Universal CPU Command Structure

Supports ST-4X, ST-5, and ST-6 Cameras

December 15, 1993

Includes ST-4X Version 1.0 Firmware
Includes ST-5 Version 1.0 Firmware
Includes ST-6 Version 3.0 Firmware

Santa Barbara Instrument Group
1482 East Valley Road • Suite 31
PO Box 50437
Santa Barbara, California 93150

Table of Contents

1. What's New

Since the last release of this document (dated February 15, 1993) this specification has been expanded to support 2 new cameras: the ST-4X and the ST-5. These cameras use the same CPU as the ST-6 (now referred to as the Universal CPU), and thus have a very similar interface. The major differences between the three cameras are highlighted in the table below:

Camera	Pixels	Dark Vane	Frame Transfer	Temperature Regulation
ST-4X	192 x 164	No	No	Open Loop
ST-5	320 x 240	No	Yes	Closed Loop Regulation
ST-6	375 x 242	Yes	Yes	Closed Loop Regulation

For users who are familiar with previous versions of this document, you should read the section "Working with Different Camera Configurations" and then browse the individual commands. While several new commands have been added, all previously existing commands are unchanged so that all current ST-6 software can be used without modification. The new commands are:

get_cpu_info - Returns information about the model of camera the CPU supports.
put_line, put_uncompressed_line - Allows the Host to upload image data into the CPU's image buffers.
shrink_image - Like the **reduce_image** command except the image is reduced by averaging a 2 x 2 block of pixels.
pulse_out - Sends pulses out the AUX port for control of motors, etc.
flush_ccd - Flushes the CCD by doint vertical transfers.

2. Introduction

This document describes the software interface to the ST-4X, ST-5, and ST-6 CCD cameras. These cameras consists of two components: the Optical Head and the CPU. The CPU interfaces to the Host computer through three-wire RS232 or, for long cable runs, through differential RS422. The CPU is microcontroller based with internal firmware that controls the operation of the Optical Head and provides frame storage capability for captured images[1]. The CPU communicates to and is controlled by the Host computer through a packet oriented protocol that involves the Host acting as a Master and the CPU acting as a slave. The CPU only sends data to the Host at the request of the Host. In general the flow of data from the Host computer to the CPU is as shown below:

1. The Host builds a command packet to be sent to the CPU.
2. The Host sends the packet to CPU.
3. The CPU responds to the command from host in one of the following ways:
   a) The CPU returns an ACK message to the Host to acknowledge receipt of a command that has no formal response.
   b) The CPU builds and returns a packet to the Host containing data relevant to the command issued by the Host.

[1] The CPU has three image buffers referred to as the Dark Buffer, the Light Buffer and the Accumulation Buffer. The Dark and Light buffers are 16 bit buffers and the Accumulation buffer is 32 bits. Commands affecting the image buffers allow you to select the destination buffer.

c) Upon parsing the packet the CPU detects a checksum error in the packet and returns a NACK message to the Host to indicate retransmission of the command is needed.
d) Upon parsing the packet the CPU detects an unknown command, a command with an incorrect length, or a parameter out of range within the command and returns a CAN message to the Host indicating an error in the Host's programming.

4. Depending on the CPU's response, the Host may need to resend the command packet to the CPU, for example when a NACK or no response at all is returned from the CPU.

Since the communications between the Host and the Camera are not assumed to be operating in a error-free environment some mechanism for error recovery is provided (as demonstrated by the NACK response for example). Inherent in the design of the packet protocol is the ability to determine proper transmission and receipt of data. To facilitate operations in a non-perfect communications environment the packet protocol implemented in the CPU adheres to the following rules:

- The CPU only transmits data to the Host in response to commands received from the Host, and then only sends the data once.
- Each command sent to the CPU will elicit some kind of response from the CPU.
- The CPU will respond within a fixed period of time after having received the command from the host. From the Host's point of view if the ST-6 has not responded within 0.1 seconds from transmission of the final byte in the packet then it can be assumed that the CPU did not receive the command and it should be retransmitted.
- The CPU expects that all the data received within a packet will arrive in a burst. If the CPU is in the middle of receiving a packet and there is a dead time of 2.56 seconds (between consecutive bytes) the CPU will resynchronize its packet parsing, looking for the start of a new packet.
- When the CPU is transmitting data to the host in response to a command sent from the host it will build the packet and transmit it in a burst, without any appreciable delay between bytes.

3. Packet Structure

The commands sent from the Host computer to the CPU are structured into packets. The packets contain additional information to aid in the receipt and detection of transmission errors. The packet structure is shown below:

A5H = Start Byte
XXH = Command Byte
XXH = Data Length N (Low Byte)
XXH = Data Length N (High Byte)
XXH = Data Byte 1
XXH = Data Byte 2
.
.
.

XXH = Data Byte N
XXH = Checksum (Low Byte)
XXH = Checksum (High Byte)

Several items about the packet structure are worth noting:

- All packets start with the Start Byte A5 hexadecimal.
- There are 256 possible commands due to the one-byte command.
- The length bytes contained in the packet tell the data length. The packet length including the start byte, command byte, length bytes and checksum bytes is equal to the Data Length plus 6.
- The Checksum is the 16 bit unsigned sum of all the bytes in the packet except the checksum bytes themselves. The checksum is modulo 65536, meaning if the sum overflows 16 bits it just wraps around.
- The CPU has buffers to be able to receive and send packets up to 1024 bytes in length. The longest packet the Host software will receive from the CPU is 756 bytes long (the response to the **get_line** command) so 1024 byte buffers on the Host should be adequate.

In addition to the A5 hexadecimal Start of Packet byte, the following byte constants are defined:

A5H = Start of Packet
06H = ACK (Packet received ok)
15H = NAK (Bad checksum)
18H = CAN (Bad command, length, parameter)

4. Serial Data Format

At power-up the CPU expects communications with the Host computer over its COM port at 9600 baud. The baud rate can be reprogrammed for other data rates (using the **set_com_baud** command discussed later). The data format is 8 data bits, no parity bits, and 1 stop bit (a total of 10 bits per byte including the start bit).

5. Working with Different Camera Configurations

Since the Universal CPU can be connected to several model cameras with different firmware and features the Host Computer should first establish a communications link with the CPU and then identify the type of instrument at the other end of the link.

Establishing a communications link with the CPU is most easily achieved by sending the CPU the **get_rom_version** command. Try 9600 baud if you had not previously communicated with the CPU, or try the last baud rate you had previously used if you had established a link. If that fails (the CPU does not give a valid response) you should scan the baud rates supported by your software trying to establish a link. Since the CPU powers up at 9600 baud and will only change baud rates when the Host instructs it to, scanning only the rates supported by your software should suffice.

