Lil’ Mario

Taja Sellers
25 April 2003
EEL 5666 Intelligent Machines Design Lab
Dr. A. Antonio Arroyo
TAs: Uriel Rodriguez
Jason Plew

Table of Contents

Abstract

Lil' Mario is an autonomous robot a voice command portable CD player. It uses a voice recognition kit, IR sensors, and a CD player to provide its owner with hours of entertainment.

Introduction

Lil' Mario was an idea of mine that was developed last semester. I thought it would be fun and interesting to build a robot that would play a CD and didn't require a remote control. Since, I am always misplacing mine. This paper covers the steps needed to build Lil' Mario and make him function.

Integrated System

I used the Atmega323 evaluation board as my main microcontroller. I did require the assistance of a PIC16F873. This device was used to drive my servos. The other systems (voice recognition, obstacle avoidance and CD player) were controlled my the Atmega323. An inexpensive portable CD player provided the base for my CD player. A RF control box was built using the Voice Direct 364, TWS-434A RF transmitter, and an encoder. Lil' Mario will has a RWS-434 RF receiver on its platform. Two Sharp GP2D12 IR detectors and 3 bump switches were used for object avoidance. The bump switches are a precaution, in case the IR detectors fail. After receiving power, Lil' Mario will perform little dances that consist of going around in circles, spelling out my name, and drawing a triangle.

Mobile Platform

Lil' Mario is a circular three level platform with two wheels. The top level holds two speakers, 4 relays, decoder, and RF receiver. The second level holds a drawer which contains a CD player. The third level holds the microprocessor and other electrical components. The battery packs and servos are under the first level. A small furnisher remover slider was placed on the bottom for stability. Bump switches and IR emitters and detectors were used for obstacle avoidance, two are located on the front, while the third is on the back.

Actuation

Lil' Mario has two wheels powered by continous rotation servos. This was convenient, because I didn't have to modify them. Originally, the servos were controlled by the PWM from the Atmega323. However, the PWM stopped functioning when an external interrupt was added to my program. Therefore, a pic was used to generate the pulses used to control my servos.

Sensors

There are sensors on Lil' Mario:

- Bump swithches
- IR proximity detectors
- Voice recognition

Bump Switches

In order to preserve some of my A/D ports a voltage divider circuit was used to implement the bump switches. Figure 1 is the schmetic for the voltage divider. The picture was provided by David Winkler creator of Chester.

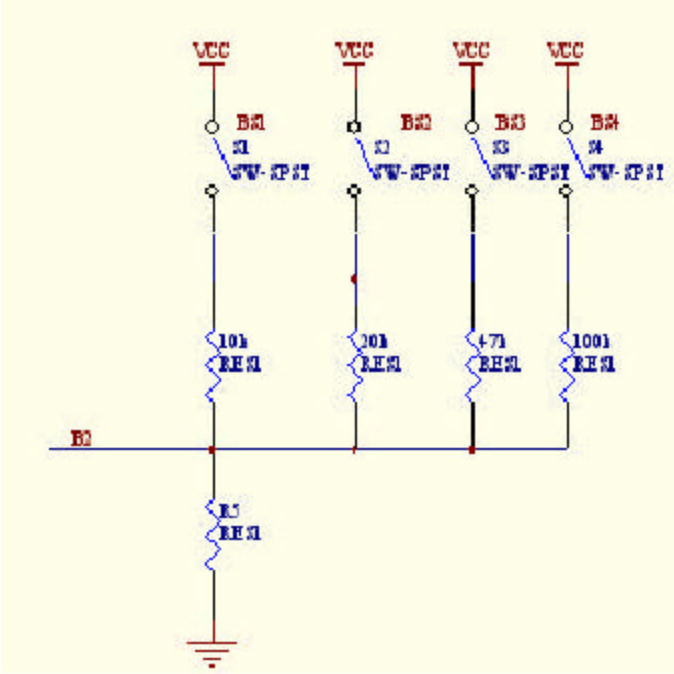

Figure 1

IR Proximity Detectors

Object avoidance will be accomplished using Sharp's GP2D12 IR detectors placed on the front of the robot at an angle from each other. These sensors can detect objects from 4" to 30" away. The except angle is not known. Only two values were used close and very close. The distance for very close was chosen to be approximately 5" and close to be about 8". This gave Lil' Mario enough time to turn around if he needed to.

