



CMUCAM V1.12 interface
Specific for RS232
By: Alexis Mesa

Relevant Documentation:

Before tackling the CMUCAM, you must become familiarized with the following documents:

XMEGA Manual:

http://www.atmel.com/dyn/resources/prod_documents/doc8077.pdf

AVR XMEGA AI Device Datasheet:

http://www.atmel.com/dyn/resources/prod_documents/doc8067.pdf

CMUCAM manual

<http://www.seattlerobotics.com/CMUcamManualv15A.pdf>

Objective:

Interface the CMUCAM V1.12 via RS232 with the computer using the java applet provided by the Seattle Robotics and HyperTerminal in order to calibrate (focus) the camera and become familiarized with the commands. As seen in Figure 1 focusing is an essential part in performance. Then write code for the PVR board using the RS232 serial connection.

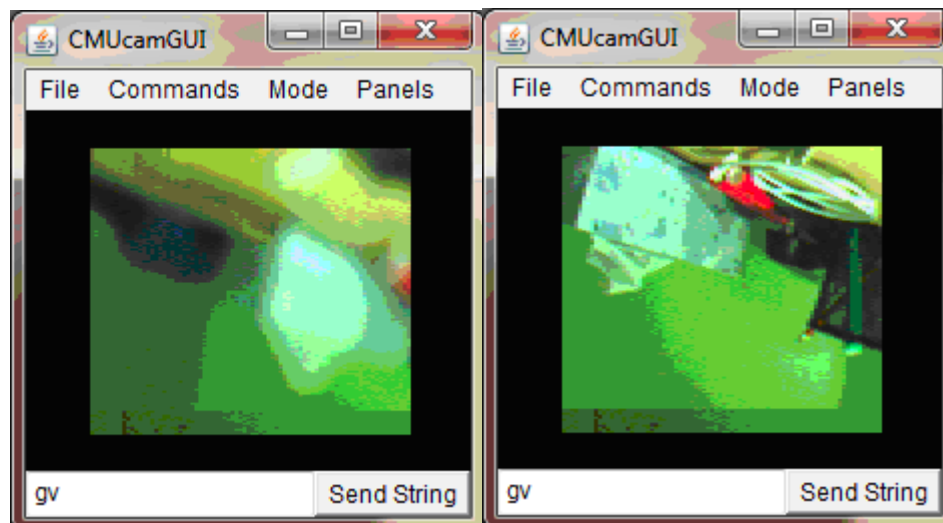


Figure 1

Procedure:

Section 1: Hardware

If you don't have the Boe-Bot version skip to the next section:

If you do have the Boe-Bot version a couple extra steps must be taken:

In order to connect to the computer an RS232 connection must be established

The Boe-Bot version is stripped down hence it does not include a level shifter (MAX2332), a couple of capacitors and a power supply. Hence the board must be populated with the blue components of Figure 2 except the Regulator. A 5V 200mA power supplied must be provided to the power terminal (according to PVR manual on 500mA could be taken out of the 5V power supply) and a couple of jumpers must be soldered as seen in Figure 3

Jumper 1 is used so that the power switch is on all the time.

Jumper 2 bypasses the regulator hence it is not necessary.

