# SpotCam Driver SDK Reference (Windows/MacOS/Linux)

for the Diagnostic Instruments, Inc. SPOT™ family of digital cameras


version 4.7.5.10
3 July 2008

# Table of Contents

# Overview

The SpotCam driver provides an API through which a applications can control the Diagnostic Instruments SPOT family of digital cameras. There are versions of the driver for Windows (32-bit DLL), Mac OS X (32-bit framework), and Linux (32-bit dynamically-loadable shared library). The API is nearly identical across all three platforms, but not all camera models may be supported on all platforms. This document describes the driver functionality and API. 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 other files for hardware and system access. The files required and installation procedures vary by platform. For Windows and Macintosh, the SPOT application installers will install all of the necessary files and make any necessary system changes. There are also SpotCam driver installers for the different platforms available for download from: http://www.diaginc.com/downloads/spotdll. These installers install the SpotCam driver without the SPOT appplication.

**Windows:** The SpotDrvr.dll file can be used to install or update the DLLs and device drivers. When the SpotCam driver is installed by the SPOT application or SpotCam driver installer, the path of the SpotCam.dll file is recorded in the Registry at: HKLM\SOFTWARE\Diagnostic Instruments, Inc.\SPOT Camera for Windows\Driver Path. Applications should look look for this Registry entry to determine the location of the SpotCam.dll file. The driver can be dynamically loaded with the Windows LoadLibrary API function. It can also be statically linked.

**Mac OS X:** The Spot application installer installs all necessary frameworks and drivers needed. A SpotCamInstaller application can also be used to install the SpotCam.framework and SpotPCI.kext driver, separate from the Spot application. These are installed in Macintosh HD/Library/Frameworks, and Macintosh HD/System/Library/Extensions, respectively.

**Linux:** The Linux version of the driver requires a kernel with the raw1394 module for support of SPOT cameras with an IEEE-1394/FireWire interface. For PCI cameras, the WinDriver$_{TM}$ device driver must be installed. The driver can be dynamically loaded with the POSIX dlopen API function or statically linked.

# Concepts and Usage

## Device Selection and Initialization

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

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

## Camera Attributes and Information

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

An application can obtain information about the current camera's image sensor by querying the values of the SPOT_IMAGESENSORTYPE, SPOT_IMAGESENSORMODELDESCR, SPOT_PIXELSIZE, and SPOT_MAXIMAGERECTSIZE. The SPOT_IMAGESENSORTYPE parameter will be set to one of the defined SPOT_IMAGESENSORTYPE_*xxx* values which indicate the type of device (eg. CCD or CMOS), whether the device is a frame-transfer or interline device, and the types of gain provided (eg. conventional or electron-multiplication). The SPOT_IMAGESENSORMODELDESCR parameter provides a text description of the sensor model (eg. “Kodak KAI-4020CM”). The SPOT_PIXELSIZE parameter indicates the horizontal and vertical size of the sensor's pixels in nm, and the SPOT_MAXIMAGERECTSIZE parameter indicates the width and height, in pixels, of the sensor's imaging area.

Some SPOT cameras use color mosaic image sensors. The SPOT_ATTR_MOSAIC bit flag returned by SpotGetCameraAttributes indicates if

the current camera has such a sensor. An application can determine the mosaic pattern of the sensor by querying the SPOT_MOSAICPATTERN parameter.

# Image Acquisition Parameters

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 the current value of any parameter value. Calling SpotGetValue for any parameter before that parameter has been explicitly set will return its default value.

## Readout Circuit

Some SPOT cameras provide multiple readout circuits. An application can determine how many readout circuits are provided by the current camera by querying the SPOT_NUMBERREADOUTCIRCUITS parameter. If more than one readout circuit is provided, the application can specify which one to use with the SPOT_READOUTCIRCUIT parameter. A text description of the readout circuit can be obtained by querying the SPOT_READOUTCIRCUITDESCR parameter after setting SPOT_READOUTCIRCUIT. The lists of available bit depths, gains, pre-amp gains, and horizontal readout frequencies may also change after SPOT_READOUTCIRCUIT is set, so an application should obtain separate lists for each of these parameters for each readout circuit.

## Bit Depth

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

## Color Enable and Order

For cameras with color filters, an application can specify the filter color or colors to be used for exposure computation or image acquisition by setting the SPOT_COLOR_ENABLE or SPOT_COLOR_ENABLE2 parameter. For cameras with clear color filter states, the SPOT_COLOR_ENABLE2 parameter must be used to enable the clear state. For monochrome exposures with slider cameras, 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.

For multi-shot color acquisitions, an application can specify the order in which the color channels are acquired using the SPOT_COLORORDER parameter. For example, in a three-shot RGB acquisition, setting SPOT_COLORORDER to "BGR" will cause the blue channel data to be acquired first, followed by green and then red.

## Binning

Many SPOT cameras support hardware binning. An application can determine if binning is supported by the current camera by querying the SPOT_BINSIZES parameter. If binning is supported, a list of supported bin sizes will be returned. 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. A bin size of one indicates no binning.

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.

## Imaging Area

An application can specify the area of the image sensor which is to be used for image acquisition by setting the SPOT_IMAGERECT. By default, the full sensor imaging area is used. The size of the image sensor can be determined by querying the SPOT_MAXIMAGERECTSIZE parameter. The minimum allowable imaging area can be determined by querying the SPOT_MINIMAGERECTSIZE parameter.

## Image Data Format and Buffer Size

The SpotCam driver writes image data in different formats depending on the platform, the image bit depth, and other parameters. For 8 bpp images, the image data buffer will contain one byte per pixel, and for higher bit-depth monochrome images, the buffer will contain one word

per pixel with the upper 2, 4, or 6 bits of each word set to zero for 14, 12, and 10 bpp images, respectively.
For 24 bpp images, the image data buffer will contain three or four bytes per pixel, depending on whether or not an alpha channel is present. The value SPOT_24BPPIMAGEBUFFERFORMAT parameter determines the number of bytes per pixel and their ordering. On Windows, the default value is SPOT_24BPPIMAGEBUFFERFORMATBGR, which corresponds to the standard BITMAP format. On MacOS, the default value is SPOT_24BPPIMAGEBUFFERFORMATARGB. On Linux, the default value is SPOT_24BPPIMAGEBUFFERFORMATBGRA. Refer to SpotCam.h for the list of supported formats and their values.

For higher bit-depth color images, the image buffer will contain three words for each pixel in red-green-blue order. The upper bits will be set as they are for monochrome images.

The Windows and Linux versions of the SpotCam driver normally make certain assumptions about the size of the image data buffer. For 8 and 24 bpp images, they assume that each row of the image data is padded with zero or more bytes so that the total number of bytes per row is an integer multiple of four. For other bit depths, the assumption is that image buffer contains the exact number of bytes required for the image with no row padding. An application can provide image buffers which are larger than the default sizes by setting the SPOT_NUMBERBYTESPERIMAGEROW parameter to indicate the actual number of image data buffer bytes used per image row. This allows for the use of data buffers which contain extra bytes for image borders.

In the MacOS version of the driver, each image acquisition function includes an argument which specifies the number of buffer bytes per image line. An application can pass a value of zero for these parameters and specify the number of bytes per line with the SPOT_NUMBERBYTESPERIMAGEROW instead.

The SpotCam driver does not write any data to alpha channel or line padding bytes.

The SPOT_IMAGEORIENTATION parameter determines whether images are saved top-down or bottom-up. A value of 1 indicates that the first values in the image data buffer correspond to the first line of the image, while a value of -1 indicates that the data buffer starts with the last image line. In the MacOS and Linux versions of the driver, the default value is 1, while in the Windows version, the default is -1.

## Exposure and Gain

An application can determine the minimum and maximum supported exposure durations of the current camera by querying the SPOT_EXPOSURELIMITS or SPOT_EXPOSURELIMITS2 parameter. The exposure durations returned for the SPOT_EXPOSURELIMITS parameter are expressed in ms, while those returned for SPOT_EXPOSURELIMITS2 are expressed in the units specified by the SPOT_EXPOSUREINCREMENT parameter.

Some SPOT camera models support multiple methods of setting gain. These are referred to as gain ports. An application can determine how many gain ports are supported by the current camera and readout circuit by querying the value of SPOT_MAXGAINPORTNUMBER. An application can specify which gain port to use by setting SPOT_GAINPORTNUMBER. Gain port 0 supports a set of discrete integer gain values, while the other gain ports support continuous ranges of values. For gain port 0, the SPOT_PORT0GAINVALS8 and SPOT_PORT0GAINVALS16 parameters can be queried to obtain lists of the available gain values at 8 and 10 or more bits per channel, respectively. For the other gain ports, the lower and upper gain value limits can be obtained by querying the SPOT_PORT*n*GAINVALLIMITS parameters. For cameras which support a continuous range of gain values, gain port 0 is used for backward-compatibility. It will provide a subset of discrete gain values within the range supported by one of the other gain ports. The SPOT_PORT*n*GAINATTRIBUTES parameters provide attribute information about each of the gain ports. The SPOT_GAINATTR_*xxx* flags defined in SpotCam.h describe the various gain port attributes. The SPOT_GAINATTR_SAMEASPORT0 attribute bit indicates that the gain port includes the same gain values as port 0. The SPOT_GAINATTR_COMPUTABLE bit indicates that the SpotGetActualGainValue and SpotGetActualLiveGainValue functions can be called to determine the actual floating-point gain values.

An application can set the exposure and gain to be used with the SPOT_EXPOSURE or SPOT_EXPOSURE2 parameters for still image acquisitions, and SPOT_LIVEEXPOSURE for live mode.

## Exposure Units

Most exposure-related parameters use 32-bit unsigned integers to hold exposure durations. These exposure durations are expressed in units specified by the SPOT_EXPOSUREINCREMENT parameter. The default is 500ns. Some SPOT cameras are capable of exposing for durations which are too large to be expressed as a 32-bit unsigned integer in 500ns units. The value of SPOT_EXPOSUREINCREMENT can be increased, preferably by a multiple of a power of 2 to allow longer exposures to be expressed. The SPOT_EXPOSURELIMITS and SPOT_EXPOSURE parameters, which express exposure durations in millisecond increments can also be used for getting and setting very large exposure duration values.

## Exposure Computation

For most SPOT cameras, the SpotCam driver provides exposure computation functionality. An application can determine if the current camera supports this feature by checking the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes. For cameras which provide exposure computation, an application can explicitly compute exposure by calling SpotComputeExposure or SpotComputeExposure2. The application can then set the SPOT_AUTOEXPOSE parameter to FALSE and the SPOT_EXPOSURE or SPOT_EXPOSURE2 parameter with the computed exposure values so that they will be used for subsequent image acquisitions. When the SPOT_AUTOEXPOSE parameter is TRUE, the SpotCam driver will automatically compute exposure before acquiring images.

For live image acquisition, exposure is automatically computed when SpotGetLiveImages is called with the bComputeExposure argument set to TRUE. An application can explicitly set the live mode exposure with the SPOT_LIVEEXPOSURE parameter.

When exposure is computed, for camera and modes which support multiple linear 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. During exposure computation, 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 parameter for still acquisition and SPOT_LIVEBRIGHTNESSADJ for live mode. The default adjustment factor is 1.0.

The exposure values will be computed for the pixels included in the image area set by the SPOT_EXPOSURECOMPRECT parameter. By default, the full sensor imaging area is used. The exposure computation may be affected by the current setting of the SPOT_BITDEPTH parameter. For many 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. For live mode, the exposure will always be computed for the bit depth implied by the camera type and the arguments passed to SpotGetLiveImages. The values of SPOT_COLORENABLE and SPOT_COLORENABLE2 are ignored, and the image bit depth will always be either 8 or 24 bpp.

