ZX-CCD

Vision board

Features :

- Track user defined color blobs at 17 Frames Per Second
- Find the centroid of the blob
- Gather mean color and variance data
- Transfer a real-time binary bitmap of the tracked pixels in an image
- Arbitrary image windowing
- Adjust the camera's image properties
- Dump a raw image
- 80x143 Resolution
- 115,200 / 38,400 / 19,200 / 9600 baud serial communication
- Slave parallel image processing mode off a single camera bus

Packing List

- ZX-CCD with image sensor board
- ZX-U2S USB interfacing board for setting and calibration
- Documentation and CD-ROM
- miniB USB cable
- 2 of JST3A-8 interface cable

1. Introduce

The ZX-CCD can be used to track or monitor color. The best performance can be achieved when there are highly contrasting and intense colors. For instance, it could easily track a red ball on a white background, but it would be hard to differentiate between different shades of brown in changing light.

Tracking colorful objects can be used to localize landmarks, follow lines, or chase a moving beacon. Using color statistics, it is possible to monitor a scene, detect a specific color or do primitive motion detection. If the camera detects a drastic color change, then chances are something in the scene changed. Using "line mode", the ZX-CCD can act as an easy way to get low resolution binary images of colorful objects. This can be used to do more sophisticated line following that includes branch detection, or even simple shape recognition. These more advanced operations would require custom algorithms that would post process the binary images sent from the ZX-CCD. As is the case with a normal digital camera, this type of processing might require a computer or at least a fast microcontroller.