Voice Recognition

I used Sensory's Voice Direct 364 speech recognition kit which can recognize up to 15 words in continuous listening mode, figure 2.

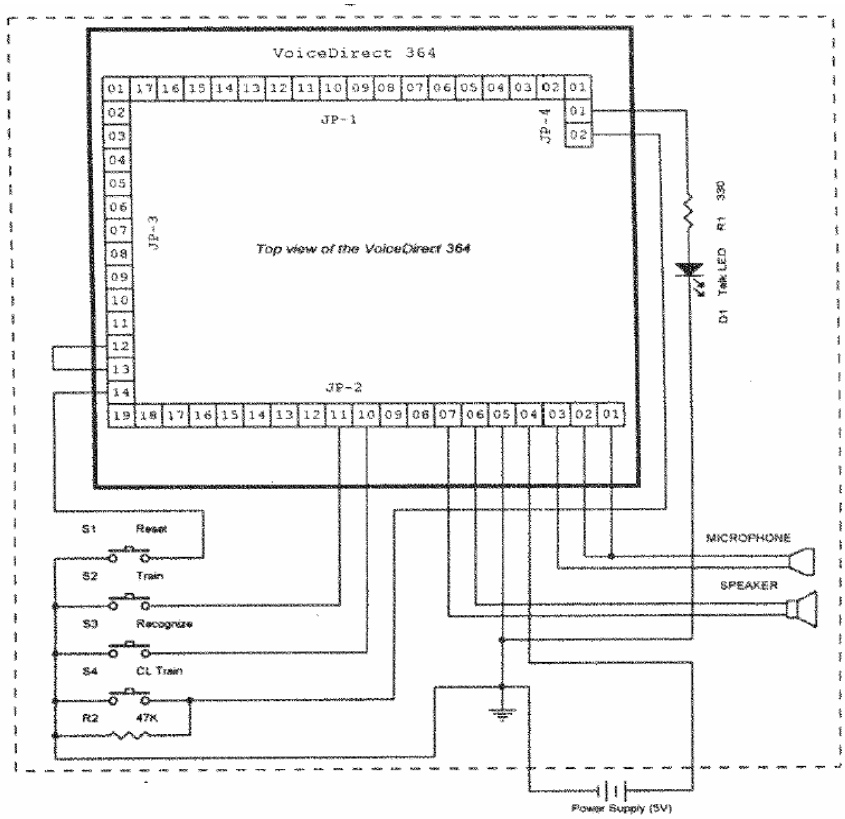

Figure 2

Table 1 shows the commands that Lil' Mario is capable of recognizing and the outputs sent out by the kit. CL is the keyword that the kit will be listening for. Once, it is recognized there is a 3 second window for the user to say the next word. The kit is a good kit considering the price. However, you have keep your distance from the microphone and voice the same during recording and recognizing. In addition, to that the word list affects the accurancy of the kit. That caused me to vary the number of syllables in my command list. I do not like the fact the output pin only goes high for a second. That has caused me some problems.

CL = Mario	Data Out
CL + Playmusic	0x01
CL + Pause	0x02
CL + Discontinue	0x04
CL + Reverse	0x08
CL + Fastforward	0x10
CL + Trackone	0x20
CL + Selecttwo	0x40
CL + Gotonumberthree	0x80
CL + Choosefour	0x81
CL + Optionfive	0x82
CL + Six	0x84
CL + Trackseven	0x88
CL + Eight	0x90
CL + Picknine	0xa0
CL + Zero	0xc0

Table 1

Behaviors

Lil' Mario will perform the following behaviors:

- Play a CD
- Respond to voice commands
- Object Avoidance
- Perform a little dance that consist of it moving around in circles and other configurations

Play a CD

I used an inexpensive portable CD player to play the CDs. I removed the play/pause, reverse, fast forward and stop buttons to uncover the pads. A piece of wire was place on each metal contact and connected as shown in figure 3.

