ST-7/ST-8 Driver/Library

Version 1.07 - June 1, 1995

Santa Barbara Instrument Group

Table of Contents

1. Introduction

This document describes the software interface to Santa Barbara Instrument Group's ST-7/8 CCD Cameras. PC software interfaces to the ST-7/8 through a driver/library supplied by SBIG. The driver/library is available as a linkable ST7DRV.LIB file and is accompanied by a C ST7DRV.H header file that contains the function prototype and struct and enum declarations.

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

2. DLL 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 ST-7/8. 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:

```
int far ST7Command(int Command, void far *Parameters, void far *Results)
```

where Command is the Command to be executed, Parameters and Results are far 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:

- 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 element within the structs:

BOOLEAN - unsigned short (2 bytes) with 0 = FALSE and 1 = TRUE
enum - Enumerated signed 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:

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 Checksium Error
16 = EEPROM Write /Read Error
17 = Shutter Error

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. 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.

2.1.1. 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 during integration (normal setting, ABG protection active)
    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:

- 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.

2.1.2. 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 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.
- 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.

2.1.3. 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 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 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 mya want to use the Update Clock command to resynchronize the DOS clock after reading out an image.
- The 3x3 binning mode is supported by the Imaging CCD only.

2.1.4. 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:
- Any arbitrary region can be readout using the Dump Lines and Readout Line commands by varying the starting pixel and pixel length parameters.
- The 3x3 binning mode is supported by the Imaging CCD only.
- 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.

2.1.5. Dump Lines - Command 4
The Dump Lines command is used to discard all of the active pixels in a row on the CCD.

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:
- 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.
- The 3x3 binning mode is supported by the Imaging CCD only.

2.2. Temperature Related Commands

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

2.2.1. 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:

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

Calculation of Setpoint from Temperature T in °C

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

$$\text{setpoint} = \frac{MAX\_AD}{\left(\frac{R\_BRIDGE}{r} + 1.0\right)}$$

Calculation of Temperature T in °C from Setpoint

$$r = \frac{R\_BRIDGE}{\left(\frac{MAX\_AD}{\text{setpoint}} - 1.0\right)}$$

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

2.2.2. Query Temperature Status - Command 6

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

Results Struct:

struct QueryTemperatureStatusResults {
    BOOLEAN **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.

2.3. 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.

2.3.1. 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.

2.3.2. Pulse Out - Command 8

The Pulse Out command is used to position the CFW-6A or to cancel a positioning command in progress.

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

}

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.
- The status for this command will include the bit CS_PULSE_IN_ACTIVE when the CFW-6 input is active (pulled low).

2.4. 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.

2.4.1. 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

}

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.

2.4.2. 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
}

2.4.3. Get CCD Info - Command 11

The Get CCD Info command is used by the PC to determine the model of ST-7/8 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
        3,4, etc. - reserved for future expansion
}

Standard Results Struct :
requests 0 and 1 -
struct GetCCDInfoResults0 {
    uint **firmwareVersion** - version of the firmware in the resident microcontroller
    uint **cameraType** - constant specifying the type of camera, 4=ST-7, 5=ST-8
    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
}

2.4.4. 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
}

2.4.5. 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 {
BOOLEAN **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
}

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

Note that the camera will cease communications with the PC while the reinitialize shutter command is in progress. After the cmarea 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.

2.4.6. 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 ST7Command function when you call it.

2.4.7. Read Offset - Command 16

The Read Offset command is used to measure the CCDs offset. In the ST-7/ST-8 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 Draft Version.

Changes Incorporated in Version 1.01

- Improved frame to frame repeatibility in readout times.

Changes Incorporated in Version 1.02

- Added reduction in Imaging CCD Vdd to reduce readout amplifier glow.

Changes Incorporated in Version 1.03

- Improved frame to frame repeatibility in readout times.
- Increased strobe width to camera for improved long-cable operation.

Changes Incorporated in Version 1.04

- Added ability to skip shutter dealy in End Exposure command.
- Modified Update Clock command to only update DOS Tick Count, not Real Time Clock.
- Modified shutter delay for Burr Brown A/D settling.

Changes Incorporated in Version 1.07

- Switched to using Timer 2 for time delays for better compatibility with various PC BIOS.