SPOTCam Driver DLL

Table of Contents

Overview

The SpotCam driver dll provides an API through which a applications can control the Diagnostic Instruments SPOT family of digital cameras. This document describes the API functionality. The SpotCam.h header file should be referenced for further information about values and structures used by the SpotCam driver.

Driver Installation

The SpotCam driver requires various device driver and DLL files for hardware access. The SpotCam stand-alone installer or SpotDrvr DLL should be used to install and update the SpotCam driver and related DLLs and device drivers. The installer and SpotDrvr DLL will store the path of the SpotCam.dll file in the Registry at: HKLM\SOFTWARE\Diagnostic Instruments, Inc.\SPOT Camera for Windows\Driver Path. An application which dynamically loads SpotCam.dll driver can look for this Registry entry to determine the location of the file.

Concepts and Usage

Device Selection and Initialization

The SpotCam driver can be loaded like any other DLL. Once the SpotCam driver has been loaded, if there is a chance that more than one SPOT camera is installed on the machine, the application should obtain a list of available camera devices by calling SpotFindDevices and specify which device to use by setting either the SPOT_DRIVERDEVICENUMBER or SPOT_DEVICEUID parameters with SpotSetValue.

Applications should use the szDescription member of the SPOT_DEVICE_STRUCT structs when displaying choices to the user. After a device has been specified, SpotInit should be called to initialize the driver. If neither SPOT_DRIVERDEVICENUMBER nor SPOT_DEVICEUID are set before SpotInit is called, the driver will use the first device it finds. If the UID of the desired device is already know, SPOT_DEVICEUID can be set without calling SpotFindDevices. No camera operations can be done until SpotInit has successfully returned.

After SpotInit has returned successfully, the application should query the driver to obtain information about the camera by calling SpotGetCameraAttributes, SpotGetValue, and SpotGetVersionInfo2.

Parameter Setting

The SpotSetValue function is used to set parameters for camera operation. Before acquiring an image, various parameters must be set. SpotGetValue can be called to retrieve parameter values.

Image Acquisition

The SpotCam driver provides two basic modes of image acquisition: still capture and live mode. Still capture mode is used to acquire high-quality images, while live mode is used to quickly acquire multiple images for purposes of positioning and focusing. Live mode images may be of a lower quality than still captures.

Auto-Exposure Computation

The SpotCam driver currently provides auto-exposure functionality for all SPOT family camera. When auto-exposure is enabled, the SpotCam driver will automatically compute an exposure before acquiring an image. For camera and modes which allow multiple gain levels, the driver will choose the highest allowable gain value necessary to keep the exposure time short. An application can limit the gain value which will be chosen by setting the SPOT_AUTOGAINLIMIT parameter for still capture mode or SPOT_LIVEAUTOGAINLIMIT for live mode. The exposure time values to be computed can be limited on both low and high ends by setting certain parameters. The SPOT_MINEXPOSUREMSEC parameter can used to set the minimum allowable exposure for both still capture and live mode. SPOT_MAXEXPOSUREMSEC is used to set the maximum allowable exposure for still capture, and SPOT_LIVEMAXEXPOSUREMSEC is used to limit the maximum allowable exposure for live mode. When computing exposure, the driver will adjust the gain value, if possible, to satisfy the exposure limit settings.

An application can specify whether the brightfield or darkfield method should be used when computing exposure by setting the SPOT_IMAGETYPE parameter to SPOT_IMAGEBRIGHTFLD or SPOT_IMAGEDARKFLD, respectively. In the brightfield method, the exposure is computed so that at least 1% of the pixels in acquired images will be at full-scale brightness. In the darkfield method, the exposure is computed so that only the brightest image pixels will be at full-scale. After the exposure and gain values are computed, they are adjusted by the

brightness adjustment factor which can be set by the SPOT_BRIGHTNESSADJ or SPOT_BRIGHTNESSADJX1000 parameters. The default adjustment factor is 1.

The exposure values will be computed for the pixels included in the image area set by the SPOT_EXPOSURECOMPRECT parameter. By default, the full chip is used. The exposure computation may be affected by the current setting of the SPOT_BITDEPTH parameter. For some cameras, more gain levels are available at 8 and 24 bpp than at higher bit depths. For cameras with color filters, if a color bit depth has been specified, separate exposure time values will be computed for each of the colors enabled by the SPOT_COLORENABLE or SPOT_COLORENABLE2 parameters. For monochrome bit depths with color filter cameras, the SPOT_COLORENABLE or SPOT_COLORENABLE2 parameters specify which filter color should be used.

An application can set the SPOT_AUTOEXPOSE parameter to TRUE to enable auto-exposure computation when acquiring images with SpotGetImage or SpotGetSequentialImages. For live mode, the bComputeExposure argument to SpotGetLiveImages is used to specify auto-exposure. An application can also explicitly compute exposure by calling SpotComputeExposure or SpotComputeExposure2. The exposures computed by these functions can be used for future still image captures by setting the SPOT_EXPOSURE or SPOT_EXPOSURE2 parameters with the computed exposures.

Future models of cameras may not support auto-exposure, so an application should check the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes to determine if auto-exposure is supported for the current camera.

Color Enable

For cameras with color filters, an application can specify the filter color or colors to be used for exposure computation and/or image acquisition by setting the SPOT_COLORENABLE or SPOT_COLOR_ENABLE2 parameters. For slider cameras and monochrome exposures, if the color filter is not being used, all of the members of the SPOT_COLOR_ENABLE_STRUCT or SPOT_COLOR_ENABLE_STRUCT2 struct should be set to FALSE.

White Balance

For color cameras, white balance assures that the pixel intensities in red, green, and blue are in the proper ratios for accurate color reproduction. For cameras with color filters, the white balance values determine the ratios of computed exposure times. For color mosaic cameras, the white balance values are used to scale the pixel values in the three colors. An application can set the white balance values to be used by the SpotCam driver for exposure computation and/or image acquisition via the SPOT_WHITEBALANCE or SPOT_WHITEBALANCEX1000 parameters. For cameras which are capable of auto-exposure computation, the SpotComputeWhiteBalance or SpotComputeWhiteBalanceX1000 functions can be called to compute the white balance values for the image currently in the camera's view. The application can specify the area of the image sensor to be used for white balance computation by setting the SPOT_WHITEBALCOMPRECT parameter. White balance values are always normalized so that the smallest value is 1.0.

Bit Depth

For most SPOT family cameras, the SpotCam driver can return images at multiple bit depths. An application can determine which bit depths are supported by the current camera by checking the value of SPOT_BITDEPTHS. Bit depths below 24 represent monochrome images. Before computing exposure or acquiring any images, an application should set the SPOT_BITDEPTH parameter to the desired bit depth value.

Binning

Many SPOT family cameras support hardware binning. An application can determine if binning is supported by the current camera by the calling SpotGetValue with SPOT_BINSIZES. Binning can be enabled by setting the SPOT_BINSIZE parameter to the desired value. When binning is enabled, the values of the pixels in each *n* x *n* (where *n* is the bin size) square on the image sensor are added together as the data is read. For color mosaic cameras which support hardware binning, only even bin sizes are supported, and binned images are monochrome.

Future mosaic camera models may support color binning. When color binning is enabled for these cameras, the resultant images will be color. An application can check the value of SPOT_COLORBINSIZES to determine which binning levels, if any, are supported by the current camera. The color bin size can be set by the SPOT_COLORBINSIZE parameter.

A bin size of one indicates no binning.

Imaging Area

An application can specify the area of the image sensor which is to be used for image acquisition and/or exposure computation by setting the SPOT_IMAGERECT or SPOT_EXPOSURECOMPRECT parameters. By default, the full sensor imaging area is used. The size of the image

sensor can be determined by calling SpotGetValue with SPOT_MAXIMAGERECTSIZE.

Live Image Mode

Live image mode provides an efficient mechanism for quickly acquiring images for purposes of positioning and focusing. An application can check the SPOT_ATTR_LIVEMODE attribute bit to determine if the current camera supports live mode. Live images are always 8 (monochrome) or 24 (color) bpp. On many SPOT family cameras, dual amplifier circuits are used to simultaneously read data from the two halves of the image sensor, so the image readout is twice as fast, but there may be a slight brightness variation between the left and right halves of images.

The SpotGetLiveImages function is used to acquire live images. The caller must provide a single image data buffer of the appropriate size and bit depth to which all acquired images will be written. The SPOT_ACQUIREDLIVEIMAGESIZE parameter can be queried after the various parameters have been set to determine the width and height of the images which will be acquired. The function will run in a loop until it is aborted, and the application will be notified as each image frame is ready to be displayed or processed.

Some cameras use separate amplifier circuits for live mode an still capture, so the exposures required to acquire images of a particular brightness level in the two modes may be slightly different. For this reason, an exposure conversion factor is provided. The exposure conversion factor can be applied to live mode exposures to obtain equivalent exposures for still image capture. An application can query the value of SPOT_EXPOSURECONVFACTOR or SPOT_EXPOSURECONVFACTORX1000 to obtain the pre-programmed conversion factor, or it can call SpotComputeExposureConversionFactor or SpotComputeExposureConversionFactorX1000 to compute an appropriate conversion factor for the current conditions.

Still Image Capture

Still images are captured by calling SpotGetImage or SpotGetSequentialImages. For rapid acquisition of multiple images (streaming) or time-lapse acquisition, SpotGetSequentialImages should be used. Applications must provide image buffers of the correct size. The SPOT_ACQUIREDIMAGESIZE parameter can be queried after the various parameters have been set to determine the width and height of the images which will be acquired.

Horizontal Readout Frequency

Some SPOT family cameras provide multiple horizontal readout frequencies for still image capture. The horizontal readout frequency is the frequency at which the pixels on each line of the image sensor are read. An application can specify a higher frequency to increase data throughput or a lower frequency to reduce readout noise. The SPOT_HORIZREADOUTFREQUENCIES value can be queried to determine which readout frequency choices are available for the current camera. The readout frequency is set by the SPOT_HORIZREADOUTFREQUENCY parameter.

Shifting-Pixel Acquisition

Some SPOT cameras have the ability to shift the position of their image sensor. With these cameras, higher resolution images can be obtained by acquiring multiple images with the image sensor in different positions. An application can determine if a camera has this capability by checking the SPOT_ATTR_SENSORSHIFTING bit of the value returned by SpotGetCameraAttributes. The value of SPOT_MAXPIXELRESOLUTIONLEVEL specifies the number of extra higher resolution levels that are available with the current camera. An application can specify which resolution level to use by setting SPOT_PIXELRESOLUTIONLEVEL. The default value is zero.

When SpotGetImage or SpotGetSequentialImages are called for higher resolution acquisition, multiple shots of the same exposure duration will be done for each image to be acquired. The value of lInfo provided with the SPOT_STATUSGETIMAGE status specifies how many exposures will be done per image. Higher resolution levels are not supported for live mode.

Image Sensor Clear Modes

Some SPOT family cameras provide options as to how the image sensor should be cleared of accumulated charge when not acquiring images. Normally, the cameras perform continuous sensor clearing when idle. On many cameras, the clear cycle will be immediately interrupted when an exposure is to begin. For some cameras, the default clearing behavior can be overridden. An application can query the SPOT_CLEARMODES parameter to determine which mode options, if any, are available for the current camera. The clear mode is set via the SPOT_CLEARMODE parameter. The clear mode options are represented by the defined SPOT_CLEARMODE*xxx* values.

Image Enhancements and Corrections

The SpotCam driver provides several options for improving the quality of acquired images. An application can enable any of these options by setting the appropriate parameters.

Black-Level Subtraction

Normally, cameras are configured so that pixels with no light reaching them will have a value of zero. For monochrome still image capture at bit depths greater than 8 bpp, an application can specify that the black level be set above zero by setting the SPOT_SUBTRACTBLACKLEVEL parameter to FALSE.

Sensor Defect Correction

Since all image sensor chips contain some defective pixels and columns, each SPOT family camera is tested at assembly time to determine which pixels and columns are defective, and the locations of these pixels and columns are stored in the camera's non-volatile memory. An application can enable defect correction for still image capture by setting the SPOT_CORRECTCHIPDEFECTS parameter to TRUE. When defect correction is enabled, the SpotCam driver will replace the values of defective pixels and columns with the weighted averages of neighboring pixels and columns.

Color Enhancement

For color cameras, the SpotCam driver provides a color enhancement feature which makes image colors in appear more accurate. Specific color profiles are provided for each camera model. An application can obtain the color profile for the current camera by calling SpotGetValue with the SPOT_COLORPROFILE parameter. Likewise, the application can specify its own color profile for color enhancement by setting the SPOT_COLORPROFILE parameter. Color enhancement can be enabled separately for still image acquisition and live mode. The SPOT_ENHANCECOLORS and SPOT_LIVEENHANCECOLORS parameters are used to enable color enhancement for still acquisition and live mode, respectively.

Noise Filtering

The SpotCam driver provides a noise filter which replaces individual noisy pixels with the weighted average of the neighboring pixels. A pixel is considered to be noisy if its value differs from that of its nearest neighbors by more than the application-specified percentage. The noise threshold percentage is set via the SPOT_NOISEFILTERTHRESPCT parameter. A value of zero disables noise filtering. Noise filtering is not supported in live mode.

Bias Frame Subtraction

The SpotCam driver provides the ability to acquire bias frames and subtract them from acquired images. A bias frame is acquired with all light blocked from the camera and contains information regarding pixel offset and shading. When acquiring a still image with black-level subtraction disabled, the bias frame can be subtracted from the image data to produce a consistent black background.

Bias frames are acquired by calling SpotGetBiasFrame. When auto-exposure is enabled (SPOT_AUTOEXPOSE == TRUE), separate bias frames will be acquired for each allowable gain level. When auto-exposure is disabled, bias frames will be acquired at the currently set gain value.

Bias frame subtraction is enabled by setting the SPOT_BIASFRMSUBTRACT parameter to the name of the file containing the bias frame. A bias frame can only be used to correct images from the same camera from which it was obtained, and for the same gain, bin size, and imaging area which were used to acquire it. Bias frame subtraction is only applicable when black-level subtraction is disabled and is not supported in live mode.

Flatfield Correction

Flatfield correction is used to compensate for uneven illumination. An application can acquire a flatfield to use for corrections by calling SpotGetFlatfield or SpotGetFlatfield2 and can set the flatfield to be used by setting the SPOT_FLATFLDCORRECT parameter to the name of the file containing the flatfield.

A flatfield image can only be used to correct images acquired with the same camera. The flatfield must have been acquired with a bit depth greater than or equal to the bit depth of the image to be corrected, and for color filter cameras, the flatfield must contain all of the color channels which the image to be corrected contains. For shifting-pixel cameras, a flatfield can only be used to correct images acquired at the same resolution level. The area of the sensor used for the flatfield acquisition must completely contain the area used for the image to be corrected. Flatfield correction is not supported in live mode.

Background Image Subtraction

Background image subtraction is used to remove the image components attributable to dark current in the image sensor and extraneous background illumination. A background image can be acquired by calling SpotGetBackgroundImage, and background image subtraction can be

enabled by setting the SPOT_BKGDIMAGESUBTRACT to the name of the file containing the background image.

A background image can only be used to correct images acquired by the same camera. The background image must have been acquired with a bit depth greater than or equal to the bit depth of the image to be corrected, and for color filter cameras, the background image must contain all of the color channels which the image to be corrected contains. For shifting-pixel cameras, a background image can only be used to correct images acquired at the same resolution level. The area of the sensor used for the background image acquisition must completely contain the area used for the image to be corrected. Background image subtraction is not supported in live mode.

Exposure Timestamps

The SpotCam driver provides timestamps for still captured images. The timestamp values are recorded at the time that the first exposure of the acquisition begins. An application can query the driver for timestamp of the current or last acquisition by calling SpotGetExposureTimestamp.

Temperature Readout and Regulation

Some SPOT cameras provide image sensor temperature readout and regulation capabilities. An application can check the SPOT_ATTR_TEMPERATUREREADOUT and SPOT_ATTR_TEMPERATUREREGULATION attribute bits returned by SpotGetCameraAttributes to determine if the current camera supports these capabilities. An application can request the current sensor temperature by calling SpotGetSensorCurrentTemperature. The camera automatically saves the sensor temperature value at the beginning of each exposure, and that value can be obtained after the exposure completes by calling SpotGetSensorExposureTemperature. An application can set the regulation temperature by setting the SPOT_REGULATETEMPERATURE parameter. All temperature values are expressed in tenths of a degree C.