## Pre-Amp Gain

Some SPOT cameras provide programmable pre-amplifier gain. An application can query the SPOT_PREAMPGAINVALS parameter to determine if the current camera supports this feature. The list of available pre-amp gain values may depend on the current readout circuit setting. The pre-amp gain value can be set with the SPOT_PREAMPGAINVAL parameter.

## Image Sensor Response Mode

Some SPOT camera models provide options which affect the response of the image sensor. An application can query the SPOT_SENSORRESPONSEMODES parameter to obtain a list of available options. The desired mode is set with the SPOT_SENSORRESPONSEMODE parameter. Refer to the SPOT_SENSORRESPMODE_*xxx* flags defined in SpotCam.h for the various options.

## 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 durations, while 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 for exposure computation and image acquisition via the SPOT_WHITEBALANCE parameter. For cameras which are capable of auto-exposure computation, the function can be called to compute the white balance values for the image currently in the camera's view. The application can specify a sub-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.

## Color Filter Slider Position Detection

SPOT cameras with sliding color filters may be able to provide information regarding the current position of the color filter slider. For these cameras, the SpotCam driver will automatically check to determine if the slider is in the appropriate position prior to still image acquisition or exposure or white balance computation when the SPOT_MONITORFILTERPOS parameter value is TRUE. An application can determine if this functionality is supported by the current camera by checking the SPOT_ATTR_SLIDERPOSITIONDETECTION bit returned by the SpotGetCameraAttributes function.

## Horizontal Readout Frequency

Some SPOT cameras provide multiple horizontal readout frequencies for still image capture. The horizontal readout frequency is the frequency at which each line of the image sensor is read. Increasing the frequency will increase data throughput at the expense of readout noise. The SPOT_HORIZREADOUTFREQUENCIES value can be queried to determine which readout frequency choices are available for the current camera and readout circuit. The readout frequency is set by the SPOT_HORIZREADOUTFREQUENCY parameter.

## Vertical Readout Period

Some SPOT cameras provide multiple vertical readout periods for still image capture. The vertical readout period is the amount of time taken to vertically shift each row of pixels on the image sensor. An application can query the SPOT_VERTSHIFTPERIODS parameter to obtain a list of the available vertical shift period values for the current camera. Reducing the vertical readout period will speed up the image readout at the expense of image quality. When using short vertical readout periods, it may be necessary to increase the vertical clock voltage to attain satisfactory image quality. This is done by setting a positive value for the SPOT_VERTCLOCKVOLTAGEBOOST parameter. The SPOT_MAXVERTCLOCKVOLTAGEBOOST parameter can be queried to determine the maximum available vertical clock voltage boost level.

## Image Sensor Clear Modes

Some SPOT camera models provide options as to how to clear the image sensor of accumulated charge. Most SPOT cameras normally perform continuous sensor clearing when idle, and on many cameras, the clear cycle will be immediately preempted when an exposure is to begin. An application can query the SPOT_SENSORCLEARMODES parameter to determine which mode options, if any, are available for the current camera. The clear mode is set via the SPOT_SENSORCLEARMODE parameter. The clear mode options are represented by the defined SPOT_SENSORCLEARMODE*xxx* values ORed together.

## Pixel Resolution

Some SPOT cameras provide options for acquiring images at higher and/or lower pixel resolutions than normal. An application can determine the minimum and maximum pixel resolution levels available with the current camera by querying the SPOT_MINPIXELRESOLUTIONLEVEL and SPOT_MAXPIXELRESOLUTIONLEVEL parameters, respectively. Negative resolution values indicate below normal resolutions, while positive values indicate greater than normal resolutions. A value of zero indicates the normal default resolution. The SPOT_PIXELRESOLUTIONIMGSIZEFACTORS parameter can be queried to determine the approximate factor by which acquired image size will be changed for each of the resolution levels. The parameter will provide an array of float values, one for each resolution level from minimum to maximum. An application can specify which resolution level to use by setting SPOT_PIXELRESOLUTIONLEVEL. The default value is zero.

Higher resolutions are obtained by taking multiple shots 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. Higher resolution levels are not supported for live mode.

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 image sensor is automatically moved to the appropriate position before each shot. The number of shots per single image acquisition for a mosaic camera is equal to: $4r^2$ where r is the resolution level. For monochrome cameras, the number of shots is equal to: $(r + 1)^2$. The number of shots is also given in the lInfo value provided with the SPOT_STATUSGETIMAGE status notification.

Low (negative) resolution levels are obtained by averaging or skipping pixels on the image sensor during readout. Using low resolutions will therefore generally result in higher image download rates. Low resolution levels are supported in both live and still capture modes. The live mode resolution can be set using the SPOT_LIVEPIXELRESOLUTIONLEVEL parameter. Only certain SPOT camera models support low resolutions.

## Sensor Defect Correction

All image sensor chips contain at least some defective pixels. For most SPOT models, each camera is tested at assembly time to determine which pixels are defective, and the locations of these pixels are stored in the camera's non-volatile memory. For these camera, 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. Applications should check the SPOT_ATTR_CHIPDEFECTCORRECTION attribute bit return by SpotGetCameraAttributes to determine if defect correction is supported by the current camera.

## Black-Level Subtraction

Most SPOT camera models are configured so that pixels with no light reaching them will normally have a value of zero. For monochrome still image capture from these cameras 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. Black-level subtraction is normally enabled for live images acquired with SpotGetLiveImages, but it can be disabled for some camera models and monochrome images by setting the SPOT_LIVESUBTRACTBLACKLEVEL parameter to FALSE. An application can check the SPOT_ATTR_BLACKLEVELSUBTRACT attribute bit returned by SpotGetCameraAttributes to determine if the current camera supports black-level subtraction for still images. The SPOT_LIVESUBTRACTBLACKLEVEL parameter is supported for camera models which also support live image histograms and scaling (see section on Live Image Mode).

## Color Enhancement

For color cameras, the SpotCam driver provides a color enhancement feature which makes image colors appear more accurate. Specific color profiles are provided for each camera model. An application can obtain the input and output color profiles for the current camera by querying the SPOT_INPUTCOLORPROFILE and SPOT_OUTPUTCOLORPROFILE parameters. Likewise, the application can specify its own color profile for color enhancement by setting these parameters. An application can also set the color rendering intent to be used for the enhancements. The SPOT_ENHANCECOLORS and SPOT_LIVEENHANCECOLORS parameters are used to enable color enhancement for still acquisition and live mode, respectively, and SPOT_COLORRENDERINGINTENT is used to set the color rendering intent.

## Noise Filtering

The SpotCam driver provides a noise filter feature 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 for multi-shot high resolution acquisition.

## 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 no light reaching the camera's image sensor 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, separate bias frames may 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.

## 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, it must have been acquired with a bit depth greater than or equal to the bit depth of the image to be corrected and at the same resolution level as the image to be corrected. For color filter cameras, it must contain all of the color channels which the image to be corrected contains. 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.

## Flatfield Correction

Flatfield correction is used to compensate for non-uniform illumination and sensitivity of the the image sensor. An application can acquire a flatfield to use for corrections by calling 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, it must have been acquired with a bit depth greater than or equal to the bit depth of the image to be corrected and at the same resolution level as the image to be corrected. For color filter cameras, it must contain all of the color channels which the image to be corrected contains. The area of the sensor used for the flatfield acquisition must completely contain the area used for the image to be corrected. For cameras which do not support black-level subtraction, flatfield correction can only be done if bias frame subtraction is also done. Full flatfield correction is not supported in live mode, but some cameras will do partial flatfield correction in live mode.

# Image Acquisition Modes

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.

## Still Image Acquisition

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 should be queried after the various parameters have been set to determine the width and height of the images which will be acquired.

## Sequential Image Acquisition

The SpotCam driver provides an efficient mechanism for acquiring multiple images through the SpotGetSequentialImages function. This function can be used to acquire multiple images as quickly as possible or at a fixed time interval. It can also be used in a 'run until abort' mode to continuously acquire images for an indefinite period. Some SPOT camera models may do exposure/readout overlapping during sequential image acquisition to maximize the acquisition frame rate. This means that the camera may begin a new exposure during the readout of the previous image frame. The camera's firmware automatically determines whether or not to do overlapping based on the exposure, image size, and readout time. To achieve the highest possible frame rate for sequential image acquisition, SpotGetSequentialImages should be called with SPOT_INTERVALSHORTASPOSSIBLE for nIntervalMSec and TRUE for bDeferProcessing.

For some SPOT camera models, multiple exposure durations can be specified for manual exposure, single-shot acquisition. The camera will cycle through the array of provided exposure durations, acquiring one image at each exposure duration from the array. When the end of the array is reached, the process will be repeated until all images have been acquired. Exposure/readout overlapping will be done wherever applicable. An application can determine whether or not the current camera supports this feature and how many different exposure durations it supports by querying the SPOT_MAXNUMBERSEQIMAGEEXPDURS parameter.

Because of the nature of the image sensors in some SPOT camera models, the minimum possible exposure for sequential image acquisition (interval of SPOT_INTERVALSHORTASPOSSIBLE) may be limited. An application can determine the minimum possible fast sequential image acquisition exposure by querying the SPOT_MINFASTSEQIMAGEEXPDUR after setting all applicable parameters. The returned value will be expressed in the units specified by the SPOT_EXPOSUREINCREMENT parameter. The actual exposures will not be any shorter than the value indicated.

An application can synchronize exposures with other processes by specifying an acquisition interval of zero and using its callback function. Prior to the beginning of each exposure, the application's callback will be called with one of the SPOT_STATUSEXPOSING*xxx* status values, and the exposure will not actually begin until the callback returns. Immediately after each exposure ends, the callback will be called with one of the SPOT_STATUSIMAGEREAD*xxx* values. If software exposure synchronization is not required, the sequential acquisition interval should be either INTERVALSHORTASPOSSIBLE or a positive number.

For sequential acquisition with deferred processing, a disk-caching option is provided. When disk-caching is enabled, raw image data is cached to disk, instead of RAM, during acquisition. After all of the acquisitions have completed, the raw data is read back from disk and

processed. An application can enable disk-caching by setting the SPOT_SEQIMAGEDISKCACHEPATH with the NULL-terminated path of the directory to which cache files should be written. The SpotCam driver will automatically delete cache files when the images are processed.

## 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 or 24 bpp. On many SPOT cameras, live mode normally makes use of multiple amplifier circuits to simultaneously read data from different parts of the image sensor, so the image readout is faster, but there may be minor brightness and/or offset variations between the parts of images. For some of these cameras, an application can specify that live mode should only use a single amplifier channel, so as to improve live image quality at the expense of speed, by setting the SPOT_FORCESINGLECHANLIVEMODE parameter to TRUE. An application can determine if the current camera has multiple live mode amplifiers by checking the SPOT_ATTR_MULTICHANLIVEMODE attribute bit returned by SpotGetCameraAttributes. The SPOT_ATTR_SINGLECHANLIVEMODE attribute bit indicates if the camera supports single-channel live mode.

The SpotGetLiveImages function is used to acquire live images. All acquired images will be written to the same image buffer. The caller may provide this buffer or allow the SpotCam driver to provide it. The SPOT_ACQUIREDLIVEIMAGESIZE parameter should 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.

