

We looked on handyboard.com and on the message board as well as google. The google searches usually just ended up with people selling CMUcam or were links to message boards where no one as far as we found gave a good answer. We couldn't find any specific instructions on how to get/use the raw data, but did read the CMUcam Manual. It didn't have the greatest instructions on how to use the raw data, but still we extracted most of useful info. Here, we explain some of CMUcam's general functions. If you look at the second half of the document, there is more information on the actual raw data material. Although we could not quite figure out how it works, we hope that this information is useful.

Here is some general documentation on the CMU Cam and some additional information on how to implement shape tracking using a bitmap

init_camera(); This function only needs to be done one time before the camera is used.

clamp_camera_yuv(); This is the function that does the white balancing.

setWin(x1, y1, x2, y2); Sets the window size

Global Variables

track_size stores the approximate number of pixels matching in the blob
(range is 0 – 254)

track_x stores the pixel x coordinate of the centroid of the color blob

track_y stores the pixel y coordinate of the color blob(note: 0,0 is the center; 40,80 is upper right and -40,-80 is lower left)

track_area stores the size of the bounding rectangle of the color blob (range is 0, 11076).

track_confidence stores the confidence for seeing the blob

Defined Variable

```
#define cmu_WAIT 14
#define cCMUCAM_DATA_SIZE 17

//N message starts here
#define cmu_SERVO 0

//M message starts here, N matches
#define cmu_MX 1
#define cmu_MY 2

//C message starts here, N and M match
#define cmu_X1 3
#define cmu_Y1 4
```

```
#define cmu_X2 5
#define cmu_Y2 6
#define cmu_PIXELS 7
#define cmu_CONFIDENCE 8

//S message starts here returned by TW and GM starts below
#define cmu_RMEAN 10
#define cmu_GMEAN 11
#define cmu_BMEAN 12
#define cmu_RDEV 13
#define cmu_GDEV 14
#define cmu_BDEV 15

#define CMUCAM_DATA_SIZE 17
```

Related Variables (stored in `cmcsci2a.asm`)

`cmu_packet;` //stores what kind of packet was returned 'M' or 'C'
`cmu_new_val;` //set to 1 when a packet is received and 0 when file being transferred
`cmu_start;` //pointer to the head of the storage array. Set to point to `cmucam_data[]`.

Two Main Important Buffers:

1) persistent char `cmucam_data[CMUCAM_DATA_SIZE];`

The array stores the data that is returned from the camera. The type of the packet is stored in the variable 'cmu_packet'. The remaining values depend on the type of the first.

2) persistent char `cmdbuff[35];`

This is the raw command buffer the minimum raw command is 3 bytes.

Format: 2 chars command id such as SW(set window), TC(track color), or GM. Next byte is the number of the //parameter bytes that follow. A `cmdbuff[2] == 0`, implies there are no more bytes to //command.

Maximum length command appears to be the CR of 16 register settings.

Functions

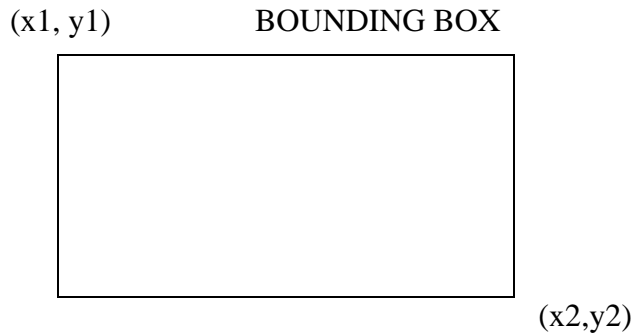
Result data from **track()**

```
int track_size = cmucam_data[cmu_PIXELS];
//track size is the number of pixels of the target
```

```
int track_x = (cmucam_data[cmu_X1] + cmucam_data[cmu_X2]) / 2;
//track_x is the X value of the centroid
```

```
int track_y = (cmucam_data[cmu_Y1] + cmucam_data[cmu_Y2]) / 2;
```

```
//track_y is the Y value of the centroid  
  
int track_area = (cmucam_data[cmu_X2] - cmucam_data[cmu_X1] *  
                 (cmucam_data[cmu_Y2] - cmucam_data[cmu_Y1]));  
//track_area is area of the bounding box
```



void setWin(int x1, int y1, int x2, int y2)

The set window function. The maximum window size parameters are (1, 1, 80, 143). (parameters (top left corner x, top left corner y, bottom right corner x, bottom right corner y)).

```
//track color function  
//returns -1 if the command fails  
//returns 0 if the command succeeds and nothing is being tracked  
//returns a positive number if a confident color is detected.  
//parameters red min, red max, green min, green max, blue min, blue max
```

```
int trackRaw(int rmin, int rmax, int gmin, int gmax, int bmin, int bmax);  
//this function sets all the global variables  
//track_size, track_area, track_confidence, track_x, track_y  
//
```

```
persistent int CAM_MINDELTA = 3;  
persistent int CAM_DEVFACT = 1;
```

cmdbuff ← Command buffer that is sent to “send_R_command(cmdbuff, n);”

SHAPE TRACKING

LM active\r (active is Boolean variable)

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 (**T**rack **C**olor) and TW (**T**rack **C**olor **F**ound in **C**entral **R**egion of the current **W**indow):

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 other words data is transferred by rows instead of columns. 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 (**G**et the **M**ean **C**olor value of the image in the current window):

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

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 **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

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

y_2 -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 **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

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

*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