External Trigger Input

Many SPOT family cameras provide an external trigger feature which allows external hardware to precisely control exposures. There are two basic external trigger modes: edge mode and bulb mode. When external triggering is enabled, the camera will wait for a trigger pulse before beginning an exposure. In edge trigger mode, the length of the exposure will be determined by the exposure setting. In bulb mode, the length of the exposure is controlled by the width of the trigger pulse. With some cameras, an application can specify whether the trigger active state is high or low, and a delay, in microseconds, for the camera to wait between the trigger and the beginning of exposure.

For sequential image acquisition, if edge trigger mode is being used, the application can specify whether a separate trigger pulse should be required for each image acquisition, or if a single pulse should trigger the sequence. External triggers cannot be used with auto-exposure, and bulb trigger mode can only be used for single-shot (monochrome or mosaic) acquisition.

An application can determine which external trigger modes, if any, are supported by the current camera by checking the SPOT_ATTR_EDGETRIGGER and SPOT_ATTR_BULBTRIGGER returned by SpotGetCameraAttributes. It can determine whether or not trigger delay is supported by getting the values for SPOT_EXTERNALTRIGGERDELAYLIMITS. The CAMERA_ATTRIBUTE_TRIGACTIVESTATE attribute bit returned by SpotGetCameraAttributes indicates whether or not the trigger active state can be set for the current camera. The trigger mode is set by the SPOT_EXTERNALTRIGGERMODE parameter. The trigger active state is set by the SPOT_EXTERNALTRIGGERACTIVESTATE parameter, and the trigger delay is set by SPOT_EXTERNALTRIGGERDELAY.

TTL Output

All SPOT family cameras provide a TTL output signal which can be used to control external hardware. When TTL output is enabled, the TTL signal will be set to the active state before an exposure begins. An application can specify whether the TTL output active state is high or low and the amount of delay time between the setting of the TTL output and the beginning of exposure. Some cameras provide accurate TTL output delay timing to microsecond resolution. An application can check the SPOT_ATTR_ACCURATETTLDELAYTIMING attribute bit returned by SpotGetCameraAttributes to determine whether or not the current camera supports accurate TTL output delay timing. The TTL output active state is set by the SPOT_TTLOUTPUTACTIVESTATE parameter. TTL output for exposures is enabled with the SPOT_TTLOUTPUTENABLE parameter, and the delay is set with SPOT_TTLOUTPUTDELAY.

For many operations, such as live mode and exposure computation, if TTL output is enabled, the output will be set active once at the beginning of the operation and inactive at the end of the operation. For sequential image acquisition, an application can specify whether the TTL output should be set for each image acquisition, or once for the entire sequence. TTL output and external trigger can not both be enabled concurrently. When TTL output is disabled, the output signal will remain constant the inactive state level.

Status Notifications

The SpotCam driver normally uses an application-supplied callback function to notify the application when a status change event (eg. exposure begin, live image ready) occurs. An application can register a callback by calling SpotSetCallback. For situations where it is impractical or impossible to use a callback function (eg. LabView), an alternate status change notification mechanism is provided. If an application cannot register a callback function, it should set the SPOT_WAITFORSTATUSCHANGES parameter to TRUE after calling SpotInit. When this parameter is TRUE, API functions which perform lengthy camera operations, such as SpotComputeExposure, SpotGetImage, SpotGetLiveImages, etc., will return immediately with the SPOT_RUNNING return code. The application will then need to call SpotWaitForStatusChange in a loop until the returned status value is SPOT_STATUSIDLE, SPOT_STATUSABORTED, or SPOT_STATUSERROR. The lInfo value returned with the status will indicate the return value of the operation. Once SPOT_WAITFORSTATUSCHANGES has been set to TRUE, it cannot be reset until SpotExit is called.

The following code fragment illustrates use of SPOT_WAITFORSTATUSCHANGES and SpotWaitForStatusChange:

```
BOOL bVal;
int nRetVal, nStatus;
long lInfo;

bVal = TRUE;
SpotSetValue(SPOT_ WAITFORSTATUSCHANGES, &bVal);
nRetVal = SpotGetLiveImages(TRUE, SPOT_COLORRGB, SPOT_ROTATENONE, FALSE, FALSE, pImageBuffer);
if (nRetVal == SPOT_RUNNING)
{
   while (TRUE)
   {
      MSG stMsg;
      // Process OS events
      while (PeekMessage(&stMsg, 0, 0, 0, PM_REMOVE))
      {
         if (stMsg.message == WM_QUIT)
         {
            PostQuitMessage(stMsg.wParam);
            break;
         }
         DispatchMessage(&stMsg);
      }
      if (SpotWaitForStatusChange(&nStatus, &lInfo, 250))
      { // We got a notification
         switch (nStatus)
         {
         case SPOT_STATUSERROR:
            DisplayError(lInfo); // Display the error to the user
         case SPOT_STATUSIDLE:
         case SPOT_STATUSABORTED:
            return;
         case SPOT_STATUSLIVEIMAGEREADY:
            DisplayImage();  // Display the new live image frame
         }
      }
   }
```

Shutting Down the Driver

After the application has finished with a camera, it should call SpotExit to release the resources which were allocated for the camera.

Special Issues

Compiler Floating-Point Compatibility

The SpotCam driver uses the IEEE754 representation for float values. For applications which use a different representation, alternative versions of functions and parameters which use integers instead of floats are provided. These functions and parameters have names ending with "X1000".

Actual Gain Values

Gain values throughout the SpotCam API are represented as integers. The SpotGetActualGainValue and SpotGetActualGainValueX1000 functions can be called to obtain the actual non-integer gain values. For most SPOT cameras, the values returned by these functions will be the same as the integer gain values.

Gains below 1.0

Some SPOT family cameras provide actual gains less than 1.0. These small gain values should only be used when SPOT_BINSIZE is set to a value of 2 or greater, or the pixels in acquired images will saturate below full-scale.

Gain Ports

Future SPOT camera models may support more than one method of setting gain or "gain port". An application can determine how many gain ports are supported by the current camera by getting the value of SPOT_MAXGAINPORTNUMBER. For all current camera models, this value will be zero. An application can specify which gain port to use by setting SPOT_GAINPORTNUMBER. For gain port 0, the driver will provides a list of supported gain values (SPOT_PORT0GAINVALS8 and SPOT_PORT0GAINVALS12). For the other gain ports, the driver provides the lower and upper gain value limits (SPOT_PORT*n*GAINVALLIMITS and SPOT_PORT*n*LIVEGAINVALLIMITS). For these gain ports, SpotGetActualGainValue or SpotGetActualGainValueX1000 should be called to determine the actual gain values.

Long Exposures

In most of the SpotCam API functions and parameters which deal with exposures, the exposure duration values are expressed as unsigned 32-bit integers in increments which are 500ns by default. The longest exposure which can be expressed this way is approximately 35 minutes. Some SPOT cameras are capable of exposures longer than this. To express longer exposure durations, an application should either use the older functions and parameters (SPOT_EXPOSURE, SpotComputeExposure) which use millisecond increments, or it should set the exposure increment (SPOT_EXPOSUREINCREMENT) to a multiple of 500 (eg. 1000000 for 1 ms).

System Event Handling

The driver will automatically allows events in the calling application thread's event queue to be processed while executing long operations such as exposing. This allows UI events such as abort button pushes to be handled. An application can disable event handling by setting the SPOT_MESSAGEENABLE parameter to FALSE. It may be advisable to disable this feature when timestamp API functions are being called by threads other than the thread handling the application's UI.

Synchronizing Exposures with Other Processes

Some applications may need to synchronize the camera exposures with some external process, such as changing the illumination or specimen position. The external trigger and TTL output features provide two hardware methods of achieving such synchronization. If a software method is needed, the application-supplied callback function or SpotWaitForStatusChange function can be used. When SpotGetImage or SpotGetSequentialImages are called, the SpotCam driver will notify the application with one of the SPOT_STATUSSHUTTEROPEN*xxx* status values immediately prior to the beginning of each exposure. Except in the case where SpotGetSequentialImages has been called with nIntervalMSec = SPOT_INTERVALSHORTASPOSSIBLE, the exposure will not actually begin until the callback or SpotWaitForStatusChange returns. If the application is not ready for the exposure at this time, it can block the return of its callback or defer calling SpotWaitForStatusChange until it is safe to expose. As soon as the exposure completes, the application will be notified with one of the SPOT_STATUSIMAGEREAD*xxx* status values. At that time, the application can safely assume that the exposure has ended.

Obtaining Raw Color Mosaic Pixel Data

The SpotCam driver provides the ability for applications to obtain the raw pixel data from mosaic cameras for still captured images. The raw pixel data is saved as a monochrome 12 or 14 bpp image. To obtain raw mosaic data, the application must set SPOT_RETURNRAWMOSAICDATA to TRUE and SPOT_BITDEPTH to the maximum monochrome bit depth supported by the camera before calling the acquisition functions.

IEEE1394/FireWire Isochronous Bus Bandwidth

When controlling an IEEE1394/FireWire camera, the SpotCam driver allocates isochronous bus bandwidth as necessary. By default, the driver will allocate as much bandwidth as is needed for efficient data transfer. Howevere, an application can limit the amount of bandwidth which the driver is allowed to allocate by setting the SPOT_BUSBANDWIDTH value to SPOT_MEDIUMBW or SPOT_LOWBW. Reducing the bandwidth will increase the amount of time needed for image data transfers. On slow machines, reducing bandwidth may help to reduce or eliminate image data corruption problems.

Device Change Notifications

For some SPOT family camera models, the SpotCam driver has the ability to notify an application when a camera is added or removed from the machine. The application can register a callback function for receiving these notifications with the SpotSetDeviceNotificationCallback function. Device change notification is not provided for 1394/FireWire cameras on Win98/Me.

API Function Reference

The header file, SpotCam.h contains all of the header information necessary for interfacing to the driver. Refer to this file for information regarding prototypes and definitions. Information about the specifications and capabilities of the various camera models can be obtained from the user documentation for each model. The following sections describe the driver's API functions in alphabetical order.

SpotClearStatus

The SpotClearStatus function is used to reset the current status value to either SPOT_STATUSDRVNOTINIT or SPOT_STATUSIDLE, depending on whether or not the driver has been initialized.

```
void WINAPI SpotClearStatus();
```

Remarks

This function is useful for clearing the driver's status value after an error has occurred. It is a good idea to call this function at the beginning of each operation.

See Also

SpotQueryStatus, SpotSetCallback

SpotCloseExternalShutter

The SpotCloseExternalShutter function is used to explicitly set the TTL output to the inactive state.

```
int WINAPI SpotCloseExternalShutter();
```

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.

Remarks

The SpotOpenExternalShutter and SpotCloseExternalShutter functions can be used by the application to exercise more direct control over the TTL output signal. Alternatively, SpotSetTTLOutputState can be called to perform the same operations. If these functions are to be called, it may be advisable to disable the automatic control of the TTL output by setting SPOT_TTLOUTPUTENABLE to FALSE. The active state of the TTL output is set by the SPOT_TTLOUTPUTACTIVESTATE parameter.

See Also

SpotSetTTLOutputState, SpotOpenExternalShutter

SpotComputeExposure

The SpotComputeExposure function is used to compute appropriate exposure times and gain level for the image currently in the camera's view. If exposure resolution finer than 1ms is desired, SpotComputeExposure2 should be called instead.

```
int WINAPI SpotComputeExposure(
        EXPOSURE_STRUCT *pstExposure
);
```

Parameters

pstExposure

Points to an EXPOSURE_STRUCT structure to hold the computed exposure times and gain level.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The exposure computation will be affected by various parameter settings as indicated in the table in the appendix. The application should set all relevant parameters prior to calling this function. If the SPOT_EXPOSURECOMPRECT parameter has been set, the exposure will be computed for the pixels in the specified area. Otherwise, the exposure will be computed for the area specified by the SPOT_IMAGERECT parameter, or the full chip area, by default. The computed exposure times will be limited by the values set for the SPOT_MINEXPOSUREMSEC and SPOT_MAXEXPOSUREMSEC parameters, and the computed gain value will be limited by the value set for SPOT_AUTOGAINLIMIT. The computed exposure will be adjusted by the value set for the SPOT_BRIGHTNESSADJ or SPOT_BRIGHTNESSADJX1000 parameter. The SPOT_IMAGETYPE value will determine how the exposure is computed. For color exposures on color filter cameras, the ratio of the computed exposure times for the different colors will correspond to the current white balance values set by SPOT_WHITEBALANCE or SPOT_WHITEBALANCEX1000.

For monochrome exposures and mosaic cameras, the computed exposure time value will be placed in the lExpMSec member of the EXPOSURE_STRUCT structure. The application can force the camera to use the computed exposure values by setting the SPOT_AUTOEXPOSE parameter to FALSE and SPOT_EXPOSURE to the EXPOSURE_STRUCT filled by this function.

See Also

SpotComputeExposure2

SpotComputeExposure2

The SpotComputeExposure2 function is used to compute appropriate exposure times and gain level for the image currently in the camera's view. It provides finer exposure time resolution than SpotComputeExposure.

```
int WINAPI SpotComputeExposure2(
        EXPOSURE_STRUCT2 *pstExposure
```

);

Parameters

pstExposure

Points to an EXPOSURE_STRUCT2 structure to hold the computed exposure times and gain level.

Return Value

Refer to the section for SpotComputeExposure.

Remarks

The exposure computation will be affected by various parameter settings as indicated in the table in the appendix. The application should set all relevant parameters prior to calling this function. If the SPOT_EXPOSURECOMPRECT parameter has been set, the exposure will be computed for the pixels in the specified area. Otherwise, the exposure will be computed for the area specified by the SPOT_IMAGERECT parameter, or the full chip area, by default. The computed exposure times will be limited by the values set for the SPOT_MINEXPOSUREMSEC and SPOT_MAXEXPOSUREMSEC parameters, and the computed gain value will be limited by the value set for SPOT_AUTOGAINLIMIT. The computed exposure will be adjusted by the value set for the SPOT_BRIGHTNESSADJ or SPOT_BRIGHTNESSADJX1000 parameter. The SPOT_IMAGETYPE value will determine how the exposure is computed. For color exposures on color filter cameras, the ratio of the computed exposure times for the different colors will correspond to the current white balance values set by SPOT_WHITEBALANCE or SPOT_WHITEBALANCEX1000.

The exposure time values returned via the EXPOSURE_STRUCT2 structure are expressed in the increments specified by the SPOT_EXPOSUREINCREMENT parameter (500 ns by default). For monochrome exposures and mosaic cameras, the computed exposure time value will be placed in the dwExpCur member of the EXPOSURE_STRUCT2 structure. The application can force the camera to use the computed exposure values by setting the SPOT_AUTOEXPOSE parameter to FALSE and SPOT_EXPOSURE2 to the EXPOSURE_STRUCT2 filled by this function.

See Also

SpotComputeExposure

SpotComputeExposureConversionFactor

The SpotComputeExposureConversionFactor function is used to compute a factor which can be used for converting live mode exposure values for still image capture for cameras which have separate amplifier circuits for live and still image capture modes.

```
int WINAPI SpotComputeExposureConversionFactor(
    float *pfConvFactor
);
```

Parameters

pfConvFactor

Points to an float value to hold the computed conversion factor.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.

SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The exposure conversion factor is useful in acquiring images with the same brightness as live mode images. An application should check the SPOT_ATTR_DUALAMPLIFIER bit returned by SpotGetCameraAttributes to determine whether or not it needs to call this function. If this bit is 0, this function need not be called. The exposure conversion factor will be computed for the area specified by the SPOT_EXPOSURECOMPRECT parameter.

SpotComputeExposureConversionFactorX1000

The SpotComputeExposureConversionFactor function is used to compute a factor which can be used for converting live mode exposure values for still image capture for cameras which have separate amplifier circuits for live and still image capture modes. It is the same as SpotComputeExposureConversionFactor, except that the returned value is an integer.

```
int WINAPI SpotComputeExposureConversionFactor(
    long *plConvFactor
);
```

Parameters

plConvFactor

Points to an long value to hold the computed conversion factor multiplied by 1000.

Return Value

Refer to the section for SpotComputeExposureConversionFactor.

