Serial Commands

The serial communication parameters are as follows:

- 1,200 to 115,200 Baud
- 8 Data bits
- 1 Stop bit
- No Parity
- No Flow Control (Not Xon/Xoff or Hardware)

All commands are sent using visible ASCII characters (123 is 3 bytes “123”). Upon a successful transmission of a command, the ACK string should be returned by the system. If there was a problem in the syntax of the transmission, or if a detectable transfer error occurred, a NCK string is returned. After either an ACK or a NCK, a \r is returned. When a prompt (’\r’ followed by a ‘:’) is returned, it means that the camera is waiting for another command in the idle state. White spaces do matter and are used to separate argument parameters. The \r (ASCII 13 carriage return) is used to end each line and activate each-command. If visible character transmission causes too much overhead, it is possible to use varying degrees of raw data transfer.

See Raw Mode on page 48 for information on configuring ascii vs raw text packets.

Functionally Grouped Command Listing

Buffer Commands

Camera Module Commands

Data Rate Commands

Servo Commands

Image Windowing Commands

Auxiliary I/O Commands

Color Tracking Commands

Histogram Commands

Frame Differencing Commands

Color Statistics Commands

System Level Commands

Alphabetical Command Listing

\r

This command is used to set the camera board into an idle state. Like all other commands, you should receive the acknowledgment string “ACK” or the not acknowledge string “NCK” on failure. After acknowledging the idle command the camera board waits for further commands, which is shown by the ‘:’ prompt. While in this idle state a \r by itself will return an “ACK” followed by \r and : character prompt. This is how you stop the camera while in streaming mode.

Example of how to check if the camera is alive while in the idle state:

```
:
ACK
:
```

BM active \r

This command sets the mode of the CMUcam’s frame buffer. A value of 0 (default) means that new frames are constantly being pushed into the frame buffer. A value of 1, means that only a single frame remains in the frame buffer. This allows multiple processing calls to be applied to the same frame. Instead of grabbing a new frame, all commands are applied to the current frame in memory. So you could get a histogram on all three channels of the same image and then track a color or call get mean and have these process a single buffered frame. Calling **RF** will then read a new frame into the buffer from the camera. When **BM** is off, **RF** is not required to get new frames.

See RF on page 47 to read a new frame when buffer mode is enabled.

Example of how to track multiple colors using buffer mode:

Processing on an already buffered image is much faster than processing a new image.

```
:BM 1
ACK
:PM 1
ACK
:TC 200 240 0 30 0 30
ACK
T 20 40 10 30 30 50 20 30
:RF
ACK
:TC 0 30 200 240 0 30
ACK
T 30 50 20 40 40 60 22 31
```

CR [reg1 value1 [reg2 value2 ... reg16 value16]]\r

This command sets the **C**amera's internal **R**egister values directly. The register locations and possible settings can be found in the Omnivision CMOS camera documentation. All the data sent to this command should be in decimal visible character form unless the camera has previously been set into raw mode. It is possible to send up to 16 register-value combinations. Previous register settings are not reset between CR calls; however, you may overwrite previous settings. Calling this command with no arguments resets the camera and restores the camera registers to their default state. This command can be used to hard code gain values or manipulate other low level image properties.

See page 25 for more information on YCrCb color space.

Register		Value	Effect
5	Contrast	0-255	
6	Brightness	0-255	
18	Color Mode		
		36	YCrCb Auto White Balance On
		32	YCrCb Auto White Balance Off
		44	RGB Auto White Balance On
		40	*RGB Auto White Balance Off
17	Clock Speed		
		0	*50 fps
		1	26 fps
		2	17 fps
		3	13 fps
		4	11 fps
		5	9 fps
		6	8 fps
		7	7 fps
		8	6 fps
		10	5 fps
19	Auto Exposure		
		32	Auto gain off
		33	*Auto gain on

* indicates the default state

Example of switching into YCrCb mode with White Balance off

```
:CR 18 32
ACK
:
```

CP boolean \r

This command toggles the **C**amera module's **P**ower. A value of 0, puts the camera module into a power down mode. A value of 1 turns the camera back on while maintaining the current camera register values. This should be used in situations where battery life needs to be extended, while the camera is not actively processing image data. Images in the frame buffer may become corrupt when the camera is powered down.

