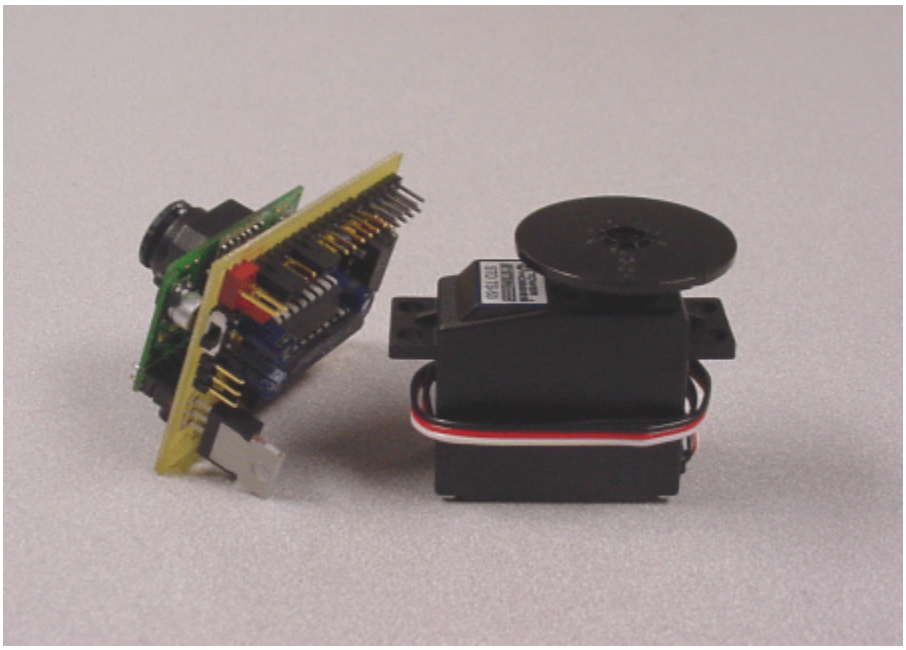
CMUcam Vision Board



User Manual

Contents

This is the CMUcam Manual v2.00 for the CMUcam v1.12 firmware.
For more information go to http://www.cs.cmu.edu/~cmucam or contact us at cmucam@cs.cmu.edu

Edited by Charles Rosenberg and Illah Nourbakhsh

Introduction

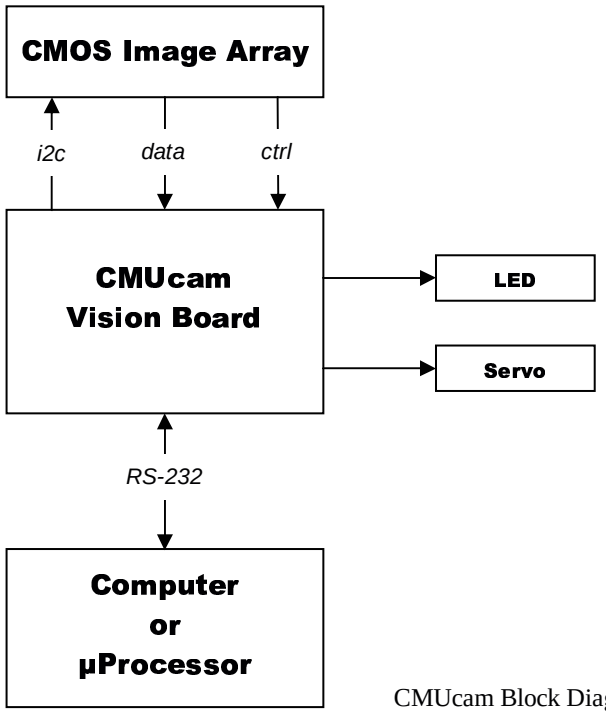
The CMUcam is a SX28 microcontroller (http://www.ubicom.com/products/processors/sx28ac.html) interfaced with a OV6620 Omnivision CMOS camera (http://www.ovt.com/omnicmoss.htm) on a chip that allows simple high level data to be extracted from the camera's streaming video. The board communicates via a RS-232 or a TTL serial port and has the following functionality:

- 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
- Automatically detect a color and drive a servo to track an object upon startup
- Ability to control 1 servo or have 1 digital I/O pin

Typical Configurations and Uses

The CMUcam 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 CMUcam 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 CMUcam. As is the case with a normal digital camera, this type of processing might require a computer or at least a fast microcontroller.

The most common configuration for the CMUcam is to have it communicate to a master processor via a standard RS232 serial port. This "master processor" could be a computer, PIC, Basic Stamp, Handy Board, Brainstem or similar microcontroller setup. The CMUcam 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 CMUcam 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 CMUcam supports various baud rates to accommodate slower processors. For even slower processors, the camera can operate in "poll mode". (See "PM" command pg 22.) 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. (See "DM" command pg 19.) Due to the communication delays, both poll mode and delay mode will lower the total frame rate that can be processed.



CMUcam Block Diagram

Getting Started

The CMUcam usually comes as a kit or as a fully assembled unit. If the camera is in a kit form, the first steps should be to assemble the kit. This will require that you are already comfortable with soldering electronic boards. Refer to the "Assembled Board View" on page 12 to get a general idea of what your final board should look like. In almost all cases, the Scenix SX28 chip that came with the CMUcam will already be programmed. You do NOT need to buy an SX-Key unless you wish to upgrade the firmware yourself or modify the current CMUcam firmware. Once you have the board constructed, make sure that you have a serial cable and power cable for the camera. The CMUcam can use a supply which produces anywhere from 6 to 9 volts of DC power capable of supplying at least 200mA of current. This can be provided by either an AC adaptor (possibly included) or a battery supply. These should be available at any local electronics store.

The camera consists of two major components, the CMUcam board and the CMOS camera module. This CMOS camera module must be attached to the CMUcam at all times in order for the system to function. Make sure it is connected so that it matches the orientation shown on the cover page of the manual.