Once the link has been established, you should issue the CPU a **get_cpu_info** command to identify the camera model and capabilities. If the model is not supported by your software you should not continue to send commands to the CPU and you may want to inform the user. Using the data returned by the **get_cpu_info** command you should determine the features the CPU will support and use that information to tailor other commands you supply to the CPU. The information returned from the **get_cpu_info**

command includes the following: CPU Type (ST-4X, ST-5 or ST-6), Supported Capabilities (Temperature Regulation, etc.), Image Buffer Dimensions (Height, Width), Readout Modes (Number of Pixels and Pixel Dimensions).

6. Commands

This section describes the individual commands that the Host computer can send to the CPU. Each command has a name and a command byte (the command byte is the 2nd byte in the packet sent to the CPU). Individual commands can have zero or more data bytes (depending on the command itself). If the command has no data then the Data Length bytes in the packet are both zero. Data within commands is described in terms of the following parameter types:

byte	Unsigned 8 bit integer.
boolean	Two bytes in length, least significant byte first. A value of 1 designates TRUE and a value of 0 designates FALSE.
buffer	Enumerated unsigned integer, two bytes in length, least significant byte first. A value of 0, 1, or 2 designates the Dark, Light, and Accumulation image buffers respectively.
enum	Enumerated unsigned integer value with a set of allowed values. Two bytes in length, least significant byte first.
int	Unsigned integer, two bytes in length, least significant byte first.
signed int	Same as int only signed.
long	Unsigned long integer, 4 bytes in length, least significant byte first followed by the next 3 bytes in order of increasing significance.
signed long	Same as long only signed.

Some commands sent from the Host to the CPU elicit only an ACK response from the CPU (an ACK is the single byte 06 hexadecimal) where as other commands involve the CPU building and sending a packet of data back to the Host computer. In this case, the response looks like the packet sent to the CPU, starting with an A5 Start Byte and the same Command Byte, but with a Data Length and Data format specific to the command, all followed by the two byte Checksum.

An additional case to consider is commands with a "delayed response". Several PU commands can take a variable amount of time to complete. Rather than the CPU waiting for the command to finish before sending the ACK message, the CPU sends the command to a foreground processor and sends the ACK back right away. The Host can then monitor the progress of the command by using the **get_activity_status** command. If a command with a "delayed response" actually needs to return some data to the host (such as the **cal_cent** command) the command returns the ACK right away and then puts the computed result into a global result buffer which can be downloaded when the command has finished processing using the **get_result_buf** command.

Each of the CPU's commands are described in the sections below. The commands have been broken down into logical groups.

Exposure Related Commands
The commands described in this section are used to start or stop exposures, control the shutter, adjust the head offset or read the heads video output level.

take_image - Command 01H
Purpose: Control shutter, Clear CCD, Time exposure and Readout CCD.
Parameters:
exposure_time (long)-integration time in hundredths of a second
line_start (int)-top line to readout (value depends on readout mode, 0 thru max height - 1)
line_len (int)-number of lines to readout (value depends on readout mode, 1 thru max height)
pixel_start (int)-leftmost pixel to readout (value depends on readout mode, 0 thru max width - 1)
pixel_len (int)-number of pixels to readout (value depends on readout mode, 1 thru max width)
enable_dcs (boolean)-on ST-6 do double correlated readout if TRUE, on other cameras parameter doesn't affect readout and should be set FALSE
dc_restore (boolean)-on ST-6 do DC restore on readout if TRUE. On an ST-4X reduce the CCD amplifier bias during integration to reduce the readout glow if TRUE. On the ST-5 this parameter doesn't affect readout and should be set FALSE
abg_state (enum)-antiblooming gate state
0 = Low during integration (ABG shut off)
1 = Clocked during integration (normal setting, ABG protection active)
2 = Mid during integration (test setting)
abg_period (int)-antiblooming period in 4.3 microsecond steps when ABG clocked, minimum value of 30 counts (129 µs). The recommended values are shown in the table below based on the amount of antiblooming protection required:

Desired ABG Protection	abg_period
Highest amount of ABG protection (extremely bright object in field of view)	600
Medium amount of ABG protection (normal viewing)	6000
Minimal ABG protection (when lowest noise readout is required with effectively no ABG protection)	65535

dest_buffer (buffer)-destination buffer for readout
auto_dark (boolean)-subtract dark buffer before saving to destination buffer when TRUE
readout_mode (enum)-specifies the readout format of the CCD to be used in digitizing the image. Possible values for this parameter are returned by the **get_cpu_info** command as different model cameras support different readout modes. The readout modes supported by the various cameras are:

ST-4X - the two modes supported are HIGH (readout_mode set to 0) for a resolution of 192 x 164 pixels with an A/D gain of 7.2 e^-/count and LOW (readout mode set to 1) for a resolution of 96 x 82 pixels with an A/D gain of 14.4e-/count.

ST-5 - the two modes supported are HIGH (readout_mode set to 0) for a resolution of 320 x 240 pixels with an A/D gain of 3.0 e^-/count and LOW (readout mode set to 1) for a resolution of 160 x 120 pixels with an A/D gain of 6.0 e-/count.

ST-6 - combines 2 pixels horizontally when readout_mode is 1 (375 x 242 pixels), or combines 2 pixels vertically when readout_mode is 0 (750 x 121 pixels) and has been supplemented by the modes discussed below:

Version 2.0 ROM Change -
The following binning modes are supported by the Version 2.0 and later ROMs:

readout_mode value	*0*	*1*	*2*	*3*	*4*
mode name	*-*	*HIGH*	*MEDIUM*	*LOW*	*HI_HRES*
Vertical Binning	*2**	*1*	*1*	*2*	*2*
Horizontal Binning	*1*	*2**	*3*	*3*	*1*
Max Image Columns	*750*	*375*	*250*	*250*	*750*
Max Image Rows	*121*	*242*	*242*	*121*	*121*

and the following additional modes are supported by the Version 2.01 and 3.01 ROMs

readout_mode value	*5*	*6*	*7*	*8*
mode name	*SPECT_HI*	*SPECT_MED*	*SPECT_LOW*	*SPECT_X_MED*
Vertical Binning	*8*	*8*	*8*	*242*
Horizontal Binning	*1*	*2**	*3*	*2**
Max Image Columns	*750*	*375*	*250*	*375*
Max Image Rows	*30*	*30*	*30*	*1*

*where * represents off-chip binning where 1 A/D count = 6.7 e-, otherwise on-chip binning is used where 1 A/D count = 3.35 e-*

Version 3.01 ROM Change -
The following additional mode is supported by the Version 3.01 ROMs

readout_mode value	*9*
mode name	*SPECT_X_HI*
Vertical Binning	*242*
Horizontal Binning	*1*
Max Image Columns	*750*
Max Image Rows	*1*

on-chip binning is used where 1 A/D count = 3.35 e-

open_shutter (boolean)-for an ST-6 open shutter for exposure when TRUE, for other cameras parameter does not affect readout

ST-6 Version 2.0 or Higher ROM Change -
0 = Shutter closed for integration and readout.
1 = Shutter open for integration, closed for readout.
2 = Shutter open for integration and readout.

