Tracking, Mating Wheeled Mobile Robots
(Abbott and Costello)

Chad Sylvester
Intelligent Machines Design Laboratory
Department of Electrical and Computer Engineering
August 7, 2003

Table of Contents

Abstract

Much effort has been placed on the development of multiple autonomous wheeled mobile robots for numerous applications including exploration, formation and following, transportation, path planning, mapping, and cleaning. In addition to this list, these robots are useful in environments and tasks that can be covered more efficiently or effectively with multi-agents.

Computing an optimal path for these robots can be cumbersome and often can require heavy computing power which slows them down and consumes a great deal of power. Wheeled mobile robots often have the advantage over other types of robots in that they can carry more weight compared to their size. This would allow them to support an onboard computer; however this paper will assume the robots only have the performance of a moderate microprocessor. A method for controlling two autonomous wheeled mobile robots with position feedback in a rendezvous maneuver will be discussed which can be easily adapted to vehicle formation, follow the leader operations, and multiple robot carrying tasks.

This paper further presents the two robots on which multi-agent tasks are accomplished. A detailed description of the hardware, software, and communication will be discussed.

Executive Summary

This paper will discuss the hardware, software, and communication aspects involved with the development of the docking and talking. The mobile platform, actuation, sensors, and the power source will be discussed followed by behavior, theory, experiments, and results.

Introduction

The robots presented in this paper are for the purpose of completion of the IMDL course and the start of a multi-agent project for autonomous Ackerman steering vehicles. The theory behind the calculations made for this project are briefly discussed and the results are shown.

The general goal for these robots is that while one remain still the other is able to maneuver around the workspace and dock with the first one. After docking is achieved the robots perform an Abbott and Costello routine called “Who’s on First?”. This is done to show the capabilities of docking and communicating.

Integrated System

There are many examples of multi-agent autonomous robots. Among those are the wheeled mobile robots. This project consists of two wheeled mobile robots for tracking and mating of which there is one inspecting robot and one that needs to be maintained. The visionary plan for this system can be applied to a robot that requires regular

maintenance checks. The service robot finds and pulls up to the other robot and makes a link through a hard-line serial port connection. The examining robot then uploads or downloads necessary information and then leaves or goes to inspect another robot.

After a brief look at the make up of the similarities, there are two major discussions involved in the detailed description of these two robots. The first will solely incorporate the robot to be analyzed and probed. Following this outline will be a comprehensive framework of the seeking robot.

Mobile Platform

The mobile platform is constructed mostly from 1/8" aluminum. The box that holds the LetATwork II development board and most of the electronics is constructed from lexan, a nonconductive material. The top of the vehicle is also made from lexan and holds the LEDs for position feedback and IR sensors for obstacle avoidance. The steering mechanism was taken from a Motor Works die-cast replica of baja racer.

Actuation

There are two motors for each robot. One motor is used to controlling the velocity of the robot. The differential gear box is made from a set of spur gears. This gear box is actuated with a DC motor from Pololu which can be seen in figure 1.

Figure 1. Pololu DC motor and Gearbox (item number 70093)

Because Ackerman steering is being used, a second motor is used to for directional control with an 81-MG Hitec servo motor from Servo City as seen in figure 2. This servo motor was chosen because of its small size and rather high torque.

Figure 2. Hitec servo motor (HS 81MG) used for steering

The wheels are also taken from the die-cast model and are JB welded to nuts and screwed to the axis.

Sensors

Phasespace Camera System

The main sensing unit in the system is the position feedback from the camera system developed by Phasespace.


Figure 3. Front view of the four cameras from Phasespace

Figure 4. The camera system setup in the SAMM lab

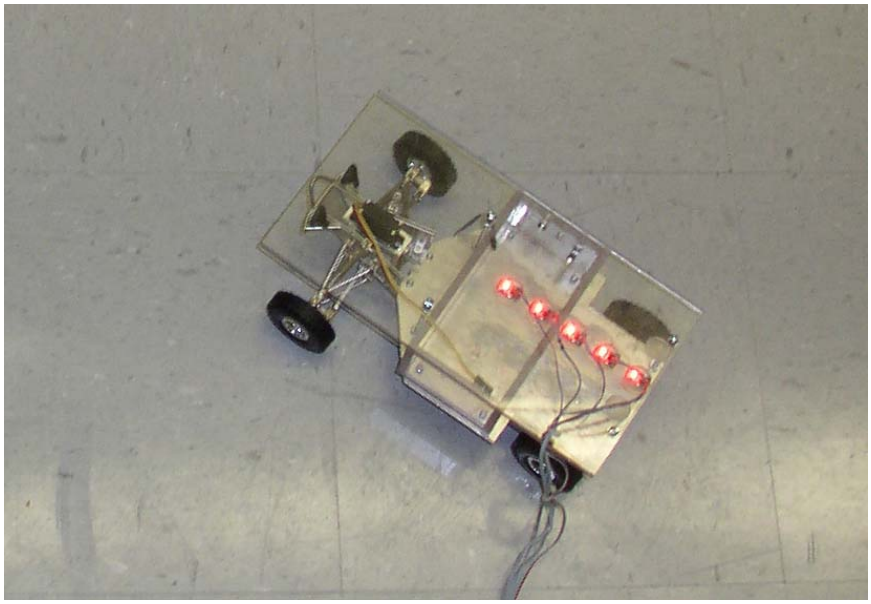


Figure 5. LEDs mounted on vehicle

GP2D12 IR Sensors

In addition this elaborate camera system, there are also six IR sensors incorporated in the body of the vehicle. This allows the robots to "see" what obstacles are around them. Two sensors are placed on the front. Two sensors are placed on the rear of the vehicle. One IR sensor is placed on either side of the robot. The two in the front and the two in the rear are used to be sure the robot does not run over or run into any objects. The ones on each side are used to check for the capabilities for the robot to turn. If an object goes

undetected on the sides the robot could clip the object. The IR sensors were calibrated and incorporated into the system.

Power

The power for the robots is provided by a 7.2 volts battery pack of NiMH. The power for the serial data transfer is a 9V supply from the wall.

Specified Differences

As discussed before, there are a few key differences in the manufacturing of these robots. One robot functions to find and mate with the second robot. Therefore, the first robot extends the connection while the second has incorporated a task of receiving the probe.

Stationary Robot to be Probed

The inspected robot is the member of multi-agent team that needs to be frequently examined. A port was designed for the serial connection with another robot. More importantly is the reception mechanism for the link.

For the simplest demonstration this robot does not need any actuators for driving or steering only for the reception device. In the most basic example there is a limit switch that lets this robot know that the first robot has docked.

Inspecting Robot

The assessing robot is given the task of trajectory planning through the feedback information from the Phasespace vision system. Since the feedback information is given in x, y, z coordinates some manipulation needs to be made to the data. By allowing the vehicle to react to the error signal, it pulls up to the rear of the stationary robot and slows to a stop. When distance between the robots is correct, it extends a mechanism used for the docking.

Behaviors

There are several behaviors that the robots go through to perform the docking maneuvers. The first of these is the reception of a signal from the PC to start the docking. If the command is not receive from the computer the robots will wait. After the docking process is initiated, the robots' position and orientation are continually updated and transmitted to the robot via a serial communication.

Command Reception

The information is sent from PC where the robots' position and orientation are modified to each robot's microprocessor through the wireless serial communication as discussed in the previous sections.

LCD display of communication

The LCD displays several pieces of information to inform the user what the robot is "thinking". During the docking maneuver, the displays will say "Docking" and then continue to display the position and orientation components. This allows the user to make sure the camera system is continually updating. If the robots were to move out of the coverage zone of the positioning system, the information would not change until the robot moved back into the proper workspace.

Obstacle Avoidance

The obstacle avoidance is performed based on the sensor feedback of the GP2D12 IR sensors. With the front IR sensors looking crosseyed, the robots can see directly in front and to the sides within about 2 feet of the robot. If an obstacle is detected measurements are taken to avoid it.

Docking Maneuvers

To get position data for feedback information the cameras detect a world frame x,y,z position of 10 leds. Five leds are on each robot.

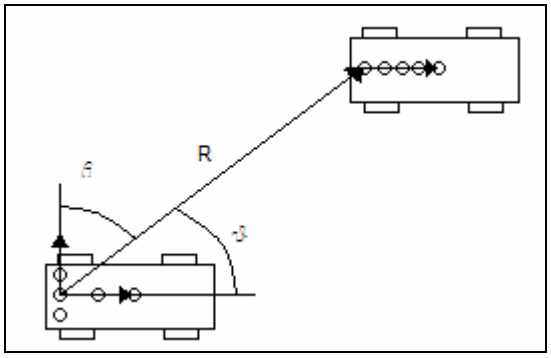


Figure 6. Pictoral demonstration of position calculations

The relative angles and distances are used for feedback. The vector R is found by:

$$R = (x_1 - x_2)i + (y_1 - y_2)j + (z_1 - z_2)k \qquad \text{Eq(1)}$$

where:

x1, y1, z1 are associated with an led on the moving robot
x2, y2, z2 are associated with an led on the stationary robot

The angle at which the stationary robot is to the moving robot is found by taking the dot product between both vectors on the stationary robot and R.

$$\theta = \cos^{-1}\left(\frac{(R \cdot b_2)}{\|R \cdot b_2\|}\right) \qquad \text{Eq(2)}$$

$$\beta = \cos^{-1}\left(\frac{(R \cdot c_2)}{\|R \cdot c_2\|}\right) \quad \text{Eq(3)}$$

where:

c_2 defines a vector orthogonal to b_2

b_2 defines the heading vector of the stationary robot

Because the dot product only gives a value in the 1st and 2nd quadrants, the following logical statement is used to find the accurate angle.

If (β>90) Then $\theta = -\theta$ Eq(4)
Else $\theta = \theta$

Now that the position is fully determined, the heading of the moving robot with respect to the stationary one can be determined similarly by:

$$\psi = \cos^{-1}\left(\frac{(b_1 \cdot b_2)}{\|b_1 \cdot b_2\|}\right) \quad \text{Eq(5)}$$

$$\phi = \cos^{-1}\left(\frac{(b_1 \cdot c_2)}{\|b_1 \cdot c_2\|}\right) \quad \text{Eq(6)}$$

where:

b_1 defines the heading vector of the moving robot

Because the dot product only gives a value in the 1st and 2nd quadrants, the following logical statement is used to find the accurate angle.

If (ϕ>90) Then $\psi = -\psi$ Eq(7)
Else $\psi = \psi$

In order to have the robot be in line with the stationary robot for docking a weight must be placed on the direction perpendicular to the heading on the error signal of theta. This is done with the following equation.

$$\theta_1 = \tan^{-1}\left(\frac{(k_\theta \sin(\theta))}{\cos(\theta)}\right) \quad \text{Eq(8)}$$

where:

θ_1 is the weighted position angle

k_θ is the weight on the position error

A weight must also be included in the ψ error signal. This is done with the following equation.

$$\psi_1 = k_\psi \left(\psi_{desired} - \psi\right) \quad \text{Eq(9)}$$

where:

ψ_1 is the weighted heading angle
$\psi_{desired}$ is the desired heading (180^o for this project)
k_ψ is the weight on the heading error

This leads to setting the steering angle. The equation 10 gives this angle.

$$\gamma = \gamma_0 + k_\gamma \left(\theta_1 - \psi_1\right) \quad \text{Eq(10)}$$

where:

γ is the steering angle with the center at γ_0
k_γ is the weight on the steering angle

The speed is based on the both the distance error, R, and the update rate of the IR sensors and the position feedback. The speed is limited to be slow enough that the robot will not run into obstacles in between readings of the sensors. The following equation gives this shows speed as a function of the distance error.

$$speed = k_R \left(R_{desired} - \|R\|\right) \quad \text{Eq(11)}$$

where:

$R_{desired}$ is the desired distance between the robots
k_R is the weight on the distance error
*note: This linear value for speed is only good for a small region. When the robots get close there must be a minimum signal to make the robot move

Now a check needs to be performed for plausibility of the movement. This does not yet include obstacle avoidance. If the goal position is within a circle with radius equal to the turning radius of the robot on either side of the robot, a maneuver to back away and try docking again must be performed.

$$\text{IF}\left(\rho < \sqrt{R^2 + \rho^2 + 2R\rho \cdot \cos\left(180 - \theta\right)}\right) \quad \text{Eq(12)}$$

THEN (Reverse and update speed with γ)

A check can now be done for obstacles. If the IR sensors see an obstacle outside of a predetermined radius a short algorithm is performed to maneuver away from the obstacle.

