Starlight Xpress SDK Documentation

Issue F (19/05/2015) Applies to version 1.04

Introduction:

The Starlight Xpress original SDK consists of a single file - SxUSB.dll. This file provides all the functions for communicating with the range of Starlight Xpress cameras. It has been written in C & can be compiled with VS C++ 2010.

Using the SDK:

This section describes the general sequence of events to use the SDK to access the main camera functions.

Getting Connected:

The SxUSB can connect to twenty connected SX imagers or guiders, plenty for most practical purposes however the host program must manage the cameras.

1. Start by allocating an array of up to twenty object handles.
2. Call the SxOpen() function passing a pointer to the array of twenty handles
3. Check the return value which should equal the number of SX cameras attached.

Gathering Data:

1. Create an array of "t_sxccd_params" structures for each connected camera.
2. Call the sxGetCameraParams() for each connected camera in turn, placing the data in the array created in step 1
3. Create an array of "long" for the firmware version of each connected camera.
4. Call the sxGetFirmwareVersion() function and store the returned data in the array created in step 3 above.
5. Create arrays of "int" for the camera model number of each connected camera
6. Call the sxGetCameraModel () function and store the returned data in the array created in step 5 above.
7. All the necessary data is now available for each of the connected cameras. The "extra_caps" byte in the "t_sxccd_params" contains bit flags to indicate some of the extra capabilities of the camera as described in the table below.

0x01	0x02	0x04	0x08	0x10	0x20	0x40	0x80
Star2K	X	Set	Guider attached	Cooler fitted	Shutter Fitted	X	X

Table 1

Notes: "Star2K" means that a four output port is available for controlling a telescope mount. "Guider attached" means that this camera has a guide camera connected to a special port on the main camera, (SXVR & SXVF, not available on Trius models). Both the camera & guider are controlled through the one camera USB port.

Taking an Image:

This section applies to taking an image using the progressive scan (H series) cameras. The exposure timing is normally done using the camera's internal timer for exposures of less than one second and using the host computers timing for exposures over one second.

1. Call the "sxClearPixels()" function with the "flags" parameter set to 0x00. This will wipe the CCD and start an exposure.
2. If the exposure is greater than one second then skip to step 3 otherwise call the sxExposePixels() function filling in all the exposure parameters including the exposure time. Now call the sxReadPixels() function to download the image into the host supplied pixel array parameter. Note no other functions should be called between sxExposePixels() & completion of the image download that would require data to be sent back from the camera. Failure to observe this rule will cause the USB system to become frozen. Continue at step 4.
3. If the exposure is more than one second then use the host PC to time the exposure (following on from step 1), when the exposure time has elapsed call the sxLatchPixels() function filling in all the exposure parameters. Now call the sxReadPixels() function to download the image into the host supplied pixel array parameter. Note no other functions should be called between sxLatchPixels() & completion of the image download that would require data to be sent back from the camera. Failure to observe this rule will cause the USB system to become frozen.
4. For long exposures it is recommended to call the "sxClearPixels()" function with the "flags" parameter set to 0x08 (CCD_FLAGS_NO_WIPE_FRAME). This will clear the CCD vertical registers. It should be called about every 30 seconds and at least 4 seconds before the end of the exposure.
5. The host PC should process the returned pixel array data for subsequent display.

Data Structures:

The "t_sx_ccd_params" structure contains all the essential data about the camera. An instance of this structure is filled using the "`sxGetCameraParams`" function. The entries are described in the comment lines.

```
struct t_sxccd_params
{
    USHORT hfront_porch;      // black pixels at start of a line
    USHORT hback_porch;       // black pixels at end of a line
    USHORT width;             // total number of active pixels in a line
    USHORT vfront_porch;      // black lines at start of a frame
    USHORT vback_porch;       // black lines at end of a frame
    USHORT height;            // total number of active lines
    float  pix_width;         // pixel width dimension in microns
    float  pix_height;        // pixel height dimension in microns
    USHORT color_matrix;      // reserved do not use
    BYTE   bits_per_pixel;    // usually 16 or 8 for a guider
    BYTE   num_serial_ports;  // usually 2
    BYTE   extra_caps;        // bit flags for camera features
```

```
    BYTE    vclk_delay;         // reserved, do not use
};
```

Functions:

Default Return Value: If the return value is not explicitly described then if the function succeeds, the return value is a nonzero value returned in a LONG or ULONG.

- `int sxOpen(HANDLE *sxHandles)`
  Description: is used to obtain a handle for each connected camera. The array of handles is filled with a handle for each SX camera it can find up to a maximum of 20 cameras.
  Parameters: A pointer to an array of 20 HANDLE. The array will be filled with a handle for each active camera found.
  Return Value: The **int** return value is the total number of cameras found, or zero for none or on error.

- `void sxClose(HANDLE sxHandle)`
  Description: Will close a single camera from the HANDLE parameter supplied
  Parameter: A HANDLE for the camera to be disconnected.

- `LONG sxReset(HANDLE sxHandle)`
  Description: Will reset the camera & its USB port from the HANDLE parameter supplied. It cancels all current actions (except image download) & places the camera in its idle state.
  Parameter: A HANDLE for the camera to be reset.

