Mapping and Navigational Control for an Autonomous Wheelchair

Senior Design Final Report

Dana Schultz

Kathleen Shea

April 26, 2012

# Table of Contents

[Abstract 3](#_Toc323213972)

[Introduction 4](#_Toc323213973)

[Background 4](#_Toc323213974)

[Purpose 4](#_Toc323213975)

[Project Goals 4](#_Toc323213976)

[Subsystems 5](#_Toc323213977)

[Wheelchair Motor Control 5](#_Toc323213978)

[Distance and Velocity Measurements 7](#_Toc323213979)

[Obstacle Detection 9](#_Toc323213980)

[User Interface 13](#_Toc323213981)

[Feature Map 16](#_Toc323213982)

[Navigation 18](#_Toc323213983)

[Overall System 20](#_Toc323213984)

[Appendix A: Code 21](#_Toc323213985)

[Demo.c 22](#_Toc323213986)

[Demo Encoder.c 37](#_Toc323213987)

[DemoTouchscreen.c 49](#_Toc323213988)

[DemoUltrasonic.c 61](#_Toc323213989)

[DemoURD.c 71](#_Toc323213990)

[Micro-SD Card Code Functions 83](#_Toc323213991)

[SystemDemo.c 128](#_Toc323213992)

[USART\_Test.c 146](#_Toc323213993)

[Appendix B: Parts 147](#_Toc323213994)

[Cost Breakdown 147](#_Toc323213995)

[Additional Parts 147](#_Toc323213996)

**Table of Figures**

[Figure 1 - Motor Control System - Generated Voltages 5](#_Toc323214190)

[Figure 2 - Function of Motor Control System 6](#_Toc323214191)

[Figure 3 - Motor Control System Interface 6](#_Toc323214192)

[Figure 4 - Determination of Direction of Motion 7](#_Toc323214193)

[Figure 5 - Mechanical Shaft Encoder Interface 8](#_Toc323214194)

[Figure 6 - Ultrasonic Sensor Locations 10](#_Toc323214195)

[Figure 7 - Sensor Average Responses 11](#_Toc323214196)

[Figure 8 - Ultrasonic Sensor Interface 12](#_Toc323214197)

[Figure 9 - 4-Wire Touchscreen Construction 13](#_Toc323214198)

[Figure 10 - ADS7843 Differential Reference Mode 14](#_Toc323214199)

[Figure 11 - LCD and Touchscreen Interface 15](#_Toc323214200)

[Figure 12 - Feature Map Cell Representations 16](#_Toc323214201)

[Figure 13 - SD Card and Microcontroller C Interface 17](#_Toc323214202)

[Figure 14 - Feature and Local Map Comparison 18](#_Toc323214203)

[Figure 15 - Sample Feature and Local Map 19](#_Toc323214204)

[Figure 16 - System Block Diagram 20](#_Toc323214205)

# 

# Abstract

Typical powered wheelchairs provide a means of transportation for many with limited mobility. However, the traditional joystick requires full mobility of the hand and wrist in order to be operated safely, leaving those without this dexterity no viable transportation option. Autonomous “smart” wheelchairs attempt to fill this hole.

A “smart” wheelchair is an autonomous device; capable of navigating through a person’s home or office with little user input. “Smart” wheelchairs are designed to work with a variety of interface options, providing those with limited dexterity alternative means to control their wheelchair.

The flexibility and complex design of autonomous wheelchairs have made those currently available expensive. Ongoing research has been aimed at designing a cheaper, alternative control system that could be easily attached to a powered wheelchair. The goal of this project is to determine methods for mapping and navigational control for the wheelchair. The control system acquires data from eighteen sensors and uses the data to navigate around a pre-programmed map. The map is stored on a micro SD card. The control system also provides user interface in the form of a touchscreen LCD. This designed system will be an easy to use and cost effective alternative to current autonomous “smart” wheelchair technology.

# Introduction

## Background

There has been ongoing research into the development of autonomous wheelchairs. Autonomous wheelchairs aid in the mobility for those unable to control a traditional joystick. Currently “smart” wheelchairs that are available are expensive. An alternative to a traditional autonomous wheelchair is needed for those who have limited dexterity and cannot afford an autonomous wheelchair that is currently on the market. Research into an autonomous wheelchair needs to be done to provide a cheaper alternative to what is on the market today.

## Purpose

Research has been ongoing at the University of Wyoming into an alternative to current “smart” wheelchairs. An efficient and cost effective system that can be attached to a standard power wheelchair will benefit those with limited dexterity.

A “smart” wheelchair that can navigate an environment with minimal user input is needed. There are several different interface options. Directional push buttons, switches, and sip and puff controllers are just a few examples. These leave those with limited dexterity with few options. Push buttons and switches can be hard for this type of person to handle.

With a system that can be attached to typical powered wheelchair those with limited dexterity will have an alternative. Making the system cost effective will allow those who cannot afford a “smart” wheelchair that is currently on the market with options.

## Project Goals

For this project a combination of hardware and software will be used to integrate a control system developed by Dr. Steven Barrett. A sensor system will be used for obstacle detection, mechanical shaft encoders for distance and velocity measurements. A feature map will be used as a preprogrammed map on a SD card. A local map will be updated onto the microcontroller itself. The user interface consists of a resistive touchscreen LCD.

# Subsystems

In order to effectively describe the designed system, it has been broken down into smaller subsystems. Each of these is described in detail below.

## Wheelchair Motor Control

In order to drive the wheelchair, our system will use the motor control system designed by Dr. Barrett. This system replicates the voltages sent from the joystick of the Quickie P300 to the wheelchair’s motors. As shown in Figure 1 below, the system replicates the two voltages, Vx and Vy, generated by the joystick for forward, reverse, left and right motion. The neutral value for Vx and Vy is 5.6 V. To drive the wheelchair forward, Vy is increased to 6.6 V. To drive the wheelchair in reverse, Vy is decreased to 4.6 V. Similarly, to turn left, Vx is decreased to 4.6 V and to turn right Vx is increased to 6.6 V.



#### Figure - Motor Control System - Generated Voltages

The motor control system will be used as a low level system. The navigation system will send digital signals to the motor control system, indicating the desired direction of motion and speed. The motor control system will then generate the required voltages. This is shown in Figure 2 on page 6. The direction signals are active low and only one will be activated in code at a time.



#### Figure - Function of Motor Control System

Communication between the navigational system and the motor control system will be handled by Microcontroller C. The navigational system is run at 3.3 V and the motor control system at 5 V, so the digital signals output by Microcontroller C will be level shifted and then connected to the motor control system. For this purpose, the selected level shifter was the MC14504BCPGOS-ND, a low power hex level shifter. The schematic for the interface between Microcontroller C, the level shifter, and the motor control system is shown in Figure 3 below.



#### Figure - Motor Control System Interface

## Distance and Velocity Measurements

Accurate measurements of distance and direction travelled are essential in the accuracy of the navigational system. Koyo TRD-S360 light duty incremental shaft encoders were selected for this purpose. The encoders are mechanically connected to the exterior of the left and right wheels on the wheelchair. The encoders output pulse trains with each pulse corresponding to one degree of rotation. The direction of motion can be determined using two signals offset by 90° provided by the encoder as illustrated in

Figure 4 below. When there is a falling edge in the Out A signal, the current value of Out B indicates the direction of motion. If Out B is high, the shaft is spinning clockwise and if Out B is low the shaft is spinning counterclockwise.



#### Figure - Determination of Direction of Motion

The encoders also provide noise minimization by outputting complement signals (A’ and B’) for each pulse train. Connecting these signals to a differential amplifier will reduce the noise created by the wheelchair’s motors. The selected differential amplifier is the INA2134PA-ND. This amplifier was selected for its low distortion, high CMRR, and fixed gain of 0 dB. The interface circuitry between the shaft encoders and differential amplifiers is shown in

Figure 5 below.



#### Figure - Mechanical Shaft Encoder Interface

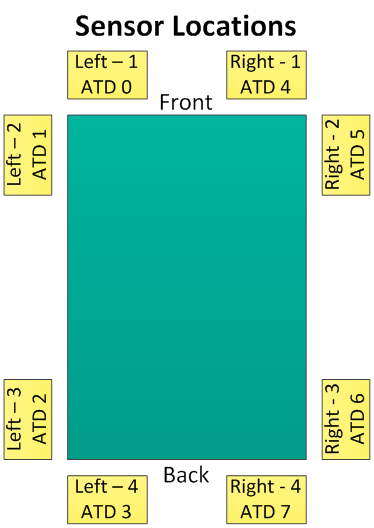
In order to ensure accurate determination of the distance travelled, the A signals for both the left wheel and right wheel are connected to external interrupts on the microcontroller with high precedence. Right A is connected to INT1 and Left A is connected to INT2. The microcontroller is programmed to trigger an interrupt when it senses a falling edge on both of these pins. The interrupt subroutine (ISR) then looks at the B signal for that wheel, determines the direction of motion, and increments a pulse counter. A separated interrupt is triggered every time Timer/Counter2 overflows. This ISR updates the position of the wheelchair based on the direction of motion and pulse count information determined by the other two ISR’s. The frequency with which the position is updated can be changed by modifying the prescaler for Timer/Counter2. Currently, the prescaler is set to 1024, resulting in an interrupt every 32.7 ms. More testing would need to be performed to determine the necessary interval between position updates.

To demonstrate the functionality of the encoder system a demo program was written. This program uses the information from the shaft encoders to track the wheelchair’s position and displays this information to the LCD. The wheelchair is represented by a small square on the screen. As the wheelchair drives forward, a green line is drawn on the screen relative to the speed and amount of motion in the forward direction. Reverse motion follows a similar procedure, except the drawn line is red. For left turns, a magenta square is drawn and for right turns a cyan square is drawn. The orientation of the wheelchair is also tracked by this system. The code for this demo can be found in Appendix A and is titled DemoEncoder.c.

## Obstacle Detection

Obstacle detection is a big part of an autonomous wheelchair. The sensors are need for safe navigation. The sensors can be used for navigation through doorways and docking with furniture as well as obstacles such as columns in a hallway. Our system uses eight ultrasonic sensors for obstacle detection. We are using the Ultrasonic Range Finder –XL-Maxsonar EZ3. A sensor is placed on each corner of each side as shown in

Figure 6 below.



#### Figure - Ultrasonic Sensor Locations

The sensors were characterized from 24 cm to 100 cm. The sensor does not send an accurate signal under 24 cm. To compensate for this we moved the sensors in on the wheelchair frame so that the accurate readings start at the edge of the wheelchair.

Figure 7 below shows the average responses from a standalone sensor and the two sensors mounted on the front of the wheelchair. After approximately 24 cm the plot shows how closely the voltage values follow one another.



#### Figure - Sensor Average Responses

The transmission pin from the ultrasonic sensor is input to the ADC pins on the ATmega1284p. This is shown in Figure 8 on the next page. The conversions from the sensor and the analog to digital converter are off by one. By adding one to the analog to digital converter result, the correct distance value is given. The sensor values have been tested and verified.