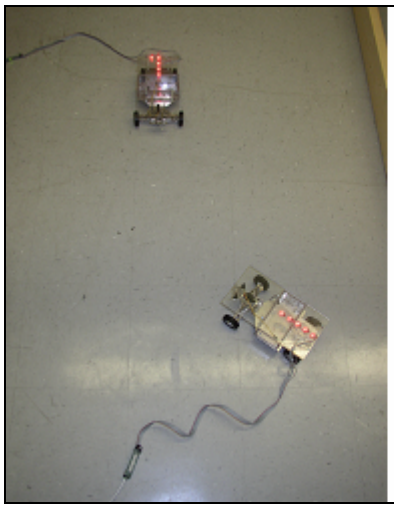
If all these checks and updates went without any errors the motors can be updated and the loop can start over and get new data. If there were errors, the motors are not updated with the weighted information because too much time has passed since the data was taken.

Docking Communication

When the robots are docked the motors have turned off, data is transmitted between the robots. For a good demonstration, the robots carry out an Abbott and Costello routine called "Who's on First?".

Experimental Layout and Results

The first test was to determine whether or not the positioning system was feeding back relatively accurate data. Figure 7 illustrates the results of this test. A tape measure was used to measure the distance between the robots and then compared with the data given by the positioning system. All of these units are in millimeters. The tape was only accurate to about a centimeter. As one can tell, the results are very accurate. This test was successful.



Sensor	Measured
206	210
478	480
772	770
1022	1020

Figure 7. Distance test for the positioning system

The second test was to check the quadrants to determine relative position between the robots. This second test proved successful as shown in figure 8. In quadrant one, X and Y are both positive. On the right side of the Y-axis the Xs are positive and negative on the other side. Above the X-axis the Ys are positive and negative below.

X = -350.5 Y = 280.3	X = 392.1 Y = 275.1
X = -331.8 Y = -271.6	X = 368.7 Y = -376.3

Figure 8. Relative position test for the positioning system

The third test was to check the relation between the robots. This task was difficult but proved very successful. The follower robot was positioned facing above and below the leader robot as seen in figure 9. The left picture of the follower robot facing down gives a positive angle. Facing the robot to the other side gives a negative reading.

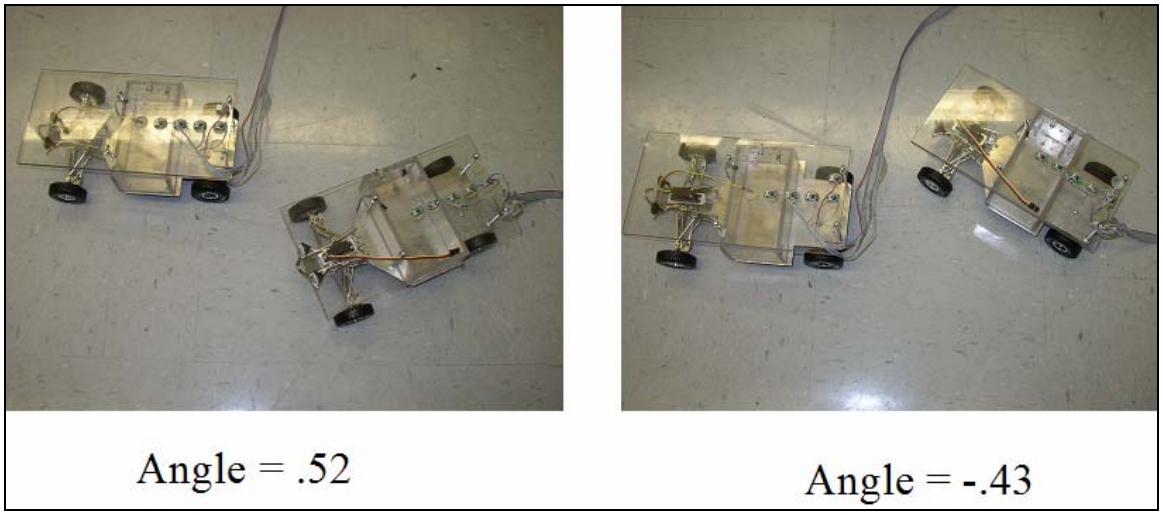


Figure 9. Relative orientation test for the positioning system.

The first experiment implemented was a simple docking procedure. While one robot remained still the other moved while receiving the feedback of its position from the aforementioned camera system. Figure 10 shows a series of this test.

Figure 10. Series of a docking procedure

Another test was done, but this time obstacle avoidance was added into the algorithm. The robot sees a wall, goes in reverse and then approaches the other robot and continues the docking process. This demonstration can be seen in figure 11.

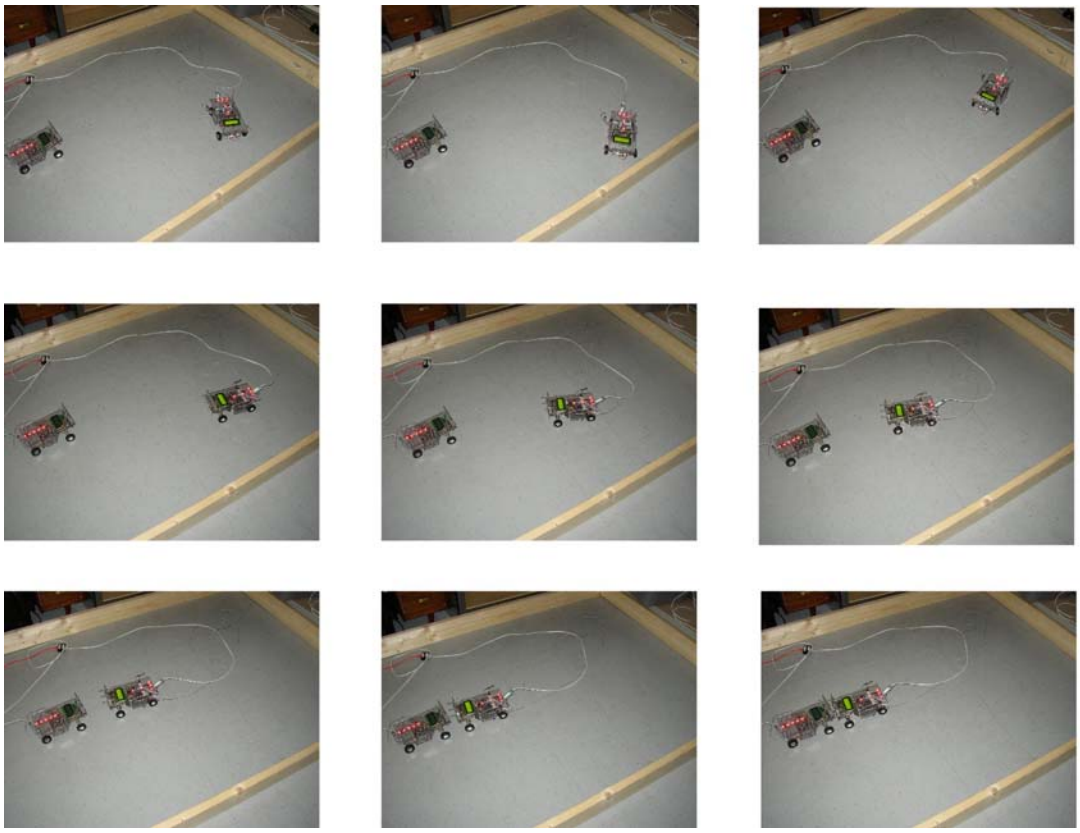


Figure 11. Series of docking with obstacle avoidance

Conclusions

This project attempts to begin an ongoing research project in the path planning and docking of multi-agent robots. The ongoing project will incorporate the numerous advantages of incorporating two robots into tasks. Much work is needed on the communication between the robots. The wireless communication that is mentioned in this paper was replaced by a hard-line serial connection between the PC and the robot because the wireless communication was reliable enough to send data quickly.

Overall, the project was a great success. The robots were able to dock/mate and communicate with each other. The camera system was used as position feedback and was very reliable sensor. Had the communication between the PC and robot been better, the docking process would be smoother.

Documentation

i. Lists and summary of relevant websites
   a. AVR Freaks
   b. C help
   c. Budget Robotics
   d. Phasespace Camera System

ii. Datasheets
   a. IR Sensors (GP2D12)
   b. Atmel mega128
      i. LetATWork II
   c. Hitec Servo HS 81MG
   d. Pololu Motor
   e. TI Motor Driver
   f. Hex Inverter (Not Gate)