Response: ACK

Status Values:
0=Idle, 1=Sent to foreground, 2=Waiting for shutter, 3=Flushing CCD,
4=Timing exposure, 5=Waiting for **end_exposure**, 6=Transferring CCD,
7=Waiting for readout, 8=Reading CCD, 9=Post processing,
100-341=Digitizing line n where n=status-100

Notes:

- If exposure_time=0 then **take_image** command clears the CCD and then waits for an **end_exposure** command to terminate the integration and start the readout.
- On sub-frame readout the image data is positioned at the correct position in destination buffer, not at 0,0.
- When the **take_image** command is digitizing the image try to keep the communications with the CPU to a minimum, interrogating it only 3 to 4 times per second.
- Use the **get_cpu_info** command to determine the maximum allowable image dimensions for each readout mode.
- When the auto_dark parameter is set TRUE, a bias of 100 counts is added to the resulting difference to stop clipping pixels below zero.
- If a **flush_ccd** command is in progress when the **take_image** command is issued the **take_image** command is ignored.

ST-6 Cameras Only

- The **take_image** command treats the image buffers as 750 pixels wide by 121 pixels high or 375 pixels wide by 242 pixels high depending on whether the horizontal binning is 1 or 2 and greater respectively.
- As a suggestion always use the **set_head_offset** command prior to the **take_image** command in case the CPU has been powered down.
- For a low noise readout set the enable_dcs parameter TRUE and the dc_restore parameter FALSE. For a rapid readout set the enable_dcs parameter FALSE and the dc_restore parameter TRUE.
- When either the enable_dcs or dc_restore parameters are TRUE the readout software adds a bias of 100 A/D counts to the video level.

end_exposure - Command 02H

Purpose: Transfers and reads out CCD.

Parameters:
abort (boolean)-Skip transfer and readout of CCD if TRUE

Response: ACK

Status Values:
0=Idle, 6=Transferring CCD, 7=Waiting for readout, 8=Reading CCD,
9=Post processing, 100-341=Digitizing line n where n=status-100

Notes:

- Aborts exposure in progress if not waiting for **end_exposure** command.
- Use this command after issuing a **take_image** command with the exposure_time set to zero.

shutter_control - Command 04H

Purpose: Open or close the shutter on an ST-6.
Parameters:
close (boolean)-Close (activate) shutter if TRUE, open if FALSE
*Response:*ACK
Status Values:
0=Idle, Shutter open, 1=Shutter closed
Notes:

- Physical activation of the shutter only occurs if an exposure is not in progress when this command is received.
- While the ST-6 is the only camera that has a shutter, sending this command to an ST-4X or an ST-5 will not generate an error. The camera will simply ignore the command and return an ACK.

set_head_offset - Command 0FH

Purpose: Set the electrical offset in the optical head of the ST-6.
Parameters:
offset (int)-Setting of offset DAC in head, allowed values are 0-255
*Response:*ACK
Status Values:
0=Idle, 2=In progress
Notes:

- Physical activation of the head offset only occurs if an exposure is not in progress and the **flush_ccd** command is idle when this command is received.
- Use the **read_blank_video** command to determine the proper setting for the offset parameter.
- As a suggestion always use the **set_head_offset** command prior to the **take_image** command in case the CPU has been powered down.
- While the ST-6 is the only camera that requires setting the head offset, sending this command to an ST-4X or an ST-5 will not generate an error. The camera will simply ignore the command and return an ACK.

read_blank_video - Command 12H

Purpose: Read the output level of the CCD at the black level as digitized by the A/D.

Parameters:

enable_dcs (boolean)-For ST-6 cameras enables DCS when TRUE. Other cameras ignore the setting of this parameter.

head_offset (int)-For ST-6 cameras this is the setting of the offset DAC in the head with allowed values of 0-255. Other cameras ignore the setting of this parameter.

Response: (immediate packet)

video (int)-CCD black level A/D value

Status Values:

0=Idle, 2=In progress

Notes:

- Physical activation of this command only occurs if an exposure is not in progress and the **flush_ccd** command is idle when this command is received.

ST-6 Cameras Only

- This command can be used to determine the correct setting for the head offset by repetitively executing this command until the returned video is as small as possible but still above a few hundred counts.
- Increasing the head_offset value will increase the video level by approximately 7000 counts.

flush_ccd - Command 27H

Purpose: Provide the capability of additional external flush or clear the CCD through vertical transfers.

Parameters:

times (int)-Number of flush cycles to perform

*Response:*ACK

Status Values:

0=Idle, 2=In progress

Notes:

- Physical activation this command only occurs if an exposure is not in progress when this command is received.
- It is not necessary to issue a **flush_ccd** command prior to using the **take_image** command. The **take_image** command flushes the CCD four times at the start of the integration.

ST-6 Cameras Only

- This command is only available in ST-6 version 3.0 and later ROMS.

Image Processing Commands

The commands described in this section are image processing commands, clearing image buffers, co-adding buffers, etc.

clr_buf - Command 06H

Purpose: Clears one of the image buffers by filling it with zeros.
Parameters:
buf (buffer)-Destination buffer to clear
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress
Notes:

- You do not need to call this function prior to using the **take_image** command since the buffers are overwritten by that command.

get_line - Command 07H

Purpose: Transmit a compressed line of image data
Parameters:
buf (buffer)-Source image buffer from which data will be transmitted
line_start (int)-line to transmit (value depends on the size of the image buffers, 0 thru max height - 1)
pixel_start (int)-leftmost pixel to transmit (value depends on the size of the image buffers, 0 thru max width - 1)
pixel_len (int)-number of pixels to transmit (value depends on the size of the image buffers, 1 thru max width)
Response:(immediate packet)
line_start (int)-line being sent
compressed_data_1 (byte) - 1st byte of compressed image data
.
.
compressed_data_N (byte) - last byte of compressed image data
Status Values:
0=Idle, 2=In progress
Notes:

- The compression algorithm used is described at the end of this document.
- Using compression you can effectively double the image download throughput.
- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- The **get_line** command treats the image buffers as 375 pixels wide by 242 pixels high. If the **take_image** command has been issued with a 750 pixel wide readout mode then you will have to make 2 calls to this command to get the data. The 1st line received will be the left half of the digitized image and the 2nd line received will be the right half of the digitized image.

get_uncompressed_line - Command 1FH

Purpose: Transmit an uncompressed line of image data.