For the purpose of increasing the live mode frame rate, most SPOT cameras provide a live mode acceleration feature. An application can query the SPOT_MAXLIVEACCELERATIONLEVEL parameter to determine how many acceleration levels, if any, are provided by the current camera, and the desired acceleration level can be set via the SPOT_LIVEACCELERATIONLEVEL parameter. When acceleration is enabled, the camera will skip one or more alternate image lines during readout, and the output images will contain duplicated line data. Acceleration may not be available on some cameras when single-channel live mode readout is being used. In such cases, the value of the SPOT_LIVEACCELERATIONLEVEL parameter is ignored.

The SpotCam driver provides real-time RGB gamma adjustments and color enhancements for live mode images. These features are controlled by the SPOT_LIVEGAMMAADJ and SPOT_LIVEENHANCECOLORS parameters, respectively.

An auto-brightness control feature is provided to maintain the brightness level of live mode images when auto-exposure is used. This feature is enabled by setting the SPOT_LIVEAUTOBRIGHTNESS parameter to TRUE. After the exposure has been computed and live image acquisition has begun, pixel brightness is periodically sampled and the exposure and/or gain may be automatically adjusted to maintain the pixel brightness at its original level. The SPOT_LIVEAUTOBRIGHTNESSADJ parameter can be queried to determine the amount of automatic adjustment which is being used to maintain brightness.

For low-light level imaging, a live image scaling feature is provided on newer SPOT cameras. When this feature is used, full bit depth live image data is scaled before being converted to 8 bits per channel. An application can specify either manual or auto-scaling. When manual scaling is used, the application specifies black and white histogram point values and a scale value. The scale value may be equal to the maximum possible pixel value for the camera's native bit depth, or it may be any other positive value. The raw live image data is first additively adjusted so that pixels at or below the specified black point, appropriately scaled, will have a value of zero. The live mode data is then multiplicatively adjusted by the ratio of the difference of the specified white and black point values and the scale value. When auto-scaling is used, the same operations are performed, except that the black and white point values are automatically computed by examining the histogram of each frame's raw pixel data. The application can specify the percentages of pixels (overflow percentages) which should be at the black and white points as well as the image sensor area to be used for creating the histogram. Auto-histogram stretching cannot be used concurrently with auto-brightness control.

The SPOT_LIVEIMAGESCALING parameter is used to enable and disable live image scaling. When auto-scaling is used, this parameter can be queried at any time to obtained the computed black and white point values. An application can also obtain live image histogram data when scaling is not enabled by setting the SPOT_LIVEHISTOGRAM parameter. Both features can be disabled by passing a NULL-pointer for the parameter. An application can check the SPOT_ATTR_LIVEHISTOGRAM bit returned by SpotGetCameraAttributes to determine if the camera supports these features. The acquisition of live image histogram data requires that single-channel readout be used. When histogram buffers are provided via the SPOT_LIVEHISTOGRAM parameter or auto-scaling is enabled, single-channel readout will be used automatically, and the frame rate will be slower that it would with multi-channel readout. The image scaling itself adds very little or no overhead. The live image histograms always reflect the full bit depth image data before any scaling has been applied.

The following parameters which affect live mode images can be changed while live mode is running, and the change will take effect immediately: SPOT_LIVEBRIGHTNESSADJ, SPOT_LIVEGAMMAADJ, SPOT_LIVEEXPOSURE, SPOT_LIVEENHANCECOLORS,

and SPOT_LIVEIMAGESCALING. Also, SPOT_LIVEAUTOBRIGHTNESS can be set to FALSE while live mode is running to disable auto-brightness adjustment. If any other parameters need to be changed while live mode is running, live mode should be aborted, and the parameters changed after SpotGetLiveImages returns.

An application can determine the exposure used for live mode by querying the SPOT_LIVEEXPOSURE parameter. If the live mode exposure was obtained by auto-exposure, the obtained exposure duration value(s) should be multiplied by the live mode brightness adjustment value (SPOT_LIVEBRIGHTNESSADJ). If live mode auto-brightness adjustment (SPOT_LIVEAUTOBRIGHTNESS) was enabled, the live exposure also needs to be multiplied by the value of the SPOT_LIVEAUTOBRIGHTNESSADJ parameter.

In order to capture a still image with the same brightness level as live mode images, a conversion of the live mode exposure may be necessary. Since some cameras use separate amplifier circuits for live mode and still capture, 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. This 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 to obtain the pre-programmed conversion factor, or it can call SpotComputeExposureConversionFactor to compute an appropriate conversion factor for the current conditions.

## 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 acquisition by calling SpotGetExposureTimestamp after the exposure finishes.

## Image Sensor Cooling

Most SPOT camera models include a cooling fan and many also include a Peltier cooler for the image sensor. An application can determine whether the current camera has a fan or cooler by querying the SPOT_FANSPEEDS and SPOT_COOLINGLEVELS parameters. These parameters indicate the minimum and maximum fan speed and cooling level settings, respectively, supported by the camera. A value of zero indicates the off state. If the camera's fan speed and/or cooling levels cannot be modified, the minimum and maximum values returned for these parameters will be equal. If the returned minimum and maximum values are both zero, the camera does not provide the feature. For cameras with settable fan speeds, the SPOT_FANSPEED and SPOT_FANEXPOSURESPEED parameters can be used to control the speed. For cameras with settable cooling levels, the SPOT_COOLINGLEVEL parameter can be used to set the cooling level.

For cameras which have settable fan speeds, the SpotCam driver provides a feature in which the fan will be automatically slowed or stopped prior to exposure and returned to normal after exposure. This feature is enabled by setting the SPOT_FANEXPOSURESPEED parameter to the desired speed for exposure. The SPOT_FANEXPOSUREDELAYMS parameter should be set to the amount of time to wait, in ms, after changing the fan speed, before beginning exposure. When the camera's cooler is enabled, the fan will normally not be shut off for exposures longer than five minutes, so as to avoid camera overheating.

For cameras with settable cooling levels, the cooler is normally left running at its current setting after SpotExit is called to shut down the camera. An application can specify that the cooler should be automatically turned off upon shutdown by setting the SPOT_COOLERMODEONEXIT parameter to zero prior to calling SpotExit.

## Temperature Readout and Regulation

Some cooled 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 to TRUE and SPOT_REGULATEDTEMPERATURE to the desired regulation temperature. All temperature values are expressed in tenths of a degree C.

## External Trigger Input

Many SPOT cameras provide an external trigger feature which allows external hardware to precisely control exposures. There are two basic external trigger modes: edge and bulb. When external triggering is enabled, the camera will wait for a trigger pulse before beginning an exposure. In edge trigger mode, the exposure will begin with the beginning of the trigger pulse, and the length of the exposure will be controlled by the exposure setting. In bulb mode, the exposure will begin with the beginning of the trigger pulse and continue until the end of the 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 beginning of the trigger pulse 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 entire sequence. External triggers cannot be used with auto-exposure, and bulb trigger mode can only be used for single-shot 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 bits returned by SpotGetCameraAttributes. It can determine whether or not trigger delay is supported by getting the values for the SPOT_EXTERNALTRIGGERDELAYLIMITS parameter. 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 current SPOT camera models provide a TTL output signal which can be used to control external hardware. An application can check the SPOT_ATTR_TTLOUTPUT bit returned by SpotGetCameraAttributes to determine if the current camera has a TTL output. An application can explicitly activate or deactivate their TTL output by calling SpotSetTTLOutputState. For some SPOT cameras, this may not work during an exposure. On some cameras or PCI cards, the physical output port is labeled “TTL Output” while on others it is labeled “Sync Output”.

By default, the TTL output level is low in the inactive state and high in the active state, but this can be changed by setting the SPOT_TTLOUTPUTACTIVESTATE parameter.

All cameras with TTL output can automatically activate it before beginning exposure with a programmable delay period. Many cameras can also activate their TTL output after the beginning of exposure. An application can check the SPOT_ATTR_TTLOUTPUTDURINGEXPOSURE attribute bit to determine if the current camera has this capability. An application can enable the automatic TTL output activation with the SPOT_ENABLETTLOUTPUT parameter and set the delay with SPOT_TTLOUTPUTDELAY or SPOT_TTLOUTPUTDELAYMS. A positive value for SPOT_TTLOUTPUTDELAY or SPOT_TTLOUTPUTDELAYMS indicates that the TTL output is to be activated before exposure, while a negative value indicates that the output should be activated after 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. If this attribute bit is not set, TTL output delay timing can only be set to millisecond resolution.

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.

# Internal Shutters

Some SPOT cameras have internal mechanical shutters which can be used to block all light to the image sensor for dark frame acquisition. Future camera models may also be able to use shutter to control illumination during exposure. An application can check the SPOT_ATTR_INTERNALSHUTTER and SPOT_ATTR_EXPOSURESHUTTER attribute bits returned by SpotGetCameraAttributes to determine if the current camera has an internal shutter and if that shutter can be used to control the illumination during exposure. The SPOT_SHUTTERMODE parameter is used to set the mode of operation of the internal shutter. When this parameter is set to SPOT_SHUTTERMODE_OPEN, the camera's shutter will be kept open at all times, except during bias frame acquisition. Then the parameter is set to SPOT_SHUTTERMODE_CLOSED, the shutter will be kept closed at all times. The SPOT_SHUTTERMODE_NORMAL value indicates normal shutter operation. For cameras which use the shutter for exposure, this means that the shutter will be kept closed except during exposures. For other cameras, SPOT_SHUTTERMODE_NORMAL is the same as SPOT_SHUTTERMODE_OPEN.

# 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), a polling mechanism is provided. To enable polling, an application 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. Polling mode is generally not recommended as it is slightly less efficient than using callbacks.

