

SBIG Universal Driver/Library

Version 4.22
September 9, 2002

Santa Barbara Instrument Group
147A Castilian Drive
Santa Barbara, CA 93117
Phn: (805) 571-7244 Fax: (805) 571-1147
Web: <www.sbig.com> EMail: <sbig@sbig.com>

Table of Contents

1. INTRODUCTION 3

2. DRIVER INTERFACE..... 4

2.1 DRIVER RELATED COMMANDS..... 5

 2.1.1 *Open Driver - Command 17*..... 5

 2.1.2 *Close Driver - Command 18*..... 5

 2.1.3 *Open Device - Command 27*..... 5

 2.1.4 *Close Device - Command 28*..... 6

 2.1.5 *Get Driver Info - Command 10*..... 6

 2.1.6 *Get/Set Driver Handle – Commands 33/34*..... 6

2.2 EXPOSURE RELATED COMMANDS..... 7

 2.2.1 *Start Exposure - Command 1*..... 7

 2.2.2 *End Exposure - Command 2*..... 8

 2.2.3 *Start Readout – Command 35*..... 9

 2.2.4 *Readout Line - Command 3*..... 9

 2.2.5 *Read Subtract Line - Command 14*..... 10

 2.2.6 *Dump Lines - Command 4*..... 11

 2.2.7 *End Readout - Command 25*..... 11

 2.2.8 *Get Line - Command 31*..... 12

2.3 TEMPERATURE RELATED COMMANDS 12

 2.3.1 *Set Temperature Regulation - Command 5*..... 12

 2.3.2 *Query Temperature Status - Command 6*..... 14

2.4 EXTERNAL CONTROL COMMANDS..... 14

 2.4.1 *Activate Relay - Command 7*..... 14

 2.4.2 *Pulse Out - Command 8*..... 15

 2.4.4 *TX Serial Bytes - Command 19*..... 15

 2.4.5 *Get Serial Status - Command 20*..... 15

 2.4.6 *AO Tip Tilt - Command 21*..... 15

 2.4.7 *AO Set Focus - Command 22*..... 16

 2.4.8 *AO Delay - Command 23*..... 16

2.5 GENERAL PURPOSE COMMANDS..... 16

 2.5.1 *Establish Link - Command 9*..... 16

 2.5.2 *Get CCD Info - Command 11*..... 17

 2.5.3 *Get Turbo Status - Command 24*..... 18

 2.5.4 *Query Command Status - Command 12*..... 18

 2.5.5 *Miscellaneous Control - Command 13*..... 19

 2.5.6 *Update Clock - Command 15*..... 20

 2.5.7 *Read Offset - Command 16*..... 20

 2.5.8 *Get US Timer – Command 26*..... 20

 2.5.9 *Set/Get IRQL – Commands 29/30*..... 20

 2.5.10 *Get Link Status – Command 32*..... 20

 2.5.11 *Get Error String – Command 36*..... 21

 2.5.12 *Set Driver Control – Command 37*..... 21

 2.5.13 *Get Driver Control – Command 38*..... 21

 2.5.14 *USB AD Control – Command 39*..... 22

3. UTILITY PROGRAMS..... 22

 3.1 *SBIGDriverChecker.exe*..... 22

 3.2 *EthSim.exe*..... 23

 3.3 *SBIGUDRVJournalRx.exe*..... 23

 3.3 *SetClock.exe*..... 24

 3.5 *GetPortID.exe*..... 24

4. REVISION HISTORY 24

1. Introduction

This document describes the software interface to Santa Barbara Instrument Group's Universal Driver Library (SBIGUDRV). The SBIGUDRV driver supports all of SBIG's Parallel, Ethernet and USB based cameras and accessories.

The driver/library is available in many forms that support the 32 bit Windows 95¹/98/Me/NT/2000/XP environments as well as the older, and mostly obsolete, 16 bit DOS and Windows 3.X environments (Parallel based camera only). The distribution diskette contains the following files:

SBIGUDRV.H - Use this include file with all programs calling the driver. It includes the function prototype and many struct definitions for interfacing to the driver. You may need to set the TARGET variable based upon your target environment:

DOS, Win 3.1 – ENV_DOS

Win 95/98/ME/NT/2000/XP – ENV_WIN

\DOS\SBIGUDRV.LIB - Use this LIB file to link with your DOS and 16 bit Windows 3.X programs.

It only supports parallel port based cameras, as there is no system level support for USB and Ethernet. Note that you must use the large or huge model for your program as the driver was compiled using the large model where all functions and pointers are far.

\Tools – This directory contains some utility programs that will help you get your custom program up and running with the SBIG Universal Driver/Library. See section 3 for a description of each of these tools.

\Tools\SBIGDriverChecker.exe – This is a utility program that checks the drivers installed on a system against the latest drivers in the SBIG Drivers directory. A complete description is in Section 3.

\Tools\SBIG Drivers\SBIGUDRV.DLL - This is a 32 bit Windows DLL. It is built with the “stdcall” calling convention, which means it can be called from Visual C++ or Visual Basic.

\Tools\SBIG Drivers\SBIGUDRV.VXD - This is a protected mode low-level driver for 32 bit Windows 95/98/Me that is used in conjunction with the SBIGUDRV.DLL.

\Tools\SBIG Drivers\SBIGUDRV.SYS – This is a low-level driver for Windows NT/2000/XP that is used in conjunction with the SBIGUDRV.DLL.

\Tools\SBIG Drivers\SBIGUSBE.SYS, \Win\SBIG Drivers\SBIGULDR.SYS – These files are the low-level USB driver and firmware loader. Please refer the “Installing USB.PDF” Application Note for how to install these drivers.

\Tools\SBIG Drivers\SBIGUSBE.INF – This is the USB driver information file.

\Win\SBIGUDRV.LIB - This is an import library that you link with your 32 bit Windows program. Include this in your "project" file.

\Win\Visual C - Use the contents of this directory for Visual C++ Programs. Includes a sample Visual Studio 6 project.

\Win\Visual Basic – Use the contents of this directory for Visual Basic Programs. Includes some sample programs.

\Win\Delphi – Contains files from a user that got the previous version of the driver (PARDRV) to work under Delphi

\Docs – Contains this and supplemental documentation.

¹ There is no support for USB based cameras under Windows 95 or NT.