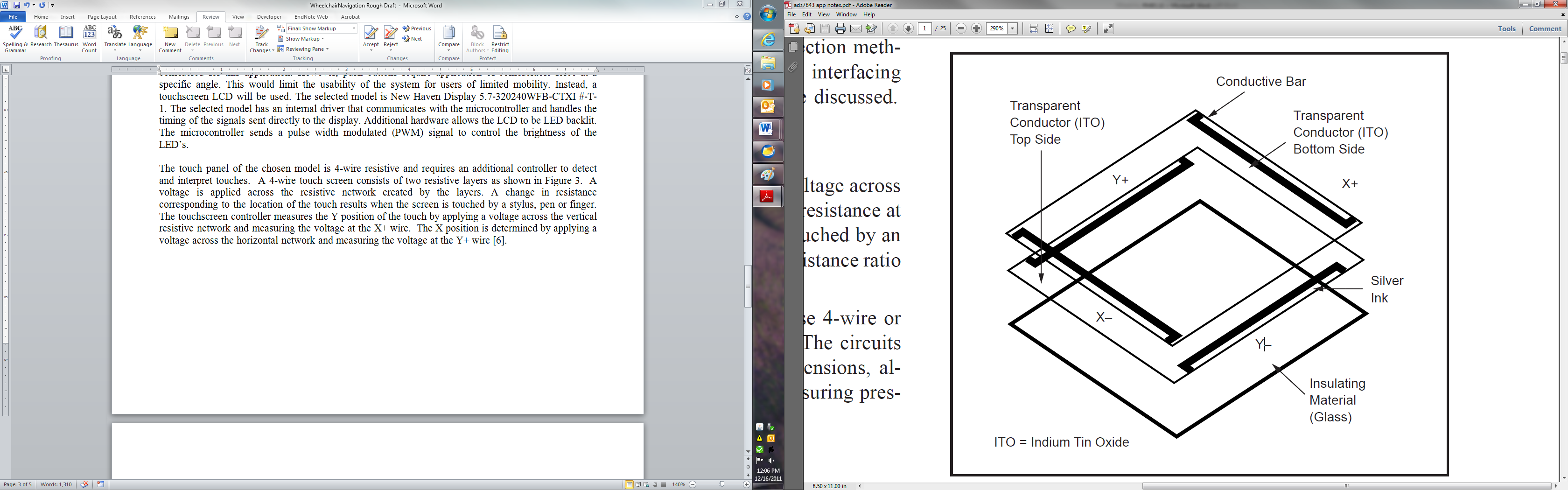
#### Figure - Ultrasonic Sensor Interface

The ultrasonic sensors used provide the information that is needed for obstacle detection. The values from the A-D converter are consistent with the distance in cm that the object is away from the sensor. The sensor conversions are controlled by a timed interrupt. The interrupt triggers when Timer/Counter0 overflows. The prescaler for this timer is set to 256, so an interrupt occurs every 8.19 ms. Each time the interrupt triggers, analog to digital conversion occurs for a single sensor. A counter is used to cycle through the sensors. The code for this subsystem can be found in Appendix A and is titled DemoUltrasonic.c.above

## User Interface

The user interface for this system requires user input as well as feedback to the user in the form of a display. Traditional push buttons with a graphic liquid crystal display (LCD) were considered for this application. However, push buttons require application of considerable force at a specific angle. This would limit the usability of the system for users of limited mobility. A traditional menu system would also introduce complexity for the user as they try to select a destination. As an alternative, a touchscreen LCD will be used. The selected model is New Haven Display 5.7‐320240WFB‐CTXI #‐T‐1. The selected model has an internal driver that communicates with the microcontroller and handles the timing of signals sent directly to the display. Additional hardware allows the LCD to be LED backlit. The microcontroller sends a pulse width modulated (PWM) signal to control the brightness of the LED’s.

The touch panel of the chosen model is 4-wire resistive and requires an additional controller to detect and interpret touches. A 4-wire touchscreen consists of two resistive layers as shown in Figure 9. A voltage is applied across the resistive network created by the layers. A change in resistance corresponding to the location of the touch results when the screen is touched by a stylus, pen or finger. A touchscreen controller measures the Y position of the touch by applying a voltage across the vertical resistive network and measuring the voltage at the X+ wire. The X position is determined by applying a voltage across the horizontal network and measuring the voltage at the Y+ wire. The touchscreen controller then sends digital coordinates for the touch to the navigational control system microcontroller via serial communication.

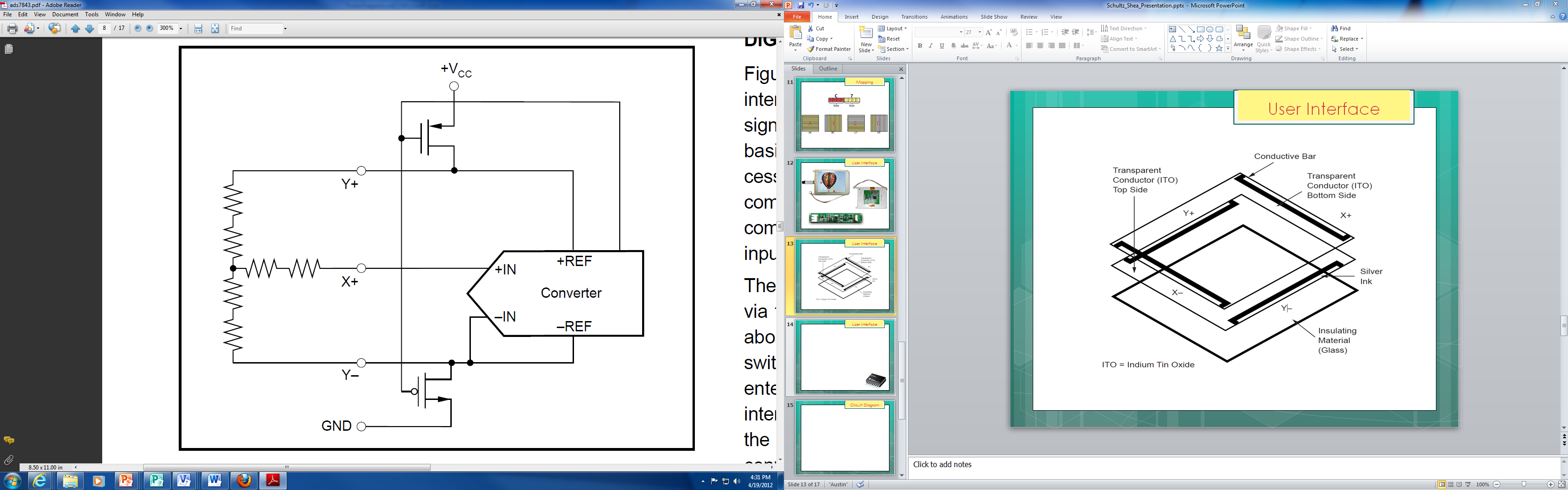


#### Figure - 4-Wire Touchscreen Construction

Source: *Touch Screen Controller Tips,* 1999, p. 1

The selected touchscreen controller is the ADS7843 made by Texas Instruments. One advantage of this touchscreen controller is that it provides a differential reference mode for the analog to digital conversion. In this mode, the positive and negative voltages on the touchscreen are connected to the reference voltages of the ATD converter. As shown in

Figure 10 below, when the y coordinate is being determined, the positive reference voltage is connected to Y+ and the negative reference voltage is connected to Y-. The voltage is then measured at X+. This mode helps to reduce the noise inherent in resistive touchscreens.



#### Figure - ADS7843 Differential Reference Mode

Source: *ADS7843 Datasheet,* 2000, p. 8

The selected touchscreen controller communicates with the microcontroller via SPI. When the user touches the LCD, the touchscreen controller triggers and interrupt on Microcontroller A via external interrupt INT0. The microcontroller then sends commands to the touchscreen controller, indicating which coordinate information it wants. The touchscreen controller performs an analog to digital conversion and sends the data back to the microcontroller. Since resistive touchscreens take time to settle, a delay is introduced in code between the triggering of the interrupt and the beginning of the communication with the touchscreen controller. The interface circuitry between the LCD, touchscreen controller, and microcontroller is shown in Figure 11 below.



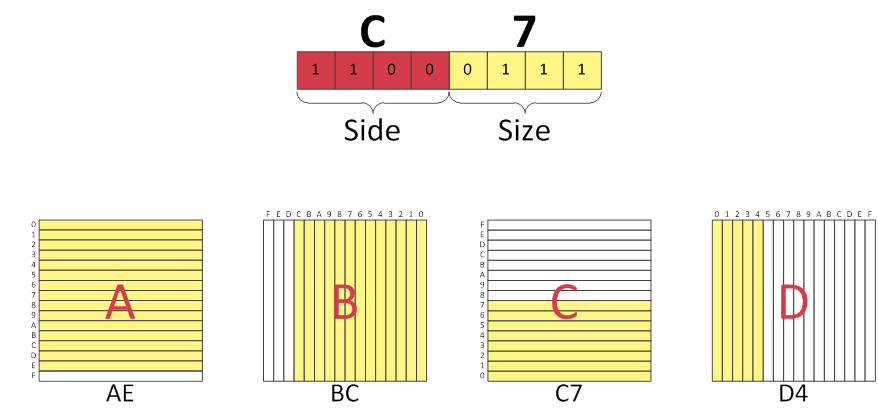
#### Figure - LCD and Touchscreen Interface

The touchscreen LCD will be used to provide all user interfacing for this system. The user will select their destination location by touching the appropriate location on a map displayed on the LCD. As the system is driving, the current position of the wheelchair will be displayed on a map on the LCD. All menu functions and options will also be handled through the touchscreen. Example code demonstrating the functionality of the LCD and touchscreen can be found in Appendix A. The LCD code is used in the demonstrations of all other subsystems, so the LCD code is located in many different code files. Code files of particular interest are: DemoTouchscreen.c and DemoURD.c.

## Feature Map

The feature map is a character array. Each cell of the array corresponds to a twelve inch square of floor space. The value of the square identifies what occupies the space. The most significant nibble indicates what side of the square the obstacle is protruding from and the least significant nibble represents how far the obstacle is protruding into the space.

Figure 12 shows a representation of this concept. For example, the byte 0xC7 indicates an obstacle protruding from the bottom of the square and filling one half of the twelve inch square.



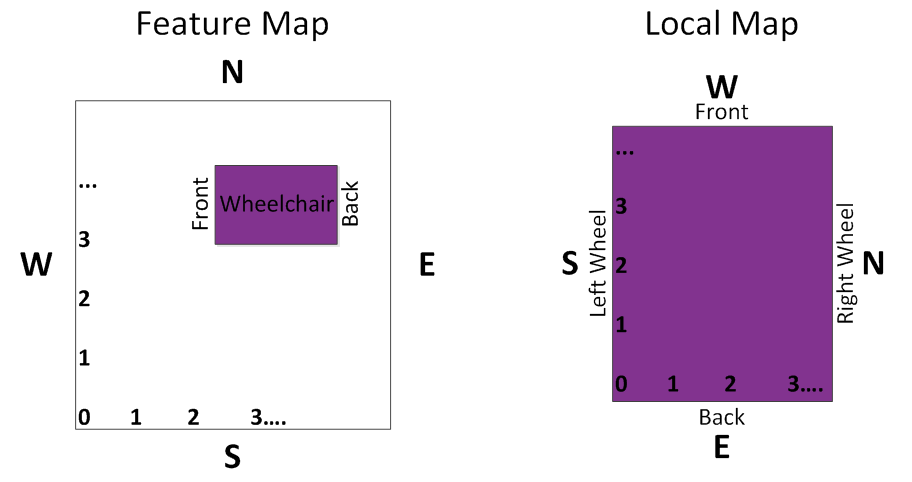
#### Figure - Feature Map Cell Representations

Due to the size of the feature map an external device is need for storage. A Micro-SD card can be used to store the feature map. The Micro-SD card will communicate with a second microcontroller, Microcontroller C, through SPI. The connections from the Micro-SD breakout board to Microcontroller C can be seen in Figure 13 on page 17. The code for the Micro-SD card has been written and can be found in Appendix A under the title SD\_Code.c. There are glitches in the code that need to be debugged.