Appendices

i. lcd.c (Functions used for LCD display)
ii. lcd.h (Setup for LCD display)
iii. motor.c (Functions used for servos and motors)
iv. motor.h (Setup for servos and motors)
v. global.h (Contains all global settings)
vi. serial.c (Functions used for the serial ports)
vii. serial.h (Setup for the serial ports)
viii. senddata.c (Program to send data to robots from PC)
ix. vector2.c (Program to gather data about robots positions and orientations)
x. Program Code for Robot 1
xi. Routine for Robot 1
xii. Program Code for Robot 2
xiii. Routine for Robot 2

```
/*************************************************************************
*****
 Title   :   HD44780U LCD library
 Author: Chad Sylvester
 Original Author:    Peter Fleury <pfleury@gmx.ch>  http://jump.to/fleury
 Date:      July 28, 2003
 Software:  Programmers Notepad and PonyProg
 Target:    any AVR device, memory mapped mode only for AT90S4414/8515/Mega

 DESCRIPTION
     Basic routines for interfacing a HD44780U-based text lcd display

     Originally based on Volker Oth's lcd library,
     changed lcd_init(), added additional constants for lcd_command(),
     added 4-bit I/O mode, improved and optimized code.

     Library can be operated in memory mapped mode (LCD_IO_MODE=0) or in
     4-bit IO port mode (LCD_IO_MODE=1). 8-bit IO port mode not supported.

     Memory mapped mode compatible with Kanda STK200, but supports also
     generation of R/W signal through A8 address line.

 USAGE
     See the C include lcd.h file for a description of each function

*************************************************************************
*****/
#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include "lcd.h"


/*
** constants/macros
*/
#define PIN(x) (*(&x - 2))  /* address of data direction register of port x */
#define DDR(x) (*(&x - 1))  /* address of input register of port x          */


#if LCD_IO_MODE
#define lcd_e_delay() __asm__ __volatile__( "rjmp 1f\n 1:" );
#define lcd_e_high()    sbi(LCD_E_PORT, LCD_E_PIN)
#define lcd_e_low()     cbi(LCD_E_PORT, LCD_E_PIN)
#define lcd_e_toggle()  toggle_e()
#endif
```

```
#if LCD_IO_MODE
#if LCD_LINES==1
#define LCD_FUNCTION_DEFAULT    LCD_FUNCTION_4BIT_1LINE
#else
#define LCD_FUNCTION_DEFAULT    LCD_FUNCTION_4BIT_2LINES
#endif
#else
#if LCD_LINES==1
#define LCD_FUNCTION_DEFAULT    LCD_FUNCTION_8BIT_1LINE
#else
#define LCD_FUNCTION_DEFAULT    LCD_FUNCTION_8BIT_2LINES
#endif
#endif


/*
** function prototypes
*/
static void delay(uint16_t us);
#if LCD_IO_MODE
static void toggle_e(void);
#endif

/*
** local functions
*/

static void delay(uint16_t us)
/* delay for a minimum of <us> microseconds    */
/* with a 4Mhz crystal, the resolution is 1 us */
{
   while ( us ) us--;
}


#if LCD_IO_MODE
static void toggle_e(void)
/* toggle Enable Pin */
{
   lcd_e_high();
   lcd_e_delay();
   lcd_e_low();
}
#endif
```

```
#if LCD_IO_MODE
static void inline lcd_write(uint8_t data,uint8_t rs)
{
   /* configure data pins as output */
   outp(0xFF, DDR(LCD_DATA_PORT) );

   if (rs) {   /* write data        (RS=1, RW=0) */

      /* output high nibble first */
      outp( ((data>>4)&0x0F)|(1<<LCD_RS_PIN), LCD_DATA_PORT );
      lcd_e_toggle();

      /* output low nibble */
      outp( (data&0x0F)|(1<<LCD_RS_PIN), LCD_DATA_PORT );
      lcd_e_toggle();

   } else {    /* write instruction (RS=0, RW=0) */

      /* output high nibble first */
      outp( (data>>4)&0x0F, LCD_DATA_PORT );
      lcd_e_toggle();

      /* output low nibble */
      outp( data&0x0F, LCD_DATA_PORT );
      lcd_e_toggle();
   }

   /* all data pins high (inactive) */
   outp(0x0F, LCD_DATA_PORT);
}
#else
#define lcd_write(d,rs) if (rs) *(volatile uint8_t*)(LCD_IO_DATA) = d; else *(volatile
uint8_t*)(LCD_IO_FUNCTION) = d;
/* rs==0 -> write instruction to LCD_IO_FUNCTION */
/* rs==1 -> write data to LCD_IO_DATA */
#endif


#if LCD_IO_MODE
static uint8_t lcd_read(uint8_t rs)
{
   register uint8_t dataH, dataL;

   if (rs) sbi(LCD_RS_PORT, LCD_RS_PIN);    /* RS=1: read data      */
   else    cbi(LCD_RS_PORT, LCD_RS_PIN);    /* RS=0: read busy flag */
```

```
    sbi(LCD_RW_PORT, LCD_RW_PIN);           /* RW=1  read mode      */

    /* configure data pins as input */
    outp(0xF0, DDR(LCD_DATA_PORT));

    lcd_e_high();
    lcd_e_delay();
    dataH = inp(PIN(LCD_DATA_PORT));        /* read high nibble first */
    lcd_e_low();

    lcd_e_delay();                          /* Enable 500ns low       */

    lcd_e_high();
    lcd_e_delay();
    dataL = inp(PIN(LCD_DATA_PORT));        /* read low nibble        */
    lcd_e_low();

    return ( (dataH<<4) | (dataL&0x0F) );
}
#else
#define lcd_read(rs) (rs) ? *(volatile uint8_t*)(LCD_IO_DATA+LCD_IO_READ) :
*(volatile uint8_t*)(LCD_IO_FUNCTION+LCD_IO_READ)
/* rs==0 -> read instruction from LCD_IO_FUNCTION */
/* rs==1 -> read data from LCD_IO_DATA */
#endif


static uint8_t lcd_waitbusy(void)
/* loops while lcd is busy, reads address counter */
{
    register uint8_t c;

    /* wait until busy flag is cleared */
    while ( (c=lcd_read(0)) & (1<<LCD_BUSY)) {}

    /* the address counter is updated 4us after the busy flag is cleared */
    delay(2);

    /* now read the address counter */
    return (lcd_read(0));  // return address counter
}


static inline void lcd_newline(uint8_t pos)
/* goto start of next line */
{
```

```
   register uint8_t addressCounter;


#if LCD_LINES==1
   addressCounter = 0;
#endif
#if LCD_LINES==2
   if ( pos < (LCD_START_LINE2) )
      addressCounter = LCD_START_LINE2;
   else
      addressCounter = LCD_START_LINE1;
#endif
#if LCD_LINES==4
   if ( pos < LCD_START_LINE3 )
      addressCounter = LCD_START_LINE2;
   else if ( (pos >= LCD_START_LINE2) && (pos < LCD_START_LINE4) )
      addressCounter = LCD_START_LINE3;
   else if ( (pos >= LCD_START_LINE3) && (pos < LCD_START_LINE2) )
      addressCounter = LCD_START_LINE4;
   else
      addressCounter = LCD_START_LINE1;
#endif
   lcd_command((1<<LCD_DDRAM)+addressCounter);

}/* lcd_newline */


/*
** PUBLIC FUNCTIONS
*/

void lcd_command(uint8_t cmd)
/* send commando <cmd> to LCD */
{
   lcd_waitbusy();
   lcd_write(cmd,0);
}


void lcd_gotoxy(uint8_t x, uint8_t y)
/* goto position (x,y) */
{
#if LCD_LINES==1
   lcd_command((1<<LCD_DDRAM)+LCD_START_LINE1+x);
#endif
#if LCD_LINES==2
```

```
    if ( y==0 )
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE1+x);
    else
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE2+x);
#endif
#if LCD_LINES==4
    if ( y==0 )
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE1+x);
    else if ( y==1)
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE2+x);
    else if ( y==2)
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE3+x);
    else /* y==3 */
        lcd_command((1<<LCD_DDRAM)+LCD_START_LINE4+x);
#endif

}/* lcd_gotoxy */


int lcd_getxy(void)
{
    return lcd_waitbusy();
}


void lcd_clrscr(void)
/* clear lcd and set cursor to home position */
{
    lcd_command(1<<LCD_CLR);
}


void lcd_home(void)
/* set cursor to home position */
{
    lcd_command(1<<LCD_HOME);
}


void lcd_putc(char c)
/* print character at current cursor position */
{
    register uint8_t pos;

    pos = lcd_waitbusy();   // read busy-flag and address counter
    if (c=='\n')
```

```
        lcd_newline(pos);
    else
        lcd_write(c, 1);
}


void lcd_puts(const char *s)
/* print string on lcd (no auto linefeed) */
{
    register char c;

    while ( (c = *s++) ) {
        lcd_putc(c);
    }
}


void lcd_puts_p(const char *progmem_s)
/* print string from program memory on lcd (no auto linefeed) */
{
    register char c;

    while ( (c = PRG_RDB(progmem_s++)) ) {
        lcd_putc(c);
    }
}

void lcd_putn(double data)
// print double on lcd (no auto linefeed)
{
//char * transferString = " ";
int in=0;

if (data < 0.00){
in = 1;
lcd_putc('-');
data=-1*data;}

int index = 0;
double datad = data;
int datai = data;

if (data>=1.0){
index = 1;}
if (data>=10.0){
index = 10;}
```

```
if (data>=100.0){
index = 100;}
if (data>=1000.0){
index = 1000;}
if (data>=10000.0){
index = 10000;}

int n=0;
int temp = 0;
uint8_t tempI =0;
for(n=1;n<=index;n=n*10){
//if (n==1){lcd_puts("n=1");}
//if (n==10){lcd_puts("n=10");}
//if (n==100){lcd_puts("n=100");}
//if (n==1000){lcd_puts("n=1000");}
//if (n==10000){lcd_puts("n=10000");}

tempI = data*n/index-temp;
temp = (temp+tempI)*10;
char tempC = tempI + 48;
lcd_putc(tempC);
//in = in+1;
//transferString[in] = tempC;
//lcd_putc(transferString[in]);
}

int datadec = (datad-datai)*100;
if (datadec > 0){
lcd_putc('.');
//in = in+1;
//transferString[in] = '.';
//lcd_putc(transferString[in]);
tempI = datadec / 10;
char tempC = tempI + 0x30;
lcd_putc(tempC);
//in = in+1;
//transferString[in] = tempC;
//lcd_putc(transferString[in]);

tempI = datadec-(datadec / 10)*10;
tempC = tempI + 0x30;
lcd_putc(tempC);
//in = in+1;
//transferString[in] = tempC;
//lcd_putc(transferString[in]);
}
```

```
//return transferString;

//const char *s = transferString;
//register char c;

  //  while ( (c = *s++) ) {
    //    lcd_putc(c);
    //}
}

void lcd_init(uint8_t dispAttr)
/* initialize display and select type of cursor */
/* dispAttr: LCD_DISP_OFF, LCD_DISP_ON, LCD_DISP_ON_CURSOR,
LCD_DISP_CURSOR_BLINK */
{
#if LCD_IO_MODE
    /*------ Initialize lcd to 4 bit i/o mode -------*/

    outp( 0xFF, DDR(LCD_DATA_PORT) );        /* all port bits as output */

    delay(16000);       /* wait 16ms or more after power-on       */

    /* initial write to lcd is 8bit */
    outp(LCD_FUNCTION_8BIT_1LINE>>4,LCD_DATA_PORT);
    lcd_e_toggle();
    delay(4992);        /* delay, busy flag can't be checked here */

    outp(LCD_FUNCTION_8BIT_1LINE>>4,LCD_DATA_PORT);
    lcd_e_toggle();
    delay(64);          /* delay, busy flag can't be checked here */

    outp(LCD_FUNCTION_8BIT_1LINE>>4,LCD_DATA_PORT);
    lcd_e_toggle();
    delay(64);          /* delay, busy flag can't be checked here */

    outp(LCD_FUNCTION_4BIT_1LINE>>4,LCD_DATA_PORT); /* set IO mode to
4bit */
    lcd_e_toggle();

    /* from now the lcd only accepts 4 bit I/O, we can use lcd_command() */
#else
    /*----- Initialize lcd to 8 bit memory mapped mode ------*/

    /* enable external SRAM (memory mapped lcd) and one wait state */
    outp((1<<SRE)|(1<<SRW), MCUCR);
```

```
    /* reset lcd */
    delay(16000);                       /* wait 16ms after power-on     */
    lcd_write(LCD_FUNCTION_8BIT_1LINE,0);   /* function set: 8bit interface */
    delay(4992);                        /* wait 5ms                     */
    lcd_write(LCD_FUNCTION_8BIT_1LINE,0);   /* function set: 8bit interface */
    delay(64);                          /* wait 64us                    */
    lcd_write(LCD_FUNCTION_8BIT_1LINE,0);   /* function set: 8bit interface */
    delay(64);                          /* wait 64us                    */
#endif
    lcd_command(LCD_FUNCTION_DEFAULT);      /* function set: display lines  */
    lcd_command(LCD_DISP_OFF);              /* display off                  */
    lcd_clrscr();                       /* display clear                */
    lcd_command(LCD_MODE_DEFAULT);          /* set entry mode               */
    lcd_command(dispAttr);                  /* display/cursor control       */

}/* lcd_init */
```

```
/************************************************************************
**
 Title   :   C include file for the HD44780U LCD library (lcd.c)
  Author: Chad Sylvester
 Original Author:    Peter Fleury <pfleury@gmx.ch>  http://jump.to/fleury
 Date:      July 28, 2003
 Software:  Programmers Notepad and PonyProg
 Target:    any AVR device, memory mapped mode only for AT90S4414/8515/Mega

 DESCRIPTION
     Basic routines for interfacing a HD44780U-based text lcd display

     Originally based on Volker Oth's lcd library,
     changed lcd_init(), added additional constants for lcd_command(),
     added 4-bit I/O mode, improved and optimized code.

     Library can be operated in memory mapped mode (LCD_IO_MODE=0) or in
     4-bit IO port mode (LCD_IO_MODE=1). 8-bit IO port mode not supported.

     Memory mapped mode compatible with Kanda STK200, but supports also
     generation of R/W signal through A8 address line.


************************************************************************
***/
#ifndef LCD_H
#define LCD_H
#include <inttypes.h>


/* change these definitions to adapt setting */

#define LCD_LINES           4     /* visible lines */
#define LCD_LINE_LENGTH  0x40     /* internal line length */
#define LCD_START_LINE1  0x80     /* DDRAM address of first char of line 1 */
