExposure | Integer | r/w | Current snapshot exposure in sensor units |
| MinSnapshotExposureMs | Float | r | Min snapshot exposure in milliseconds |
| MaxSnapshotExposureMs | Float | r | Max snapshot exposure in miliseconds |
| SnapshotExposureMs | Float | r/w | Working snapshot exposure in milliseconds |
| DACrawOffset<br>DACfineOffset | Integer | r/w | These properties determine the black reference level at the output of the output amplifier. Setting value 127 for DACrawOffset property gives the highest offset voltage; value setting 0 gives the lowest offset voltage. Physically, sensor pixel array consists of even and odd columns that have their own output amplifiers. Ideally, if the two output paths have no offset mismatch, the DACfineOffset property must be set to 64. Deviation from this value can be used to compensate the internal sensor mismatch. This value may slightly change from one camera to another and from one working frequency to another |
| LastError | Integer | r | Last Windows 32 error code |

**Table 2. SMX-150 frequency codes and frequency values**

| Code | Value |
|---|---|
| 0 | 40MHz |
| 1 | 24MHz |
| 2 | 20MHz |
| 3 | 16MHz |
| 4 | 13MHz |
| 5 | 12MHz |

| Code | Value |
|---|---|
| 6 | 10MHz |
| 7 | 8MHz |
| 8 | 6.66MHz |

## *SMX-150 Camera ActiveX Functions*

### procedure GetGain

*procedure GetGain(out Main: Integer; out R: Integer; out G: Integer; out B: Integer);*

This function returns current gain settings. The Main gain setting is the value by which all pixels in the pixel array are multiplied, before any digital conversion. R, G and B values are corresponding coefficients for color cameras for each individual channel. It is necessary to recognize two different approaches for amplifiers. The Main amplifier is implemented in hardware and is located in the sensor package. All other amplifiers are emulated in software. Whole range of Main value represents gain settings from 0dB up to 12.4dB, 0dB corresponds to 0 value, 12.4dB – to 16.

**Table 3. SMX-150 frequency codes and frequency values**

| Code | Value |
|---|---|
| 00 | 0dB |
| 01 | 1.37dB |
| 02 | 1.62dB |
| 03 | 1.96dB |
| 04 | 2.33dB |
| 05 | 2.76dB |
| 06 | 3.5dB |
| 07 | 4.25dB |
| 08 | 5.2dB |
| 09 | 6.25dB |
| 10 | 7.89dB |
| 11 | 9.21dB |
| 12 | 11dB |
| 13 | 11.37dB |
| 14 | 11.84dB |
| 15 | 12.32dB |
| 16 | 12.42dB |

Whole range of software emulated amplifiers represents gain settings from -6dB to 6dB, -6dB corresponds to -120, 6dB – to 120. Values are located linearly in the interval.

### procedure SetGain

*procedure SetGain(Main: Integer; R: Integer; G: Integer; B: Integer);*

This function configures hardware and software emulated amplifiers. The Main gain setting is the value by which all pixels in the pixel array are multiplied, before any digital conversion. R, G and B values are corresponding coefficients for color cameras for each individual channel. These values are ignored for monochrome sensor. It is necessary to recognize two different approaches for amplifiers. The Main amplifier is implemented in hardware and is located in the sensor package. All other amplifiers are emulated in software. Whole range of Main value represents gain settings from 0dB up to 12.4dB, 0dB corresponds to 0 value, 12.4 – to 16.

SetGain,The Main Gain value should be in the range of 0..16,The RGB Gain values should be in the range of -120..120.

### function GetLastError

*function GetLastError: Integer;*

This function returns Windows 32 last error if any.

## *Video operations with SMX -150*

During video mode operation two processes exposure and read-out overlaps in time. While part of the frame exposed to the light, other part id read-out into USB. This way fast frame rates can be achieved.

There are two approaches of getting data from the camera in video mode. The first one is to get pointer to the driver supplied storage with new frame, the second one – is to copy driver buffer contents into user supplied buffer. The first approach works much faster, because it excludes relatively expensive copy step from data processing, but it puts time constraints during which frame must be processed. Second one is slower, but you may do with frame data whatever you want, not worrying about driver to override your data with new frame data. Depending on computer speed, necessary data processing, amount of available memory, etc. one of two approaches needs to be chosen. There are two functions, representing each approach:

**function GetFramePtr: Integer** – you are getting pointer to driver memory with new frame data;

**Function GrabFrameToBuffer(Buffer: Integer; BufferSize: Integer): Boolean** – copies the new frame data into user supplied Buffer, with buffer size BufferSize.

## *Snapshot operations with SMX -150*

During snapshot operations all pixels in the pixel array get exposed to light in the same time. This mode is suitable for capturing fast moving objects. In snapshot mode exposure time is added to the read-out time which is effectively halving frame rate.

### function GetSnapshot

*function GetSnapshot(TimeOut: Integer; ExternalTrigger:Boolean; ExternalTriggerPolarity: Boolean; SnapshotMode: Boolean; Buffer: Integer; BufferSize: Integer; o ut RetSize: Integer) : Boolean*

This function allows getting one snapshot out of the camera. TimeOut – this value allows configuring amount of time after which the operation will be aborted. ExternalTrigger – this value if TRUE instructs camera to wait for external trigger signal only. ExternalTriggerPolarity – if this value is set to TRUE, camera will wait for positive polarity trigger signal and for negative polarity otherwise. SnapshotMode – this parameter must be TRUE at all times. The frame data will be stored into Buffer. Size of the Buffer is communicated to driver using BufferSize parameter. Upon success RetSize will contain amount of data actually valid in the buffer. In case of error or timeout RetSize will contain negative value or zero.

## procedure GetMultipleSnapshots

*procedure GetMultipleSnapshots(TimeOut: Integer; ExternalTrigger: Boolean; ExternalTriggerPolarity: Boolean; SnapshotMode: Boolean; FrameSize: Integer; FramesPtr: Integer; FrameNum: Integer; out FramesReturn: Integer);*

This function allows getting multiple snapshot out of the camera. TimeOut – this value allows configuring amount of time after which the operation will be aborted. ExternalTrigger – this value if TRUE instructs camera to wait for external trigger signal only. ExternalTriggerPolarity – if this value is set to TRUE, camera will wait for positive polarity trigger signal and for negative polarity otherwise. SnapshotMode – this parameter must be TRUE at all times. The frame data will be stored into buffer pointed by FramesPtr. Size of each frame, which should be stored into the buffer, is communicated to driver using FrameSize parameter. FrameNum is the number of frames requested. Upon success FramesReturn will contain number of frames actually stored in the buffer. In case of error or timeout, FramesReturn will contain negative value or zero.

There are two helper functions, which allow allocating large frame buffers for multiple snapshot operations. They aid programming languages, like Visual Basic, etc, which fail to do effective memory management themselves. First of these functions will allocate as much memory as possible for current state of operational system. FrameSize gives the function number of bytes required to store one frame. Upon return you get FramePtr containing pointer to allocated buffer and FrameNum – number of frames fit into this buffer.

*function AllocMemory(FrameSize: Integer; out FramesPtr: Integer; out FrameNum: Integer): Boolean;*

Second function allows deallocating previously allocated memory. Failing to call this function after using AllocMemory() will lead to memory leaks.

*Function FreeMemory(FrameSize:Integer; FramesPtr:Integer; FrameNum:Integer): Boolean;*