2. Driver Interface

The PC's software interface to the camera is through a LIB file that you link with your software. The model for the driver is a single external function that takes an integer command and pointers to command parameter and results structs. The memory allocation for these structs is the responsibility of the calling program. The driver acts upon the command and fills in the response. The C prototype for the function is shown in the SBIGUDRV.H file and a typical example is:

```
short SBIGUnivDrvCommand (short Command, void *Parameters, void *Results)
```

where Command is the Command to be executed, Parameters and Results are pointers to the structs. The function returns an error code indicating whether the camera was able to initiate or complete the command. The commands supported by the driver are grouped into the following sections discussed individually below (note that an enum type is defined in the SBIGUDRV.H file for the supported commands, parameter and results structures):

- Driver Related Commands
- Exposure Related Commands
- Temperature Related Commands
- External Control Commands
- General Purpose Commands

Getting back to the driver, it is written and documented assuming you are programming and proficient in C. As you can see, the function prototype is a C function, and as you will see the Parameters and Results parameters will end up being pointers to structs using the following elements within the structs:

LOGICAL - unsigned short (2 bytes) with 0 = FALSE and 1 = TRUE
enum - Enumerated unsigned short (2 bytes) with an allowed set of values
int - signed short (2 bytes)
uint - unsigned short (2 bytes)
long - signed long (4 bytes)
ulong - unsigned long (4 bytes)

Some commands don't require Parameters structs and some don't require Results structs. In those cases you should pass a NULL pointer to the driver.

Note that for compatibility with standard Windows API DLLs we changed the struct alignment to 8 bytes in the version 4.X drivers. Make sure your calling program uses 8 byte struct alignment.²

The supported commands are discussed in the sections below. For each command the Parameters and Results structs are shown except in the case where one or both do not exist. The function error return codes for each of the commands will vary from command to command (note a PAR_ERROR enum type is defined in the SBIGUDRV.H file for these return codes).

² DOS still uses 2 byte struct alignment

For each command, a command status is maintained internally by the driver, and can be monitored with the Query Command Status command. The command status for each of the commands varies from command to command but in general will be from one of the following:

- 0 = Idle
- 1 = Command In Progress

2.1 Driver Related Commands

The Commands in this section are used to open and close the driver and get driver related information. At the application level you must Open the Driver (allowing access to the top level of the driver) and then Open the Device (selecting the hardware interface) in order to communicate with the camera.

2.1.1 Open Driver - Command 17

The Open Driver command is used to initialize the driver and should be your first call to the driver. It takes no Parameters and returns no Results. Just pass NULL pointers to the Parameters and Results arguments of the SBIGUDrvCommand function when you call it.

2.1.2 Close Driver - Command 18

The Close Driver command is used to close the driver and should be your last call to the driver. There must be one call to Close Driver for each call to Open Driver. This command takes no Parameters and returns no Results. Just pass NULL pointers to the Parameters and Results arguments of the SBIGUDrvCommand function when you call it.

2.1.3 Open Device - Command 27

The Open Device command is used to load and initialize the low-level driver. You will typically call this second (after Open Driver).

Parameters Struct:

```
struct OpenDeviceParams {
    enum deviceType
        0 = unused
        1,2,3 – LPT 1,2,3
        0x7F00 – USB
        0x7F01 – Ethernet
    uint lptBaseAddress – for LPT1,2,3 base port address of the LPT port
    ulong ipAddress - for Ethernet the IP address of the camera/accessory
}
```

This command returns no Results. Just pass a NULL pointer to the Results arguments of the SBIGUDrvCommand function when you call it.

Notes:

- The **lptBaseAddress** is required for LPT1,2,3 under Windows 95/98/Me. This is typically 0x378 for LPT1 and 0x278 for LPT2 but can vary from machine to machine and can be found from the Device Manager control panel. Under Windows NT/2000/XP you can leave this set to 0 as the driver gets this information from the OS.

- The **ipAddress** is required for Ethernet. Use the four bytes of the long with the ms byte specifying the ms part of the address. For example if the desired IP address is 192.168.0.1 use 0xC0A80001 (0xC0=192, 0xA8 = 168, etc.)

2.1.4 Close Device - Command 28

The Close Driver command is used to close the low-level driver. You will typically call this second to last (right before Close Driver). There must be one call for Close Device for every call to Open Device.

The Close Device command takes no Parameters and returns no Results. Just pass NULL pointers to the Parameters and Results arguments of the SBIGUDrvCommand function when you call it.

2.1.5 Get Driver Info - Command 10

The Get Driver Info command is used to determine the version and capabilities of the DLL/Driver. For future expandability this command allows you to request several types of information. Initially the standard request and extended requests will be supported but as the driver evolves additional requests will be added.

Parameters Struct:

```
struct GetDriverInfoParams {
    enum request - type of driver information desired
        0 = standard request
        1 = extended request
        2,3, etc. - reserved for future expansion
}
```

Standard and Extended Results Struct:

```
struct GetDriverInfoResults0 {
    uint version - driver version in BCD with the format XX.XX
    char name[64] - driver name, null terminated string
    uint maxRequest - maximum request response available from this driver
}
```

The standard request returns the version and name information for the high level DLL and the extended request returns the version and name information for the low level VXD (Win 9X/Me) or SYS (Win NT/2000/XP)

2.1.6 Get/Set Driver Handle – Commands 33/34

The Get/Set Driver Handle commands are for use by applications that wish to talk to multiple cameras on various ports at the same time. If your software only wants to talk to one camera at a time you can ignore these commands.

The Get Driver Handle command takes a NULL Parameters pointer and a pointer to a GetDriverHandleResults struct for Results. The Set Driver Handle command takes a pointer to a SetDriverHandleParams struct for Parameters and a NULL pointer for Results. To establish links to multiple cameras do the following sequence:

- Call Open Driver for Camera 1
- Call Open Device for Camera 1
- Call Establish Link for Camera 1
- Call Get Driver Handle and save the result as Handle1

- Call Set Driver Handle with INVALID_HANDLE_VALUE in the handle parameter
- Call Open Driver for Camera 2
- Call Open Device for Camera 2
- Call Establish Link for Camera 2
- Call Get Driver Handle and save the result as Handle2

Then, when you want to talk to Camera 1, call Set Driver Handle with Handle1 and when you want to talk to Camera 2, call Set Driver Handle with Handle2. To shut down you must call Set Driver Handle, Close Device and Close Driver in that sequence for each camera.

Each time you call Set Driver Handle with INVALID_HANDLE_VALUE you are allowing access to an additional camera up to a maximum of four³ cameras. These cameras can be on different LPT ports, multiple USB cameras or at different Ethernet addresses. There is a restriction though due to memory considerations. You can only have a single readout in process at a time for all cameras. Readout begins with the Start Readout or Readout Line commands and ends with the End Readout command. If you try to do multiple interleaved readouts the data from the multiple cameras will be commingled. To avoid this, simply readout one camera at a time in an atomic process.

2.2 Exposure Related Commands

The commands in the section are used to initiate, complete or cancel an exposure in the camera. For each exposure the camera needs to be instructed to start the exposure, stop the exposure, and readout the image on a row-by-row basis.

2.2.1 Start Exposure - Command 1

The Start Exposure command is used to initiate an exposure. The PC specifies the exposure time, etc. and then monitors the exposure's progress with the Query Command Status command discussed below.

