LAMPIRAN A

FOTO ROBOT TERBANG
Tampak Samping Kiri

Tampak Samping Kanan

A-2
LAMPIRAN B

PROGRAM PADA PENGONTROL MIKRO

ATTINY2313
/* *************************************************************************/
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 12/17/2009
Author  : F4CG
Company : F4CG
Comments:

Chip type           : ATtiny2313
Clock frequency     : 11.059200 MHz
Memory model        : Tiny
External SRAM size  : 0
Data Stack size     : 32
***************************************************************************/

#include <tiny2313.h>
#include <delay.h>
#include <stdio.h>

unsigned int x,s;
unsigned char dat[32];

#define RXB8 1
#define TXB8 0
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index, rx_rd_index, rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
    char status, data;
    status = UCSRA;
    data = UDR;
    if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN)) == 0)
    {
        rx_buffer[rx_wr_index] = data;
        if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index = 0;
        if (++rx_counter == RX_BUFFER_SIZE)
        {
            rx_counter = 0;
            rx_buffer_overflow = 1;
        }
    }
    if (data == 255) // carry return
    {
        x = 0;
        s = 1;
    }
    if (s == 1)
    {
        dat[x] = data;
        x++;
    }
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
    char data;
    while (rx_counter == 0);
    data = rx_buffer[rx_rd_index];
    if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index = 0;
    #asm("cli")
    --rx_counter;
    #asm("sei")
    return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Declare your global variables here

void main(void)
{
    // Declare your local variables here

    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=0x80;
    CLKPR=0x00;
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    // Input/Output Ports initialization
    // Port A initialization
    // Func2=In Func1=In Func0=In
    // State2=T State1=T State0=T
    PORTA=0x00;
    DDRA=0x00;

    // Port B initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTB=0x00;
    DDRB=0x00;

    // Port D initialization
    // Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTD=0x00;
    DDRD=0x00;

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: Timer 0 Stopped
    // Mode: Normal top=FFh
    // OC0A output: Disconnected
    // OC0B output: Disconnected
    TCCR0A=0x00;
    TCCR0B=0x00;
    TCNT0=0x00;
    OCR0A=0x00;
    OCR0B=0x00;

    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: Timer 1 Stopped
    // Mode: Normal top=FFFFh
    // OC1A output: Discon.
    // OC1B output: Discon.
    // Noise Canceler: Off
    // Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// Interrupt on any change on pins PCINT0-7: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Universal Serial Interface initialization
// Mode: Disabled
// Clock source: Register & Counter=no clk.
// USI Counter Overflow Interrupt: Off
USICR=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 38400
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x06;
UBRRH=0x00;
UBRRL=0x11;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;

// Global enable interrupts
#asm("sei")
printf("RS\\r"); // Reset the camera
delay_ms(100); // must delay …can delay less than 100ms
printf("\\r");
delay_ms(100);
printf("sv 0 122\\r");
delay_ms(100);
printf("PM 0\r\n"); // turn poll mode on
delay_ms(100);
printf("RM 3\r\n"); // turn on raw data mode for packets received from camera
delay_ms(100);
printf("SW 1 1 80 143\r\n"); // not necessary this is default window
delay_ms(100);
printf("tc 100 200 16 60 16 60 \r\n"); //printf("tc 16 50 60 180 60 160 \r\n");
delay_ms(300);
while (1) {
    // Place your code here
    if (dat[9] >= 1) {
        printf("RS\r\n"); // Reset the camera
delay_ms(100); // must delay …can delay less than 100ms
        printf("sv 0 0\r\n");
delay_ms(800);
    }
    // else
    // {
    //     printf("RS\r\n"); // Reset the camera
    //     delay_ms(100); // must delay …can delay less than 100ms
    //     printf("sv 0 200\r\n");
    //     delay_ms(600);
    // }
    printf("RS\r\n"); // Reset the camera
delay_ms(100);
    printf("sv 0 190\r\n");
delay_ms(600);
    printf("tc 100 200 16 60 16 60 \r\n"); //printf("tc 16 50 60 180 60 160 \r\n");
delay_ms(200);
}
Sensor Kamera CMUCam2+
Setting Up the Hardware

In order to initially test your CMUCam2, you will need a serial cable, a power adapter and a computer. The CMUCam2 can use a power supply which produces anywhere from 6 to 15 volts of DC power capable of supplying at least 200mA of current. This can be provided by either an AC adapter (possibly included) or a battery supply. These should be available at any local electronics store. The serial cable should have been provided with your CMUCam2.

Make sure that you have the CMOS sensor board connected to the CMUCam2 board so that it is in the same orientation as the picture shown on the cover of this manual.

First, connect the power. Make sure that the positive side of your power plug is facing away from the main components on the board. If the camera comes with an AC adapter, make sure that the connector locks into the socket correctly.

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. The serial cable should be connected so that the ribbon part of the cable faces away from the board. You must also connect the serial pass through jumper.

Check to make sure that the clock jumper is connected. This allows the clock to actively drive the processor.

Once everything is wired up, try turning the board on. The power LED should illuminate green and only one LED should remain on. Both LEDs turn on upon startup, and one turns off after the camera has been successfully configured.
Demo Mode

Demo mode causes the camera to call track window and then drive two standard hobby servos towards the object being tracked. This can be initiated autonomously at startup. First you need to plug a pan and/or tilt servo into serve ports 0 and 1. Servo port 0 is for the pan, while 1 is for the tilt. Next, make sure that the servos are being powered by either the internal servo power jumper or by an external power source. While holding down the push button, turn the camera on. The tracking LED should begin rapidly blinking. Immediately release the push button and wait for the LED to stop blinking. Next, point the camera at a colored object and press the push button again. This should grab the color of the object and begin automatically serving towards it. If the servos appear to be driving in the reverse direction, add the appropriate servo direction jumper. During this period when the LED is blinking, the camera is adjusting to the light conditions in the room. Try not to hold the object in front of the camera while this is occurring. Experiment with different colors and lighting. You will notice that some work much better than others.