#### Figure - SD Card and Microcontroller C Interface

To speed up processing during navigation, a local map is stored on the Microcontroller A. The local map will be oriented with the first row corresponding to the front of the wheelchair. The local map may need to be rotated when loading the data from the feature map. The rotation algorithm will also need to correct the first nibble of the individual cells of the map. The main feature map stored on the Micro-SD card is oriented so that the first row corresponds with North. The rotation of the local map from the feature map is seen in Figure 14 on the following page.



#### Figure - Feature and Local Map Comparison

## Navigation

The wheelchair is designed to be a “smart” wall follower. The system will drive six inches from the right wall of the hallway. When an obstacle on the right wall is encountered, the wheelchair shifts to the left to avoid the obstacle and then shifts right on the far side of the obstacle and resumes its wall following. When an unexpected obstacle is encountered in front of the wheelchair, it navigates around the obstacle and then uses the map and encoder information to help it resume wall following.

When a navigational decision must be made, the decision will be made based on the current coordinates of the wheelchair and the coordinates of the destination in relation to the feature map. For example, when the wheelchair encounters a hallway intersection at which it can turn left or right, it selects which direction to turn based upon its coordinates and the coordinates of its destination rather than turning right and continuing to follow the right wall.

To better explain the functionality of the navigational system, a few sample navigational decisions are outlined with the use of Figure 15 on the following page.



#### Figure - Sample Feature and Local Map

***Sample Navigational Decision***

Consider the wheelchair indicated by the green cells in Figure 15. This wheelchair is at the point where it needs to make a navigation decision. It has the option of turning either left or right. If the wheelchair’s destination was in the cell indicated by the purple cell, then it would turn right. In this case, the destination’s x-coordinate would be greater than the wheelchair’s, indicating a right turn. If the wheelchair’s destination was the orange cell, then the destination’s x-coordinate would be greater than the wheelchair’s, indicating a left turn.

***Sample Local Map Rotation***

Consider the wheelchair indicated by the blue cells in Figure 15. This wheelchair is headed to the left, toward the orange cell. The local map of this wheelchair is shown in the right of the figure. As the local map was rotated, the individual cell values were changed to reflect this rotation. In this case, A’s on the feature map became B’s in the local map and C’s on the feature map became D’s in the local map.

# Overall System

The final designed system is described by the block diagram shown in

Figure 16. Microcontroller A is indicated by the purple block. The motor control subsystem is indicated by the red block. The distance and velocity measuring subsystem is indicated by the mechanical shaft encoders in the green block. Obstacle detection is indicated by the ultrasonic sensors in the blue block. The user interface system is indicated by the yellow blocks. The storage for the mapping and navigational system is indicated by the teal blocks.



#### Figure - System Block Diagram

The final system consists of five interrupt driven subsystems. The highest priority interrupt, INT0, is connected to the touchscreen controller. The mechanical shaft encoder A channels are connected to INT1 and INT2 and comprise the next level interrupts. The B channels are connected to digital input pins and their values are read when the A channels trigger an interrupt. The position of the wheelchair is updated by a timed interrupt with the fourth level of priority. Information from the ultrasonic sensors is read by another timed interrupt with the fifth level of priority.

# Appendix A: Code

[Demo.c 22](#_Toc323213986)

[Demo Encoder.c 37](#_Toc323213987)

[DemoTouchscreen.c 49](#_Toc323213988)

[DemoUltrasonic.c 61](#_Toc323213989)

[DemoURD.c 71](#_Toc323213990)

[Micro-SD Card Code Functions 83](#_Toc323213991)

[SystemDemo.c 128](#_Toc323213992)

[USART\_Test.c 146](#_Toc323213993)

## Demo.c

//Demo Ultrasonic and Encoder

//Authors: Dana Schultz

//Created: April 12, 2012

//Revised: April 13, 2012

//Version: 2

//Uses Atmel ATmega1284 microcontroller to demonstrate the shaft encoders

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

void VariableRectangle\_LCD(unsigned int, unsigned int, unsigned int, unsigned int,

unsigned char, unsigned char, unsigned char);

void ColorGrid(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Encoder(void);

void Initialize\_EncoderPorts(void);

void Initialize\_EncoderInterrupts(void);

void RightWheel\_ISR(void);

void LeftWheel\_ISR(void);

void UpdatePosition\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void);

void ReadUltrasonic\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

//INT1

#pragma interrupt\_handler RightWheel\_ISR:3

//INT2

#pragma interrupt\_handler LeftWheel\_ISR:4

//Timer/Counter2 Overflow

#pragma interrupt\_handler UpdatePosition\_ISR:12

//Timer/Counter0 Overflow

#pragma interrupt\_handler ReadUltrasonic\_ISR:19

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

char numTouch;

//Encoder Variables

//Bottom left corner (SW corner) of the map equates to position (0,0)

unsigned int X\_position;

unsigned int Y\_position;

//Counters for encoder pulses

char pulses\_LW;

char pulses\_RW;

char lastpulses\_LW;

char lastpulses\_RW;

//Direction variables

char direction\_LW;

char direction\_RW;

char orientation;

//Ultrasonic Sensor Data Variables

unsigned int Left1, Left2, Left3, Left4;

unsigned int Right1, Right2, Right3, Right4;

char SensorCounter;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Keyword Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Wheel rotation direction definitions

#define NOMOTION 0

#define CLOCKWISE 1

#define COUNTERCLOCKWISE 2

//Left Wheel - CLOCKWISE = reverse motion, COUNTERCLOCKWISE = forward motion

//Right Wheel - CLOCKWISE = forward motion, COUNTERCLOCKWISE = reverse motion

//Wheelchair orientation definitions

#define NORTH 0

#define SOUTH 1

#define EAST 2

#define WEST 3

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

Initialize\_Encoder();

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0x00,0x33,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Writes rectangle of selected area and color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void VariableRectangle\_LCD(unsigned int startX, unsigned int endX, unsigned int startY,

unsigned int endY, unsigned char red, unsigned char green,

unsigned char blue)

{

unsigned int sizeX = endX - startX;

unsigned int sizeY = endY - startY;

unsigned int x,y;

WindowSet\_LCD(startX, endX, startY, endY);

WriteCommand\_LCD(0x2c);

for(y=0;y<=sizeY;y++)

{

for(x=0;x<=sizeX;x++)

{

SendPixel\_LCD(red,green,blue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ColorGrid(void)

{

unsigned int x,y;

WindowSet\_LCD(0,319,0,239);

WriteCommand\_LCD(0x2c);

//Screen fills right to left, bottom to top

//Basic Colors

for(x=0;x<80;x++)

{

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0x00,0x00); //Red

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0xff,0x00); //Green

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0x00,0xff); //Blue

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0x00,0x00); //Black

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0xff,0xff); //White

}

}

//Red Variations

for(x=0;x<80;x++)

{

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0xff,0xff); //Cyan

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0x00,0xff); //Magenta

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0xff,0x00); //Yellow

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0x66,0xcc); //Hot Pink

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0xff,0x33); //Spring Green

}

}

//Green Variations

for(x=0;x<80;x++)

{

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0xcc,0x66); //Blue/Green

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0xff,0xcc,0x66); //Light Orange

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x00,0x33,0x66); //Navy

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x66,0x00,0x00); //Dark Brown

}

for(y=0;y<64;y++)

{

SendPixel\_LCD(0x33,0x66,0x00); //Evergreen

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

numTouch = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK = 0x00; //disable INT0

EICRA = 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

numTouch++;

Monochrome\_LCD(0x00, 0x00, 0x00); //Black screen

Touch\_X = 0;

Touch\_Y = 0;

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X += GetX();

Touch\_Y += GetY();

}

//Average 8 samples

Touch\_X = Touch\_X >> 3;

Touch\_Y = Touch\_Y >> 3;

EIFR = 0x01; //Clear interrupt flag

if(numTouch == 3)

{

Initialize\_Ultrasonic(); //Turns on Ultrasonic System

TIMSK2 = 0x00; //Turns encoder timed interrupt off

}

else if(numTouch == 6)

{

asm("CLI"); //Global interrupt disable

TIMSK0 = 0x00;

ColorGrid();

}

else if(numTouch > 3)

{

Monochrome\_LCD(0x66, 0x00, 0xcc);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Encoder(void)

{

X\_position = 157;

Y\_position = 117;

orientation = NORTH;

pulses\_RW = 0;

pulses\_LW = 0;

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

Initialize\_EncoderPorts();

Monochrome\_LCD(0x00, 0x00, 0x00); //Black background

VariableRectangle\_LCD(X\_position, X\_position+6, Y\_position, Y\_position+6, 0xff, 0xff, 0x00);

Initialize\_EncoderInterrupts();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Ports Associated with Encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_EncoderPorts(void)

{

PORTD &= 0xe7; //Sets PORTD[3:4] for input - Right Wheel

PORTB &= 0xf3; //Sets PORTB[2:3] for input - Left Wheel

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Encoder Interrupts INT1 and INT2 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_EncoderInterrupts(void)

{

//Initialize INT1 and INT2

EIMSK &= 0xf9; //disable INT1 and INT2

EICRA &= 0xeb; //set INT1 and INT2 for falling edge trigger

EICRA |= 0x28;

EIMSK |= 0x06; //enable INT1 and INT2

//Initialize Timer/Counter0 Overflow Interrupt

TCCR2B = 0x04; //Prescaler of 1/64 - will result in interrupt every 2.05 ms

TIMSK2 = 0x01; //Enable Timer/Counter2 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Right Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RightWheel\_ISR(void)

{

pulses\_RW++; //Increment Pulse Counter

//Determine direction of motion

if((PIND & 0x10) == 1)

{

direction\_RW = COUNTERCLOCKWISE;

}

else

{

direction\_RW = CLOCKWISE;

}

EIFR = 0x02; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Left Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LeftWheel\_ISR(void)

{

pulses\_LW++; //Increment Pulse Counter

//Determine direction of motion

if((PINB & 0x08) == 1)

{

direction\_RW = COUNTERCLOCKWISE;

}

else

{

direction\_RW = CLOCKWISE;

}

EIFR = 0x04; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timed ISR - occurs every 2.05 ms \*/

/\*Uses directions and pulses from encoders to determine wheelchair position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UpdatePosition\_ISR(void)

{

char datred = 0x00;

char datgreen = 0x00;

int pulses;

if((pulses\_RW == lastpulses\_RW) || (pulses\_LW == lastpulses\_LW))

{

//Wheelchair is not moving

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

//Yellow for no motion

datred = 0xff;

datgreen = 0xff;

}

else if((pulses\_RW < 60) && (pulses\_LW < 60))

{

//Not enough pulses to update position

}

else if((direction\_LW == CLOCKWISE) && (direction\_RW == CLOCKWISE))

{

//Turning Left - change of orientation will be handled in code for turning

}

else if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == COUNTERCLOCKWISE))

{

//Turning Right - change of orientation will be handled in code for turning

}

else

{

//Check for greater number of pulses

if(pulses\_LW > pulses\_RW)

{

pulses = pulses\_LW;