Parameters:

buf (buffer)-Source image buffer from which data will be transmitted

line_start (int)-line to transmit (value depends on the size of the image buffers, 0 thru max height - 1)

pixel_start (int)-leftmost pixel to transmit (value depends on the size of the image buffers, 0 thru max width - 1)

pixel_len (int)-number of pixels to transmit (value depends on the size of the image buffers, 1 thru max width)

Response: (immediate packet)

line_start (int)-line being sent

data_1 (int)-1st pixel

.

.

data_N (int)-last pixel

Status Values:

0=Idle, 2=In progress

Notes:

- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- The **get_uncompressed_line** command treats the image buffers as 375 pixels wide by 242 pixels high. If the **take_image** command has been issued with a 750 pixel wide readout mode then you will have to make 2 calls to this command to get the data. The 1st line received will be the left half of the digitized image and the 2nd line received will be the right half of the digitized image.

put_line - Command 22H

Purpose: Send a compressed line of image data up to the CPU for storage in one of the image buffers

Parameters:

buf (buffer)-Destination image buffer where data will be placed
line_start (int)-line being sent (value depends on the size of the image buffers, 0 thru max height - 1)
pixel_start (int)-leftmost pixel being sent (value depends on the size of the image buffers, 0 thru max width - 1)
pixel_len (int)-number of pixels being sent (value depends on the size of the image buffers, 1 thru max width)
compressed_data_1 (byte) - 1st byte of compressed image data
.
.
compressed_data_N (byte) - last byte of compressed image data

Response:ACK

Status Values:

0=Idle, 2=In progress

Notes:

- The compression algorithm used is described at the end of this document.
- Using compression you can effectively double the image upload throughput.
- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- This command is only available in ST-6 version 3.0 and later ROMS.
- The **put_line** command treats the image buffers as 375 pixels wide by 242 pixels high. If you want to upload an image simulating a 750 pixel wide readout mode then you will have to make 2 calls to this command to send the data. The 1st line sent will be the left half of the image and the 2nd line sent will be the right half of the image.

put_uncompressed_line - Command 23H

Purpose: Send an uncompressed line of image data up to the CPU for storage in one of the image buffers

Parameters:

buf (buffer)-Destination image buffer where data will be placed
line_start (int)-line being sent (value depends on the size of the image buffers, 0 thru max height - 1)
pixel_start (int)-leftmost pixel being sent (value depends on the size of the image buffers, 0 thru max width - 1)
pixel_len (int)-number of pixels being sent (value depends on the size of the image buffers, 1 thru max width)
data_1 (int) - 1st pixel
.
.
data_N (int) - last pixel

Response:ACK

Status Values:

0=Idle, 2=In progress

Notes:

- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- This command is only available in ST-6 version 3.0 and later ROMS.
- The **put_uncompressed_line** command treats the image buffers as 375 pixels wide by 242 pixels high. If you want to upload an image simulating a 750 pixel wide readout mode then you will have to make 2 calls to this command to send the data. The 1st line sent will be the left half of the image and the 2nd line sent will be the right half of the image.

get_readout_peak - Command 03H

Purpose: Get the peak pixel value and location found during the previous readout.

Response: (immediate packet)

peak_value (int)-Peak pixel value found during readout
peak_x (int)-Location of peak in x
peak_y (int)-Location of peak in y

Status Values:

0=Idle, 2=In progress

Notes:

- If this command is issued after the **take_image** command with the auto_dark parameter TRUE it will report the peak value after the dark frame subtraction.

cal_width - Command 1EH

Purpose: Calculates the peak and the average pixel value in the box, then calculates the height and width at half the amplitude of the peak over the average for the data contained within the box, placing the results into the global result buffer.

Parameters:

buf (buffer)-source buffer to use in calculating the width
x_offset (int)-leftmost pixel of box (value depends on the size of the image buffers, 0 thru max width -1)
y_offset (int)-top line of box (value depends on the size of the image buffers, 0 thru max height - 1)
x_length (int)-width of box (value depends on the size of the image buffers, 1 thru max width)
y_length (int)-height of box (value depends on the size of the image buffers, 1 thru max height)

Response:ACK immediately plus buffered results

Results: (placed into global results buffer)

width (int)-full width at half amplitude of star in box
height (int)-full height at half amplitude of star in box

Status Values:

0=Idle, 1=Sent to foreground, 2=In progress

Notes:

- You must use the **get_result_buf** command once this command has finished executing to get the results of this command.
- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- This command assumes the image buffers are 375 pixels wide by 242 pixels high and will produce meaningless results if applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode.

cal_cent - Command 08H

Purpose: Calculates the centroid of a given box of pixels and places the result in the global results buffer.

Parameters:

buf (buffer)-source buffer to use in calculating the centroid
x_offset (int)-leftmost pixel of box (value depends on the size of the image buffers, 0 thru max width -1)
y_offset (int)-top line of box (value depends on the size of the image buffers, 0 thru max height -1)
x_length (int)-width of box (value depends on the size of the image buffers, 1 thru max width)
y_length (int)-height of box (value depends on the size of the image buffers, 1 thru max height)

Response:ACK immediately plus buffered results

Results: (placed into global results buffer)

cent_x (long)-x centroid of box multiplied by 65536
cent_y (long)-y centroid of box multiplied by 65536
sum (long)-sum of pixels over threshold

Status Values:

0=Idle, 1=Sent to foreground, 2=In progress

Notes:

- You must use the **get_result_buf** command once this command has finished executing to get the results of this command.
- Use the **get_cpu_info** command to determine the maximum size of the image buffers.
- This command calculates a threshold level at the average pixel value within the box and uses that as a discrimination level in calculating the centroid.

ST-6 Cameras Only

- This command assumes the image buffers are 375 pixels wide by 242 pixels high and will produce meaningless results if applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode.
- Version 1.0 ROMs calculate a threshold level at half the maximum pixel value above the average pixel value within the box and uses that as a discrimination level in calculating the centroid.

reduce_image - Command 09H

Purpose: Reduce image by combining and averaging 4 x 4 pixels into 1 pixel
Parameters:
buf (buffer)-source/destination buffer to reduce
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress
Notes:

- This command overwrites the data in the image buffer, placing the upper left corner of the reduced image at pixel 0,0.