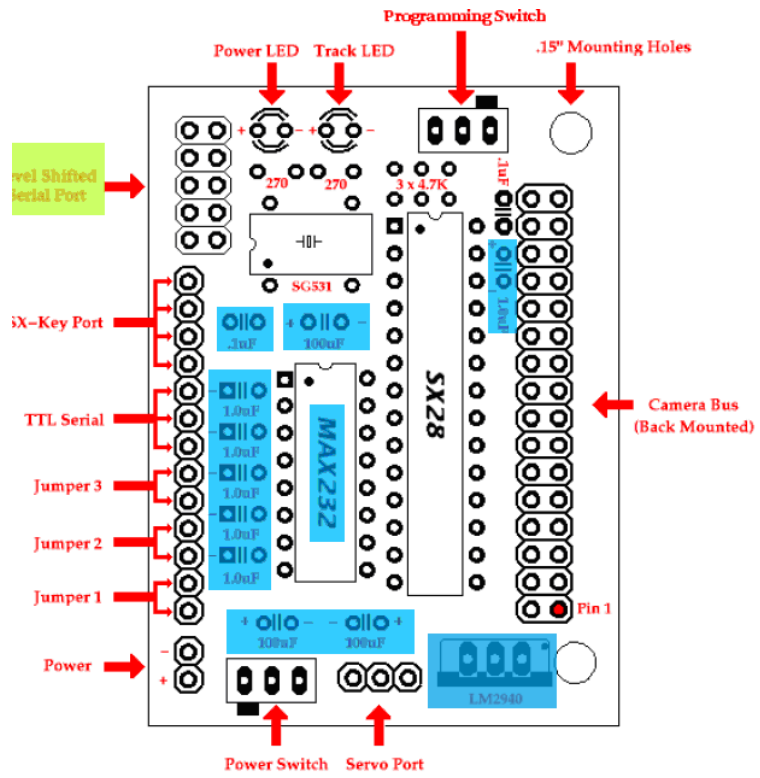


Figure 2

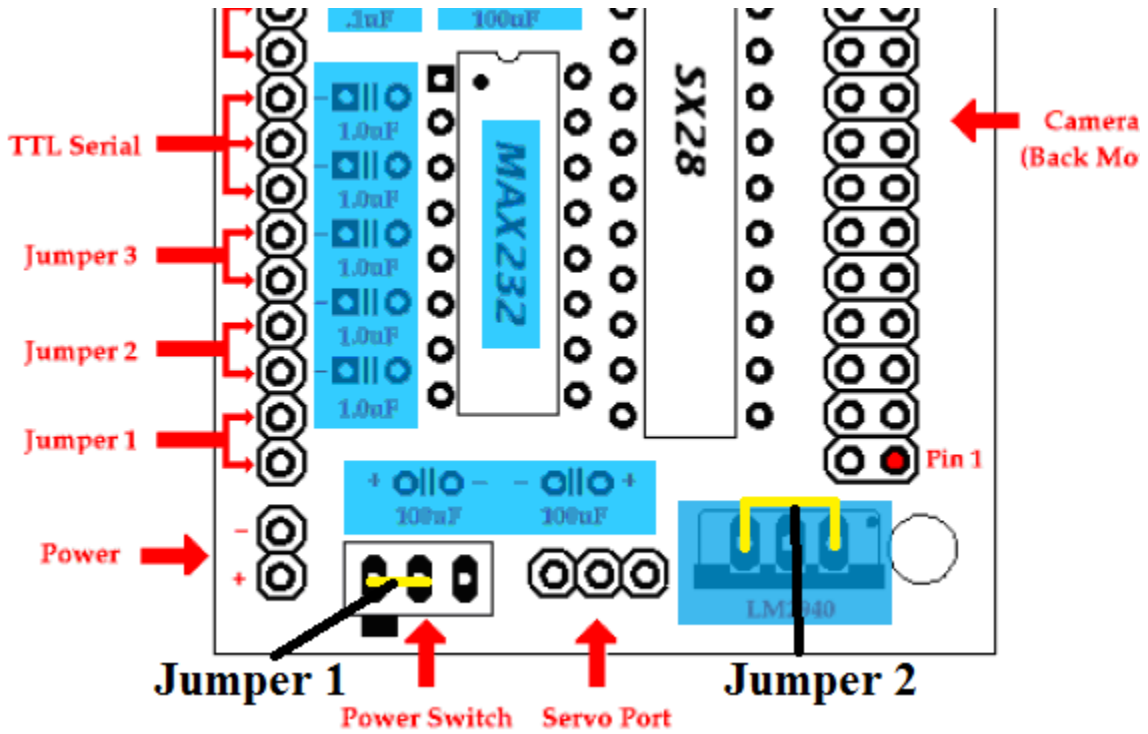


Figure 3

Once all the components (see Bill of Materials on CMUCAM manual) and jumpers are soldered, board is ready for power up and start communicating with the computer.

Section 2: Communication with the computer (HyperTerminal).

From the CMUCAM Manual:

Step 1: If one does not already exist, build a serial and/or power cable as seen in Figure 4

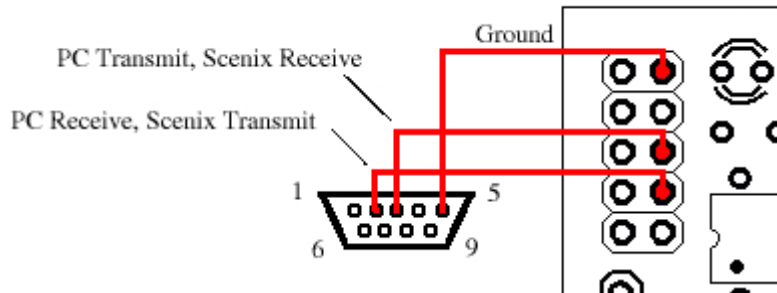


Figure 4

Step 2: Plug both of them in.

Step 3: Open Hyper Terminal (Windows Built in). If you have Windows 7, HyperTerminal is not installed hence you need to download it. There are plenty of free 30 day trials online.

Step 4: Inside the terminal emulator use a COM port, set the communication protocol to 115,200 Baud, 8 Data bits, 1 Stop bit, no parity. Once opened, go to “properties”, “Settings” tab, ASCII Setup and make sure send line feeds, append line feed and echo are check as seen in Figure 5.