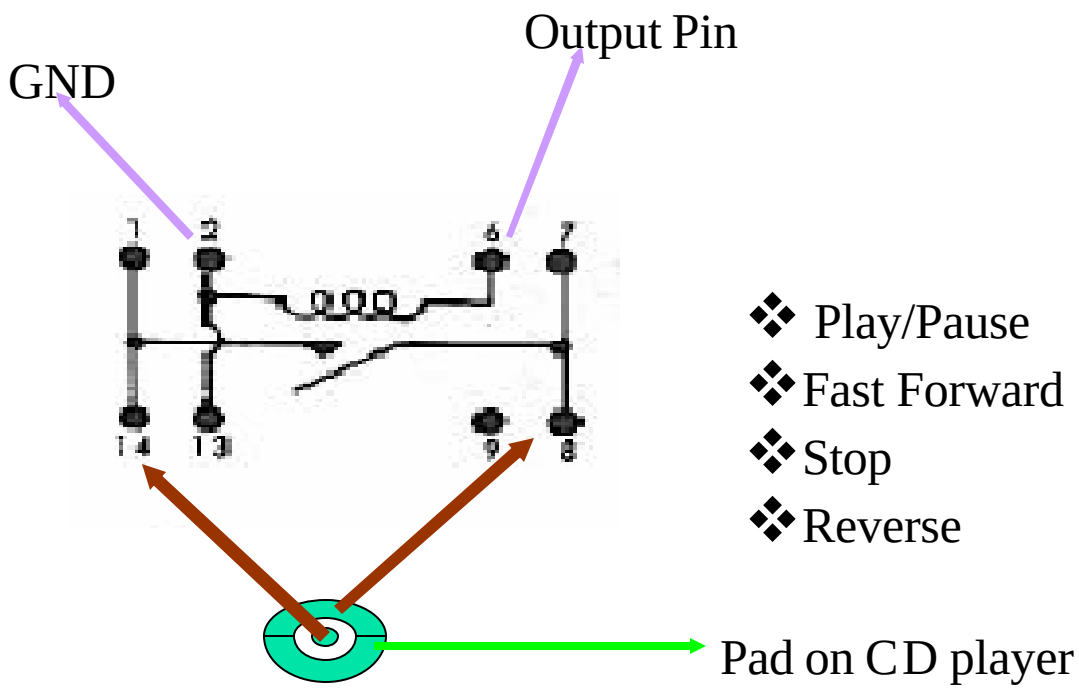

figure 3

Conclusion

In conclusion, this has been a very trying experience. I am tired mentally and physically, but it was all worth it. I learned a lot about robot's, Murray's law, wood, and the benefits of glue and tape. I'll like to thank the Dr. Arroyo, Dr. Schwartz, the TA's, my classmates, and the students from previous semesters.

Appendix

AD Code Courtsey of C. Andrew Davis creator of Satyagraha

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>
typedef unsigned char u08;
u08 AD(u08 channel)
{
      u08 L,H;
      outp(channel,ADMUX);
      sbi(ADCSR,ADSC);
      loop_until_bit_is_set(ADCSR,ADIF);
      L = inp(ADCL);
         H = inp(ADCH);
         sbi(ADCSR,ADIF);
      return H;

}

void wait(int time)
{
          volatile int a,b,c,d;
          for (a=0; a<time; ++a){
         for(b=0;b<10;++b)  {
         for(c=0;c<66;++c)  {
           d=a+1;
           }
           }
      }
      return;
 }

int main(void)
{

 u08 irLeft;

  /*AD setup*/
          outp(0x00,DDRA);
          outp(0xFF,DDRC);
          outp(0xDE,ADCSR);
```

```
 for(;;){    irLeft=AD(0x20);
            outp(~irLeft,PORTC);

            wait(300); }



}
```

Servo Testing Courtsey of C. Andrew Davis creator of Satyagraha for the atmega323

```
#include <io.h>
#include <sig-avr.h>

int main(void)
{
        outp(0xFF,DDRD);
        outp(0xA1,TCCR1A);
        outp(0x04,TCCR1B);
        outp(0x00,TCNT1H);
        outp(0x00,TCNT1L);
        outp(0x00,OCR1AH);

        outp(0x14,OCR1BL);
      outp(0x10,OCR1AL);