Remarks

Refer to the section for SpotComputeExposureConversionFactor.

SpotComputeWhiteBalance

The SpotComputeWhiteBalance function is used to compute white balance values for the image currently in the camera's view.

```
int WINAPI SpotComputeWhiteBalance(
    WHITE_BAL_STRUCT *pstWhiteBal
);
```

Parameters

pstWhiteBal

Points to a WHITE_BAL_STRUCT structure to hold the computed white balance values.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	**Meaning**
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.

SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_ERRVALOUTOFRANGE	The computed white balance is out of range for the camera.
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

This function acquires one or more images to determine the correct RGB ratios so that the brightest pixels will be white. It is therefore necessary the the image in view contains a white area. If the SPOT_WHITEBALCOMPRECT parameter has been set prior to calling this function, the specified area will be used for computing white balance. Otherwise the entire sensor area will be used, except that for color mosaic cameras, the outermost 100 pixels on all sides are ignored. The application can force the camera to use the computed white balance values by setting SPOT_WHITEBALANCE parameter to the WHITE_BAL_STRUCT filled by this function. This function is only supported for color cameras. The maximum white balance value which can be returned by this is specified by the SPOT_MAXWHITEBALANCERATIO value.

See Also

SpotComputeWhiteBalanceX1000

SpotComputeWhiteBalanceX1000

The SpotComputeWhiteBalanceX1000 function is used to compute white balance values for the image currently in the camera's view. It is the same as the SpotComputeWhiteBalance function, except that it returns results as integers in a SPOT_WHITE_BAL_INT_STRUCT structure.

```
int WINAPI SpotComputeWhiteBalanceX1000(
        WHITE_BAL_INT_STRUCT *pstWhiteBal
);
```

Parameters

pstWhiteBal

Points to a WHITE_BAL_INT_STRUCT structure to hold the computed white balance values.

Return Value

Refer to the section for SpotComputeWhiteBalance

Remarks

This function acquires one or more images to determine the correct RGB ratios so that the brightest pixels will be white. It is therefore necessary the the image in view contains a white area. If the SPOT_WHITEBALCOMPRECT parameter has been set prior to calling this function, the specified area will be used for computing white balance. Otherwise the entire sensor area will be used, except that for color mosaic cameras, the outermost 100 pixels on all sides are ignored. The application can force the camera to use the computed white balance values by setting SPOT_WHITEBALANCEX1000 parameter to the SPOT_WHITE_BAL_INT_STRUCT filled by this function. This function is only supported for color cameras. The maximum white balance value which can be returned by this is specified by the SPOT_MAXWHITEBALANCERATIOX1000 value.

See Also

SpotComputeWhiteBalance

SpotDumpCameraMemory

The SpotDumpCameraMemory function is used to dump the contents of the camera's memory to a file for diagnostic purposes.

```
int WINAPI SpotDumpCameraMemory(
        char *pszFileName
);
```

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	**Meaning**
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory

Remarks

This function can be used to dump the contents of the camera's memory to a file to be sent back to Diagnostic Instruments, Inc., upon request, for diagnosing a camera which is malfunctioning.

SpotExit

The SpotExit function is used to de-initialize the SPOT camera driver, releasing memory and resources allocated by the driver for the camera.

```
int WINAPI SpotExit( );
```

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.

Remarks

This function must be called once for each successful call to SpotInit. It is safe to call this function multiple times.

See Also

SpotInit

SpotFindDevices

The SpotFindInterfaceCards function is used to obtain a list of available camera devices on the machine.

```
int WINAPI SpotFindDevices(
        SPOT_DEVICE_STRUCT *pstDevices,
        int * pnNumDevices
);
```

Parameters

pstDevices

Points to an array of SPOT_DEVICE_STRUCT structures.

pnNumDevices

Points to an int value to receive the number of devices found by the function.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDEVDRVLOAD	There was an error loading or opening the device driver.
SPOT_ERRNODEVICESFOUND	No supported devices were found.
SPOT_ERROUTOFMEMORY	out of memory

Remarks

This function supersedes SpotFindInterfaceCards. It can be called to obtain a list of all available camera devices. The pstDevices parameter should point to a block of memory large enough to hold SPOT_MAX_DEVICES structures. The text strings in the szDescription member of the SPOT_DEVICE_STRUCT structs can be displayed to users to allow them to select the desired device. The DeviceUID member of the SPOT_DEVICE_STRUCT will only be filled for cameras which are currently connected and running.

To specify which device the driver should use, before calling SpotInit, an application must set either the SPOT_DEVICEUID or SPOT_DRIVERDEVICENUMBER parameter. The SPOT_DEVICEUID parameter is set with one of the DeviceUID values obtained by this function. The SPOT_DRIVERDEVICENUMBER parameter is set with the zero-based index of one of the SPOT_DEVICE_STRUCT structs obtained by this function.

See Also

SpotFindInterfaceCard

SpotFindInterfaceCards

The SpotFindInterfaceCards function has been superseded by SpotFindDevices. It is used to obtain a list of available camera interface cards on the machine.

```
void WINAPI SpotFindInterfaceCards(
        SPOT_INTF_CARD_STRUCT *pstIntfCards,
        int *pnNumIntfCards
);
```

Parameters

pstIntfCards

Points to an array of SPOT_INTF_CARD_STRUCT structures.

pnNumIntfCards

Points to an integer value to receive the number of interface cards found by the function.

Return Value

Refer to the section for SpotFindDevices.

Remarks

This function can be called to obtain a list of all available camera interface cards. The pstIntfCards parameter should point to a block of memory large enough to hold SPOT_MAX_INTF_CARDS structures.

For backward-compatibility, all newer camera device models will be mapped to one of the existing SPOT_CARDTYPE_*xxx* values defied in SpotCam.h. Insight 1394/FireWire cameras will therefore appear as Insight PCI cards.

An application can specify which device the driver should use by setting the SPOT_DRIVERDEVICENUMBER parameter with the zero-based index of one of the SPOT_INTF_CARD_STRUCT structs obtained by this function prior to calling SpotInit.

See Also

SpotFindDevices

SpotGetActualGainValue

The SpotGetActualGainValue function is used to get actual gain values for cameras which provide non-integer gains.

```
int SpotGetActualGainValue(
        int nGainPort,
        int nGainValue,
        float *pfActualGainValue
);
```

Parameters

nGainPort

The gain port to be used.

nGainValue

The integer gain value to be used.

pfActualGainValue

Points to a float to receive the actual gain value.

Return Value

If the function is successful, it will return SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRVALOUTOFRANGE	A value passed to this function is out of range.

Remarks

Some SPOT family cameras provide non-integer gain levels. This function can be used to determine what the actual gain value is for any particular gain level.

See Also

SpotGetActualGainValueX1000, SpotGetActualLiveGainValue, SpotGetActualLiveGainValueX1000

SpotGetActualLiveGainValue

The SpotGetActualGainLiveValue function is used to get actual live mode gain values for cameras which provide non-integer gains.

```
int SpotGetActualLiveGainValue(
        int nGainPort,
        int nGainValue,
        float *pfActualGainValue
);
```

Parameters

nGainPort

The gain port to be used.

nGainValue

The integer gain value to be used.

pfActualGainValue

Points to a float to receive the actual gain value.

Return Value

If the function is successful, it will return SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRVALOUTOFRANGE	A value passed to this function is out of range.

Remarks

Some SPOT family cameras provide non-integer gain levels. This function can be used to determine what the actual gain value is for any particular live mode gain level.

See Also

SpotGetActualGainValue, SpotGetActualGainValueX1000, SpotGetActualLiveGainValueX1000

SpotGetActualGainValueX1000

The SpotGetActualGainValueX1000 function is used to get actual gain values for cameras which provide non-integer gains. It is the same as SpotGetActualGainValue, except that the actual gain value is expressed as an integer.

```
int SpotGetActualGainValueX1000(
        int nGainPort,
        int nGainValue,
        int *pnActualGainValue
);
```

Parameters

nGainPort

The gain port to be used.

nGainValue
The integer gain value to be used.
pnActualGainValue
Points to a int to receive the actual gain value multiplied by 1000.

Return Value
If the function is successful, it will return SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRVALOUTOFRANGE	A value passed to this function is out of range.

Remarks
Refer to the section for SpotGetActualGainValue.

See Also
SpotGetActualGainValue, SpotGetActualLiveGainValue, SpotGetActualLiveGainValueX1000

SpotGetActualLiveGainValueX1000

The SpotGetActualLiveGainValueX1000 function is used to get actual live mode gain values for cameras which provide non-integer gains. It is the same as SpotGetActualLiveGainValue, except that the actual gain value is expressed as an integer.

```
int SpotGetActualLiveGainValueX1000(
        int nGainPort,
        int nGainValue,
        int *pnActualGainValue
);
```

Parameters
nGainPort
The gain port to be used.
nGainValue
The integer gain value to be used.
pnActualGainValue
Points to a int to receive the actual gain value multiplied by 1000.

Return Value
If the function is successful, it will return SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRVALOUTOFRANGE	A value passed to this function is out of range.

Remarks
Refer to the section for SpotGetActualLiveGainValue.

See Also
SpotGetActualGainValue, SpotGetActualGainValueX1000, SpotGetActualLiveGainValue

SpotGetBackgroundImage

The SpotGetBackgroundImage function is used to acquire a background image for correcting acquired images.

```
int WINAPI SpotGetBackgroundImage(
        char *pszFileName,
        int nNumFramesToAvg
```

);

Parameters

pszFileName

Points to NULL-terminated character string which specifies the name of the file to be created to hold the acquired background image.

nNumFramesToAvg

The number of image frames to acquire and average.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBKGDTOOBRIGHT	The image background is too bright.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRFILEOPEN	The background image file cannot be created or written.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The file name string should contain a fully qualified path name. If auto-exposure is enabled when this function is called, exposure will be computed and the application's callback function will be called with the SPOT_STATUSWAITINGFORMOVETOBKGD status to indicate that the specimen should be removed so that background image acquisition can proceed. After the specified number of images are be acquired, the application's callback will be called with the SPOT_STATUSWAITINGFORBLOCKLIGHT status to indicate that all light to the camera should be blocked. Background image subtraction is applied during the post-processing of still captured images. The SPOT_BKGDIMAGESUBTRACT must be set to enable bias frame subtraction. Each background image can only be used to correct images acquired with the same camera.

When calling this function, the application must specify a number of frames to acquire. The driver will acquire the specified number of images and average their values to reduce read noise.

See Also

SpotGetBiasFrame, SpotGetFlatfield, SpotGetFlatfield2

SpotGetBiasFrame

The SpotGetBiasFrame function is used to acquire a bias frame for correcting acquired images.

```
int WINAPI SpotGetBiasFrame(
        char *pszFileName,
        int nNumFramesToAvg
);
```

Parameters

pszFileName

Points to NULL-terminated character string which specifies the name of the file to be created to hold the acquired bias frame.

nNumFramesToAvg

The number of image frames to acquire and average.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRFILEOPEN	The bias frame file cannot be created or written.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The file name string should contain a fully qualified path name. If auto-exposure is enabled when this function is called, acquisitions will be done at various gain levels. External trigger and TTL output settings are ignored. Bias frame subtraction is applied during the post-processing of still captured images. The SPOT_BIASFRMSUBTRACT must be set to enable bias frame subtraction. Each bias frame can only be used to correct images acquired with the same camera.

When calling this function, the application must specify a number of frames to acquire. The driver will acquire the specified number of images and average their values to reduce read noise. All light to the camera should be blocked before calling this function.

See Also

SpotGetFlatfield, SpotGetFlatfield2, SpotGetBackgroundImage

SpotGetCameraAttributes

The SpotGetCameraAttributes function is used to get the current camera's attributes.

```
int WINAPI SpotGetCameraAttributes(
        DWORD *pdwAttributes
);
```

Parameters

pdwAttributes

Points to a DWORD to hold the retrieved camera attribute flags

Return Value

If the function is successful, it will return SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.

Remarks

The attributes are returned as bit-mapped flags **OR**ed together. The values can be one or more of the following:

Flag	Meaning
SPOT_ATTR_1394	The camera is a 1394/FireWire device.
SPOT_ATTR_ACCURATETTLDELAYTIMING	The camera can provide accurate timing µsecond timing of external trigger and/or TTL output delays.
SPOT_ATTR_AUTOEXPOSURE	The camera can compute exposure and white balance (if applicable).
SPOT_ATTR_BULBTRIGGER	The camera accepts an external trigger input and supports bulb trigger mode.

SPOT_ATTR_CLEARFILTER	The camera has a color filter with a clear setting.
SPOT_ATTR_COLOR	The camera can return color images.
SPOT_ATTR_COLORFILTER	The camera has a color filter and acquires color images with multiple shots.
SPOT_ATTR_DUALAMPLIFIER	The camera has separate amplifier circuits for live and captured image modes.
SPOT_ATTR_EDGETRIGGER	The camera accepts an external trigger input and supports edge trigger mode.
SPOT_ATTR_FILTERWHEEL	The camera has a mechanical color filter wheel.
SPOT_ATTR_LIVEMODE	The camera supports the live image mode.
SPOT_ATTR_MOSAIC	The camera has a color mosaic sensor.
SPOT_ATTR_SENSORSHIFTING	The camera can shift the position of the image sensor for greater pixel resolution.
SPOT_ATTR_SLIDER	The camera has a sliding color filter.
SPOT_ATTR_SLIDERPOSITIONDETECTION	The camera can detect the color filter slider position.
SPOT_ATTR_TEMPERATUREREADOUT	The camera can provide readings of the image sensor's temperature.
SPOT_ATTR_TEMPERATUREREGULATION	The camera can regulate the image sensor's temperature.
SPOT_ATTR_TRIGGERACTIVESTATE	The camera's external trigger active state (high or low) can be set.
SPOT_ATTR_TTLOUTPUT	The camera has a TTL output

SpotGetFlatfield

The SpotGetFlatfield function is used to acquire a flatfield which can be used to correct acquired images.

```
int WINAPI SpotGetFlatfield(
        char *pszFileName
);
```

Parameters

pszFileName

Points to NULL-terminated character string which specifies the name of the file to be created to hold the acquired flatfield.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera’s color filter is not in the “Color” position.
SPOT_ERRCOLORFILTERNOTOUT	The camera’s color filter is not in the “B/W’ position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRFILEOPEN	The flatfield file cannot be created or written.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The file name string should contain a fully qualified path name. Separate flatfield files should be acquired for each magnification level. Flatfields are always acquired using auto-exposure, no binning, and the maximum bit depth per color channel. The currently enabled colors are used. The external trigger mode setting is ignored. Flatfield correction is applied during the post-processing of still captured images. The SPOT_FLATFLDCORRECT must be set to enable flatfield correction. A particular flatfield file can only be used to correct an acquired image if it was acquired with the same camera and with all of the filter colors used to acquire the image to be corrected.

See Also

SpotGetBiasFrame, SpotGetFlatfield2, SpotGetBackgroundImage

SpotGetFlatfield2

The SpotGetFlatfield2 function is used to acquire a flatfield for correcting acquired images. It is a replacement for SpotGetFlatfield, providing better results.

```
int WINAPI SpotGetFlatfield2(
        char *pszFileName,
        int nNumFramesToAvg
);
```

Parameters

pszFileName

Points to NULL-terminated character string which specifies the name of the file to be created to hold the acquired flatfield.

nNumFramesToAvg

The number of image frames to acquire and average.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRFILEOPEN	The flatfield file cannot be created or written.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

The file name string should contain a fully qualified path name. Separate flatfield files should be acquired for each magnification level. Flatfields are always acquired using auto-exposure, no binning, and the maximum bit depth per color channel. The currently enabled colors are used. The external trigger mode setting is ignored. Flatfield correction is applied during the post-processing of still captured images. The SPOT_FLATFLDCORRECT must be set to enable flatfield correction. A particular flatfield file can only be used to correct an acquired image if it was acquired with the same camera and with all of the filter colors used to acquire the image to be corrected.

When calling this function, the application must specify a number of frames to acquire. The driver will acquire the specified number of images and average their values to reduce read noise. The driver will then call the application's callback with the SPOT_STATUSWAITINGFORBLOCKLIGHT status value which will indicate that all light should be blocked from the camera's sensor. The callback function should not return until the light has been blocked. When the callback returns, if the abort flag has not been set, the driver will acquire *nNumFramesToAvg* more images with a short exposure, average them, and subtract the average from the average of the first set of images.