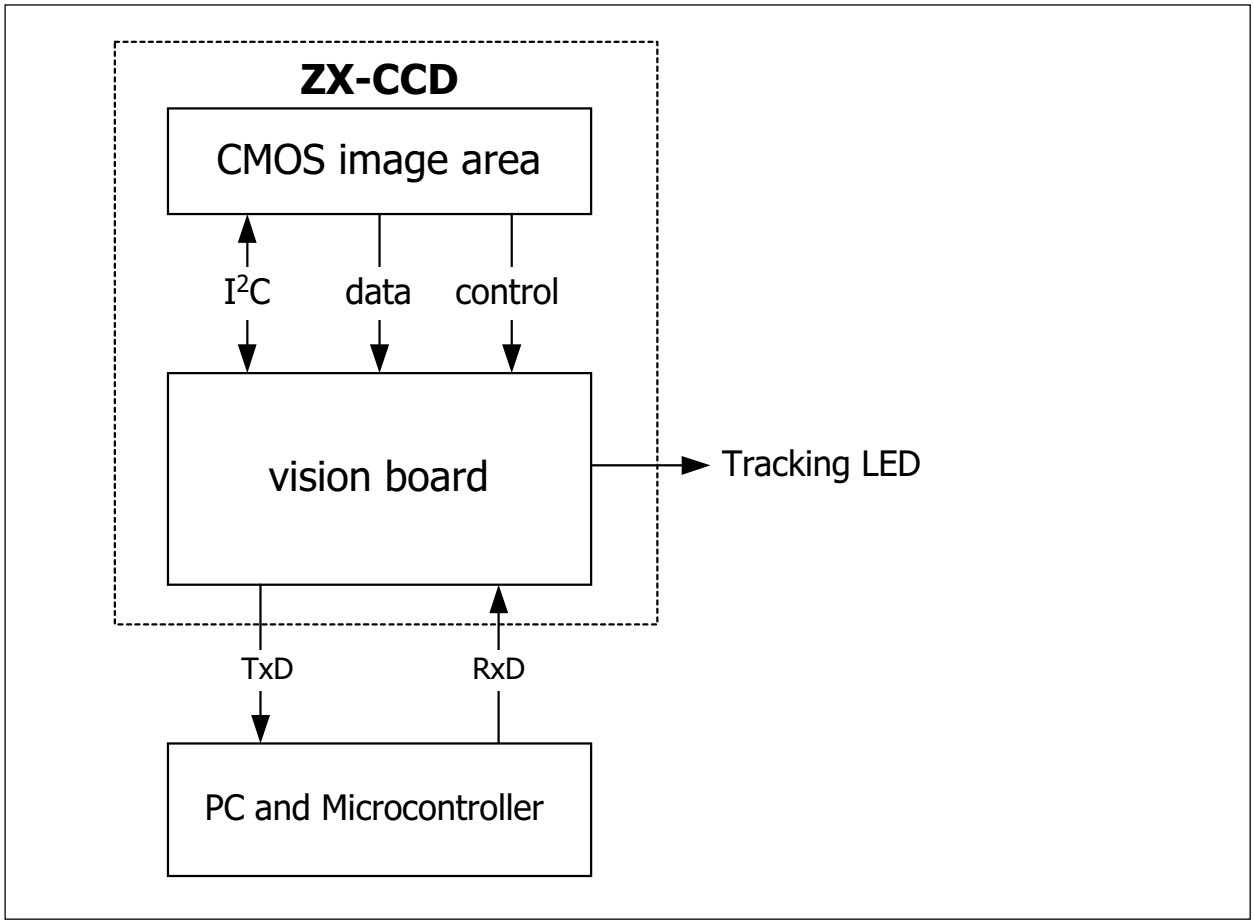


Figure 1 : ZX-CCD diagram

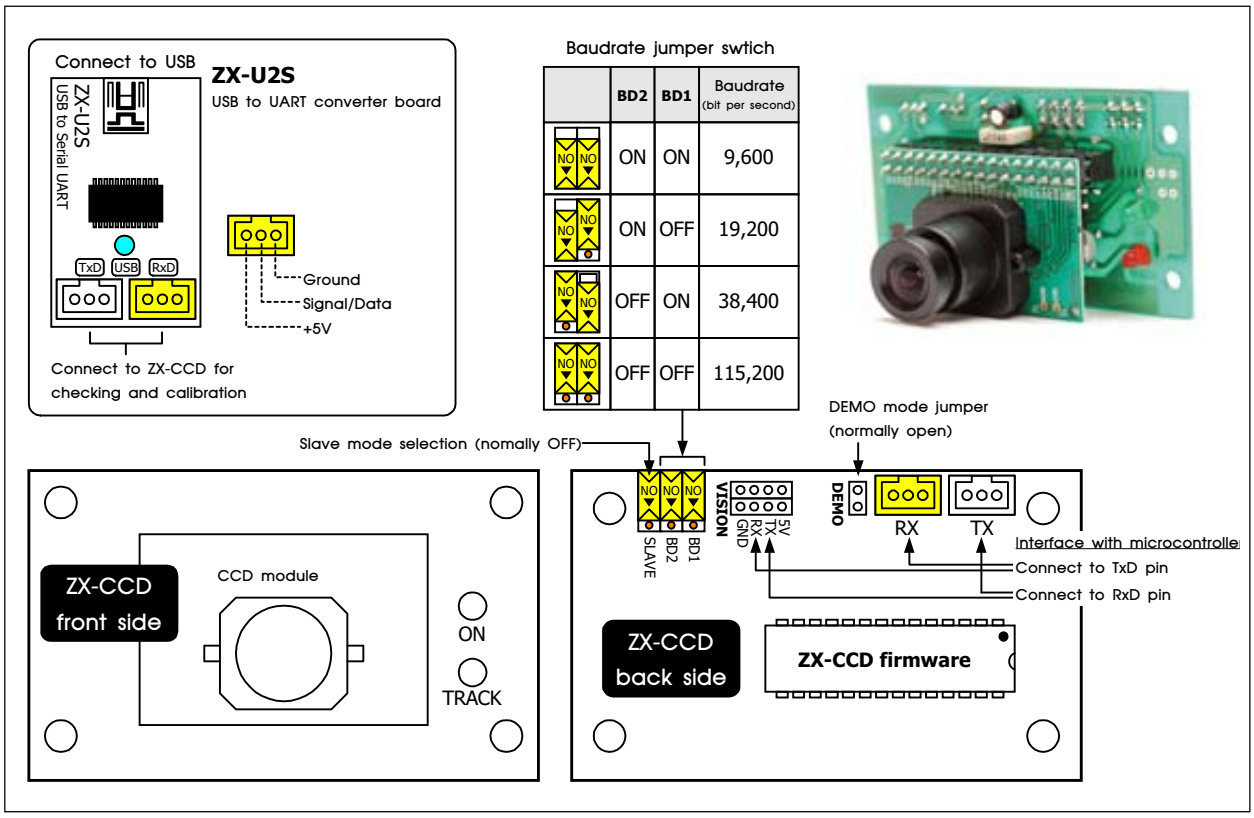


Figure 2 : ZX-CCD Hardware configuration

The most common configuration for the ZX-CCD is to have it communicate to a computer via USB port by using ZX-U2S USB to serial converter board (included in package) The ZX-CCD is small enough to add simple vision to embedded systems that can not afford the size or power of a standard computer based vision system. Its communication protocol is designed to accommodate even the slowest of processors. If your device does not have a fully level shifted serial port, you can also communicate to the ZX-CCD over the TTL serial port. This is the same as a normal serial port except that the data is transmitted using 0 to 5 volt logic.

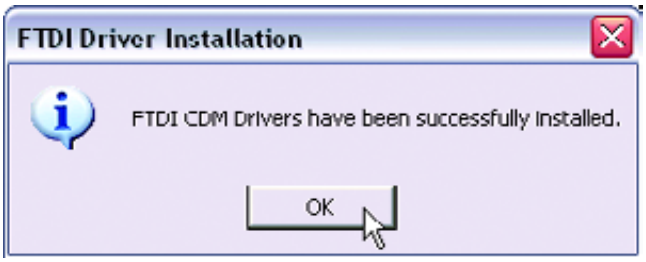
The ZX-CCD supports various baud rates to accommodate slower processors. For even slower processors, the camera can operate in "poll mode". In this mode, the host processor can ask the CMUcam for just a single packet of data. This gives slower processors the ability to more easily stay synchronized with the data. It is also possible to add a delay between individual serial data characters using the "delay mode" command. Due to the communication delays, both poll mode and delay mode will lower the total frame rate that can be processed.

2. Supply Preparation

The suitable power supply for ZX-CCD is +5V. User can use from both signal connector RxD or TxD from microcontroller board or ZX-U2S (in calibration mode). The ZX-CCD need 70mA current consumption.

3. ZX-U2S Driver installation

Testing the ZX-CCD, requires computer. ZX-CCD can interface computer va USB pot by using the ZX-U2S board. First, you must install the USB driver. Double click at USBDriverInstallerV2.x.x.exe (x.x is number of version) file from the bundled CD-ROM to start the driver instalation. The installation dialogue-box will appear below.

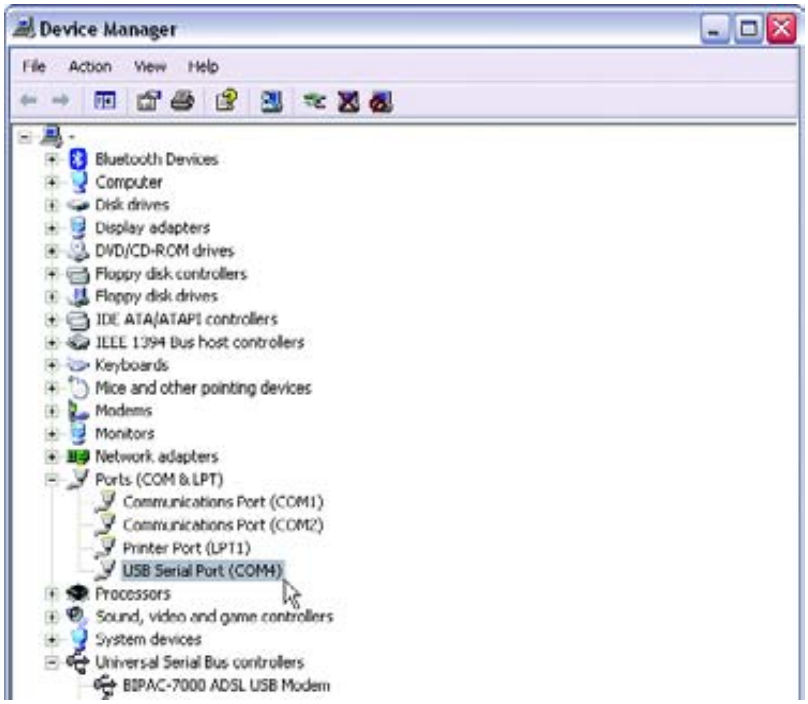


3.1 Check the USB serial port address

(1) Plug the USB cable or ZX-U2S. Wait until LED indicator at USB position on ZX-U2S is turned-on.