ST-6 Cameras Only

- This command assumes the image buffers are 375 pixels wide by 242 pixels high and will produce meaningless results if applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode.

shrink_image - Command 24H

Purpose: Reduce image by combining and averaging 2 x 2 pixels into 1 pixel
Parameters:
buf (buffer)-source/destination buffer to reduce
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress
Notes:

- This command overwrites the data in the image buffer, placing the upper left corner of the shrunk image at pixel 0,0.

ST-6 Cameras Only

- This command is only available in ST-6 version 3.0 and later ROMS.
- This command assumes the image buffers are 375 pixels wide by 242 pixels high and will produce meaningless results if applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode.

accum_image - Command 0AH

Purpose: Add the light buffer to the accumulation buffer with X and Y offsets.
Parameters:
x_offset (signed int)-x offset of light buffer relative to accumulation buffer. Positive values shifts light buffer to right
y_offset (signed int)-y offset of light buffer relative to accumulation buffer. Positive values shifts light buffer down
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress
Notes:
ST-6 Cameras Only

- This command assumes the image buffers are 375 pixels wide by 242 pixels high and will produce meaningless results if applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode.

sub_offset - Command 0BH

Purpose: Subtract a constant offset from an image buffer.
Parameters:
buf (buffer)-source/destination buffer to subtract constant from
count (signed long)-value to subtract from buffer
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress

get_minmax - Command 0CH

Purpose: Scan a box of pixels within an image buffer for the minimum and maximum pixel values.
Parameters:
buf (buffer)-source buffer to scan
x_offset (int)-leftmost pixel of box (value depends on the size of the image buffers, 0 thru max width -1)
y_offset (int)-top line of box (value depends on the size of the image buffers, 0 thru max height -1)
x_length (int)-width of box (value depends on the size of the image buffers, 1 thru max width)
y_length (int)-height of box (value depends on the size of the image buffers, 1 thru max height)
Response:ACK immediately plus buffered results
Results: (placed into global results buffer)
min_x (int)-x position of minimum
min_y (int)-y position of minimum
min_value (long)-minimum value
max_x (int)-x position of maximum
max_y (int)-y position of maximum
max_value (long)-maximum value
Status Values:
0=Idle, 1=Sent to foreground, 2=In progress
Notes:

- Use the **get_cpu_info** command to determine the maximum size of the image buffers.

ST-6 Cameras Only

- This command assumes the image buffers are 375 pixels wide by 242 pixels high. If applied after the **take_image** command has been used to capture an image with the 750 pixel wide readout mode then you will have to use the whole buffer as the source and translate the coordinates of the results to get the correct results.

sub_dark - Command 1CH

Purpose: Subtract the dark buffer from the light buffer, saving the results in the light buffer.

Response:ACK

Status Values:

0=Idle, 1=Sent to foreground, 2=In progress

Notes:

- This command adds a bias of 100 counts to the difference to aviod clipping pixels below zero.

Relay Commands

The commands in this section are used to control the relays within the CPU.

activate_relay - Command 0DH

Purpose: Activate or deactivate one or more relays for a period of time.

Parameters:

t_xplus (int)-x plus activation duration in hundredths of a second
t_xminus (int)-x minus activation duration in hundredths of a second
t_yplus (int)-y plus activation duration in hundredths of a second
t_yminus (int)-y minus activation duration in hundredths of a second
t_alarm (int)-alarm activation duration in hundredths of a second

Response:ACK

Status Values:

0=All idle, 1-31=Set of bits for active relays: x plus=8, x minus=4, y plus=2, y minus=1, alarm=16

Notes:

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

Temperature Regulation Commands

The commands in this section are used to control the CPU temperature regulation, which varies from camera model to camera model.

regulate_temp - Command 0EH

Purpose: Enable or disable the CPU temperature regulation.

Parameters:

enable (boolean)-enable temperature regulation when TRUE
setpoint (int)-temperature or thermistor setpoint in A/D units
samp_rate (int)-temperature sampling rate in hundredths of a second
p_gain (int)-proportional gain term
i_gain (int)-integral gain term
reset_brownout (boolean)-reset brownout detector when TRUE

Response:ACK

Status Values:

0=Idle, temperature regulation off, 1=Enabled

Notes:

- Since ST-4X cameras do not have temperature regulation, the settings of all parameters in this command except the reset_brownout parameter are ignored by ST-4X cameras.
- On power up the ST-5 and ST-6 CPUs read the current CCD temperature and start regulating the temperature at that setpoint. The ST-4X CPU ramps the TE cooler drive to the maximum drive level.

ST-5 Cameras Only

- Good values for the samp_rate, i_gain and p_gain parameters are 10, 164, and 1000 respectively.
- To convert from temperatures in °C to setpoints/thermistor readings in A/D units or vice versa use the following constants in the formulas at the end of this command:

T_0 = 25.0	R_RATIO = 9.1
R_0 = 3.0	R_BRIDGE = 9.09
DT = 50.0	MAX_AD = 8192

ST-6 Cameras Only

- Good values for the samp_rate, i_gain and p_gain parameters are 10, 200, and 1000 respectively.
- A bug in the Version 1.0 ROM exists where after disabling the temperature regulation you must wait until the CPU ramps the TE cooler drive down to zero (which can be monitored with the **get_temp_status** command) before reenabling the regulation.
- To convert from temperatures in °C to setpoints/thermistor readings in A/D units or vice versa use the following constants in the formulas at the end of this command:

T_0 = 25.0	R_RATIO = 9.1
R_0 = 3.0	R_BRIDGE = 27.0
DT = 50.0	MAX_AD = 65536

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

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

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

output_temp - Command 10H

Purpose: Output TE cooler drive value.

Parameters:

value (int)-temperature drive output (0 - max TE Drive)

Response:ACK

Status Values:

0=Idle, 2=In progress

Notes:

- Use the **get_cpu_info** command to determine the maximum allowed setting for the value parameter above. The CPUs will accept higher values but will clip the output value to the maximum.
- This command is functional only if the temperature regulation is disabled.

read_thermistor - Command 1DH

Purpose: Reads the thermistor A/D value and returns the result.

Response: (immediate packet)

thermistor (int)-digitized thermistor reading

Status Values:

0=Idle, 2=In progress

Notes:

- Since the ST-4X does not have temperature regulation capability, this command returns 0 for these cameras.
- See the Notes for the **regulate_temp** command for how to convert the thermistor reading to °C.

get_temp_status - Command 20H

Purpose: Gets the status of the temperature regulation.