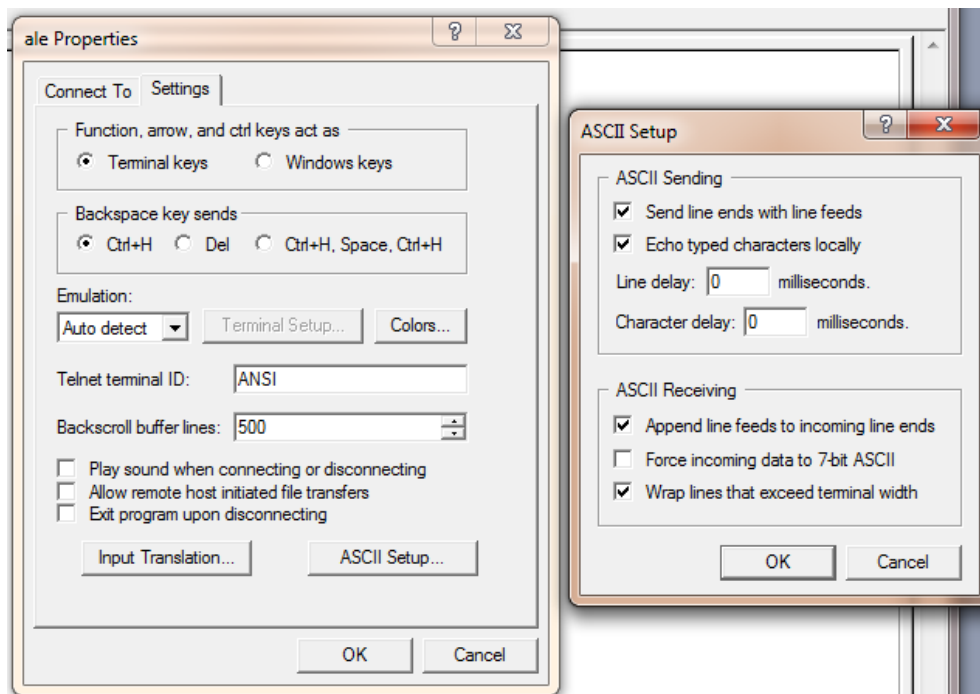


Figure 5

Step 5: Turn on the CMUcam board; the Power LED (red) should light up and the Track LED should not. Make sure that on the bottom left part of HyperTerminal says connected and that the Baud rate jumpers on the board are off (Baud Rate of 115200bps, see page 16 of CMUCAM manual)

Step 6: 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.

If nothing happens check if power is connected (red LED), check baud rates of both the HyperTerminal and Board, check of connection is the same as Figure 4 and if the right COM port was selected

Step 7: 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.

Section 3: Communication with the computer (java applet).

In this section you will calibrate the camera.

Go to <http://www.cs.cmu.edu/~cmucam/downloads.html> and download the beta version GUI. From the CMUCAM Manual:

Step 1: 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 2: 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 3: Focusing

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.

I strongly suggest to use java program for calibration and the HyperTerminal to play around with the commands since the java program is plagued with bugs.

Section 4: Communication via RS232 on the PVR (code is almost the same for TTL)

Since an RS232 connection must be made in order to communicate with the computer, it is convenient to use the same connection to interface the board. However if TTL connection is preferred, the code shown could be adapted to it.

Commented code for communication via RS232 on the PVR TX0 RX0

```
//Initializaiton
PORTE_DIR = PIN3_bm; //Pin 3 of port E is output
PORTE_OUT = PIN3_bm; //Pin 3 of port E is TX0
PORTE.DIRCLR = PIN2_bm; //Pin 2 of port E is RX0
USARTE0_CTRLA = 0x03; // USART Control Register C: ASYNCHRONOUS,
no parity 1 stop bit 8 bit word
USARTE0_BAUDCTRLA = 0x06; // Page 238 of Atmel manual, fbaud=115200 (my
case)=32MHz/(16*((2^BSCALE) * BSEL)+1))
USARTE0_BAUDCTRLB = 0xC1; // BSEL= 262 and BSCALE= -4 in 2s comp .
USARTE0_CTRLB |= 0x08; // TX0 is on
USARTE0_CTRLB |= 0x10; // RX0 is on
static char *temp;

//Program
CMUsend("RS\r"); //Reset the camera
delay_ms(5000); //Long enough to wait for ACK and to get the camera ready

CMUsend("L1 1\r"); //To turn on green light
temp=CMUreceive(); //Receive ACK
delay_ms(5); //For some reason that I havent figure out, there must be a delay between
commands
CMUsend("PM 1\r"); //Activate polling mode
temp=CMUreceive(); //Receive ACK
delay_ms(5);

CMUsend("TC 155 255 0 30 0 30\r"); //Track color
temp=CMUreceive(); // Receive ACK
temp=CMUreceive(); // Receive M packet
delay_ms(5);

lcd_write(temp); //LCD Write function

//CMUCAM functions
void CMUsend(char *command)
{
    int i = 0;
    while (command[i] != '\0') //While command does not end do...
    {
        // Data Register Empty Flag: check if data register is empty
        while (!(USARTE0_STATUS & (1<<USART_DREIF_bp)));
        //USARTE0_DATA is shared by the transmit and receive
    }
}
```

```

        USARTE0_DATA = command[i];

        //if data is sent TXCIF is set
        while (!(USARTE0_STATUS & (1<<USART_TXCIF_bp)));
        i++;
    }
}

char CMUreceive(void)
{
    int i=0;
    static char *data;
    do
    {
        // wait for receive is complete
        while (!(USARTE0_STATUS & (1<<USART_RXCIF_bp))){ }
        //Put data into string
        data[i] = USARTE0_DATA;
    } while (data[i++] != '\r'); //Since all transfers are ended by \r wait for it to happen
    return data;
}

```