See SL and SD on pages 49 and 50 to decrease camera power consumption even more.

CT boolean \r

This command toggles the **C**amera **T**ype while the camera is in slave mode. Since the CMUcam2 can not determine the type of the camera without communicating with the module, it is not possible for it to auto-detect the camera type in slave mode. A value of 0, sets the CMUcam2 into ov6620 mode. A value of 1 sets it into ov7620 mode. The default slave mode startup value assumes the ov6620.

See slave mode on page 22.

DC value \r

This command sets the **C**hannel that is used for frame **D**ifferencing commands. A value of 0, sets the frame differencing commands LF and FD to use the red (Cr) channel. A value of 1 (default) sets them to use the green (Y) channel, and 2 sets them to use the blue (Cb) channel.

See LF and FD on page 39 and page 34.

DM value \r

This command sets the **D**elay **M**ode which controls the delay between characters that are transmitted over the serial port. This can give slower processors the time they need to handle serial data. The value should be set between 0 and 255. A value of 0 (default) has no delay and 255 sets the maximum delay. Each delay unit is equal to the transfer time of one bit at the current baud rate.

DS x_factor y_factor \r

This command allows **D**own **S**ampling of the image being processed. An x_factor of 1 (default) means that there is no change in horizontal resolution. An x_factor of 2, means that the horizontal resolution is effectively halved. So all commands, like send frame and track color, will operate at this lower down sampled resolution. This gives you some speed increase and reduces the amount of data sent in the send frame and bitmap linemodes without clipping the image like virtual windowing would. Similarly, the y_factor independently controls the vertical resolution. (Increasing the y_factor downsampling gives more of a speed increase than changing the x_factor.) The virtual window is reset to the full size whenever this command is called.

Example of down sampling the resolution by a factor of 2 on both the horizontal and vertical dimension.

```
:DS 2 2
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

FD threshold \r

See LF on page 39 to load a new baseline frame to difference off of.

This command calls **F**rame **D**ifferencing against the last loaded frame using the **LF** command. It returns a type T packet containing the middle mass, bounding box, pixel count and confidence of any change since the previously loaded frame. It does this by calculating the average color intensity of an 8x8 grid of 64 regions on the image and comparing those plus or minus the user assigned *threshold*. So the larger the threshold, the less sensitive the camera will be towards differences in the image. Usually values between 5 and 20 yield good results. (In high resolution mode a 16x16 grid is used with 256 regions.)

See MD on page 44 to see how to reduce motion noise.

FS boolean \r

This command sets the **F**rame **S**treaming mode of the camera. A value of 1, enables frame streaming, while a 0 (default) disables it. When frame streaming is active, a send frame command will continuously send frames back to back out the serial connection.

See SF on page 50.

GB \r

See Demo Mode on page 15.

This command **G**ets a **B**utton press if one has been detected. This command returns either a 1 or a 0. If a 1 is returned, this means that the button was pressed sometime since the last call to Get Button. If a 0 is returned, then no button press was detected.

GH <channel> \r

See HC and HT commands on pages 38 and 39.

This command **G**ets a **H**istogram of the *channel* specified by the user. The histogram contains 28 bins each holding the number of pixels that occurred within that bin's range of color values. So bin 0 on channel 0 would contain the number of red pixels that were between 16 and 23 in value. If no arguments are given, get histogram uses the last channel passed to get histogram. If get histogram is first called with no arguments, the green channel is used. The value returned in each bin is the number of pixels in that bin divided by the total number of pixels times 256 and capped at 255.

GI \r

This command **G**ets the auxiliary I/O **I**nput values. When get inputs is called, a byte is returned containing the values of the auxiliary IO pins. This can be used to read digital inputs connected to the auxiliary I/O port. The aux I/O pins are internally lightly pulled high. See page 22 for pin numbering. Note that the pins are pulled up internally by the processor.

Example of how to read the auxiliary I/O pins. (in this case, pins 0 and 1 are high, while pins 2 and 3 are low).

