XLi Camera SDK
Version 17
June 2008

Introduction

The XLi Camera SDK consists of an ActiveX library containing a collection of programming-language independent procedures and functions (methods) that gives the application-writer full control of XLi SDK compatible Cameras.

These functions and procedures are called via an ActiveX/COM interface with the following GUID:

Object ID

IXLISetting = Interface(IUnknown)

['{730EC5BC-FD9A-4BA4-9186-E0D0FF6E8BDC}']

This interface is declared in the SDK type library file: XLISETTING.TLB

Sample applications to demonstrate the SDK, written in Borland Delphi, C++ Builder, Microsoft VB and C#, can be found in the SDK folder of this CD.

Import the type library "xlisetting.tlb" from within your chosen development environment to enable access to all the functions, procedures and types available in the SDK.

Before accessing the type library you need to initialise the COM library and create an instance of the IXLISetting interface. Before calling any other IXLISetting methods, first call one of the following initialisation methods:

function SelectDevice(number, mode:integer; window:HWND) : integer; stdcall;

When the window handle is non-zero this method builds a filter graph with full graphics support and initialises the interface to the chosen camera. number represents a XLi camera, indexed from 1. mode is 0 for high resolution or 1 for low resolution. window is the handle of a window you provide that will be used to display live video.

When the window handle is non-zero, your program will have access to all the methods of the SDK.

If window handle is zero a filter graph won't be built and will have access to the same subset of the full SDK as the Init procedure.

A return value of zero indicates an error. A non-zero value may be cast as an IBaseFilter, allowing us to query for other DirectShow interfaces.

procedure Init(camera:IUnknown); stdcall;

This method provides access to a small subset of the full SDK. It connects to an existing DirectShow filter graph with no support for graphics etc.

Methods available after calling either initialisation method

procedure SetVal(item:Txlireg; value: Integer); stdcall;

Write an integer value for the given camera setting. See below, for the possible values of Txlireg.

function GetVal(item:Txlireg) : Integer; stdcall;

Read the integer value for the given camera setting. See below, for the possible values of Txlireg.

function GetMin(item:Txlireg) : Integer; stdcall;

Return the minimum value allowed for the given camera setting.

function GetMax(item:Txlireg) : Integer; stdcall;

Return the maximum value for a given camera setting. Some values such as XLEXPOSURE change according to various camera modes so should be read regularly.

function SensorToReal(item:Txlireg; value:Integer) : Double; stdcall;

function RealToSensor(item:Txlireg; value:Double) : Integer; stdcall;

These functions convert between internal camera units (Sensor) that you set in a register, and real world units (Real) that you might display to a user, and vice versa. valid choices for item are XLEXPOSURE (milliseconds),XLGAIN (gain factor).

function GetHistogram(data:integer) : HResult; stdcall;

Retrieve a histogram of image luminance values. Pass a pointer to an array of 256 bytes cast as an integer. (see note [#1]), each value represents the amount of pixels of that luminance value, normalised to a range of 0..255.

function SetCurve(table:integer) : HResult; stdcall;

function GetCurve(data:integer) : HResult; stdcall;

These functions Set and Read back the luminance curve correction table. Pass a pointer to an array of 256 bytes cast as an integer. (see note #1) represents the luminance value that should be output in place of the array index value. The normal, default, setting is to have each value match its index, aka linear mode. i.e. data[0]=0 data[1]=1 ... data[255]=255. Use this feature to achieve custom gamma, back light compensation, image negative, and other effects. SetCurve(Nil) resets to the default linear curve.

#1 COM type libraries do not support pointers to a byte so this SDK uses an integer type in place of a pointer. It is the developers responsibility to cast the integer to a compatible pointer supported by the programming environment. We chose to use integer as this makes casting easier for DotNet developers.