(2) Check the Virtual COM port or USB Serial port address by clicking Start → Control Panel → System → Hardware → Device Manager

(3) See the list of USB serial port and remember the COM port address to work with ZX-U2S board. Normally it will create COM3 or higher. In this example is **COM4**



3.2 Virtual COM port with CMUCAM2GUI and RS-232 Terminal operation notice

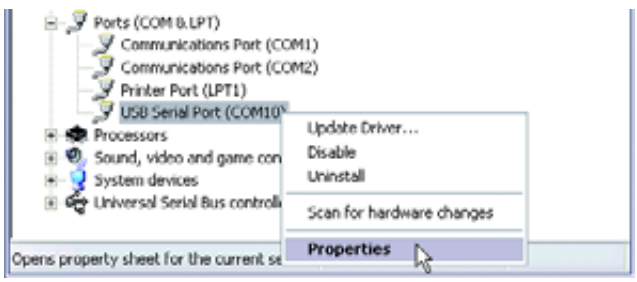
Normally CMUCAM2 and RS-232 Terminal software can interface with COM port not higher than COM9. Thus, user must make sure the USB serial port address not higher than COM9. If higher, please do following procedure.

(1) Connect the ZX-U2S to Computer USB port.

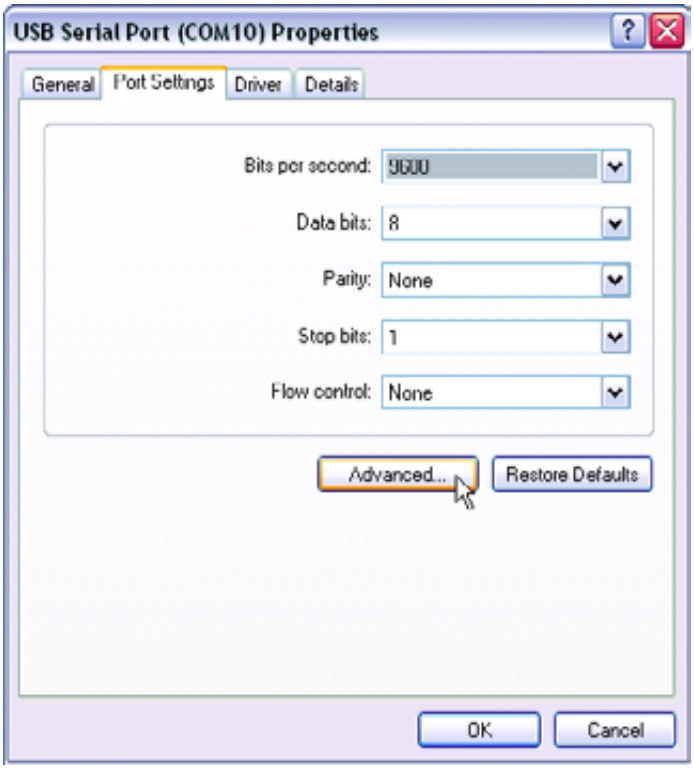
(2) Check the COM port address by clicking at **Start → Control Panel →System**

(3) Select the **Hardware** tab and click on the **Device Manager** button.

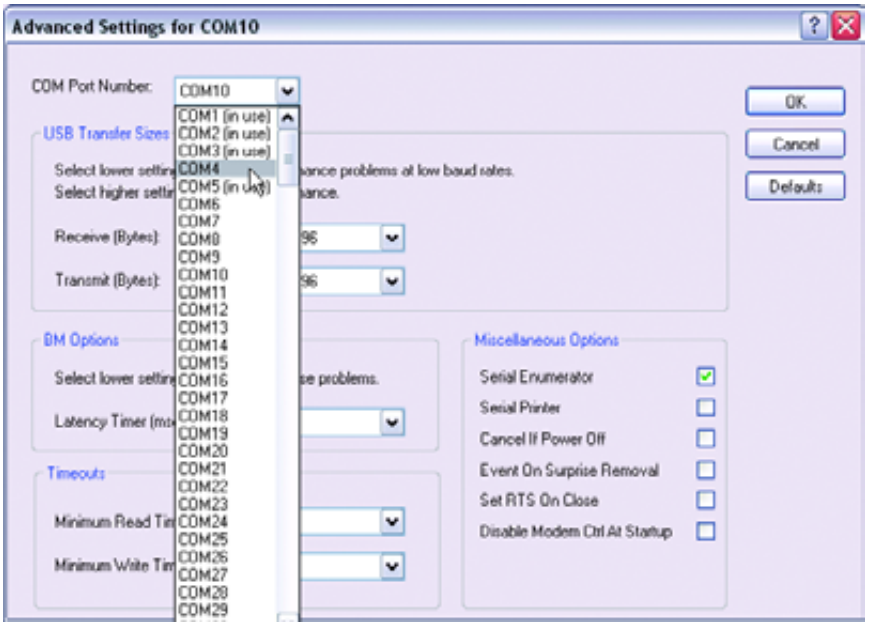
(4) Check the hardware listing. At the Port listing, you will found **USB Serial port (COM x)**.If COM port is higher than COM9 (this example is COM10), please click on the right-button mouse and select to **Properties**



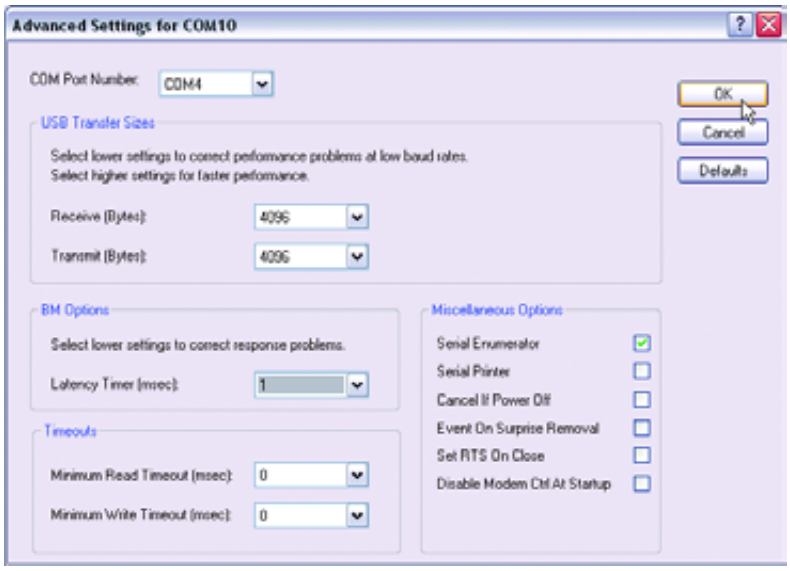
(5) The **USB Serial Port (COM10) Properties** window will appear. Select the **Port Setting** tab and set all value following the figure below and click on the **Advance** button