Once you have the board assembled, the next step is to double check all of your connections. Make sure that the positive side of your power plug is facing away from the main components on the board. If the camera came with an AC adaptor, make sure that the connector locks into the socket correctly (It should have a plastic extrusion that makes plugging it in backwards very difficult). Now that the camera has power, connect the serial link between the camera and your computer. This link is required initially so that you can test and focus your camera. See the serial link segment of the "Ports" section on page 13 for more information about connecting the serial cable. Check to make sure that the "Programming Switch" is in the off position. If your board has only one switch, then the programming switch was replaced by a jumper. This switch or jumper should be in the off position (for jumpers, this means that the jumper IS inserted).

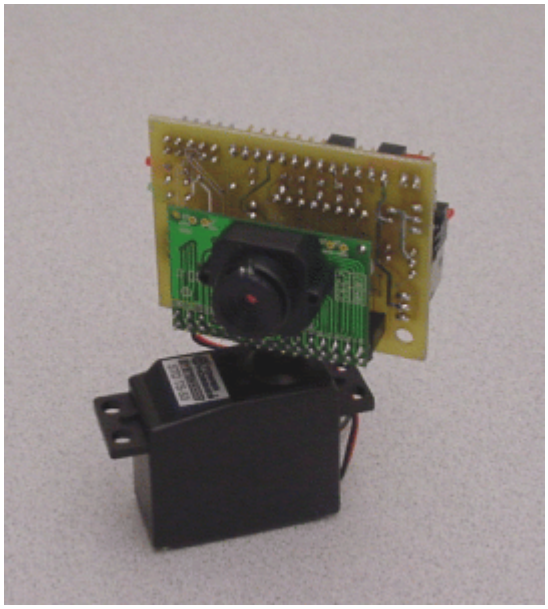
Once everything is wired up, try turning the board on. If all is well, the power LED should illuminate. Now you should be ready to test your CMUcam (see next page). If the LED does not illuminate, double check all of your connections and confirm that your power supply is correctly working.

Testing the Board

Once you have set the board up and downloaded the firmware, a good way to test the system is to connect it to the serial port of a computer.

Step 1: If one does not already exist, build a serial and/or power cable (see previous section for help).
Step 2: Plug both of them in.
Step 3: Open the terminal emulator of your choice. Examples of terminal emulators are TERA term (Windows), Hyper Terminal (Windows Built in) and Minicom (Unix)
Step 4: Inside the terminal emulator set the communication protocol to 115,200 Baud, 8 Data bits, 1 Stop bit, no parity, local echo on, and if possible turn on "add line feed" (add \n to a received \r). These setting should usually appear under “serial port” or some other similar menu option.
Step 4: Turn on the CMUcam board; the Power LED should light up and the Track LED should not.
Step 5: You should see the following on your terminal emulator:
CMUcam v1.12
:
Once you have seen this, the board was able to successfully configure the camera and start the firmware.
Step 6: Type gv followed by the enter key. You should see the following:
:gv
ACK
CMUcam v1.12
:
This shows the current version of the firmware. If this is successful, your computer serial port is also configured correctly and both transmit and receive are working.
Step 7: Focus the camera lens using the CMUcam java GUI or by graphically interpreting the Dump Frame Packet yourself. Usually the lens is focused when it sits a few rotations out from its closest position to the CMOS array. Turning the lens and re-dumping the frame incrementally should provide a good feel for when the image is sharpest. (See “Focusing with the CMUcam GUI” on the next page for more information)

Step 8: Try running the camera in demo mode (see page 9).



Focusing with the CMUcam Graphical User Interface (GUI)

When you first run your CMUcam, the lens will most likely not be in focus. In order to focus the camera you need to look at some dumped images. The easiest way to do this is using a graphical user interface that can display the CMUcam frame dump packets. One option is to use the CMUcamGUI, a java program that can be found on the CMUcam website.

Step 1: Testing if you have java installed

The first step is to determine if your computer already has java installed. The easiest way to do this is go to the "start bar" under windows and select "run". Inside the run dialog, type "command" to get a dos prompt. In unix or later versions of the Mac OS, open up a shell. Now try typing "java -version" into your command line. If a message that says "java version "1.x.xx" appears then java is installed! If instead you get "command not found" or some similar message, then you need to go to java.sun.com and download a copy of java (J2SE, JDK, JRE are all valid things to install). Sun should have platform specific instructions on how to install java. Also be sure that your version of java is newer than version 1.3. If it is not, then you will need to download a new copy of java.

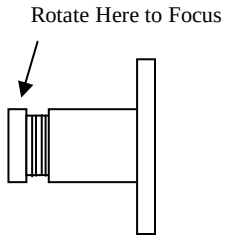
Step 2: Running the CMUcamGUI

Once you have java installed, download a copy of the latest CMUcamGUI java program. Unzip the CMUcamGUI.zip file. Now, go back to the DOS prompt or shell that you used in step 1. Using "cd", navigate to the CMUcamGUI directory that you just unzipped. You can type "dir" (dos) or "ls" (unix) to see the contents of your current directory. Once you are inside the CMUcamGUI directory make sure that you see a file called "CMUcamGUI.class". If you do not see that file, then either you did not decompress the the ZIP file, or you are in the wrong directory. If you see the CMUcamGUI.class file, then type "java CMUcamGUI".

Step 3: Grabbing a Frame

You should now see a dialog box that asks you to select the correct COM port. In windows, type in the number of the COM port that the CMUcam is connected to and press the "okay" button. In unix, make sure that the path to your com port is correct and then press "okay". The CMUcamGUI should now open and display the message "Camera OK and idle…" in the "Output Window" dialog box. That means that the CMUcamGUI found and was able to communicate with the camera. Once this works, go to the "Commands" menu and select "Dump Frame". After a few seconds you should see an image appear in the window.