- `ULONG sxSetTimer(HANDLE sxHandle, ULONG msec)`
  Description: Will set the camera's internal timer to the value supplied & start the timer counting down.
  Parameters: A HANDLE for the camera to be reset. A ULONG (4 byte) timer value in milliseconds.

- `ULONG sxGetTimer(HANDLE sxHandle)`
  Description: Will return the camera's internal timer value in milliseconds
  Parameters: A HANDLE for the camera.
  Return Value: The **ULONG** return value is the number of milliseconds remaining to be counted down.

- `ULONG sxClearPixels(HANDLE sxHandle, USHORT flags, USHORT camIndex)`
  Description: Will either latch the CCD image or clear the CCD vertical registers or wipe the entire CCD depending on the state of the bits in the "flags" parameter.
  Parameters: sxHandle – a HANDLE for the camera, flags – a USHORT containing bit flags to determine the function action see the table below, camIndex a USHORT when zero selects the main camera, when non-zero selects the attached guider.

"flags" bit values. If CCD_EXP_FLAGS_NOCLEAR_FRAME (0x40) is set this function performs no action. See SxSDK.h for CCD_EXP_FLAGS_xxx definitions. The field bits are used individually for interlaced camera usually the MX series. When using an H series progressive readout camera then both field bits are set together.

Flags = CCD_EXP_FLAGS +	_FIELD_ODD 0x01	_FIELD_EVEN 0x02	_NOWIPE_FRAME 0x08	_NOCLEAR_REGISTER 0x80
Latch Odd Frame (0x89)	1	0	1	1
Latch Even Frame (0x8A)	0	1	1	1
Latch Both Frames (0x8B)	1	1	1	1
Clear vertical registers (0x08)	0	0	1	0
Wipe entire CCD (0x80)	0	0	0	1

Table 2

- `LONG sxLatchPixels(HANDLE sxHandle, USHORT flags, USHORT camIndex, USHORT xoffset, USHORT yoffset, USHORT width, USHORT height, USHORT xbin, USHORT ybin)`

  Description: Will latch the CCD frame (odd, even or both) finishing the exposure & then download the frame data to the host PC. Once the download starts it must be completed. No camera commands should be sent until the download has completed.

  Parameters:

  sxHandle - HANDLE for the camera
  flags - USHORT containing bit flags to select the frame to be downloaded, see Table 1.
  camIndex - USHORT if zero selects main camera, when non-zero selects attached guider.
  xoffset - USHORT sets the number pixels in from the start of the line.
  yoffset - USHORT sets the number lines down from the start of the frame.
  width - USHORT sets the number pixels in a line.
  height - USHORT sets the number lines in the frame.
  xbin - USHORT sets the number pixels to accumulate for a line
  ybin - USHORT sets the number lines to accumulate for a frame

- `LONG sxExposePixels(HANDLE sxHandle, USHORT flags, USHORT camIndex, USHORT xoffset, USHORT yoffset, USHORT width, USHORT height, USHORT xbin, USHORT ybin, ULONG msec)`

  Description: Will start a timed exposure which when completed will latch the CCD frame (odd, even or both) finishing the exposure & then download the frame data to the host PC. Once the download starts it must be completed. No camera commands should be sent until the download has completed.

  Parameters:
  sxHandle - HANDLE for the camera
  flags - USHORT containing bit flags to select the frame to be downloaded, see Table 1.
  camIndex - USHORT if zero selects main camera, when non-zero selects attached guider.
  xoffset - USHORT sets the number pixels in from the start of the line.
  yoffset - USHORT sets the number lines down from the start of the frame.
  width - USHORT sets the number pixels in a line.
  height - USHORT sets the number lines in the frame.
  xbin - USHORT sets the number pixels to accumulate for a line
  ybin - USHORT sets the number lines to accumulate for a frame
  msec - ULONG (4 byte) timer value in milliseconds.

- `LONG sxExposePixelsGated(HANDLE sxHandle, USHORT flags, USHORT camIndex, USHORT xoffset, USHORT yoffset, USHORT width, USHORT height, USHORT xbin, USHORT ybin, ULONG msec)`

  Description: Identical to "sxExposePixels" above but needs an external hardware trigger to start the exposure.

- `LONG sxReadPixels(HANDLE sxHandle, USHORT *pixels, ULONG count)`

  Description: Reads the image data into an array of USHORT. This function is used after the exposure is started with "sxLatchPixels" or "sxExposePixels" to download the image when the exposure is complete.

  Parameters:
  sxHandle - HANDLE for the camera
  *pixels – pointer to an array of USHORT, this array must be large enough to hold the image size.
  count – ULONG The number of pixels to read.
  Return value: A successful read will return the number of bytes read. A zero value indicates a buffer allocation error. A negative result is the GetLastError() turned into a negative value.

- `ULONG sxGetCameraParams(HANDLE sxHandle, USHORT camIndex, struct t_sxccd_params *params)`

  Description: Reads the camera size & capability parameters

  Parameters:
  sxHandle - HANDLE for the camera
  camIndex - USHORT if zero selects main camera, when non-zero selects attached guider.
  *params – a pointer to a structure of t_sxccd_params. This structure is loaded with the camera or guider size & capability, see the Data Structures section for more details.