```
:GI
3
ACK
:
```

GM \r

This command will **G**et the **M**ean color value in the current image. If, optionally, a subregion of the image is selected via virtual windowing, this function will only operate on the selected region. The mean values will be between 16 and 240 due to the limits of each color channel on the CMOS camera. It will also return a measure of the average absolute deviation of color found in that region. The mean together with the deviation can be a useful tool for automated tracking or detecting change in a scene. In YCrCb mode RGB maps to CrYCb.

See page 45 to see how the OM command can create a custom S Packet.

This command returns a Type S data packet that by default has the following parameters:

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation\r

Example of how to grab the mean color of the entire window:

```
:SW 1 1 40 143
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

GS servo \r

This command will **G**et the last position that was sent to the **S**ervos.

See SV command on page 53.

Example of how to use get servo:

```
:GS 1
ACK
128
:
```

GT \r

This command **G**ets the current **T**rack color values. This is a useful way to see what color values track window is using.

This example shows how to get the current tracking values:

```
:TW
 ACK
 T 12 34 ....
:GT
ACK
200 16 16 240 20 20
:
```

GV \r

This command **G**ets the current **V**ersion of the firmware and camera module version from the camera. It returns an ACK followed by the firmware version string. c6 means that it detects an OV6620, while c7 means that it detected an OV7620.

Example of how to ask for the firmware version and camera type:

```
:GV
ACK
CMUcam2 v1.00 c6
```

GW \r

This command **G**ets the current virtual **W**indowing values. This command allows you to confirm your current window configuration. It returns the x1, y1, x2 and y2 values that bound the current window.

HC #_of_bins scale \r

See GH on page 35 to see how to get histograms.

This command lets you **C**onfigure the **H**istogram settings. The first parameter takes one of three possible values. A value of 0 (default) will cause GH to output 28 bins. A value of 1 will generate 14 bins and a value of 2 will generate 7 bins. The scale parameter (default 0) allows you to better examine bins with smaller counts. Bin values are scaled by 2^{scale} where scale is the second parameter of the command.

#_of_bins

Input	Bins
0	28
1	14
2	7

HD boolean \r

See LF and FD on pages 39 and 34 to see how to use the more basic frame differencing commands.

This command enables or disables **H**iRes frame **D**ifferencing. A value of 0 (default) disables the high resolution frame differencing mode, while a value of 1 enables it. When enabled, frame differencing will operate at 16x16 instead of 8x8. The captured image is still stored internally at 8x8. The extra resolution is achieved by doing 4 smaller comparisons against each internally stored pixel. This will only yield good results when the background image is relatively smooth, or has a uniform color.

HR state \r

This sets the camera into **HiR**es mode. This is only available using the OV6620 camera module. A *state* value of 0 (default) gives you the standard 88x143, while 1 gives you 176x287. HiRes mode truncates the image to 176x255 for tracking so that the value does not overflow 8 bits.

HT boolean \r

See GH on page 35 to see how to get a histogram.

This command enables or disables **H**istogram **T**racking. When histogram tracking is enabled, only values that are within the color tracking bounds will be displayed in the histograms. This allows you to select exact color ranges giving you more detail, and ignoring any other background influences. A value of 0 (default) will disable histogram tracking, while a value of 1 will enable it. Note that the tracking noise filter applies just like it does with the TC and TW commands.

L0 boolean \r
L1 boolean \r

These commands enable and disable the two tracking **L**EDs. A value of 0 will turn the LED off, while a value of 1 will turn it on. A value of 2 (default) will leave the LED in automatic mode. In this mode, LED 1 turns on when the camera confidently detects an object while tracking and provides feedback during a send frame. In automatic mode, LED 0 does nothing, so it can be manually set.

LF \r

See FD on page 34.

This command **L**oads a new **F**rame into the processor's memory to be differenced from. This does not have anything to do with the camera's frame buffer. It simply loads a baseline image for motion differencing and motion tracking.