        for(;;) {}

}
```

Bump Skirt for atmega323

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>

#define AVR_MEGA            0
#define Front_L       131
#define Front_C             81
#define Front_R             45
#define Back          23
#define Front_L_C     155
#define Front_R_C     104
```

```
#define error                0x9d
#define for_RM                      0x09
#define for_LM                      0x19
#define rev_RM                      0x18
#define rev_LM                      0x09
#define stop1                0x12
#define left_LM                     0x19
#defineright_RM              0x09
#define Bleft_RM             0x19
#define Bright_LM            0x09


typedef unsigned char u08;

u08 AD(u08 channel)
{
      u08 L,H;
      outp(channel,ADMUX);
      sbi(ADCSR,ADSC);
      loop_until_bit_is_set(ADCSR,ADIF);
      L = inp(ADCL);
         H = inp(ADCH);
         //outp(~H,PORTC);
      //wait(100);
         sbi(ADCSR,ADIF);
      return H;

}



void wait(int time)
{
          volatile int a,b,c,d;
          for (a=0; a<time; ++a){
         for(b=0;b<10;++b)  {
         for(c=0;c<66;++c)  {
           d=a+1;
           }
           }
      }
      return;
 }
```

```
int main(void)
{
    u08 bump;

        /*PWM setup*/
        outp(0xFF, DDRD);

        outp(0x00,TCNT1L);          /* timer counter register */
        outp(0x00,TCNT1H);

        sbi(TCCR1A,7);              /*bit5,4: 1,0 - clear: 1,1 - set: 0,1 - toggle */
        cbi(TCCR1A,6);                          /*if 1 timer is reset to $00 */
        sbi(TCCR1A,5);
        cbi(TCCR1A,4);
        cbi(TCCR1A,1);                          /*bit2,1,0:  clock prescaler */
        sbi(TCCR1A,0);

        sbi(TCCR1B,2);


        /*AD setup*/
        cbi(DDRA,0);  //IR sensor
        cbi(DDRA,1);  //IR sensor
        cbi(DDRA,2);   //bump switches
        outp(0xFF,DDRC);
        outp(0xDE,ADCSR);

        for(;;)   {

                        bump = AD(0x22);


                        if(bump >= 129 && bump <= 156){  //contact was made with
Front_L or Front_L_C
                            outp(rev_RM,OCR1AL);
                            outp(rev_LM,OCR1BL);
                            wait(300);
                            outp(for_RM,OCR1AL);
                            outp(stop1,OCR1BL);
                            wait(370);} //70


                    if(bump >= 103 && bump <= 105){   //contact was made with
Front_R_C
                        outp(rev_RM,OCR1AL);
                            outp(rev_LM,OCR1BL);
```

```
                wait(300);
                outp(stop1,OCR1AL);
                outp(for_LM,OCR1BL);
                wait(370);}


            if(bump >= 81 && bump <= 82){     //contact was made with Front_C
              outp(rev_RM,OCR1AL);
                outp(rev_LM,OCR1BL);
                wait(300);}


            if(bump >= 44 && bump <= 46){     //contact was made with Front_R
              outp(rev_RM,OCR1AL);
                outp(rev_LM,OCR1BL);
                wait(300);
                outp(stop1,OCR1AL);
                outp(for_LM,OCR1BL);
                wait(370);}

            if(bump >= 22 && bump <=24 ){    //contact was made with Back
                outp(for_RM,OCR1AL);
                outp(for_LM,OCR1BL);
                wait(300);}



            outp(stop1,OCR1AL);
            outp(stop1,OCR1BL);



        }

}
```

CD Player

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>


/*CONSTANTS*/
#define playmusic      0x01
#define pause          0x02
```

```
#definediscontinue      0x04
#define reverse                 0x08
#define fastforward     0x10
#define trackone        0x20
#define selecttwo       0x40
#define gotonumberthree         0x80
#define choosefour      0x81
#define optionfive      0x82
#define six             0x84
#define trackseven      0x88
#define eight           0x90
#define picknine        0xa0
#define zero            0xc0
#define playbut                 4
#define forwardbut      5
#define reversebut      6
#define stopbut         7



volatile unsigned char track,prevtrack,value=0,value1=0,count1=0;
volatile int flag1=0,flag2=0;


void wait(int time)
{
          volatile int a,b,c,d;
          for (a=0; a<time; ++a){
         for(b=0;b<10;++b)  {
         for(c=0;c<66;++c)  {
           d=a+1;
           }
           }
      }
      return;
 }