Step 4: Focusing



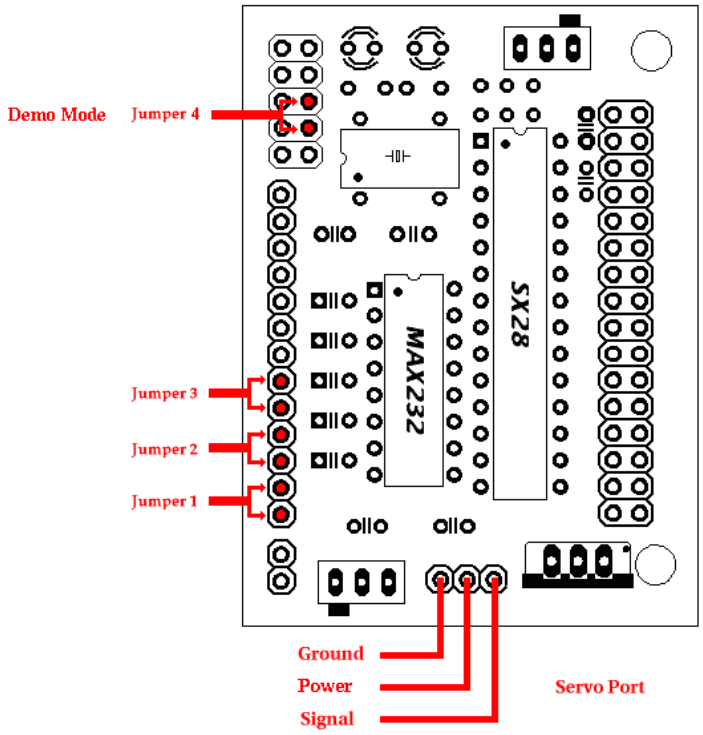
CMUcam Lens Mount

Once you have the ability to grab frames from the camera, you should be able to rotate the front part of the CMUcam 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.

Demo Mode

Jumper 4 puts the camera into a demo mode. Demo mode causes the camera to call the track window command and then begin outputting a standard hobby servo PWM signal from the servo output. The servo attempts to drive the camera mounted on it towards the middle mass of the color detected on startup. You need to plug the servo into the CMUcam's servo port as can be found on the diagram below. Try and center the servos range of motion before you attach the CMUcam.

Note that upon powering up in this mode, the camera will wait for 5 seconds before grabbing a color and servoing. When it first starts up auto gain and white balance are both on. Then after they have stabilized for 5 seconds the camera turns them off, switches into YCrCb mode and calls TW. (See page 25 for TW details) The servo tracking can be enabled manually (without the jumper) by sending the MM 2 command and then calling TW. If, when the servo is mounted on the camera, it appears to be tracking in the incorrect direction also set jumper 1. When jumper 1 and jumper 4 are set, the board does not go into slave mode; it runs demo mode and reverses the direction of the servo. Jumper 4 uses the UART port, which inhibits the use of the serial port while in Demo Mode.



The following steps are performed during power up in demo mode:

1. RS is sent to reset the camera.

2. Pause 5 seconds to allow the camera's auto adjusting parameters to stabilize.

3. The camera register string "CR 18 32 19 32" is sent.

   This selects YCrCb mode and turns off auto gain.

4. Execute the "TW" command.

Notes on Better Tracking

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

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.

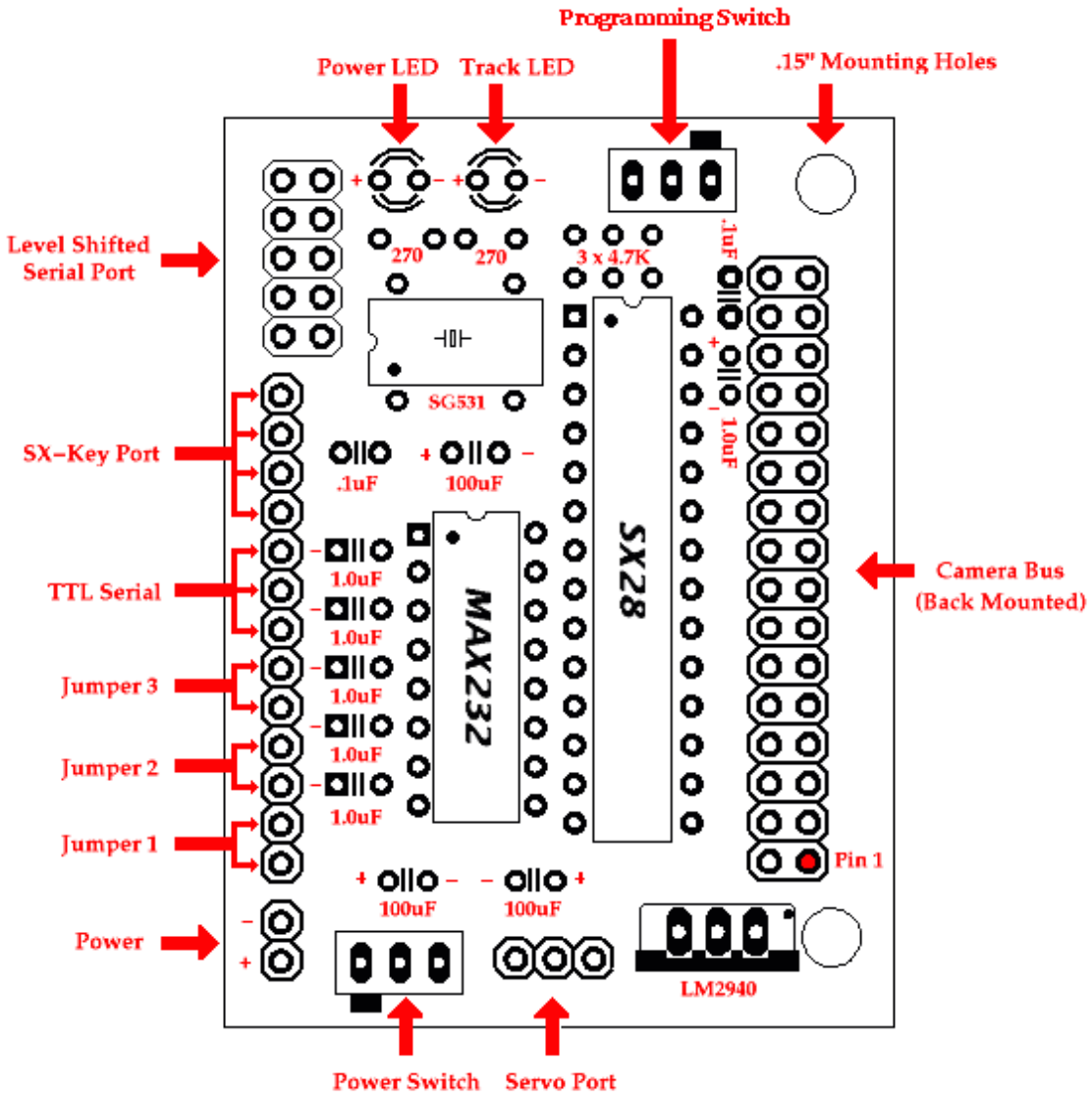
The functions provided by the camera board are meant to give the user a toolbox of color vision functions. Actual applications may greatly vary and are left up to the imagination of the user. The ability to change the viewable window, grab color / light statistics and track colors can be interwoven by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