See Also

SpotGetBiasFrame, SpotGetFlatfield, SpotGetBackgroundImage

SpotGetImage

The SpotGetImage function is used to capture still images.

```
int WINAPI SpotGetImage(
        short nReserved,
        BOOL bReserved,
        short nSkipLines,
        void *pImageBuffer,
        long *plRedPixelCnts,
        long *plGreenPixelCnts,
        long *plBluePixelCnts
);
```

Parameters

nReserved

This parameter was originally used to specify the bit depth of images to be captured. It should preferably be set to 0, but for backward-compatibility, it may be set to the image bit depth.

bReserved

Reserved for future use. This value should be set to FALSE.

nReserved

Reserved for future use. This value should be set to 0.

pImageBuffer

Points to the buffer where the acquired pixel values are to be placed by the driver. For 8 and 24 bpp images, this buffer corresponds to the bmBits member of a Windows BITMAP structure. See the Remarks section for more information.

plRedPixelCnts

Points to a buffer for holding red histogram values for the acquired image for color images. For monochrome images, the histogram values for the entire image are placed in this buffer. May be NULL.

plGreenPixelCnts

Points to a buffer for holding green histogram values for the acquired image. May be NULL.

plBluePixelCnts

Points to a buffer for holding blue histogram values for the acquired image. May be NULL.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBIASFRMINCOMPATIBLE	The bias frame file is not compatible with the current camera or settings
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRFILEOPEN	The bias frame, flatfield, or background image file cannot be opened or read.
SPOT_ERRFLATFLDINCOMPATIBLE	The flatfield file is not compatible with the current camera or settings
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_ERRBKGDIMAGEINCOMPATIBLE	The background image file is not compatible with the current camera or settings
SPOT_ERRVALOUTOFRANGE	A value passed to this function or SpotSetValue is out of range for the operation.
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

For 8 and 24 bpp images, this buffer pointed to by pImageBuffer must correspond to the bmBits member of a Windows BITMAP structure. The buffer must contain one byte per pixel for 8 bpp and 3 bytes per pixel for 24 bpp images, with zero to three padding bytes per image line so that the number of bytes per line is an integer multiple of four. For other bit depths, the buffer must contain one word per pixel for monochrome and three words (R-G-B) per pixel for color image. The image data is placed in the lowest bits of each word, and there is no padding. All images are stored in bottom-up orientation. An application can determine the width and height of the image which will be acquired by checking the value of SPOT_ACQUIREDIMAGESIZE after setting all relevant parameters.

If auto-exposure is enabled at the time this function is called, the exposure will be computed before an image is acquired. The exposure will be computed according to the same rules which apply to SpotComputeExposure and SpotComputeExposure2.

For monochrome images, the buffer pointed to by plRedPixelCnts (if any) will be filled with the histogram values regardless of which color is enabled.

When an external trigger is used in edge trigger mode, the camera will wait for the rising edge of an external trigger pulse before exposing. The camera will use the exposure specified by the SPOT_EXPOSURE or SPOT_EXPOSURE2 parameters.

When an external trigger is used in bulb trigger mode, the camera will expose until the falling edge of the pulse is detected. The camera will use the gain value specified by the SPOT_EXPOSURE or SPOT_EXPOSURE2 parameters. The minimum and maximum exposure durations are restricted by the camera hardware. Bulb trigger mode cannot be used with color images on non-mosaic color cameras.

See Also

SpotGetLiveImages, SpotGetSequentialImages

SpotGetLiveImages

The SpotGetLiveImages function is used to acquire a continuous stream of images.

```
int WINAPI SpotGetLiveImages(
        BOOL bComputeExposure,
        short nFilterColor,
        short nRotateDirection,
        BOOL bFlipHoriz,
        BOOL bFlipVert,
        void *pImageBuffer
);
```

Parameters

bComputeExposure

A flag specifying whether or not the exposure times and gain level should be computed before acquiring live images. If this value is FALSE, live image acquisition will be done using the exposure and gain specified by the SPOT_LIVEEXPOSURE parameter or the last used live image exposure and gain if the exposure was not explicitly set.

nFilterColor

For color filter cameras, a value specifying which filter color should be used for live image acquisition. Choices are SPOT_COLORRED, SPOT_COLORGREEN, SPOT_COLORBLUE, SPOT_COLORCLEAR, SPOT_COLORRGB, and SPOT_COLORNONE. If SPOT_COLORRGB is used, the acquired images will be 24 bpp. Otherwise, acquired images will be 8 bpp. SPOT_COLORNONE may be used if the color filter is in the “B/W” position on slider cameras. When using mosaic or monochrome cameras, this value is ignored.

nRotateDirection

A value specifying how acquired images should be rotated. Choices are SPOT_ROTATENONE, SPOT_ROTATELEFT, and SPOT_ROTATERIGHT. Values of SPOT_ROTATELEFT and SPOT_ROTATERIGHT will cause the acquired images to be rotated 90° to the left or right, respectively.

bFlipHoriz

A flag specifying whether or not the acquired images should be flipped horizontally.

bFlipVert

A flag specifying whether or not the acquired images should be flipped vertically.

pImageBuffer

Points to the buffer where the acquired images are to be placed by the driver. This buffer corresponds to the bmBits member of a Windows BITMAP structure. See the Remarks section for more information.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the 1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_ERRVALOUTOFRANGE	A value passed to this function or SpotSetValue is out of range for the operation.
SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

An application can determine if live mode is supported by the currently connected camera by calling SpotGetCameraAttributes and checking the SPOT_ATTR_LIVEMODE bit.

The function will not return until an error occurs or the operation is aborted. If the application has set a callback function, the callback function will be called with the status value SPOT_STATUSLIVEIMAGEREADY as each image is ready. An application can safely display or save the image at this time. The status value will be set to SPOT_STATUSABORTED when the operation is aborted by the application.

If bComputeExposure is TRUE, the exposure will be computed before live image acquisition begins. The computed gain will be limited to the value specified by the SPOT_LIVEAUTOGAINLIMIT parameter. The computed exposure will be limited by the SPOT_MINEXPOSUREMSEC and SPOT_LIVEMAXEXPOSUREMSEC parameter settings. The external trigger mode setting is ignored. The application can determine the exposure and gain used by querying the SPOT_LIVEEXPOSURE and SPOT_LIVEAUTOBRIGHTNESSADJ or SPOT_LIVEAUTOBRIGHTNESSADJX1000 parameters.

All images acquired by this function will be either 8 or 24 bpp and will be stored bottom-up as standard Windows bitmaps. The buffer must contain one byte per pixel for monochrome and 3 bytes per pixel for color images, with zero to three padding bytes per image line so that the number of bytes per line is an integer multiple of four. The application can determine the size of the images which will be acquired by checking the value of SPOT_ACQUIREDLIVEIMAGESIZE after setting all relevant parameters. The application must take into account whether or not the acquired images are to be rotated when allocating the image buffer. If nRotateDirection is not SPOT_ROTATENONE, the image will be rotated before being placed into the buffer.

The parameters: SPOT_LIVEENHANCECOLORS, SPOT_LIVEGAINADJ, SPOT_LIVEGAINADJX1000, SPOT_LIVEGAMMAADJ, SPOT_LIVEGAMMAADJX1000, and SPOT_LIVEEXPOSURE can be changed while the live image acquisition is running and will take effect immediately. Changes to other parameters will take effect only after live image acquisition has been stopped and restarted.

Once SpotGetLiveImages has been called, no calls should be made to SpotInit, SpotExit, or any function which requires image acquisition until the after function has returned. Calling any of these functions while live image acquisition is running will result in an error code being returned.

Images returned by this function may be of a lower quality than those returned by SpotGetImage or SpotGetSequentialImages. If a stream of high quality images is desired, SpotGetSequentialImages should be used instead.

See Also

SpotGetImage, SpotGetSequentialImages

SpotGetSensorCurrentTemperature

The SpotGetSensorCurrentTemperature function is used to obtain the current temperature of the camera's image sensor.

```
int WINAPI SpotGetSensorCurrentTemperature(
        short *pnTemperature,
        BOOL *pbIsNewValue
);
```

Parameters

pnTemperature

Points to a short integer to receive the temperature value, in tenths of a degree C.

pbIsNewValue

Points to a BOOL value which will be set to TRUE if the temperature value has been updated since the last time this function was called.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.
SPOT_ERROUTOFMEMORY	out of memory

Remarks

An application can call this function periodically to determine the current temperature of the camera's image sensor. The application can determine if the camera is capable of providing temperature readout by checking the SPOT_ATTR_TEMPERATUREREADOUT bit returned by SpotGetCameraAttributes.

SpotGetSensorExposureTemperature

The SpotGetSensorExposureTemperature function is used to obtain the temperature of the camera's image sensor at the time that the last exposure began.

```
int WINAPI SpotGetSensorExposureTemperature(
        short *pnTemperature
);
```

Parameters

pnTemperature

Points to a short integer to receive the temperature value, in tenths of a degree C.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.
SPOT_ERROUTOFMEMORY	out of memory

Remarks

An application can call this function after acquiring an image to determine the temperature of the camera's image sensor at the beginning of the exposure. The application can determine if the camera is capable of providing temperature readout by checking the SPOT_ATTR_TEMPERATUREREADOUT bit returned by SpotGetCameraAttributes.

SpotGetSequentialImages

The SpotGetSequentialImages function is used to acquire a series of images.

```
int WINAPI SpotGetSequentialImages(
        int nNumImages,
        int nIntervalMSec,
        BOOL bAutoExposeOnEach,
        BOOL bUseTriggerOrSetTTLOuputOnEach
        BOOL bDeferProcessing,
        void **ppImageBuffers
);
```

Parameters

nNumImages

The number of image to acquire.

nIntervalMSec

The interval to wait from the beginning of one acquisition to the next, in milliseconds. If this value is SPOT_INTERVALSHORTASPOSSIBLE, images will be acquired as quickly as possible. This value will be ignored if the external trigger is being used for each image.

bAutoExposureOnEach

A flag specifying whether the exposure times and gain level should be computed before acquiring each image. If auto-exposure is disabled, this parameter will be ignored. If auto-exposure is enabled, and this value is FALSE, the exposure and gain will only be computed before the first image, and the same exposure and gain will be used for each subsequent image in the sequence. If this value is TRUE, the exposure and gain will be recomputed for each image in the sequence.

bUseTriggerOrSetTTLOuputOnEach

A flag, which, if external triggering is enabled, specifies whether or not the acquisition of each image in the sequence should require an external trigger pulse, or, if TTL output is enabled, specifies whether or not the TTL output should be set for each image acquisition.

bDeferProcessing

A flag specifying that the processing of images should be deferred until all the images in the sequence have been acquired. This option saves processing time between image acquisitions and may be useful if a short interval is required.

ppImageBuffers

Points to an array of buffers where the pixel data of the acquired images are to be placed by the driver. Refer to the section on SpotGetImage for a description of the image buffers. This value may be NULL, in which case the application will need to call SpotRetrieveSequentialImage once for each image acquired.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ABORT	The operation was aborted by the application.
SPOT_ERRBIASFRMINCOMPATIBLE	The bias frame file is not compatible with the current camera or settings
SPOT_ERRBRIGHTNESSCHANGED	The brightness was apparently changed during exposure computation.
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERAERROR	camera malfunction
SPOT_ERRCOLORFILTERNOTIN	The camera's color filter is not in the "Color" position.
SPOT_ERRCOLORFILTERNOTOUT	The camera's color filter is not in the "B/W' position.
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERREXPTOOLONG	Computed exposure is too long for the camera or is longer than the set maximum.
SPOT_ERREXPTOOSHORT	Computed exposure is too short for the camera or is shorter than the set minimum.
SPOT_ERRFILEOPEN	The bias frame, flatfield, or background image file cannot be opened or read.
SPOT_ERRFLATFLDINCOMPATIBLE	The flatfield file is not compatible with the current camera or settings.
SPOT_ERRINSUF1394ISOCBANDWIDTH	There is insufficient bandwidth available on the1394 bus to transfer image data.
SPOT_ERRINSUF1394ISOCRESOURCES	There are insufficient 1394 resources available on the machine to transfer image data.
SPOT_ERRNO1394ISOCCHANNEL	There are no isochronous channels available on the 1394 bus.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERROUTOFMEMORY	out of memory
SPOT_ERRBKGDIMAGEINCOMPATIBLE	The background image file is not compatible with the current camera or settings
SPOT_ERRVALOUTOFRANGE	A value passed to this function or SpotSetValue is out of range for the operation.

SPOT_RUNNING	The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE)

Remarks

This function acquires images in the same way as SpotGetImage. Refer to the Remarks section for SpotGetImage for information about the image buffer.

If the specified interval is greater than zero, the SpotCam driver may periodically call the application's callback function (if one has been set) with the status SPOT_STATUSSEQIMAGEWAITING to indicate that it is waiting before acquiring the next image in the sequence. The application's callback will be called with the SPOT_STATUSGETIMAGE, SPOT_STATUSSHUTTEROPEN*xxx*, SPOT_STATUSIMAGEREAD*xxx*, and SPOT_STATUSSEQIMAGEREADY statuses for each image acquired. If an application needs to perform a task such as moving the specimen or changing the illumination between acquisitions, it can do so after receiving the SPOT_STATUSIMAGEREAD*xxx* notification for one acquisition and before returning from its callback for the SPOT_STATUSSHUTTEROPEN*xxx* status notification for the next acquisition. The next exposure will not actually begin until the callback returns, unless nIntervalMSec has been set to SPOT_INTERVALSHORTASPOSSIBLE, in which case, the exposure may begin automatically before the callback returns.

If an external trigger is being used in edge trigger mode, and the bUseTriggerOrSetTTLOuputOnEach argument is FALSE, the camera will wait for an external trigger pulse before exposing the first image, and each subsequent acquisition will begin as determined by the specified interval. If bUseTriggerOrSetTTLOuputOnEach is TRUE, the camera will wait for an external trigger pulse before beginning each exposure. In bulb trigger mode, a separate trigger pulse is required for each acquisition.

If TTL output is enabled, and the bUseTriggerOrSetTTLOuputOnEach argument is FALSE, the TTL output will be set to the active state before the beginning of the first exposure and to the inactive state after the last exposure ends. If bUseTriggerOrSetTTLOuputOnEach is TRUE, the TTL output will be set active and inactive separately for each acquisition.

If the ppImageBuffers argument is NULL, the application will need to call SpotRetrieveSequentialImage to retrieve each image when its callback is called with the SPOT_STATUSSEQIMAGEREADY status.

If the bDeferProcessing argument is TRUE, and the value of SPOT_BITDEPTH is 8 or 24, the driver will create temporary image data buffers for all of the images to be acquired. These temporary buffers will contain two bytes per pixel per color, since the image data occupies more than 8 bits per pixel/channel per color until it is processed. Each image's temporary buffer is freed after the image is processed. It may therefore be advisable for applications to pass NULL for ppImageBuffers and retrieve each image via SpotRetrieveSequentialImage, to conserve memory.

This function will not return until all of the requested images have been acquired and processed, the application aborts it, or an error occurs.

For the fastest possible sequential acquisition, bDeferProcessing should be TRUE and nIntervalMSec should be SPOT_INTERVALSHORTASPOSSIBLE.

