```
/*****************************************
************

This program was produced by the

CodeWizardAVR V2.03.4 Standard

Automatic Program Generator

© Copyright 1998-2008 Pavel Haiduc, HP
InfoTech s.r.l.

http://www.hpinfotech.com


Project :

Version :

Date    : 12/30/2015

Author  :

Company :

Comments:


Chip type           : ATmega32

Program type        : Application

Clock frequency     : 12.000000 MHz

Memory model        : Small

External RAM size   : 0

Data Stack size     : 512

*****************************************
************/


#include <mega32.h>


#include <delay.h>

#include <stdio.h>

#include <stdlib.h>

#define servo1 PORTB.1

#define servo2 PORTB.2

#define servo3 PORTB.3

#define servo4 PORTB.4

#define motor  PORTB.0


#define g PORTD.2

#define w PORTD.3

#define r PORTD.4

#define b PORTD.5


unsigned char tampil[16];

int pwm,s1,s2,s3,s4,kaizo,kaizo1;

int i,a,x,y,z,r1,g1,b1,w1,kr1,kg1,kb1,kw1;

int warna,scan,sr,sb,sg,temp;

// Alphanumeric LCD Module functions

#asm

   .equ __lcd_port=0x15 ;PORTC

#endasm

#include <lcd.h>


#define RXB8 1

#define TXB8 0

#define UPE 2

#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;

    };

  };
}


#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


// USART Transmitter buffer
```

```
#define TX_BUFFER_SIZE 8

char tx_buffer[TX_BUFFER_SIZE];


#if TX_BUFFER_SIZE<256

unsigned char tx_wr_index,tx_rd_index,tx_counter;

#else

unsigned int tx_wr_index,tx_rd_index,tx_counter;

#endif


// USART Transmitter interrupt service routine

interrupt [USART_TXC] void usart_tx_isr(void)

{

if (tx_counter)

  {

  --tx_counter;

  UDR=tx_buffer[tx_rd_index];

  if (++tx_rd_index == TX_BUFFER_SIZE)
tx_rd_index=0;

  };

}


#ifndef _DEBUG_TERMINAL_IO_

// Write a character to the USART Transmitter
buffer

#define _ALTERNATE_PUTCHAR_

#pragma used+

void putchar(char c)

{

while (tx_counter == TX_BUFFER_SIZE);

#asm("cli")

if (tx_counter || ((UCSRA &
DATA_REGISTER_EMPTY)==0))

  {

  tx_buffer[tx_wr_index]=c;

  if (++tx_wr_index == TX_BUFFER_SIZE)
tx_wr_index=0;

  ++tx_counter;

  }

else

  UDR=c;

#asm("sei")

}

#pragma used-

#endif


// Standard Input/Output functions

#include <stdio.h>


// Timer 0 overflow interrupt service routine

interrupt [TIM0_OVF] void timer0_ovf_isr(void)

{

// Place your code here


kaizo++;

if(kaizo>2000)  kaizo=0;

if(s1>kaizo)    servo1=1;

else            servo1=0;

if(s2>kaizo)    servo2=1;

else            servo2=0;

if(s3>kaizo)    servo3=1;

else            servo3=0;
```

```
}

// Timer 2 overflow interrupt service routine
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
// Place your code here

kaizo1++;
if(kaizo1>2000) kaizo1=0;
if(s4>kaizo1)    servo4=1;
else             servo4=0;
if(pwm>kaizo1)   motor=1;
else             motor=0;

}

#include <delay.h>

#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

// Declare your global variables here

void kalibrasi()
{
    r=1;
    g=0;
    b=0;
    w=0;
    lcd_gotoxy(0,0);
    sprintf(tampil," data r %3d ",read_adc(0));
    lcd_puts(tampil);
     delay_ms(50);
     kr1=read_adc(0);
//     delay_ms(300);

    r=0;
    g=1;
    b=0;
    w=0;
    lcd_gotoxy(0,0);
     sprintf(tampil," data g %3d ",read_adc(0));
     lcd_puts(tampil);
```