Response: (immediate packet)

enabled (boolean)-temperature regulation enabled if TRUE
setpoint (int)-setpoint thermistor reading in A/D units
output (int)-TE cooler drive level (0-max TE Drive)
samp_rate (int)-temperature regulation sample rate in hundredths of a second
p_gain (int)-proportional gain term
i_gain (int)-integral gain term
brownout_detected (boolean)-TRUE if CPU detected a brownout and shut off the TE cooler drive

Status Values:

0=Idle, 2=In progress

Notes:

- Since ST-4X cameras do not have temperature regulation, these cameras will return with the enabled parameter set FALSE, and the setpoint, samp_rate, p_gain, and i_gain parameters set to 0.
- The CPU monitors the input voltage level and if it drops (brownout conditions) it will shut down the TE cooler and set the brownout_detected flag TRUE. If this occurs you will have to reenable the temperature regulation.

Aux Port Commands

Commands in this section deal with the auxiliary RS232/RS422 communications port on the CPU. The AUX port can be used to interface to other serial based instruments with the CPU acting as a gateway.

tx_to_aux - Command 11H

Purpose: Pass data to AUX port and relay the response.
Parameters:
data_1 (byte)-1st byte of data to transmit to AUX port
.
.
data_N (byte)-last byte of data to transmit to AUX port
Response:ACK
Status Values:
0=Idle, 1=Sent to foreground, 2=Sending, 3=Done sending, echoing
Notes:

- If data length is zero then the CPU shuts off transmission to the AUX port, and stops echoing. This is used to disable the AUX port after it has been enabled.
- This command dumps the AUX port's input buffer prior to sending the data and clears the error status bits.
- The AUX output buffer in the CPU is 1024 bytes long as well as the COM receive and transmit buffers. While the CPU can communicate to the Host over the COM port and to peripherals on the AUX port at different baud rates, the buffers will ultimately set the amount of data that can be passed back and forth per transaction.
- The **tx_to_aux** command violates the restriction that the CPU only transmits data to the host in response to a command from the host in that after issuing the **tx_to_aux** command the CPU continues to echo data received from the AUX port directly back to the host until disable with a **tx_to_aux** command with no data.

set_aux_control - Command 13H

Purpose: Set the baud rate, etc. of the AUX port.
Parameters:
baud (long)-baud rate to set AUX port to
control (int)-bits for least significant 5 bits in 80186EB's S1CON register
b0=mode bit 0
b1=mode bit 1
b2=mode bit 2
b3=parity enable when 1
b4=even parity when 1
Response:ACK
Status Values:
0=Idle, 2=In progress
Notes:

- At power up the CPU configures the AUX port for 9600 baud, 8 data bits, no parity, 1 stop bit. The 8 data bits and 1 stop bit represents mode 001 in bits b2-b0 in the control parameter above.

get_aux_status - Command 14H

Purpose: Reports status of errors on the AUX port.

Response: (immediate packet)

errs (int)-accumulated error bits from the AUX port
b2=overrun error when 1
b4=framing error when 1
b7=parity error when 1

Status Values:

0=Idle, 2=In progress

Notes:

- The error flags in the errs result are cleared by calling the **tx_to_aux** command.

pulse_out - Command 26H

Purpose: Sends pulses of a given width and period out the AUX port.

Parameters:

synchronous (boolean)-when this parameter is TRUE all interrupts in the CPU are disabled while the pulses are being generated for maximum pulse width and period integrity.
number_pulses (int)-number of pulses to generate, 0 cancels pulses in progress
pulse_width (int)-width of pulses in units of 0.435µs with a minimum of 43.5µs
pulse_period (int)-period of pulses in units of 0.435µs with a minimun of 108.75µs

Response:ACK

Status Values:

0=Idle, 2=In progress

Notes:

- Activation of this command only occurs if the **take_image**, **tx_to_aux** and **loopback_aux_test** commands are idle when this command is received.
- Receipt of a **take_image**, **tx_to_aux** or **loopback_aux_test** command cancels any **pulse_out** command in progress.
- If precise pulse width and pulse period control is required you should set the *synchronous* parameter TRUE but be aware that the CPU will not respond to any commands for the duration of a synchronous **pulse_out** command.

ST-6 Cameras Only

- This command is only available in ST-6 version 3.0 and later ROMS.

General Purpose Commands

This section describes general purpose commands within the CPU.

get_cpu_info - Command 25H

Purpose:Return the camera model and capabilities.

Response: (immediate packet)

version (int)-Version of the response to this command. This is set to 1 indicating the following format for the subsequent returned parameters:

cpu (enum)-Indicates the CPU type, 0=ST-4X, 1=ST-5, 2=ST-6

firmware_version (int)-This is the firmware version, interpreted as a four digit BCD number in the format XX.XX. For example a return result of 1234H would indicate a firmware version of 12.34.

name (32 chars)-NULL terminated text string identifying the camera

has_shutter (boolean)-TRUE if the camera has an integral electromechanical shutter and supports the open_shutter parameter in the **take_image** command and the **shutter_control** command

needs_offset (boolean)-TRUE if the camera requires using the **set_head_offset** command to initialize the optical head prior to issuing the **take_image** command

variable_dcs (boolean)-TRUE if the camera supports programmable double correlated sampling for the **take_image** command. If FALSE, the camera always operates with double correlated sampling.

variable_dcr (boolean)-TRUE if the camera supports programmable dc restore for the **take_image** command. If FALSE the camera does not require dc restore since the camera always uses double correlated sampling.

has_temp_control (boolean)-TRUE if the camera has temperature regulation and thus requires the use of the **regulate_temp** command. If FALSE the temperature control is open loop and is controlled with the **output_temp** command.

max_te_drive (int)-Indicates the maximum TE cooler drive value as used in the **output_temp** and **get_temp_status** commands

image_width (int)-Indicates the width of the image buffers

image_height (int)-Indicates the height of the image buffers

readout_modes (int)-Indicates the number of readout modes supported by the **take_image** command. For each of the supported modes, the following 6 parameters are given (up to a maximum of 20 readout modes):

mode (int)-Indicates the mode number you should supply to the **take_image** command in its readout_mode parameter to select this mode.

width (int)-Indicates the maximum readout width in pixels for this mode

height (int)-Indicates the maximum readout height in pixels for this mode

gain (int)-A four digit BCD number indicating the number of electrons per A/D unit, interpreted as XX.XX. For example a setting of 1096H indicates 10.96 electrons/count.

pixel_width (long)-An eight digit BCD number indicating the individual pixel width in microns in the format XXXXXX.XX. For example a setting of 00001375H indicates a pixel width of 13.75 microns.

pixel_height (long)-An eight digit BCD number indicating the individual pixel height in microns in the format XXXXXX.XX. For example a setting of 00001600H indicates a pixel height of 16.00 microns.