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)   // Do polling
   {
      .... // Process OS events
      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
         }
      }
   }
```

# System Event Handling

The SpotCam driver provides various mechanisms for allowing an application to handle system and UI events during long acquisition operations. If a busy callback has been registered with the SpotSetBusyCallback function, the registered function will be called periodically (about four times per second) during long operations giving the application an opportunity to process events and/or abort the operation.

In the Windows version of the SpotCam driver, the driver will also periodically flush the application's Windows message queue during long acquisition operations if the SPOT_MESSAGEENABLE parameter is set to TRUE. This option should preferably be disabled when API functions are being called by threads other the one handling the application's UI.

These mechanisms are automatically disabled when polling mode is enabled (SPOT_WAITFORSTATUSCHANGES is TRUE).

# 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. If SpotExit is not explicitly called by the application, it will be called automatically when the SpotCam driver is unloaded. On Windows, this automatic call may not always be safe, so applications should explicitly call SpotExit.

# Special Issues

## Compiler Floating-Point Compatibility (Windows platform)

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 SpotGetActualLiveGainValue functions can be called to obtain the actual non-integer gain values for still-capture and live modes, respectively. 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 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.

## Exposure Units and 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 SpotComputeExposure function and SPOT_EXPOSURE and SPOT_EXPOSURELIMITS parameters which use millisecond increments, or it should set the SPOT_EXPOSUREINCREMENT parameter to a multiple of 500 (eg. 1000000 for 1 ms).

## 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 image. To obtain raw mosaic data, the application must set SPOT_RETURNRAWMOSAICDATA to TRUE, SPOT_BINSIZE to 1, and SPOT_BITDEPTH to the maximum monochrome bit depth supported by the camera before calling the acquisition functions. The values of the pixels will be multiplied by the current white balance values.

## Line-Skipping

Some SPOT cameras support line-skipping for still image capture. Line-skipping can be used to speed up the image readout and download by sacrificing vertical image resolution. When line-skipping is enabled, the camera will skip the specified number of lines of the image sensor between each of the lines read. An application can determine if the current camera supports this feature and the maximum number of lines which can be skipped by querying the SPOT_MAXNUMBERSKIPLINES parameter. A value of zero indicates that the feature is not supported. The application can set the number of lines to skip with the SPOT_NUMBERSKIPLINES parameter. The height value returned by the SPOT_ACQUIREDIMAGESIZE parameter will reflect the current line-skipping setting.

## 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. However, 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 camera models, the SpotCam driver has the ability to notify an application when a camera is added or removed from the computer or powered on or off. The application can register a callback function for receiving these notifications with the SpotSetDeviceNotificationCallback function. Note that the calls to the application's callback function may be made from multiple threads.

## System Power State Control

Before acquiring images from SPOT IEEE1394/FireWire cameras, the SpotCam driver makes certain system calls to prevent power state changes which can cause image data corruption during acquisition. These power state issues primarily affect notebook PCs with speed-stepping processors. There is a small time penalty (5ms or less) for making these calls. If it is known that the host computer does not have any power state issues, this overhead can be eliminated by setting the SPOT_ENABLEPOWERSTATECONTROL parameter to FALSE. This functionality is currently supported in the Windows and MacOS versions of the driver.

## Firmware Updating

The SpotCam driver provides a facility for updating the firmware of cameras and interface cards which support the capability. An application can determine which cameras have this capability by checking for the SPOT_ATTR_FIRMWAREUPDATE bit in the

dwAttributes member of the SPOT_DEVICE_STRUCT. The firmware update is done by the SpotUpdateFirmware function. Applications should normally not need to support this feature as Diagnostic Instruments will provide a stand-alone firmware updating utility with each firmware update release.

# 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.

**See Also**

SpotQueryStatus, SpotSetCallback

## 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_ERRNOTCAPABLE | The camera model is not capable of performing the operation. |
| SPOT_ERROUTOFMEMORY | out of memory |

| | |
|---|---|
| SPOT_RUNNING | The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE) |

**Remarks**

Applications should determine if the current camera supports exposure computation by checking the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes before calling this function.

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 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.

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**

Applications should determine if the current camera supports exposure computation by checking the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes before calling this function.

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 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.

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_ERRNOTCAPABLE | The camera model is not capable of performing the operation. |
| SPOT_ERROUTOFMEMORY | out of memory |
| SPOT_RUNNING | The operation is running (when SPOT_WAITFORSTATUSCHANGES is TRUE) |

**Remarks**

Applications should determine if the current camera supports exposure computation by checking the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes before calling this function.

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_ERRNOTCAPABLE | The camera model is not capable of performing the operation. |
| 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**

Applications should determine if the current camera supports exposure computation by checking the SPOT_ATTR_AUTOEXPOSURE attribute bit returned by SpotGetCameraAttributes before calling this function. Cameras which do not support exposure computation cannot compute white balance.

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 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_ERRCAMERABUSY | The camera is currently performing another operation. |
| 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. If this function returns SPOT_ERRCAMERABUSY, it indicates that the camera is in the process of performing an operation which must either complete or be aborted before SpotExit can be called successfully.

**See Also**

SpotInit

# SpotFindDevices

The SpotFindDevices 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.

# SpotGetActualGainValue

The SpotGetActualGainValue function is used to get actual gain values for cameras which provide non-integer gains. An application should check the value of the SPOT_PORT*n*GAINATTRIBUTES parameter for the SPOT_GAINATTR_COMPUTABLE bit to verify that the actual

gain can be computed before calling this function.

```
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 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. An application should check the value of the SPOT_PORT*n*GAINATTRIBUTES parameter for the SPOT_GAINATTR_COMPUTABLE bit to verify that the actual gain can be computed before calling this function.

```
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 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. An application should check the value of the SPOT_PORT*n*GAINATTRIBUTES parameter for the SPOT_GAINATTR_COMPUTABLE bit to verify that the actual gain can be computed before calling this function.

```
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. An application should check the value of the SPOT_PORT*n*GAINATTRIBUTES parameter for the SPOT_GAINATTR_COMPUTABLE bit to verify that the actual gain can be computed before calling this function.

```
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 SpotGetActualGainValue.

**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**

SpotGetBackgroundImageCompatibilityInformation, SpotGetBiasFrame, SpotGetFlatfield, SpotGetFlatfield2

# SpotGetBackgroundImageCompatibilityInformation

The SpotGetBackgroundImageCompatibilityInformation function is used to obtain information about a background image to allow an application to determine is it can be used to correct acquired images.

```
int WINAPI SpotGetBackgroundImageCompatibilityInformation(
        char *pszFileName,
        SPOT_BKGD_IMAGE_COMPATIBILITY_INFO_STRUCT *pstInfo
);
```

**Parameters**

*pszFileName*

Points to NULL-terminated character string which specifies the name of the bias frame file to use.

*PstInfo*

Points to a SPOT_BKGD_IMAGE_COMPATIBILITY_INFO_STRUCT struct to hold the retrieved 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_ERRFILEOPEN | The specified file cannot be opened. |
| SPOT_ERRINVALIDFILE | The specified file is not a valid background image file, or the background image is not valid for the current camera. |

# 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**

SpotGetBiasFrameCompatibilityInformation, SpotGetFlatfield, SpotGetFlatfield2, SpotGetBackgroundImage

# SpotGetBiasFrameCompatibilityInformation

The SpotGetBiasFrameCompatibilityInformation function is used to obtain information about a bias frame to allow an application to determine is it can be used to correct acquired images.

```
int WINAPI SpotGetBiasFrameCompatibilityInformation(
        char *pszFileName,
        SPOT_BIAS_FRAME_COMPATIBILITY_INFO_STRUCT *pstInfo
);
```

**Parameters**

*pszFileName*

Points to NULL-terminated character string which specifies the name of the bias frame file to use.

*PstInfo*

Points to a SPOT_BIAS_FRAME_COMPATIBILITY_INFO_STRUCT struct to hold the retrieved 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_ERRFILEOPEN | The specified file cannot be opened. |
| SPOT_ERRINVALIDFILE | The specified file is not a valid bias frame file, or the bias frame is not valid for the current camera. |

# 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. |
| SPOT_ATTR_BLACKLEVELSUBTRACT | The camera can do black level subtraction. |
| 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_CHIPDEFECTCORRECTION | The driver can correct for pixel defects in the camera's image sensor. |
| 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_EXPOSURESHUTTER | The camera's internal mechanical shutter can be used during exposure. |
| SPOT_ATTR_FILTERWHEEL | The camera has a mechanical color filter wheel. |
| SPOT_ATTR_FIRMWAREUPDATE | The camera's firmware can be updated through the SpotCam API. |
| SPOT_ATTR_INTERNALSHUTTER | The camera contains an internal mechanical shutter. |
| SPOT_ATTR_LIVEHISTOGRAM | The camera can provide histogram information and do stretching in live mode. |
| SPOT_ATTR_LIVEMODE | The camera supports the live image mode. |
| SPOT_ATTR_MOSAIC | The camera has a color mosaic sensor. |
| SPOT_ATTR_MULTICHANLIVEMODE | The camera can use multiple parallel readout channels for live mode. |
| SPOT_ATTR_SENSORSHIFTING | The camera can shift the position of the image sensor for greater pixel resolution. |
| SPOT_ATTR_SINGLECHANLIVEMODE | The camera can use a single readout channel for live mode. |
| 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. |
| SPOT_ATTR_TTLOUTPUTDURINGEXPOSURE | The camera can activate its TTL output during an exposure. |

# SpotGetCameraErrorCode

The SpotGetCameraErrorCode function is used to retrieve the error code from the camera after an error occurs. Currently, only certain camera models support this function. This error code may be useful in diagnosing camera malfunctions.

int WINAPI SpotGetCameraErrorCode();

**Return Value**

The function will return a value which will depend on the camera model and type of error. If no error has occurred, or the current

camera does not provide error codes, the function will return zero.

## SpotGetExposureTimestamp

The SpotGetExposureTimestamp function is used to obtain the timestamp of the beginning of the last exposure for still image acquisition.

```
int WINAPI SpotGetExposureTimestamp(
        SPOT_TIMESTAMP_STRUCT *pstTimestamp
);
```

**Parameters**

*pstTimestamp*

Points to a SPOT_TIMESTAMP_STRUCT struct to receive the timestamp.

**Return Value**

If the function succeeds, the return value is SPOT_SUCCESS. If the driver has not been initialized, SPOT_ERRDRVNOTINIT is returned:

**Remarks**

The exposure timestamp for each image acquisition can be obtained as soon as one of the SPOT_STATUSIMAGEREAD*xxx* status values has been received for that acquisition.

## 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_ERRIMAGETOOBRIGHT | The acquired image is too bright (non-autoexposure cameras only). |
| 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 (for cameras which support it), no binning, and the maximum bit depth. 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**

SpotGetFlatfieldCompatibilityInformation , 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_ERRIMAGETOOBRIGHT | The acquired image is too bright (non-autoexposure cameas only). |
| 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 (for the cameras which support it), no binning, and the maximum bit depth. 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**

SpotGetFlatfieldCompatibilityInformation, SpotGetBiasFrame, SpotGetFlatfield, SpotGetBackgroundImage

# SpotGetFlatfieldCompatibilityInformation

The SpotGetFlatfieldCompatibilityInformation function is used to obtain information about a flatfield to allow an application to determine is it can be used to correct acquired images.

```
int WINAPI SpotGetFlatfieldCompatibilityInformation(
        char *pszFileName,
        SPOT_FLATFIELD_COMPATIBILITY_INFO_STRUCT *pstInfo
);
```

**Parameters**

*pszFileName*

Points to NULL-terminated character string which specifies the name of the flatfield file to use.

*PstInfo*

Points to a SPOT_FLATFIELD_COMPATIBILITY_INFO_STRUCT struct to hold the retrieved 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_ERRFILEOPEN | The specified file cannot be opened. |
| SPOT_ERRINVALIDFILE | The specified file is not a valid flatfield file, or the flatfield is not valid for the current camera. |

# SpotGetImage

The SpotGetImage function is used to capture still images.

*Windows and Linux:*

```
int WINAPI SpotGetImage(
        short nReserved1,
        BOOL bReserved,
        short nReserved2,
        void *pImageBuffer,
        long *plRedPixelCnts,
        long *plGreenPixelCnts,
        long *plBluePixelCnts
);
```

*MacOS:*

```
int WINAPI SpotGetImage(
        short nReserved1,
        unsigned long lRowBytes,
        BOOL bReserved,
        short nReserved2,
        void *pImageBuffer,
        long *plRedPixelCnts,
```