#define LCD_START_LINE2  0xC0     /* DDRAM address of first char of line 2 */
#define LCD_START_LINE3  0x94     /* DDRAM address of first char of line 3 */
#define LCD_START_LINE4  0xD4     /* DDRAM address of first char of line 4 */


#define LCD_IO_MODE      1        /* 0: memory mapped mode, 1: IO port mode */

#if LCD_IO_MODE
/*
 *  IO mode
 *  change LCD_PORT is you want to use a different port, but all lcd pins must be
```

```
 *  on  the SAME port in the same order
 */
#define LCD_PORT         PORTA     /* all lcd pins must be on SAME port */
#define LCD_DATA_PORT    LCD_PORT  /* port for 4bit data (Pin 0..3) */
#define LCD_RS_PORT      LCD_PORT  /* port for RS line */
#define LCD_RS_PIN       4
#define LCD_RW_PORT      LCD_PORT  /* port for RW line */
#define LCD_RW_PIN       5
#define LCD_E_PORT       LCD_PORT  /* port for Enable line */
#define LCD_E_PIN        6

#elif defined(__AVR_AT90S4414__) || defined(__AVR_AT90S8515__) ||
defined(__AVR_ATmega64__) || \
     defined(__AVR_ATmega8515__)|| defined(__AVR_ATmega103__) ||
defined(__AVR__ATmega128__) || \
     defined(__AVR_ATmega161__) || defined(__AVR_ATmega162__)
/*
 * memory mapped mode is only supported when the device has an external data memory
interface
 */
#define LCD_IO_DATA      0xC000    /* A15=E=1, A14=RS=1                */
#define LCD_IO_FUNCTION  0x8000    /* A15=E=1, A14=RS=0                */
#define LCD_IO_READ      0x0100    /* A8 =R/W=1 (R/W: 1=Read, 0=Write   */
#else
#error "external data memory interface not available for this device, use 4-bit IO port
mode"

#endif


/* you shouldn't need to change anything below this line */


/* instruction register bit positions */
#define LCD_CLR             0      /* DB0: clear display */
#define LCD_HOME            1      /* DB1: return to home position */
#define LCD_ENTRY_MODE      2      /* DB2: set entry mode */
#define LCD_ENTRY_INC       1      /*   DB1: 1=increment, 0=decrement  */
#define LCD_ENTRY_SHIFT     0      /*   DB2: 1=display shift on        */
#define LCD_ON              3      /* DB3: turn lcd/cursor on */
#define LCD_ON_DISPLAY      2      /*   DB2: turn display on */
#define LCD_ON_CURSOR       1      /*   DB1: turn cursor on */
#define LCD_ON_BLINK        0      /*     DB0: blinking cursor ? */
#define LCD_MOVE            4      /* DB4: move cursor/display */
#define LCD_MOVE_DISP       3      /*   DB3: move display (0-> cursor) ? */
#define LCD_MOVE_RIGHT      2      /*   DB2: move right (0-> left) ? */
```

```
#define LCD_FUNCTION       5     /* DB5: function set */
#define LCD_FUNCTION_8BIT   4     /*   DB4: set 8BIT mode (0->4BIT mode) */
#define LCD_FUNCTION_2LINES 3     /*   DB3: two lines (0->one line) */
#define LCD_FUNCTION_10DOTS 2     /*   DB2: 5x10 font (0->5x7 font) */
#define LCD_CGRAM          6     /* DB6: set CG RAM address */
#define LCD_DDRAM          7     /* DB7: set DD RAM address */
#define LCD_BUSY           7     /* DB7: LCD is busy */

/* set entry mode: display shift on/off, dec/inc cursor move direction */
#define LCD_ENTRY_DEC           0x04   /* display shift off, dec cursor move dir */
#define LCD_ENTRY_DEC_SHIFT     0x05   /* display shift on,  dec cursor move dir
*/
#define LCD_ENTRY_INC_          0x06   /* display shift off, inc cursor move dir */
#define LCD_ENTRY_INC_SHIFT     0x07   /* display shift on,  inc cursor move dir */

/* display on/off, cursor on/off, blinking char at cursor position */
#define LCD_DISP_OFF            0x08   /* display off                        */
#define LCD_DISP_ON             0x0C   /* display on, cursor off             */
#define LCD_DISP_ON_BLINK       0x0D   /* display on, cursor off, blink char     */
#define LCD_DISP_ON_CURSOR      0x0E   /* display on, cursor on              */
#define LCD_DISP_ON_CURSOR_BLINK 0x0F   /* display on, cursor on, blink char
*/

/* move cursor/shift display */
#define LCD_MOVE_CURSOR_LEFT    0x10   /* move cursor left  (decrement)
*/
#define LCD_MOVE_CURSOR_RIGHT   0x14   /* move cursor right (increment)
*/
#define LCD_MOVE_DISP_LEFT      0x18   /* shift display left                 */
#define LCD_MOVE_DISP_RIGHT     0x1C   /* shift display right                */

/* function set: set interface data length and number of display lines */
#define LCD_FUNCTION_4BIT_1LINE  0x20   /* 4-bit interface, single line, 5x7 dots
*/
#define LCD_FUNCTION_4BIT_2LINES 0x28   /* 4-bit interface, dual line,   5x7 dots
*/
#define LCD_FUNCTION_8BIT_1LINE  0x30   /* 8-bit interface, single line, 5x7 dots
*/
#define LCD_FUNCTION_8BIT_2LINES 0x38   /* 8-bit interface, dual line,   5x7 dots
*/


#define LCD_MODE_DEFAULT     ((1<<LCD_ENTRY_MODE) |
(1<<LCD_ENTRY_INC) )
```

```
/*
** function prototypes
*/
extern void lcd_command(uint8_t cmd);
extern void lcd_gotoxy(uint8_t x, uint8_t y);
extern void lcd_clrscr(void);
extern void lcd_home(void);
extern void lcd_putc(char c);
extern void lcd_puts(const char *s);
extern void lcd_puts_p(const char *progmem_s);
extern void lcd_init(uint8_t dispAttr);
extern void lcd_putn(double data);
/*
** macros for automatically storing string constant in program memory
*/
#ifndef P
#define P(s) ({static const char c[] __attribute__ ((progmem)) = s;c;})
#endif
#define lcd_puts_P(__s)         lcd_puts_p(P(__s))


#endif //LCD_H
```

```
#include "motor1.h"

 void pwm_init(void){

        TCCR1A = 170;
        TCCR1B = 0x12;

        ICR1 = 20000;

        TCCR3A = 170;
        TCCR3B = 0x12;

        ICR3 = 20000;
                // Set the Steering angle at 1500
 }

void servo(int servonumber,int angle) {
                if (angle>maxang)
                {
                angle = maxang;
                }

                if (angle<minang)
                {
                angle = minang;
                }
                if (servonumber == 1){
        OCR1A = angle;}
                if (servonumber == 2){
        OCR1B = angle;}
                if (servonumber == 3){
        OCR1C = angle;}
                if (servonumber == 4){
        OCR3A = angle;}
                if (servonumber == 5){
        OCR3B = angle;}
                if (servonumber == 6){
        OCR3C = angle;}
        }

void motor(int motornumber,int speed) {
if (motornumber == 2){
if (speed>0){
cbi(PORTE,7);
}
```

```
if (speed<0){
sbi(PORTE,7);
speed=-1*speed;
}

if (speed>maxspeed){
        speed = maxspeed;}

if (speed<minspeed){
        speed = minspeed;}

OCR1B = speed;
}


if (motornumber == 1){
if (speed>0){
cbi(PORTB,0);
}

if (speed<0){
sbi(PORTB,0);
speed=-1*speed;
}
if (speed>maxspeed){
        speed = maxspeed;}

if (speed<minspeed){
        speed = minspeed;}

OCR1A = speed;

}
}
```

```
// The definitions for the motors and servos

void pwm_init(void);
void servo(int servonumber,int angle);
void motor(int motornumber,int speed);
```

```
// contains all the global variables needed for the routines

#define F_CPU 16000000              // 16MHz processor


typedef unsigned char  u08;
typedef         char  s08;
typedef unsigned short u16;
typedef         short s16;
typedef u08           bool;

// Vehicle Constants
double rad = 200.00;

// Global Position
double r = 0.00;
double psi = 0.00;
double theta = 0.00;
double y = 0.00;
double x = 0.00;
double theta1 = 0.00;
double rspeed = 0.00;
double ang1 = 0.00;

// Porportional Control Constants
double k = 2.50;
double kang = 1.0500;
double ktheta = 3.00;
double kspeed = 15.00;
double desired_r = 327.00;
double desired_psi = 180.00;
double desired_theta = 0.00;
double rcheck = 0.00;

//Motor and Servo Globals
// Right is 1250
// Left is 1850
int minspeed = -10000;
int maxspeed = 10000;
int minang = 1150;
int maxang = 1850;
int Rmax = 1150;
int Lmax = 1850;
int step = 150;
int DELAY = 100;
//double angle = 0.00;
```

```
//AD Globals
int ad1 = 10;
int ad2 = 10;
int ad3 = 10;
int ad4 = 10;
int ad5 = 10;
int ad6 = 10;
int ad7 = 10;
int ad8 = 10;
int data = 0;
int adval0 = 10;
int adval1 = 10;
int adval2 = 10;
int adval3 = 10;
int adval4 = 10;
int adval5 = 10;
int adval6 = 10;
int adval7 = 10;
int advalFL = 10;
int advalFR = 10;

int ch;
int count = 0;
//void setPortBit(char port, int pin);
//void clearPortBit(char port, int pin);

/*void setPortBit(char port, int pin){
        if (port=='B') {
          PORTB |= (1<<pin);
        }
}

void clearPortBit(char port, int pin){
        if (port=='B')   {
          PORTB &= ~(1<<pin);}
}*/
```

```
#include "serial.h"

void usart0_init(uint16_t baudrate){
// set buad rate

UBRR0H = (unsigned char) (baudrate >> 8);
UBRR0L = baudrate;

// enable TX and RX

UCSR0B |= (1 << RXEN0) | (1 << TXEN0);

// 8 data bits, 1 stop bit, no parity

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00);
}

unsigned int usart0_receive(void){
while (!(UCSR0A & (1 << RXC0)));

return UDR0;
}

void usart0_transmit(uint8_t data){
while (!(UCSR0A & (1 << UDRE0)));

UDR0 = data;
}

void get_data(void){
//lcd_puts("getting data");
int decp=0;
double tenp = 0.00;
int signp=1;
double numberp = 0.00;
int rbyte;

int dect=0;
double tent = 0.00;
int signt=1;
double numbert = 0.00;

int decr=0;
double tenr = 0.00;
int signr=1;
double numberr = 0.00;
```

```
for(;;){
rbyte = usart0_receive();
//lcd_putc(rbyte);
if (rbyte=='n'){
break;}
if (rbyte==27){
ch=27;
break;}
}

do{
rbyte = usart0_receive();
//lcd_putc(rbyte);
if (rbyte==27){
ch=27;
break;}
if((rbyte<='9')&(rbyte>='-'))
{
if (rbyte=='-')
        {if(numberp==0.00)
        {
        signp=-1;}
        }
if((rbyte>='.')&(rbyte<='9')){
        if(rbyte=='.'){
        decp=1;}

        if (decp==0){
        numberp=numberp*10+rbyte-48;      }

        if (decp>=1){
        decp=decp+1;
        if (decp == 1){
        tenp = 10.00;
        }
        if (decp == 2){
        tenp = 100.00;
        }
        if (decp == 3){
        tenp = 1000.00;
        }
        if (decp == 4){
        tenp = 10000.00;
        }
        if (decp == 5){
```

```
            tenp = 100000.00;
            }
            if (decp == 6){
            tenp = 1000000.00;
            }
            numberp=numberp+(rbyte-48)/(tenp);
            }
            }
} // end if valid number
} while(rbyte!=' ');// end psi

psi=numberp*signp;


do{
rbyte = usart0_receive();
//lcd_putc(rbyte);
if (rbyte==27){
ch=27;
break;}
if((rbyte<='9')&(rbyte>='-'))
{
if (rbyte=='-')
            {if(numbert==0.00)
            {
            signt=-1;}
            }
if((rbyte>='.')&(rbyte<='9')){
            if(rbyte=='.'){
            dect=1;}

            if (dect==0){
            numbert=numbert*10+rbyte-48;        }

            if (dect>=1){
            dect=dect+1;
            if (dect == 1){
            tent = 10.00;
            }
            if (dect == 2){
            tent = 100.00;
            }
            if (dect == 3){
            tent = 1000.00;
            }
            if (dect == 4){
```

```
            tent = 10000.00;
            }
            if (dect == 5){
            tent = 100000.00;
            }
            if (dect == 6){
            tent = 1000000.00;
            }
            numbert=numbert+(rbyte-48)/(tent);
            }
            }
} // end if valid number
} while(rbyte!=' ');// end theta

theta=numbert*signt;

do{
rbyte = usart0_receive();
//lcd_putc(rbyte);
if (rbyte==27){
ch=27;
break;}
if((rbyte<='9')&(rbyte>='-'))
{
if (rbyte=='-')
            {if(numberr==0.00)
            {
            signr=-1;}
            }
if((rbyte>='.')&(rbyte<='9')){
            if(rbyte=='.'){
            decr=1;}

            if (decr==0){
            numberr=numberr*10+rbyte-48;        }

            if (decr>=1){
            decr=decr+1;
            if (decr == 1){
            tenr = 10.00;
            }
            if (decr == 2){
            tenr = 100.00;
            }
            if (decr == 3){
            tenr = 1000.00;
```

```
        }
        if (decr == 4){
        tenr = 10000.00;
        }
        if (decr == 5){
        tenr = 100000.00;
        }
        if (decr == 6){
        tenr = 1000000.00;
        }
        numberr=numberr+(rbyte-48)/(tenr);
        }
        }
} // end if valid number
} while(rbyte!='m');// end r

r=numberr*signr;
}
```

```
void usart1_init(uint16_t baudrate){
// set buad rate

UBRR1H = (unsigned char) (baudrate >> 8);
UBRR1L = baudrate;

// enable TX and RX

UCSR1B |= (1 << RXEN1) | (1 << TXEN1);

// 8 data bits, 1 stop bit, no parity

UCSR1C |= (1 << UCSZ11) | (1 << UCSZ10);
}

unsigned int usart1_receive(void){
while (!(UCSR1A & (1 << RXC1)));
```

```
return UDR1;
}