LM type mode \r

This command enables **L**ine **M**ode which transmits more detailed data about the image. It adds prefix data onto either **T** or **S** packets. This mode is intended for users who wish to do more complex image processing on less reduced data. Due to the higher rate of data, this may not be suitable for many slower microcontrollers. These are the different types and modes that line mode applies to different processing functions:

Type	Mode	Effected Command	Description
0	0	TC TW	Default where line mode is disabled
0	1	TC TW	Sends a binary image of the pixels being tracked
0	2	TC TW	Sends the Mean, Min, Max, confidence and count for every horizontal line of the tracked image.
1	0	GM	Default where line mode is disabled
1	1	GM	Sends the mean values for every line in the image
1	2	GM	Sends the mean values and the deviations for every line being tracked in the image
2	0	FD	Default where line mode is disabled
2	1	FD	Returns a bitmap of tracked pixels much like type 0 mode 0 of track color
2	2	FD	Sends the difference between the current image pixel value and the stored image. This gives you delta frame differenced images.
2	3	LF FD	This gives you the actual averaged value for each element in a differenced frame. It also returns these values when you load in a new frame. This can be used to give a very high speed gray scale low resolution stream of images.

Note, that the “mode” of each “type” of linemode can be controlled independently.

Line Mode Type 0: Track Color

Mode 1: Bitmap of tracked region

When the linemode type is 0 and the mode is set to 1, TC or TW will send a binary bitmap of the image as it is being processed. It will start this bitmap with an 0xAA flag value (hex value AA not in human readable form) followed by the Xsize and Ysize of the binary image. The value 0xAA will not occur in the data stream. This is followed by bytes each of which contains the binary value of 8 pixels being streamed from the top-left to the bottom-right of the image. The bits for each row are padded with zeros to fill an integral number of bytes. The binary bitmap is terminated by two 0xAA's. This is then followed by the normally expected standard **T** data packet.

See TC on page 53.

Example of TC with line mode on:

```
:LM 0 1
ACK
:TC
ACK
(raw data: AA Xsize Ysize XX XX …. XX XX XX AA AA) T 55 90 45 72 65 106 18 51
(raw data: AA Xsize Ysize XX XX …. XX XX XX AA AA) T 55 90 46 72 65 106 18 52
```

Mode 2: Per row statistics in the tracked region

When the linemode type is 0 and the mode is set to 2, TC or TW will send various statistics about each row that is being tracked. It sends the minimium x value, the maximium x value, the average x value, the count of tracked pixels on that line and the confidence. This can be especially useful for line following applications since you can essentially get a trace of the middle of the line. Like other linemode options, this new data is sent as a prefixed packet. The packet starts with an 0xFE, followed by the number of rows (the y-size) that it will send. The packet will then contain, the xLineMean, xLineMin, xLineMax, line pixel count, and line confidence for each row. These will all be sent as raw values. The packet terminates with a 0xFD followed by a normal T packet.

See OM on page 45 to see how to mask these line mode data packets.

0xFE y-size xLineMean xLineMin xLineMax LineCount Conf ... 0xFD Tpacket

Line Mode Type 1: Get Mean

Mode 1: Per line statistics

See GM on page 36.

When the linemode type is 1 and the mode is set to 1, GM will send a raw (not human readable) mean value of every line being processed. These packets start with an 0xFE. The data is sent in the following raw format rLineMean, gLineMean, bLineMean, and terminate with an 0xFD.

0xFE Rmean Gmean Bmean ... 0xFD Mpacket

Example of GM with line mode on

```
:LM 1 1
ACK
:GM
ACK
(raw data: FE XX XX XX … XX XX XX FD) M 45 56 34 10 15 8
(raw data: FE XX XX XX … XX XX XX FD) M 45 56 34 10 15 8
```

See OM on page 45 to see how to mask these line mode data packets.

Mode 2: More per line statistics

When the linemode type is 1 and the mode is set to 2, GM will send a raw (not human readable) mean value and deviation for every line being processed. These packets are started with an 0xFE. The data is sent in the following raw format rLineMean, gLineMean, bLineMean, rDeviation, gDeviation, bDeviation and terminate with an 0xFD.

0xFE Rmean Gmean Bmean Rdev Gdev Bdev ... 0xFD Mpacket

Line Mode Type 2: Frame Differencing

Mode 1: Bitmap for tracked pixels

See FD on page 34.

When the linemode type is 2 and the mode is set to 1, FD will send a binary bitmap of the image as it is being processed. It will start this bitmap with an 0xAA flag value (hex value AA not in human readable form) followed by the Xsize and Ysize of the binary image. The value 0xAA will not occur in the data stream. This is followed by bytes each of which contains the binary value of 8 (or 16) pixels being streamed from the top-left to the bottom-right of the image. The binary bitmap is terminated by two 0xAA's. This is then followed by the normally expected standard **T** data packet.