SIGNAL(SIG_INTERRUPT1)
{
      /*cbi(GICR,7);*/

      outp(0xaa,PORTC);
      wait(300);
```

```
        count1++;

        if (flag1 == 0)
        {
            value = inp(PINB);

            outp(~value,PORTC);
            wait(300);

            if (value == 0x20 || value == 0x40)
                  flag1 = 1;
            }


        else
            {
              value1 = inp(PINB);
              flag1 = 2;  /*needed to break out of while loop at the end of main*/

            }

         outp(~flag1,PORTC);
        wait(300);
}


void playtrack(int count)
{

            /*if (command == reverse)*/
              cbi(PORTA,reversebut);

            /*if (command == fastforward)*/
              cbi(PORTA,forwardbut);

             if (count == 0)             /*ignore track 0*/
            return;

        sbi(PORTA,stopbut);
            wait(50);
        cbi(PORTA,stopbut);


        wait(705);                  /*CD player needs time to stop*/
```

```
      if (count == 1)           /*to play track 1 fastforward not needed, therefore skip that
step*/
         goto skip;



      while(count > 0)
         {

               sbi(PORTA,forwardbut);
               wait(50);
               cbi(PORTA,forwardbut);
               wait(20);
           count--;


         }



skip:  sbi(PORTA,playbut);
         wait(50);
         cbi(PORTA,playbut);


}


int main(void)

{
         int flag=0; track=0;

         /*CD Player*/
         sbi(DDRA,7);          /*stop*/
         sbi(DDRA,6);          /*backward*/
         sbi(DDRA,5);          /*fastforward*/
         sbi(DDRA,4);          /*play or pause*/

         /*received data from RF*/
         outp(0x00,DDRB);
         outp(0xff,DDRC);

         /*external interrupt,int1, rising edge*/
         cbi(DDRD,3);
         sbi(GICR,7);
```

```
        sbi(MCUCR,3);
        sbi(MCUCR,2);

        sei();

        for(;;){

start:          while(count1 == 0)
                    {
                     outp(0xcc,PORTC);

                    }


                  switch(value1) {
                case 0x20:
                    value1 = 1;
                    break;

                case 0x40:
                    value1 = 2;
                    break;

                   case 0x80:
                    value1 = 3;
                    break;

                case 0x81:
                    value1 = 4;
                    break;

                 case 0x82:
                    value1 = 5;
                    break;

                case 0x84:
                    value1 = 6;
                    break;

                   case 0x88:
                    value1 = 7;
                    break;

                case 0x90:
                    value1 = 8;
                    break;
```

```
case 0xa0:
    value1 = 9;
    break;

case 0xc0:
    value1 = 0;
    break;

default:
    break;


  }



 if(value == playmusic)
         {
           /*outp(~value,PORTC);
           wait(300);*/
           sbi(PORTA,playbut);
           wait(50);
           cbi(PORTA,playbut);
           count1 = 0;


          }



 else if(value == pause)
         {
           /*outp(~value,PORTC);
           wait(100);*/
           sbi(PORTA,playbut);
           wait(50);
           cbi(PORTA,playbut);
           count1 = 0;


          }
```

```
else if(value == fastforward && flag == 0)
            {
              /*outp(~value,PORTC);
              wait(100);*/
              sbi(PORTA,forwardbut);
              flag = 1;
              count1 = 0;

              /*outp(~flag,PORTC);
              wait(100);*/

             }



else if(value == discontinue && flag == 1)
            {

              cbi(PORTA, forwardbut);
              flag = 0;
              count1 = 0;
              /*outp(0x0f,PORTC);
              wait(100);*/

             }


   else if(value == reverse && flag == 0)
            {
              /*outp(~value,PORTC);
              wait(100);*/
              sbi(PORTA,reversebut);
              flag = 2;


             }

 else if(value == discontinue && flag == 2)
            {

              cbi(PORTA,reversebut);
              flag = 0;
              count1 = 0;
              /*outp(0xf0,PORTC);
              wait(100);*/
```

```
        }


else if(value == discontinue)
        {
          /*outp(~value,PORTC);
          wait(100);*/
          sbi(PORTA,stopbut);
          wait(50);
          cbi(PORTA,stopbut);
       count1 = 0;