Parameters Struct:

```
struct StartExposureParams {
    enum ccd - the CCD to use in the exposure
        0 = Imaging CCD
        1 = Tracking CCD
    ulong exposureTime - integration time in hundredths of a second
    enum abgState - antiblooming gate state during integration
        0 = Low during integration (ABG shut off)
        1 = Clocked at low rate
        2 = Clocked at medium rate
        3 = Clocked at high rate
    enum openShutter - 0=Leave Shutter alone, 1=Open Shutter for Exposure and Close for
        Readout, 2=Close Shutter for Exposure and Readout
}
```

The status for this command (from the Query Command Status Command) consists of the following two 2-bit fields:

³ DOS and Windows 3.1 only support 1 camera due to memory considerations.

b₁b₀ = Imaging CCD Status, 00 - CCD Idle, 10=In Progress, 11=Complete

b₃b₂ = Tracking CCD Status, 00 - CCD Idle, 10=In Progress, 11=Complete

Notes:

- For the ST-7/ST-8/ST-9/ST-10/ST-1K add `START_SKIP_VDD` to the **ccd** parameter to increase the image rep rate. This bypasses the time consuming reduction of the CCDS Vdd which is normally used to reduce the readout amplifier glow for the imaging CCD. You'll get a glow in the upper-left corner of the Imaging CCD but the readout rep rate will be higher. SBIG uses this in the Turbo focus mode.
- For the ST-7/ST-8/ST-9/ST-10/ST-1K the minimum allowable exposure is `MIN_ST7_EXPOSURE` (.12 seconds) when the **openShutter** item is 1. If you ask the driver to make a shorter exposure it will take a .12 second exposure.
- The maximum exposure is 655.35 seconds for Tracking CCD and 167,777.16 seconds for Imaging CCD.
- The **abgState** only affects the Tracking CCD of the ST-7/8 and the Imaging CCD of the PixCel255.
- For the PixCel255, PixCel237 and ST-1K you need to specify the Imaging CCD since the camera is not a dual CCD design.
- For the PixCel255 and PixCel237 the **openShutter** parameter is ignored and should be set to 0. Use the Pulse Out Control command to position the Vane/Filter Wheel.

2.2.2 End Exposure - Command 2

The End Exposure command is used after the integration is complete to prepare the CCD for readout or to terminate an exposure prematurely.

Parameters Struct:

```
struct EndExposureParams {
    enum ccd - the CCD to end the exposure
        0 = Imaging CCD
        1 = Tracking CCD
}
```

Notes:

- The End Exposure command must be called at least once for each Start Exposure command issued. Several End Exposure commands can be issued without generating an error.
- For the ST-7/8/9/10/1K the End Exposure command prepares the CCD for readout. This normally involves delaying a period of time waiting for the shutter motor to turn off. You can tell the driver to skip this delay by adding `END_SKIP_DELAY` to the **ccd** enum item in order to increase the image rep rate, but you should do this only when the shutter didn't move for both the light and dark images. This means you issued the Start Exposure command with the **openShutter** item set to 0 (leave shutter alone) for the light image and with the **openShutter** item set to 2 (shutter closed for integration and readout) for the dark frame. This scenario only occurs when you are using the Tracking CCD while the Imaging CCD is integrating.
- With the PixCel255, PixCel237 and ST-1K the **ccd** parameter should be set to 0 for the Imaging CCD.

2.2.3 Start Readout – Command 35

The Start Readout command is used to inform the driver about the area you intend to readout in subsequent calls to the Readout Line or Read Subtract Line commands. Calling this command is optional (but suggested) and optimizes the readout throughput for small areas on USB and Ethernet based cameras.

Parameters Struct:

```
struct StartReadoutParams {
    enum ccd - the CCD that will be read out
        0 = Imaging CCD
        1 = Tracking CCD
    enum readoutMode - binning mode utilized during readout
        0 = No binning, high resolution
        1 = 2x2 on-chip binning, medium resolution
        2 = 3x3 on-chip binning, low resolution (ST-7/8/9/10/1K/237 only)
        xx03 = Nx1 on-chip binning (ST-7/8/9/10/1K only)
        xx04 = Nx2 on-chip binning (ST-7/8/9/10/1K only)
        xx05 = Nx3 on-chip binning (ST-7/8/9/10/1K only)
        6 = No binning, high resolution (ST-7/8/9/10/1K only)
        7 = 2x2 binning with vertical binning off-chip (ST-7/8/9/10/1K only)
        8 = 3x3 binning with vertical binning off-chip (ST-7/8/9/10/1K only)
        9 = 9x9 binning (ST-7/8/9/10/1K only)
    uint top – topmost row to readout (0 based)
    uint left – left most pixel to readout (0 based)
    uint height – image height
    uint width – image width
}
```

Notes:

- See the notes for the Readout Line command.
- The Start Readout command does not actually readout any pixels. It just tells the driver which pixels you will readout using the Readout Line command.
- The Start Readout command does discard **top** lines from the CCD. You do not need to call Dump Lines prior to calls to Readout Line after using the Start Readout Command.
- Even though you specify **left** and **width** parameters with this command you must pass the same values in the **pixelStart** and **pixelLength** parameters in subsequent calls to the Readout Line command.

2.2.4 Readout Line - Command 3

The Readout Line command is used to digitize some or all of the active pixels in a row.

Parameters Struct:

```
struct ReadoutLineParams {
    enum ccd - the CCD to readout
        0 = Imaging CCD
        1 = Tracking CCD
```

```

enum readoutMode - binning mode utilized during readout
    see Start Readout command above
uint pixelStart - left most pixel to readout
uint pixelLength - number of pixels to digitize
}

```

Results Struct:

Rather than passing a pointer to a Results struct, pass a pointer to the destination array of unsigned short integers where the Readout Line command should place the digitized pixel data.

Notes:

- Any arbitrary region can be readout using the Dump Lines and Readout Line commands by varying the **pixelStart** and **pixelLength** parameters.
- Interrupts are disabled for the duration of the line readout. You may want to use the Update Clock command to resynchronize the DOS clock after reading out an image.
- The PixCel255 and the Tracking CCDs only support the 1x1 and 2x2 binning modes. The 3x3 binning mode is supported by the Imaging CCD and by the PixCel237 only.
- With the PixCel255, PixCel237 and ST-1K the **ccd** parameter needs to be set to 0 for the Imaging CCD.
- When binning modes are used, the **pixelStart** and **pixelLength** parameters are in terms of binned pixels.
- You can get the dimensions of the camera's CCD(s) using the Get CCD Info command.
- The Nx1, Nx2 and Nx3 binning modes of the ST-7/8/9/10/1K support variable binning (N=1 thru 255) in the vertical direction. You specify the amount of vertical binning in the most significant byte of the readout mode.
- The 1x1, 2x2 and 3x3 off-chip binning modes offer non-streaked horizontal readout for non-Antiblooming versions of the ST-7/8/9/10/1K. These detectors bloom both horizontally and vertically under saturating conditions and these readout modes remove the horizontal blooming. They are not necessary with the antiblooming versions of these cameras and the standard on-chip readout modes can be used.
- Readout mode 9 with 9x9 binning is roughly 3 times faster than 3x3 mode and is intended for a fast Focus/Center mode.