Status Values:

0=Idle, 2=In progress

Notes:
- The packet length of this response will vary from camera model to camera model as different models support different numbers of readout modes.

ST-6 Cameras Only
- This command is only available in ST-6 version 3.0 and latr ROMS.

get_activity_status - Command 05H

Purpose:Report activity status of any command to the Host.
Parameters:
command (int)-command to get status word for
Response: (immediate packet)
command (int)-command for which status is being reported
status (int)-status of above command
Status Values:
0=Idle, 2=In progress
Notes:
- Use this command to monitor the progress of any command that has been sent to the foreground for processing.

get_result_buf - Command 15H

Purpose: Send the contents of the result buffer to the Host.
Response: (immediate packet)
command (int)-command for which results were posted to the result buffer
results_1 (byte)-1st byte of result buffer
.
.
result_N (byte)-last byte of the result buffer
Status Values:
0=Idle, 2=In progress
Notes:
- Each command that posts results to the result buffer will have a different result length as described in those commands (for example the **cal_cent** command posts 3 longs or 12 bytes of data).

call_remote - Command 16H

Purpose: Call a remote function that had been previously uploaded to the CPU.
Parameters:
offset (int)-offset of remote function address
segment (int)-segment of remote function address
Response:ACK
Status Values:
0=Idle, 2=In progress

write_block - Command 17H

Purpose: Write a block of data into memory.
Parameters:
offset (int)-offset of destination address
segment (int)-segment of destination address
data_1 (byte)-1st byte of data to write to memory
.
.
data_N (byte)-last byte of data to write to memory
Response:ACK
Status Values:
0=Idle, 2=In progress
Notes:

- Do not send more than 1000 bytes of data at one time to avoid overflowing the COM receive buffer in the CPU which is 1024 bytes long.

read_block - Command 18H

Purpose: Read a block of data from memory.
Parameters:
offset (int)-offset of data address
segment (int)-segment of data address
length (int)-number of bytes to read
Response: (immediate packed)
data_1 (byte)-1st byte of data read from memory
.
.
data_N (byte)-last byte of data read from memory
Status Values:
0=Idle, 2=In progress
Notes:

- Do not request more than 1000 bytes of data at one time to avoid overflowing the COM transmit buffer in the CPU which is 1024 bytes long.

get_rom_version - Command 19H

Purpose: Report CPU internal firmware version.
Response:
firmware_version (int)-this is the firmware version
Status Values:
0=Idle, 2=In progress
Notes:

- Interpret the firmware version as a 4 digit BCD number with a two digit fraction (for example version 257 decimal = 0201H would be version 2.01)
- You should check the ROM version prior to issuing commands that are not supported by all ROMS. For example before you use the binning modes (readout_mode parameter) of the **take_image** command other than 0 and 1 you should check that the ROM supports those modes.

set_com_baud - Command 1AH

Purpose: Set the baud rate of the COM port for communications with the Host.
Parameters:
baud (long)-baud rate to set COM port to
Response:ACK
Status Values:
0=Idle, 2=In progress
Notes:

- The CPU sends the ACK at the old baud rate then switches to the new rate. You must then send and the CPU must receive a **get_rom_version** command within 1.0 second or the CPU will switch back down to 9600 baud.

reset - Command 1BH

Purpose: Perform a cold reset of the CPU.
Response:ACK
Notes:

- The CPU will restart communicating at 9600 baud just as if freshly powered up.

Test Commands

The commands in this section are for testing the CPU.

loopback_aux_test - Command 21H

Purpose: Test the AUX port for loopback capability.
Parameters:
baud (long)-baud rate to use in testing AUX port
Response:
sent (int)-number of characters sent out AUX port
errors (int)-number of errors in receiving data at AUX port
Status Values:
0=Idle, 2=In progress
Notes:

- For this test to pass the AUX port must be made to loop back upon itself with the RX tied to the TX.

7. Data Compression Algorithm

This section describes the data compression used by the CPU in response to the **get_line** and **put_line** commands. The data compression technique is essentially delta compression where each pixel is compared to the previous one sent and if within a small delta then the delta is sent as a byte rather than two bytes. This is a simple compression technique that can give at best 2:1 compression. The CPU's data compression algorithm is as follows:

1) The CPU transmits the first pixel uncompressed as a 16 bit unsigned integer, most significant byte first. It then uses the pixel value as the Base in the following.
2) The CPU calculates Delta = $Pixel_n$ - Base.
3) If -64 <= Delta <= 63 then the CPU transmits a single byte with the most significant bit (b7) clear and Delta in the least significant 7 bits. It then sets Base to the value of $Pixel_n$ and goes to the next pixel and step 2).
4) If -8192 <= Delta <= 8191 (14 bits) then the CPU transmits two bytes. In the first byte it sets the most significant bit (b7) and clears bit b6. In the remaining 6 bits it places the most significant 6 bits of Delta. In the second byte it places the least significant 8 bits of Delta. It then sets Base to the value of $Pixel_n$ and goes to the next pixel and step 2).
5) Otherwise the CPU transmits two bytes. First it calculates Val = $Pixel_n / 4$. In the first byte it sets the most significant two bits (b7 & b6), and in the least significant 6 bits it puts the most significant 6 bits of newly calculated Val. In the second byte it places the least significant bits of the newly calculated Val. Finally, it sets Base to Val*4 and goes to the next pixel and step 2).

The compressed data is buried within the packet response to the **get_line** command with an A5 start of packet, the get_line command byte, a two byte data length, the compressed data, and a two byte checksum.

8. The Image Buffers

The CPU contains 3 image buffers referred to as the dark buffer, the light buffer, and the accumulation buffer. The dark buffer is used to hold images taken with the shutter closed or the camera covered. The light buffer is used to hold images taken with the shutter open or the camera exposed to light. For best noise performance you should take a dark exposure then a light exposure and subtract the dark exposure from the light exposure to remove the fixed pattern and dark current noise associated with the dark image. The exposures should be taken at the same temperature and have identical exposure times. Finally, the accumulation buffer can be used for coadding light images. While the dark and light buffers hold 16 bit values read by the A/D converter, the accumulation buffer is 32 bits, allowing the accumulation of 65536 images without saturating.

The image buffers are tailored to the CCD in each model camera. The image buffers are treated as arrays by the CPU firmware with the height and width of the image array set by the height and width of the CCD in its highest resolution readout mode (smallest pixel size, largest number of pixels). This information is returned by the **get_cpu_info** command.