The following steps are performed during power up in demo mode:
1. RS is sent to the camera
2. Pause 5 seconds while blinking the LED to allow the camera to stabilize
3. The Camera register string "CK 18 31 19 32" is sent.
4. Auto Servo Mode is enabled.
5. TW is called.
Board Layout

Parts

Power
The input power to the board goes through a 5 volt regulator. It is ideal to supply the board with between 6 and 15 volts of DC power that is capable of supplying at least 200 milliamperes of current.

The servos can either be powered by internal power, or by the external servo power connector. To run them off of internal power, connect a jumper across the "servo internal power jumper". To run them off of external power, leave the jumper open, and connect another 5 volt supply to the servo power connector. Do not connect an external servo supply while the servo power jumper is in place. If the servos are drawing too much power or seem to be noisy, try soldering a large valued capacitor across the "Extra Servo Capacitor" connections. The external servo power is not switched by the main power switch.
Serial Port

The CMUCam2 has a standard level shifted serial port to talk to a computer as well as a TTL serial port for talking to a microcontroller. The level shifted serial port only uses 3 of the 10 pins. It is in a 2x5 pin configuration that fits a standard 9 pin ribbon cable clip-on serial socket 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 CMUCam2 board. Make sure the serial jumper is in place when you use this mode.

The TTL connector can be used to talk to a microcontroller without the use of a level shifting chip. It operates between 0 and 5 volts. Remove the Serial Jumper when you use this mode.

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.
Servo Port

The CMICore has the ability to control 5 servos. This can be useful if you do not wish to use a separate servo controller. The servo port can also be used as a general purpose digital outputs.

Configuration Jumpers

The jumpers can be used to set the camera's baudrate or configure various modes of operation.

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>115200 Baud</td>
<td>000</td>
</tr>
<tr>
<td>57600 Baud</td>
<td>001</td>
</tr>
<tr>
<td>38400 Baud</td>
<td>002</td>
</tr>
<tr>
<td>19200 Baud</td>
<td>011</td>
</tr>
<tr>
<td>9600 Baud</td>
<td>010</td>
</tr>
<tr>
<td>4800 Baud</td>
<td>010</td>
</tr>
<tr>
<td>2400 Baud</td>
<td>011</td>
</tr>
<tr>
<td>1200 Baud</td>
<td>001</td>
</tr>
</tbody>
</table>

X - jumper closed
_ - jumper open

Pan and Tilt Reverse Jumpers

During Auto Serve Mode, or demo mode it may be necessary to reverse the direction of the pan or tilt servo. Connecting the pan and/or tilt jumper will cause auto serve mode to send the opposite command to each servo. Note, this only works for auto serve mode, and not for normal servo operations.
Slave Mode Jumper

The CMUCam2 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 the “3M” jumper 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 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 set up to be in slave mode. In the 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 CMUCam2. 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.

Auxiliary I/O

The CMUCam has 4 auxiliary Input Output ports that can be used for reading data from external devices. Note, that pin 3 is used for the sleep deeply command.

![Diagram of CMUCam2](image-url)
Analog Video Port

Using the OV7620 camera module, you will be able to get a PAL video signal from the analog port of the CMUCam2. This would sync up with any PAL monitor, but will not work with a standard NTSC monitor.

The OV7620 camera module will output a standard black and white NTSC video signal.

To use this output, it is necessary to keep the camera at its maximum frame rate (the default) and switch it into YCrCb mode in order to see the image on a monitor.
## Functionally Grouped Command Listing

### Buffer Commands
- BS Buffer Mode 38
- SF Send Frame 47

### Camera Module Commands
- CM Camera Register 18
- CP Camera PIR 52
- CO Camera Type 12

### Data Rate Commands
- DR Delay Mode 18
- DT Dist Mode 46
- DS Dist Mode 47
- DL Delay Mode 48
- DM Delay Mode 48

### Safe Commands
- ST Active Position 18
- SF Active Distance 12
- SM Safe Mode 36
- SE Active Mode 11
- SO Active Output 11

### Image Window/Commands
- SW Small Frame 56
- SH Small Capture 55
- SW Small Window 55
- SH Small Mode 36
- SW Get Window 38
- FD Frame Difference 40

### Auxiliary I/O Commands
- CB Get Button 18
- CI Get Available CI 18
- LID LED Control 18

### Alphabetical Command Listing
- BS Buffer Mode 38
- CO Camera PIR 52
- CM Camera Register 18
- CP Camera Type 12
- CS Camera Status 32
- CF Camera Focus 32
- CY Capture Type 22
- CH Capture Status 22
- CM Camera Mode 36
- CS Capture Status 32
- CT Capture Type 32
- CM Camera Mode 36
- CO Capture Status 32
- CY Capture Type 32
- CH Capture Status 32
- CM Capture Mode 36
- CO Capture Status 32
- CY Capture Type 32
- CH Capture Status 32
- CM Capture Mode 36
- CO Capture Status 32
- CY Capture Type 32
- CH Capture Status 32
- CM Capture Mode 36