}

else

{

pulses = pulses\_RW;

}

//Update pulse counters

pulses\_RW = 0;

pulses\_LW = 0;

//Moving forward

if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == CLOCKWISE))

{

if(orientation == NORTH)

{

Y\_position++;

}

else if(orientation == SOUTH)

{

Y\_position--;

}

else if(orientation == EAST)

{

X\_position++;

}

else

{

X\_position--;

}

datgreen = 0xff; //Green for moving forward

datred = 0x00;

}

else //Moving in reverse

{

if(orientation == NORTH)

{

Y\_position--;

}

else if(orientation == SOUTH)

{

Y\_position++;

}

else if(orientation == EAST)

{

X\_position--;

}

else

{

X\_position++;

}

datred = 0xff; //Red for moving backwards

datgreen = 0x00;

}

}

VariableRectangle\_LCD(X\_position, X\_position+6, Y\_position, Y\_position+6, datred, datgreen, 0x00);

//Store current pulse counts

lastpulses\_LW = pulses\_LW;

lastpulses\_RW = pulses\_RW;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Sensor Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with ultrasonic sensors \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void)

{

//Initialize PortA for input

DDRA = 0x00;

//Initialize Variables

SensorCounter = 1;

Left1 = 0;

Left2 = 0;

Left3 = 0;

Left4 = 0;

Right1 = 0;

Right2 = 0;

Right3 = 0;

Right4 = 0;

//Demo - draw wheelchair

Monochrome\_LCD(0x00, 0x00, 0x00); //black background

//Draw white rectangle 80X140 pixels in center of screen

VariableRectangle\_LCD(120, 200, 50, 190, 0xff, 0xff, 0xff);

//Initialize ADC

ADMUX = 0x00; //Select Channel 0 for dummy conversion

ADCSRA = 0xc3; //Enable ADC, single conversion,

//prescaler of 8 for accurate results

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

//Initialize Timer/Counter0 Interrupt

TCCR0B = 0x04; //Prescaler of 1/256 - will result in interrupt every 8.19 ms

TIMSK0 = 0x01; //Enable Timer/Counter0 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timer/Counter0 ISR - occurs every 8.19 ms \*/

/\*Initiates ATD Conversion and stores results \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadUltrasonic\_ISR(void)

{

//Temporary data storage variables

unsigned int datalow = 0;

unsigned int datahigh = 0;

unsigned int distance = 0;

char datred = 0;

char datgreen = 0;

char datblue = 0;

//Perform ATD Conversion

ADCSRA |= 0x43; //Start single conversion, prescaler = 8

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

datalow = ADCL; //Read low data register first

//datahigh = ((unsigned int)(ADCH << 8)); //Read high data register

datahigh = ADCH;

datahigh = datahigh << 8;

distance = datalow | datahigh;

//Demo - assign color based on ADC value

if((25<= distance) && (distance < 30))

{

//Red sensor

datred = 0xff;

}

else if((30<= distance) && (distance < 35))

{

//Cyan sensor

datgreen = 0xff;

datblue = 0xff;

}

else if((35<= distance) && (distance < 40))

{

//Yellow sensor

datred = 0xff;

datgreen = 0xff;

}

else if((40<= distance) && (distance < 45))

{

//Green Sensor

datgreen = 0xff;

}

else if((45<= distance) && (distance < 50))

{

//Blue Sensor

datblue = 0xff;

}

else if((50<= distance) && (distance < 55))

{

//Magenta Sensor

datred = 0xff;

datblue = 0xff;

}

else

{

//White Sensor

datred = 0xff;

datgreen = 0xff;

datblue = 0xff;

}

switch(SensorCounter)

{

case 1:

Left1 = distance; //Store data for sensor Left1

//Draw Sensor Left1

VariableRectangle\_LCD(120, 150, 10, 40, datred, datgreen, datblue);

ADMUX = 0x01; //Select Channel 1 for next conversion

break;

case 2:

Left2 = distance; //Store data for sensor Left2

//Draw Sensor Left2

VariableRectangle\_LCD(80, 110, 50, 80, datred, datgreen, datblue);

ADMUX = 0x02; //Select Channel 2 for next conversion

break;

case 3:

Left3 = distance; //Store data for sensor Left3

//Draw Sensor Left3

VariableRectangle\_LCD(80, 110, 160, 190, datred, datgreen, datblue);

ADMUX = 0x03; //Select Channel 3 for next conversion

break;

case 4:

Left4 = distance; //Store data for sensor Left4

//Draw Sensor Left4

VariableRectangle\_LCD(120, 150, 200, 230, datred, datgreen, datblue);

ADMUX = 0x04; //Select Channel 4 for next conversion

break;

case 5:

Right1 = distance; //Store data for sensor Right1

//Draw Sensor Right1

VariableRectangle\_LCD(170, 200, 10, 40, datred, datgreen, datblue);

ADMUX = 0x05; //Select Channel 5 for next conversion

break;

case 6:

Right2 = distance; //Store data for sensor Right2

//Draw Sensor Right2

VariableRectangle\_LCD(210, 240, 50, 80, datred, datgreen, datblue);

ADMUX = 0x06; //Select Channel 6 for next conversion

break;

case 7:

Right3 = distance; //Store data for sensor Right3

//Draw Sensor Right3

VariableRectangle\_LCD(210, 240, 160, 190, datred, datgreen, datblue);

ADMUX = 0x07; //Select Channel 7 for next conversion

break;

default:

Right4 = distance; //Store data for sensor Right4

//Draw Sensor Right4

VariableRectangle\_LCD(170, 200, 200, 230, datred, datgreen, datblue);

ADMUX = 0x00; //Select Channel 0 for next conversion

SensorCounter = 0;

break;

}

//Draw white rectangle 80X140 pixels in center of screen

VariableRectangle\_LCD(120, 200, 50, 190, 0xff, 0xff, 0xff);

SensorCounter++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

## Demo Encoder.c

//Demo Ultrasonic

//Authors: Dana Schultz

//Created: April 12, 2012

//Revised: April 14, 2012

//Version: 3

//Uses Atmel ATmega1284 microcontroller to demonstrate the shaft encoders

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

void VariableRectangle\_LCD(unsigned int, unsigned int, unsigned int, unsigned int,

unsigned char, unsigned char, unsigned char);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Encoder(void);

void Initialize\_EncoderPorts(void);

void Initialize\_EncoderInterrupts(void);

void RightWheel\_ISR(void);

void LeftWheel\_ISR(void);

void UpdatePosition\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

//INT1

#pragma interrupt\_handler RightWheel\_ISR:3

//INT2

#pragma interrupt\_handler LeftWheel\_ISR:4

//Timer/Counter2 Overflow

#pragma interrupt\_handler UpdatePosition\_ISR:12

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

//Encoder Variables

//Bottom left corner (SW corner) of the map equates to position (0,0)

unsigned int X\_position;

unsigned int Y\_position;

//Counters for encoder pulses

unsigned int pulses\_LW;

unsigned int pulses\_RW;

char lastpulses\_LW;

char lastpulses\_RW;

//Direction variables

char direction\_LW;

char direction\_RW;

char orientation;

char pixelR;

char pixelG;

char pixelB;

unsigned int turnCountR;

unsigned int turnCountL;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Keyword Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Wheel rotation direction definitions

#define NOMOTION 0

#define CLOCKWISE 1

#define COUNTERCLOCKWISE 2

//Left Wheel - CLOCKWISE = reverse motion, COUNTERCLOCKWISE = forward motion

//Right Wheel - CLOCKWISE = forward motion, COUNTERCLOCKWISE = reverse motion

//Wheelchair orientation definitions

#define NORTH 0

#define SOUTH 1

#define EAST 2

#define WEST 3

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

Initialize\_Encoder();

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0x00,0x33,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Writes rectangle of selected area and color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void VariableRectangle\_LCD(unsigned int startX, unsigned int endX, unsigned int startY,

unsigned int endY, unsigned char red, unsigned char green,

unsigned char blue)

{

unsigned int sizeX = endX - startX;

unsigned int sizeY = endY - startY;

unsigned int x,y;

WindowSet\_LCD(startX, endX, startY, endY);

WriteCommand\_LCD(0x2c);

for(y=0;y<=sizeY;y++)

{

for(x=0;x<=sizeX;x++)

{

SendPixel\_LCD(red,green,blue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK &= 0xfe; //disable INT0

EICRA |= 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

Monochrome\_LCD(0x00, 0x00, 0x00); //Black screen

Touch\_X = 0;

Touch\_Y = 0;

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X += GetX();

Touch\_Y += GetY();

}

//Average 8 samples

Touch\_X = Touch\_X >> 3;

Touch\_Y = Touch\_Y >> 3;

EIFR = 0x01; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Encoder(void)

{

X\_position = 157;

Y\_position = 123;

orientation = NORTH;

pulses\_RW = 0;

pulses\_LW = 0;

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

turnCountL = 0;

turnCountR = 0;

pixelR = 0x00;

pixelG = 0x00;

pixelB = 0x00;

Initialize\_EncoderPorts();

Monochrome\_LCD(0x00, 0x00, 0x00); //Black background

Initialize\_EncoderInterrupts();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Ports Associated with Encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_EncoderPorts(void)

{

DDRD &= 0xe7; //Sets PORTD[3:4] for input - Right Wheel

PORTD &= ~0x10; //Disable pullup resistor PD4

DDRB &= 0xf3; //Sets PORTB[2:3] for input - Left Wheel

PORTB &= ~0x08; //Disable pullup resistor PB3

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Encoder Interrupts INT1 and INT2 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_EncoderInterrupts(void)

{

//Initialize INT1 and INT2

EIMSK &= 0xf9; //disable INT1 and INT2

EICRA &= 0xeb; //set INT1 and INT2 for falling edge trigger

EICRA |= 0x28;

EIMSK |= 0x06; //enable INT1 and INT2

//Initialize Timer/Counter0 Overflow Interrupt

TCCR2B = 0x06; //Prescaler of 1/1024

TIMSK2 = 0x01; //Enable Timer/Counter2 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Right Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void RightWheel\_ISR(void)

{

pulses\_RW++; //Increment Pulse Counter

//Determine direction of motion

if((PIND & 0x10) == 0x10)

{

direction\_RW = COUNTERCLOCKWISE;

VariableRectangle\_LCD(312,319,0,8,0xff,0x00,0x00); //red

}

else

{

direction\_RW = CLOCKWISE;

VariableRectangle\_LCD(312,319,0,8,0x00,0xff,0x00); //green

}

EIFR = 0x02; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Left Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LeftWheel\_ISR(void)

{

pulses\_LW++; //Increment Pulse Counter

//Determine direction of motion

if((PINB & 0x08) == 0x08)

{

direction\_LW = COUNTERCLOCKWISE;

VariableRectangle\_LCD(0,8,0,8,0xff,0x00,0x00);

}

else

{

direction\_LW = CLOCKWISE;

VariableRectangle\_LCD(0,8,0,8,0x00,0xff,0x00);

}

EIFR = 0x04; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timed ISR - occurs every 2.05 ms \*/

/\*Uses directions and pulses from encoders to determine wheelchair position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UpdatePosition\_ISR(void)

{

int pulses;

if((pulses\_RW == lastpulses\_RW) || (pulses\_LW == lastpulses\_LW))

{

//Wheelchair is not moving

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

//Yellow for no motion

pixelR = 0xff;

pixelG = 0xff;

pixelB = 0x00;

pulses\_RW = 0;

pulses\_LW = 0;

turnCountL = 0;

turnCountR = 0;