See Also

SpotGetImage, SpotRetrieveSequentialImage

SpotGetValue

The SpotGetValue function is used to retrieve the values of the various parameters and camera values.

```
int WINAPI SpotGetValue(
        short nParam,
        void *pValue
);
```

Parameters

nParam

Identifies the parameter whose value is to be retrieved. May be one of the following values:

Value	**Meaning**
SPOT_ACQUIREDIMAGESIZE	The width and height, in pixels, of images acquired by SpotGetImage or

	SpotGetSequentialImages.
SPOT_ACQUIREDLIVEIMAGESIZE	The width and height, in pixels, of images acquired by SpotGetLiveImages
SPOT_AUTOEXPOSE	The auto-exposure setting. Specifies whether exposure computation is performed automatically when SpotGetImage, SpotGetSequentialImages, SpotGetFlatfield, SpotGetFlatfield2, or SpotGetBackgroundImage are called.
SPOT_AUTOGAINLIMIT	The maximum gain level to be allowed for computed exposures for non-live mode images.
SPOT_BIASFRMSUBTRACT	The name of the file to be used for bias frame subtraction or NULL if bias frame subtraction is disabled.
SPOT_BINSIZE	The binning size to be used for all modes of image acquisition.
SPOT_BINSIZELIMITS	The minimum and maximum allowable bin size values (superseded by SPOT_BINSIZES).
SPOT_BINSIZES	The allowable bin size values.
SPOT_BITDEPTH	The bit depth to be used for still image capture and exposure computation.
SPOT_BITDEPTHS	The allowable bit depth values for still image capture.
SPOT_BRIGHTNESSADJ	The adjustment factor to be applied to computed exposures to adjust brightness.
SPOT_BRIGHTNESSADJX1000	The adjustment factor to be applied to computed exposures to adjust brightness multiplied by 1000.
SPOT_BRIGHTNESSADJLIMITS	The minimum and maximum allowable brightness adjustment values.
SPOT_BRIGHTNESSADJLIMITSX1000	The minimum and maximum allowable brightness adjustment valuesmultiplied by 1000.
SPOT_BUSBANDWIDTH	The desired maximum 1394 bus bandwidth level.
SPOT_CLEARMODE	The mode to be used for clearing the image sensor.
SPOT_CLEARMODES	The clear modes which supported by the camera.
SPOT_COLORBINSIZE	The color binning size to be used for all modes of image acquisition.
SPOT_COLORBINSIZES	The allowable color binnning size values.
SPOT_COLORENABLE	The filter colors which are enabled for image acquisition and exposure computation.
SPOT_COLORENABLE2	The filter colors which are enabled for image acquisition and exposure computation.
SPOT_COLORORDER	The order in which the colors channels are to be acquired for multi-shot images.
SPOT_INPUTCOLORPROFILE	The name of the file containing the input ICC profile information for color enhancements.
SPOT_OUTPUTCOLORPROFILE	The name of the file containing the output ICC profile information for color enhancements.
SPOT_CORRECTCHIPDEFECTS	Enables or disables chip defect correction for still captured images.
SPOT_DEVICEUID	The unique ID of the current camera device.
SPOT_DRIVERDEVICENUMBER	The zero-based index of the current camera device in the list of found devices.
SPOT_ENABLETTLOUTPUT	Enables or disables the TTL output.
SPOT_ENHANCECOLORS	Enables or disables automatic color enhancement for still captured images.
SPOT_EXPOSURE	The exposure times and gain to be used for acquisition of still captured images.
SPOT_EXPOSURE2	The exposure times and gain to be used for acquisition of still captured images.
SPOT_EXPOSURECOMPRECT	The area of sensor chip to be used for exposure computation (NULL for full chip).
SPOT_EXPOSURECONVFACTOR	The factor to use to convert live mode exposures to still image acquisition exposures.
SPOT_EXPOSURECONVFACTORX1000	The factor to use to convert live mode exposures to still image acquisition exposures multiplied by 1000.
SPOT_EXPOSUREINCREMENT	The exposure increment, in nanoseconds, of exposure values (except where they are explicitly expressed in msec).
SPOT_EXPOSURELIMITS	The minimum and maximum allowable exposure durations in msec.
SPOT_EXPOSURELIMITS2	The minimum and maximum allowable exposure durations.
SPOT_EXPOSURERESOLUTION	The smallest exposure duration increment supported by the camera in nsec.
SPOT_EXTERNALSHUTTERLAG	The wait time, in msec after raising the TTL output before exposing.
SPOT_EXTERNALTRIGGERACTIVESTATE	The level (high or low) of the external trigger input required to trigger acquisition.
SPOT_EXTERNALTRIGGERDELAY	The delay between the trigger signal and the beginning of exposure in µsec.
SPOT_EXTERNALTRIGGERDELAYLIMITS	The minimum and maximum allowable external trigger delay values in µsec.
SPOT_EXTERNALTRIGGERMODE	The external trigger mode.
SPOT_FLATFLDCORRECT	The name of the file to be used for flatfield correction or NULL if flatfield correction is disabled.
SPOT_GAINPORTNUMBER	The currently selected gain port.
SPOT_GAINVALS12	The allowable gain values for 12+ bit per channel still image capture.
SPOT_GAINVALS8	The allowable gain values for 8 bit per channel still image capture.

SPOT_HORIZREADOUTFREQUENCIES	The allowable still image capture sensor horizontal readout frequencies in kHz.
SPOT_HORIZREADOUTFREQUENCY	The frequency at which each line of the sensor is read for still captured images, in kHz.
SPOT_IMAGERECT	The area of the sensor chip to be used for image acquisition.
SPOT_IMAGETYPE	The image type (bright or dark field) used for exposure computation.
SPOT_LIVEACCELERATIONLEVEL	The acceleration level for live image mode.
SPOT_LIVEAUTOBRIGHTNESS	Enables or disables auto brightness adjustments in live mode.
SPOT_LIVEAUTOBRIGHTNESSADJ	The current live mode auto-brightness adjustment factor.
SPOT_LIVEAUTOBRIGHTNESSADJX1000	The current live mode auto-brightness adjustment factor multiplied by 1000.
SPOT_LIVEAUTOGAINLIMIT	The maximum gain level to be allowed for computed exposures for live mode exposure computation.
SPOT_LIVEBRIGHTNESSADJ	The brightness adjustment used for live image acquisition.
SPOT_LIVEBRIGHTNESSADJX1000	The brightness adjustment used for live image acquisition multiplied by 1000
SPOT_LIVEENHANCECOLORS	Enables or disables automatic color enhancement for live mode images.
SPOT_LIVEEXPOSURE	The exposure durations and gain used for live image acquisition.
SPOT_LIVEGAINVALS	The allowable gain values for live image acquisition.
SPOT_LIVEGAMMAADJ	The gamma adjustment applied to live images.
SPOT_LIVEGAMMAADJX1000	The gamma adjustment applied to live images multiplied by 1000.
SPOT_LIVEMAXEXPOSUREMSEC	The maximum allowable exposure for live image acquisition in msec.
SPOT_MAXEXPOSUREMSEC	The maximum allowable exposure still image capture in msec.
SPOT_MAXGAINPORTNUMBER	The maximum gain port number.
SPOT_MAXIMAGERECTSIZE	The maximum allowable image area width and height, which is the size of the camera's image sensor chip.
SPOT_MAXLIVEACCELERATIONLEVEL	The maximum allowable live image acceleration level.
SPOT_MAXPIXELRESOLUTIONLEVEL	The maximum allowable pixel resolution level.
SPOT_MAXWHITEBALANCERATIO	The maximum allowable white balance ratio.
SPOT_MAXWHITEBALANCERATIOX1000	The maximum allowable white balance ratio multiplied by 1000.
SPOT_MESSAGEENABLE	Enables or disables processing of OS messages during camera operation.
SPOT_MINEXPOSUREINCREMENT	The minimum exposure increment which the camera supports.
SPOT_MINEXPOSUREMSEC	The minimum exposure, in msec, to be allowed for computed exposures.
SPOT_MINIMAGERECTSIZE	The minimum allowable image area width and height
SPOT_MONITORFILTERPOS	Enables or disables the monitoring of the color filter position for slider cameras.
SPOT_MOSAICPATTERN	The description of the mosaic color pattern on the camera's sensor chip.
SPOT_NOISEFILTERTHRESPCT	The threshold percentage to be used for noise filtering or 0 to disable noise filtering.
SPOT_PIXELRESOLUTIONLEVEL	The pixel resolution level used for image acquisition.
SPOT_PIXELSIZE	The x and y sizes of the sensor pixels, in nm.
SPOT_PORT0GAINVALS12	The allowable port 0 gain values for 12+ bit per channel still image capture. (same as SPOT_GAINVALS12)
SPOT_PORT0GAINVALS8	The allowable port 0 gain values for 8 bit per channel still image capture. (same as SPOT_GAINVALS8)
SPOT_PORT0LIVEGAINVALS	The allowable port 0 gain values for live image acquisition. (same as SPOT_LIVEGAINVALS)
SPOT_PORT1GAINVALLIMITS	The minimum and maximum allowable port 1 gain values for still image capture.
SPOT_PORT1LIVEGAINVALLIMITS	The minimum and maximum allowable port 1 live mode gain values.
SPOT_PORT2GAINVALLIMITS	The minimum and maximum allowable port 2 gain values for still image capture.
SPOT_PORT2LIVEGAINVALLIMITS	The minimum and maximum allowable port 2 live mode gain values.
SPOT_PORT3GAINVALLIMITS	The minimum and maximum allowable port 3 gain values for still image capture.
SPOT_PORT3LIVEGAINVALLIMITS	The minimum and maximum allowable port 3 live mode gain values.
SPOT_REGULATEDTEMPERATURE	The temperature to which the image sensor is regulated, in tenths of a degree C.
SPOT_REGULATEDTEMPERATURELIMITS	The minimum and maximum allowable regulated temperature values, in tenths of a degree C.
SPOT_REGULATETEMPERATURE	Enables or disables temperature regulation for the image sensor.
SPOT_RETURNRAWMOSAICDATA	Enables or disables the return of raw image data from mosaic cameras.
SPOT_SUBTRACTBLACKLEVEL	Enables or disables automatic black level subtraction for acquisition of 12+ bpp images.
SPOT_BKGDIMAGESUBTRACT	The name of the file to be used for background image subtraction or NULL if background image subtraction is disabled.
SPOT_TTLOUTPUTACTIVESTATE	The level (high or low) of the TTL output signal when active.
SPOT_TTLOUTPUTDELAY	The delay between the activation of the TTL output signal and the beginning of exposure, in µsec.

SPOT_TTLOUTPUTDELAYLIMITS	The minimum and maximum allowable TTL output delay values, in µsec.
SPOT_WAITFORSTATUSCHANGES	Enables use of SpotWaitForStatusChange for status notifications
SPOT_WHITEBALANCE	The red, green, and blue white balance ratios.
SPOT_WHITEBALANCEX1000	The red, green, and blue white balance ratios multiplied by 1000.
SPOT_WHITEBALCOMPRECT	The area of the sensor chip to be used for white balance computation (NULL for full chip).

pValue

Points to the memory where the retrieved value of the specified parameter is to be placed. The data type pointed to depends on the value of nParam. The following data types are used:

Value	**Data Type pointed to**
SPOT_ACQUIREDIMAGESIZE	two short values, first value is width, second is height
SPOT_ACQUIREDLIVEIMAGESIZE	two short values, first value is width, second is height
SPOT_AUTOEXPOSE	BOOL
SPOT_AUTOGAINLIMIT	short
SPOT_BIASFRMSUBTRACT	char (NULL-terminated string)
SPOT_BINSIZE	short
SPOT_BINSIZELIMITS	two short values, first value is minimum, second is maximum
SPOT_BINSIZES	array of short values, first value is count of values following
SPOT_BITDEPTH	short
SPOT_BITDEPTHS	array of short values, first value is count of values following
SPOT_BRIGHTNESSADJ	float
SPOT_BRIGHTNESSADJX1000	long
SPOT_BRIGHTNESSLIMITS	two float values, first value is minimum, second is maximum
SPOT_BRIGHTNESSLIMITSX1000	two long values, first value is minimum, second is maximum (multiplied by 1000)
SPOT_BUSBANDWIDTH	short (SPOT_HIGHBW, SPOT_MEDIUMBW, or SPOT_LOWBW)
SPOT_CLEARMODE	DWORD
SPOT_CLEARMODES	DWORD
SPOT_COLORENABLE	SPOT_COLOR_ENABLE_STRUCT structure
SPOT_COLORENABLE2	SPOT_COLOR_ENABLE_STRUCT2 structure
SPOT_COLORORDER	char (NULL-terminated string (eg. “RBG”, “RG”, etc.))
SPOT_INPUTCOLORPROFILE	char (NULL-terminated string)
SPOT_OUTPUTCOLORPROFILE	char (NULL-terminated string)
SPOT_CORRECTCHIPDEFECTS	BOOL
SPOT_DEVICEUID	SPOT_DEVICE_UID union
SPOT_DRIVERDEVICENUMBER	short
SPOT_ENABLETTLOUTPUT	BOOL
SPOT_ENHANCECOLORS	BOOL
SPOT_EXPOSURE	SPOT_EXPOSURE_STRUCT structure
SPOT_EXPOSURE2	SPOT_EXPOSURE_STRUCT2 structure
SPOT_EXPOSURECOMPRECT	RECT struct
SPOT_EXPOSURECONVFACTOR	float
SPOT_EXPOSURECONVFACTORX1000	long
SPOT_EXPOSUREINCREMENT	long
SPOT_EXPOSURELIMITS	two long values, first value is minimum, second is maximum
SPOT_EXPOSURELIMITS2	two DWORD values, first value is minimum, second is maximum (in increments defined by SPOT_EXPOSUREINCREMENT)
SPOT_EXPOSURERESOLUTION	long
SPOT_EXTERNALSHUTTERLAG	short
SPOT_EXTERNALTRIGGERACTIVESTATE	short (SPOT_TRIGACTIVESTATE_LOW or SPOT_TRIGACTIVESTATE_HIGH)
SPOT_EXTERNALTRIGGERDELAY	long
SPOT_EXTERNALTRIGGERDELAYLIMITS	two long values, first value is minimum, second is maximum (in µsec)
SPOT_EXTERNALTRIGGERMODE	short (SPOT_TRIGMODENONE, SPOT_TRIGMODEEDGE, or SPOT_TRIGMODEBULB)
SPOT_FLATFLDCORRECT	char (NULL-terminated string)
SPOT_GAINPORTNUMBER	short

SPOT_GAINVALS12	array of short values, first value is count of values following
SPOT_GAINVALS8	array of short values, first value is count of values following
SPOT_HORIZREADOUTFREQUENCIES	array of long values, first value is count of values following
SPOT_HORIZREADOUTFREQUENCY	long
SPOT_IMAGERECT	RECT structure
SPOT_IMAGETYPE	short (SPOT_IMAGEBRIGHTFLD or SPOT_IMAGEDARKFLD)
SPOT_LIVEACCELERATIONLEVEL	short
SPOT_LIVEAUTOBRIGHTNESS	BOOL
SPOT_LIVEAUTOBRIGHTNESSADJ	float
SPOT_LIVEAUTOBRIGHTNESSX1000	long
SPOT_LIVEAUTOGAINLIMIT	short
SPOT_LIVEBRIGHTNESSADJ	float
SPOT_LIVEBRIGHTNESSADJX1000	long
SPOT_LIVEENHANCECOLORS	BOOL
SPOT_LIVEEXPOSURE	SPOT_EXPOSURE_STRUCT2 structure
SPOT_LIVEGAINVALS	array of short values, first value is count of values following
SPOT_LIVEGAMMAADJ	float
SPOT_LIVEGAMMAADJX1000	long
SPOT_LIVEMAXEXPOSUREMSEC	long
SPOT_MAXEXPOSUREMSEC	long
SPOT_MAXGAINPORTNUMBER	short
SPOT_MAXIMAGERECTSIZE	array of short values, first value is maximum width, second is maximum height
SPOT_MAXLIVEACCELERATIONLEVEL	short
SPOT_MAXPIXELRESOLUTIONLEVEL	short
SPOT_MAXWHITEBALANCERATIO	float
SPOT_MAXWHITEBALANCERATIOX1000	long
SPOT_MESSAGEENABLE	BOOL
SPOT_MINEXPOSUREINCREMENT	long
SPOT_MINEXPOSUREMSEC	short
SPOT_MINIMAGERECTSIZE	array of short values, first value is minimum width, second is minimum height
SPOT_MONITORFILTERPOS	BOOL
SPOT_MOSAICPATTERN	short (one of the SPOT_MOSAIC*xxx* values)
SPOT_NOISEFILTERTHRESPCT	short
SPOT_PIXELRESOLUTIONLEVEL	short
SPOT_PIXELSIZE	array of two longs, first value is x size, second is y size
SPOT_PORT0GAINVALS12	array of short values, first value is count of values following
SPOT_PORT0GAINVALS8	array of short values, first value is count of values following
SPOT_PORT0LIVEGAINVALS	array of short values, first value is count of values following
SPOT_PORT1GAINVALLIMITS	two long values, first value is minimum port 1 gain, second is maximum
SPOT_PORT1LIVEGAINVALLIMITS	two long values, first value is minimum port 1 live mode gain, second is maximum
SPOT_PORT2GAINVALLIMITS	two long values, first value is minimum port 2 gain, second is maximum
SPOT_PORT2LIVEGAINVALLIMITS	two long values, first value is minimum port 2 live mode gain, second is maximum
SPOT_PORT3GAINVALLIMITS	two long values, first value is minimum port 3 gain, second is maximum
SPOT_PORT3LIVEGAINVALLIMITS	two long values, first value is minimum port 3 live mode gain, second is maximum
SPOT_REGULATEDTEMPERATURE	short
SPOT_REGULATEDTEMPERATURELIMITS	two short values, first value is minimum, second is maximum
SPOT_REGULATETEMPERATURE	BOOL
SPOT_RETURNRAWMOSAICDATA	BOOL
SPOT_SUBTRACTBLACKLEVEL	BOOL
SPOT_BKGDIMAGESUBTRACT	char (NULL-terminated string)
SPOT_TTLOUTPUTACTIVESTATE	short
SPOT_TTLOUTPUTDELAY	long
SPOT_TTLOUTPUTDELAYLIMITS	two long values, first value is minimum, second is maximum
SPOT_WAITFORSTATUSCHANGES	BOOL
SPOT_WHITEBALANCE	SPOT_WHITE_BAL_STRUCT structure
SPOT_WHITEBALANCEX1000	SPOT_WHITE_BAL_INT_STRUCT structure
SPOT_WHITEBALCOMPRECT	RECT structure

Return Values

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRINVALIDPARAM	invalid value for nParam
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.