void usart1_transmit(uint8_t data){
while (!(UCSR1A & (1 << UDRE1)));

UDR1 = data;
}


int get_angle(void){
//lcd_puts("getting data");
int signa=1;
int numbera = 0;
int rbyte;

for(;;){
rbyte = usart0_receive();
//lcd_putc(rbyte);
if (rbyte=='n'){
break;}
if (rbyte==27){
ch=27;
break;}
}
while(1){
do{
rbyte = usart0_receive();
if (rbyte=='.'){
        break;}
//lcd_putc(rbyte);
if (rbyte==27){
ch=27;
break;}
if((rbyte<='9')&(rbyte>='-'))
{
if (rbyte=='-')
        {if(numbera==0.00)
        {
        signa=-1;}
        }

if((rbyte>='.')&(rbyte<='9')){
        numbera=numbera*10+rbyte-48;}
```

```
} // end if valid number
} while(rbyte!='m');// end r
if ((numbera>Rmax-1)&(numbera<Lmax+1)||(numbera==0)){
break;
}
}
return numbera*signa;
}
```

```
//function prototypes

void usart0_init(uint16_t baudrate);
unsigned int usart0_receive(void);
void usart0_transmit(uint8_t data);
void get_data(void);
int get_angle(void);

void usart1_init(uint16_t baudrate);
unsigned int usart1_receive(void);
void usart1_transmit(uint8_t data);
```

```
/* Program to interface with the serial port*/

#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define PORT1 0x3F8
/* Defines Serial Ports Base Address*/
/* COM1 0x3F8*/
/* COM2 0x2F8*/
/* COM3        0x3E8*/
/* COM4 0x2E8*/

void wait(int waittime){ /* Create a waiting function*/

        int time1, time2, time3;
        for (time1 = 0; time1 < waittime; time1++) {
        for (time2 = 0; time2 < 500; time2++){
        for (time3 = 0; time3 < 50; time3++);
        }}}

void main(void)
{
int i;
int j;
int k;
int m;
int ch;
int outch;
int choice;
FILE *fpwhole;

outportb(PORT1+1, 0); /* Turn off interrupts-Port

outportb(PORT1+3, 0x80); /* Set DLAB on*/
outportb(PORT1+0, 0x02); /* Set Baud rate - Divisor Latch Low Byte*/
                        /* Default 0x03 = 38,400 BPS*/
                        /*        0x01 = 115,200 BPS*/
                        /*          0x02 = 56,700 BPS*/
                        /*          0x06 = 19,200 BPS*/
                        /*          0x0C = 9,600 BPS*/
                        /*          0x18 = 4,800 BPS*/
                        /*          0x30 = 2,400 BPS*/

outportb(PORT1+1,0x00); /*Set Baud Rate - Divisor Latch High Byte*/
```

```
outportb(PORT1+3,0x03); /*8 bit, No Parity, 1 Stop Bit*/
outportb(PORT1+2,0xC7); /*FIFO Control Register*/
outportb(PORT1+4,0x0B); /*Turn on DTR, RTS, and OUT2*/

clrscr();
printf("Please Choose one of the following options\n");
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");

do{
if(kbhit()){
choice = getch();}
if (choice==27){
break;}

if (choice==49){
clrscr();
printf("Docking");
for(;;) {
outportb(PORT1,49);

fpwhole = fopen("ang.txt","r");
clrscr();
while((outch=getc(fpwhole))!=EOF){
if(outch==109){
fseek(fpwhole,0,SEEK_SET);
while((outch=getc(fpwhole))!=EOF){
printf("%c",outch);
outportb(PORT1,outch);
wait(30);

if (kbhit()){
clrscr();
printf("Please Choose one of the following options\n");
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");
choice = 7;
break;} /* end kbhit*/
} /* end EOF*/
if (choice == 7){
break;}
} /* end if*/
```

```
if (choice == 7){
break;}
} /* end EOF*/
if (choice == 7){
break;}
} /* end forever*/
} /* end choice = 49*/

if (choice==50){

clrscr();
printf("Obstacle Avoidance\n");
printf("Press any key to stop");
for(;;){
outportb(PORT1,50);
wait(30);

if(kbhit()){
clrscr();
printf("Please Choose one of the following options\n");
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");
choice = 7;
break;
} /* end if kbhit*/

} /* end forever*/
} /* end if 50*/

if (choice==51){
for(m=0;m<500;m++){
outportb(PORT1,51);}
for(;;){
fpwhole = fopen("ang.txt","r");

clrscr();
while((outch=getc(fpwhole))!=EOF){
if(outch==109){
fseek(fpwhole,0,SEEK_SET);
while((outch=getc(fpwhole))!=EOF){
printf("%c",outch);
outportb(PORT1,outch);
wait(500);
```

```
if(kbhit()){
clrscr();
printf("Please Choose one of the following options\n");
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");
choice = 7;
break;}

} /* end while EOF*/

} /* end 109*/
if (choice == 7){
break;}
} /* end EOF*/
if (choice == 7){
break;}
fclose(fpwhole);

wait(500);

if (kbhit()){
clrscr();
printf("Please Choose one of the following options\n");
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");
break;
}

}
}

if (choice==52){
clrscr();
printf("Robot Communication\n");
printf("Press any key to stop");
for(;;){
outportb(PORT1,52);
wait(40);

if(kbhit()){
clrscr();
printf("Please Choose one of the following options\n");
```

```
printf("(1) Docking\n");
printf("(2) Obstacle Avoidance\n");
printf("(3) Check Communication\n");
printf("(4) Robot Communication\n");
choice = 7;
break;
} /* end if kbhit*/

} /* end forever*/
} /* end if 52*/



if (kbhit()){
        ch = getch();
}

outportb(PORT1,27);
} while(ch!=27);

} /*end main*/□
```

```
// examplejp.cc
// simple point tracking program with desired output

#include <WINDOWS.H>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
#include <math.h>
#include "owl.h"

// Prototypes
void cross(double s1[3], double s2[3]);

// Globals
int a = 1;
int center = 1400;
double k = 16.0;
double kphi = 2.500;
double ktheta = 1.7500;
double kspeed = 15.00;
double desired_r = 327.00;
double desired_phi = 180.00;//198.00;
double desired_theta = 0.00;//20.00;
double rcheck = 0.00;
double rad = 500.00; //750.00
double PVu1;
double PVu2;
double PVu3;
double PVx;
double PVy;
double PVz;
double PVmag;
double PVd[3] = {0.00,0.00,0.00};
double PVu[3] = {0.00,0.00,0.00};
double cr[3] = {0.00,0.00,0.00};
double theta = 0;
double phi = 0;
double Costperpu[3] = {0.00,0.00,0.00};
double mags1s2 = 0;
double pi = 3.1415926;

// change these to match your configuration
#define BOX_NUMBER 0x20
#define MARKER_COUNT 10
#define SERVER_NAME "10.227.244.79"
```

```
// for hard real time
#define INIT_FLAGS 0

int n =0;
// for soft real time
//#define INIT_FLAGS OWL_POSTPROCESS

struct print_error {
  int err;
  print_error(int err) : err(err) { }
};

ostream &operator<<(ostream &out, print_error p);
ostream &operator<<(ostream &out, OWLMarker &m);


void create_point_tracker(int tracker, int box_id, int marker_count)
{
  cout << "point tracker " << tracker
       << ", box 0x" << hex << box_id << dec
       << ", markers " << marker_count << endl;

  // create tracker
  owlTrackeri(tracker, OWL_CREATE, OWL_POINT_TRACKER);

  // set markers
  for(int i = 0; i < marker_count; i++)
    owlMarkeri(MARKER(tracker, i), OWL_SET_LED, LED(box_id, i));

  // activate tracker
  owlTracker(tracker, OWL_ENABLE);
}  // end create_point_tracker


int main(void)
{

//FILE *file1;
//file1 = fopen("example1.txt", "w");
//FILE *filedata;
//filedata = fopen("filedata.txt", "w");
FILE *filewhole;
filewhole = fopen("c:\\TC\\whole.txt", "w");
FILE *fileang;
fileang = fopen("c:\\TC\\ang.txt", "w");
```

```
OWLMarker markers[32];

if(owlInit(SERVER_NAME, INIT_FLAGS) < 0) return 0;

// create tracker 0
create_point_tracker(0, 0x20, 5);
create_point_tracker(1, 0x21, 5);

// check for errors
if(!owlGetStatus())
  {
    int err = owlGetError();
    cerr << "error in point tracker setup: " << print_error(err) << endl;
    return 0;
  }

// set default frequency and start streaming
owlSetFloat(OWL_FREQUENCY, OWL_MAX_FREQUENCY);

// Set the matrix of markers equal to 0
float markermatrix[10][3]=  {{0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00},
                                                        {0.00, 0.00, 0.00}};

int kf1;

// main loop
for(;;)
{
// Set the matrix of markers equal to 0
          int w,v;
          for (v=1;v<11;v++){
                    for(w=1;w<4;w++){
                              markermatrix[v][w]=  0.00;
                    }}
int err;

    // get some markers
    int n = owlGetMarkers(markers, 32);
```

```
    // check for error
    if((err = owlGetError()) != OWL_NO_ERROR)
        {
         cerr << "error: " << print_error(err) << endl;
         break;
        }

    // no data yet.  wait for a bit
    if(n == 0)
        {
         timeval t = {0, 1000};
         select(0, 0, 0, 0, &t);

         continue;
        }

    if(n > 0)
        {
         for(int i = 0; i < n; i++)
         {

          if(markers[i].cond > 0)
                {
                        markermatrix[i+1][1] = markers[i].x;
                        markermatrix[i+1][2] = markers[i].y;
                        markermatrix[i+1][3] = markers[i].z;
                } // end if
//                  fprintf(file1, "%f %f %f\n", markers[i].x, markers[i].y, markers[i].z);
//            cout << i << ") " << markers[i] << endl;

         }  // end for int
         }  // end if n>0

// Calculations for Position Vector
int kl1 = 1;
int kf1 = kl1+5;
if(markermatrix[kl1][1]>=0.001 || markermatrix[kl1][1]<=-.001)
{
if(markermatrix[kf1][1]>=0.001 || markermatrix[kf1][1]<=-.001)
{
double PVx = markermatrix[kl1][1]-markermatrix[kf1][1];
double PVy = markermatrix[kl1][2]-markermatrix[kf1][2];
double PVz = markermatrix[kl1][3]-markermatrix[kf1][3];
double PVmag = pow(PVx*PVx+PVy*PVy+PVz*PVz,.5);
```

```
PVd[1] = PVx;
PVd[2] = PVy;
PVd[3] = PVz;

PVu[1] = PVx/PVmag;
PVu[2] = PVy/PVmag;
PVu[3] = PVz/PVmag;
//double PVu1 = PVx/PVmag;
//double PVu2 = PVy/PVmag;
//double PVu3 = PVz/PVmag;
//printf("PVu = %f %f %f\n",PVu[1],PVu[2],PVu[3]);

// Calculations for Heading
int kl2 = kl1+1;
int kf2 = kf1+1;
if(markermatrix[kl2][1]>=0.001 || markermatrix[kl2][1]<=-.001)
{
if(markermatrix[kf2][1]>=0.001 || markermatrix[kf2][1]<=-.001)
{
double Abbx = markermatrix[kf2][1]-markermatrix[kf1][1];
double Abby = markermatrix[kf2][2]-markermatrix[kf1][2];
double Abbz = markermatrix[kf2][3]-markermatrix[kf1][3];
double Abb = pow(Abbx*Abbx+Abby*Abby+Abbz*Abbz,.5);
double Abbu[3];
Abbu[1] = Abbx/Abb;
Abbu[2] = Abby/Abb;
Abbu[3] = Abbz/Abb;
//printf("Abbu = %f %f %f\n",Abbu[1],Abbu[2],Abbu[3]);

double Costx = markermatrix[kl2][1]-markermatrix[kl1][1];
double Costy = markermatrix[kl2][2]-markermatrix[kl1][2];
double Costz = markermatrix[kl2][3]-markermatrix[kl1][3];
double Cost = pow(Costx*Costx+Costy*Costy+Costz*Costz,.5);
double Costu[3];
Costu[1] = Costx/Cost;
Costu[2] = Costy/Cost;
Costu[3] = Costz/Cost;
//printf("Costu = %f %f %f\n",Costu[1],Costu[2],Costu[3]);
kl1 = 3;
kl2 = 5;

double Costperpx = markermatrix[kl2][1]-markermatrix[kl1][1];
double Costperpy = markermatrix[kl2][2]-markermatrix[kl1][2];
double Costperpz = markermatrix[kl2][3]-markermatrix[kl1][3];
double Costperp =
pow(Costperpx*Costperpx+Costperpy*Costperpy+Costperpz*Costperpz,.5);
```

```
Costperpu[1] = Costperpx/Costperp;
Costperpu[2] = Costperpy/Costperp;
Costperpu[3] = Costperpz/Costperp;
//printf("Costperpu = %f %f %f\n",Costperpu[1],Costperpu[2],Costperpu[3]);

// Use the dot product to find the phi
phi = acos(Abbu[1]*Costu[1]+Abbu[2]*Costu[2]+Abbu[3]*Costu[3]);
//printf("phi = %f\n", phi*180/3.14);

// Use the dot product to find the theta
theta = acos(PVu[1]*Costu[1]+PVu[2]*Costu[2]+PVu[3]*Costu[3]);
//printf("theta = %f\n",theta*180/3.14);

// Use the dot product to find the alpha (heading phi)
double alpha =
acos(Abbu[1]*Costperpu[1]+Abbu[2]*Costperpu[2]+Abbu[3]*Costperpu[3]);
//printf("Alpha = %f\n",alpha*180/pi);
if (alpha > pi/2){
        phi = 2*pi-phi;}

// Use the dot product to find the beta (orientation theta)
double beta = acos(PVu[1]*Costperpu[1]+PVu[2]*Costperpu[2]+PVu[3]*Costperpu[3]);
//printf("Beta = %f\n",beta*180/pi);
if (beta > pi/2){
theta = 2*pi-theta;}
//printf("Theta = %f\n",theta);
//FILE *filetheta;
//filetheta = fopen("theta.txt", "w");
//fprintf(filetheta,"%s","t");
//fprintf(filetheta, "%f",theta*180/pi);
//fprintf(filetheta,"%s","t");
//fclose(filetheta);

// Calulate X and Y positions
PVmag = pow(PVd[1]*PVd[1]+PVd[2]*PVd[2]+PVd[3]*PVd[3],.5);

double X;
X = (PVmag*cos(theta));

double Y;
Y = (PVmag*sin(theta));

//FILE *filephi;
//filephi = fopen("phi.txt", "w");
//fprintf(filephi,"%s","p");
//fprintf(filephi, "%f",phi*180/pi);
```

```
//fprintf(filephi,"%s","p");
//fclose(filephi);
n=n+1;

//printf("Theta = %f\n", theta*180/3.14);
//printf("X = %f\n", X);
//printf("Y = %f\n", Y);
//printf("phi = %f\n", phi*180/3.14);
//printf("PVmag = %f\n",PVmag);

//fprintf(filedata, "X = %f\n",X);
//fprintf(filedata, "Y = %f\n",Y);
//fprintf(filedata, "phi = %f\n",phi*180/pi);

//clrscr();
if((PVmag!=0.00)&(phi<360)&(phi>0)&(theta<360)&(theta>0)){

//printf("phi = %f\n", phi*180/pi);
//printf("Theta = %f\n", theta*180/pi);
//printf("R = %f\n", PVmag);

fseek(filewhole,0L,SEEK_SET);
fprintf(filewhole,"%s","n");