2.2.5 Read Subtract Line - Command 14

The Read Subtract Line command is identical to the Readout Line command except that it subtracts the data that is stored in memory prior to the readout from the readout data. The Data stored in the array is:

$$\text{Data}[n] = \text{CCD}[n] - \text{Data}[n] + 100$$

The subtraction adds a bias of 100 to prevent the data from clipping and makes sure the data doesn't overflow or underflow.

Parameters Struct:

```

struct ReadoutLineParams {
    enum ccd - the CCD to readout
        0 = Imaging CCD
        1 = Tracking CCD
}

```

```

enum readoutMode - binning mode utilized during readout
    see Start Readout command above
uint pixelStart - left most pixel to readout
uint pixelLength - number of pixels to digitize
}

```

Results Struct:

Rather than passing a pointer to a Results struct, pass a pointer to the destination array of unsigned short integers where the Read Subtract Line command should place the digitized pixel data.

Notes:

- See the notes for the Readout Line command.
- The data is subtracted in place. The Read Subtract command digitizes a pixel, subtracts the value in the destination array, adds 100 counts to avoid clipping at 0 and then stores that result in the destination array.

2.2.6 Dump Lines - Command 4

The Dump Lines command is used to discard all of the active pixels in a row on the CCD. You would use this for example when a partial frame readout is desired to discard lines above a desired region.

Parameters Struct:

```

struct DumpLinesParams {
    enum ccd - the CCD to dump lines
        0 = Imaging CCD
        1 = Tracking CCD
    enum readoutMode - binning mode utilized during readout
        see Start Readout command above
    uint lineLength - number of lines to dump
}

```

Notes:

- See the notes for the Readout Line command.
- Unused rows of pixels can be dumped faster than they can be read out. Using the Dump Lines command for sub-array readout can speed up image throughput.

2.2.7 End Readout - Command 25

The End Readout command is used after readout of the CCD is complete to prepare the CCD for the idle state.

Parameters Struct:

```

struct EndReadoutParams {
    enum ccd - the CCD to end the exposure
        0 = Imaging CCD
        1 = Tracking CCD
}

```

Notes:

- The End Readout command should be called at least once per readout after calls to the Readout Line, Read Subtract Line or Dump Lines command are complete. Several End Readout commands can be issued without generating an error.

- For the ST-7/8/9/10/1K the End Readout command prepares the CCD for the idle state. This normally involves turning off the CCD preamp and unfreezing the TE cooler if Auto TE Freeze mode has been enabled (see the Set Temperature Regulation command).
- For the other cameras (255, 237) the End Readout Command does nothing at the current time. For future compatibility you should call this command at the end of the readout phase.
- With the PixCel255, PixCel237 and ST-1K the **ccd** parameter should be set to 0 for the Imaging CCD.

2.2.8 Get Line - Command 31

The current driver does not use this command. It was added in a previous version and never removed. It could be reassigned in the future.

2.3 Temperature Related Commands

The commands in this section are used to program or monitor the CCD's temperature regulation. Note that the cameras contain two temperature-sensing thermistors, one in the housing measuring the ambient temperature on one on the CCD.

2.3.1 Set Temperature Regulation - Command 5

The Set Temperature Regulation command is used to enable or disable the CCD's temperature regulation.

Parameters Struct:

```
struct SetTemperatureRegulationParams {
    enum regulation - 0=regulation off, 1=regulation on, 2=regulation override,
                    3=freeze te cooler, 4=unfreeze te cooler,
                    5=enable auto freeze, 6=disable auto freeze
    uint ccdSetpoint - CCD temperature setpoint in A/D units if regulation on or TE drive level (0-
                    255 = 0-100%) if regulation override
}
```

Notes:

- The setpoint above is in A/D units. To convert from temperature in °C to A/D setpoint units and to convert the thermistor readings from the Query Temperature Status command to °C use the following formulas, noting that the CCD thermistor and Ambient thermistor require different constants:

$T_0 = 25.0$	$R_0 = 3.0$
$MAX_AD = 4096$	
$R_RATIO_{CCD} = 2.57$	$R_RATIO_{Ambient} = 7.791$
$R_BRIDGE_{CCD} = 10.0$	$R_BRIDGE_{Ambient} = 3.0$
$DT_{CCD} = 25.0$	$DT_{Ambient} = 45.0$

Calculation of Setpoint from Temperature T in °C

$$r = R_0 \times e^{\left(\frac{\ln(R_RATIO) \times (T_0 - T)}{DT}\right)}$$

$$setpoint = \frac{MAX_AD}{\left(\frac{R_BRIDGE}{r} + 1.0\right)}$$

Calculation of Temperature T in °C from Setpoint

$$r = \frac{R_BRIDGE}{\left(\frac{MAX_AD}{setpoint} - 1.0\right)}$$

$$T = T_0 - DT \times \left(\frac{\ln\left(\frac{r}{R_0}\right)}{\ln(R_RATIO)}\right)$$

- *What the heck is the "Freeze" all about?* This only pertains to ST-7/8/9/10/1K cameras and freezing the TE cooler means telling the temperature regulation circuitry in the camera to keep the TE cooler power at the same level it's currently at until we come back later and unfreeze it. It's essentially a way of telling the camera to be very "quite" for a period of time. What you do with the freeze commands is use them to freeze the TE cooler for readout and unfreeze it when you're done with the readout. This insures the absolute lowest noise readout possible. One thing you *don't* want to do is keep the TE frozen for a long period of time if you're not reading out the CCD because then you'll degrade the performance of the temperature regulation circuitry.

To take advantage of the freeze feature you first tell the camera to enable temperature regulation with the *regulation* item set to 1 and the *setpoint* item set to the desired setpoint temperature, just like you normally would. This allows the camera to achieve regulation at the setpoint temperature. Then, just before you start the readout you call this command with the *regulation* item set to 3 (the *setpoint* does not matter). After readout call this command with the *regulation* item set to 4 (again, *setpoint* doesn't matter). Don't forget to unfreeze the TE or you'll get poor temperature regulation.

You can also get the camera to automatically freeze and unfreeze the TE cooler for you by calling this command with the *regulation* item set to 5. After doing this the driver will freeze the TE at the first sign of readout and unfreeze it when you call the End Readout command. If you use the auto-freeze feature don't forget that call to End Readout. Finally to disable the auto freeze function call this command with the *regulation* item set to 6.