```
    long *plGreenPixelCnts,
    long *plBluePixelCnts
);
```

**Parameters**

*nReserved1*

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

*lRowBytes*

Specifies the number of image data buffers bytes per image row. *(MacOS only)*

*bReserved*

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

*nReserved2*

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. 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 bpp images, this buffer pointed to by pImageBuffer must correspond to the bmBits member of a Windows BITMAP structure with one byte per pixel and zero to three padding bytes per image line so that the number of bytes per line is an integer multiple of four. For 24 bpp images, the format of the buffer depends on the value of SPOT_24BPPIMAGEBUFFERFORMAT. By default, the Windows 24 bpp BITMAP format is used, with three bytes per pixel (BGR). Each image line will contain zero to three padding bytes 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 least significant bits of each word, and there is no padding. The images will be stored in the orientation specified by the SPOT_IMAGEORIENTATION parameter. The default orientation is bottom-up. An application can determine the width and height, in pixels, 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, and the camera supports exposure computation, 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.

If the application has registered a callback function, it will be called with one of the SPOT_STATUSEXPOSING*xxx* status values prior to exposure, and the exposure will not actually begin until the callback returns. Immediately after the exposure ends, the callback will be called with one of the SPOT_STATUSIMAGEREAD*xxx* status values. An application can use these status notifications to synchronize exposures with external processes.

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

*Windows and Linux:*

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

*MacOS:*

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

**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, or if the camera does not support exposure computation, 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, SPOT_COLORRG, SPOT_COLORRB, SPOT_COLORGB, and SPOT_COLORNONE. If SPOT_COLORRGB, SPOT_COLORRG,

SPOT_COLORRB, or SPOT_COLORGB are 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 parameter may be NULL, in which case the buffer will be allocated and freed by the SpotCam driver. See the Remarks section for more information.

*lRowBytes*

Specifies the number of image data buffers bytes per image row. *(MacOS only)*

**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 checking the SPOT_ATTR_LIVEMODE bit returned by SpotGetCameraAttributes.

The function will not return until an error occurs, or the operation is aborted. If the application has set a callback function, it will be called with the status value SPOT_STATUSLIVEIMAGEREADY as each image is ready. The lInfo value provided with this status will contain the address of the image data buffer. 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, and the camera supports exposure computation, 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 in the orientation specified by the SPOT_IMAGEORIENTATION parameter (bottom-up by default). For monochrome images, the buffer must contain one byte per pixel, and for color images, three or four bytes as specified by the SPOT_24BPPIMAGEBUFFERFORMAT parameter. Each image line will contain zero to three padding bytes so that the number of bytes per line is an integer multiple of four. The application can

determine the size, in pixels, 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 or accessing the image buffer. If nRotateDirection is not SPOT_ROTATENONE, the image will be rotated before being placed into the buffer.

If the application passes a NULL pointer to this function for the pImageBuffer parameter, the SpotCam driver will allocate the image buffer before acquiring any images, and free the buffer before returning from the function. The address of the image buffer will provided in the lInfo value provided with the SPOT_STATUSLIVEIMAGEREADY status notification.

Images returned by this function may be of a lower quality than those returned by SpotGetImage or SpotGetSequentialImages. On many SPOT cameras, multiple amplifier circuits are normally used for live mode to simultaneously read data from different parts of the image sensor, so there may be minor brightness and/or offset variations between parts of images. For some of these cameras, an application can specify that live mode should only use a single amplifier channel so as to improve live image quality at the expense of speed by setting the SPOT_FORCESINGLECHANLIVEMODE parameter to TRUE. An application can determine if the current camera has multiple live mode amplifiers by checking the SPOT_ATTR_MULTICHANLIVEMODE attribute bit returned by SpotGetCameraAttributes. The SPOT_ATTR_SINGLECHANLIVEMODE attribute bit indicates if the camera supports single-channel live mode. For high-quality multiple image acquisition, SpotGetSequentialImages should be used.

If image scaling has been enabled with the SPOT_LIVEIMAGESCALING parameter, or if histogram buffers have been provided with the SPOT_LIVEHISTOGRAM parameter, live mode will automatically use single-channel readout even on cameras which normally use multi-channel readout. Once histogram buffers have been specified with the SPOT_LIVEHISTOGRAM parameter, they must not be deleted until after SpotGetLiveImages returns. The live image histogram data for each image frame is available at the same time as the image data.

The parameters: SPOT_LIVEENHANCECOLORS, SPOT_LIVEGAINADJ, SPOT_LIVEGAINADJX1000, SPOT_LIVEGAMMAADJ, SPOT_LIVEGAMMAADJX1000, SPOT_LIVEEXPOSURE, and SPOT_LIVEIMAGESCALING can be changed, and SPOT_LIVEAUTOBRIGHTNESS can be set to FALSE while the live image acquisition is running and will take effect immediately. All other parameter changes 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.

**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**

The application can determine if the camera is capable of providing temperature readout by checking the SPOT_ATTR_TEMPERATUREREADOUT bit returned by SpotGetCameraAttributes.

**See Also**

SpotGetSensorExposureTemperature

# 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 determine if the camera is capable of providing temperature readout by checking the SPOT_ATTR_TEMPERATUREREADOUT bit returned by SpotGetCameraAttributes. This function can be called as soon as one of the SPOT_STATUSIMAGEREAD*xxx* statuses is received to indicate that an exposure has just ended.

**See Also**

SpotGetSensorCurrentTemperature

# SpotGetSequentialImages

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

*Windows and Linux:*

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

*MacOS:*

```
int WINAPI SpotGetSequentialImages(
        int nNumImages,
        int nIntervalMSec,
```