Example of TC with line mode on:

```
:LM 2 1
ACK
:FD 10
ACK
(raw data: AA XX XX XX … XX XX XX AA AA) T 5 10 4 9 8 100
(raw data: AA XX XX XX … XX XX XX AA AA) T 5 10 4 9 8 100
```

Mode 2: Deltas between reference frame

When the linemode type is 2 and the mode is set to 2, FD will send the values of the differences between the current image and the original saved frame. The packet starts with 0xFC followed by the xSize and ySize of the image buffer that is to be sent. A single value for each pixel is transmitted and the packet ends with an 0xFD. The delta values are capped at +/- 112 with 128 added to the delta, so 128 means zero difference. This forces the values to remain in the 16-240 range.

```
:LM 2 2
ACK
:FD 10
ACK
(raw data: FC xSize ySize  XX XX XX … XX XX XX FD)
(raw data: FC xSize ySize  XX XX XX … XX XX XX FD)
```

Mode 3: Deltas between reference frame

When the linemode type is 2 and the mode is set to 3, LF and FD will send a binary bitmap of the internally stored image that they are operating on. This image is stored in the same format as mode 2 of frame differencing.

MD threshold \r

See FD on page 34.

This command is almost identical to FD except that it **M**asks the first frame it **D**ifferences on. Any motion detected on the first frame is masked out, so that areas with high amounts of noise are ignored. Basically, if you call frame differencing and there is always an area of the frame that is moving, then MD will mask out that portion of the image so subsequent calls to FD will ignore that portion of the image. Calling the LF command will clear any masked pixels.

NF threshold \r

This command controls the **N**oise **F**ilter setting. It accepts a value that determines how many consecutive active pixels before the current pixel are required before the pixel should be detected (default 2). The range is between 0 and 255.

Example of how to turn off noise filtering:

```
:NF 0
ACK
:
```

OM packet mask \r

This command sets the **O**utput **M**ask for various packets. The first argument sets the type of packet:

#	**Tracking Type**	**Packet**
0	Track Color	T
1	Get Mean	S
2	Frame Difference	T
3	Non-tracked packets*	T
4	Additional Count Information**	T, H
5	Track Color Line Mode 2	T
6	Get Mean Line Modes 1 and 2	S

The *mask* should be a single byte that represents the bitwise mask of the tracking packet. So a value 255 would allow all the parameters to be printed, while a value of 3 would only allow the first two parameters to be printed. Each mask for each paket type is stored separately and remains set until the camera is reset.

*Non-Tracked packets are packets that are printed when the object being tracked is not detected. If this is set to 0, then no packet is printed when the object is not found. If this is set to 1, then just a "T 0" is sent when no object is found. If this is set to 2 (default) then the packet is identical to a tracked packet of that type.

**The additional count information flag lets you get access to the full 16 bit count values for color tracking or histogramming. A value of 0 (default) disables the 16 bit values. A value of 1, adds the 16 bit count of tracked pixels in 2 separate bytes, the first for the LSB and the second for the MSB. A value of 2 will add a 16 bit count of all pixels used to generate a histogram as the first two bytes following the H in the histogram packet. A value of 3, enables both modes simultaneously.

Example of how to only show Mx and My in a T packet:

```
:OM 0 3
ACK
:TC 200 230 0 30 0 30
T 23 45
```

PD boolean \r

See TI on page 53 to find out how to use inverse tracking for better edge following.

This command enables the **P**ixel **D**ifference mode. By default, the mode is off. A value of 1 causes the difference between the current pixel and the previous pixel to be used by all processing commands instead of the original pixel value. This essentially does a horizontal edge detecting convolution on the image. So the intensity of the remaining lines in each channel is proportional to the sharpness of an edge found in that channel. The best way to understand this command is to try enabling pixel differencing and try sending a frame. Notice what types of lines appear stronger than others. You can then track these edges based on their intensity using track color etc. The difference values are capped at +/- 112 with 128 added to each delta so a value of 128 indicates a 0 difference. This forces the values to remain between 16 and 240. This command applies to all commands.