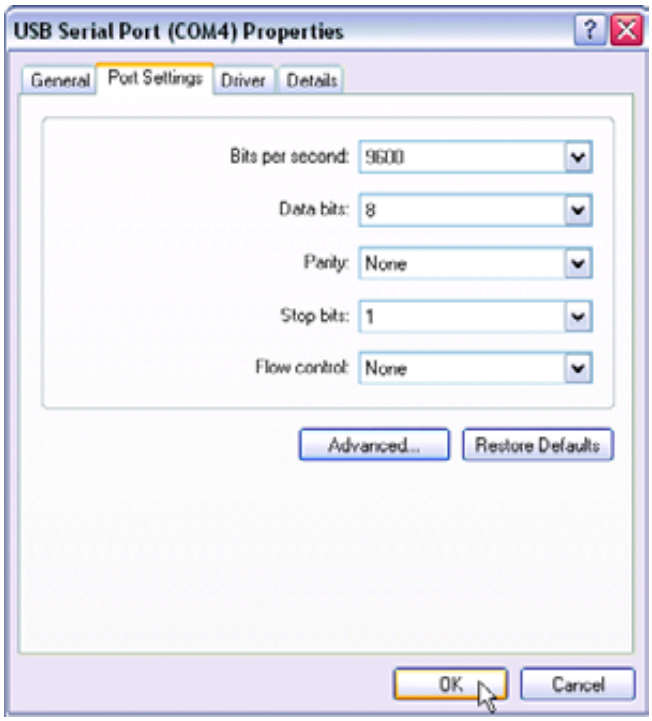
(6) The Advanced Setting for COM10 will appear. Click on the **COM Port Number** box to change to ***COM4*** or another port in range ***COM1*** to ***COM9***.



(7) Set the value following the figure below. Especially at the **Latency Timer (msec) suggested to set to 1** and check the box at **Serial Enumerator.** Click **OK** button.



(8) Go back to the USB Serial Port Properties. Now the *COM port number at the title bar will change to COM4. Click on the* **OK** button.



(9) Remove the ZX-U2S from USB port and re-plug again. Check the USB Serial port address. The new address must be **COM4**. Now the ZX-U2S is ready for using with all ZX-CCD software.

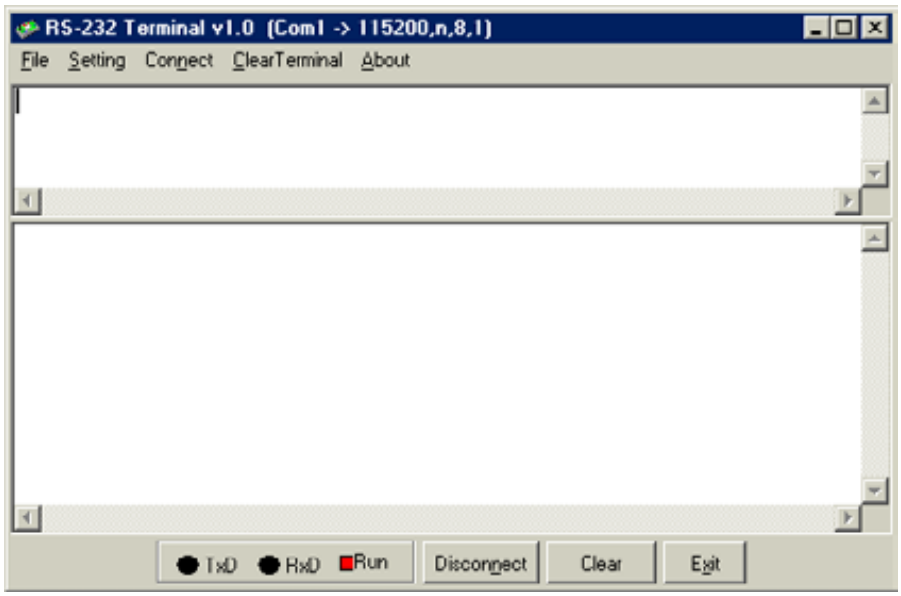
4. Testing ZX-CCD

Easiest to test ZX-CCD operation is Computer testing. Two softwares would be use in tesing; **RS-232 Terminal** and **CMUCAM2GUI.** The RS-232 Terminal is used for interfacing, reading and control. CMUCAM2GUI is used for focus adjustment.

Step of testing as :

(1) Install RS-232 Terminal software by click RS-232 TerminalSetup.exe file in CD-ROM and click **OK** until install complete.

(2) Run by select **Start→ Program→ RS-232 Termina**l. Main window of this program will appear.



(3) Enter menu **Setting → Serial port** to select serail port connected. After that select baudrate at same menu; **Setting → Baud rate (bps)** to 115200, select data bit by **Setting → Data** to 8-bit and select parity bit by menu **Setting → Parity** to None. Click Connect Button to connect ZX-CCD with serial port. User can adjust the text format by entered to menu **Setting → Terminal**

(4) Remove BD1 and BD2 jumper for select baudrate 115,200 bits per second. Remove DEMO and SLAVE jumper too.

(5) Apply supply voltage to ZX-CCD. At output window of RS-232 Terminal will show message :

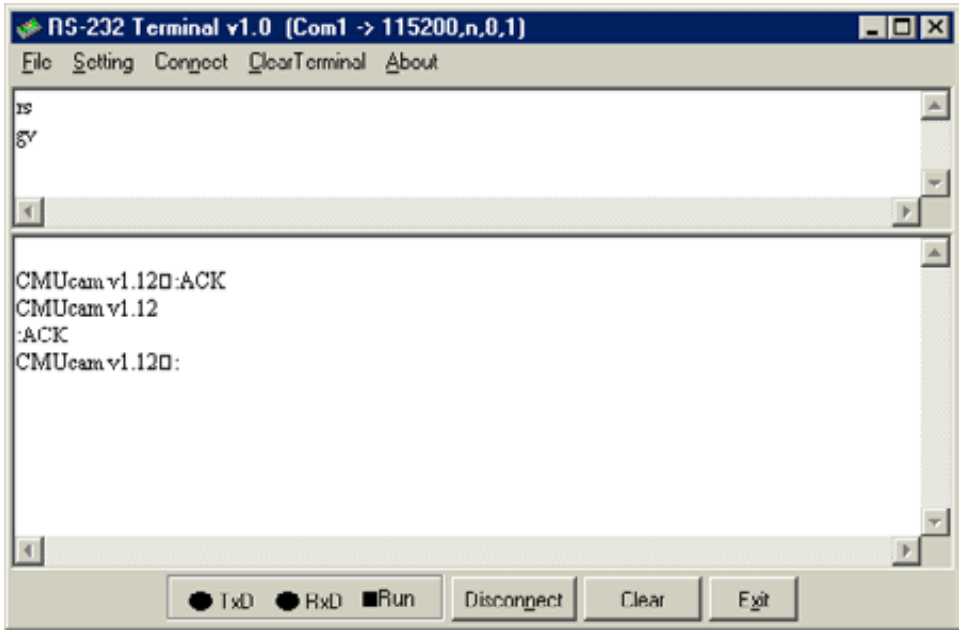
```
CMUcam V1.12 :
```

(6) Type RS command into input window of RS-232 Terminal, press Enter for testing reset iV--CAM. Thjis message will show :

```
:ACK
CMUCAM1.12
```