//printf("n");
fprintf(filewhole, "%f ",phi*180/pi);
//printf("%f ",phi*180/pi);
fprintf(filewhole, "%f ",theta*180/pi);
//printf( "%f ",theta*180/pi);
fprintf(filewhole, "%f",PVmag);
//printf("%f",PVmag);
fprintf(filewhole,"%s","m");
//printf("m\n");
}
double r = PVmag;
theta = theta*180/pi;
phi = phi*180/pi;

r=sqrt(r*r+desired_r*desired_r-2*r*desired_r*cos((theta-desired_theta)/180*pi));
//printf("PVmag = %f\n",PVmag);
//printf("phi = %f\n",phi);
//printf("Theta = %f\n",theta);
double y = sin(theta/180*pi)*r;
double x = cos(theta/180*pi)*r;
double theta1 = (tan(ktheta*y/x))*180/pi;
double ang1 = center-k*(kphi*(phi-desired_phi)-theta1);
//printf("Y = %f\n",y);
```

```
//printf("X = %f\n",x);
//printf("Theta1 = %f\n",theta1);
//printf("Ang1 = %f\n",ang1);
//printf("Steering = %f\n",ang1);
if (ang1<1150){
ang1=1150;}
if (ang1>1850){
ang1=1850;}

//avgerageangle = 0;
//double rspeed = kspeed*r;
//if (rspeed<5000){
//rspeed=5000;}

// Check physical limitations
if ((theta<90)&(theta>0)&(phi-180>theta)){
double alpha = 270+(theta-phi);}
else if ((theta<90)&(theta>0)&(phi-180<theta)){
alpha = -90-(theta-phi);}
else if ((theta<360)&(theta>270)&(180-theta+phi>0)){
alpha = -90+(theta-phi);}
else if ((theta<360)&(theta>270)&(180-theta+phi<0)){
alpha = 270-(theta-phi);}


rcheck = sqrt(rad*rad + r*r-2*r*rad*cos(alpha/180*pi));
//printf("Rcheck = %f\n",rcheck);
//printf("a = %d\n",a);
double avgrcheck = (avgrcheck*(a-1)+rcheck)/a;
if (a == 20){
printf("AvgRcheck = %f\n",avgrcheck);
printf("R = %f\n",r);
a=1;
if ((400>rcheck)&(r<300)&(theta>20)&(theta<340)){
ang1 = -1*((center-ang1)+center);
printf("Steering = %f\n",ang1);
//printf("Rcheck = %f\n",rcheck);
} // end backing from rcheck
else{
printf("Steering = %f\n",ang1);
}
avgrcheck=0;
if (r<80){
fseek(fileang,0L,SEEK_SET);
fprintf(fileang,"%s","n");
//printf("n");
```

```
fprintf(fileang,"%f",0.00);
//printf("%f",ang1);
fprintf(fileang,"%s","m");
//printf("m\n");
}
else{
//ang = ang+(ang1-ang)*.5;  // Go toward other robot
fseek(fileang,0L,SEEK_SET);
fprintf(fileang,"%s","n");
//printf("n");
fprintf(fileang,"%f",ang1);
//printf("%f",ang1);
//fseek(fileang,-6L,SEEK_CUR);
fprintf(fileang,"%s","m");
//printf("m\n");
}
} // average rcheck
else{
        a = a+1;}

} //end if
} //end if
}//end if
}//end if


PVmag = 0;
phi = 0;
theta = 0;

if (kbhit()){
break;}

} //end forever
fclose(filewhole);
fclose(fileang);
//fclose(filedata);

// cleanup
owlDone();

} //end main

// printing operators
ostream &operator<<(ostream &out, print_error p)
{
```

```
  if(p.err > 0)
    switch(p.err) {
    case OWL_NO_ERROR: out << "No Error"; break;
    case OWL_INVALID_VALUE: out << "Invalid Value"; break;
    case OWL_INVALID_ENUM: out << "Invalid Enum"; break;
    case OWL_INVALID_OPERATION: out << "Invalid Operation"; break;
    default: out << "0x" << hex << p.err << dec; break;
    }
  else out << p.err;

  return out;
}

ostream &operator<<(ostream &out, OWLMarker &m)
{
  out << m.x << " " << m.y << " " << m.z;
  return out;
}
```

```
/*****************************************************************
* This program allows the user to choose which routine to perform *
*Author: Chad Sylvester                                          *
*Date: July 28, 2003
          *
*Version: 1.0
                    *
*****************************************************************/

//include files
#include "global.h"
#include <avr/io.h>
#include <inttypes.h>
#include <string.h>
#include <math.h>
#include "lcd.h"
#include "serial.h"
#include "adc.h"
#include "motor1.h"
#include "routine.h"

// function prototypes
void wait(int waittime);
void io_init(void);


void io_init(void){
       DDRB = 0xff;// Most of PORTB is used for motors and servos
       DDRE = 0xff; // more servos
 }

void wait(int waittime){ // Create a waiting function

          int time1, time2, time3;
          for (time1 = 0; time1 < waittime; time1++) {
          for (time2 = 0; time2 < 500; time2++){
          for (time3 = 0; time3 < 50; time3++);
          }}
}

void main(void){
wait(500);
io_init();
pwm_init();
usart0_init(16);
```

```
usart1_init(16);
lcd_init(LCD_DISP_ON_BLINK);
adc_init();
wait(50);

// Set the initial angle of the servo to 1500 (center)
int ang = 1500;
servo(1,ang);

// Set the initial speed of the motor to 0
int speed = 10;
motor(2,speed);

double avgerageangle = 0;
int i = 0;
while(1){
ch = 0;
lcd_clrscr();
// Wait for an input from the serial port to start this routine
lcd_puts(" Waiting for Input");
lcd_command(LCD_START_LINE2);
lcd_puts("  from Computer");

for(;;){
int choice = usart0_receive();
//lcd_putc(choice);
if ((choice < 53)&(choice > 48)){

if (choice == 49){
lcd_clrscr();
lcd_puts("Docking");
do{

// Check for obstacles
int angmax = 1500;
int angmin = 1500;
int newang = 1500;
int turn = 0;
int g = 0;
// Read ADC signals
wait(100);
int advalFLnew = adc(FL);
advalFL = (advalFL+advalFLnew)/2;
wait(100);
int advalFRnew = adc(FR);
advalFR = (advalFR+advalFRnew)/2;
```

```
/*if ((advalFR > 300)&(advalFL > 300)||(advalFR > 500)||(advalFL > 500)&(r>500)){
lcd_clrscr();
lcd_puts("Avoid Obstacle");
//lcd_command(LCD_START_LINE2);
//lcd_puts("R = ");
//lcd_putn(r);
speed = -1;
motor(2,speed);
wait(10);

if(advalFR-advalFL>25){
turn = 1;
newang = maxang;}
else if(advalFR-advalFL<-25){
turn = 2;
newang = minang;}
else {
turn = 0;
newang = maxang;}

while ((advalFR>200)&(advalFL>200)){
wait(100);
advalFLnew = adc(FL);
advalFL = (advalFL+advalFLnew)/2;
wait(100);
advalFRnew = adc(FR);
advalFR = (advalFR+advalFRnew)/2;

// Go backwards
speed = speed +(minspeed - speed)*.5;

if (speed<0){
motor(2,speed);

if (turn == 1){
//turn right
ang = ang + (Rmax-ang)*.5;
servo(1,ang);
//wait(10);
}

else if (turn==2){
//turn left
ang = ang + (Lmax-ang)*.5;
servo(1,ang);
```

```
}

else{
ang = ang+(1500-ang)*.5;
servo(1,ang);
wait(10);
}

if (ang>angmax){
angmax = ang;
}

if (ang<angmin){
angmin = ang;
}

} // end if speed < 0
} // end while > 200

speed = 1;

motor(2,speed);

for (i=0;i<2000;i++){
speed = speed+(maxspeed-speed)*.5;
motor(2,speed);
ang = ang + (newang-ang)*.5;
servo(1,ang);
wait(1);
}

} // end if (& || ||)
*/
// Steering
if ((advalFR-advalFL>50)&(advalFR>250)&(r>700)){
ang = ang + (Lmax-ang)*.5;  //turn right
servo(1,ang);}

else if ((advalFL-advalFR>50)&(advalFL>250)&(r>500)){
ang = ang + (Rmax-ang)*.5;  //turn left
servo(1,ang);}

else{
//lcd_clrscr();
//lcd_puts("Get Angle Info");
int ang1 = get_angle();
```

```
count = count+1;
/*lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Angle = ");
lcd_putn(ang1);
lcd_command(LCD_START_LINE2);
lcd_puts("Count = ");
lcd_putn(count);
*/
if(ang1<0){
//lcd_clrscr();
//lcd_puts("Backwards");
for (g = 1;g<10;g++){
if (speed>=0.00){
speed = -10.00;
motor(2,0.00);
wait(20);}
else if (speed<0.00){
speed = speed+(minspeed-speed)*.5;
lcd_command(LCD_START_LINE3);
lcd_puts("Speed = ");
lcd_putn(speed);
motor(2,speed);}
//}  // end for g

//for (g = 1;g<100;g++){
if (ang1<0){
ang1 = -1.00*ang1;}
ang = ang + (ang1-ang)*.8;
/*lcd_command(LCD_START_LINE2);
lcd_puts("Angle = ");
lcd_putn(ang);
*/
servo(1,ang);
} // end for g
}
else if (ang1 == 0){
motor(2,0);
for(;;){
lcd_puts("ROUTINE");
routine();}

}

else
```

```
{
//lcd_clrscr();
//lcd_puts("Forward");

for (g = 1;g<10;g++){
if (speed<=0.00){
speed=10.00;
motor(2,0.00);
wait(20);}
else if (speed>0){
speed = speed+(maxspeed-speed)*.5;}
lcd_command(LCD_START_LINE3);
lcd_puts("Speed = ");
lcd_putn(speed);
motor(2,speed);
//} // end for g

//for (g = 1;g<100;g++){
ang = ang + (ang1-ang)*.8;
//lcd_command(LCD_START_LINE2);
//lcd_puts("Angle = ");
//lcd_putn(ang);
servo(1,ang);
wait(10);
} // end for g

} // end else

} // end else
// Reset Values
r = 0.00;
psi = 0.00;
theta = 0.00;
ang1 = 0;
} while(ch!=27);
} // end Choice 1 : Docking

motor(2,0);    // Stop Motor between choices
servo(1,1500); // Set steering to center between choices

if (choice == 50 ){ //Obstacle Avoidance
lcd_clrscr();
lcd_puts("Obstacle Avoidance");
while(1){
int angmax = 1500;
int angmin = 1500;
```

```
int newang = 1500;
int turn = 0;
// Read ADC signals
wait(100);
int advalFLnew = adc(FL);
advalFL = (advalFL+advalFLnew)/2;
wait(100);
int advalFRnew = adc(FR);
advalFR = (advalFR+advalFRnew)/2;

// Display IRs and speed on LCD
lcd_clrscr(); // clear the LCD screen
lcd_puts("Obstacle Avoidance");
/*lcd_command(LCD_START_LINE2);
lcd_puts("Speed = ");
lcd_putn(speed);
lcd_command(LCD_START_LINE3);
lcd_puts("FL IR=");
lcd_putn(advalFL);
lcd_command(LCD_START_LINE4);
lcd_puts("FR IR=");
lcd_putn(advalFR);*/


if ((advalFR > 300)&(advalFL > 300)||(advalFR > 500)||(advalFL > 500)){
speed = -1;
motor(2,speed);
wait(10);

if(advalFR-advalFL>25){
turn = 1;
newang = maxang;}
else if(advalFR-advalFL<-25){
turn = 2;
newang = minang;}
else {
turn = 0;
newang = maxang;}

while ((advalFR>200)&(advalFL>200)){
wait(100);
advalFLnew = adc(FL);
advalFL = (advalFL+advalFLnew)/2;
wait(100);
advalFRnew = adc(FR);
advalFR = (advalFR+advalFRnew)/2;
```

```
// Go backwards
speed = speed +(minspeed - speed)*.5;

if (speed<0){
motor(2,speed);

if (turn == 1){
//turn right
ang = ang + (Rmax-ang)*.5;
servo(1,ang);
}

else if (turn==2){
//turn left
ang = ang + (Lmax-ang)*.5;
servo(1,ang);
}

else{
ang = ang+(1500-ang)*.5;
servo(1,ang);
wait(10);
}

if (ang>angmax){
angmax = ang;
}

if (ang<angmin){
angmin = ang;
}

} // end if speed < 0
} // end while > 200

speed = 1;

motor(2,speed);

for (i=0;i<2000;i++){
speed = speed+(maxspeed-speed)*.5;
motor(2,speed);
ang = ang + (newang-ang)*.5;
servo(1,ang);
wait(1);
```

```
}

} // end if (& || ||)



else {
avgerageangle = 0;
if (speed>0){
speed = speed+(maxspeed-speed)*.5;
motor(2,speed);
wait(10);

if ((advalFR-advalFL>50)&(advalFR>250)){
//turn right
ang = ang + (Lmax-ang)*.5;
servo(1,ang);
}

else if ((advalFL-advalFR>50)&(advalFL>250)){
//turn left
ang = ang + (Rmax-ang)*.5;
servo(1,ang);
}

else{
ang = 1500+(1500-ang)*.5;
servo(1,ang);
wait(10);
}
}

else{
speed = speed+(maxspeed-speed)*.5;
motor(2,speed);
wait(10);
}

} // end else
}// end while(1)
}// end Choice 1: Obstacle Avoidance

motor(2,0);    // Stop Motor between choices
servo(1,1500); // Set steering to center between choices

if (choice == 51){
```

```
lcd_clrscr();
lcd_puts("Check Communication");
do{  //Start of serial
get_data();
r=sqrt(r*r+desired_r*desired_r-2*r*desired_r*cos((theta-desired_theta)/180*M_PI));
lcd_clrscr();
lcd_puts("R = ");
lcd_putn(r);

lcd_command(LCD_START_LINE2);
lcd_puts("Psi = ");
lcd_putn(psi);

lcd_command(LCD_START_LINE3);
lcd_puts("Theta = ");
lcd_putn(theta);
}while(ch!=27);

} // end Choice 3 : Communication Check

motor(2,0);    // Stop Motor between choices
servo(1,1500); // Set steering to center between choices

if (choice == 52){
lcd_clrscr();
rspeed=0;
motor(2,rspeed);

while(1){
routine();
} // end while(1)
} // end Choice 4 : Robot Communication

if (ch == 27){
break;}

} // end wait for serial input
} // end forever
} // end while(1)
} // end main

#include <serial.c>
#include <motor1.c>
#include <lcd.c>
#include <adc.c>
```

```
#include <routine.c>



void routine(void){

//globals
int hold = 10000;
int hold1 = 10000;
int ten = 5;
int rc = 0;
int z = 0;

while(1){

//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well Costello, I'm");
lcd_command(LCD_START_LINE2);
lcd_puts("going to New York");
lcd_command(LCD_START_LINE3);
lcd_puts("with you. You ");
lcd_command(LCD_START_LINE4);
lcd_puts("know, Bucky Harris,");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("the Yank's manager");
lcd_command(LCD_START_LINE2);
lcd_puts("gave me a job as");
lcd_command(LCD_START_LINE3);
lcd_puts("coach for as long as");
lcd_command(LCD_START_LINE4);
lcd_puts("you're on the team.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
//lcd_putc(rc);
if ((rc<125)&(rc>45)){
break;}}
```

```
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Right, certainly do.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Oh, I'll tell you");
lcd_command(LCD_START_LINE2);
lcd_puts("their names, but you");
lcd_command(LCD_START_LINE3);
lcd_puts("know strange as it");
lcd_command(LCD_START_LINE4);
lcd_puts("may seem, they give");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("these ball players");
lcd_command(LCD_START_LINE2);
lcd_puts("now a days, very");
lcd_command(LCD_START_LINE3);
lcd_puts("peculiar names.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Strange names, pet");
```

```
lcd_command(LCD_START_LINE2);
lcd_puts("names. Like, Dizzy");
lcd_command(LCD_START_LINE3);
