Parallel Driver/Library

Version 2.6 - December 12, 1997

Santa Barbara Instrument Group

Table of Contents

1.   Introduction

This document describes the software interface to Santa Barbara Instrument Group's parallel port based CCD Cameras. This includes the ST-7, ST-8, PixCel255 (ST-5C) and PixCel 237 (ST-237). PC software interfaces to the camera through a driver/library supplied by SBIG.

The driver/library is available in many forms that support the 16 bit DOS and Windows 3.X environments and the 32 bit Windows 95 environment. The distribution diskette contains the following files:

PARDRV.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.

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

SBIG32.LIB - This is an import library that you link into your 32 bit Windows 95 program. Include this in your "project" file.

SBIG32.DLL - This is a 32 bit Windows 95 DLL for non Visual Basic programmers. You should copy it to your application's directory. It is build with the C calling convention.

SBIG32VB.DLL - This is the 32 bit Windows 95 DLL for Visual Basic 5 programmers. You should copy it to your application's directory. It is build with the Basic (standard) calling convention.

SBIG32.VXD - This is a protected mode low level driver for 32 bit Windows 95 that is used in conjunction with the SBIG32.DLL and SBIG32VB.DLL files. Copy this file to your \WINDOWS\SYSTEM directory.

The ST-7 and ST-8 are cooled CCD cameras that connect to the PC via the PC's parallel port. This helps alleviate the bottleneck associated with the serial ports and maintains the ease of use associated with this standard, built-in interface. Some of the highlights of the ST-7/8 include:

- Large Format, 768 x 512 (ST-7) or 1536 x 1024 (ST-8) Pixel CCD
- Dual CCD Enables Simultaneous Imaging and Tracking
- Low Noise 16 Bit Readout
- Electromechanical Shutter
- Temperature Regulated Cooling
- Support for the CFW-6A Motorized Color Filter Wheel
- Standard SBIG Telescope Interface for Tracking

The PixCel255 is a low-cost cooled CCD camera that is available thru Celestron. Some of the highlights of the PixCel255 include:

- Based upon the TI TC255 CCD
- Low Noise 16 Bit Readout
- Internal electromechanical vane which can be upgraded to a filter wheel
- Temperature Regulated Cooling

• Standard SBIG Telescope Interface for Tracking

The PixCel237 is a low-cost cooled CCD camera that is available thru Celestron. Some of the highlights of the PixCel237 include:

• Based upon the TI TC227 CCD
• Low Noise 12 Bit Readout
• Internal electromechanical vane which can be upgraded to a filter wheel
• Temperature Regulated Cooling
• Standard SBIG Telescope Interface for Tracking

2.   Driver Interface

The PC's software interface to the camera is through a LIB file that you link with your software. This library is available to third-party developers to ease software support of the SBIG parallel based cameras. 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:

short ParDrvCommand(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 PARDRV.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.

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 but will be from one of the following (note than an enum type is defined in the PARDRV.H file for these return codes):

> 0 = No Error
> 1 = Camera Not Found
> 2 = Exposure Already in Progress
> 3 = Exposure Not in Progress
> 4 = Bad PC Command
> 5 = Bad Camera Command
> 6 = Bad Parameter
> 7 = Transmission Timeout
> 8 = Receive Timeout
> 9 = NAK Received
> 10 = CAN Received
> 11 = Unknown Response
> 12 = Bad Length
> 13 = A/D Timeout
> 14 = Keyboard Escape
> 15 = EEPROM Checksum Error
> 16 = EEPROM Write /Read Error
> 17 = Shutter Error
> 18 = Unknown Camera
> 19 = Driver Not Found
> 20 = Driver Not Open
> 21 = Driver Not Closed

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

One thing that is **very important**, and has been a cause of trouble for people developing their own code using the parallel driver is that it is critical that you set the struct alignment option in your compiler to 2 bytes. Since all the entries in the structs passed and returned to the parallel driver are multiples of 2 bytes this means there are no unused or filler bytes. Where this will typically manifest itself is with the Take Image command where the

exposure time (a long, 4 bytes) will tend to cause struct misalignment. In C this is a simple matter of setting the struct alignment in the compile options to 2 bytes. In Visual Basic it's a bit more insidious. What you have to do is declare any elements of structs that use longs as 2 integers or 4 bytes. For example the declaration in Visual Basic for the TakeImageParams would be:

2.1   Driver Related Commands

The Commands in this section are used to open and close the driver and get driver related information. The driver should be opened at the start of your program and closed when you exit your program.

2.1.1   Open Driver - Command 17

The Open Driver command is used to load the driver into memory on systems using dynamic loading and to initialize the driver. While this is currently only required with the 32 bit Windows 95 version of the driver it is a good idea to use this command on all platforms.

      The Open Driver command takes no Parameters and returns no Results. Just pass NULL pointers to the Parameters and Results arguments of the ParDrvCommand function when you call it.

2.1.2   Close Driver - Command 18

The Close Driver command is used to close the driver and free the memory on systems using dynamic loading. While this is currently only required with the 32 bit Windows 95 version of the driver it is a good idea to use these command on all platforms.

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

2.1.3   Get Driver Info - Command 10

The Get Driver Info command is used by the PC 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 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
          1,2, etc. - reserved for future expansion
}

Standard 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
}

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:

b_1b_0 = Imaging CCD Status, 00 - CCD Idle, 10=In Progress, 11=Complete
b_3b_2 = Tracking CCD Status, 00 - CCD Idle, 10=In Progress, 11=Complete

Notes:
- For the ST-7/ST-8 the minimum allowable exposure is .11 seconds. If you ask the driver to make a shorter exposure it will take a .11 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 and PixCel237 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 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 0x8000 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 and PixCel237 the ccd parameter should be set to 0 for the Imaging CCD.