Remarks

An application can call SpotGetValueSize prior to calling this function to determine required size of the buffer pointed to by pValue.

Refer to the table in the appendix regarding the applicability of each of the parameters to the various camera models.

See Also

SpotGetValueSize, SpotSetValue

SpotGetValueSize

The SpotGetValueSize function can be used to determine the size of the buffer which needs to be passed to SpotGetValue.

```
int WINAPI SpotGetValueSize(
        short nParam
);
```

Parameters

nParam

Identifies the parameter whose value is to be retrieved. May be any of the value which can be passed to SpotGetValue.

Return Values

If the function succeeds, the return value will be the maximum size, in bytes, of the data which could be retrieved by SpotGetValue for the specified parameter and the current camera. If an invalid parameter is specified, or the size is unknown because the camera is not initialized, the return value will be zero.

Remarks

An application can call this function prior to calling SpotGetValue to determine how much memory to allocate for the data buffer whose address is passed as the pValue argument. The application should call SpotInit prior to calling this function for all camera-specific parameters.

See Also

SpotGetValue

SpotGetVersionInfo

The SpotGetVersionInfo function is used to query the camera driver for version and camera information. It has been superseded by SpotGetVersionInfo2 which provides more detailed camera information.

```
void WINAPI SpotGetVersionInfo(
        SPOT_VERSION_STRUCT *pstVerInfo
);
```

Parameters

pstVerInfo

Points to a SPOT_VERSION_STRUCT structure to hold the retrieved information.

See Also

SpotGetVersionInfo2

SpotGetVersionInfo2

The SpotGetVersionInfo2 function is used to query the camera driver for version and camera information.

```
void WINAPI SpotGetVersionInfo2(
        SPOT_VERSION_STRUCT2 *pstVerInfo
);
```

Parameters

pstVerInfo

Points to a SPOT_VERSION_STRUCT2 structure to hold the retrieved information.

SpotInit

The SpotInit function initializes the SPOT camera driver.

```
int WINAPI SpotInit();
```

Return Value

If the function succeeds, the return value is SPOT_SUCCESS or SPOT_WARNUNSUPPCAMFEATURES. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVALREADYINIT	the driver is already initialized
SPOT_ERRCAMANDCARDINCOMPATIBLE	the camera and interface card to which it is connected are incompatible with each other
SPOT_ERRCAMERABUSY	The camera is currently performing another operation.
SPOT_ERRCAMERANOTSUPPORTED	the camera model is not supported by this version of the driver
SPOT_ERRDEVDRVLOAD	There was an error loading or opening the device driver.
SPOT_ERRDMASETUP	there was an error setting up the DMA buffer (SPOT RT/Insight only)
SPOT_ERRDRVALREADYINIT	driver has already been initialized
SPOT_ERRNOCAMERAINFOFILE	A *Dnnn*.cif (SPOT RT-SE) file could not be found for the camera.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERRNODEVICESFOUND	no camera devices were found
SPOT_ERROUTOFMEMORY	out of memory
SPOT_ERRREADCAMINFO	there was an error reading camera information (SPOT RT/Insight only)
SPOT_ERRREGISTRYQUERY	error reading from the Windows registry
SPOT_ERRVXDOPEN	VxD loader error (Win95/98/Me only)

Remarks

This function must be called before calling any other SpotCam API functions unless otherwise noted. It should only be called once until the SpotExit function is called. If there is a chance that more than one SPOT family camera may be installed on the computer, the caller should specify which camera to use by setting the value for SPOT_DEVICEUID or SPOT_DRIVERDEVICENUMBER prior to calling this function. SpotFindDevices can be called to obtain a list of available cameras. If no device specification is made, the first camera found by the driver will be initialized.

If the camera initialization succeeds, but the SpotCam driver determines that the current camera provides functionality that it does not support, the function will return SPOT_WARNUNSUPPCAMFEATURES. All camera features which are supported by the driver will function normally. This warning will generally indicate that the SpotCam driver software should be updated to take full advantage of the camera's features.

See Also

SpotExit

SpotOpenExternalShutter

The SpotOpenExternalShutter function is used to explicitly set the TTL output signal to the active state.

```
int WINAPI SpotOpenExternalShutter();
```

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.

Remarks

Refer to the section on SpotCloseExternalShutter. SpotSetTLLOutputState can be called instead of this function.

See Also

SpotSetTTLOutputState, SpotCloseExternalShutter

SpotQueryCameraPresent

The SpotQueryCameraPresent function can be used to determine whether or not a camera is currently connected and powered.

```
int WINAPI SpotQueryCameraPresent(
        BOOL *pbCameraPresent
);
```

Parameters

pbCameraPresent

Points to a BOOL value which indicates whether the camera is present and powered or not

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRCAMERABUSY	camera is currently performing another operation
SPOT_ERRDRVNOTINIT	The driver is not initialized.

Remarks

SpotInit must be called prior to calling this function. This function will return FALSE if the camera power is off.

SpotQueryColorFilterPosition

The SpotQueryColorFilterPosition function can be used to determine the current color filter position for some slider cameras.

```
int WINAPI SpotQueryColorFilterPosition(
        BOOL *pbFilterIn,
        BOOL *pbFilterOut
);
```

Parameters

pbFilterIn

Points to a BOOL value which indicates whether or not the color filter is in the “Color” position

pbFilterOut

Points to a BOOL value which indicates whether or not the color filter is in the “B/W” position

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	Meaning
SPOT_ERRCAMERABUSY	camera is currently performing another operation

SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOCAMERARESP	The camera is not responding.
SPOT_ERRNOTCAPABLE	camera is not an RT slider

Remarks

SpotInit must be called prior to calling this function. An application can check the SPOT_ATTR_SLIDERPOSITIONDETECTION bit returned by SpotGetCameraAttributes to determine if the current camera is capable of detecting its filter slider position. If the color filter is not properly locked in either the "Color" or "B/W" position, both returned values will be FALSE.

SpotQueryStatus

The SpotQueryStatus function can be used to query the driver for current status information and/or abort camera operation.

```
int WINAPI SpotQueryStatus(
        BOOL bAbort,
        long *plInfo
);
```

Parameters

bAbort

Flag to abort the current camera operation (aborts if TRUE).

plInfo

Points to a long value where the driver will place information associated with the status.

Return Value

The function will return one of the SPOT_STATUS*xxx* values defined in SpotCam.h.

Remarks

This function provides a way to obtain driver status information without setting a callback function. See the SpotCam.h file for information about status values.

If bAbort is TRUE, this function will return the status that the driver had prior to the current operation being aborted, which will not necessarily be SPOT_STATUSABORTED.

See Also

SpotClearStatus, SpotSetAbortFlag, SpotSetCallback

SpotRetrieveSequentialImage

The SpotRetrieveSequentialImage function is used by the application to retrieve an image acquired by a call to SpotGetSequentialImages.

```
int WINAPI SpotRetrieveSequentialImage(
        void *pImageBuffer
);
```

Parameters

pImageBuffer

Points to a buffer where the pixel data of the retrieved image is to be placed by the driver. Refer to the section on SpotGetImage for a description of the image buffer.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	**Meaning**
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOIMAGEAVAILABLE	there is no sequential image to retrieve
SPOT_ERROUTOFMEMORY	out of memory

Remarks

This function is used in conjunction with SpotGetSequentialImages when it is called with NULL for the ppImageBuffers argument. As each sequential image becomes available, the application will be notified by a call to its callback function with a status value of SPOT_STATUSSEQIMAGEREADY. The application should normally call SpotRetrieveSequentialImage before returning from its callback function. If the value of the bDeferProcessing argument passed to SpotGetSequentialImages is TRUE, the application will be notified of the availability of images only after all of the sequential images have been acquired, and this function can be called even after SpotGetSequentialImages returns.

See Also

SpotGetImage, SpotGetSequentialImages, SpotSetCallback

SpotSetAbortFlag

The SpotSetAbortFlag function is used by the application to set a pointer to an abort flag. The abort flag is periodically checked by the camera driver during camera operation, and if it is found to be TRUE, the current process is aborted.

```
void WINAPI SpotSetAbortFlag(
        BOOL *pbAbort
);
```

Parameters

pbAbort

Points to a BOOL value which the driver will periodically check. When the value is set to TRUE the current camera operation, if any, will abort.

Remarks

Setting the value of the abort flag to TRUE will cause all camera operations to abort. The application must reset the abort flag before resuming camera operation. The memory where the abort flag resides must persist until SpotExit is called.

The application can also abort camera operations by calling SpotQueryStatus.

See Also

SpotQueryStatus

SpotSetCallback

The SpotSetCallback function is used by the application to set a pointer to a callback function which the driver will call periodically during camera operation.

```
void WINAPI SpotSetCallback(
        SPOTCALLBACK pfnCallback,
        DWORD dwUserData
);
```

Parameters

pfnCallback

Points to a function of type SPOTCALLBACK, which the driver will call periodically during camera operations.

dwUserData

An application-defined value, which will be passed to the callback function by the driver, each time the callback is called.

Remarks

The callback function provides the application with feedback from the camera driver as to what the driver is currently doing and an opportunity to perform certain tasks. The callback function is called periodically during camera operations. Each time the callback function is called, the current driver status is passed to it along with associated information. Refer to the SpotCam.h file for information regarding the status values. In most cases, the current camera operation will be suspended until the callback returns. If the callback does not return quickly, performance may be negatively impacted, and in certain situations, image data may be corrupted.

See Also

SpotClearStatus, SpotQueryStatus

SpotSetDeviceNotificationCallback

The SpotSetDeviceNotificationCallback function is used by the application to set a pointer to a callback function which the driver will call when cameras are added to or removed from computer.

```
void WINAPI SpotSetDevice NotificationCallback(
        SPOTDEVNOTIFYCALLBACK pfnCallback,
        DWORD dwUserData
);
```

Parameters

pfnCallback

Points to a function of type SPOTDEVNOTIFYCALLBACK, which the driver will call when a camera is added to or removed from computer.

dwUserData

An application-defined value, which will be passed to the callback function by the driver, each time the callback is called.

Remarks

Device notifications are only supported for 1394 cameras and some newer PCI card cameras.

SpotSetTTLOutputState

The SpotSetTTLOutputState function is used to explicitly set the state of the TTL output signal.

```
int WINAPI SpotSetTTLOutputState(
        BOOL bSetActive
);
```

Parameters

bSetActive

Specifies whether the TTL output should be set to the active or inactive state.

Return Value

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	**Meaning**
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.

Remarks

SpotSetTTLOutputState can be used by the application to exercise more direct control over the TTL output signal. If this function is to be called, it may be advisable to disable the automatic control of the external shutter via SpotSetValue and the SPOT_EXTERNALSHUTTERENABLE parameter. The active state of the TTL output is defined by the SPOT_TTLOUTPUTACTIVESTATE parameter.

See Also

SpotOpenExternalShutter, SpotCloseExternalShutter

SpotSetValue

The SpotSetValue function is used to set the various parameters which control camera operation.

```
int WINAPI SpotSetValue(
        short nParam,
        void *pValue
);
```

Parameters

nParam

Identifies the parameter whose value is to be set. May be one of the following values:

SPOT_AUTOEXPOSE
SPOT_AUTOGAINLIMIT
SPOT_BIASFRMSUBTRACT
SPOT_BINSIZE
SPOT_BITDEPTH
SPOT_BUSBANDWIDTH
SPOT_CLEARMODE
SPOT_COLORBINSIZE
SPOT_COLORENABLE
SPOT_COLORENABLE2
SPOT_COLORORDER
SPOT_INPUTCOLORPROFILE
SPOT_OUTPUTCOLORPROFILE
SPOT_CORRECTCHIPDEFECTS
SPOT_DEVICEUID
SPOT_DRIVERDEVICENUMBER
SPOT_ENHANCECOLORS
SPOT_EXPOSURE
SPOT_EXPOSURE2
SPOT_EXPOSUREADJ
SPOT_EXPOSUREADJX1000
SPOT_EXPOSURECOMPRECT
SPOT_EXPOSUREINCREMENT
SPOT_EXTERNALSHUTTERENABLE
SPOT_EXTERNALSHUTTERLAG
SPOT_EXTERNALTRIGGERACTIVESTATE
SPOT_EXTERNALTRIGGERDELAY
SPOT_EXTERNALTRIGGERMODE
SPOT_FLATFLDCORRECT
SPOT_FLUORESCENCECOLORS
SPOT_GAINPORTNUMBER
SPOT_IMAGERECT
SPOT_IMAGETYPE
SPOT_HORIZREADOUTFREQUENCY
SPOT_HORIZREGULATEDTEMPERATURE
SPOT_LIVEACCELERATIONLEVEL
SPOT_LIVEAUTOBRIGHTNESS
SPOT_LIVEAUTOBRIGHTNESSADJ
SPOT_LIVEAUTOBRIGHTNESSADJX1000
SPOT_LIVEAUTOGAINLIMIT
SPOT_LIVEENHANCECOLORS
SPOT_LIVEEXPOSURE
SPOT_LIVEGAINADJ
SPOT_LIVEGAINADJX1000
SPOT_LIVEGAMMAADJ
SPOT_LIVEGAMMAADJX1000
SPOT_LIVEMAXEXPOSUREMSEC
SPOT_MAXEXPOSUREMSEC
SPOT_MESSAGEENABLE
SPOT_MINEXPOSUREMSEC

SPOT_MONITORFILTERPOS
SPOT_NOISEFILTERTHRESPCT
SPOT_PIXELRESOLUTIONLEVE
SPOT_REGULATETEMPERATURE
SPOT_RETURNRAWMOSAICDATA
SPOT_SUBTRACTBLACKLEVEL
SPOT_REGULATETEMPERATURE
SPOT_BKGDIMAGESUBTRACT
SPOT_TTLOUTPUTACTIVESTATE
SPOT_TTLOUTPUTDELAY
SPOT_TTLOUTPUTENABLE
SPOT_WAITFORSTATUSCHANGES
SPOT_WHITEBALANCE
SPOT_WHITEBALANCEX1000
SPOT_WHITEBALCOMPRECT

Refer to the section on SpotGetValue for descriptions of these parameters.

pValue

Points to the value(s) to be set for the specified parameter. The data type pointed to depends on the value of nParam. See the section on SpotGetValue for the data types used.

Return Values

If the function succeeds, the return value is SPOT_SUCCESS. If the function fails, one of the following values is returned:

Value	**Meaning**
SPOT_ERRDRVNOTINIT	The driver is not initialized.
SPOT_ERRINVALIDPARAM	The value for nParam is invalid.
SPOT_ERRNOTCAPABLE	The camera model is not capable of performing the operation.
SPOT_ERRVALOUTOFRANGE	The value is out of range.

Remarks

Refer to the section on SpotGetValue for the descriptions and data types for the various parameters and to the table in the appendix regarding the applicability of each of the parameters to the various camera models.