lcd_puts("Dean, and…");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit((LCD_START_LINE2));
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command((LCD_START_LINE1));
lcd_puts("Daffy Dean.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("French?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Goofe' Dean, oh I");
```

```
lcd_command(LCD_START_LINE2);
lcd_puts("see! Well let's see,");
lcd_command(LCD_START_LINE3);
lcd_puts("we have on the bags,");
lcd_command(LCD_START_LINE4);
lcd_puts("we have Who's on");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("first, What's on");
lcd_command(LCD_START_LINE2);
lcd_puts("second, and I Don't");
lcd_command(LCD_START_LINE3);
lcd_puts("Know is on third.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I say, Who's on");
lcd_command(LCD_START_LINE2);
lcd_puts("first, What's on");
lcd_command(LCD_START_LINE3);
lcd_puts("second, and I Don't");
lcd_command(LCD_START_LINE4);
lcd_puts("Know's on third.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well I should.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}
```

```
while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who is on first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's the man's");
lcd_command(LCD_START_LINE2);
lcd_puts("name.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yeah.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
```

```
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's it.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yeah.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Certainly.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("That’s right.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Every dollar of it.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That’s it.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("He does, every");
lcd_command(LCD_START_LINE2);
lcd_puts("dollar! Sometimes");
lcd_command(LCD_START_LINE3);
lcd_puts("his wife comes down");
lcd_command(LCD_START_LINE4);
lcd_puts("and collects it.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
lcd_command(LCD_START_LINE2);
lcd_puts("What's wrong with");
lcd_command(LCD_START_LINE3);
lcd_puts("that?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
```

```
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's how he signs");
lcd_command(LCD_START_LINE2);
lcd_puts("it!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, what's on second");
lcd_command(LCD_START_LINE2);
lcd_puts("base.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who is on first!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well don't change");
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("the players around!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Take it easy, buddy.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's right.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Alright.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, What is on");
lcd_command(LCD_START_LINE2);
lcd_puts("second!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who's on first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Oh, he's on third.");
lcd_command(LCD_START_LINE2);
lcd_puts("We're not talking");
lcd_command(LCD_START_LINE3);
lcd_puts("about him. Now let's");
```

```
lcd_command(LCD_START_LINE4);
lcd_puts("get back to first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well you mentioned");
lcd_command(LCD_START_LINE2);
lcd_puts("his name.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, Who's playing");
lcd_command(LCD_START_LINE2);
lcd_puts("first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("What’s on second.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("He’s on third.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Alright, what do you");
lcd_command(LCD_START_LINE2);
lcd_puts("want to know?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Why do you insist on");
```

```
lcd_command(LCD_START_LINE2);
lcd_puts("putting Who on third");
lcd_command(LCD_START_LINE3);
lcd_puts("base?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, What is on");
lcd_command(LCD_START_LINE2);
lcd_puts("second.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, Who is on first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Both:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");

lcd_clrscr();
lcd_command((LCD_START_LINE1));
lcd_puts("Third base!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Sure.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Why.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("Well I just thought");
lcd_command(LCD_START_LINE2);
lcd_puts("I'd tell you.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who is playing first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No, What is on");
lcd_command(LCD_START_LINE2);
lcd_puts("second.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("No, Who is on first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Both:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");

lcd_clrscr();
lcd_command((LCD_START_LINE1));
lcd_puts("Third base!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Why!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("No, he’s center");
lcd_command(LCD_START_LINE2);
lcd_puts("field.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well that’s the");
lcd_command(LCD_START_LINE2);
lcd_puts("fellow’s name.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Sure.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
```

```
lcd_command(LCD_START_LINE1);
lcd_puts("Tomorrow.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I'm telling you");
lcd_command(LCD_START_LINE2);
lcd_puts("then.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Tomorrow.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What time what?");
```

```
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now listen, Who");
lcd_command(LCD_START_LINE2);
lcd_puts("is not pitching.");
lcd_command(LCD_START_LINE3);
lcd_puts("Who is on…");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What's on second!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Both:
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");
```

```
lcd_clrscr();
lcd_command((LCD_START_LINE1));
lcd_puts("Third base!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Certainly.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Today.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now you've got it.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("So they tell me.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now that's the first");
lcd_command(LCD_START_LINE2);
lcd_puts("thing that you've");
lcd_command(LCD_START_LINE3);
lcd_puts("said right.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well that's all you");
lcd_command(LCD_START_LINE2);
lcd_puts("have to do!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
```

```
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("No you don't! You");
lcd_command(LCD_START_LINE2);
lcd_puts("throw the ball to");
lcd_command(LCD_START_LINE3);
lcd_puts("Who!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's different.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You're not saying");
lcd_command(LCD_START_LINE2);
lcd_puts("that.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}
```

```
while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You throw it to Who.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's it.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Listen, you ask me.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
```

```
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You throw the ball");
lcd_command(LCD_START_LINE2);
lcd_puts("to Who?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's it.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
```

```
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You just changed");
lcd_command(LCD_START_LINE2);
lcd_puts("them around.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Yes.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Oh…What?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Abbott:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("Oh, that’s our short");
lcd_command(LCD_START_LINE2);
lcd_puts("stop.");

wait(20000);

} // end while

} // end routine
```

```
/*********************************************************************
* This program is used by the still robot (Costello) and prints info *
*Author: Chad Sylvester                                              *
*Date: July 28, 2003
          *
*Version: 1.0
                    *
*********************************************************************/

//include files
#include "global.h"
#include <avr/io.h>
#include <inttypes.h>
#include <string.h>
#include <math.h>
//#include <avr\signal.h>
//#include <avr\interrupt.h>
#include "lcd.h"
#include "serial.h"
//#include "uart0.h"
#include "adc.h"
#include "motor1.h"
#include "routine.h"

// function prototypes
void wait(int waittime);
void io_init(void);
void setPortBit(char port, int pin);
void clearPortBit(char port, int pin);

void setPortBit(char port, int pin){
        if (port=='B') {
          PORTB |= (1<<pin);
        }
}

void clearPortBit(char port, int pin){
        if (port=='B')   {
          PORTB &= ~(1<<pin);}
}


void io_init(void){
      DDRB = 0xff;// Most of PORTB is used for motors and servos
      DDRE = 0xff; // more servos
```

```
                    //PORTE = 128;
 }

void wait(int waittime){ // Create a waiting function

        int time1, time2, time3;
        for (time1 = 0; time1 < waittime; time1++) {
        for (time2 = 0; time2 < 500; time2++){
        for (time3 = 0; time3 < 50; time3++);
        }}
}

void main(void){
wait(500);
io_init();
usart1_init(16);
lcd_init(LCD_DISP_ON_BLINK);
wait(50);

lcd_puts("   Who's On First");
lcd_command(LCD_START_LINE2);
lcd_puts("Abbott and Costello");

routine();
} // end main

#include <serial.c>
//#include <uart0.c>
#include <motor1.c>
#include <lcd.c>
#include <adc.c>
#include <routine.c>

// Costello stays still while Abbott comes to look for him. When they dock Costello
// sends the info to Abbott

//globals
int hold = 10000;
int hold1 = 10000;
int ten = 5;
int rc = 0;

void routine(void){
int z = 0;
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("   Who's On First");
lcd_command(LCD_START_LINE2);
lcd_puts("Abbott and Costello");

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look Abbott, if");
lcd_command(LCD_START_LINE2);
lcd_puts("you're the coach,");
lcd_command(LCD_START_LINE3);
lcd_puts("you must know all");
lcd_command(LCD_START_LINE4);
lcd_puts("the players.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well I never met the");
lcd_command(LCD_START_LINE2);
lcd_puts("guys, so you'll have");
lcd_command(LCD_START_LINE3);
lcd_puts("to tell me their");
lcd_command(LCD_START_LINE4);
lcd_puts("names, and then I'll");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("know whos playing on");
lcd_command(LCD_START_LINE2);
lcd_puts("the team.");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You mean funny");
lcd_command(LCD_START_LINE2);
lcd_puts("names?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("His brother Daffy?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("And their French");
lcd_command(LCD_START_LINE2);
lcd_puts("cousin.");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Goofe'.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's what I want");
lcd_command(LCD_START_LINE2);
lcd_puts("to find out.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Are you the");
lcd_command(LCD_START_LINE2);
lcd_puts("manager?");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You going to be");
lcd_command(LCD_START_LINE2);
lcd_puts("the coach too?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("And you don't know");
lcd_command(LCD_START_LINE2);
lcd_puts("the fellow's names?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well then who is");
lcd_command(LCD_START_LINE2);
lcd_puts("on first?");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I mean the fellow's");
lcd_command(LCD_START_LINE2);
lcd_puts("name.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The guy on first.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The first baseman.");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The guy playing");
lcd_command(LCD_START_LINE2);