Txlireg item list

	Txlireg	RW	Description
0	XLREDGAIN,	RW	Red Channel Gain
1	XLBLUEGAIN	RW	Blue Channel Gain
2	XLGAIN	RW	All Channel Gain
3	XLEXPOSURE	RW	Exposure Setting
4	XLLUMINANCE	RW	Luminance target value
5	XLSATURATION	RW	Saturation Level
6	XLCONTRAST	RW	Contrast Level
7	XLBLACKLEVEL	RW	Brightness/Black level
8	XLAWB	RW	Auto White Balance (0=off 1=on)
9	XLAGC	RW	Auto Gain Control (0=off 1=on)
10	XLAEC	RW	Auto Exposure Control (0=off 1=on)
11	XLGAMMA	RW	Gamma * 100, range 0 to 200, default=100
12	XLROTATE	RW	Rotate flags. bit 0 = horizontal flip, bit 1 = vertical flip
13	XLRESOLUTION	R	Read resolution (0=high 1=low)
14	XLSENSOR	R	Read image sensor type
15	XLYAVG	R	Average observed luminance
16	XLBAVG	R	Average Blue channel luminance
17	XLGrAVG	R	Average Green (Red line) luminance
18	XLGbAVG	R	Average Green (Blue line) luminance
19	XLRAVG	R	Average Red Channel luminance
20	XLAWBSPEED	RW	Auto White balance speed (0=slow, 1=slowest, 2,3=fast)
21	XLAWBSTEP	RW	AWB step size (0=fine, 1,3=coarse, 2=medium)
22	XLBAND	RW	Banding control (0=off, 1=50Hz, 2=60Hz)
23	XLTFRAME	RW	Frame Rate Adjustment (frame PERIOD in mS * 256)
24	XLLUMRANGE	RW	Luminance Target size (+- this value)
29	XLFLATFIELD	RW	flat field Correction (0=off 1=on)
30	XLAECTYPE	RW	auto exposure Style (0=LumTarget, 1=PeakWhite)
31	XLTFRAMEAVG	R	Frame rate being achieved over USB, (in mS * 256)
32	XLPROCESSINGMODE	RW	Mono Picture (0) or Colour (1) or Raw Bayer (2)
33	XLISCOLOURCAMERA	R	Is this a colour camera (1) or not (0)
34	XLHASBUTTON	R	Has button(1) or not (0)
35	XLVERSION	R	Returns the SDK version

	Txlireg	RW	Description
36	XLTHRESHOLD	RW	AWB threshold level
37	XLFACTORYRESET	R	Use to get the id of our factory reset windows message
38	XLGREENGAIN	RW	Green Channel Gain
39	XLBLACKBIAS	RW	Black level correction before gamma is applied
40	XLREDOUTPUT	RW	Final output for Red
41	XLGREENOUTPUT	RW	Final output for Green
42	XLBLUEOUTPUT	RW	Final output for Blue

Additional methods available after calling SelectDevice:

procedure Freeze(lock:WordBool); stdcall;

Pause the video display image, this does not stop the image overlay drawing commands from working, nor does it physically stop the camera. set lock to true to freeze, and false to unfreeze.

function Grab : Integer; stdcall;

Grab the image data for the currently displayed/paused image. Data is a byte array of pixels stored in RGB24 format (see note #1 on previous page).

Procedure GetResolution(out width, height:Integer); stdcall;

Obtains the width and height of the current video mode.

function ShowPropertyPage(left,top:Integer) : HWND; stdcall;

Display the camera controls dialog window and return a handle to the window.

function CameraControls(window:HWND; left,top:Integer) : HWND; stdcall;

As ShowPropertyPage function but with additional “window” parameter. This is the handle of a window to where the camera controls page will send its notification messages. Set “window” to 0 for no notification messages, in which case, it is the same the ShowPropertyPage fuction.

procedure GetVideoSource(out left, top, width, height: Integer); stdcall;

Returns the source video rectangle.

procedure SetVideoSource(left,top,width,height:Integer); stdcall;

Select the portion of the camera image to be displayed, the selected region will be scaled to fit the selected video window size.

procedure SetVideoWindow(left,top,width,height:Integer); stdcall;
procedure SetVideoSourceZoom(X,Y:Integer; scale:Double); stdcall;

These two methods are no longer supported and simply stubs.

procedure MouseToCameraCoords(MouseX, MouseY : integer; out CameraX, CameraY : integer); stdcall;

Use to convert mouse coordinates to camera coordinates.

procedure GetFormatList(out formats : OleVariant); stdcall;

Formats is an OleVariant, that on return contains an array of widestrings. Each element in this array is a format supported by the camera sensor. Simple query the variant for its dimensions.

procedure GetDeviceName(out name : widestring); stdcall;

On return, name will hold the current friendly name of the camera.

procedure GetDeviceList(out devices : OleVariant); stdcall;

Devices is an OleVariant, that on return contains an array of widestrings. Each element in this array is a XLi camera device. Simple query the variant for its dimensions to find the number and types of cameras available.

function SetLed(value: WordBool): WordBool; stdcall;

This is a stub intended for possible future developments.

procedure ShutDown; stdcall;

Stops the current camera and destroys its filter graph.

procedure FactoryReset; stdcall;

Resets the camera to its factory settings then broadcasts a factory reset message so that interested windows have the opportunity to read any camera values that may have changed. Use GetVal(XLFACTORYRESET) to return the id of the windows message that will be sent. (Note: The id is not a constant). This allows you to check for the message with in the message handler for your window.