(7) Type GV command for reading version number of firmware. This message will show :

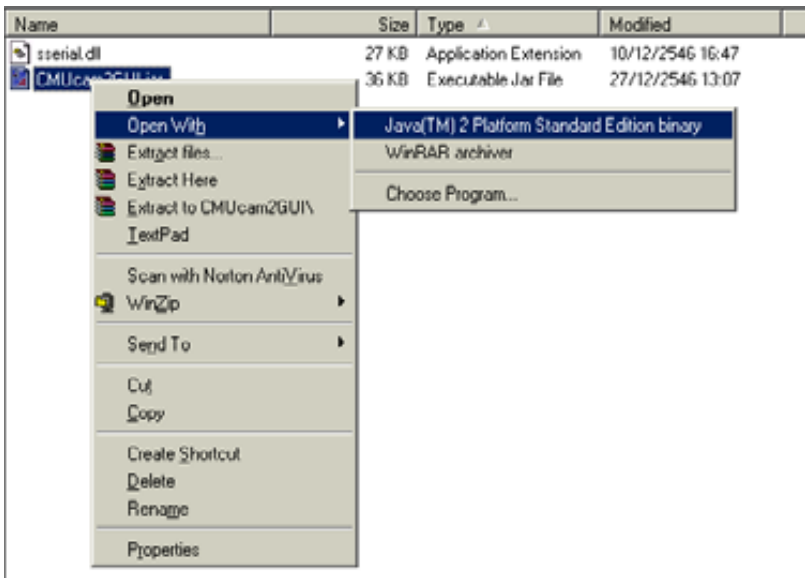
```
:ACK
CMUCAM1.12
```



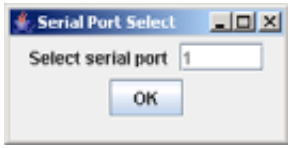
(8) If ZX-CCD work following above, it confirm ZX-CCD ready to interface with computer. Next testing is focusing. The software testing is **CMUcam2GUI**. Copy file ***CMUcam2GUI.jar*** that contain in ***CMUcam2GU***I folder in CD-ROM into user's harddisk.

(9) CMUcam2GUI need JaVA runtime to work. User must install JAVA runtime machinein their system. If not have, install this software from ZX-CCD CD-ROM by enter to CD-ROM and double-click a***t jre-1_5_0_04-windows-i586-p.exe*** file. Click **OK** until installation complete.

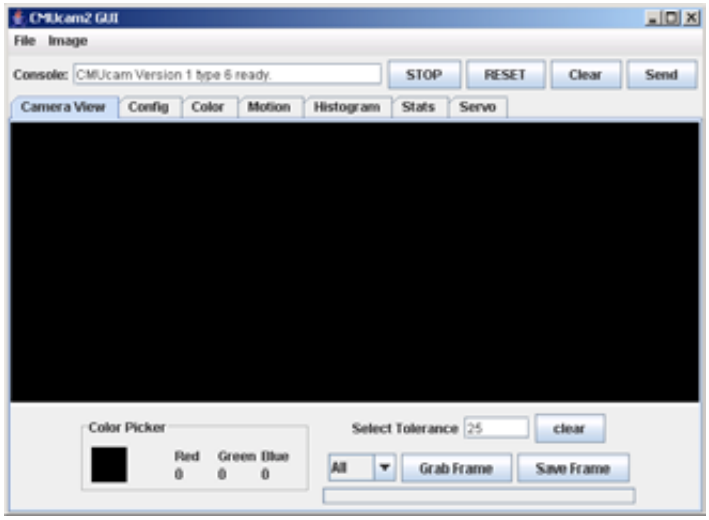
(10) Enter to folder **CMUcamGUI → Stand_alone**. Open file ***CMUcam2GUI.jar*** by right-button mouse and select Java(TM)2 Platform Standard Edition Binary. See the figure below



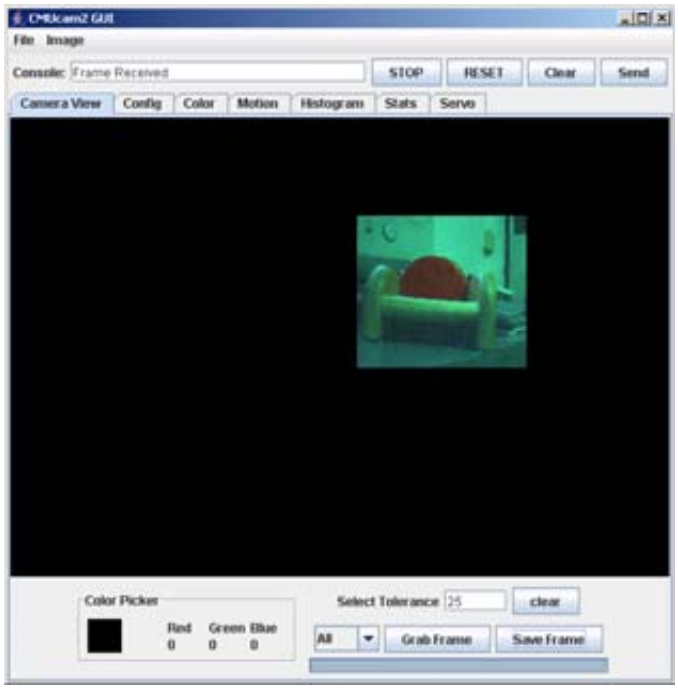
(11) Serial port selection window will appear.



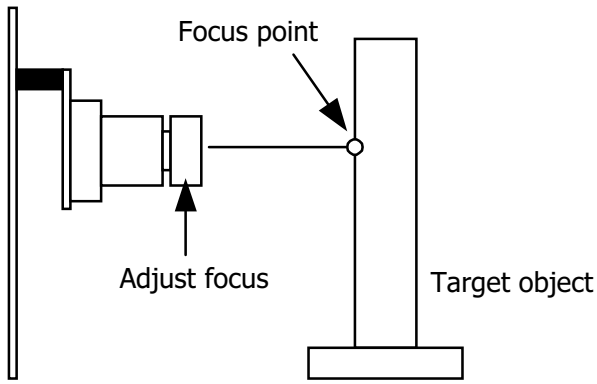
(12) Select the serial port that is created by Virtual COM port driver in Topic 3 of this documentation. Just a moment, main window of this software will appear.Console box showmessage : **CMUcam Version 1 type 6 ready**. It means ZX-CCD connect with CAMcam2GUI successful.