```
    BOOL bAutoExposeOnEach,
    BOOL bUseTriggerOrSetTTLOuputOnEach
    BOOL bDeferProcessing,
    void **ppImageBuffers,
    unsigned long lRowBytes
);
```

**Parameters**

*nNumImages*

The number of image to acquire. If this value is SPOT_INFINITEIMAGES, images will be acquired until the operation is aborted.

*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. The value should be zero if the exposures need to be synchronized through software with an external process. This value will be ignored if an 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 activated 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. If a disk-caching path has been specified (with the SPOT_SEQIMAGEDISKCACHEPATH parameter), the raw image data will be cached to disk. Otherwise, it will be cached in memory.

*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. If *nNumImages* is SPOT_INFINITEIMAGES, only the first buffer in the array, if any, will be used, and all images will be placed in the same buffer after processing.

*lRowBytes*

Specifies the number of image data buffers bytes per image row. *(MacOS only)*

**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, except that it is more efficient for multiple image acquisition. 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_STATUSEXPOSING*xxx*, SPOT_STATUSIMAGEREAD*xxx*, and SPOT_STATUSSEQIMAGEREADY statuses for each image acquired.

If an application needs to synchronize exposures with some external process such as moving the specimen or changing the illumination between acquisitions, it should specify zero for nIntervalMSec and watch for SPOT_STATUSEXPOSING*xxx* and SPOT_STATUSIMAGEREAD*xxx* status notifications. Prior to the beginning of each exposure, the application will receive a notification with one of the SPOT_STATUSEXPOSING*xxx* status values, and the exposure will not actually begin until the callback returns (or SpotWaitForStatusChange is called again). Immediately after each exposure ends, a notification with one of the SPOT_STATUSIMAGEREAD*xxx* values will be receieved. If software exposure synchronization is not required, nIntervalMSec should be either INTERVALSHORTASPOSSIBLE or a positive number.

For manual exposure, single-shot acquisitions, and some camera models, multiple exposure durations are supported. An application can determine if this feature is supported for the current camera and how many exposures can be specified by querying the SPOT_MAXNUMBERSEQIMAGEEXPDURS parameter. The application can specify an array of exposures to be used with the SPOT_SEQIMAGEEXPDURS parameter. The exposures are expressed in units specified with the SPOT_EXPOSUREINCREMENT parameter. When multiple exposure durations are being used, each sequential acquisition is done using the corresponding exposure duration from the array. The SpotCam driver will cycle through the exposure duration array as many times as needed to acquire all images.

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 always required for each acquisition.

If TTL output is enabled, and the bUseTriggerOrSetTTLOuputOnEach argument is FALSE, the TTL output will be activated before the beginning of the first exposure and deactivated after the last exposure ends. If bUseTriggerOrSetTTLOuputOnEach is TRUE, the TTL output will be activated and deactivated separately for each acquisition.

If the ppImageBuffers argument is NULL, the application will need to call SpotRetrieveSequentialImage to retrieve each image after its the SPOT_STATUSSEQIMAGEREADY status notification is received for that image. It can call SpotRetrieveSequentialImage at any time after receiving the notification.

If the bDeferProcessing argument is TRUE, the acquired raw image data will be cached until the last acquisition completes. If a cache directory has been specified with the SPOT_SEQIMAGEDISKCACHEPATH parameter, the raw data will be cached to the disk as indicated. Otherwise, the raw data will be cached in RAM , and temporary image data buffers may need to created for all of the images acquired, even if the application has provided buffers. In such a case, it is therefore be advisable for applications to pass NULL for ppImageBuffers and retrieve each image via SpotRetrieveSequentialImage. Disk-caching can be disabled by setting the SPOT_SEQIMAGEDISKCACHEPATH parameter with a NULL-pointer or NULL string.

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. For cameras with frame-transfer sensors, when SPOT_INTERVALSHORTASPOSSIBLE is specified for the interval, frame transfer mode will be used. After setting all relevant parameters, an application should query the SPOT_MINFASTSEQIMAGEEXPDUR parameter to determine the minimum exposure possible when the interval is SPOT_INTERVALSHORTASPOSSIBLE.

This function can be used to acquire images in an infinite loop by passing SPOT_INFINITEIMAGES for nNumImages. If an image buffer is supplied, all images will be placed in the same buffer. If no buffer is supplied (ppImageBuffers is NULL), SpotRetrieveSequentialImage must be called to retrieve each image at the time that the SPOT_STATUSSEQIMAGEREADY status is reported.

**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_24BPPIMAGEBUFFERFORMAT | The format used for storing 24 bpp images in memory. |
| 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_BKGDIMAGESUBTRACT | The name of the file to be used for background image subtraction or NULL if background image subtraction is disabled. |
| SPOT_BRIGHTNESSADJ | The adjustment factor to be applied to computed exposures to adjust brightness. |
| SPOT_BRIGHTNESSADJLIMITS | The minimum and maximum allowable brightness adjustment values. |
| SPOT_BRIGHTNESSADJLIMITSX1000 | The minimum and maximum allowable brightness adjustment values multiplied by 1000. |
| SPOT_BRIGHTNESSADJX1000 | The adjustment factor to be applied to computed exposures to adjust brightness multiplied 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 binning 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_COLORRENDERINGINTENT | The rendering intent to be used for color enhancements |
| SPOT_COOLERMODEONEXIT | The mode for the cooler after SpotExit is called. |
| SPOT_COOLINGLEVEL | The level of cooling of the camera's image sensor |

| | |
|---|---|
| SPOT_COOLINGLEVELS | The minimum and maximum supported levels of cooling of the camera's image sensor. |
| 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_ENABLEPOWERSTATECONTROL | Enables or disables the control of system power state changes |
| 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_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_FANEXPOSUREDELAYMS | The delay to wait before exposing after changing the fan speed, in ms. |
| SPOT_FANEXPOSURESPEED | The speed to which the fan is set for exposure. |
| SPOT_FANSPEED | The speed setting for the camera's cooling fan. |
| SPOT_FANSPEEDS | The minimum and maximum speed settings supported for the camera's cooling fan. |
| SPOT_FLATFLDCORRECT | The name of the file to be used for flatfield correction or NULL if flatfield correction is disabled. |
| SPOT_FORCESINGLECHANLIVEMODE | Forces live mode to use a single amplifier circuit. |
| SPOT_GAINPORTNUMBER | The currently selected gain port. |
| SPOT_GAINVALS16 | The allowable gain values for 10-16 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_IMAGESENSORMODELDESCR | The text description of the model of image sensor chip in the camera. |
| SPOT_IMAGESENSORTYPE | The type of image sensor chip in the camera. |
| SPOT_IMAGETYPE | The image type (bright or dark field) used for exposure computation. |
| SPOT_INPUTCOLORPROFILE | The name of the file containing the input ICC profile information for color enhancements. |
| 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_LIVEHISTOGRAM | The buffers for holding live image histogram data. |

| | |
|---|---|
| SPOT_LIVEIMAGESCALING | Specifies live mode image scaling. |
| SPOT_LIVEMAXEXPOSUREMSEC | The maximum allowable exposure for live image acquisition in msec. |
| SPOT_LIVEPIXELRESOLUTIONLEVEL | The pixel resolution level used for live image acquisition. |
| SPOT_LIVESUBTRACTBLACKLEVEL | Enables or disables automatic black level subtraction for live monochrome images. |
| 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_MAXNUMBERSEQIMAGEEXPDURS | The maximum number of sequential acquisition exposure durations supported. |
| SPOT_MAXNUMBERSKIPLINES | The maximum allowable number of image lines to skip during readout. |
| SPOT_MAXPIXELRESOLUTIONLEVEL | The maximum allowable pixel resolution level. |
| SPOT_MAXVERTCLOCKVOLTAGEBOOST | The maximum allowable vertical clock voltage boost 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_MINEXPOSUREMSEC | The minimum exposure, in msec, to be allowed for computed exposures. |
| SPOT_MINIMAGERECTSIZE | The minimum allowable image area width and height |
| SPOT_MINFASTSEQIMAGEEXPDUR | The minimum possible exposure for fast sequential image acquisition (SPOT_INTERVALSHORTASPOSSIBLE). |
| SPOT_MINPIXELRESOLUTIONLEVEL | The minimum allowable pixel resolution level. |
| 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_NUMBERBYTESPERIMAGEROW | The number of bytes in the image data buffer per image row. |
| SPOT_NUMBERREADOUTCIRCUITS | The number of image readout circuits available in the camera. |
| SPOT_NUMBERSKIPLINES | The number of image lines to skip during readout. |
| SPOT_OUTPUTCOLORPROFILE | The name of the file containing the output ICC profile information for color enhancements. |
| SPOT_PIXELRESOLUTIONIMGSIZEFACTORS | The approximate image size adjustment factors for each pixel resolution level. |
| SPOT_PIXELRESOLUTIONLEVEL | The pixel resolution level used for still image acquisition. |
| SPOT_PIXELSIZE | The x and y sizes of the sensor pixels, in nm. |
| SPOT_PORT0GAINATTRIBUTES | The attributes of gain port 0 |
| SPOT_PORT0GAINVALS16 | The allowable port 0 gain values for 10-16 bit per channel still image capture. (same as SPOT_GAINVALS16) |
| 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_PORT1GAINATTRIBUTES | The attributes of gain port 1 |
| 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_PORT2GAINATTRIBUTES | The attributes of gain port 2 |
| 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_PORT3GAINATTRIBUTES | The attributes of gain port 3 |
| 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_PREAMPGAINVALS | The allowable pre-amplifier gain values. |
| SPOT_PREAMPGAINVAL | The pre-amplifier gain value. |
| SPOT_READOUTCIRCUIT | The index of the image data readout circuit to be used during image acquisitions. |
| SPOT_READOUTCIRCUITDESCR | The text description of the currently selected image readout circuit. |
| 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_SENSORRESPONSEMODE | The sensor response mode. |
| SPOT_SENSORRESPONSEMODES | The sensor response modes which supported by the camera. |
| SPOT_SEQIMAGEDISKCACHEPATH | The path to which cache files should be written during sequential acquisition |
| SPOT_SEQIMAGEEXPDURS | The exposure durations to be used for sequential acquisition |
| SPOT_SHUTTERMODE | The mode of operation of the camera's internal shutter. |
| SPOT_SUBTRACTBLACKLEVEL | Enables or disables automatic black level subtraction for acquisition of 10+ bpp monochrome images. |
| 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. A negative value indicates activation during exposure. |
| SPOT_TTLOUTPUTDELAYLIMITS | The minimum and maximum allowable TTL output delay values, in µsec. |
| SPOT_TTLOUTPUTDELAYMS | The wait time, in msec after raising the TTL output before exposing. |
| SPOT_VERTCLOCKVOLTAGEBOOST | The vertical clock voltage boost level. |
| SPOT_VERTSHIFTPERIOD | The vertical shift period, in ns. |
| SPOT_VERTSHIFTPERIODS | The allowable vertical shift periods, in ns. |
| 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_24BPPIMAGEBUFFERFORMAT | short (one of the SPOT_24BPPIMAGEBUFFERFORMAT*xxx* values). |
| 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_BKGDIMAGESUBTRACT | char (NULL-terminated string) |
| 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_COLORRENDERINGINTENT | short (one of the defined SPOT_COLORRENDERINGINTENT_*xxx* values) |
| SPOT_COOLERMODEONEXIT | short (0:turn off, 1:leave on) |
| SPOT_COOLINGLEVEL | short |
| SPOT_COOLINGLEVELS | two short values, first value is minimum, second is maximum |
| SPOT_CORRECTCHIPDEFECTS | BOOL |
| SPOT_DEVICEUID | SPOT_DEVICE_UID union |

| | |
|---|---|
| SPOT_DRIVERDEVICENUMBER | short |
| SPOT_ENABLEPOWERSTATECONTROL | BOOL |
| 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_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_FANEXPOSUREDELAYMS | short |
| SPOT_FANEXPOSURESPEED | short |
| SPOT_FANSPEED | short |
| SPOT_FANSPEEDS | two short values, first value is minimum, second is maximum |
| SPOT_FLATFLDCORRECT | char (NULL-terminated string) |
| SPOT_FORCESINGLECHANLIVEMODE | BOOL |
| SPOT_GAINPORTNUMBER | short |
| SPOT_GAINVALS16 | 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_IMAGESENSORMODELDESCR | char (NULL-terminated string) |
| SPOT_IMAGESENSORTYPE | DWORD (one of the SPOT_IMAGESENSORTYPE_*xxx* values) |
| SPOT_IMAGETYPE | short (SPOT_IMAGEBRIGHTFLD or SPOT_IMAGEDARKFLD) |
| SPOT_INPUTCOLORPROFILE | char (NULL-terminated string) |
| 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_LIVEHISTOGRAM | SPOT_LIVE_HISTOGRAM_STRUCT structure |
| SPOT_LIVEIMAGESCALING | SPOT_LIVE_IMAGE_SCALING_STRUCT structure |
| SPOT_LIVEMAXEXPOSUREMSEC | long |
| SPOT_LIVEPIXELRESOLUTIONLEVEL | short |
| SPOT_LIVESUBTRACTBLACKLEVEL | BOOL |
| 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_MAXNUMBERSEQIMAGEEXPDURS | short |
| SPOT_MAXNUMBERSKIPLINES | short |
| SPOT_MAXPIXELRESOLUTIONLEVEL | short |
| SPOT_MAXVERTCLOCKVOLTAGEBOOST | short |
| SPOT_MAXWHITEBALANCERATIO | float |
| SPOT_MAXWHITEBALANCERATIOX1000 | long |
| SPOT_MESSAGEENABLE | BOOL |
| SPOT_MINEXPOSUREMSEC | short |
| SPOT_MINIMAGERECTSIZE | array of short values, first value is minimum width, second is minimum height |
| SPOT_MINFASTSEQIMAGEEXPDUR | QWORD |
| SPOT_MINPIXELRESOLUTIONLEVEL | short |
| SPOT_MONITORFILTERPOS | BOOL |
| SPOT_MOSAICPATTERN | short (one of the SPOT_MOSAIC*xxx* values) |
| SPOT_NOISEFILTERTHRESPCT | short |
| SPOT_NUMBERBYTESPERIMAGEROW | long |
| SPOT_NUMBERREADOUTCIRCUITS | short |
| SPOT_NUMBERSKIPLINES | short |
| SPOT_OUTPUTCOLORPROFILE | char (NULL-terminated string) |
| SPOT_PIXELRESOLUTIONIMGSIZEFACTORS | array of floats, one for each pixel resolution level from minimum to maximum |
| SPOT_PIXELRESOLUTIONLEVEL | short |
| SPOT_PIXELSIZE | array of two longs, first value is x size, second is y size |
| SPOT_PORT0GAINATTRIBUTES | DWORD (SPOT_GAINATTR_*xxx* bit-fields) |
| SPOT_PORT0GAINVALS16 | 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_PORT1GAINATTRIBUTES | DWORD (SPOT_GAINATTR_*xxx* bit-fields) |
| 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_PORT2GAINATTRIBUTES | DWORD (SPOT_GAINATTR_*xxx* bit-fields) |
| 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_PORT3GAINATTRIBUTES | DWORD (SPOT_GAINATTR_*xxx* bit-fields) |
| 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_PREAMPGAINVALS | array of float values, first value is count of values following |
| SPOT_PREAMPGAINVAL | float |
| SPOT_READOUTCIRCUIT | short |
| SPOT_READOUTCIRCUITDESCR | char (NULL-terminated string) |
| SPOT_REGULATEDTEMPERATURE | short |
| SPOT_REGULATEDTEMPERATURELIMITS | two short values, first value is minimum, second is maximum |
| SPOT_REGULATETEMPERATURE | BOOL |
| SPOT_RETURNRAWMOSAICDATA | BOOL |
| SPOT_SENSORRESPONSEMODE | DWORD (SPOT_SENSORRESPONSEMODE_*xxx* bit-fields) |
| SPOT_SENSORRESPONSEMODES | array of DWORDs, first value is count of values following |
| SPOT_SEQIMAGEDISKCACHEPATH | char (NULL-terminated string) |
| SPOT_SEQIMAGEEXPDURS | array of DWORD values, first value is the count |
| SPOT_SHUTTERMODE | short (SPOT_SHUTTERMODE_NORMAL, SPOT_SHUTTERMODE_OPEN, or SPOT_SHUTTERMODE_CLOSED) |
| SPOT_SUBTRACTBLACKLEVEL | BOOL |
| SPOT_TTLOUTPUTACTIVESTATE | short |
| SPOT_TTLOUTPUTDELAY | long |
| SPOT_TTLOUTPUTDELAYLIMITS | two long values, first value is minimum, second is maximum |
| SPOT_TTLOUTPUTDELAYMS | short |

| | |
|---|---|
| SPOT_VERTCLOCKVOLTAGEBOOST | short |
| SPOT_VERTSHIFTPERIOD | long |
| SPOT_VERTSHIFTPERIODS | array of long values, first value is count of values following |
| 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.

**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 one of the SPOT_WARN*xxx* warning values. 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 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_ERRDRVALREADYINIT | driver has already been initialized |
| SPOT_ERRNOCAMERAPOWER | The camera is not powered on |
| 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_ERRREGISTRYQUERY | error reading from the Windows registry |

**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 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 the SPOT_WARNUNSUPPCAMFEATURES warning value. This warning will generally indicate that the SpotCam driver software should be updated to take full advantage of the camera's features. If the proper input color profile cannot be found for the camera, SPOT_WARNINVALIDINPUTICC will be returned. The camera will be able to acquire color images, but color enhancement will not be available.

**See Also**

SpotExit

## 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 slider cameras which support this capability.

```
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 to 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. May be NULL.

**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.

*Windows and Linux:*

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

*MacOS:*

```
int WINAPI SpotRetrieveSequentialImage(
        void *pImageBuffer,
        unsigned long lRowBytes
);
```

**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.

*lRowBytes*

Specifies the number of image data buffers bytes per image row. *(MacOS only)*

**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. At this time, the image is ready, and SpotRetrieveSequentialImage can be called to retrieve it. Each call to this function will retrieve the next available image which has not yet been retrieved. 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.

**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**

This function should be called after SpotInit. 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

# SpotSetBusyCallback

The SpotSetBusyCallback function is used by the application to set a pointer to a callback function which the SpotCam driver will call periodically during long camera operations, giving the application opportunities to process system events and abort the operation.

```
void WINAPI SpotSetBusyCallback(
        SPOTBUSYCALLBACK pfnCallback,
        DWORD dwUserData
);
```

**Parameters**

*pfnCallback*

Points to a function of type SPOTBUSYCALLBACK, which the driver will call periodically during long camera operations. The callback should return FALSE to continue operation or TRUE to abort.

*dwUserData*

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

**Remarks**

The busy callback function provides an optional mechanism for handling system events and aborting during long camera operations. The registered callback function will be called about four times per second during long camera exposures.