See Also

SpotGetValue

SpotWaitForStatusChange

The SpotWaitForStatusChange function is used to wait for a status notification from the SpotCam driver.

```
BOOL WINAPI SpotWaitForStatusChange(
        int *pnStatus,
        long *plInfo,
        int nTimeoutMSec
);
```

Parameters

pnStatus

Points to an int where the driver will place the current status value.

lInfo

Points to a long value where the driver will place information associated with the status.

nTimeoutMSec

The timeout period (in msec) after which the function will return if the status doesn't change. A value of -1 indicates an infinite wait.

Return Values

The function will return TRUE if a status change occurs within the specified timeout period, FALSE otherwise.

Remarks

This function can only be called if the SPOT_WAITFORSTATUSCHANGES parameter has been set to TRUE and the last SpotCam API call returned SPOT_RUNNING. It is used to block application execution until the current operation has completed and to provide important status notifications during the operation. Refer to the Status Notifications section at the top of this document for more information.

This function should always be called from the same thread as the API function for which it is waiting. If these functions are called by the application's main thread, a positive timeout value should be specified so that the application can process OS events. If OS events are being handled by another thread, a value of -1 may be specified for the timeout, and the function will not return until a status change occurs. If zero is specified for the timeout, the function will return immediately, indicating whether or not a status change occurred since the last call.

Appendix A – Applicability of parameters set with SpotSetValue

The following table lists all of the parameters which can be set with SpotSetValue and their applicability to exposure computation, still image capture, and live image mode.

Parameter	Exposure Computation	Still Image Capture	Live Image Mode
SPOT_AUTOEXPOSE		X	
SPOT_AUTOGAINLIMIT	X	1	
SPOT_BIASFRMSUBTRACT		X	X
SPOT_BINSIZE	X	X	X
SPOT_BITDEPTH	X	X	
SPOT_BRIGHTNESSADJ	X	1	
SPOT_BRIGHTNESSADJX1000	X	1	
SPOT_CLEARMODE		X	
SPOT_COLORBINSIZE	X	X	X
SPOT_COLORENABLE	X	X	X
SPOT_COLORENABLE2	X	X	X
SPOT_COLORORDER		X	
SPOT_CORRECTCHIPDEFECTS		X	
SPOT_INPUTCOLORPROFILE		X	X
SPOT_OUTPUTCOLORPROFILE		X	X
SPOT_ENHANCECOLORS		X	
SPOT_EXPOSURE		X	
SPOT_EXPOSURE2		X	
SPOT_EXPOSURECOMPRECT	X	1	1
SPOT_EXTERNALTRIGGERACTIVESTATE		X	
SPOT_EXTERNALTRIGGERDELAY		X	
SPOT_EXTERNALTRIGGERMODE		X	
SPOT_FLATFLDCORRECT		X	
SPOT_FLUORESCENCECOLORS		2	
SPOT_GAINPORTNUMBER	X	X	X
SPOT_IMAGERECT	2	X	X
SPOT_IMAGETYPE	X	1	1
SPOT_LIVEACCELERATIONLEVEL			X
SPOT_LIVEAUTOBRIGHTNESS			1
SPOT_LIVEAUTOGAINLIMIT			1
SPOT_LIVEBRIGHTNESSADJ			1
SPOT_LIVEBRIGHTNESSADJX1000			1
SPOT_LIVEENHANCECOLORS			X
SPOT_LIVEEXPOSURE			3
SPOT_LIVEGAMMAADJ			X
SPOT_LIVEGAMMAADJX1000			X
SPOT_LIVEMAXEXPOSUREMSEC			1
SPOT_MAXEXPOSUREMSEC	X	1	
SPOT_MINEXPOSUREMSEC	X	1	
SPOT_MONITORFILTERPOS	X	X	
SPOT_NOISEFILTERTHRESPCT		X	
SPOT_PIXELRESOLUTIONLEVEL		X	
SPOT_RETURNRAWMOSAICDATA		X	
SPOT_SUBTRACTBLACKLEVEL		X	
SPOT_BKGDIMAGESUBTRACT		X	
SPOT_TTLOUTPUTACTIVESTATE	X	X	X
SPOT_TTLOUTPUTDELAY	X	X	X
SPOT_TTLOUTPUTENABLE	X	X	X
SPOT_WHITEBALANCE	4	X	X
SPOT_WHITEBALANCEX1000	4	X	X

Notes:

- X - parameter is applicable
- 1 - applicable only when auto-exposure enabled
- 2 - The area specified for SPOT_IMAGERECT is used for exposure computation unless a value is set for SPOT_EXPOSURECOMPRECT
- 3 - applicable only when auto-exposure is disabled
- 4 - applicable only to cameras with color filters

Appendix B – Avoiding Problems

The following issues should be considered to avoid problems:

1. The pValue argument to SpotGetValue and SpotSetValue must contain the address of data of the correct type for the specified parameter. For Boolean parameters, the BOOL (not bool) data type is used.
2. The abort flag whose address is passed to SpotSetAbortFlag must remain valid until SpotSetAbortFlag is called again to change the abort flag or SpotExit is called.
3. Unless an application is guaranteed that only one SPOT family camera will be present on the machine, it should set SPOT_DEVICEUID or SPOT_DRIVERDEVICENUMBER to specify which camera is to be used. If the UID of the desired camera is not already known, SpotFindDevices should be called first to enumerate the SPOT cameras present.
4. Applications should return from their callbacks as quickly as possible except in specific cases where the callback is being used to synchronize the beginning of exposure. In other cases, callbacks which do not return quickly may cause degraded performance and/or image data corruption.
5. It is imperative that the image buffers provided by the application are the appropriate size. If an application supplies a buffer which is too small, and application or system crash may result. The SPOT_ACQUIREDIMAGESIZE and SPOT_ACQUIREDLIVEIMAGESIZE parameters should be queried to determine the width and height of the images to be acquired, and the set bit depth needs to be considered. For 8 and 24 bpp images, each image line must contain a number of bytes which is integer-divisible by four, so padding bytes may need appended. For live mode, if image rotation is specified, the image buffer must be sized accordingly.
6. When acquiring color images from cameras with color filters, only the color channels which correspond to the colors which were enabled will be filled with data.
7. Before calling any SpotCam API functions from a thread other than the thread running the application's main message loop, it is advisable to set SPOT_MESSAGEENABLE to FALSE. This will help prevent the SpotCam driver from allowing the calling thread to handle OS messages.
8. Note that for many camera models, more gain levels are allowed for 8 bit per channel acquisition than for higher bit depths. Applications should always check both SPOT_GAINVALS8 and SPOT_GAINVALS12 to determine which gain values are allowed at the different bit depths.
9. For cameras which provide gain levels less than 1.0 (as returned by SpotGetActualGainValue or SpotGetActualGainValueX1000), the gain should not be set below 1.0 unless 2x2 or greater binning is being used. If binning is not used, the acquired images will saturate below full-scale.
10. Make sure that callback functions are defined as type VOID WINAPI.

Appendix C – Revision History

Version 4.5

1. Support for SPOT and SPOT Enhanced cameras has been dropped.
2. Support for SPOT Flex, Pursuit, and Xplorer cameras has been added.
3. The requirement for camera information files for SPOT RT-SE cameras has been eliminated.
4. New attributes flags: SPOT_ATTR_ACCURATETTLDELAYTIMING, SPOT_ATTR_AUTOEXPOSURE, SPOT_ATTR_COLORFILTER, SPOT_ATTR_DUALAMPLIFIER, SPOT_ATTR_FILTERWHEEL,SPOT_ATTR_TTLOUTPUT, SPOT_ATTR_SENSORSHIFTING, SPOT_ATTR_SLIDERPOSITIONDETECTION, SPOT_ATTR_TEMPERATUREREADOUT, SPOT_ATTR_TEMPERATUREREGULATION, and SPOT_ATTR_TRIGGERACTIVESTATE have been defined for use with SpotGetCameraAttributes.
5. New return values: SPOT_WARNUNSUPPCAMFEATURES, SPOT_WARNINVALIDINPUTICC, and SPOT_WARNINVALIDOUTPUTICC have been defined for SpotInit.
6. New error codes: SPOT_ERRBIASFRMINCOMPATIBLE, SPOT_ERRBKGDTOOBRIGHT, SPOT_ERRINVALIDFILE, and SPOT_ERRBKGDIMAGEINCOMPATIBLE have been defined.
7. A new option: SPOT_INTERVALSHORTASPOSSIBLE for the nIntervalMSec argument for SpotGetSequentialImages has been defined.
8. New parameters: SPOT_ACQUIREDIMAGESIZE, SPOT_ACQUIREDLIVEIMAGESIZE, SPOT_BIASFRMSUBTRACT, SPOT_BINSIZES, SPOT_CLEARMODE, SPOT_CLEARMODES, SPOT_COLORBINSIZE, SPOT_COLORBINSIZES, SPOT_EXPOSURERESOLUTION, SPOT_EXTERNALTRIGGERACTIVESTATE, SPOT_EXTERNALTRIGGERDELAY, SPOT_EXTERNALTRIGGERDELAYLIMITS, SPOT_GAINPORTNUMBER, SPOT_HORIZREADOUTFREQUENCIES, SPOT_HORIZREADOUTFREQUENCY, SPOT_INPUTCOLORPROFILE, SPOT_LIVEAUTOBRIGHTNESSADJ, SPOT_LIVEAUTOBRIGHTNESSADJX1000SPOT_LIVEENHANCECOLORS, SPOT_MAXGAINPORTNUMBER, SPOT_MAXPIXELRESOLUTIONLEVEL, SPOT_MINIMAGERECTSIZE, SPOT_OUTPUTCOLORPROFILE, SPOT_PIXELRESOLUTIONLEVEL, SPOT_PIXELSIZE, SPOT_PORT0GAINVALS12, SPOT_PORT0GAINVALS8, SPOT_PORT0LIVEGAINVALS, SPOT_PORT1GAINVALLIMITS, SPOT_PORT1LIVEGAINVALLIMITS, SPOT_PORT2GAINVALLIMITS,

SPOT_PORT2LIVEGAINVALLIMITS, SPOT_PORT3GAINVALLIMITS, SPOT_PORT3LIVEGAINVALLIMITS, SPOT_REGULATEDTEMPERATURE, SPOT_REGULATEDTEMPERATURELIMITS, SPOT_REGULATETEMPERATURE, SPOT_BKGDIMAGESUBTRACT, SPOT_TTLOUTPUTACTIVESTATE, SPOT_TTLOUTPUTDELAY, and SPOT_TTLOUTPUTDELAYLIMITS have been defined.

9. The SPOT_STATUSGETIMAGE status notification has been modified to include the number of exposures to be done as the lInfo value.
10. New status values: SPOT_STATUSEXTERNALTRIGGERDELAY, SPOT_STATUSTTLOUTPUTDELAY, SPOT_STATUSWAITINGFORCOLORFILTER, and SPOT_STATUSWAITINGFORMOVETOBKGD have been defined.
11. The SPOT_VERSION_STRUCT2 has been modified to include additional information.
12. New API functions: SpotDumpCameraMemory, SpotGetActualGainValue, SpotGetActualGainValueX10000, SpotGetActualLiveGainValue, SpotGetActualLiveGainValueX10000, SpotGetBiasFrame, SpotGetBiasFrameCompatibilityInformation, SpotGetExposureTimestamp, SpotGetFlatfieldCompatibilityInformation, SpotGetSensorCurrentTemperature, SpotGetSensorExposureTemperature, SpotGetBackgroundImage, SpotGetBackgroundImageCompatibilityInformation, and SpotSetTTLOutputState have been added.
13. The behavior of SpotGetSequentialImages has been modified slightly to allow for more flexibility in retrieving images.

Version 4.0.9

1. A bug which caused light columns to appear on images acquired with SPOT and SPOT Enhanced cameras when defect corrections were enabled has been fixed.
2. Some improvements have been made to the exposure computation routines to fix the problem of SPOT_ERREXPTOOLONG errors when computing exposure for bright images.
3. A work-around for a Windows bug which caused the loading of 1394 support DLLs to fail under certain circumstances was added.
4. A couple of bugs which resulted in crashes or black images when SpotGetSequentialImages was called for mosaic cameras under certain circumstances were fixed.

Version 4.0.8

1. A bug which affected column defect corrections for mosaic cameras under certain circumstances has been fixed.
2. The flatfield correction functionality has been improved.
3. A fix for a problem which occurred with certain 1394 adapter cards has been added.
4. Exposure computation has been improved.

Version 4.0.5

1. A bug which caused a crash when applying flatfield corrections to 36 bpp images acquired from Insight 4 megapixel mosaic cameras has been fixed.
2. A fix for a problem which caused corruption of images from 1394/FireWire cameras on some notebook PCs has been added.

Version 4.0.3

1. Support has been added for the new SPOT Insight 4 megapixel and the latest model of RT-KE cameras.
2. A bug which caused the abort flag to be ignored at times in SpotGetSequentialImages has been fixed.
3. A problem which occurred when SpotFindDevices or SpotFindInterfaceCards was called after calling SpotInit for 1394 cameras has been corrected.

Version 4.0.2

1. Support has been added for the SPOT RT-SE mosaic camera.
2. A problem with white balance with three-shot color cameras and short exposures has been corrected.
3. A problem which caused flickering in three-shot color live mode images when SPOT_LIVEAUTOBRIGHTNESS was enabled has been corrected.
4. A timing problem which occasionally caused SpotGetImage to fail or to acquire a corrupted image when called immediately after the return of SpotGetLiveImages for RT-SE18 cameras has been fixed.
5. A bug which caused SpotGetFlatfield to fail for 1394/FireWire cameras under certain conditions has been fixed.

Version 4.0.1.3

1. The bug which prevented color enhancement from being done for color images from Insight mosaic cameras has been fixed.
2. The bug which caused crashes when acquiring 12 bit per channel images with SpotGetImage from Insight and RT-SE18 cameras has been fixed.
3. The bug which caused crashes when applying flatfield correction to 12bpp images with SPOT_SUBTRACTBLACKLEVEL disabled has been

fixed.
4. The bug which limited readout speed for 1394/FireWire monochrome and color filter cameras with binning enabled has been fixed.

Version 4.0.1

1. The bug affecting flatfield correction has been fixed.
2. Several problems with RT-SE18 camera support have been fixed including black level subtraction and SPOT_ERRNOCAMERARESP errors when changing exposure while running live mode.
3. The duplicate SPOT_STATUSIMAGEREAD callback call which was being made for 1394/FireWire cameras has been eliminated.
4. SpotSetValue now ignores the red, green, and blue exposure values for monochrome and mosaic cameras.
5. SpotSetValue now returns SPOT_ERRVALOUTOFRANGE if the value for SPOT_LIVEGAMMAADJ is less than .001 or SPOT_LIVEGAMMAADJX1000 is less than 1.
6. The bug which caused a failure in finding the ChipInfo.dat file for SPOT and SPOT Enhanced cameras has been fixed.

Version 4.0

1. Support has been added for the new Insight 1394/FireWire cameras.
2. A new attribute flag: SPOT_ATTR_1394has been added for use with SpotGetCameraAttributes to identify 1394 cameras.
3. A new parameter: SPOT_DEVICEUID, has been added for use with SpotGetValue and SpotSetValue to allow an application to better specify which camera to use if more than one are installed.
4. A new parameter: SPOT_BUSBANDWIDTH, has been added for use with SpotGetValue and SpotSetValue to allow an application to specify the amount of 1394 bus bandwidth which should be used.
5. A new function: SpotSetDeviceNotificationCallback has been added to allow an application to set a callback to receive notifications when 1394 cameras are added or removed.
6. New functions: SpotComputeExposureConversionFactor and SpotComputeExposureConversionFactorX1000 have been added to allow an application to compute the conversion factor to be applied to live mode exposures to capture images with the same brightness level as live mode.
7. A new parameter: SPOT_EXPOSURECOMPRECT has been added to allow an application to set an area of the sensor chip to be used for exposure computation.
8. New error code values: SPOT_ERRCAMANDCARDINCOMPATIBLE, SPOT_ERRINSUF1394ISOCBANDWIDTH, SPOT_ERRINSUF1394ISOCRESOURCES, and SPOT_ERRNO1394ISOCCHANNEL have been added.
9. A new function: SpotGetFlatfield2 has been added to acquire better flatfields.
10. A new status value: SPOT_STATUSWAITINGFORBLOCKLIGHT has been added to support the new flatfield acquisition feature.
11. A new function: SpotGetValueSize has been added to allow applications to query for the maximum size of the data which SpotGetValue may return.