}

else if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == CLOCKWISE))

{

//Moving forward

if((pulses\_RW > 60) || (pulses\_LW > 60))

{

if(orientation == NORTH)

{

Y\_position += 6;

}

else if(orientation == SOUTH)

{

Y\_position -= 6;

}

else if(orientation == EAST)

{

X\_position += 6;

}

else

{

X\_position -= 6;

}

//Reset pulse counters

pulses\_RW = 0;

pulses\_LW = 0;

turnCountR = 0;

turnCountL = 0;

}

pixelR = 0x00; //Green for moving forward

pixelG = 0xff;

pixelB = 0x00;

}

else if((direction\_LW == CLOCKWISE) && (direction\_RW == COUNTERCLOCKWISE))

{

//Moving in reverse

if((pulses\_RW > 60) || (pulses\_LW > 60))

{

if(orientation == NORTH)

{

Y\_position -= 6;

}

else if(orientation == SOUTH)

{

Y\_position += 6;

}

else if(orientation == EAST)

{

X\_position -= 6;

}

else

{

X\_position += 6;

}

pulses\_RW = 0;

pulses\_LW = 0;

turnCountR = 0;

turnCountL = 0;

}

pixelR = 0xff; //Red for moving backwards

pixelG = 0x00;

pixelB = 0x00;

}

else if((direction\_LW == CLOCKWISE) && (direction\_RW == CLOCKWISE))

{

//Turning Left

turnCountL++;

if (turnCountL > 60)

{

//Turned Left

turnCountL = 0;

turnCountR = 0;

pulses\_RW = 0;

pulses\_LW = 0;

//Magenta for turning left

pixelR = 0xff;

pixelG = 0x00;

pixelB = 0xff;

VariableRectangle\_LCD(X\_position, X\_position+6, 240-Y\_position, 246-Y\_position, pixelR, pixelG, pixelB);

//Update Orientation

if(orientation == NORTH)

{

orientation = WEST;

X\_position -= 6;

}

else if(orientation == WEST)

{

orientation = SOUTH;

Y\_position -= 6;

}

else if(orientation == SOUTH)

{

orientation = EAST;

X\_position += 6;

}

else

{

orientation = NORTH;

Y\_position += 6;

}

VariableRectangle\_LCD(X\_position, X\_position+6, 240-Y\_position, 246-Y\_position, pixelR, pixelG, pixelB);

}

}

else if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == COUNTERCLOCKWISE))

{

//Turning Right

turnCountR++;

if (turnCountR > 60)

{

//Turned Left

turnCountL = 0;

turnCountR = 0;

pulses\_RW = 0;

pulses\_LW = 0;

//Cyan for turning left

pixelR = 0x00;

pixelG = 0xff;

pixelB = 0xff;

VariableRectangle\_LCD(X\_position, X\_position+6, 240-Y\_position, 246-Y\_position, pixelR, pixelG, pixelB);

//Update Orientation

if(orientation == NORTH)

{

orientation = EAST;

X\_position += 6;

}

else if(orientation == EAST)

{

orientation = SOUTH;

Y\_position -= 6;

}

else if(orientation == SOUTH)

{

orientation = WEST;

X\_position -= 6;

}

else

{

orientation = NORTH;

Y\_position += 6;

}

VariableRectangle\_LCD(X\_position, X\_position+6, 240-Y\_position, 246-Y\_position, pixelR, pixelG, pixelB);

}

}

VariableRectangle\_LCD(X\_position, X\_position+6, 240-Y\_position, 246-Y\_position, pixelR, pixelG, pixelB);

//Store current pulse counts

lastpulses\_LW = pulses\_LW;

lastpulses\_RW = pulses\_RW;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

## DemoTouchscreen.c

//Demo Touchscreen

//Authors: Dana Schultz

//Created: April 16, 2012

//Revised: April 20, 2012

//Version: 3

//Uses Atmel ATmega1284 microcontroller to demonstrate the touchscreen

//Displays four quadrants, color of each changes to white when touched and

//then changes back

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

void Square\_LCD(unsigned char);

void ColorGrid(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

ColorGrid();

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0xff,0xff,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Displays square in top right corner of specified color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Square\_LCD(unsigned char Quad)

{

unsigned int x,y;

if(Quad == 1)

{

WindowSet\_LCD(0,159,0,119);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 2)

{

WindowSet\_LCD(160,319,0,119);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 3)

{

WindowSet\_LCD(0,159,120,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 4)

{

WindowSet\_LCD(160,319,120,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else

{

WindowSet\_LCD(0,319,0,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff, 0x00, 0xff); //Magenta

}

for(x=0;x<160;x++)

{

SendPixel\_LCD(0x00, 0xff, 0xff); //Cyan

}

}

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0x00, 0xff, 0x33); //Spring Green

}

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff, 0x33, 0x66); //Coral

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Displays grid of different colors on LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ColorGrid(void)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

//Screen fills right to left, bottom to top

//Basic Colors

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x00,0x00); //Red

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0xff,0x00); //Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x00,0xff); //Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0xff,0x00); //Yellow

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0xff,0xff); //Cyan

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x00,0xff); //Magenta

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x00,0x00); //Black

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0xff,0xff); //White

}

}

//Red Variations

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x33,0x66); //Coral

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x33,0xcc); //Medium Pink

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0xcc,0x33); //Golden Orange

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0xcc,0x66); //Light Orange

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x66,0x33); //Deep Orange

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x66,0xcc); //Hot Pink

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x66,0x00); //Orange

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xff,0x00,0x33); //Bright Orange Red

}

}

//Green Variations

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0xff,0x66); //Sea Foam Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0xff,0xcc); //Aquamarine

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0xff,0x33); //Yellow green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0xff,0x33); //Lime Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0xff,0xcc); //Easter Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0xff,0x66); //Light Yellow Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0xff,0x00); //Grass Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0xff,0x33); //Spring Green

}

}

//Blue Variations

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0xcc,0xff); //Sky Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0x33,0xff); //Med Purple

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0x66,0xff); //Cerulean Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x33,0xff); //Blue-purple

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0xcc,0xff); //Robin's Egg Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0x66,0xff); //Violet

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x66,0xff); //Soft Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x33,0xff); //Medium Blue

}

}

//Darker Colors

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x80,0x00,0x80); //Purple

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x00,0x00); //Dark Brown

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0x66,0x00); //Evergreen

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x33,0x00); //Light Brown

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x33,0x66); //Navy

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x66,0x33); //Sage Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x33,0x00,0x66); //Cobalt

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x00,0x33); //Dark Berry

}

}

for(x=0;x<40;x++)

{

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x66,0xcc); //Nice Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0xcc,0x66); //Blue/Green - Very Nice

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0x66,0x00); //Burnt Orange

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0xcc,0x00); //Rich Green

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x00,0xcc); //Purple Blue

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0xcc,0x00,0x66); //Magenta

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x66,0x00,0x66); //Berry

}

for(y=0;y<40;y++)

{

SendPixel\_LCD(0x00,0x66,0x66); //Teal

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK = 0x00; //disable INT0

EICRA = 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

Touch\_X = 0;

Touch\_Y = 0;

Delay\_ms(300);

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X = GetX();

Touch\_Y = GetY();

}

if((Touch\_X < 165) && (Touch\_Y < 165))

{

Square\_LCD(3);

}

else if((Touch\_X > 165) && (Touch\_Y < 165))

{

Square\_LCD(4);

}

else if((Touch\_X > 165) && (Touch\_Y > 165))

{

Square\_LCD(2);

}

else if((Touch\_X < 165) && (Touch\_Y > 165))

{

Square\_LCD(1);

}

else

{

Monochrome\_LCD(0xff,0xff,0xff);

}

Delay\_ms(750);

Square\_LCD(0);

EIFR = 0x01; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

## DemoUltrasonic.c

//Demo Ultrasonic

//Authors: Dana Schultz

//Created: April 12, 2012

//Revised: April 13, 2012

//Version: 2

//Uses Atmel ATmega1284 microcontroller to demonstrate the ultrasonic sensors

//Displays wheelchair with eight sensors around it

//Colors of the sensors change with respect to distance of sensed object

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

void VariableRectangle\_LCD(unsigned int, unsigned int, unsigned int, unsigned int,

unsigned char, unsigned char, unsigned char);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void);

void ReadUltrasonic\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

//Timer/Counter0 Overflow

#pragma interrupt\_handler ReadUltrasonic\_ISR:19

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

//Ultrasonic Sensor Data Variables

unsigned int Left1, Left2, Left3, Left4;

unsigned int Right1, Right2, Right3, Right4;

char SensorCounter;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

Initialize\_Ultrasonic();

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0x00,0x33,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Writes rectangle of selected area and color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void VariableRectangle\_LCD(unsigned int startX, unsigned int endX, unsigned int startY,

unsigned int endY, unsigned char red, unsigned char green,

unsigned char blue)

{

unsigned int sizeX = endX - startX;

unsigned int sizeY = endY - startY;

unsigned int x,y;

WindowSet\_LCD(startX, endX, startY, endY);

WriteCommand\_LCD(0x2c);

for(y=0;y<=sizeY;y++)

{

for(x=0;x<=sizeX;x++)

{

SendPixel\_LCD(red,green,blue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK = 0x00; //disable INT0

EICRA = 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

Monochrome\_LCD(0x00, 0x00, 0x00);

Touch\_X = 0;

Touch\_Y = 0;

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X += GetX();

Touch\_Y += GetY();

}

//Average 8 samples

Touch\_X = Touch\_X >> 3;

Touch\_Y = Touch\_Y >> 3;

EIFR = 0x01; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Sensor Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with ultrasonic sensors \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void)

{

//Initialize PortA for input

DDRA = 0x00;

//Initialize Variables

SensorCounter = 1;

Left1 = 0;

Left2 = 0;

Left3 = 0;

Left4 = 0;

Right1 = 0;

Right2 = 0;

Right3 = 0;

Right4 = 0;

//Demo - draw wheelchair

Monochrome\_LCD(0x00, 0x00, 0x00); //black background

//Draw white rectangle 80X140 pixels in center of screen

VariableRectangle\_LCD(120, 200, 50, 190, 0xff, 0xff, 0xff);

//Initialize ADC

ADMUX = 0x00; //Select Channel 0 for dummy conversion

ADCSRA = 0xc3; //Enable ADC, single conversion,

//prescaler of 8 for accurate results

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

//Initialize Timer/Counter0 Interrupt

TCCR0B = 0x04; //Prescaler of 1/256 - will result in interrupt every 8.19 ms

TIMSK0 = 0x01; //Enable Timer/Counter0 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timer/Counter0 ISR - occurs every 8.19 ms \*/

/\*Initiates ATD Conversion and stores results \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadUltrasonic\_ISR(void)

{

//Temporary data storage variables

unsigned int datalow = 0;

unsigned int datahigh = 0;

unsigned int distance = 0;

char datred = 0;

char datgreen = 0;

char datblue = 0;

//Perform ATD Conversion

ADCSRA |= 0x43; //Start single conversion, prescaler = 8

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

datalow = ADCL; //Read low data register first

//datahigh = ((unsigned int)(ADCH << 8)); //Read high data register

datahigh = ADCH;

datahigh = datahigh << 8;

distance = datalow | datahigh;

//Demo - assign color based on ADC value

if((25<= distance) && (distance < 30))

{

//Red sensor

datred = 0xff;