# SpotSetCallback

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

```
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 or powered on or off.

```
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 not supported for all SPOT camera models. The device notification callback calls may be made from multiple threads, so the application should not make any assumptions about the thread in which the calls will be made.

# 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 an external shutter via SpotSetValue and the SPOT_ENABLETTLOUTPUT parameter. The active state of the TTL output is defined by the SPOT_TTLOUTPUTACTIVESTATE parameter. This function may not have any effect if it is called while the camera is busy performing another operation.

# 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_24BPPIMAGEBUFFERFORMAT
SPOT_AUTOEXPOSE
SPOT_AUTOGAINLIMIT
SPOT_BIASFRMSUBTRACT
SPOT_BINSIZE
SPOT_BITDEPTH
SPOT_BKGDIMAGESUBTRACT
SPOT_BRIGHTNESSADJ
SPOT_BRIGHTNESSADJX1000
SPOT_BUSBANDWIDTH
SPOT_CLEARMODE
SPOT_COLORBINSIZE
SPOT_COLORENABLE
SPOT_COLORENABLE2
SPOT_COLORORDER
SPOT_COLORRENDERINGINTENT
SPOT_COOLERMODEONEXIT
SPOT_COOLINGLEVEL
SPOT_CORRECTCHIPDEFECTS
SPOT_DEVICEUID
SPOT_DRIVERDEVICENUMBER
SPOT_ENABLEPOWERSTATECONTROL
SPOT_ENABLETTLOUTPUT
SPOT_ENHANCECOLORS
SPOT_EXPOSURE
SPOT_EXPOSURE2

SPOT_EXPOSUREADJ
SPOT_EXPOSUREADJX1000
SPOT_EXPOSURECOMPRECT
SPOT_EXPOSUREINCREMENT
SPOT_EXTERNALTRIGGERACTIVESTATE
SPOT_EXTERNALTRIGGERDELAY
SPOT_EXTERNALTRIGGERMODE
SPOT_FANEXPOSUREDELAYMS
SPOT_FANEXPOSURESPEED
SPOT_FANSPEED
SPOT_FLATFLDCORRECT
SPOT_FORCESINGLECHANLIVEMODE
SPOT_GAINPORTNUMBER
SPOT_HORIZREADOUTFREQUENCY
SPOT_HORIZREGULATEDTEMPERATURE
SPOT_IMAGEORIENTATION
SPOT_IMAGERECT
SPOT_IMAGETYPE
SPOT_INPUTCOLORPROFILE
SPOT_LIVEACCELERATIONLEVEL
SPOT_LIVEAUTOBRIGHTNESS
SPOT_LIVEAUTOGAINLIMIT
SPOT_LIVEBRIGHTNESSADJ
SPOT_LIVEBRIGHTNESSADJX1000
SPOT_LIVEENHANCECOLORS
SPOT_LIVEEXPOSURE
SPOT_LIVEGAINADJ
SPOT_LIVEGAINADJX1000
SPOT_LIVEGAMMAADJ
SPOT_LIVEGAMMAADJX1000
SPOT_LIVEHISTOGRAM
SPOT_LIVEIMAGESCALING
SPOT_LIVEMAXEXPOSUREMSEC
SPOT_LIVEPIXELRESOLUTIONLEVEL
SPOT_LIVESUBTRACTBLACKLEVEL
SPOT_MAXEXPOSUREMSEC
SPOT_MESSAGEENABLE
SPOT_MINEXPOSUREMSEC
SPOT_MONITORFILTERPOS
SPOT_NOISEFILTERTHRESPCT
SPOT_NUMBERBYTESPERIMAGEROW
SPOT_NUMBERSKIPLINES
SPOT_OUTPUTCOLORPROFILE
SPOT_PIXELRESOLUTIONLEVE
SPOT_PREAMPGAINVAL
SPOT_READOUTCIRCUIT
SPOT_REGULATETEMPERATURE
SPOT_REGULATETEMPERATURE
SPOT_RETURNRAWMOSAICDATA
SPOT_SENSORRESPONSEMODE
SPOT_SEQIMAGEDISKCACHEPATH
SPOT_SEQIMAGEEXPDURS
SPOT_SHUTTERMODE
SPOT_SUBTRACTBLACKLEVEL
SPOT_TTLOUTPUTACTIVESTATE
SPOT_TTLOUTPUTDELAY
SPOT_TTLOUTPUTDELAYMS
SPOT_VERTCLOCKVOLTAGEBOOST

SPOT_VERTSHIFTPERIOD
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

# SpotUpdateFirmware

The SpotUpdateFirmware function is used to update the firmware for all cameras and interface cards which provide this capability and for which firmware is available.

```
int WINAPI SpotUpdateFirmware(
        char *pszFirmwareFileName,
        DWORD dwControlFlags,
        DWORD *pdwResultFlags
);
```

**Parameters**

*pszFWPkgFileName*

Name of the firmware package file supplied by Diagnostic Instruments.

*dwControlFlags*

Bit-mapped flags to control the firmware update process.

*pdwResultFlags*

Points to bit-mapped flags which provide information about the results of the firmware update process.

**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_ERRCAMERABUSY | The camera is currently performing another operation. |
| SPOT_ERRCAMERAERROR | camera malfunction |
| SPOT_ERRDEVDRVLOAD | There was an error loading or opening the device driver. |
| SPOT_ERRFILEOPEN | The specified firmware package file cannot be opened or read. |
| SPOT_ERRINVALIDFILE | The specified file is not a valid firmware package file. |
| SPOT_ERRNOCAMERARESP | The camera is no longer connected or power is off. |

| | |
|---|---|
| SPOT_ERRNODEVICESFOUND | No firmware update-capable cameras or interface cards were found. |
| SPOT_ERRNOTHINGTODO | No firmware updates are needed. |
| SPOT_ERROUTOFMEMORY | out of memory. |

**Remarks**

Before calling this function, any other instances of the SpotCam driver should be unloaded. This function will attempt to load firmware into any currently connected devices for which newer firmware exists in the specified package file. An application can force a load of older firmware to the camera and/or interface card by setting the SPOT_UPDATEFWCONTROL_FORCEUPDATECAMERA and/or SPOT_UPDATEFWCONTROL_FORCEUPDATEINTFCARD bits, respectively, in the dwControlFlags argument. When one of these control flags is specified, the device will be loaded with the newest compatible firmware found in the package, even if it is older than what the device currently contains.

After a successful firmware update, the DWORD pointed to by pdwResultFlags may contain information regarding the results of the update. Refer to SpotCam.h for definitions of the various SPOT_UPDATEFWRESULT_xxx result flags. In all cases, SpotFindDevices should be called after update process completes to obtain a new device list.