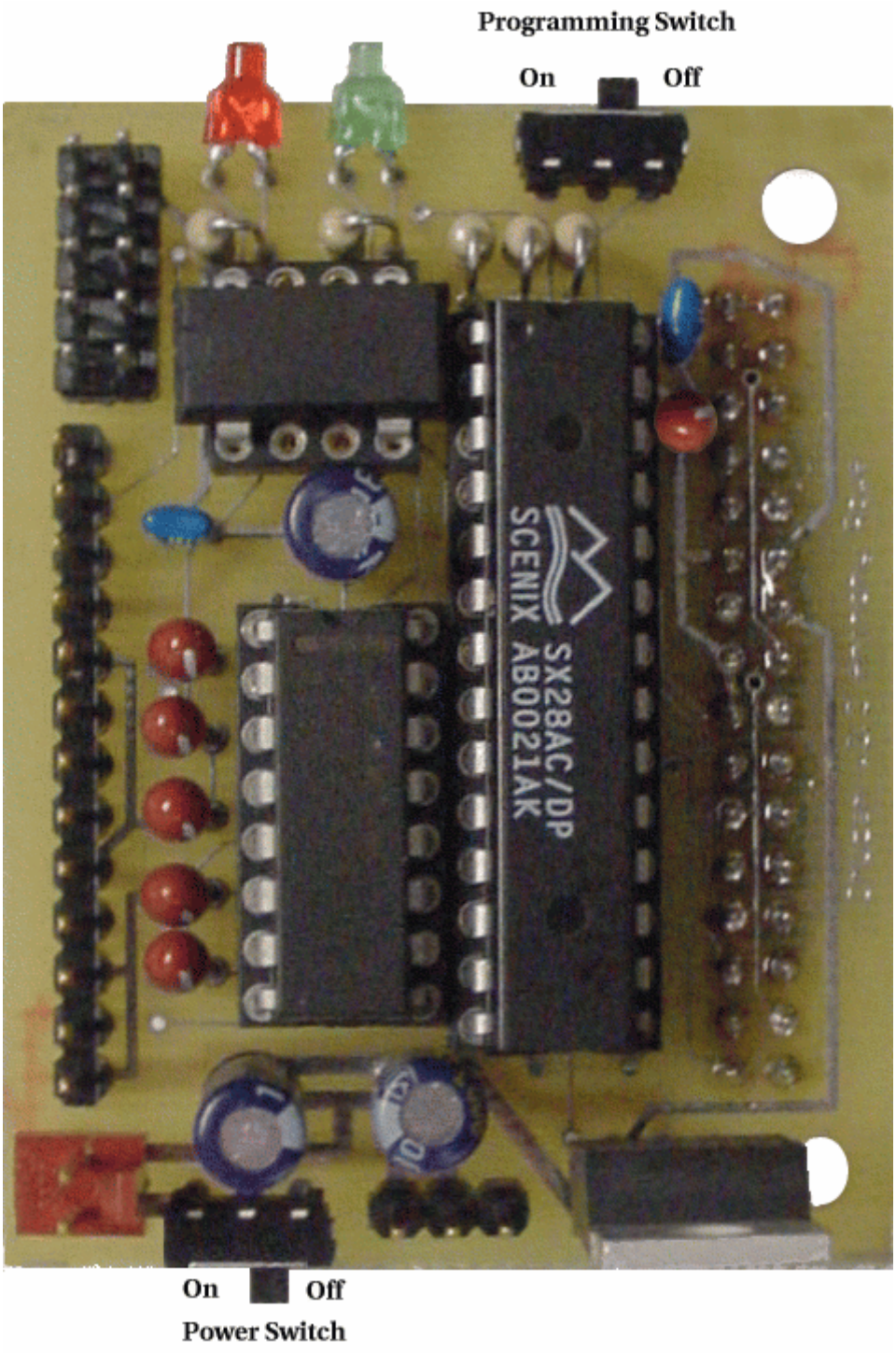
Analog Video Output

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 CMUcam 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.