(13) Remove cover rubber from Camera lens. Click **Grab Frame** button. Wait a moment. The image that grab will display on left screen of main window. But the image will mirror. Enter to menu Image select **Flip Horizontal**. The image will filp to normal.



(14) Once user have the ability to grab frames from the camera, user should be able to rotate the front part of the ZX-CCD lens and see the image change. Try to get the picture to be as sharp as possible by dumping frames and changing the position of the lens a small amount each time. Usually the camera is in focus when the lens is a few rotations away from the base. Adjust until the image grabbed sharp and clear.



5. Notes on Better Tracking

5.1 Better Tracking with Auto-gain and White Balance

Auto-gain is an internal control that adjusts the brightness level of the image to best suit the environment. It attempts to normalize the lights and darks in the image so that they approximate the overall brightness of a hand adjusted image. This process iterates over many frames as the camera automatically adjusts its brightness levels. If for example a light is turned on and the environment gets brighter, the camera will try and adjust the brightness to dim the overall image.

White balance on the other hand attempts to correct the camera's color gains. The ambient light in your image may not be pure white. In this case, the camera will see colors differently. The camera begins with an initial guess of how much gain to give each color channel. If active, white balance will adjust these gains on a frame-by-frame basis so that the average color in the image approaches a gray color.

Empirically, this "gray world" method has been found to work relatively well. The problem with gray world white balance is that if a solid color fills the camera's view, the white balance will slowly set the gains so that the color appears to be gray and not its true color. Then when the solid color is removed, the image will have undesirable color gains until it re-establishes its gray average.

When tracking colors, like in demo mode, you may wish to allow auto-gain and white balance to run for a short period and then shut them off. While on for a period of about 5 seconds, the camera can set its brightness gain and color gains to what it sees as fit. Then turning them off will stop the camera from unnecessarily changing its settings due to an object being held close to the lens or shadows etc. If auto-gain and white balance where not disabled and the camera changed its settings for the RGB values, then the new measured values may fall outside the originally selected color tracking thresholds.

YCrCb is a different color space definition from the more commonly known RGB space. In YCrCb the illumination data is stored in a separate channel. Because of this property, in YCrCb mode the camera may be more resistant to changes in illumination. Because it is a different color space, images in YCrCb do not look like standard RGB images when directly mapped by a frame dump program. The RGB channels map to CrYCb. So in YCrCb mode, the value returned as the red parameter is actually Cr, the green parameter is Y and the blue parameter is Cb. So if you wish to track a red object, you need to look at a dumped frame to see what that object's colors map to in YCrCb. It should then be possible to find the Cr and Cb bounds while giving a very relaxed Y bound showing that illumination is not very important. Below are the transformations used by the camera to convert RGB into YCrCb:

```
RGB -> CrYCb
Y=0.59G + 0.31R + 0.11B
Cr=R-Y
Cb=B-Y
```

6. About the CMOS Camera

From power up, the camera can take up to 15 seconds to automatically adjust to the light. Drastic changes in the environment, such as lights being turned on and off, can induce a similar readjustment time.

When using the camera outside, due to the sun's powerful IR emissions, even on relatively cloudy days, it will probably be necessary to use either an IR filter or a neutral density camera filter to decrease the ambient light level. The field of view depends on the lens attached to the camera. It is possible to special order the camera with wider or narrower lenses. Individual lenses can be purchased separately.

The functions provided by the camera board are meant to give the user a toolbox of color vision functions. Actual applications may greatly vary and are left up to the imagination of the user. The ability to change the viewable window, grab color / light statistics and track colors can be interwoven by the host processor to create higher level functionality.

One notable property of the CMOS sensor array is that it returns values between 16 and 240 for each pixel. This effect is noticeable when the camera is tracking colors, getting mean color data or dumping a frame. This limited range on the data does not depend on the mode of the camera and will still exist in YCrCb mode.

6.1 Analog Video Output

Pin 32 on the camera bus is a black and white analog output pin. It is possible to connect the analog output to a TV or multi-sync monitor. Due to the clock rate of the camera, the analog output does not correctly synchronize with standard NTSC or PAL systems. If you have some system that can synchronize with a non-standard signal, it may be possible to monitor the video while processing. On versions 1.22 and lower of the board, it is required that you cut the connection on the ZX-CCD controller board between camera bus pins 31 and 32 (this is fixed in version 1.23 and higher). Then, connect the central line on a 75 Ohm coax input plug to pin 32 on the camera bus. Next, connect the outer shield to the camera's common ground (pin 31 on the camera bus). Finally, power up the camera and adjust the frame rate until you see the best results. Most standard TVs can at least see a skewed flickering image when the frame rate clock divider is set to 0 (CR 17 0). Setting frame rates higher than 17 fps will stop the CMUcam's processing from working. So while you are sending data to a monitor, you can not dump frames or perform any processing.

7. Serial Command Set

The serial communication parameters are as follows:

- 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 exerts too much overhead, it is possible to use varying degrees of raw data transfer. (See Raw Mode command, “RM”.)

8. Basic Command Set

\r (the "enter" or "return" key)

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.

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

```
:
ACK
:
```

GM\r

This command will Get the Mean color value in the current image. As with the TC command this function only operates on a selected region of the image. The mean values will be between 16 and 240. 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.

Type S data packet format

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
```

L1 value\r

This command is used to control the green LED tracking Light. It accepts 0, 1 and 2 (default) as inputs. 0 disables the tracking light while a value of 1 turns on the tracking light. A value of 2 puts the light into its default auto mode. In auto mode and while tracking, the light turns on when it detects the presence of an object that falls within the current tracking threshold. This command is useful as a debugging tool.

Example of how to toggle the Tracking Light on and then off

```
:L1 2
ACK
:L1 0
ACK
```

MM mode\r

This command controls the Middle Mass mode which adds the centroid coordinates to the normal tracking data. A mode value of 0 disengages middle mass, a value of 1(default) engages middle mass and a value of 2 engages the mode and turns on the servo PWM signal that tries to center the camera on the center of color mass (see the Demo Mode Jumper description). This mode is good if you want a single point representation of where the object is or if there is too much small background noise to get a good bounding box. To switch the direction of the servo it is necessary to set the 2nd bit (counting from 0 i.e. bit-wise OR in the value 4) of the mode value. If the 3rd bit is set (bit-wise OR in the value 8), then tracking a color will return a type N packets that is identical to a type M packet, only it contains the current servo position as its first return value (see Output Data Packet Description section).

Example of how to disable Middle Mass mode

```
:MM 0
ACK
:TC
ACK
C 38 82 53 128 35 98
C 38 82 53 128 35 98
C 38 82 53 128 35 98
```

NF active\r

This command controls the Noise Filter setting. It accepts a Boolean value 1 (default) or 0. A value of 1 engages the mode while a value of 0 deactivates it. When the mode is active, the camera is more conservative about how it selects tracked pixels. It requires 2 sequential pixels for a pixel to be tracked.

Example of how to turn off noise filtering

```
:NF 0
ACK
:
```

PM mode\r

This command puts the board into Poll Mode. Setting the mode parameter to 1 engages poll mode while 0 (default) turns it off. When poll mode is engaged only one packet is returned when an image processing function is called. 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
:
```

RS \r

This command ReSets the vision board. Note, on reset the first character is a `/r`.