# SpotWaitForStatusChange

The SpotWaitForStatusChange function is used to wait for a status notification from the SpotCam driver when operating in polling mode.

```
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 control application execution and to provide important status notifications during the operation. It should be called in a loop until the returned status value is SPOT_STATUSERROR, SPOT_STATUSIDLE, or SPOT_STATUSABORTED. Each time the function is called, driver execution will proceed until the next status change. 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. Because of the overhead involved, polling mode should preferably be used only in situations in which callbacks cannot be used.

# Appendix A – 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. The application's abort flag MUST be set to FALSE before calling any Spotcam function which does image acquisition (eg. SpotGetImage, SpotComputeExposure, etc.). Older versions of the SpotCam driver unintentionally reset the abort flag, but this behavior was corrected. If the application supplies the abort flag, it is solely responsible for setting and resetting it.
4. Unless an application is guaranteed that only one SPOT 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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_GAINVALS16 to determine which gain values are allowed at the different bit depths.
10. 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.
11. One SpotCam API acquisition function should not be called before another returns. For example, if SpotGetLiveImages has been called, and the abort flag has been set to TRUE, SpotGetImage should not be called until SpotGetLiveImages has returned. If it, or any other acquisition function is called, the SPOT_ERRCAMERABUSY error will be returned. This applies also to the SpotCompute*xxx* functions. If polling mode is enabled (SPOT_WAITFORSTATUSCHANGES is TRUE), and the abort flag is set to TRUE, SpotWaitForStatusChange must be called repeatedly until the returned status value is SPOT_STATUSIDLE, SPOT_STATUSABORTED, or SPOT_STATUSERROR.
12. Some SPOT cameras are capable of very long exposures, and therefore the SPOT_EXPOSURELIMITS2 parameter may return 0xffffffff as the upper exposure limit. If this happens, the SPOT_EXPOSURELIMITS parameter, which expresses the exposures in ms increments should be queried to determine the actual upper exposure duration limit.

# Appendix B – Revision History

## Version 4.7.5.10

1. Support has been added for the new Spot Idea Flex camera models.
2. A few problems affecting exposure computation with a minimum exposure setting have been fixed.
3. A problem which could cause SPOT_ERRNOCAMERARESP errors during sequential acquisition with Xplorer/Pursuit cameras under specific critical timing conditions when SpotGetSequentialImages was called with bDeferProcessing=FALSE has been fixed. This fix also involves device driver and firmware updates.
4. A bug which affected live mode auto-brightness under certain circumstances for USB cameras has been fixed.
5. A few bugs which affected sequential acquisition with external triggers, disk-caching, and synchronous mode has been fixed. The fix for

the USB cameras requires a firmware update.
6. Improvements have been made in flatfield correction for some Idea camera models.
7. A problem which affected acquisition from IEEE-1394 camera models when using TTL/Sync output and exposures longer than 200 ms has been fixed.
8. A bug which affected live mode with Xplorer/Pursuit color mosaic cameras has been fixed.
9. SpotGetLiveImage now correctly returns SPOT_ABORT.

## Version 4.7.5.3

1. A problem with auto-TTL output in IEEE-1394 cameras has been fixed.

## Version 4.7.5.2

1. A bug which caused incorrect values to be returned for SPOT_ACQUIREDLIVEIMAGESIZE and crashing when setting odd pixel boundaries with mosaic USB cameras has been fixed.
2. A bug which caused SpotGetBiasFrameCompatibilityInformation and SpotGetBackgroundImageCompatibilityInformation to fail has been fixed.
3. A bug which affected live mode readout with horizontal flipping for mosaic Xplorer/Pursuit cameras has been fixed.
4. A bug which affected live mode behavior with auto-brightness when specifying a sensor area for exposure computation on USB cameras has been fixed.

## Version 4.7.4.5

1. Support for the new 5 MP and 1.3 MP CMOS Idea camera models has been added.
2. The type of SPOTBUSYCALLBACK has been changed to BOOL.
3. A few minor bugs have been fixed.

## Version 4.7.4.5

1. Support for the new SPOT Xplorer/Pursuit mosaic camera models has been added.
2. A bug which caused crashing when SpotGetFlatfield was called for USB cameras has been fixed.
3. A bug which caused image brightness to be incorrect for USB cameras with SpotGetSequentialImages under certain circumstances has been fixed.
4. Several enhancements have been made in USB camera support.
5. A bug which occasionally caused errors (code 72) during acquisitions from Boost cameras has been fixed.
6. The firmware revision number for the Boost cameras are now returned properly by SpotGetVersionInfo2.
7. A couple of bugs which affected image brightness for live acquisition with intensity scaling have been fixed.

## Version 4.7.4

1. Support for SPOT camera models with USB interfaces has been added.
2. A new busy callback mechanism has been added. The SpotSetBusyCallback has been added to register the busy callback.
3. A new error code: SPOT_ERRNOCAMERAPOWER has been defined for SpotInit for use with some camera models.
4. A new disk-caching option has been added for sequential acquisition. The SPOT_SEQIMAGEDISKCACHEPATH parameter has been added for this feature.
5. A new multiple-exposure option for sequential acquisition has been added. The SPOT_MAXNUMBERSEQIMAGEEXPDURS and SPOT_SEQIMAGEEXPDURS parameters have been added for this feature.
6. A bug which caused SpotGetSequentialImages to return a SPOT_ERRNOCAMERARESP error when both live image scaling and acceleration were enabled has been fixed.
7. Sequential image acquisition for an infinite number of images and deferred processing is now supported.
8. The logic in SpotGetSequentialImages has been changed to allow acquired images to be retrieved, with deferred processing, after aborting.
9. A bug which occasionally caused exposure timestamps for Pursuit/Xplorer cameras to be incorrect has been fixed.
10. A bug which caused the value of SPOT_REGULATEDTEMPERATURE to default to zero instead of the proper value has been fixed.
11. A bug in the device driver (SpotKP.sys) which could occasionally cause SPOT_ERRNOCAMERARESP errors during sequential acquisition with Pursuit/Xplorer cameras has been fixed.
12. A bug which caused problems when SPOT_PIXELRESOLUTIONLEVEL was set to a non-zero value and binning was also enabled has been fixed.
13. A bug which occasionally caused SpotGetLiveImages to return SPOT_ERRNOCAMERARESP for Pursuit/Xplorer cameras in three-shot color mode when exposure or certain other parameters were changed has been fixed.

14. A bug which caused exposure computation to fail for RT SE-18 cameras has been fixed.
15. Support for a new sensor response mode: SPOT_SENSORRESPMODE_GLOWSUPPRESSSION has been added.
16. Support for Boost cameras has been improved.
17. A few issues which could occasionally result is errors when acquiring images from 1394 cameras have been resolved.
18. The delay between calls to SpotGetImage and the beginning of camera exposure has been reduced.
19. Multiple instances of the SpotCam DLL can no longer call SpotInit for the same camera. The first instance to initialize a camera must call SpotExit before another instance instance can initialize it.
20. A new parameter: SPOT_PIXELRESOLUTIONIMGSIZEFACTORS, has been added to allow applications to determine the approximate factor by which acquired image sizes will be affected for the pixel resolution setting.
21. A new parameter: SPOT_LIVEPIXELRESOLUTIONLEVEL, has been added to allow the setting of negative pixel resolution for live acquisition.
22. A new parameter: SPOT_COOLERMODEONEXIT, has been added to allow the specification of what should be done with the cooler upon shutdown.
23. A bug which caused the last column in reconstructed mosaic color images to be brighter or darker has been fixed.
24. A couple of bugs which affected live image histograms when auto-brightness control is enabled have been fixed.
25. A bug which could cause images acquired from IEEE-1394 cameras to be over-exposed has been fixed.

## Version 4.6.3

1. A bug which prevented background subtraction from being applied to images captured with SpotGetSequentialImages and Pursuit/Xplorer A change has been made to allow Pursuit/Xplorer cameras to do exposure/readout overlapping even when SpotGetSequentialImages is called with an interval of zero.
1. A new function: SpotUpdateFirmware, has been added for updating firmware for those camera models which support firmware updating.
2. A bug which caused SpotWaitForStatusChange to do an infinite wait when the nTimeOutMSec argument is zero has been fixed.
3. A bug which affected the color of higher resolution images has been fixed.
4. New parameters: SPOT_SENSORRESPONSEMODE and SPOT_SENSORRESPONSEMODES have been added to support the new sensor response mode setting for some SPOT Xplorer/Pursuit cameras.
5. A new parameter: SPOT_SENSORCLEARMODES has been added to provide a list of available sensor clear modes. It replaces SPOT_CLEARMODES.
6. The SPOT_CLEARMODE parameter has been renamed SPOT_SENSORCLEARMODE.
7. Live image scaling can now be disabled by setting the SPOT_LIVEIMAGESCALING parameter to NULL.
8. Live image histograms can now be disabled by setting the SPOT_LIVEHISTOGRAM parameter to NULL.
9. A bug which could cause crashing when SPOT_LIVEHISTOGRAM is TRUE for Boost cameras has been fixed.
10. A bug which caused crashing when processing images acquired from Pursuit and Xplorer cameras in certain cases has been fixed.
11. A bug which caused occasional crashes and image corruption with Flex cameras has been fixed.

## Version 4.6.2

1. Support for the new SPOT Flex monochrome and Boost cameras has been added.
2. The declarations for the SpotFindInterfaceCards, SpotOpenExternalShutter, and SpotCloseExternalShutter functions have been dropped although the functions are still defined in the SpotCam driver for legacy application support.
3. The parameter: SPOT_GAINVALS12 has been changed to SPOT_GAINVALS16, and SPOT_PORT0GAINVALS12 has been changed to SPOT_PORT0GAINVALS16.
4. SpotExit has been modified to return SPOT_ERRCAMERABUSY if the camera is currently busy.
5. A new option has been added to SpotGetSequentialImages to acquire images until aborted.
6. New API function: SpotGetCameraErrorCode has been added.
7. The image processing overhead has been reduced.
8. A bug which limited the effectiveness of flatfield correction for Xplorer and Pursuit camera has been fixed.
9. A bug which affected live mode auto-brightness for Xplorer and Pursuit SE cameras has been fixed.
10. A bug which could cause various functions to return SPOT_ERRNOCAMERARESP instead of SPOT_ABORT when aborted at certain points has been fixed.
11. A bug which caused the TTL output to be left active after acquisition of a color image from Pursuit and Xplorer cameras has been fixed.
12. Support for single-channel live mode readout has been added for many SPOT cameras. The SPOT_ATTR_SINGLECHANLIVEMODE attribute flag and the SPOT_FORCESINGLECHANLIVEMODE parameter have been added for this new capability.
13. Support for new live image histogram and scaling features have been added. The SPOT_LIVEHISTOGRAM and SPOT_LIVEIMAGESCALING parameters have been added for these features.
14. Support for a new line-skipping feature has been added to allow faster still image readout from the cameras which support it. The new parameters: SPOT_MAXNUMBERSKIPLINES and SPOT_NUMBERSKIPLINES have been added to support this feature.
15. New camera attribute bits: SPOT_ATTR_BLACKLEVELSUBTRACT, SPOT_ATTR_CHIPDEFECTCORRECTION,

SPOT_ATTR_EXPOSURESHUTTER, SPOT_ATTR_INTERNALSHUTTER, SPOT_ATTR_LIVEHISTOGRAM, SPOT_ATTR_MULTICHANLIVEMODE, SPOT_ATTR_SINGLECHANLIVEMODE, SPOT_ATTR_TTLOUTPUTDURINGEXPOSURE, and SPOT_ATTR_FIRMWAREUPDATE have been defined for use with SpotGetCameraAttributes.

16. Support has been added for TTL output activation during exposure for most SPOT camera models.
17. A new error code: IDS_IMAGETOOBRIGHT has been defined.
18. A new parameter: SPOT_ENABLEPOWERSTATECONTROL for enabling of disabling the time-consuming calling of OS power state functions for IEEE1394/FireWire cameras has been added.
19. New parameters: SPOT_PORT0GAINATTRIBUTES, SPOT_PORT1GAINATTRIBUTES, SPOT_PORT2GAINATTRIBUTES, and SPOT_PORT3GAINATTRIBUTES have been created for obtaining information about the camera's gain ports.
20. A new parameter: SPOT_IMAGESENSORMODELDESCR has been added for retrieving a text description of the camera's image sensor model.
21. A new parameter: SPOT_NUMBERBYTESPERIMAGEROW has been added for specifying the number of image data buffer bytes per image row to accommodate image buffers which are not of the default size, such as buffers with image borders.
22. New parameters: SPOT_COOLINGLEVEL, SPOT_COOLINGLEVELS, SPOT_FANEXPOSUREDELAYMS, SPOT_FANEXPOSURESPEED, SPOT_FANSPEED, SPOT_FANSPEEDS, SPOT_IMAGESENSORTYPE, SPOT_MAXVERTCLOCKVOLTAGEBOOST, SPOT_NUMBERREADOUTCIRCUITS, SPOT_PREAMPGAINVAL, SPOT_PREAMPGAINVALS, SPOT_READOUTCIRCUIT, SPOT_READOUTCIRCUITDESCR, SPOT_VERTCLOCKVOLTAGEBOOST, SPOT_VERTSHIFTPERIOD, and SPOT_VERTSHIFTPERIODS have been added to support features in new camera models.

## Version 4.5.9

1. A bug which caused incorrect values to be returned for SPOT_PORT0LIVEGAINVALS and Xplorer and Pursuit cameras has been fixed.
2. A few bugs which affected polling mode have been fixed.

## Version 4.5.7.2

1. A problem which could cause 1394 cameras to lock up while rapidly acquiring images under unusual circumstances has been fixed.
2. A bug which caused the driver to not support certain older SPOT RT monochrome cameras has been fixed.
3. A bug which caused image corruption when acquiring 24 bpp images with SPOT_24BPPIMAGEBUFFERFORMAT set to one of the SPOT_24BPPIMAGEBUFFERFORMAT*xxx* values other than SPOT_24BPPIMAGEBUFFERFORMATBGR and SPOT_ENHANCECOLORS set to TRUE has been fixed.
4. A problem which could cause still captured images to occasionally be brighter than live mode images with equivalent exposure for some SPOT RT and Insight PCI cameras has been corrected.
5. A bug which caused SpotWaitForStatusChange to return TRUE when no operation was pending has been fixed.
6. A problem which caused SpotWaitForStatusChange to occasionally return with a status of SPOT_STATUSIDLE, SPOT_STATUSABORTED, or SPOT_STATUSERROR while the camera was still considered busy has been fixed.
7. SpotGetBackgroundImage, SpotGetBiasFrame, SpotGetFlatfield, and SpotGetFlatfield2 now check if a file with the specified name can be created before acquiring any image data.

## Version 4.5.6

1. A bug which caused SpotGetValue to return an incorrect value for SPOT_MAXGAINPORTNUMBER has been fixed.
2. A bug which affected acquisitions and exposure and white balance computation on SPOT RT three-shot and older Insight three-shot cameras has been fixed.
3. A minor bug which caused the exposure value to be incorrectly copied to the dwExpDur member of the SPOT_EXPOSURE_STRUCT2 struct for SpotGetValue for monochrome acquisition from three-shot cameras has been fixed.
4. A minor bug which could leave Flex cameras in shifting mode for white balance computation, resulting in computation taking longer than necessary, has been fixed.
5. A minor problem with exposure computation which occasionally resulted in a SPOT_ERREXPTOOSHORT error being incorrectly returned has been fixed.
6. A bug which affected with IEEE-1394 Insight color filter cameras and some single-color exposures has been fixed.
7. The functionality of SpotGetLiveImages has been enhanced with a new option to allocate the image buffer in the SpotCam driver.
8. A bug which caused incorrect values to be returned for SPOT_ACQUIREDLIVEIMAGESIZE for mosaic cameras with no binning and an imaging area with an odd width or height has been fixed.
9. A bug which caused crashes in SpotGetLiveImages for mosaic cameras with no binning and an imaging area with an odd width or height has been fixed.
10. A bug which could cause serious problems when calling SpotGetBiasFrame or SpotGetBackgroundImage after SpotGetLiveImages has

been fixed.
11. A bug which caused problems when SPOT_WAITFORSTATUSCHANGES was set to TRUE has been fixed.
12. New parameters: SPOT_IMAGEORIENTATION and SPOT_24BPPIMAGEBUFFERFORMAT have been added to allow applications to specify the way in which image data should be stored.

## 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.