Graphics Overlay Types

GFXHandle = Integer;

TGPColor = Ulong;

GFXType = enumerated list of object types

0. GFXTNone No object
1. GFXTText TTF Text
2. GFXTRect Solid Rectangle
3. GFXTCircle Hollow Circle/Oval
4. GFXTLine Straight Line
5. GFXTPath Complex Line
6. GFXTMark Predefined Shape
7. GFXTPxDist Pixel distance measurement
8. GFXTPxArea Pixel area measurement

GFXMarkType = enumerated list of Shapes

0. GFXMCrossHair Cross Hair with thin lines
1. GFXMFatHair Cross Hair with wide lines
2. GFXMXY Cross Hair that extends to edges of picture

Graphics Create Functions

(Note: Functions that return a wordbool indicate success with true and failure with false. Functions that return a GFXHandle, return the handle of the created object if successful or -1 to indicate a failure.)

function GFX_AddLabel(X,Y:Integer; Angle:Integer; colour:TGPColor; FontSize:Integer; const LineText:WideString; PixelMeasurement:Double; Style GFXType) : GFXHandle; stdcall;

Add a text label that depending on the Style parameter, represents an ordinary label or pixel measurement. The angle to display the text in degrees, may range from -90 to +90, with 0 degrees representing horizontal.

function GFX_AddText(X,Y:Integer; colour:TGPColor; const LineText:WideString; FontSize:Integer) : GFXHandle; stdcall;

A subset of GFX_AddLabel. Use it for adding a horizontal text label.

function GFX_AddPixelMeasurement(X,Y:Integer; colour:TGPColor; FontSize:Integer; PixelMeasurement:Double; MeasureStyle:GFXType) : GFXHandle; stdcall;

Add a text label to the image that represents a pixel measurement.

function GFX_AddRect(X,Y:Integer; colour:TGPColor;Width, Height:Integer) : GFXHandle; stdcall;

function GFX_AddCircle(X,Y:Integer; colour:TGPColor; Width, Height:Integer) : GFXHandle; stdcall;

function GFX_AddLine(X,Y:Integer; colour:TGPColor; X2,Y2:Integer) : GFXHandle; stdcall;

function GFX_AddMark(X,Y:Integer; colour:TGPColor; MarkStyle:GFXMarkType; Size:Integer) : GFXHandle; stdcall;

function GFX_AddPath(X,Y:Integer; colour:TGPColor; X2,Y2:Integer) : GFXHandle; stdcall;

function GFX_AddPoint(id:GFXHandle; X,Y:Integer) : WordBool; stdcall;

Create a multi-part line object, and add additional points to it. Good for free hand drawing modes or other complex shapes.

Graphics Manipulation Functions

function GFX_GetOrigin(id:GFXHandle; out X,Y:Integer) : WordBool; stdcall;

Obtains the X,Y origin of the object. Result is true if successful.

function GFX_MoveXY(id:GFXHandle; X,Y:Integer) : Wordbool; stdcall;

Move an objects origin point to a new position.

function GFX_DeleteID(id:GFXHandle) : WordBool; stdcall;

Destroy an object.

function GFX_SetColour(id:GFXHandle; colour:TGPColor) : WordBool; stdcall;

Set the colour of an object. Colours are in hex 0xAARRGGBB, Alpha, Red, Green, Blue. eg. 0xFF00FF00 is opaque green, 0x80FF0000 is half transparent red.

function GFX_GetColour(id: GFXHandle) : TGPColor; stdcall;

Get the colour of an object. See GFX_SetColour above.

function GFX_SetText(id:GFXHandle; const LineText:WideString) : WordBool; stdcall;

Change the text string of a text object.

function GFX_GetTextAngle(id: GFXHandle): Integer; stdcall;

Returns the angle in degrees of a text object.

function GFX_SetTextAngle(id: GFXHandle; Angle: Integer): WordBool; stdcall;

Change the angle in degrees of a text object. Accepts values in the range -90 to +90, with 0 degrees representing horizontal.

function GFX_SetSize(id:GFXHandle; Size:Integer) : WordBool; stdcall;

Change the size of objects that use a size parameter, eg line thickness.

function GFX_Highlight(id:GFXHandle; Show:WordBool) : WordBool; stdcall;

Highlight this object by drawing a dashed bounding rectangle.

function GFX_Clickable(id:GFXHandle; canclick:WordBool) : WordBool; stdcall;

Set if this object can be found by the Find call.

function GFX_Filled(id:GFXHandle; dofill:WordBool) : WordBool; stdcall;

Set if this object (Path, Circle, or Rect) should be filled (true) or outline (false,default).

function GFX_Area(id:GFXHandle) : Double; stdcall;

Calculate the Area enclosed by a Path, in pixels squared, only works on paths that do not cross themselves, and treats the path as if it were closed by a line connecting the first and last points.

function GFX_Find(X,Y:Integer) : GFXHandle; stdcall;

Return the handle for the first object which is clickable and who's bounding box intersects this coordinate.

function GFX_GetType(id:GFXHandle) : GFXType; stdcall;

Return the object type of an object.

function GFX_GetBound(id:GFXHandle; out X,Y,Width,Height:Integer) : WordBool; stdcall;

Returns the bounding rectangle for an object.

NB: the object must have been drawn on the video stream at least once before the bounds are known. So you should wait for at least one frame interval after creating the object before asking its bounds.

function GFX_SetMeasurement(id:GFXHandle, value:double) : WordBool; stdcall;

Change a measurement type value.