Example of how to reset the camera

```
:rs
ACK
CMUcam v1.12
:
```

SW [x y x2 y2] \r

This command Sets the Window size of the camera. It accepts the x and y Cartesian coordinates of the upper left corner 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. SW 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 1,1,80,143.

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

```
:SW 35 65 45 75
ACK
:
```

TC [Rmin Rmax Gmin Gmax Bmin Bmax]\r

This command begins to Track a Color . It takes in the minimum and maximum RGB (CrYCb) values and outputs a type M or C data packet (set by the MM command). The smaller type C packet encodes a bounded box that contains pixels of the currently defined color, the number of found pixels (scaled: actual value is (pixels+4)/8) that fall in the given color bounds and a confidence ratio. The default type M packet also includes the center of mass of the object. The resolution of the processed image is 80x143. The X values will range from 1 to 80 and the y values will range from 1 to 143. A packet of all zeros indicates that no color in that range was detected. The confidence value is a ratio of the pixels counted within the given range and the area of the color bounding box. It returns a value which ranges from 0 to 255. Under normal operations any value greater then 50 should be considered a very confident lock on a single object. A value of 8 or lower should be considered quite poor. With no arguments, the last color tracking parameters will be repeated.

Type M packet

M mx my x1 y1 x2 y2 pixels confidence\r

Type C packet

C 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
M 50 80 38 82 53 128 35 98
M 52 81 38 82 53 128 35 98
M 51 80 38 82 53 128 35 98
```

TW \r

This command will Track the color found in the central region of the current Window. After the color in the current window is grabbed, the track color function is called with those parameters and on the full screen. 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 of C or M color packet.

Note, your set window will only be used for grabbing the color to track and then the window will return to 80x143.

Example of how to use Track Window:

```
:TW
ACK
S 240 50 40 12 7 8
C 2 40 12 60 10 70
C 3 41 12 61 11 70
C 2 40 12 60 13 70
C 3 42 12 62 9 70
C 4 45 12 60 8 70
```

9. Output Data Packet Descriptions - Basic

All output data packets are in ASCII viewable format.

ACK

This is the standard acknowledge string that indicates that the command was received and fits a known format.

NCK

This is the failure string that is sent when an error occurred. The only time this should be sent when an error has not occurred is during binary data packets.

Type C packet

This is the return packet from a color tracking command.

C x1 y1 x2 y2 pixels confidence\r

x1 - The left most corner's x value

y1 - The left most corner's y value

x2 - The right most corner's x value

y2 -The right most corner's y value

pixels -Number of Pixels in the tracked region, scaled and capped at 255: (pixels+4)/8

confidence -The (# of pixels / area) x 256 of the bounded rectangle and capped at 255

Type M packet

This is the return packet from a color tracking command with Middle Mass mode on.

M mx my x1 y1 x2 y2 pixels confidence\r

mx - The middle of mass x value

my - The middle of mass y value

x1 - The left most corner's x value

y1 - The left most corner's y value

x2 - The right most corner's x value

y2 -The right most corner's y value

pixels -Number of Pixels in the tracked region, scaled and capped at 255: (pixels+4)/8

confidence -The (# of pixels / area) x 256 of the bounded rectangle and capped at 255

Type S data packet format

This is a statistic packet that gives information about the camera's view

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation\r

Rmean - the mean Red or Cr (approximates r-g) value in the current window

Gmean - the mean Green or Y (approximates intensity) value found in the current window

Bmean - the mean Blue or Cb (approximates b-g) found in the current window

Rdeviation - the *deviation of red or Cr found in the current window

Gdeviation- the *deviation of green or Y found in the current window

Bdeviation- the *deviation of blue or Cb found in the current window

Note : **deviation: The mean of the absolute difference between the pixels and the region mean.*

10. Advanced Command Set

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

This command sets the Camera's internal Register values directly. The register locations and possible settings can be found in the Omnivision 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.

Useful Settings:

Register	**Values**
3 Saturation	0-255 (default = 128)
5 Contrast	0-255 (default = 72)
6 Brightness	0-255 (default = 128)

17 Clock Speed

2 = 17 fps (default)
3 = 13 fps
4 = 11 fps
5 = 9 fps
6 = 8 fps
7 = 7 fps
8 = 6 fps
10 = 5 fps
12 = 4 fps

18 Color Mode

0 = YCrCb*, AGC off, Auto White-balance Off
4 = YCrCb*, AGC off, Auto White-balance On
32 = YCrCb*, AGC on, Auto White-balance Off
36 = YCrCb*, AGC on, Auto White-balance On
8 = RGB, AGC off, Auto White-balance Off
12 = RGB, AGC off, Auto White-balance On
40 = RGB, AGC on, Auto White-balance Off (default)
44 = RGB, AGC on, Auto White-balance On

19 Auto Adjust : *Turns off or on both the Auto White-balance and Auto-gain Controls*

32 = Auto Adjust Off
33 = Auto Adjust On (default)

45 Band Filter : Use the Band Filter with Fluorescent lighting

3 = Band Filter Off (default)
7 = Band Filter On

Example of decreasing the internal camera clock speed (default speed is 2)

```
:CR 17 5
ACK
:
```

Note : **The red channel becomes Cr which approximates r-g, The green channel becomes Y which approximates intensity, the blue channel becomes Cb which approximates b-g*

```
RGB -> CrYCb
Y=0.59G + 0.31R + 0.11B
Cr=R-Y
Cb=B-Y
```

DF\r

This command will Dump a Frame out to the USB serial port of 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 column by column with a frame synchronize byte and a column synchronize byte. (This data can be read and displayed by the CMUcam GUI Java application.) Since the data rate required to send the raw video greatly exceeds the maximum serial port speed, only one column per frame is sent at a time. This allows you to see a slowly updating view of what the camera sees. To get the correct aspect ratio, double each column of pixels. The camera is able to dump a full resolution frame at full speed (17 columns per second) only at 115,200 baud. At lower baud rates, or 115,200 baud with added delays the frame rate must be decreased in order to see a full resolution image. With auto-gain on and at lower frame rates, the image at first may appear much brighter than usual. This is because the camera is getting frames slower than usual and takes longer to adapt. Try manually setting the brightness and contrast.

Type F data packet format

1 2 rgbrgb ... rgbrgb 2 rgbrgbr ... rgbrgb ...

1 - new frame

2 - new col

3 - end of frame

RGB (CrYCb) ranges from 16 - 240

Example of a Frame Dump from a terminal program

*(**WARNING**: This may temporarily interfere with a terminal program by sending nonvisible characters)*

