Kyle Tripician

Sensor Report
Intelligent Machines Design Laboratory
CMU Camera

Introduction

The Aluminator is an autonomous robot that searches an area for a red aluminum can and retrieves it to dispense it in a trash area. The robot utilizes many sensors in order to do this, the most important of which is the CMU camera. It uses the camera to find and track objects to then approach them. The CMU camera is a "smart camera" meaning, it is able to perform many specialized tasks. The feature I used on the camera was the "Track Color" command which enabled me to input different color coordinates, and it would return the middle mass coordinates of an object of that color range specified is found. Interfacing the camera with a computer proves to be simple and the GUI is quite user friendly. The problem at hand was interfacing it with a microprocessor.

CMU Camera

Configuring the CMU camera with an Atmel ATMega323 processor proved to be quite a feat. However, it was neither the Camera, nor the processor that gave me problems. The board I was using, the progressive MegaAVR development board seemed to be wired incorrectly. On the board, there is a male header that is supposedly connected to the transmit, receive and ground pin of the board. I was using this header to interface my hardware together, and I could get the processor to transmit to the board, but I could not get the camera to transmit to the camera. After weeks of agony I realized that the board was faulty, and the receive pin on the male header was not connected to the actual receive pin of the USART. By making a cable that connected straight to the DB9 port I was able to fix my problem and the board transmitted and received properly with the camera.

The CMU camera track color command sends back a packet of information of the coordinates of an object that it is tracking. The most used packet, the M-type packet sends back a packet similar to: M|middlex|middley|x1|y1|x2|y2|pixels|confidence. When tracking a color I was most concerned with the middle x coordinate.

One disadvantage to the CMU camera is that it sends everything back in ASCII format. In order to avoid nasty ASCII to Hex conversions, you need to enable Raw Mode, which gives you the option to send data back from the camera in "raw" format. This makes it easier for the processor to read the data. In raw mode, the M packet is begun with a 255 byte followed by an M, then the data. Code for integrating the camera with an ATMega323 chip can be found at the end of this report.

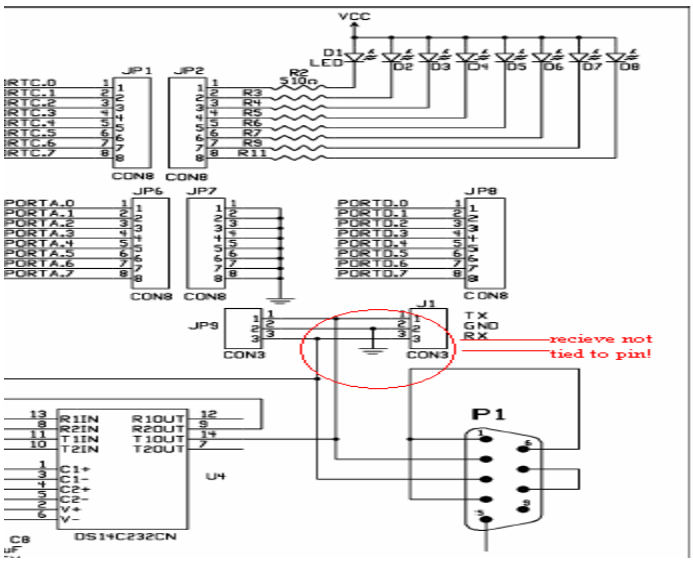

Figure 1. Progressive board schematic. Receive header not tied to receive pin.

Had it not been for the board that I was using to interface the chip and the camera with, I wouldn't have had as many problems as I did. Overall I find that the camera is a very simple to use piece of hardware, that can be very effective.

```
**********Code**********************************************
Kyle Tripician
EEL5666
Mega AVR Development board w/
Atmel ATMega323 Processor
UART Initialization for the CMU Camera
@ 38400 Kbps, 8 databits, 1 stop bit, no parity
The following code can be used and modified for educational purposes
With proper acknowledgement
**********************************************************

#include <io.h>
#include <string.h>
#include <stdlib.h>
#include <progmem.h>
#include <interrupt.h>
#include <sig-avr.h>

//Initialize UART
void uartinit(void)
{
  outp(0x98,UCSRB); //enable transmit, receive, and receive complete interrupt
  outp(0x86,UCSRC);
  outp(0x00,UBRRH);
  outp(0x09,UBRRL);
}

//Send a single byte of data
void uarttransmit(unsigned char data)
{
  // Wait for empty transmit buffer
  while (!(UCSRA & (1<<UDRE))){}
  // Put data into buffer, sends the data
  outp(data,UDR);
}


//Send a given CR terminated string
void uartstring(unsigned char * myStringIn)
{
  unsigned char *myString = myStringIn;
  unsigned char ch1;
  unsigned char gotNULL = 0;
  ch1= *myString++;
  while(!gotNULL){
    uarttransmit(ch1);
```

```
   ch1 = *myString++;
   if(ch1 == '\r'){
     gotNULL = 1;
     uarttransmit(ch1);
     }

   }
}
**************cmu.c******************
/*
Kyle Tripician
UART code.
...YET ANOTHER TRY*/

#include <io.h>
#include <sig-avr.h>
#include <stdlib.h>
#include <interrupt.h>
#include <progmem.h>
typedef unsigned char u08;
typedef unsigned int u8;
typedef           char s08;
typedef unsigned short u16;
typedef           short s16;
#include "delay.h"
volatile u8 temp;
volatile u8 i=0;
volatile u8 cmudat[9];


SIGNAL(SIG_UART_RECV){
        temp=inp(UDR);
        if(temp != 0x3A){
                if(temp == 0x20 || i==9){
                i=0;
                }
                else if(i==0){
                        if(temp == 0xFF){
                                cmudat[i]=temp;
                                i++;
                        }
                }
                else if(temp !=0x20 && i<9){
                cmudat[i]=temp;
                i++;
                }
```

```
        }

}
void blink(void){
        uartstring("L1 1\r");
        delay(1500);
        delay(1500);
        delay(15000);
        uartstring("L1 0\r");
        delay(15000);
        delay(1500);
        delay(1500);

        }
int main(void){
        outp(0xFF,DDRC);
        delay(10000);
        uartinit();
        delay(10000);
        uartstring("PM 1\r");
        delay(1000);
        uartstring("RM 3\r");
        delay(1000);
        sei();
        delay(1000);

        while(1){
        uartstring("TC 150 250 16 16 16 16\r");
        outp(cmudat[8],PORTC);
        delay(20000);
        }
}
```