ST-6 Cameras Only

The image buffers are treated differently by the image capturing routines and the data processing routines. The **take_image** command treats the image buffers as arrays with 750

columns and 121 rows for the 750 pixel wide readout modes and treats the image buffers as arrays with 375 columns and 242 rows for all other readout modes (375 pixels wide and 250 pixels wide). The image processing routines always treat the image buffers as arrays with 375 columns and 242 rows. This slightly complicates downloading images taken with the 750 pixel wide readout modes in that you have to download two 375 pixel lines to get a single 750 pixel line. The first line downloaded is the left portion of the 750 pixel line and the second line downloaded is the right portion.

9. Sample Instruction Sequences

This section describes common tasks for the CCD cameras and the required commands software packages supporting these cameras will need to perform those tasks.

Establishing a Communications Link

The first task any software package that communicates with the SBIG CCD cameras will need to do is establish a communications link with the CPU. This is a two step test: Communicating with the CPU and then switch the CPU to the fastest possible baud rate. Since the state of the CPU can be unknown, you may have to hunt around looking for the camera at different baud rates. We suggest using the following sequence to find the CPU:

1. Send the CPU a **get_rom_version** command at 9600 baud or at the previous baud rate you had used in communicating with the camera if you had previously established a communications link. If you get a valid response packet back, including a correct checksum, then you are done with the first part of the task. Otherwise try set 2.
2. Delay for a second (to allow the CPU to sync up to the byte stream), change your baud rate (the host computer) and then send the CPU a **get_rom_version** command, looking for a valid response. The baud rates you should try are 9600 (which the CPU powers up at) and any other baud rate you may have changed the CPU to. Since the CPU will only change baud rates at the request of the Host you only need to scan these baud rates. If the CPU does not respond at any baud rate then you should warn the user that the camera could not be found and as a last ditch effort you might try resetting the CPU (this shouldn't be necessary).

The second part of establishing a communications link with the CPU involves changing the CPU's baud rate to the highest speed that the Host and CPU can communicate at with good results (no communications errors). We suggest using the following sequence to reprogram the CPU's speed:

1. Using the baud rate that you were able to find the CPU with above, send the CPU a **set_com_baud** command, trying to change the CPU's baud rate to the fastest rate your software or the Host can support. The CPU will send back the ACK to the **set_com_baud** command at the old baud rate and then switch to the new rate.
2. Change the Host to the new baud rate and send the CPU a **get_rom_version** command at the new baud rate. This must be done within 1 second of step 1 above or the CPU will switch back down to 9600 baud.
3. As a final check that communications will operate reliably at the new, higher baud rate, you may want to send the CPU several **get_line**

commands and check for a valid response. The **get_line** command sends packets with hundreds of bytes, and if the communications are marginal you're probably going to see it. If the communications at the higher baud rate are unreliable then you should lower the baud rate one notch.

Finally, you should send the CPU a **get_cpu_info** command to determine the type of camera and the camera's capabilities to insure those items are compatible with your software. If the model CPU is not the one you're expecting then don't continue sending commands. Unfortunately the **get_cpu_info** command is not present in ST-6 cameras with ROMs before version 3.0. If you get a CAN back from the camera in response to the **get_cpu_info** command then it is safe to assume it is and ST-6.

Determining the Head Offset (ST-6 Cameras Only)

The ST-6 cameras have an electronic offset adjust that needs to be programmed before taking images. ST-4X and ST-5 CPUs have an auto-offset feature and don't require this procedure. The electronic offset adjust corrects for variations from CCD to CCD. Once a particular ST-6's offset has been determined, it should not vary much over time or temperature (if it changes at all it will only change one or two counts). You should determine the offset and retain the required setting to speed up the procedure on future occasions. We suggest you do the following sequence right after you have established a link with the ST-6:

1. Send the ST-6 a **read_blank_video** command with the *enable_dcs* parameter set TRUE and the *head_offset* parameter set to 175 or the previous level you had determined was correct.
2. If the returned *video* parameter in the ST-6's response is between 1000 and 10,000 then you are through. If the *video* is below 1000 then increase the *head_offset* parameter by one and reissue the **read_blank_video** command, otherwise the *video* is above 10,000 and you should decrease the *head_offset* parameter by one and reissue the **read_blank_video** command.
3. Use the value of the *head_offset* parameter that resulted in a *video* reading between 1,000 and 10,000 with the **set_head_offset** command prior to using the **take_image** command to take images. You should retain the correct setting and use that as your starting point next time you need to adjust the head offset.

You may actually want to average several readings from the **read_blank_video** command, and should be aware that bright illumination can effect the results. While the ST-6 closes its integral mechanical vane while executing the **read_blank_video** command, it won't hold off direct illumination with a bright light.

Taking an Exposure

Taking an image with the CPU can involve several steps as outlined below:

1. If the CPU is an ST-6 then you should use the **set_head_offset** command to initialize the electronic offset in the ST-6 head prior to each exposure.
2. Depending on the results of the **get_cpu_info** command you may want to program the CPU for different readout modes which trade off image spatial resolution with digitization and download time.

3. Send the CPU a **take_image** command with the parameters set as required based upon the readout mode, etc.

4. On a periodic basis (like three times a second) send the CPU a **get_activity_status** status command for the **take_image** command to determine when the CPU has finished acquiring the image. Don't interrogate the CPU faster hat 3 times per second to allow the CPU to dedicate the majority of its processing power to acquiring the image which can get processing intensive during the readout phase.

Downloading an Image

Use the **get_line** or **get_uncompressed_line** commands to download each row of the image once the **take_image** command has finished its processing. The reason the image is downloaded a line at a time is in case an error occurs in receiving the data at the Host end, the CPU needs only send the line over again, not the entire image.

Sending Data to an Instrument on the AUX Port

The CPU has an AUX serial port that can be configured by the Host to talk to other instruments, with the CPU acting as a gateway. You can talk to other RS-232/RS-422 instruments on the AUX port with a sequence outlined below:

1. Send the CPU a **set_aux_control** command to initialize the AUX port's baud rate to that required by the other instrument.

2. Send the CPU a **tx_to_aux** command with the data to send to the other instrument buried in the **tx_to_aux** command. The ST-6 will send the data out the AUX port, and echo any response from the instrument back to the Host. Note that the echoed data will not be buried in a packet, but will just be the raw data received by the CPU through its AUX port.

3, To send additional data to the instrument continue to issue **tx_to_aux** commands to the CPU. When you're done talking to the other instrument send the CPU a final **tx_to_aux** command with no data to get the CPU to shut down the AUX port echoing.