Finally, you can query whether the TE is frozen by logically anding the *enabled* item of **Query Temperature Status** command results with the REGULATION_FROZEN_MASK from

the SBIGUDRV.H header. If it's set the TE is currently frozen (either manually or by the auto-freeze feature).

2.3.2 Query Temperature Status - Command 6

The Query Temperature Status command is used to monitor the CCD's temperature regulation.

Results Struct:

```
struct QueryTemperatureStatusResults {
    LOGICAL enabled - temperature regulation is enabled when this is TRUE
    uint ccdSetpoint - CCD temperature or thermistor setpoint in A/D units
    uint power - this is the power being applied to the TE cooler to maintain temperature regulation
    and is in the range 0 thru 255
    uint ccdThermistor - this is the CCD thermistor reading in A/D units
    uint ambientThermistor - this is the ambient thermistor reading in A/D units
}
```

Notes:

- Refer to the Set Temperature Regulation command for the formula to convert between A/D units and degrees C for the thermistor readings.
- You can query whether the TE is frozen (see the Set Temperature Regulation command) by logically anding the *enabled* item of the results with the REGULATION_FROZEN_MASK from the SBIGUDRV.H header. If it's set the TE is frozen.

2.4 External Control Commands

The commands in this section are used to control the telescope position through the telescope interface or to position the CFW-6A motorized color filter wheel.

2.4.1 Activate Relay - Command 7

The Activate Relay command is used to activate one or more of the telescope control outputs or to cancel an activation in progress.

Parameters Struct:

```
struct ActivateRelayParams {
    uint tXPlus - x plus activation duration in hundredths of a second
    uint tXMinus - x minus activation duration in hundredths of a second
    uint tYPlus - y plus activation duration in hundredths of a second
    uint tYMinus - y minus activation duration in hundredths of a second
}
```

The status for this command (from the Query Command Status Command) consists of the following four bit field:

b₃ = +X Relay, 0=Off, 1= Active
 b₂ = -X Relay, 0=Off, 1= Active
 b₁ = +Y Relay, 0=Off, 1= Active
 b₀ = -Y Relay, 0=Off, 1= Active

Notes:

- This command can be used to cancel relay activations by setting the appropriate parameters to 0.

2.4.2 Pulse Out - Command 8

The Pulse Out command is used with the ST-7/8/9/10/1K to position the CFW-6A/CFW-8 and with the PixCel255 and PixCel237 to position the internal vane/filter wheel.

Parameters Struct:

```
struct PulseOutParams {
    uint numberPulses - number of pulses to generate (0 thru 255)
    uint pulseWidth - width of pulses in units of microseconds with a minimum of 9 microseconds
    uint pulsePeriod - period of pulses in units of microseconds with a minimum of 29 plus the
        pulseWidth microseconds
}
```

The status for this command (from the Query Command Status command) consists of the following bit fields:

b₀ - Normal status, 0 = inactive, 1 = pulse out in progress
 b₁-b₃ - PixCel255/237 Filter state, 0=moving, 1-5=at position 1-5, 6=unknown

Notes:

- The camera will cease communications while the Pulse Out command is in progress to maintain the best pulse width accuracy. After sending the ACK response the camera will generate the pulses and only when it has finished generating the pulses will it respond to further communications from the PC.
- With the PixCel255/237 for positioning the internal vane/filter wheel you set the numberPulses parameter to a non-zero value (typically 1), the pulseWidth to zero and the pulsePeriod to one of the following values: 0=Leave vane/filter alone, 1-5=Position vane/filter wheel at position 1 thru 5, 6=Stop motor, abort any move in progress, 7=initialize and identify vane/filter wheel.
- On the PixCel255/237 the following filter positions are defined: Position 1 = Clear/Open, Position 2 = Opaque, Position 3 = Red, Position 4 = Green and Position 5 = Blue. Positions 1 and 2 are supported by the vane and positions 1 thru 5 are supported by the filter wheel.
- You find out what type of filter wheel is installed in the PixCel255/237 using the Get CCD Info command with request number 3.

2.4.4 TX Serial Bytes - Command 19

The TX Serial Bytes command is for internal use by SBIG. It's a very low level version of commands like AO Tip Tilt that are used to send data out the ST-7/8's telescope port to accessories like the AO-7. There's no reason why you should need to use this command. Just use the dedicated commands like AO Tip Tilt.

2.4.5 Get Serial Status - Command 20

The Get Serial Status command is for internal use by SBIG. It's a very low level version of commands like AO Tip Tilt that are used to send data out the ST-7/8's telescope port to accessories like the AO-7. There's no reason why you should need to use this command. Just use the dedicated commands like AO Tip Tilt.

2.4.6 AO Tip Tilt - Command 21

The AO Tip Tilt Command is used to position an AO-7 attached to the telescope port of an ST-7/8.

Parameters Struct:

```
struct AOTipTiltParams {
    uint xDeflection - this is the desired position of the mirror in the X axis
    uint yDeflection - this is the desired position of the mirror in the X axis
}
```

Notes:

- The range for the X and Y deflection parameters are 0 through 4095. The mirror is centered at 2048, fully to one side at 0 and fully at the other side with 4095.

2.4.7 AO Set Focus - Command 22

This command is reserved for future use with motorized focus units. Prototypes of the AO-7 had motorized focus but the feature was removed in the production units. This command is a holdover from that.

2.4.8 AO Delay - Command 23

The AO Delay Command is used to generate millisecond type delays for exposing the Tracking CCD.

Parameters Struct:

```
struct AODelayParams {
    ulong delay - this is the desired delay in microseconds
}
```

Notes:

- The computer essentially hangs while waiting for this delay to expire so be careful how you use this command.

2.5 General Purpose Commands

The commands discussed in this section are general-purpose commands which do not fall into one of the groups discussed above. They are used by the PC to interrogate the driver and camera.

2.5.1 Establish Link - Command 9

The Establish Link command is used by the PC to establish a communications link with the camera. It should be used before any other commands are issued to the camera (excluding the Get Driver Info command).

Parameters Struct:

```
struct EstablishLinkParams {
    uint sbigUseOnly – leave set to 0
}
```

Results Struct:

```
struct EstablishLinkResults {
    enum cameraType - constant specifying the type of camera as specified by the CAMERA_TYPE
    enum
}
```

Notes:

- The EstablishLinkParams struct in was modified in version 4 of the driver and no longer specifies the LPT base address. This data is now supplied to the driver through the Open Device command.
- When establishing a link to an ST-237A the **cameraType** is reported as an original ST237_CAMERA. This was done for maximum compatibility with existing 3rd party software packages. The way you distinguish an ST-237A (16 bit A/D) from the ST-237 (12 bit A/D) is by checking the **gain** item from the Get CCD Info command response. If the gain is less than 1.0 (0x100) you are talking to an ST-237A.