lcd_puts("first base.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I'm asking you who's");
lcd_command(LCD_START_LINE2);
lcd_puts("on first!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's whose name?");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well go ahead and");
lcd_command(LCD_START_LINE2);
lcd_puts("tell me.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's who?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look, you got a");
lcd_command(LCD_START_LINE2);
lcd_puts("first baseman?");
wait(hold);

for (z=1;z<ten;z++){
```

```
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who's playing");
lcd_command(LCD_START_LINE2);
lcd_puts("first?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("When you pay off the");
lcd_command(LCD_START_LINE2);
lcd_puts("first baseman every");
lcd_command(LCD_START_LINE3);
lcd_puts("month, who gets the");
lcd_command(LCD_START_LINE4);
lcd_puts("money?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("All I'm trying to");
lcd_command(LCD_START_LINE2);
lcd_puts("find out is the");
lcd_command(LCD_START_LINE3);
lcd_puts("fellow's name on");
lcd_command(LCD_START_LINE4);
lcd_puts("first base.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The guy that gets");
lcd_command(LCD_START_LINE2);
lcd_puts("the money.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who gets the money");
lcd_command(LCD_START_LINE2);
lcd_puts("on first base?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
```

```
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Whose wife?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look, all I want to");
lcd_command(LCD_START_LINE2);
lcd_puts("know is when you");
lcd_command(LCD_START_LINE3);
lcd_puts("sign up the first");
lcd_command(LCD_START_LINE4);
lcd_puts("baseman, how does");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("he sign his name to");
lcd_command(LCD_START_LINE2);
lcd_puts("the contract?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("The guy.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("How does he sign it?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("All I'm trying to");
lcd_command(LCD_START_LINE2);
lcd_puts("find out is what's");
lcd_command(LCD_START_LINE3);
```

```
lcd_puts("the guy’s name on");
lcd_command(LCD_START_LINE4);
lcd_puts("first base.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I’m not asking who’s");
lcd_command(LCD_START_LINE2);
lcd_puts("on second.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("One base at a time!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("I’m not changing");
lcd_command(LCD_START_LINE2);
lcd_puts("nobody!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("All I’m asking you,");
lcd_command(LCD_START_LINE2);
lcd_puts("who’s the guy on");
lcd_command(LCD_START_LINE3);
lcd_puts("first base?!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Okay.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
```

```
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What's the guy's");
lcd_command(LCD_START_LINE2);
lcd_puts("name on first base?!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I'm not asking you");
lcd_command(LCD_START_LINE2);
lcd_puts("who's on second!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I don't know.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
```

```
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now how did I get on");
lcd_command(LCD_START_LINE2);
lcd_puts("third base?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("If I mentioned the");
lcd_command(LCD_START_LINE2);
lcd_puts("third baseman's name");
lcd_command(LCD_START_LINE3);
lcd_puts("who did I say's");
lcd_command(LCD_START_LINE4);
lcd_puts("playing third?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What's on first?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
```

```
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I don't know.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("There I go, back on");
lcd_command(LCD_START_LINE2);
lcd_puts("third again! Will");
lcd_command(LCD_START_LINE3);
lcd_puts("you stay on third");
lcd_command(LCD_START_LINE4);
lcd_puts("base and don't go");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("off it?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now who's playing");
lcd_command(LCD_START_LINE2);
```

```
lcd_puts("third base?!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What am I putting on");
lcd_command(LCD_START_LINE2);
lcd_puts("third?!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You don't want who");
lcd_command(LCD_START_LINE2);
lcd_puts("on second?!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("I don’t know!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
// Both
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look, you got");
lcd_command(LCD_START_LINE2);
lcd_puts("outfield?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The left fielder’s");
lcd_command(LCD_START_LINE2);
```

```
lcd_puts("name?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I just thought I'd");
lcd_command(LCD_START_LINE2);
lcd_puts("ask you.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Then tell me who is");
lcd_command(LCD_START_LINE2);
lcd_puts("playing left field.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
```

```
lcd_puts("I’m not. Stay out of");
lcd_command(LCD_START_LINE2);
lcd_puts("the infield! I want");
lcd_command(LCD_START_LINE3);
lcd_puts("to know, what’s the");
lcd_command(LCD_START_LINE4);
lcd_puts("guy’s name in left");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("field?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I’m not asking who’s");
lcd_command(LCD_START_LINE2);
lcd_puts("on second.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I don’t know.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}
```

```
while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Both:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");
wait(hold);

//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("And left fielder's");
lcd_command(LCD_START_LINE2);
lcd_puts("name?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Because.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
wait(400);

for (z=1;z<ten;z++){
usart1_transmit(50);
```

```
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look, look, look,");
lcd_command(LCD_START_LINE2);
lcd_puts("you got a pitcher?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The pitcher's name?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("You don't want to");
lcd_command(LCD_START_LINE2);
lcd_puts("tell me today?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
```

```
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Well go ahead.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("What time?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("At what time");
lcd_command(LCD_START_LINE2);
lcd_puts("tomorrow are you");
lcd_command(LCD_START_LINE3);
lcd_puts("going to tell me");
lcd_command(LCD_START_LINE4);
lcd_puts("who's pitching?");
wait(hold);
```

```
for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I'll break your arm");
lcd_command(LCD_START_LINE2);
lcd_puts("you say who's on");
lcd_command(LCD_START_LINE3);
lcd_puts("first! I want to");
lcd_command(LCD_START_LINE4);
lcd_puts("know, what's the");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("pitcher's name?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I don't know!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
```

```
//Both:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Third base!");
wait(hold);

//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Got a catcher?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("The catcher's name.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Today? And");
lcd_command(LCD_START_LINE2);
lcd_puts("tomorrow's pitching?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}
```

```
while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("All we got is a");
lcd_command(LCD_START_LINE2);
lcd_puts("couple of days on");
lcd_command(LCD_START_LINE3);
lcd_puts("the team. You know,");
lcd_command(LCD_START_LINE4);
lcd_puts("I'm a catcher too.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I get behind the");
lcd_command(LCD_START_LINE2);
lcd_puts("plate, do some fancy");
lcd_command(LCD_START_LINE3);
lcd_puts("catching. Tomorrow's");
lcd_command(LCD_START_LINE4);
lcd_puts("pitching on my team");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("and a heavy hitter");
lcd_command(LCD_START_LINE2);
lcd_puts("gets up.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}
```

```
while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now, the heavy");
lcd_command(LCD_START_LINE2);
lcd_puts("hitter bunts the");
lcd_command(LCD_START_LINE3);
lcd_puts("ball. When he bunts");
lcd_command(LCD_START_LINE4);
lcd_puts("the ball, me being a");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("good catcher, I want");
lcd_command(LCD_START_LINE2);
lcd_puts("to throw the guy out");
lcd_command(LCD_START_LINE3);
lcd_puts("at first base. So I");
lcd_command(LCD_START_LINE4);
lcd_puts("pick up the ball,");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("and throw it to who?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I don't even know");
lcd_command(LCD_START_LINE2);
lcd_puts("what I'm talking");
lcd_command(LCD_START_LINE3);
lcd_puts("about!");
```

```
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Is throw the ball to");
lcd_command(LCD_START_LINE2);
lcd_puts("first base?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now who's got it?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Look, if I throw the");
lcd_command(LCD_START_LINE2);
lcd_puts("ball to first base,");
```

```
lcd_command(LCD_START_LINE3);
lcd_puts("somebodys got to get");
lcd_command(LCD_START_LINE4);
lcd_puts("it. Now who has it?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Who?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("So I pick up the");
```

```
lcd_command(LCD_START_LINE2);
lcd_puts("ball and throw it");
lcd_command(LCD_START_LINE3);
lcd_puts("to Naturally?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's what I said.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I throw the ball to");
```

```
lcd_command(LCD_START_LINE2);
lcd_puts("Naturally?");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("That's what I said!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I throw the ball to");
lcd_command(LCD_START_LINE2);
lcd_puts("who?");
```

```
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Now you ask me.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Naturally.");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Same as you!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
```

```
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Same as you! I throw");
lcd_command(LCD_START_LINE2);
lcd_puts("the ball to who.");
lcd_command(LCD_START_LINE3);
lcd_puts("Whoever it is drops");
lcd_command(LCD_START_LINE4);
lcd_puts("the ball, the guy");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("runs to second, who");
lcd_command(LCD_START_LINE2);
lcd_puts("picks up the ball,");
lcd_command(LCD_START_LINE3);
lcd_puts("throw's it to what,");
lcd_command(LCD_START_LINE4);
lcd_puts("what throw's it to I");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("don't know, I don't");
lcd_command(LCD_START_LINE2);
lcd_puts("know throw's it back");
lcd_command(LCD_START_LINE3);
lcd_puts("to tomorrow, triple");
lcd_command(LCD_START_LINE4);
lcd_puts("play!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
```

```
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("Another guy gets up,");
lcd_command(LCD_START_LINE2);
lcd_puts("and it’s a long fly");
lcd_command(LCD_START_LINE3);
lcd_puts("ball to because.");
lcd_command(LCD_START_LINE4);
lcd_puts("Why? I don’t know,");
wait(hold1);
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("he’s on third, and I");
lcd_command(LCD_START_LINE2);
lcd_puts("don’t give a darn!");
wait(hold);

for (z=1;z<ten;z++){
usart1_transmit(50);
wait(20);}

while(1){
int rc = usart1_receive();
if ((rc<125)&(rc>45)){
break;}}
//Costello:
lcd_clrscr();
lcd_command(LCD_START_LINE1);
lcd_puts("I said, I don’t give");
lcd_command(LCD_START_LINE2);
lcd_puts("a darn!");
wait(hold);

wait(20000);
} // end routine
```