PF boolean \r

This command enables the **P**acket **F**iltering mode. By default, the mode is off. A value of 1 makes it so that only the first empty packet when a tracked object disappears from the screen is displayed. No packets will be transmitted until the object returns into view. This command can help in situations where empty packets may unnecessarily tax the host processor.

PM mode \r

This command puts the board into **P**oll **M**ode. Setting the mode parameter to 1 engages poll mode while 0 (default) turns it off. When poll mode is set to 0, a continous stream of packets is returned from a processing function. When poll mode is set to a value of 1, only one packet is returned when an image processing function is called. If mode is set to a value of 2, then poll mode will wait until an object is tracked and then return. This could be useful if you would like to rapidly change parameters or if you have a slow processor that can't keep up with a given frame rate.

Example of how to get one packet at a time:

```
:PM 1
ACK
:TC 50 20 90 130 70 255
ACK
C 38 82 53 128 35 98
:
```

PS number \r

This command controls if **P**ackets should be **S**kipped or not. The default value is 0, which means that all packets will be transmitted. A value of 1 means that every other packet will be skipped. A value of 2 means that only every second packet will be displayed etc. This is useful if you need to slow down the data rate so that your processor can keep up with the data stream when poll mode is enabled.

RF \r

See BM on page 30.

This command **R**eads a new **F**rame into the buffer. This should only be used to get new data when using buffer mode (**BM**). The frame buffer is what allows multiple pass image processing on a single frame. While in buffer mode, you are constantly reprocessing the same frame until read frame is called. Under normal non-buffer mode operation, a new frame is loaded right before a processing function is called.

RM bit_flags \r

This command is used to engage the **R**aw serial transfer **M**ode. It reads the bit values of the first 3 (lsb) bits to configure settings. All bits cleared sets the default visible ASCII mode. If bit 0 is set, then all output from the camera is in raw byte packets. The format of the data packets will be changed so as not to include spaces or be formatted as readable ASCII text. Instead you will receive a 255 valued byte at the beginning of each packet, the packet identifying character (i.e. C for a color packet) and finally the packet data.There is no \r sent after each packet, so you must use the 255 to synchronize the incoming data. Any 255 valued bytes that may be sent as part of the packet are set to 254 to avoid confusion. If bit 1 is set, the "ACK\r" and "NCK\r" confirmations are not sent. If bit 2 is set, input will be read as raw byte values, too. In this mode, after the two command byte values are sent, send 1 byte telling how many arguments are to follow. (i.e. DF followed by the raw byte value 0 for no arguments) No \r character is required.

bit_flags = B2 B1 B0

B0	Output from the camera is in raw bytes
B1	"ACK\r" and "NCK\r" confirmations are suppressed
B2	Input to the camera is in raw bytes

Example of the new packet for Track Color with Raw Mode output only:

```
:RM 1
ACK
:TC 50 100 30 90 0 30
ACK
T>#$%KFDSAG@#$
```

RS \r

This command **R**e**S**ets the vision board. Note, on reset the first character is a \r. Also keep in mind that all register values are reset to their default state.

Example of how to reset the camera:

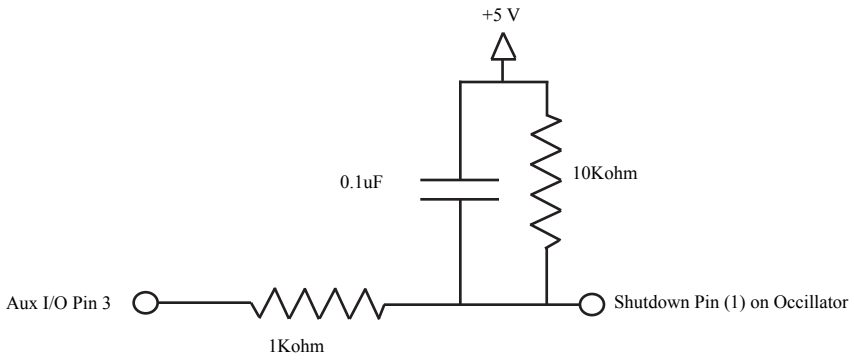
```
:RS
ACK

CMUcam2 v1.0 c6
:
```

SD \r

This command **S**leeps the camera **D**eeply to save power. This command puts the processor to sleep just as the SL command does and aditionally uses one of the auxilary I/O pins to sleep the oscillator. Wakeup from this mode is achieved by sending any character to the module, typically '\r'. The oscillator needs to shut off slightly later than the processor to ensure that that processor powers down correctly. To achieve this delay, it is neccessary to add a pullup resistor on the enable line of the oscillator and then have a resistor and capacitor in series with each other before being connected to auxiliary I/O pin 3. You will need to connect 1K series resistor between the oscillator's enable pin and the aux IO pin 3. You then need to connect a 10K resistor in parallel with a 0.1uF capacitor between the enable pin on the oscillator and +5 volts. See diagram below.

See SL on page 50, for a faster, more basic sleep command.



SF [channel] \r

This command will **S**end a **F**rame out the serial port to a computer. This is the only command that will by default only return a non-visible ASCII character packet. It dumps a type F packet that consists of the raw video data row by row with a frame synchronize byte and a column synchronize byte. (This data can be read and displayed by the CMUcam2GUI java application.) To get the correct aspect ratio, double each column of pixels. Since the image is being read from a buffer, the image resolution is not dependent on baud rate. The baud rate just controls how fast the image will be transmitted. Optionally, a channel (0-2) can be added to the command which causes send frame to only send that channel. This will effectively transmit one third of the data.

See FS on page 34, to find out how to stream frames.
See DS on page 33, to find out how to reduce data sent by send frame.

Type F data packet format
1 - new frame followed by X size and Y size
2- new col
3 - end of frame
RGB (CrYCb) ranges from 16-240
1 xSize ySize 2 r g b r g b ... r g b r g b 2 r g b r g b ... r g b r g b 3

SL active \r

This command enables **SL**eep mode by putting the processor to sleep. Sleep mode can be used when the camera is not needed in order to save power. Sending any character wakes the camera back up after a delay of up to 10ms. It is best to use '\r' to wake the camera up since this will ensure that no unforeseen command gets executed. Sleeping will disable the servo outputs.

For greater power saving see the SD and CP commands on pages 49 and 32.

SM bit_flags \r

This command sets the **S**ervo **M**ask on the CMUcam. The servo mask controls which automatic servo axes are active and which ones should report their values at the end of tracking packets. Pan and Tilt enable / disable turn off the respective automatic servo function while tracking. The servo reporting is added after all of the normal outputs in the Tracking packet, but before the final "\r". Note that automatic control only operates with 'T' packets returned by TC and TW commands.

bit_flags = B3 B2 B1 B0

B0	Pan Control Enable
B1	Tilt Control Enable
B2	Pan Report Enable
B3	Tilt Report Enable

Example of how to enable both pan and tilt automatic servoing and both pan and tilt reporting. In this case, since it doesn't see the object, the servos stay at position 128:

```
:SM 15
ACK
:TC
ACK
T 0 0 0 0 0 0 0 0 128 128
```

SO servo_number level \r

This command sets a **S**ervo **O**utput on the CMUcam to be either a constant high or low value. This essentially converts the servo outputs to be standard TTL digital outputs. The servo number (0-4) selects which servo you want to control, and a level value of either 1 or 0 switches between 5 and 0 volts. If a servo is connected and the output is set to 0, the servo is effectively turned off.

SP [pan_range_far pan_range_near pan_step
tilt_range_far tilt_range_near tilt_step] \r

This command sets the **S**ervo **P**arameters that are used by the automatic tracking control law. Changing these values can help you tune your tracking for a particular servo setup. The automatic servoing uses a two stage "bang-bang" control law. When the pixel value is greater than the "far" range, the related servo will move by the step amount. When the pixel value is between the near and far range, the servo will move by half of the step amount. Any value smaller than the near value is part of the dead zone and will not trigger any servo motion.

See page 21 for pan tilt direction jumpers.

See the SM command on page 51.