2.2.3 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
                0 = No binning, high resolution
                1 = 2x2 on-chip binning, medium resolution
                2 = 3x3 on-chip binning, low resolution (ST-7/8 only)
                xx03 = Nx1 on-chip binning (ST-7/8 only)
                xx04 = Nx2 on-chip binning (ST-7/8 only)
                xx05 = Nx3 on-chip binning (ST-7/8 only)
                6 = No binning, high resolution (ST-7/8 only)
                7 = 2x2 binning with vertical binning off-chip (ST-7/8 only)
                8 = 3x3 binning with vertical binning off-chip (ST-7/8 only)
         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 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 starting pixel and pixel length 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 ST-7/8's Tracking CCDs only support the 1x1 and 2x2 binning modes. The 3x3 binning mode is supported by the ST-7/8's Imaging CCD and by the PixCel237 only.
- With the PixCel255 and PixCel237 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 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. 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.

2.2.4   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:

$$Data[n] = CCD[n] - Data[n] + 100$$

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
                    0 = No binning, high resolution
                    1 = 2x2 binning, medium resolution
                    2 = 3x3 binning, low resolution
          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
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.

2.2.5   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
                0 = No binning, high resolution
                1 = 2x2 binning, medium resolution
                2 = 3x3 binning, low resolution
        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.

2.3   Temperature Related Commands
The commands in this section are used to program or monitor the CCD's temperature
regulation. Note that the cameras contains 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

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:

$$T_0 = 25.0 \qquad\qquad R_0 = 3.0$$
$$MAX\_AD = 4096$$
$$R\_RATIO_{CCD} = 2.57 \qquad R\_RATIO_{Ambient} = 7.791$$
$$R\_BRIDGE_{CCD} = 10.0 \qquad R\_BRIDGE_{Ambient} = 3.0$$
$$DT_{CCD} = 25.0 \qquad DT_{Ambient} = 45.0$$

Calculation of Setpoint from Temperature T in °C

$$r = R_0 \times e^{\frac{\ln(R\_RATIO) \times (T_0 - T)}{DT}}$$

$$setpoint = \frac{MAX\_AD}{\dfrac{R\_BRIDGE}{r} + 1.0}$$

Calculation of Temperature T in °C from Setpoint

$$r = \frac{R\_BRIDGE}{\dfrac{MAX\_AD}{setpoint} - 1.0}$$

$$T = T_0 - DT \times \frac{\ln\dfrac{r}{R_0}}{\ln(R\_RATIO)}$$

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.

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:

b_0 = +X Relay, 0=Off, 1= Active
b_1 = -X Relay, 0=Off, 1= Active
b_2 = +Y Relay, 0=Off, 1= Active
b_3 = -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 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:

b_0 - Normal status, 0 = inactive, 1 = pulse out in progress
b_1-b_3 - 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 verly 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 reeason 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 verly 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 reeason 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.

2.5 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 {
      enum **port** - port for communications
            0 = address specified by base address parameter
            1 = Parallel port at base address 3BC hex
            2 = Parallel port at base address 378 hex
            3 = Parallel port at base address 278 hex
      uint **baseAddress** - base address of parallel port when port above is set to 0
}

Results Struct:
struct EstablishLinkResults {
      enum **cameraType** - constant specifying the type of camera, 4=ST-7, 5=ST-8,
            6=PixCel255, 7=TCE, 8=PixCel237
}
Notes:
 - DOS assigns LPT1, LPT2, etc. at power-up. You can not guarantee that LPT1 will be at any specific base address. For reference DOS places the base address of LPT1 at 0040:0008 in the BIOS variables segment. LPT2's base address is stored at 0040:000A, LPT3 at 0040:000C and LPT4 at 0040:000E.

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. Initially 2 standard requests will be 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
            4,5, 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, 4=ST-7, 5=ST-8,
            6=PixCel255, 7=TCE, 8=PixCel237

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 XXXXX.XX format.
      ulong **pixelHeight** - a eight digit BCD number specifying the pixel height in
            microns in the XXXXX.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 = External, 1 = 2 Position, 3 = 5 Position
}

Notes:
- The ST-7/8 supports types 0, 1 and 2 requests. The PixCel255/237 supports request types 0 and 3.

2.5.3   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
}

2.5.4   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 **fanEnabled** - 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
>   enum **filterCommand** - 0=leave filter alone, 1-5=position at filter 1-5, 6=force motor
>   off

}

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

> b_7-b_0 - 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.
> b_8 - the Fan is enabled when this bit is 1
> $b_{10}b_9$ - Shutter state, 0=open, 1=closed, 2=opening, 3=closing
> $b_{12}b_{11}$ - LED state, 0=off, 1=on, 2=blink low, 3=blink high

Notes:
- The ST-7/8 have a shutter, LED and fan but no filter wheel. The setting of the filterCommand parameter is ignored and should be set to 0. To position the CFW-6/8 attached to the ST-7/8 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 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.

2.5.5   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 ParDrvCommand function when you call it.

2.5.6   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
}

3.   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 1.96
- Renamed ST-5C to PixCel255.
- Set PixCel255 e-/ADU based upon production hardware.

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 ParDrvCommand() 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 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.
- Added support for the AO-7.