- `ULONG sxSetSTAR2000(HANDLE sxHandle, BYTE star2k)`

  Description: Reads the camera size & capability parameters

  Parameters:
  sxHandle - HANDLE for the camera
  star2k – USHORT contains the guider direction in the lowest 4 bits.
  0x04 = North
  0x08 = East
  0x02 = South
  0x01 = West

- `ULONG sxWriteSerialPort(HANDLE sxHandle, USHORT portIndex, USHORT flush, USHORT count, BYTE *data)`

  Description: Writes an array of byte data to the specified port. Serial port fixed at 9600 baud, no parity, 8 data bits & one stop bit.

  Parameters:
  sxHandle - HANDLE for the camera
  portIndex – USHORT 0 = Port 1, 1=Port2.
  flush – USHORT not used.
  count – USHORT the number of byte to write
  *data – a pointer to an array of BYTE, the data to be sent.

- `ULONG sxReadSerialPort(HANDLE sxHandle, USHORT portIndex, USHORT count, BYTE *data)`

  Description: Writes an array of byte data to the specified port. Serial port fixed at 9600 baud, no parity, 8 data bits & one stop bit.

  Parameters:
  sxHandle - HANDLE for the camera
  portIndex – USHORT 0 = Port 1, 1=Port2.
  count – USHORT the number of byte to read.
  *data – a pointer to an array of BYTE, the data to be read.

- `USHORT sxGetCameraModel(HANDLE sxHandle)`

  Description: is used to obtain the model number for the connected camera. The model number can be used to determine more operational parameters of the camera
  Parameters:
  sxHandle - HANDLE for the camera
  Return Value: The USHORT return value is the model number for the camera.

- `ULONG sxGetFirmwareVersion(HANDLE sxHandle)`

  Description: is used to obtain the firmware for the camera. The version number can be used to determine the major features of the camera.
  Parameters:
  sxHandle - HANDLE for the camera
  Return Value: The ULONG return value is the firmware version number for the camera. Each byte of the ULONG contains a nibble of the version number e.g. 0x00010108 is version 1.18

- `USHORT sxGetBuildNumber(HANDLE sxHandle)`

  Description: is used to obtain the firmware build number for the camera. The build number can be used to determine the minor features of the camera.
  Parameters:
  sxHandle - HANDLE for the camera
  Return Value: The USHORT return value is the firmware build number for the camera. e.g. the USHORT 0x0454 = build number 1108.

- `ULONG sxSetCooler(HANDLE sxHandle, UCHAR SetStatus, USHORT SetTemp, UCHAR *RetStatus, USHORT *RetTemp )`

  Description: is used to set the cooler operating temperature & turn it on or off.
  Parameters:
  sxHandle - HANDLE for the camera
  SetStatus - UCHAR turns the cooler on or off
  SetTemp - USHORT sets the cooler operating point
  *RetStatus – a pointer to a UCHAR which is the current cooler on/off status
  *RetTemp – a pointer to a USHORT which is the current cooler temperature
  Cooler Temperatures are sent & received as 2 byte unsigned integers.
  Temperatures are represented in degrees Kelvin times 10
  Resolution is 0.1 degree so a Temperature of -22.3 degC = 250.7 degK is represented by the integer 2507
  Cooler Temperature = (Temperature in degC + 273) * 10.
  Actual Temperature in degC = (Cooler temperature - 2730)/10
  Note there is a latency of one reading when changing the cooler status.
  The new status is returned on the next reading.
  Maximum Temperature is +35.0degC and minimum temperature is -40degC

- `ULONG sxGetCooler(HANDLE sxHandle, UCHAR *RetStatus, USHORT *RetTemp )`

  Description: is used to get the cooler operating temperature on or off status.
  Parameters:
  sxHandle - HANDLE for the camera
  *RetStatus – a pointer to a UCHAR which is the current cooler on/off status
  *RetTemp – a pointer to a USHORT which is the current cooler temperature

- `ULONG sxGetDLLVersion(VOID)`
- **Description**: is used to obtain the version & build numbers for this DLL.
- **Parameters**: None
- **Return Value**: The ULONG return value is the DLL version MSD in bits24:31, LSD in bits 16:23 & the build number in bits 0:16.

  .

- `ULONG sxSetSerialNumber(HANDLE sxHandle, BYTE* serialNumber)`
- **Description**: is used to write the 32 byte serial number data for the connected camera to a eeprom address at 0x3FC0;
- **Parameters**:
  sxHandle - HANDLE for the camera
  serialNumber – a pointer to a BYTE array that hold the serial number data to be written.

- `USHORT sxGetCoolerVer(HANDLE sxHandle)`
  Description: is used to obtain the cooler microcontroller firmware version number.
  Parameters:
  sxHandle - HANDLE for the camera
  Return Value: The USHORT return value is the cooler firmware version number for the camera. e.g. the returned USHORT 0x07F2 = version number 2034.