Starlight Xpress DotNet SDK Draft Documentation

Issue D (19/05/2015)

Introduction:

The Starlight Xpress SDK for Visual Studio consists of 2 DLLs, SXUSB.DLL & SXDOTNET.DLL. The first file provides the low level functions for the range of Starlight Xpress cameras. The second file "SxDotNet.dll" provides higher level operations & provides the interface to the managed memory environment of Visual Studio C++/CLI, C# & Visual Basic.Net.

This SDK does not support the interlaced "MX" range of cameras yet.

Data Structures:

sxInfoPack - holds all the essential data about the camera. The members are:

- int imageWidth // The size in pixels of one line (or row) of the CCD image
- int imageHeight // The number of active lines of the CCD image
- float pixelWidth // width of pixel in microns
- float pixelHeight // height of pixel in microns
- int colorMatrix // colour reconstruction matrix
- unsigned char bitsPerPixel // The number of bits per pixel, normally 16
- bool hasStar2K // Camera is fitted with a Star 2000 guider port
- bool hasEEProm // Camera is fitted with an EEProm. Always true.
- bool hasCooler // Camera is equipped with a temperature regulated cooler
- bool hasShutter // Camera is fitted with shutter, currently only the H18.
- Int cameraModel // the camera model number
- String cameraName // the camera name currently no more than 8 characters
- int blackPixelsAtStart; // number of black pixels at start of a line
- int blackPixelsAtEnd; // number of black pixels at the end of a line
- int blackLinesAtStart; // number of black lines at the start of an image
- int blackLinesAtEnd; // number of black lines at the end of an image
- int upperFirmwareVersion; // major firmware version
- int lowerFirmwareVersion; // minor firmware version
- int buildNumber; // firmware build number
- int result; // result of readInfoPack operation.

sxExposurePack – holds all the data needed for image capture

- IntPtr camHandle; // handle to selected camera
- unsigned short flags; // used to control which fields are read on interlaced cameras
- bool useGuider; // get image from guider instead of main camera
- bool useGated; // exposure is started by external control line
- unsigned short xOffset; // image offset in horizontal direction
- unsigned short yOffset; // image offset in vertical direction
- unsigned short imageWidth; // size of image in pixels in horizontal direction
- unsigned short imageHeight; // size of image in lines in vertical direction
- unsigned short xBin; // pixel binning factor in horizontal direction
- unsigned short yBin; // pixel binning factor in vertical direction
- unsigned int exposureTime; // exposure time in milliseconds

sxCoolerPack – holds all the data needed for cooler operation

- IntPtr camHandle; // handle to selected camera
- bool coolerON; // true will turn ON the cooler
- float setPoint // the desired temperature in degrees C, -40.0 to +35.0
- bool actualState; // true if the cooler is currently ON
- float currentTemperature // the CCD temperature in degrees C, -40.0 to +35.0
- int result // > 0 if operation successful

Classes:

There are 5 classes –

sxSDKAccess, sxInfo, sxImage, sxUtilities, sxSerialPort

sxSDKAccess contains the camera opening and closing methods:

- Int32 sxOpen(array<IntPtr> sxHandle)
  - The camera opening method will return a handle for each SX camera it can find up to a maximum of 20 cameras. The parameter passed should be an array of 20 IntPtr. The array will be filled with a handle to each active camera found. The int return value is the total number of cameras found, or zero for none or on error.
- int sxClose(IntPtr sxHandle)
  - The camera closing method will close a single camera from the IntPtr handle parameter. The int return value is > 0 if successful.

sxInfo contains a single method for retrieving the camera information

- sxInfoPack readInfoPack(IntPtr camHandle)
    - This method takes a single parameter, the camera handle as an Intptr. It returns a filled sxInfoPack data structure.

sxImage contains the image capture methods

- Int32 makeImage(sxExposurePack exposureInfo, ref Int16[] pixelBuffer)
    - This method returns an int > 0 on success. It takes a filled sxExposurePack data structure and a reference to an array of Int16[] which is used to store the returned image pixel data.
- Int32 makeExposure(sxExposurePack exposureInfo, Boolean waitForExposureCompleted)
    - This method will start an exposure
    - If "waitForExposureComplete" is TRUE then wait for the function to return. Requests for cooler temperature can be sent in another thread.
    - If "waitForExposureComplete" is FALSE then time the exposure in the host program or by getting the camera timer data using "sxUtilities.getTtimer". Any command can be sent to the camera during this time
- Int32 readImage(sxExposurePack exposureInfo, ref Int16[] pixelBuffer)
    - This method is used to download the image data collected after “makeExposure” has been called. The exposure timer data is ignored. No commands should be sent to the camera that require returned data until the image download is complete.

sxUtilities contains several camera control methods

- void resetCamera(IntPtr camHandle)
    - Resets the USB bus & internal camera settings.
- int setGuiderPort(IntPtr camHandle, unsigned char portValue)
    - Sets the guider port to the char parameter port value. The camera is selected by the IntPtr camHandle parameter. The int return value is > 0 on success.
- int openShutter(IntPtr camHandle)
    - Opens the shutter on H18 camera only. The IntPtr parameter selects the camera.
- int closeShutter(IntPtr camHandle)
    - Closes the shutter on H18 camera only. The IntPtr parameter selects the camera.
- sxCoolerPack cooler(SxCoolerPack coolerInfo)
    - Controls the camera regulated cooler. The returned sxCoolerPack contains the information sent plus; result > 0 on success. The bool coolerState returns the actual on/off state of the cooler, on being

true. The Single currentTemperature is the temperature of the CCD in degrees C.

- Single coolerTemp(IntPtr camHandle)
  - Returns the CCD temperature in degrees C.
- Int32 setTimer(IntPtr camHandle, unsigned int time)
  - Sets the camera internal millisecond timer. The int return value > 0 on success. The IntPtr parameter selects the camera. The unsigned int time parameter is the value in ms for the timer.
- UInt32 getTimer(IntPtr camHandle, unsigned int time)
  - Reads the camera internal millisecond timer. The unsigned int return value is the current timer value in ms. The IntPtr parameter selects the camera.