        }


 else if(value == trackone && flag1 == 0)
                {
                  track = 1;
                  /*outp(~track,PORTC);
                  wait(300);*/
                playtrack(track);
                count1 = 0;
                 }


 else if(value == trackone && flag1 == 2)
                {
                track = 10+value1;
                /*(~track,PORTC);
                wait(100);*/
                playtrack(track);
                count1 = 0;
                flag1 = 0;


              }


else if(value == selecttwo && flag1 == 0)
                {
                  track = 2;
                  /*outp(~track,PORTC);
                  wait(300);*/
                playtrack(track);
```

```
                count1 = 0;


                 }



else if(value == selecttwo && flag1 == 2)
                 {
                   track = 20+value1;
                   /*outp(~track,PORTC);
                   wait(300);*/
                playtrack(track);
                count1 = 0;
                flag1 = 0;



                 }


else if(value == gotonumberthree)
              {
                   track = 3;
                   outp(~track,PORTC);
                   wait(300);
                playtrack(track);
                count1 = 0;

                 }



else if(value == choosefour)
              {
                   track = 4;
                   /*outp(~track,PORTC);
                   wait(300);*/
                playtrack(track);
                count1 = 0;

                 }

 else if(value == optionfive)
                 {
                   track = 5;
```

```
                    /*outp(~track,PORTC);
                    wait(300);*/
                  playtrack(track);
                  count1 = 0;

                   }

else if(value == six)
                {
                    track = 6;
                   /* outp(~track,PORTC);
                    wait(300);*/
                  playtrack(track);
                  count1 = 0;

                   }

else if(value == trackseven)
                {
                    track = 7;
                    /*outp(~track,PORTC);
                    wait(300);*/
                  playtrack(track);
                  count1 = 0;

                   }

 else if(value == eight)
                   {
                     track = 8;
                    /* outp(~track,PORTC);
                     wait(300);*/
                  playtrack(track);
                  count1 = 0;

                   }

else if(value == picknine)
                {
                    track = 9;
                    /*outp(~track,PORTC);
                    wait(300);*/
                  playtrack(track);
                  count1 = 0;

                   }
```

```
        else if(value == 0)
            count1 = 0;




            /*sbi(GICR,7);*/

            /*wait 2sec for anther command*/



           if(flag1 == 1)
             {
               outp(0xf0,PORTC);
               wait(5000);

             }


         if (flag1 == 1)
              {
                flag1 = 0;

              }


         outp(~count1,PORTC);
          wait(300);

          /*outp(0x03,PORTC)
          wait(300);
          outp(~flag,PORTC);
          wait(200);*/