}

else if((30<= distance) && (distance < 35))

{

//Cyan sensor

datgreen = 0xff;

datblue = 0xff;

}

else if((35<= distance) && (distance < 40))

{

//Yellow sensor

datred = 0xff;

datgreen = 0xff;

}

else if((40<= distance) && (distance < 45))

{

//Green Sensor

datgreen = 0xff;

}

else if((45<= distance) && (distance < 50))

{

//Blue Sensor

datblue = 0xff;

}

else if((50<= distance) && (distance < 55))

{

//Magenta Sensor

datred = 0xff;

datblue = 0xff;

}

else

{

//White Sensor

datred = 0xff;

datgreen = 0xff;

datblue = 0xff;

}

switch(SensorCounter)

{

case 1:

Left1 = distance; //Store data for sensor Left1

//Draw Sensor Left1

VariableRectangle\_LCD(120, 150, 10, 40, datred, datgreen, datblue);

ADMUX = 0x01; //Select Channel 1 for next conversion

break;

case 2:

Left2 = distance; //Store data for sensor Left2

//Draw Sensor Left2

VariableRectangle\_LCD(80, 110, 50, 80, datred, datgreen, datblue);

ADMUX = 0x02; //Select Channel 2 for next conversion

break;

case 3:

Left3 = distance; //Store data for sensor Left3

//Draw Sensor Left3

VariableRectangle\_LCD(80, 110, 160, 190, datred, datgreen, datblue);

ADMUX = 0x03; //Select Channel 3 for next conversion

break;

case 4:

Left4 = distance; //Store data for sensor Left4

//Draw Sensor Left4

VariableRectangle\_LCD(120, 150, 200, 230, datred, datgreen, datblue);

ADMUX = 0x04; //Select Channel 4 for next conversion

break;

case 5:

Right1 = distance; //Store data for sensor Right1

//Draw Sensor Right1

VariableRectangle\_LCD(170, 200, 10, 40, datred, datgreen, datblue);

ADMUX = 0x05; //Select Channel 5 for next conversion

break;

case 6:

Right2 = distance; //Store data for sensor Right2

//Draw Sensor Right2

VariableRectangle\_LCD(210, 240, 50, 80, datred, datgreen, datblue);

ADMUX = 0x06; //Select Channel 6 for next conversion

break;

case 7:

Right3 = distance; //Store data for sensor Right3

//Draw Sensor Right3

VariableRectangle\_LCD(210, 240, 160, 190, datred, datgreen, datblue);

ADMUX = 0x07; //Select Channel 7 for next conversion

break;

default:

Right4 = distance; //Store data for sensor Right4

//Draw Sensor Right4

VariableRectangle\_LCD(170, 200, 200, 230, datred, datgreen, datblue);

ADMUX = 0x00; //Select Channel 0 for next conversion

SensorCounter = 0;

break;

}

SensorCounter++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

## DemoURD.c

//Demo URD

//Authors: Dana Schultz, Kathleen Shea

//Created: April 16, 2012

//Revised: April 20, 2012

//Version: 3

//Uses Atmel ATmega1284 microcontroller to demonstrate:

//The ultrasonic sensor by displaying wheelchair with eight sensors around it

//Colors of the sensors change with respect to distance of sensed object

//The touchscreen by displaying four quadrants, color of each changes to

//white when touched and then changes back

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

void Square\_LCD(unsigned char);

void VariableRectangle\_LCD(unsigned int, unsigned int, unsigned int, unsigned int,

unsigned char, unsigned char, unsigned char);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void);

void ReadUltrasonic\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

//Timer/Counter0 Overflow

#pragma interrupt\_handler ReadUltrasonic\_ISR:19

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

unsigned char TouchCount;

//Ultrasonic Sensor Data Variables

unsigned int Left1, Left2, Left3, Left4;

unsigned int Right1, Right2, Right3, Right4;

char SensorCounter;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

Initialize\_Ultrasonic();

while(1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0xff,0xff,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Displays square in top right corner of specified color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Square\_LCD(unsigned char Quad)

{

unsigned int x,y;

if(Quad == 1)

{

WindowSet\_LCD(0,159,0,119);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 2)

{

WindowSet\_LCD(160,319,0,119);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 3)

{

WindowSet\_LCD(0,159,120,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else if(Quad == 4)

{

WindowSet\_LCD(160,319,120,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff,0xff,0xff);

}

}

}

else

{

WindowSet\_LCD(0,319,0,239);

WriteCommand\_LCD(0x2c);

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff, 0x00, 0xff); //Magenta

}

for(x=0;x<160;x++)

{

SendPixel\_LCD(0x00, 0xff, 0xff); //Cyan

}

}

for(y=0;y<120;y++)

{

for(x=0;x<160;x++)

{

SendPixel\_LCD(0x00, 0xff, 0x33); //Spring Green

}

for(x=0;x<160;x++)

{

SendPixel\_LCD(0xff, 0x33, 0x66); //Coral

}

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Writes rectangle of selected area and color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void VariableRectangle\_LCD(unsigned int startX, unsigned int endX, unsigned int startY,

unsigned int endY, unsigned char red, unsigned char green,

unsigned char blue)

{

unsigned int sizeX = endX - startX;

unsigned int sizeY = endY - startY;

unsigned int x,y;

WindowSet\_LCD(startX, endX, startY, endY);

WriteCommand\_LCD(0x2c);

for(y=0;y<=sizeY;y++)

{

for(x=0;x<=sizeX;x++)

{

SendPixel\_LCD(red,green,blue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

TouchCount = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK = 0x00; //disable INT0

EICRA = 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

TouchCount++;

Touch\_X = 0;

Touch\_Y = 0;

if(TouchCount > 3)

{

Delay\_ms(300);

TIMSK0 = 0x00; //Disable Ultrasonic Interrupt

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X = GetX();

Touch\_Y = GetY();

}

if((Touch\_X < 165) && (Touch\_Y < 165))

{

Square\_LCD(3);

}

else if((Touch\_X > 165) && (Touch\_Y < 165))

{

Square\_LCD(4);

}

else if((Touch\_X > 165) && (Touch\_Y > 165))

{

Square\_LCD(2);

}

else if((Touch\_X < 165) && (Touch\_Y > 165))

{

Square\_LCD(1);

}

else

{

Monochrome\_LCD(0xff,0xff,0xff);

}

Delay\_ms(750);

Square\_LCD(0);

}

EIFR = 0x01; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Sensor Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with ultrasonic sensors \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void)

{

//Initialize PortA for input

DDRA = 0x00;

//Initialize Variables

SensorCounter = 1;

Left1 = 0;

Left2 = 0;

Left3 = 0;

Left4 = 0;

Right1 = 0;

Right2 = 0;

Right3 = 0;

Right4 = 0;

//Demo - draw wheelchair

Monochrome\_LCD(0x00, 0x00, 0x00); //black background

//Draw white rectangle 80X140 pixels in center of screen

VariableRectangle\_LCD(120, 200, 50, 190, 0xff, 0xff, 0xff);

//Initialize ADC

ADMUX = 0x00; //Select Channel 0 for dummy conversion

ADCSRA = 0xc3; //Enable ADC, single conversion,

//prescaler of 8 for accurate results

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

//Initialize Timer/Counter0 Interrupt

TCCR0B = 0x04; //Prescaler of 1/256 - will result in interrupt every 8.19 ms

TIMSK0 = 0x01; //Enable Timer/Counter0 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timer/Counter0 ISR - occurs every 8.19 ms \*/

/\*Initiates ATD Conversion and stores results \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadUltrasonic\_ISR(void)

{

//Temporary data storage variables

unsigned int datalow = 0;

unsigned int datahigh = 0;

unsigned int distance = 0;

char datred = 0;

char datgreen = 0;

char datblue = 0;

//Perform ATD Conversion

ADCSRA |= 0x43; //Start single conversion, prescaler = 8

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

datalow = ADCL; //Read low data register first

//datahigh = ((unsigned int)(ADCH << 8)); //Read high data register

datahigh = ADCH;

datahigh = datahigh << 8;

distance = datalow | datahigh;

//Demo - assign color based on ADC value

if((25<= distance) && (distance < 30))

{

//Red sensor

datred = 0xff;

}

else if((30<= distance) && (distance < 35))

{

//Cyan sensor

datgreen = 0xff;

datblue = 0xff;

}

else if((35<= distance) && (distance < 40))

{

//Yellow sensor

datred = 0xff;

datgreen = 0xff;

}

else if((40<= distance) && (distance < 45))

{

//Green Sensor

datgreen = 0xff;

}

else if((45<= distance) && (distance < 50))

{

//Blue Sensor

datblue = 0xff;

}

else if((50<= distance) && (distance < 55))

{

//Magenta Sensor

datred = 0xff;

datblue = 0xff;

}

else

{

//White Sensor

datred = 0xff;

datgreen = 0xff;

datblue = 0xff;

}

switch(SensorCounter)

{

case 1:

Left1 = distance; //Store data for sensor Left1

//Draw Sensor Left1

VariableRectangle\_LCD(120, 150, 10, 40, datred, datgreen, datblue);

ADMUX = 0x01; //Select Channel 1 for next conversion

break;

case 2:

Left2 = distance; //Store data for sensor Left2

//Draw Sensor Left2

VariableRectangle\_LCD(80, 110, 50, 80, datred, datgreen, datblue);

ADMUX = 0x02; //Select Channel 2 for next conversion

break;

case 3:

Left3 = distance; //Store data for sensor Left3

//Draw Sensor Left3

VariableRectangle\_LCD(80, 110, 160, 190, datred, datgreen, datblue);

ADMUX = 0x03; //Select Channel 3 for next conversion

break;

case 4:

Left4 = distance; //Store data for sensor Left4

//Draw Sensor Left4

VariableRectangle\_LCD(120, 150, 200, 230, datred, datgreen, datblue);

ADMUX = 0x04; //Select Channel 4 for next conversion

break;

case 5:

Right1 = distance; //Store data for sensor Right1

//Draw Sensor Right1

VariableRectangle\_LCD(170, 200, 10, 40, datred, datgreen, datblue);

ADMUX = 0x05; //Select Channel 5 for next conversion

break;

case 6:

Right2 = distance; //Store data for sensor Right2

//Draw Sensor Right2

VariableRectangle\_LCD(210, 240, 50, 80, datred, datgreen, datblue);

ADMUX = 0x06; //Select Channel 6 for next conversion

break;

case 7:

Right3 = distance; //Store data for sensor Right3

//Draw Sensor Right3

VariableRectangle\_LCD(210, 240, 160, 190, datred, datgreen, datblue);

ADMUX = 0x07; //Select Channel 7 for next conversion

break;

default:

Right4 = distance; //Store data for sensor Right4

//Draw Sensor Right4

VariableRectangle\_LCD(170, 200, 200, 230, datred, datgreen, datblue);

ADMUX = 0x00; //Select Channel 0 for next conversion

SensorCounter = 0;

break;

}

SensorCounter++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

## Micro-SD Card Code Functions

##### RTC\_Routines Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\*\* RTC\_ROUTINES HEADER FILE \*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef \_RTC\_ROUTINES\_H\_

#define \_RTC\_ROUTINES\_H\_

#define SECONDS rtc\_register[0]

#define MINUTES rtc\_register[1]

#define HOURS rtc\_register[2]

#define DAY rtc\_register[3]

#define DATE rtc\_register[4]

#define MONTH rtc\_register[5]