Board Layout



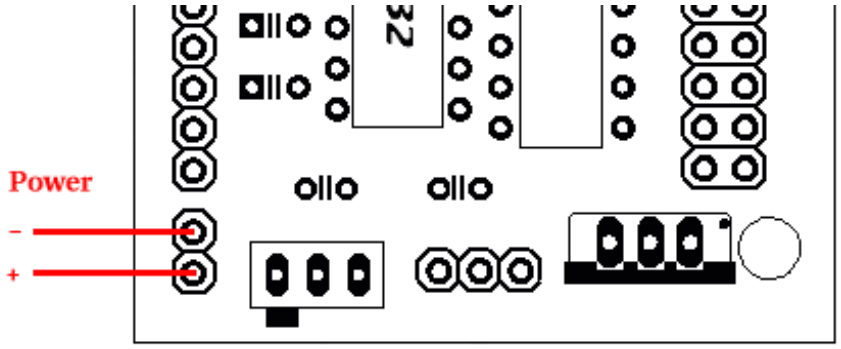
Assembled View



Ports

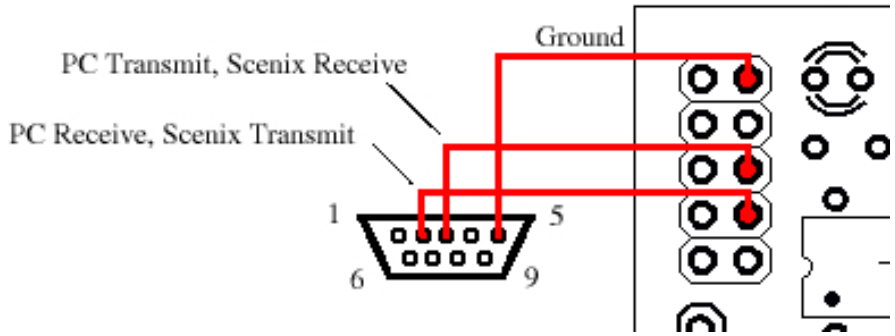
Power

The input power to the board goes through a 5 volt regulator. It is ideal to supply the board with between 6 and 9 volts of DC power that is capable of supplying at least 200 milliamperes of current.



Level Shifted Serial Port

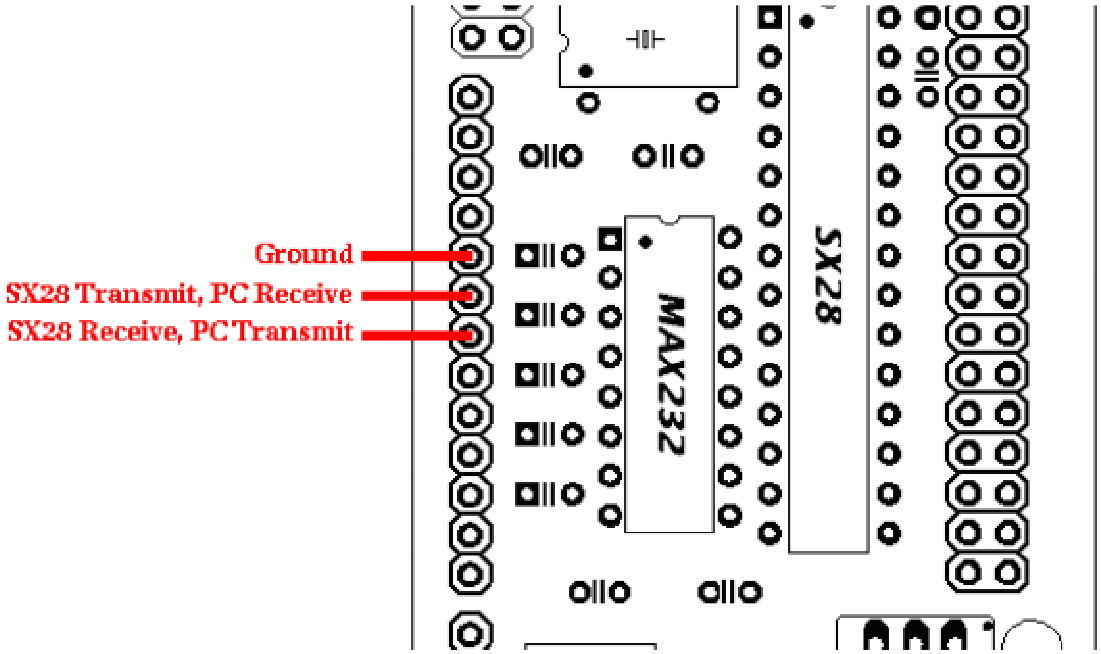
This port provides full level shifting for communication with a computer. Though it only uses 3 of the 10 pins it is packaged in a 2x5 pin configuration to fit standard 9 pin ribbon cable clip-on serial sockets and 10 pin female clip on serial headers that can both attach to a 10 wire ribbon cable. If this initially does not work, try flipping the direction that the ribbon cable connects to the CMUcam board.



The trapezoidal serial connector shown above is what the serial connector on your computer should look like.

TTL Serial Port

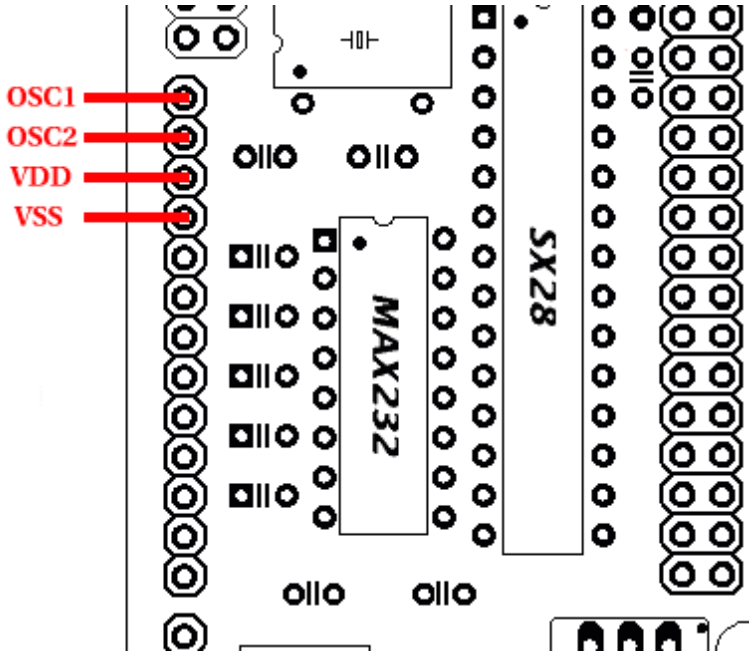
This serial port taps into the serial I/O before it goes through the MAX232 chip. This port may be ideal for communication with a microcontroller that does not have any built in level shifting. Note that, it is necessary to remove the max232 chip to use this port.