2.5.2 Get CCD Info - Command 11

The Get CCD Info command is used by the PC to determine the model of camera being controlled and its capabilities. For future expandability this command allows you to request several types of information. Currently 6 standard requests are supported but as the driver evolves additional requests will be added.

Parameters Struct:

```
struct GetCCDInfoParams {
    enum request - type of CCD information desired
        0 = standard request for Imaging CCD
        1 = standard request for Tracking CCD
        2 = extended request for Camera Info
        3 = extended request for PixCel255/237 Camera Info
        4 = secondary extended request for Imaging CCD
        5 = secondary extended request for Tracking CCD
        6,7, etc. - reserved for future expansion
}
```

Standard Results Struct :

requests 0 and 1 -

```
struct GetCCDInfoResults0 {
    uint firmwareVersion - version of the firmware in the resident microcontroller
    enum cameraType - constant specifying the type of camera, (see CAMERA_TYPE enum in
        SBIGUDRV.H)
    char name[64] - null terminated string containing the name of the camera
    uint readoutModes - number of readout modes supported
    struct readoutInfo[20] {
        uint mode - readout mode to pass to the Readout Line command
        uint width - width of image in pixels
        uint height - height of image in pixels
        uint gain - a four digit BCD number specifying the amplifier gain in e-/ADU in the
            XX.XX format.
        ulong pixelWidth - a eight digit BCD number specifying the pixel width in microns in the
            XXXXXX.XX format.
        ulong pixelHeight - a eight digit BCD number specifying the pixel height in microns in the
            XXXXXX.XX format.
    }
}
```

request 2 -

```

struct GetCCDInfoResults2 {
    uint badColumns - number of bad columns in imaging CCD
    uint columns[4] - bad columns
    enum imagingABG - type of Imaging CCD, 0= No ABG Protection, 1 = ABG Present
    char serialNumber[10] - null terminated serial number string
}

```

request 3 - For the PixCel255/237

```

struct GetCCDInfoResults3 {
    enum adSize - 0 = Unknown, 1 = 12 bits, 2 = 16 bits
    enum FilterType - 0 = Unknown, 1 = External, 2 = 2 Position, 3 = 5 Position
}

```

requests 4 and 5 - For all cameras

```

struct GetCCDInfoResults4 {
    uint capabilitiesBits – Set of bits for additional capabilities:
        b0:    0 = CCD is Full Frame Device, 1 = CCD is Frame Transfer Device
    uint dumpExtra – Number of unbinned rows to dump to transfer image area to
                    storage area
}

```

Notes:

- The ST-7/8/9/10/1K supports types 0, 1, 2, 4 and 5 requests. The PixCel255/237 supports request types 0, 3, 4 and 5.
- A zero in the height field of the readoutInfo struct signifies the xN mode vertically.
- Mode 8 with 9x9 binning on some sensors (like the ST-1K) will report the maximum pixel height/width of 99.99 even though the binned pixels are actually larger. If you need the correct pixel size multiply the pixel size from mode 0 by the binning factor.
- Requests 4 and 5 are for use when you bypass **the Start Exposure/End Exposure** commands for reading out the CCD such as when you are doing millisecond type exposures with the Tracking CCD and an AO-7. Normally the **End Exposure** command handles transferring the imaging area and dummy rows of these CCDs to the storage area but when you bypass it you must provide for that transfer by adding the number specified in the *dumpExtra* field to the amount of lines you want to discard with the **Dump Lines** command. If you don't supplement the number of lines to dump you will be digitizing data from the storage area of the CCD and won't see any star images at all.
- See the notes for the Establish Link command above.

2.5.3 Get Turbo Status - Command 24

The current driver does not use this command. It was added in a previous version and never removed. It could be reassigned in the future.

2.5.4 Query Command Status - Command 12

The Query Command Status command is used to monitor the progress of a previously requested command. Typically this will be used to monitor the progress of an exposure, relay closure or CFW-6A move command.

Parameters Struct:

```
struct QueryCommandStatusParams {
    uint command - command of which the status is desired
}
```

Results Struct:

```
struct QueryCommandStatusResults {
    uint status - command status
}
```

2.5.5 Miscellaneous Control - Command 13

The Miscellaneous Control command is used to control the Fan, LED, and shutter. The camera powers up with the Fan on, the LED on solid, and the shutter closed. The driver flashes the LED at the low rate while the Imaging CCD is integrating, flashes the LED at the high rate while the Tracking CCD is integrating and sets if on solid during the readout.

Parameters Struct:

```
struct MiscellaneousControlParams {
    LOGICAL fanEnable - set TRUE to turn on the Fan
    enum shutterCommand - 0=leave shutter alone, 1=open shutter, 2=close shutter, 3=reinitialize shutter
    enum ledState - 0=LED off, 1=LED on, 2=LED blink at low rate, 3=LED blink at high rate
}
```

The status for this command (from the Query Command Status Command) consists of the following bit fields:

- b7-b0 - Shutter edge - This is the position the edge of the shutter was detected at for the last shutter move. Normal values are 7 thru 9. Any other value including 255 indicates a shutter failure and the shutter should be reinitialized.
- b8 - the Fan is enabled when this bit is 1
- b10b9 - Shutter state, 0=open, 1=closed, 2=opening, 3=closing
- b12b11 - LED state, 0=off, 1=on, 2=blink low, 3=blink high

Notes:

- The ST-7/8/9/10/1K have a shutter, LED and fan but no filter wheel. To position the CFW-6/8 attached to these cameras use the Pulse Out command.
- The PixCel255 has no shutter, fan control or LED but does have a vane/filter wheel. The settings of the fanEnabled, shutterCommand and ledState parameters are ignored and should be set to 0.
- The PixCel237 has no shutter or LED but does have a fan that can be controlled (turned on and off) and vane/filter wheel. The settings of the shutterCommand and ledState parameters are ignored and should be set to 0. The fanEnable is used to control the fan.
- The ST-7/8/9/10/1K will cease communications with the PC while the reinitialize shutter command is in progress. After the camera responds with the command ACK the PC should not send any further commands to the camera for up to 5 seconds after issuing a reinitialize shutter command.

2.5.6 Update Clock - Command 15

The Update Clock command is used to resynchronize the DOS clock after image readout. Since the Readout Line command disables interrupts for the duration of the command, the DOS clock can miss several Tick interrupts which occur at 18.2 times per second. The Update Clock command reads the Real Time Clock chip and resynchronizes the DOS clock. Typically you would do this after you have read out the last line in an image.

The Update Clock command takes no Parameters and returns no Results. Just pass NULL pointers to the Parameters and Results arguments of the SBIGUDrvCommand function when you call it.