Variable	Description	Default
pan_range_far	Pixel distance needed to do a large pan step	16
pan_range_near	Pixel distance needed to do a small pan step	8
pan_step	Servo position change of a long pan step	5
tilt_range_far	Pixel distance needed to do a large tilt step	30
tilt_range_near	Pixel distance needed to do a small tilt step	15
tilt_step	Servo position change of a long tilt step	5

ST Rmin Rmax Gmin Gmax Bmin Bmax \r

This command allows you to **S**et **T**racking parameters without actually calling track color. These values can then be stored until you might call TC with no arguments later.

Example of how to use ST:

```
:ST 200 0 0 250 20 20
ACK
:TC
ACK
T 6 55 2 40 12 60 10 70
T 6 55 2 41 12 61 11 70
```

SV servo position \r

See SO on page 51 to learn how to disable the servos.

This command lets you set the position of one of five **S**er**V**os. The servos have an active region of between 46 and 210. A value of 128 is the center and generates a 1500 us pulse. The pulse increments by 8.68us and covers a range from 400 us to 1820 us.

Example to set servo 1 to position 200:

```
:SV 1 200
ACK
:
```

TC [Rmin Rmax Gmin Gmax Bmin Bmax] \r

See page 45 to see how the OM command can create a custom S Packet.

Using VW on page 55 to decrease the vertical resolution will allow 50 fps tracking.

This command begins to **T**rack a **C**olor . It takes in the minimum and maximum RGB (CrYCb) values and outputs a type T packet. This packet by default returns the middle mass x and y coordinates, the bounding box, the number of pixels tracked, and a confidence value. The packet can be masked using the **OM** output mask function. Remember that the color values from the CMOS camera will range from between 16 and 240. If TC is called with no arguments it will track with the precious set of tracking parameters.

Default Type T packet
T mx my x1 y1 x2 y2 pixels confidence\r

Example of how to Track a Color with the default mode parameters:

```
:TC 130 255 0 0 30 30
ACK
T 50 80 38 82 53 128 35 98
T 52 81 38 82 53 128 35 98
```

TI boolean \r

This command activates **T**rack **I**nverted mode. When track inverted mode is enabled, the camera will track colors that are outside of the user defined color range instead of inside. This is good for either tracking edges, or tracking any object that shows up against a homogenous background.

TW \r

This command will **T**rack the color found in the central region of the current **W**indow. After the color in the current window is grabbed, the track color function is called with those parameters and on the full image window. This can be useful for locking onto and tracking an object held in front of the camera. Since it actually calls track color, it returns the same type T track packet. Note, the current virtual window setting will only be used for grabbing the color to track and then the window will return to its maximum size.

The following internal steps are performed when "TW" is called:

1. Shrink the window to 1/2 the size (in each dimension) of the current window centered on the current window. (sw 30 54 50 90)

2. Call the get mean command but do not display the output. (gm)

3. Restore the window to the full image size. (sw 1 1 88 143)

4. Set the min and max value for each color channel to be the mean for that channel +/- 30.

Example of how to use Track Window:

```
:TW
ACK
T 6 55 2 40 12 60 10 70
T 6 55 2 41 12 61 11 70
```

UD <64 raw bytes> \r

This command allows you to **U**pload a **D**ifference frame buffer. The command waits for 64 raw byte values that fill up the 8 by 8 internal frame difference buffer. A '\r' cancels the transfer. A value of 0 indicates that the region should be masked and not detect motion. With this command in combination with line mode type 2, it is possible to download and upload different reference frames for frame differencing.

See LM on page 40 for instructions on downloading a difference buffer.

VW [x y x2 y2] \r

Do not try VW 0 0 88 144, this is outside of the 1 1 88 143 bounds.

This command sets the **V**irtual **W**indow size of the camera. It accepts the x and y Cartesian coordinates of the upper left corner (1,1) followed by the lower right of the window you wish to set. The origin is located at the upper left of the field of view. **VW** can be called before an image processing command to constrain the field of view. Without arguments it returns to the default full window size of for the current combination of camera type, downsampling and resolution mode. Note that reducing the vertical window size can be used to speed up processing time to achieve higher frame rates with the track color command. 50 fps can be achieved with a vertical dimension of 65 or less.

See GW on page 37 to find out how to check your window configuration.

Example of setting the camera to select the mid portion of the view:

```
:VW 35 65 45 75
ACK
```