           }

}
```

dance

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>


void wait(int time)
{
          volatile int a,b,c,d;
          for (a=0; a<time; ++a){
         for(b=0;b<10;++b)  {
         for(c=0;c<66;++c)  {
           d=a+1;
           }
           }
      }
      return;
 }


void right_fast(int time)
{

          outp(0x12,OCR1AL);
          outp(0x19,OCR1BL);
          wait(time);
}



void right_slow(int time)
{
          outp(0x12,OCR1AL);
          outp(0x15,OCR1BL);
          wait(time);
}


void backward_right_fast(int time)
{
          outp(0x12,OCR1AL);
          outp(0x09,OCR1BL);
```

```
        wait(time);

        }


void backward_right_slow(int time)
{
        outp(0x12,OCR1AL);
        outp(0x10,OCR1BL);
        wait(time);

        }


void left_fast(int time)
{

        outp(0x08,OCR1AL);
        outp(0x12,OCR1BL);
        wait(time);
}



void left_slow(int time)
{
        outp(0x10,OCR1AL);
        outp(0x12,OCR1BL);
        wait(time);
}



void backward_left_slow(int time)
{
        outp(0x14,OCR1AL);
        outp(0x12,OCR1BL);
        wait(time);

        }


void backward_left_fast(int time)
{
        outp(0x19,OCR1AL);
        outp(0x12,OCR1BL);
```

```
        wait(time);

        }


void forward_fast(int time)
{
        outp(0x09,OCR1AL);
        outp(0x19,OCR1BL);
        wait(time);
        }

void forward_slow(int time)
{
        outp(0x10,OCR1AL);
        outp(0x14,OCR1BL);
        wait(time);
        }


void reverse_fast(int time)
{
        outp(0x19,OCR1AL);
        outp(0x08,OCR1BL);
        wait(time);

        }


void reverse_slow(int time)
{
        outp(0x14,OCR1AL);
        outp(0x10,OCR1BL);
        wait(time);

        }

void stop(int time)
{
        outp(0x12,OCR1AL);
        outp(0x12,OCR1BL);
        wait(time);
        }

void taja(void)
{
```

```
        forward_fast(500);
        reverse_fast(500);
        backward_left_fast(550);//700
        reverse_fast(1000); //T

        backward_right_fast(900);//1100
        reverse_fast(1000);
        backward_left_fast(700);
        reverse_fast(900);
        forward_fast(400);
        left_fast(400);
        forward_fast(500);
        reverse_fast(500);
        right_fast(500);
        reverse_fast(600); //A

        backward_right_fast(800);
        reverse_fast(800);
        backward_left_fast(900);//1200-1300
        reverse_fast(1100);
        backward_left_fast(900);
        reverse_fast(1300);  //j

        backward_left_fast(700);
        reverse_fast(900);
        forward_fast(400);
        left_fast(400);
        forward_fast(500);
        reverse_fast(500);
        right_fast(500);
        reverse_fast(600); //A


        }


void triangle(int times)
{
    while(times > 0)
    {
        forward_fast(1000);
      right_fast(800);
        forward_fast(1000);
        right_fast(600);
        forward_fast(1000);
        right_fast(700);
```

```
        forward_fast(200);
        times--;

        }
}


void figure_8(int times)
{
   while(times > 0)
   {
        forward_fast(300);
        left_fast(600);
        forward_fast(700);
        right_fast(1500);
        forward_fast(700);
        right_fast(500);
        forward_fast(1000);
        left_fast(600);
        //forward_fast(100);
        times--;

        }


}


int main(void)
{

         /*PWM setup*/
        outp(0xFF, DDRD);
      outp(0xff,DDRC);
        outp(0x00,TCNT1L);          /* timer counter register */
        outp(0x00,TCNT1H);

        sbi(TCCR1A,7);              /*bit5,4: 1,0 - clear: 1,1 - set: 0,1 - toggle */
        cbi(TCCR1A,6);                          /*if 1 timer is reset to $00 */
        sbi(TCCR1A,5);
        cbi(TCCR1A,4);
        cbi(TCCR1A,1);                          /*bit2,1,0:  clock prescaler */
        sbi(TCCR1A,0);

        sbi(TCCR1B,2);
```

```
      left_fast(900);
      wait(50);
          //outp(0x0f,PORTC);
          //right_circle_fast(2300);
          /*wait(50);
          outp(0x03,PORTC);
          right_circle_med();
          wait(50);
          outp(0xaa,PORTC);
          right_circle_slow();
          wait(50);*/
          outp(0x12,OCR1AL);
          outp(0x12,OCR1BL);

}
```