2.5.7 Read Offset - Command 16

The Read Offset command is used to measure the CCD's offset. In the SBIG cameras the offset is adjusted at the factory and this command is for testing or informational purposes only.

Parameters Struct:

```
struct ReadOffsetParams {
    enum ccd - the CCD to measure offset
        0 = Imaging CCD
        1 = Tracking CCD
}
```

Results Struct:

```
struct ReadOffsetResults {
    uint offset - the CCD's offset
}
```

2.5.8 Get US Timer – Command 26

This command is of extremely limited (and unknown) use. When you have established a link to a parallel port based camera under Windows NT/2000/XP this command returns a counter with 1 microsecond resolution. Under all other circumstances the counter is zero.

2.5.9 Set/Get IRQ – Commands 29/30

This command allows you to control the IRQ priority of the driver under Windows NT/2000/XP. The default settings should work fine for all users and these commands should not need to be used.

We use three settings in our CCDOPS software: High = 27, Medium = 15, Low = 2. Under fast machines Low will work fine. On slower machines the mouse may get sluggish unless you select the Medium or High priority.

2.5.10 Get Link Status – Command 32

This command returns the status for communications link established with the camera.

Results Struct:

```
struct {
    MY_LOGICAL linkEstablished – TRUE when a link has been established
    uint baseAddress – base address of the LPT port
    uint cameraType – CAMERA_TYPE enum
    ulong comTotal – total number of communications with camera
    ulong comFailed – total number failed communications with camera
}
```

2.5.11 Get Error String – Command 36

This command returns a null terminated C string in English (not Unicode) corresponding to the passed error number. It's handy for reporting driver level errors to the user.

2.5.12 Set Driver Control – Command 37

This command is used to modify the behavior of the driver by changing the settings of one of the driver control parameters. Many driver options can be enabled or disabled with this command.

Parameters Struct:

```
struct SetDriverControlParams {
    enum controlParameter - the parameter to modify
        (see the DRIVER_CONTROL_PARAM enum)
    long controlValue – the value of the control parameter
}
```

Notes:

- The DCP_USB_FIFO_ENABLE parameter defaults to TRUE and can be set FALSE to disable the FIFO and associated pipelining in the USB cameras. You would do this for example in applications using TDI where you don't want data in the CCD digitized until the actual call to ReadoutLine is made.
- The DCP_CALL_JOURNAL_ENABLE parameter defaults to FALSE and can be set to TRUE to have the driver broadcast Driver API calls. These broadcasts are handy as a debug tool for monitoring the sequence of API calls made to the driver. The broadcasts can be received and displayed with the SBIGUDRVJournalRx.exe application.
Only use this for testing purposes and do not enable this feature in your released version of your application as the journaling mechanism can introduce minor artifacts in the readout.
- DCP_IVTOH_RATIO parameter is for SBIG internal use only at this time

2.5.13 Get Driver Control – Command 38

This command is used to Query the setting of one of the driver control parameters.

Parameters Struct:

```
struct GetDriverControlParams {
    enum controlParameter - the parameter to modify
        (see the DRIVER_CONTROL_PARAM enum)
}
```

Results Struct:

```
struct GetDriverControlResults {
    long controlValue – the value of the control parameter
}
```

Notes:

- See the Set Driver Control command above.

2.5.14 USB AD Control – Command 39

This command is used to modify the USB cameras A/D gain and offset registers.

Parameters Struct:

```
struct USBADControlParams {
    enum command – Imaging or Tracking CCD Gain or Offset
        (see the USB_AD_CONTROL_COMMAND enum)
    int data – command specific
}
```

Notes:

- This command is intended for OEM use only. The typical application does not need to use this command as the USB cameras initialize the A/D to factory set defaults when the camera powers up.
- For the USB_AD_IMAGING_GAIN and AD_USB_TRACKING_GAIN **commands** the allowed setting for the **data** parameter is 0 through 63. The actual Gain of the A/D (in Volts/Volt) ranges from 1.0 to 6.0 and is determined by the following formula:

$$\text{Gain} = 6.0 / (1.0 + 5.0 * ((63 - \mathbf{data}) / 63))$$

Note that the default A/D Gain set by the camera at power up is 1.2 for the Imaging CCD and 2.0 for the Tracking CCD. Furthermore, the gain item reported by the **Get CCD Info** command will always report the default factory-set gain and will not change based upon changes made to the A/D gain by this command.

- For the USB_AD_IMAGING_OFFSET and USB_AD_TRACKING_OFFSET **commands** the allowed setting for the **data** parameter is –255 through 255. Positive offsets increase the video black level in ADUs. The cameras are programmed at the factory to typically have a 900 to 1000 ADU black level offset.

3. Utility Programs

This section describes several tool programs that SBIG provides to help you get your custom application up and running.

3.1 SBIGDriverChecker.exe

This program checks the drivers installed on the system against the “SBIG Driver” directory. It reports whether the drivers installed are current and allows you to download the latest drivers from SBIG’s Servers with the Download button then update all drivers with the Update button. Please read the “Installing USB.pdf” and note that you should distribute this program and the SBIG Drivers directory with your code so users can install the latest drivers. The latest version and the associated drivers can be downloaded from the Software Downloads page of our home page www.sbig.com.

This utility needs to be run in Administrator mode on NT/2000/XP machines and depends on the SBIG Drivers directory being in the same directory as the utility. Also note that if you try to run this program from your installer and you get the error message “Could not find the SBIG Driv because the Installer is not setting the current working directory to the utility directory. You can get around this if you pass this program a command line argument with the full path to the utility. For example if these items are installed into the “C:\Program Files\My Company\My Program” directory pass

the SBIGDriverChecker.exe utility “C:\Program Files\My Company\My Program\SBIGDriverChecker.exe” without the quotes as the command line argument.

The program usually requires user intervention to update the drivers, which we feel is the safest way to go. You can however pass the program the “/s” command line argument in which case it will run in “silent” mode and automatically update the drivers and then exit.

3.2 EthSim.exe

The EthSim.exe tool allows you to simulate various model cameras on an Ethernet network. This is handy for testing your programs without a camera. To use EthSim do the following:

- Copy the EthSim.exe program and your Test Program to computers connected together through an Ethernet LAN. The two programs can run on two different computers or the same computer so long as that computer has Ethernet. You can use CCDOPS as the Test Program.
- Double-click the EthSim icon or run EthSim.exe from the command line. EthSim takes an optional command line parameter specifying the type of camera to simulate. For example, to simulate an ST-2K you would run:

```
ethsim 2K
```

with no command line parameter it defaults to an ST-7.