Programming Port

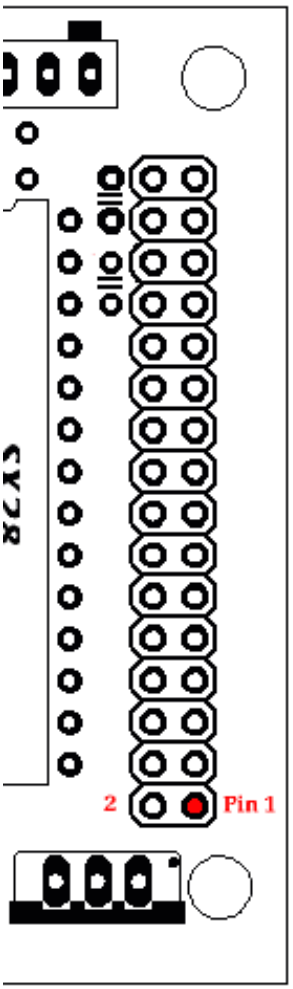
The programming port allows the firmware to be downloaded to the SX28 using a SX-Key / Blitzer or equivalent programmer. These can be purchased from Parallax Inc (www.parallaxinc.com). The SX28 is usually provided preprogrammed.

(Circuit Side Up)



Camera Bus

This bus interfaces with the CMOS camera chip. The CMOS camera board is mounted parallel to the processing part of the board and connects starting at pin 1. The female camera header should be soldered on the back of the board. (See assembled view.) For information about the Video Analog Output port see page 10.

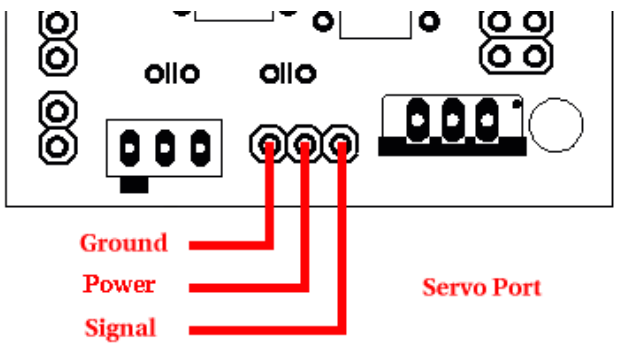


Pin Description

1~8	Y0~Y7	Digital Output Y Bus
9	PWDN	Power Down Mode
10	RST	Reset
11	SDA	I2C Serial Data
12	FODD	Odd Field Flag
13	SCL	I2C Serial Clock
14	HREF	Horizontal Window Reference
15	AGND	Analog Ground
16	VSYN	Vertical Sync
17	AGND	Analog Ground
18	PCLK	Pixel Clock
19	EXCLK	External Clock
20	VCC	+5 VDC
21	AGND	Analog Ground
22	VCC	+5 VDC
23~30	UV0–UV7	Digital output UV bus
31	GND	Common Ground
32	VTO	Video Analog Output (75 Ohm)

Servo Port

This is the output for the servo. The servo power does not go through a regulator from the board's power input. Do not use a servo with the board if the board is being run off of more than 6 volts.



Jumpers

Parallel Processing in Slave Mode (Jumper 1)

The CMUcam supports a mode of operation that allows multiple boards to process data from the same camera. If a PC104 style pass-through header is used instead of the standard double row female header, it is possible to rack multiple boards along the same camera bus. Upon startup, if jumper 1 is set, the camera becomes a slave. Slave mode stops the camera board from being able to configure or interfere with the CMOS camera's settings. Instead it just processes the format setup by the master vision board. When linking the buses together you must only have one master; all other boards should be setup to be in slave mode. In this current version of the system there is no message passing between boards other than the image data from the camera bus. This means you have to communicate to each slave board via a separate serial link. This communication to the board should be identical to using a single CMUcam. For example, you could have the master board tracking some color while the slave board could be told to get mean color data. Each board runs independently of one another and only the master can control camera registers.

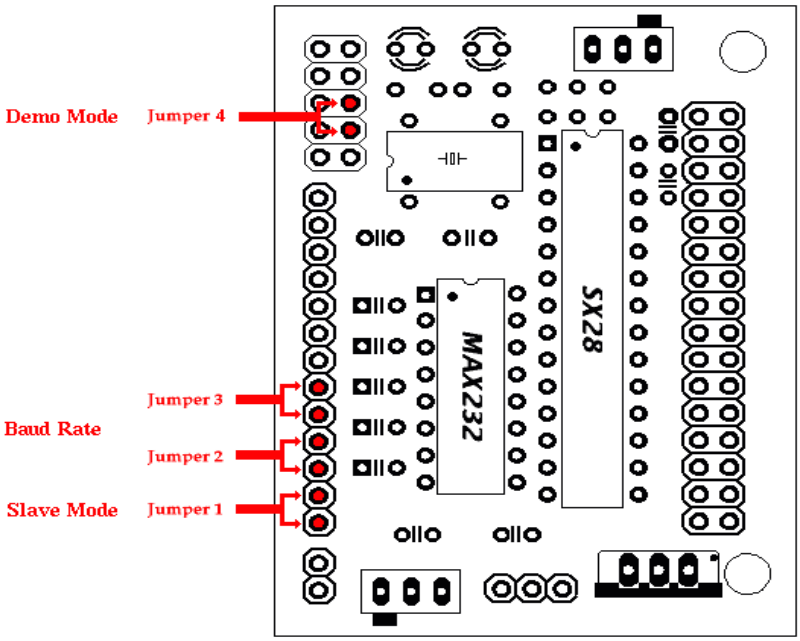
Baud Rate (Jumpers 2 and 3)