```
:DF
ACK
maKP(U A$IU AL<>U A$L*YL%*L L (G AUsonthAYA(KMAy098a34ymawvk....
```

DM value\r

This command sets the Delay before packets that are transmitted over the serial port. 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 correlates to approximately the transfer time of one bit at the current baud rate.

GV\r

This command Gets the current Version of the firmware from the camera. It returns an ACK followed by the firmware version string.

Example of how to ask for the firmware version

```
:GV
ACK
CMUcam  v1.12
```

HM active\r

This command puts the camera into Half-horizontal resolution Mode for the DF command and the LM command when dumping a bitmap image. An active value of 1 causes only every odd column to be processed. The default value of 0 disables the mode.

I1 \r

This command uses the servo port as a digital Input. Calling I1 returns either a 1 or 0 depending on the current voltage level of the servo line. The line is pulled high; because of this it is only required to pull it low or let it float to change it's state. The servo line can also be used as a digital output. (See S1 command)

Example of how to read the digital value of the servo line

```
:I1
ACK
1
```

LM active\r

This command turns on Line Mode which uses the time between each frame to transmit more detailed data about the image. It adds prefix data onto either C, M or S packets. This mode is intended for users who wish to do more complex image processing on less reduced data. Since the frame rate is not compromised, the actual processing of the data put out by the vision system must be done at a higher rate. This may not be suitable for many slower microcontrollers.

Line mode's effect on TC and TW:

When line mode is active and TC or TW is called, line mode 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). 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 vertical resolution is constrained by the transfer time of the horizontal data so lines may be skipped when outputting data. In full resolution mode, the resulting binary image is 80x48. The binary bitmap is terminated by two 0xAA's. This is then followed by the normally expected standard C or M data packet (processed at that lower resolution).

Example of TC with line mode on

```
:LM 1
:TC
(raw data: AA XX XX XX .... XX XX XX AA AA) C 45 72 65 106 18 51
(raw data: AA XX XX XX .... XX XX XX AA AA) C 46 72 65 106 18 52
```

Line mode's effect on GM:

When line mode is active and GM is called, line mode will send a raw (not human readable) mean value of every line being processed. These packets are started with an 0xFE and terminate with an 0xFD. Each byte of data between these values represents the corresponding line's mean color value. Similarly to the bitmap mode the vertical resolution is halved, because of the serial transfer time. At 17 fps 115,200 baud every other line is skipped. At any slower frame rate (still 115,200 baud) no lines will be skipped.

Example of GM with line mode on

```
:LM 1
:GM
(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
```

RM bit_flags\r

This command is used to engage the Raw serial transfer Mode. 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, followed by the packet and the packet identifying character (i.e. C for a color packet). 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 3 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.

If bit 0 is enabled, then output to the camera is in raw bytes.

If bit 1 is enabled, then the "`ACK\r`" and "`NCK\r`" confirmations are suppressed.

If bit 2 is enabled, then input to the camera is in raw bytes.

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

*(**WARNING**: This may temporarily interfere with a terminal program by sending non visible characters)*

```
:RM 1
ACK
:TC 50 20 90 130 70 255
ACK
C>%k(ai  Ck$&,.
```

SM value \r

This command is used to enable the Switching Mode of color tracking. When given a 0 it is in it's default mode where tracking will return its normal C or M color packet. **If the value is set to 1**, *the tracking commands will alternate each frame between color packets and S statistic packets*. Each statistic packet is only being sampled from an area one quarter the size of the bounded area returned from the tracking command. If no object was bounded, then no S statistic packets are returned. This can be useful for adaptive tracking or any type of tracking where you would like to get feedback from the currently bound target. After the first tracking packet is returned, the window gets set back to full size for all future packets. Note, you will get only half the number of actual color packets per time interval.

Example of how to Track Color with SM

```
:SM 1
ACK
:TC 200 255 0 30 0 30
ACK
C 2 40 12 60 10 70
S 225 20 16 2 3 1
C 5 60 20 30 12 100
S 225 19 17 1 2 1
C 0 0 0 0 0 0
C 0 0 0 0 0 0
C 0 0 0 0 0 0
C 5 60 20 30 12 100
S 225 19 17 1 2 1
```

11. Output Data Packet Descriptions - Advanced

All output data packets are in ASCII viewable format except for the F frame and prefix packets.

ACK

This is the standard acknowledge string that indicates that the command was received and fits a known format.

NCK

This is the failure string that is sent when an error occurred. The only time this should be sent when an error has not occurred is during binary data packets.

Type N packet

This is identical to a type M packet with an added value for the servo position.

N spos mx my x1 y1 x2 y2 pixels confidence\r

spos - The current position of the servo

Binary bitmap Line Mode prefix packet

This packet is in raw byte form only. It starts off with the hex value 0xAA and then streams bytes, with each byte containing a mask for 8 pixels, from the top-left to the bottom-right of the image.

(Each binary bit in the byte represents a pixel) The bitmap is then terminated with two 0xAAs.

0xAA is never transmitted as part of the data, so it should be used to signify termination of the binary bitmap. After the binary bitmap is complete, a normal tracking packet should follow.

(raw data: AA XX XX XX .... XX XX XX AA AA) C 45 72 65 106 18 51

(raw data: AA XX XX XX .... XX XX XX AA AA) C 46 72 65 106 18 52

Get mean Line Mode prefix packet

This packet prefix outputs the mean color of each row being processed. These packets are started with an 0xFE and terminate with an 0xFD. Each byte of data between these values represents the corresponding line's mean color value. Due to the serial transfer time, the vertical resolution is halved. After all rows have been completed, a normal tracking packet should follow.

```
(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
```

Type F data packet format

1 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 ...

1 - new frame

2 - new column

3 - end of frame

RGB (CrYCb) ranges from 16 - 240

RGB (CrYCb) represents a two pixels color values. Each pixel shares the red and blue.

176 columns of R G B (Cr Y Cb) packets (forms 352 pixels)

144 rows

To display the correct aspect ratio, double each column so that your final image is 352x144

It does not begin with an "F" and only sends raw data!