- Select Ethernet for the Comm link and enter the IP address of the computer running EthSim.exe
- Establish a link to the simulated camera

EthSim will generate random pixel data for both the Imaging and Tracking CCDs and simulates the following types of cameras:

ST-5C, ST-237, ST-237A, ST-7, ST-8, ST-9, ST-10, ST-1K, ST-2K

3.3 SBIGUDRVJournalRx.exe

The SBIGUDRVJournalRx.exe tool is used in conjunction with the DCP_CALL_JOURNAL_ENABLE Driver Control Parameter of the **Set Driver Control** command. When that parameter is set TRUE the driver broadcasts API calls. SBIGUDRVJournalRx.exe receives and displays those broadcasts. The broadcasts essentially tell you the sequence of API calls made to the driver.

SBIGUDRVJournalRx has a scrolling window that shows the API call sequence. In addition the program has the following controls:

- **Clear** – Clicking this button clears the log window
- **Filters** – Clicking this button brings up a dialog that allows you to filter out any or all of the API calls from the displayed log window.
- **Show Log** – Uncheck this button to disable the display of API calls
- **Send Msg** – Clicking this button simulates a broadcast from the driver to the program using the WParam and LParam data entry boxes. Each time you click this button a message should appear in the log.

Finally note that if you are not receiving API broadcasts messages from the driver it is probably because you did not enable journaling with the Set Driver Control command. Just for reference CCDOPS does not enable them.

3.3 SetClock.exe

This 16-Bit windows program, which you can distribute with you Application if you find it useful, reads the CMOS clock on the PC and sets the Windows System time to that value. It can take up to 1 second to synchronize the clocks as it waits till the CMOS clock hits the second mark. Typically you would spawn this program at the end of the readout on a Parallel camera to adjust for the seconds lost while interrupts were disabled for the readout.

3.5 GetPortD.exe

This 16-Bit DOS program is used to determine the I/O address of the LPT ports, something which we haven't figured out how to do under Windows. It reads the LPT address information from DOS, displays the address of the three LPT ports and creates a binary file named PORTADDR.DAT containing those addresses. We uses this to allow the user to select "LPT1, LPT2 or LPT3" instead of the I/O address for the Parallel port. Spawn this program, then open and read three unsigned shorts from the PORTADDR.DAT file. The first is the I/O address of LPT1, the second LPT2 and the third LPT3. You then pass this information to the Open Device command.

4. Revision History

This section details the recent changes to this specification and supporting software since the initial release of the ST-7 Driver.

Changes Incorporated in Version 1.90

- Added support for the PixCel255.
- Added a response to the EstablishLink command that reports the type of camera found.
- Renamed the main function SBIGUDrvCommand() from ST7Command() since it now supports the PixCel255 in addition to the ST-7/8.
- Added the new Error Code Unknown Camera.
- Added several ABG rates to the abgState item in the Start Exposure command.
- Added status results to the Pulse Out command that reports the current filter in position in the PixCel255.
- Added a type 3 request to the Get CCD Info command to report data on the PixCel255 vane/filter wheel configuration.

Changes Incorporated in Version 1.96

- Renamed ST-5C to PixCel255.
- Set PixCel255 e-/ADU based upon production hardware.

Changes Incorporated in Version 2.1

- Made 32 bit version available for Windows 95.
- Added the Open Driver and Close Driver commands.
- Added Driver Not Found, Driver Not Open and Driver Not Closed error codes for Open Driver and Close Driver functions.
- Added the Nx1, Nx2 and Nx3 readout modes for the ST-7/8
- Added the 1x1, 2x2 and 3x3 off-chip binning modes for the ST-7/8

Changes incorporated through Version 2.6

- Added support for the PixCel237 and the AO-7.

Changes incorporated through Version 2.70

- Added the **AO Delay** command for generating millisecond level delays.
- Added the **End Readout** command for preparing the CCD for the idle state.

- Added the ability to Freeze the TE Cooler for readout through the **Set Temperature Regulation** command.
- Added the REGULATION_FOZEN_MASK bit to the *enabled* field of the **Query Temperature Status** command results for detecting whether the TE cooler is frozen.

Changes incorporated through Version 3.3

- Added the **Open Device** and **Close Device** commands.
- Added support for the ST-9
- Added support for the ST-10
- Added support for the ST-1K
- Added support for an OEM version of the STV with Parallel Port
- Added support for the ST-237A

Changes incorporated through Version 4.0

- Added the START_SKIP_VDD option to the ccd item of the **Start Readout** command for higher imaging throughput.
- Changed calling convention to “stdcall” for compatibility with Visual C++ and Visual Basic.
- Changed calling struct member alignment to the default 8 bytes for compatibility.
- Made a single function callable from Windows 9X and Windows NT/2000/XP, using SBIGUDrvCommand() to replace ParDrvCommand() and ParDeviceCommandNH().
- Calls to **Open Device** and **Close Device** are now required.
- Added support for the USB and Ethernet based cameras and accessories.
- Removed the port information from the CC_ESTABLISH_LINK command and added it to the CC_OPEN_DEVICE command for compatibility with multiple hardware interfaces.
- Added the CC_GET and CC_SET_DRIVER_HANDLE commands for supporting multiple cameras per application.
- Added documentation for the **Get US Timer**, **Set IRQL**, **Get IRQL** commands.
- Added the **Get Line**, **Get Link Status**, **Start Readout** and **Get Error String** commands.
- Changed the minimum ST-7/8/9/10/1K exposure from 0.11 seconds to 0.12 seconds.

Changes incorporated to Version 4.1

- Added readoutMode 9 with 9x9 binning to the ST-7/8/etc for faster focus mode throughput.

Changes incorporated to Version 4.2

- Added support for the KAI2000 (ST-2K) and the TC-237 Tracking CCD
- Added requests 4 and 5 to the **Get CCD Info** command for use with the new frame transfer CCDs (TC237 Guider, KAI2000 Imager) when used with Start Exposure/End Exposure commands are bypassed as is typically done with AO exposures.
- Added the SBIGDriverChecker.exe Utility for updating the drivers to the current version.

Changes incorporated to Version 4.21

- Added the **Set Driver Control** and **Get Driver Control** commands.
- Fixed a bug in the documentation where the ipAddress was uint instead of ulong in the **Open Device** command.

Changes incorporated to Version 4.22

- Added the **USB AD Control** command.
- Added the DCP_CALL_JOURNAL_ENABLE and DCP_IVTOH_RATIO control parameters to the **Set Driver Control** command.

- Added the \Tools folder to the SBIG Universal Driver distribution with the EthSim.exe, SBIGUDRVJournalRx.exe, GetPortD.exe and SetClock.exe utility programs included.
- Released version 1.2 of the SBIGDriverChecker program that fixes a bug where some drivers could be installed in the wrong directory on some systems. Also removes those improperly installed drivers.
- Fix a bug in the documentation where the SBIGUnivDrvCommand() function was misnamed.
- Fix a bug in the documentation for the Activate Relay command regarding the bits in the command status.