#define YEAR rtc\_register[6]

unsigned int dateFAT, timeFAT;

unsigned char rtc\_register[7];

unsigned char RTC\_setStartAddress(void);

unsigned char RTC\_read(void);

unsigned char RTC\_write(void);

unsigned char RTC\_getTime(void);

unsigned char RTC\_getDate(void);

unsigned char RTC\_displayTime(void);

unsigned char RTC\_displayDate(void);

void RTC\_displayDay(void);

void RTC\_updateRegisters(void);

unsigned char RTC\_writeTime(void);

unsigned char RTC\_writeDate(void);

unsigned char RTC\_updateTime(void);

unsigned char RTC\_updateDate(void);

unsigned char getDateTime\_FAT(void);

#endif

##### I2C Routines Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\* I2C\_ROUTINES HEADER FILE \*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef \_I2C\_ROUTINES\_H\_

#define \_I2C\_ROUTINES\_H\_

#define START 0x08

#define REPEAT\_START 0x10

#define MT\_SLA\_ACK 0x18

#define MT\_SLA\_NACK 0x20

#define MT\_DATA\_ACK 0x28

#define MT\_DATA\_NACK 0x30

#define MR\_SLA\_ACK 0x40

#define MR\_SLA\_NACK 0x48

#define MR\_DATA\_ACK 0x50

#define MR\_DATA\_NACK 0x58

#define ARB\_LOST 0x38

#define ERROR\_CODE 0x7e

#define DS1307\_W 0xd0

#define DS1307\_R 0xd1

#define EEPROM\_W 0xa0

#define EEPROM\_R 0xa1

void twi\_init(void);

unsigned char i2c\_start(void);

unsigned char i2c\_repeatStart(void);

unsigned char i2c\_sendAddress(unsigned char);

unsigned char i2c\_sendData(unsigned char);

unsigned char i2c\_receiveData\_ACK(void);

unsigned char i2c\_receiveData\_NACK(void);

void i2c\_stop(void);

#endif

##### Raw Data Transfer Routines Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\* FUNCTIONS FOR SD RAW DATA TRANSFER \*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Link to the Post: http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html

#ifndef \_SD\_ROUTINES\_H\_

#define \_SD\_ROUTINES\_H\_

//Use following macro if you don't want to activate the multiple block access functions

//those functions are not required for FAT32

#define FAT\_TESTING\_ONLY

//use following macros if PB1 pin is used for Chip Select of SD

//#define SD\_CS\_ASSERT PORTB &= ~0x02

//#define SD\_CS\_DEASSERT PORTB |= 0x02

//use following macros if SS (PB4) pin is used for Chip Select of SD

#define SD\_CS\_ASSERT PORTB &= ~0x10

#define SD\_CS\_DEASSERT PORTB |= 0x10

//SD commands, many of these are not used here

#define GO\_IDLE\_STATE 0

#define SEND\_OP\_COND 1

#define SEND\_IF\_COND 8

#define SEND\_CSD 9

#define STOP\_TRANSMISSION 12

#define SEND\_STATUS 13

#define SET\_BLOCK\_LEN 16

#define READ\_SINGLE\_BLOCK 17

#define READ\_MULTIPLE\_BLOCKS 18

#define WRITE\_SINGLE\_BLOCK 24

#define WRITE\_MULTIPLE\_BLOCKS 25

#define ERASE\_BLOCK\_START\_ADDR 32

#define ERASE\_BLOCK\_END\_ADDR 33

#define ERASE\_SELECTED\_BLOCKS 38

#define SD\_SEND\_OP\_COND 41 //ACMD

#define APP\_CMD 55

#define READ\_OCR 58

#define CRC\_ON\_OFF 59

#define ON 1

#define OFF 0

volatile unsigned long startBlock, totalBlocks;

volatile unsigned char SDHC\_flag, cardType, buffer[512];

unsigned char SD\_init(void);

unsigned char SD\_sendCommand(unsigned char cmd, unsigned long arg);

unsigned char SD\_readSingleBlock(unsigned long startBlock);

unsigned char SD\_writeSingleBlock(unsigned long startBlock);

unsigned char SD\_readMultipleBlock (unsigned long startBlock, unsigned long totalBlocks);

unsigned char SD\_writeMultipleBlock(unsigned long startBlock, unsigned long totalBlocks);

unsigned char SD\_erase (unsigned long startBlock, unsigned long totalBlocks);

#endif

##### FAT32 Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\* ROUTINES FOR FAT32 IMPLEMATATION OF SD CARD \*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Link to the Post: http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html

#ifndef \_FAT32\_H\_

#define \_FAT32\_H\_

//Structure to access Master Boot Record for getting info about partioions

struct MBRinfo\_Structure{

unsigned char nothing[446]; //ignore, placed here to fill the gap in the structure

unsigned char partitionData[64]; //partition records (16x4)

unsigned int signature; //0xaa55

};

//Structure to access info of the first partioion of the disk

struct partitionInfo\_Structure{

unsigned char status; //0x80 - active partition

unsigned char headStart; //starting head

unsigned int cylSectStart; //starting cylinder and sector

unsigned char type; //partition type

unsigned char headEnd; //ending head of the partition

unsigned int cylSectEnd; //ending cylinder and sector

unsigned long firstSector; //total sectors between MBR & the first sector of the partition

unsigned long sectorsTotal; //size of this partition in sectors

};

//Structure to access boot sector data

struct BS\_Structure{

unsigned char jumpBoot[3]; //default: 0x009000EB

unsigned char OEMName[8];

unsigned int bytesPerSector; //deafault: 512

unsigned char sectorPerCluster;

unsigned int reservedSectorCount;

unsigned char numberofFATs;

unsigned int rootEntryCount;

unsigned int totalSectors\_F16; //must be 0 for FAT32

unsigned char mediaType;

unsigned int FATsize\_F16; //must be 0 for FAT32

unsigned int sectorsPerTrack;

unsigned int numberofHeads;

unsigned long hiddenSectors;

unsigned long totalSectors\_F32;

unsigned long FATsize\_F32; //count of sectors occupied by one FAT

unsigned int extFlags;

unsigned int FSversion; //0x0000 (defines version 0.0)

unsigned long rootCluster; //first cluster of root directory (=2)

unsigned int FSinfo; //sector number of FSinfo structure (=1)

unsigned int BackupBootSector;

unsigned char reserved[12];

unsigned char driveNumber;

unsigned char reserved1;

unsigned char bootSignature;

unsigned long volumeID;

unsigned char volumeLabel[11]; //"NO NAME "

unsigned char fileSystemType[8]; //"FAT32"

unsigned char bootData[420];

unsigned int bootEndSignature; //0xaa55

};

//Structure to access FSinfo sector data

struct FSInfo\_Structure

{

unsigned long leadSignature; //0x41615252

unsigned char reserved1[480];

unsigned long structureSignature; //0x61417272

unsigned long freeClusterCount; //initial: 0xffffffff

unsigned long nextFreeCluster; //initial: 0xffffffff

unsigned char reserved2[12];

unsigned long trailSignature; //0xaa550000

};

//Structure to access Directory Entry in the FAT

struct dir\_Structure{

unsigned char name[11];

unsigned char attrib; //file attributes

unsigned char NTreserved; //always 0

unsigned char timeTenth; //tenths of seconds, set to 0 here

unsigned int createTime; //time file was created

unsigned int createDate; //date file was created

unsigned int lastAccessDate;

unsigned int firstClusterHI; //higher word of the first cluster number

unsigned int writeTime; //time of last write

unsigned int writeDate; //date of last write

unsigned int firstClusterLO; //lower word of the first cluster number

unsigned long fileSize; //size of file in bytes

};

//Attribute definitions for file/directory

#define ATTR\_READ\_ONLY 0x01

#define ATTR\_HIDDEN 0x02

#define ATTR\_SYSTEM 0x04

#define ATTR\_VOLUME\_ID 0x08

#define ATTR\_DIRECTORY 0x10

#define ATTR\_ARCHIVE 0x20

#define ATTR\_LONG\_NAME 0x0f

#define DIR\_ENTRY\_SIZE 0x32

#define EMPTY 0x00

#define DELETED 0xe5

#define GET 0

#define SET 1

#define READ 0

#define VERIFY 1

#define ADD 0

#define REMOVE 1

#define LOW 0

#define HIGH 1

#define TOTAL\_FREE 1

#define NEXT\_FREE 2

#define GET\_LIST 0

#define GET\_FILE 1

#define DELETE 2

#define EOF 0x0fffffff

//\*\*\*\*\*\*\*\*\*\*\*\*\* external variables \*\*\*\*\*\*\*\*\*\*\*\*\*

volatile unsigned long firstDataSector, rootCluster, totalClusters;

volatile unsigned int bytesPerSector, sectorPerCluster, reservedSectorCount;

unsigned long unusedSectors, appendFileSector, appendFileLocation, fileSize, appendStartCluster;

//global flag to keep track of free cluster count updating in FSinfo sector

unsigned char freeClusterCountUpdated;

//\*\*\*\*\*\*\*\*\*\*\*\*\* functions \*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char getBootSectorData (void);

unsigned long getFirstSector(unsigned long clusterNumber);

unsigned long getSetFreeCluster(unsigned char totOrNext, unsigned char get\_set, unsigned long FSEntry);

struct dir\_Structure\* findFiles (unsigned char flag, unsigned char \*fileName);

unsigned long getSetNextCluster (unsigned long clusterNumber,unsigned char get\_set,unsigned long clusterEntry);

unsigned char readFile (unsigned char flag, unsigned char \*fileName);

unsigned char convertFileName (unsigned char \*fileName);

void writeFile (unsigned char \*fileName);

void appendFile (void);

unsigned long searchNextFreeCluster (unsigned long startCluster);

void memoryStatistics (void);

void displayMemory (unsigned char flag, unsigned long memory);

void deleteFile (unsigned char \*fileName);

void freeMemoryUpdate (unsigned char flag, unsigned long size);

#endif

##### USART Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\* FUNCTIONS FOR SERIAL COMMUNICATION USING UART \*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef \_UART\_ROUTINES\_H\_

#define \_UART\_ROUTINES\_H\_

#define CHAR 0

#define INT 1

#define LONG 2

#define TX\_NEWLINE {transmitByte(0x0d); transmitByte(0x0a);}

void uart0\_init(void);

unsigned char receiveByte(void);

void transmitByte(unsigned char);

void transmitString\_F(char\*);

void transmitString(unsigned char\*);

void transmitHex( unsigned char dataType, unsigned long data );

#endif

##### SPI Header File

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\*\*\* FUNCTIONS FOR SD RAW DATA TRANSFER \*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Controller: ATmega32 (Clock: 8 Mhz-internal)

//Compiler : AVR-GCC (winAVR with AVRStudio)

//Project V.: Version - 2.4.1

//Author : CC Dharmani, Chennai (India)

// www.dharmanitech.com

//Date : 24 Apr 2011

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Link to the Post: http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html

#ifndef \_SD\_ROUTINES\_H\_

#define \_SD\_ROUTINES\_H\_

//Use following macro if you don't want to activate the multiple block access functions