Version 3.5.9.1

1. Support has been added for the SPOT Enhanced color slider cameras.

Version 3.5.9

1. Support has been added for the new SPOT RT-SE18 cameras.
2. Support has been added for RT cameras with new KAI-2093M CCDs.
3. A new parameter: SPOT_MOSAICPATTERN, has been added to allow an application to determine the type of mosaic pattern used on a a mosaic camera's CCD.

Version 3.5.8

1. Some changes have been made to better support the latest RT and Insight cameras. As a result of these changes, the number of available gain levels has been reduced in some modes.
2. The bug which caused some functions to incorrectly return SPOT_ERRCAMERABUSY has been fixed.
3. The bug which caused incorrect gain values to be returned for the SPOT_GAINVALS12 parameter for SPOT, SPOT2, and SPOT Enhanced cameras has been fixed.
4. The problem which occasionally affected live images from Insight cameras with a small image area near the top of the CCD has been fixed.

Version 3.5.7.1

1. The bug which caused SpotGetValue to occasionally return an incorrect value for SPOT_EXPOSURECONVFACTORX1000 has been fixed.
2. The bug which cause SpotGetValue to return incorrect values for SPOT_LIVEEXPOSURE has been fixed.

3. The problem which resulted in incorrect coloration in some images from 3-shot RT and Insight cameras has been fixed.

Version 3.5.7

1. Support has been added for the new SPOT Enhanced ME and SPOT Enhanced ME Junior monochrome cameras.
2. A new error value: SPOT_ERRBRIGHTNESSCHANGED has been added to indicate that the brightness level has apparently changed during exposure computation.
3. A memory leak in SpotGetFlatfield has been fixed.
4. A new parameter: SPOT_MAXEXPOSUREMSEC has been added to allow an application to set an upper limit for exposure computation.
5. A new parameter: SPOT_WHITEBALCOMPRECT has been added to allow an application to specify a chip area for computation of white balance.
6. A new function: SpotFindDevices has been added to provide a more powerful and flexible way for applications to determine which devices are available. This function is intended as a replacement for SpotFindInterfaceCards.
7. The bug which caused SpotFIndInterfaceCards to miss ISA cards under certain circumstances has been fixed.

Version 3.5.6.1

1. SpotGetImage now handles the case where nbpp==12 and SPOT_RETURNRAWMOSAICDATA is TRUE correctly.
2. The bugs which caused incorrect values to be returned when calling SpotGetValue with SPOT_EXPOSURECONVFACTOR , SPOT_EXPOSURECONVFACTORX1000, and SPOT_MINEXPOSUREINCREMENT has been fixed.
3. The bug which caused problems when calling SpotSetValue with SPOT_LIVEMAXEXPOSUREMSEC has been fixed.
4. The problem with raw mosaic images from RT-KE mosaic cameras and SpotGetSequentialImages has been fixed.
5. The bug which caused SpotGetLiveImages to crash when computing exposure with RT cameras under very low light conditions has been fixed.
6. The bug which could cause a SPOT_ERRFLATFLDINCOMPATIBLE error code to be returned for mosaic cameras has been fixed.
7. A small resource leak which occurred when the dll was unloaded after SpotInit failed has been fixed.
8. SPOT_LIVEAUTOBRIGHTNESSADJ and SPOT_LIVEAUTOBRIGHTNESSADJX1000 are now supported correctly in SpotGetValue.
9. A couple of bugs which caused live auto-brightness on Insight mosaic cameras to fail to work or to result in corrupted images have been fixed.

Version 3.5.6

1. Support has been added for the new RT-SE slider, RT-KE monochrome, RT-KE color, and RT-KE slider cameras.
2. A bug which caused SpotGetSequentialImages to return SPOT_ABORT instead of SPOT_ERRCOLORFILTERNOTIN or SPOT_ERRCOLORFILTERNOTOUT when the color filter is not in the correct position has been fixed.
3. A bug which caused SpotGetSequentialImages to return SPOT_ERRFLATFLDINCOMPATIBLE for mosaic cameras when flatfield correction is enabled and bDeferProcessing is TRUE has been fixed.
4. A bug which caused changes to SPOT_LIVEGAINADJ and SPOT_LIVEGAINADJX1000 to be ignored when SpotGetLiveImages was called with bComputeExposure==FALSE has been fixed.
5. A bug which caused mosaic images to be corrupted when acquired with SpotGetSequentialImages with noise filtering enabled has been fixed.
6. A bug which caused crashes when computing white balance with mosaic cameras under certain circumstances has been fixed.

Version 3.5.5

1. Updates were made to support modifications in RT-KE camera hardware.

Version 3.5.4.1

1. A bug affecting high gains in live mode on SPOT RT-SE cameras has been fixed.
2. A bug which caused SpotGetCameraAttributes to set the external trigger attribute flag bits for all cameras has been fixed.

Version 3.5.4

1. Support for the new SPOT RT-SE and 3-Shot Insight QE cameras has been added.
2. New attribute flags: SPOT_ATTR_EDGETRIGGER and SPOT_ATTR_BULBTRIGGER have been added to allow an application to determine which external trigger modes, if any, are available on the camera.
3. A new attribute flag: SPOT_ATTR_CLEARFILTER has been added to allow an application to determine if a camera has a clear color filter setting.
4. A new parameter: SPOT_EXTERNALTRIGGERMODE has been added for setting the external trigger mode for cameras with external

triggers.
5. A new status value: SPOT_STATUSWAITINGFORTRIGGER has been added to notify an application that the camera is waiting for and external trigger pulse.
6. The bReserved argument to SpotGetSequentialImages has been changed to allow the application to specify how the external trigger should be used in acquiring sequential images.
7. The WinDriver device driver version has been updated to resolve problems with Win2000 SP3 and some motherboards with USB-2.
8. The bug which caused the computed exposure value for monochrome exposures to be returned in the wrong data member of the SPOT_EXPOSURE_STRUCT2 struct has been fixed.

Version 3.5.2

1. Support for the new SPOT RT-KE cameras has been added.
2. New parameters: SPOT_EXPOSUREINCREMENT and SPOT_MINEXPOSUREMSEC have been added to allow the application to specify the exposure increment and determine the smallest allowable increment. These options allows for the specification of longer exposures for the new RT-KE cameras.
3. New parameters: SPOT_EXPOSURECONVFACTOR and SPOT_EXPOSURECONVFACTORX1000 have been added for use with SpotGetValue for obtaining the factors to use when converting from live mode exposures to still image acquisition.
4. The bug which caused incorrect exposures for 12 bit per channel images acquired after calling SpotGetLiveImage has been fixed.
5. The bug which caused SpotInit to fail when switching camera types has been fixed.
6. The bug which caused the wrong color filter to be used in some flatfield acquisitions has been fixed.

Version 3.5

1. Support for the new SPOT Enhanced, SPOT RT QE, and SPOT Insight QE cameras has been added.
2. A new live image auto-brightness control feature has been added. New parameters: SPOT_LIVEAUTOBRIGHTNESS, SPOT_LIVEAUTOBRIGHTNESSADJ, and SPOT_LIVEAUTOBRIGHTNESSADJX1000 have been added to support this new feature.
3. A new parameter: SPOT_LIVEMAXEXPOSUREMSEC has been added to allow the application to limit the duration of live mode exposures.
4. A new callback call and status value: SPOT_STATUSIMAGEPROCESSING have been added to notify the application when image post-processing is about to begin. The application can assume when it receives this notification that image acquisition has completed, and it now is safe to do operations which would affect the acquisition.
5. The bug which affected color enhancements for some Insight cameras has been fixed.
6. The bug which affected chip defect corrections for some cameras has been fixed.
7. A couple of memory leaks in SpotGetSequentialImages have been fixed.
8. A problem with aborting image acquisition via calls to SpotSetAbortFlag has been fixed.
9. Live image brightness matching has been improved for many cameras.
10. Exposure computation for SPOT cameras has been made more efficient.

Version 3.4

1. The WinRT™ device driver has been replaced by the Jungo™ WinDriver™. This change affects the way in which camera interface cards are specified by the application (see the sections on Hardware Access and SpotInit below).
2. Support for Windows XP has been added.
3. Image acquisition speed and live mode frame rate for RT and Insight cameras have been improved, especially on Windows NT/2000/XP.
4. The API function: SpotFindInterfaceCards has been added to allow the application to determine which camera interface cards are available on the machine.
5. The ability has been added to prevent two or more applications from simultaneously actively controlling the same camera. In this situation, the SPOT_ERRCAMERABUSY error will be returned by the following functions: SpotCloseExternalShutter, SpotComputeExposure, SpotComputeExposure2, SpotComputeWhiteBalance, SpotComputeWhiteBalanceX1000, SpotGetFlatfield, SpotGetImage, SpotGetLiveImages, SpotGetSequentialImages, SpotInit, SpotOpenExternalShutter, SpotQueryCameraPresent, and SpotQueryColorFilterPosition the SPOT_ERRCAMERABUSY.
6. New parameters: SPOT_MAXWHITEBALANCERATIO and MAXWHITEBALANCERATIOX1000 have been added to allow applications to obtain the maximum allowable white balance ratio.
7. A bug affecting 24 bpp flatfields has been fixed.

Version 3.3

1. The ability to do flatfield correction on acquired images has been added. The API function: SpotGetFlatfield and the parameter: SPOT_FLATFLDCORRECT have been added for this purpose.
2. The ability to do noise filtering on acquired images has been added. The parameter: SPOT_NOISEFILTERTHRESPCT has been added for this purpose.

3. The ability to quickly acquire a sequence of images has been added. The API functions: SpotGetSequentialImages and SpotRetrieveSequentialImage and the status codes: SPOT_STATUSSEQIMAGEWAITING and SPOT_STATUSSEQIMAGEREADY have been added for this purpose.
4. The live image frame rate has been improved under WinNT/2000.
5. New error codes: SPOT_ERRNOIMAGEAVAILABLE, SPOT_ERRFILEOPEN, and SPOT_ERRFLATFLDINCOMPATIBLE have been added.
6. New parameters: SPOT_GAINVALS8 and SPOT_GAINVALS12 have been added to allow applications to obtain the allowable gain values for 8 and 12 bit per channel image acquisition separately.
7. The API functions: flatfield and SpotCloseExternalShutter have been added to allow the application to explicitly raise and lower the TTL output signal on the BNC connector on RT and Insight interface cards.

Version 3.2

1. Support has been added for the new SPOT Insight cameras.
2. Problems affecting exposure computation with SPOT RT cameras have been corrected.
3. A new function: SpotGetCameraAttributes has been added to allow the application to determine the attributes of the currently connected camera.
4. A new parameter: SPOT_ENHANCECOLORS has been added to allow the application to enable or disable automatic color enhancement for captured images.
5. A new parameter: SPOT_RETURNRAWMOSAICDATA has been added to allow the application to specify whether it wants the captured images from a mosaic chip camera to be reconstructed.
6. New parameters: SPOT_LIVEACCELERATIONLEVEL and SPOT_MAXLIVEACCELERATIONLEVEL have been added to set the acceleration level for live images and query the maximum allowable acceleration level.
7. A new return error code: SPOT_ERRCAMERANOTSUPPORTED has been added to the list of return values for SpotInit.
8. A new API function: SpotQueryCameraPresent has been added to allow the application to determine whether a camera is connected and functioning.
9. A new API function: SpotQueryColorFilterPosition has been added to allow the application to determine the current color filter position for SPOT RT slider cameras.

Version 3.1

1. A new parameter: SPOT_LIVEEXPOSURE has been added to allow the application to set and get the exposure and gain used for live images with SPOT RT cameras.
2. A new parameter: SPOT_MINEXPOSUREMSEC has been added to allow the application to set the minimum exposure duration to be allowed when computing exposure with SPOT RT cameras.
3. Support has been added for the upcoming models of the SPOT RT cameras which use the Kodak KAI-2000 CCD chip.
4. The bug which caused the default live image gamma adjustment value to be incorrectly set has been fixed.

Version 3.0.1

1. A new error return code: SPOT_ERRCAMERABUSY has been added to indicate that the camera is currently busy performing another operation and cannot execute the requested command.
2. SpotGetLiveImages now works properly for RGB images.
3. The various camera operation commands (SpotComputeWhiteBalance, SpotComputeExposure, etc.) can now be safely called immediately after SpotGetLiveImages returns.
4. The SpotComputeWhiteBalanceX1000 function has been restored.
5. The dll no longer crashes when an error occurs within a new thread.

Version 3.0

1. Support has been added for the new SPOT RT model cameras. To provide this support, the new parameters: SPOT_COLORENABLE2, SPOT_EXPOSURE2, SPOT_EXTERNALSHUTTERENABLE, SPOT_EXTERNALSHUTTERLAG, SPOT_LIVEGAINADJ, SPOT_LIVEGAINADJX1000, SPOT_LIVEGAMMAADJ, SPOT_LIVEGAMMAADJX1000, SPOT_MONITORFILTERPOS, and SPOT_EXPOSURELIMITS2, the new functions: SpotGetLiveImages, SpotComputeExposure2, and SpotGetVersionInfo2, and the new status values: SPOT_STATUSLIVEIMAGEREADY, SPOT_STATUSSHUTTEROPENCLEAR, and SPOT_STATUSIMAGEREADCLEAR have been added.
2. A new parameter: SPOT_SUBTRACTBLACKLEVEL has been added for use with SpotGetValue and SpotSetValue which allows the application to enable or disable automatic black-level subtraction for 12 bpp images.
3. Support for multi-threaded applications has been added.
4. Support has been added to allow the application to select which interface card to use in situations where multiple cards are installed in the

machine. The new parameter: SPOT_DRIVERDEVICENUMBER has been added for this purpose.

Version 2.2.2

1. A new parameter: SPOT_SUBTRACTBLACKLEVEL has been added for use with SpotSetValue/SpotGetValue which allows the enabling/disabling of automatic dark level subtraction for 8 and 12 bpp images. This value is FALSE by default.
2. The SpotComputeExposure, SpotComputeWhiteBalance, and SpotGetImage functions now properly refresh the camera's CCD chip.
3. The SpotQueryStatus function now aborts the camera operation (if bAbort is set) even after SpotSetAbortFlag has been called.

Version 2.2.1

1. The bug which caused occasional crashes when acquiring an image 1032 pixels in hight has been fixed.
2. The exposure computation algorithm has been improved.
3. A new parameter: SPOT_FLUORESCENCECOLORS has been added for use with SpotSetValue/SpotGetValue which allows the application to enable/disable processing to produce better color representation for fluorescence images.
4. A new function: SpotComputeWhiteBalanceX1000 has been added to compute the white balance and return the results in a SPOT_WHITE_BAL_INT_STRUCT structure.

Version 2.2

1. The bug which caused occasional crashes in 36 bpp image acquisition has been fixed.
2. SpotGetValue() with SPOT_WHITEBALANCE now returns correct white balance values.
3. If the camera power is shut off during operation, SpotQueryStatus now reports the error: SPOT_ERRNOCAMERARESP.
4. A few typos in the comments in SpotCam.h have been corrected.
5. A new parameter: SPOT_MESSAGEENABLE has been added for use with SpotSetValue/SpotGetValue which allows the application to enable/disable the processing of queued Windows messages during camera operations. The default value is TRUE.
6. A new parameter: SPOT_WHITEBALANCEX1000 has been added for use with SpotSetValue/SpotGetValue which allows the application to set/get white balance values as long integers using the new SPOT_WHITE_BAL_INT_STRUCT structure. The values in this structure are equal to the actual white balance values multiplied by 1000 and converted to long integers. This alternative method can be used by applications built with compilers which do not correctly handle float values according to IEEE/Intel standards.
7. A new parameter: SPOT_EXPOSUREADJX1000 has been added for use with SpotSetValue/SpotGetValue, which allows the application to set/get the exposure adjustment value as a long integer. The value is equal to the actual exposure adjustment multiplied by 1000 and converted to a long integer. This alternative method can be used by applications built with compilers which do not correctly handle float values according to IEEE/Intel standards.
8. A new parameter: SPOT_EXPOSUREADJLIMITSX1000 has been added for use with SpotGetValue which allows the application to get the minimum and maximum exposure adjustment values as long integers which are equal to the actual limits multiplied by 1000 and converted to long integers.