115,200 Baud	Jumper 2 Open	Jumper 3 Open
38,400 Baud	Jumper 2 Set	Jumper 3 Open
19,200 Baud	Jumper 2 Open	Jumper 3 Set
9,600 Baud	Jumper 2 Set	Jumper 3 Set

Because of the extra time it takes to transmit data at lower rates, frames may be skipped resulting in a lower frame rate. The slower rate will also cause the bitmap mode and dump frame resolutions to shrink.

Demo Mode (Jumper 4)

Jumper 4 puts the camera into a demo mode. Demo mode causes the camera to call the track window command and then begin outputting a standard hobby servo PWM signal from the servo output. The servo attempts to drive the camera mounted on it towards the middle mass of the color detected on startup. For more information see the "Demo Mode" section on page 8.



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”.)

\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
:

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.

Common Settings:

Register	Values	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 (default)
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
19 Auto Exposure		
	32	Auto Gain Off
	33	Auto Gain On (default)

Example of decreasing the internal camera clock speed (default speed is 2)
:CR 17 5
ACK
:

*****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 **D**ump 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 column by column with a frame synchronize byte and a column synchronize byte. (This data can be read and displayed by the CMUcamGUI 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 - new frame
2 - new col
3 - end of frame
RGB (CrYCb) ranges from 16 - 240
1 2 r g b r g b ... r g b r g b 2 r g b r g b r ... r g b r g b ...

Example of a Frame Dump from a terminal program
(WARNING: This may temporarily interfere with a terminal program by sending non-visible characters)
:DF
ACK
*maKP(U A$IU AL<>U A$L*YL%*L L (G AUsonthAYA(KMAy098a34ymawvk....*

DM *value*\r

This command sets the **D**elay before characters that are transmitted over the serial port and 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 correlates to approximately the transfer time of one bit at the current baud rate.

GM\r

This command will **G**et the **M**ean color value in the current image. If, optionally, a sub-region of the image is selected, 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 (See page 10). 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

GV\r

This command **G**ets the current **V**ersion 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 **H**alf-horizontal resolution **M**ode 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 **I**nput. 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

L1 *value*\r

This command is used to control the tracking **L**ight. 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

LM *active*\r

This command turns on **L**ine **M**ode 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

MM *mode*\r

This command controls the **M**iddle **M**ass 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. The *mode* value acts as a bit field for the following operations. Setting the bits enables the functionality.

mode = B_3 B_2 B_1 B_0

Bits	
B_0	Enable / Disable Middle Mass Mode
B_1	Enable / Disable the Servo Output
B_2	Change Servo Direction
B_3	Return N-type packet that includes the current servo position (see page 27.)

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 **N**oise **F**ilter 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 **P**oll **M**ode. 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
:

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. 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.

bit_flags = $B_2\ B_1\ B_0$

Bits	
B_0	Output to the camera is in raw bytes
B_1	"ACK\r" and "NCK\r" confirmations are suppressed
B_2	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$&,.L

RS \r

This command **R**e**S**ets the vision board. Note, on reset the first character is a /r.

Example of how to reset the camera
:rs
ACK

CMUcam v1.12
:

S1 *position* \r

This command lets you **S**et the position of servo 1. 0 turns the servo off and holds the line low. 1-127 will set the servo to that position while it is tracking or getting mean data. Any value 128 and higher sets the line high. In order for the servo to work, the camera must be in either a tracking loop or mean data gather loop. Values 0 and 128 can be useful if you wish to use the servo port as a digital output. The port can also be used as a digital input (see I1 command). The "MM" command can enable or disable the automatic servo tracking.

SM *value* \r

This command is used to enable the **S**witching **M**ode 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

SW [x y x2 y2] \r

This command **S**ets the **W**indow 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. Do NOT try "SW 0 0 80 144", this is outside of the 1 1 80 143 bounds.

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 **T**rack a **C**olor . It takes in the minimum and maximum RGB (CrYCb) values and outputs a type M or C or N data packet (set by the MM command). The smaller type C packet encodes a bounded box that contains pixels of the curently 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. Remember that the color values from the CMOS camera will range from between 16 and 240 (See page 10).

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

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 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.

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

1. Shrink the window to 1/4 the size (in each dimension) of the current window. Position the new window at the center of the camera's field of view. (sw 30 54 50 90)
2. Call the get mean command. (gm)
3. Restore the window to the full image size. (sw 1 1 80 143)
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 (not that Middle Mass mode is not active):
:TW
ACK
S 240 50 40 12 7 8
C 2 40 12 60 10 70
C 2 41 12 61 11 70

Output Data Packet Descriptions

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 **C** packet:

C x1 y1 x2 y2 pixels confidence\r

This is the return packet from a color tracking command.
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)*256 of the bounded rectangle and capped at 255

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 col 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 cols 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!

Type **M** packet:

M mx my x1 y1 x2 y2 pixels confidence\r

This is the return packet from a color tracking command with Middle Mass mode on.
mx - The middle of mass x value
m*y* - 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)*256 of the bounded rectangle and capped at 255

Type **N** packet:

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

This is identical to a type **M** packet with an added value for the servo position.
spos – The current position of the servo

Type **S** data packet format:

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation\r

This is a statistic packet that gives information about the camera's view
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
R*deviation* - the *deviation of red or Cr found in the current window
G*deviation*- the *deviation of green or Y found in the current window
B*deviation*- the *deviation of blue or Cb found in the current window

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

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

Firmware Upgrades

CAUTION: This is not usually necessary because the firmware usually comes preprogrammed!