voice

```
#include <io.h>
#include <interrupt.h>
#include <sig-avr.h>


#define mode0 PA3
#define start PA2
#define reset PA1
#define play  PA0


void wait(int time)
{
          volatile int a,b,c,d;
          for (a=0; a<time; ++a){
         for(b=0;b<10;++b)  {
         for(c=0;c<66;++c)  {
           d=a+1;
           }
           }
      }
      return;
 }


void say(int count)
{
```

```
 //reset
sbi(PORTA,reset);
wait(50);
cbi(PORTA,reset);


switch(value1) {
                case 'a':
                    count = 2;
                    break;

                case 'b':
                    count = 3;
                    break;

                   case 'c':
                     count = 4;
                    break;

                case 'd':
                     count = 5;
                    break;

                 case 'e':
                     count = 6;
                    break;

                case 1:
                     count = 7;
                    break;

                   case 2:
                     count = 8;
                    break;

                case 3:
                     count = 9;
                    break;

                case 4:
                     count = 10;
                    break;

                   case 5:
                     count = 11;
                    break;
```

```
case 6:
      count = 12;
     break;

case 7 :
      count = 13;
     break;

    case 8:
      count = 14;
     break;

case 9:
      count = 15;
     break;

case 10:
      count = 16;
     break;

    case 11:
      count = 17;
     break;

case 12:
      count = 18;
     break;

 case 13:
      count = 19;
     break;

case 14:
      count = 20;
     break;

case 15:
      count = 21;
     break;

    case 16:
      count = 22;
     break;

case 17:
```

```
      count = 23;
     break;

case 18:
      count = 24;
     break;

    case 19:
      count = 25;
     break;

case 20:
      count = 26;
     break;

case 21:
      count = 27;
     break;

    case 22:
      count = 28;
     break;

case 23:
      count = 29;
     break;

case 24 :
      count = 30;
     break;

    case 25:
      count = 31;
     break;

case 26:
      count = 32;
     break;

case 27:
      count = 33;
     break;

    case 28:
      count = 34;
     break;
```

```
                case 29:
                     count = 35;
                    break;

                default:
                    break;


                  }

   //mode0 = 1
   sbi(PORTA,mode0);

    //skip though messages, pulse !CE low
    while(count > 1)
    {
      cbi(PORTA,start);
      wait(50);
      sbi(PORTA,start);
      wait(50);
      count--;
    }

   //mode0 = 0
   cbi(PORTA,mode0);

   //play
   cbi(PORTA,start);
   wait(50);
   sbi(PORTA,start);

 }


int main(void)
{
   sbi(DDRA,0);
   sbi(DDRA,1);
   sbi(DDRA,2);
   sbi(DDRA,3);
   outp(0xff,PORTC);

   //init
   cbi(PORTA,mode0);
   sbi(PORTA,start);
```

```
   cbi(PORTA,reset);
   sbi(PORTA,play);

   wait(50);

   //reset
   sbi(PORTA,reset);
   wait(50);
   cbi(PORTA,reset);

   wait(50);

   //play intro
   outp(0x0f,PORTC);
   cbi(PORTA,start);
   wait(50);
   sbi(PORTA,start);


   wait(5000);

   outp(0xaa,PORTC);
   say(0x02); //say pause

   wait(200);

   outp(0xfc,PORTC);
   say(0x01);  //play



 }
```

servo using pic

```
' Robotics! v1.4, Program Listing 1.3: Servo centering program again.


leftrev  CON 180
leftfor  CON 110
rightrev        CON 110
rightfor        CON 180
stop1           CON   150
turn            CON 3000
gorev           CON 4000
```

```
move            VAR BYTE
counter         VAR BYTE
flag            VAR BYTE


TRISB = $00
TRISA = %11111111

PORTB.5 = 0
PORTB.4 = 0

'counter = 4000

'here:
'                       PulsOut PORTB.5, leftfor        ' Send 1.5 ms pulses to P12
 '                  PulsOut PORTB.4, rightfor
  '                 Pause 20
   '                counter = counter - 1
    '               IF counter > 0 Then here


start:

IF PORTA.0  Then GoSub backward
IF PORTA.1  Then GoSub right
IF PORTA.2  Then GoSub left
IF PORTA.3  Then GoSub forward

GoTo start


backward:


                        counter = gorev

backloop1:
                        PulsOut PORTB.5, leftrev        ' Send 1.5 ms pulses to P12
                  PulsOut PORTB.4, rightrev
                  Pause 20
             counter = counter - 1
                  IF counter > 0 Then backloop1

                  counter = turn
```

```
                PORTB.5 = 0
                        PORTB.4 = 0
backloop2:      PulsOut PORTB.5, leftfor        ' Send 1.5 ms pulses to P12
                PulsOut PORTB.4, stop1
                Pause 20
                counter = counter - 1
                IF counter > 0 Then backloop2
                Return

'right:
'               counter = turn
 '
'rightloop1:
'                       PulsOut PORTB.5, stop1        ' Send 1.5 ms pulses to P12
 '              PulsOut PORTB.4, rightfor
  '             Pause 20
   '            counter = counter - 1
    '           IF counter > 0 Then rightloop1
    '       GoTo start
     '



'right:


'                       counter = gorev

'loop1:


'                       PulsOut PORTB.5, leftrev        ' Send 1.5 ms pulses to P12
 '              PulsOut PORTB.4, rightrev
  '             Pause 20
   '            counter = counter - 1
    '           IF counter > 0 Then loop1
    '
 '              counter = turn
    '           'PORTB.5 = 0
         '              'PORTB.4 = 0
'loop2:          PulsOut PORTB.5, leftfor        ' Send 1.5 ms pulses to P12
    '           PulsOut PORTB.4, stop1
    '           Pause 20
    '           counter = counter - 1
    '           IF counter > 0 Then loop2
```