Once the board has been assembled and powered up, you can modify and download new firmware using the SX-Key downloading program. The SX-Key downloader and manual are available at:
www.parallaxinc.com/html_files/downloads/downloads_sx.htm
The firmware for the camera can be found at:
http://www.cs.cmu.edu/~cmucam/downloads.html

Downloading The Firmware

After downloading and installing the SX-Key software, it is necessary to load the CMUcam.hex firmware to the board.

step 1: Turn on the board's power.
step 2: Ensure that the programming switch is in the program mode (pushed towards the LEDs see Assembled view)
step 3: Plug in the SX-key, text side facing away from the board. (for more details see the hardware port section)
step 4: Open up the SX-Key software
step 5: Select "Device" from under the "Run" menu
step 6: Click on the "Load Hex" button
step 7: Go to the directory of the CMUcam.hex file and type its name into the dialog box. (It may not be visible)
step 8: Press "Open" to load it
step 9: Press the "Program" button
step 10: Once programming has finished, turn off the power to the board
step 11: Unplug the SX-Key and switch the programming switch into the run position
step 12: Once the power switch is turned on again the firmware should be running and will greet you with

CMUcam v1.xx
:

Download Troubleshooting

If, when you tried to program, you see or experience:

SX-Key not found
-Make sure the comm port is set correctly and power is going to the camera
Vpp Generation Failed
-Ensure that the program switch is in the correct position
Programming Failed
- Try it again and if it still occurs consult the SX-Key manual
No response
-Make sure you switched the board out of program mode. Try power cycling the board.

General Troubleshooting

In Demo Mode, the light turns on for a second and then everything stops

When both the camera and servo are active, the power required is greater. Try using a battery or voltage source rated at a higher current.

When I run java I get: Exception in thread "main" java.lang.NoClassDefFoundError: CMUcamGUI

Chances are you are not in the CMUcamGUI directory. Type “dir” at the command line prompt and make sure that you see the CMUcamGUI.class file.

I see CMUcamGUI.java but I don’t see the CMUcamGUI.class file

You should download a new copy of the GUI, because the .class files should be included. If you really need to recompile them, type “javac *.java” .

I get: 'java' is not recognized as an internal or external command, operable program or batch file.

This means that java is not correctly installed in your path. Try re-installing java and reading Sun’s documentation about setting the “classpath” variables correctly.

The power LED does not glow

The board either has a fault, or your power supply is not generating enough power. Check the power supply and look over all of the soldier connections. Try unplugging all of the cables except power and turn it on again.

I get garbage output from the camera

Try turning the camera off and unplugging it for 10 seconds. Then plug it back in and try again.

I get wavy lines or a distorted black and white image when I call dumpframe

This is most likely due to power. Make sure that you have a high enough voltage and that you are getting a clean signal. Running the camera off of fresh batteries (not an AC adaptor) is a good way to test if this is the problem.

My processor can not keep up with the serial data stream

Try running the camera in poll mode and setting a delay mode value. See pages 19 and 22 for “PM” and “DM” details.

I don’t seem to get any serial data

Make sure that the serial cable is connected on the CMUcam side correctly. If in doubt, try reversing it.

Why does SW keep giving me a NCK?

Make sure you are within the SW 1 1 80 143 bounds.

I see the CMUcam startup message, but then nothing happens

Check to make sure the transmit line on your serial cable is connected correctly.

Parts List

The following is a list of parts needed to assemble the CMUcam.

Qty	Item	Part no. (Digi-key)	Schematic Label	Unit Price
1	75 MIPS Ubicom SX28	SX28AC/DP-ND	IC1	5.18
1	OV6620 Eval Board *	BB048*		57.95
1	Maxim 232 Level Shifter	MAX232CPE-ND	IC3	3.31
1	SG-531 75.00 MHz crystal osc	SE2911-ND	QG1	5.85
1	5 Volt Regulator	LM2940CT-5.0-ND	IC2	1.80
3	4.7 Kohm 1/4 Watt Resistor	4.7KQBK-ND	R2 R3 R4	0.06
2	220 ohm 1/4 Watt Resistor	220QBK-ND	R1	0.06
6	1.0 uF Capacitor	P2105-ND	C1 C2 C3 C4 C5 C6	0.42
2	SPST Switch	EG1847-ND	S1 S2	1.11
1	3mm red LED	67-1125-ND	LED3MM	0.16
1	3mm green LED	67-1127-ND	D1	0.16
2	0.1uF Capacitor	P4923-ND	C9 C10	0.55
3	100 uF Capacitor	P5138-ND	C7 C8 C11	0.28
2	14 pin IC socket (to form a 28 pin socket)	AE8914-ND	IC1	0.32
1	8 pin IC socket **	ED3308-ND	QG1	0.42
1	16 pin IC socket	ED3316-ND	IC3	0.83
1	Double row female header	929852-01-36	JP5	3.35
1	Single row male header	929647-09-36-ND	JP1x2 JP3 JP4 JP7	1.83
(1)	Polarized 2 pin terminal housing	WM2700-ND		0.25
(2)	Crimp terminals	WM2200-ND		0.13
(1)	Polarized 2 pin terminal header	WM2000-ND		0.25
(1)	Female serial ribbon cable head	AFS09G-ND		6.14
(1)	Serial ribbon cable socket connector	ASC10G-ND		1.11

*Order Code BB048 at http://www.electronics123.com. Ask them for an evaluation board with an OV6620 from Omnivision and an IR coated lens.

**Use this for the oscillator by cutting off the inner 4 legs

( ) accessories that make interfacing easier, but are not required for functionality

CMUcam Schematic