//those functions are not required for FAT32

#define FAT\_TESTING\_ONLY

//use following macros if PB1 pin is used for Chip Select of SD

//#define SD\_CS\_ASSERT PORTB &= ~0x02

//#define SD\_CS\_DEASSERT PORTB |= 0x02

//use following macros if SS (PB4) pin is used for Chip Select of SD

#define SD\_CS\_ASSERT PORTB &= ~0x10

#define SD\_CS\_DEASSERT PORTB |= 0x10

//SD commands, many of these are not used here

#define GO\_IDLE\_STATE 0

#define SEND\_OP\_COND 1

#define SEND\_IF\_COND 8

#define SEND\_CSD 9

#define STOP\_TRANSMISSION 12

#define SEND\_STATUS 13

#define SET\_BLOCK\_LEN 16

#define READ\_SINGLE\_BLOCK 17

#define READ\_MULTIPLE\_BLOCKS 18

#define WRITE\_SINGLE\_BLOCK 24

#define WRITE\_MULTIPLE\_BLOCKS 25

#define ERASE\_BLOCK\_START\_ADDR 32

#define ERASE\_BLOCK\_END\_ADDR 33

#define ERASE\_SELECTED\_BLOCKS 38

#define SD\_SEND\_OP\_COND 41 //ACMD

#define APP\_CMD 55

#define READ\_OCR 58

#define CRC\_ON\_OFF 59

#define ON 1

#define OFF 0

volatile unsigned long startBlock, totalBlocks;

volatile unsigned char SDHC\_flag, cardType, buffer[512];

unsigned char SD\_init(void);

unsigned char SD\_sendCommand(unsigned char cmd, unsigned long arg);

unsigned char SD\_readSingleBlock(unsigned long startBlock);

unsigned char SD\_writeSingleBlock(unsigned long startBlock);

unsigned char SD\_readMultipleBlock (unsigned long startBlock, unsigned long totalBlocks);

unsigned char SD\_writeMultipleBlock(unsigned long startBlock, unsigned long totalBlocks);

unsigned char SD\_erase (unsigned long startBlock, unsigned long totalBlocks);

#endif

##### SD\_Code.c

//SD\_Code.c

//Authors: Dana Schultz, Katie Shea

//Created: Feb 28, 2012

//Revised: March 19, 2012

//Version: 3

//Uses Atmel ATmega164P

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \*\*\*\* MAIN routine FOR Interfacing microSDCARD \*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define F\_CPU 8000000UL //freq 8 MHz

#include <avr/io.h>

#include <avr/pgmspace.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include "SPI\_routines.h"

#include "SD\_routines.h"

#include "UART\_routines.h"

#include "RTC\_routines.h"

#include "i2c\_routines.h"

#include "FAT32.h"

void port\_init(void)

{

PORTA = 0x00;

DDRA = 0x00;

PORTB = 0x00;

DDRB = 0xD0; //MISO line i/p, rest o/p

PORTC = 0x00;

DDRC = 0x00;

PORTD = 0x00;

DDRD = 0xFE;

}

//call this routine to initialize all peripherals

void init\_devices(void)

{

cli(); //all interrupts disabled

port\_init();

spi\_init();

twi\_init();

uart0\_init();

MCUCR = 0x00;

GICR = 0x00;

TIMSK = 0x00; //timer interrupt sources

//all peripherals are now initialized

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int main(void)

{

unsigned char option, error, data, FAT32\_active;

unsigned int i;

unsigned char fileName[13];

\_delay\_ms(100); //delay for VCC stabilization

init\_devices();

PORTD |= 0x04; //switching ON the LED (for testing purpose only)

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F (PSTR("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

TX\_NEWLINE;

transmitString\_F (PSTR(" microSD Card Testing.. "));

TX\_NEWLINE;

transmitString\_F (PSTR("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

TX\_NEWLINE;

cardType = 0;

for (i=0; i<10; i++)

{

error = SD\_init();

if(!error) break;

}

if(error)

{

if(error == 1) transmitString\_F(PSTR("SD card not detected.."));

if(error == 2) transmitString\_F(PSTR("Card Initialization failed.."));

while(1); //wait here forever if error in SD init

}

switch (cardType)

{

case 1:transmitString\_F(PSTR("Standard Capacity Card (Ver 1.x) Detected!"));

break;

case 2:transmitString\_F(PSTR("High Capacity Card Detected!"));

break;

case 3:transmitString\_F(PSTR("Standard Capacity Card (Ver 2.x) Detected!"));

break;

default:transmitString\_F(PSTR("Unknown SD Card Detected!"));

break;

}

SPI\_HIGH\_SPEED; //SCK - 4 MHz

\_delay\_ms(1); //some delay

FAT32\_active = 1;

error = getBootSectorData (); //read boot sector and keep necessary data in global variables

if(error)

{

TX\_NEWLINE;

transmitString\_F (PSTR("FAT32 not found!")); //FAT32 incompatible drive

FAT32\_active = 0;

}

while(1)

{

TX\_NEWLINE;

transmitString\_F(PSTR("Press any key..."));

TX\_NEWLINE;

option = receiveByte();

TX\_NEWLINE;

transmitString\_F(PSTR("> 0: Erase Blocks "));

TX\_NEWLINE;

transmitString\_F(PSTR("> 1: Write single Block 2: Read single Block"));

#ifndef FAT\_TESTING\_ONLY

TX\_NEWLINE;

transmitString\_F(PSTR("> 3: Write multiple Blocks 4: Read multiple Blocks"));

#endif

TX\_NEWLINE;

transmitString\_F(PSTR("> 5: Get file list 6: Read File"));

TX\_NEWLINE;

transmitString\_F(PSTR("> 7: Write File 8: Delete File"));

TX\_NEWLINE;

transmitString\_F(PSTR("> 9: Read SD Memory Capacity a: Show Date & Time"));

TX\_NEWLINE;

transmitString\_F(PSTR("> b: Update Date c: Update Time"));

TX\_NEWLINE;

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("> Select Option (0-9/a/b/c): "));

/\*WARNING: If option 0, 1 or 3 is selected, the card data may not be detected by PC/Laptop again,

as it may disturb the FAT format. In such a case you will need to format the card again with FAT32.

This options are given for learnig the raw data transfer to & from the SD Card\*/

option = receiveByte();

transmitByte(option);

if(option >=0x35 && option <=0x39) //options 5 to 9 disabled if FAT32 not found

{

if(!FAT32\_active)

{

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("FAT32 options disabled!"));

continue;

}

}

if((option >= 0x30) && (option <=0x34)) //get starting block address for options 0 to 4

{

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("Enter the Block number (0000-9999):"));

data = receiveByte(); transmitByte(data);

startBlock = (data & 0x0f) \* 1000;

data = receiveByte(); transmitByte(data);

startBlock += (data & 0x0f) \* 100;

data = receiveByte(); transmitByte(data);

startBlock += (data & 0x0f) \* 10;

data = receiveByte(); transmitByte(data);

startBlock += (data & 0x0f);

TX\_NEWLINE;

}

totalBlocks = 1;

#ifndef FAT\_TESTING\_ONLY

if((option == 0x30) || (option == 0x33) || (option == 0x34)) //get total number of blocks for options 0, 3 or 4

{

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("How many blocks? (000-999):"));

data = receiveByte(); transmitByte(data);

totalBlocks = (data & 0x0f) \* 100;

data = receiveByte(); transmitByte(data);

totalBlocks += (data & 0x0f) \* 10;

data = receiveByte(); transmitByte(data);

totalBlocks += (data & 0x0f);

TX\_NEWLINE;

}

#endif

switch (option)

{

case '0': //error = SD\_erase (block, totalBlocks);

error = SD\_erase (startBlock, totalBlocks);

TX\_NEWLINE;

if(error)

transmitString\_F(PSTR("Erase failed.."));

else

transmitString\_F(PSTR("Erased!"));

break;

case '1': TX\_NEWLINE;

transmitString\_F(PSTR(" Enter text (End with ~):"));

i=0;

do

{

data = receiveByte();

transmitByte(data);

buffer[i++] = data;

if(data == 0x0d)

{

transmitByte(0x0a);

buffer[i++] = 0x0a;

}

if(i == 512) break;

}while (data != '~');

error = SD\_writeSingleBlock (startBlock);

TX\_NEWLINE;

TX\_NEWLINE;

if(error)

transmitString\_F(PSTR("Write failed.."));

else

transmitString\_F(PSTR("Write successful!"));

break;

case '2': error = SD\_readSingleBlock (startBlock);

TX\_NEWLINE;

if(error)

transmitString\_F(PSTR("Read failed.."));

else

{

for(i=0;i<512;i++)

{

if(buffer[i] == '~') break;

transmitByte(buffer[i]);

}

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("Read successful!"));

}

break;

//next two options will work only if following macro is cleared from SD\_routines.h

#ifndef FAT\_TESTING\_ONLY

case '3':

error = SD\_writeMultipleBlock (startBlock, totalBlocks);

TX\_NEWLINE;

if(error)

transmitString\_F(PSTR("Write failed.."));

else

transmitString\_F(PSTR("Write successful!"));

break;

case '4': error = SD\_readMultipleBlock (startBlock, totalBlocks);

TX\_NEWLINE;

if(error)

transmitString\_F(PSTR("Read failed.."));

else

transmitString\_F(PSTR("Read successful!"));

break;

#endif

case '5': TX\_NEWLINE;

findFiles(GET\_LIST,0);

break;

case '6':

case '7':

case '8': TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("Enter file name: "));

for(i=0; i<13; i++)

fileName[i] = 0x00; //clearing any previously stored file name

i=0;

while(1)

{

data = receiveByte();

if(data == 0x0d) break; //'ENTER' key pressed

if(data == 0x08) //'Back Space' key pressed

{

if(i != 0)

{

transmitByte(data);

transmitByte(' ');

transmitByte(data);

i--;

}

continue;

}

if(data <0x20 || data > 0x7e) continue; //check for valid English text character

transmitByte(data);

fileName[i++] = data;

if(i==13){transmitString\_F(PSTR(" file name too long..")); break;}

}

if(i>12) break;

TX\_NEWLINE;

if(option == '6')

readFile( READ, fileName);

if(option == '7')

writeFile(fileName);

if(option == '8')

deleteFile(fileName);

break;

case '9': memoryStatistics();

break;

case 'a':

case 'A': RTC\_displayDate();

RTC\_displayTime();

break;

case 'b':

case 'B': RTC\_updateDate();

break;

case 'c':

case 'C': RTC\_updateTime();

break;

default: TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR(" Invalid option!"));

TX\_NEWLINE;

}

TX\_NEWLINE;

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SPI initialize for SD card\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//clock rate: 125Khz

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void spi\_init(void)

{

SPCR = 0x52; //setup SPI: Master mode, MSB first, SCK phase low, SCK idle low

SPSR = 0x00;

}

unsigned char SPI\_transmit(unsigned char data)

{

// Start transmission

SPDR = data;

// Wait for transmission complete

while(!(SPSR & (1<<SPIF)));

data = SPDR;

return(data);

}

unsigned char SPI\_receive(void)

{

unsigned char data;

// Wait for reception complete

SPDR = 0xff;

while(!(SPSR & (1<<SPIF)));

data = SPDR;

// Return data register

return data;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Functions for Serial Communication using USART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//UART0 initialize

//baud rate: 19200 (for controller clock = 8MHz)

//char size: 8 bit

//parity: Disabled

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void uart0\_init(void)

{

UCSRnB = 0x00; //disable while setting baud rate

UCSRnA = 0x00; //disable while setting baud rate

UCSRnC = (1 << URSEL) | 0x06;

UBRRnL = 0x19; //set baud rate lo

UBRRnH = 0x00; //set baud rate hi

UCSRnB = 0x18;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to receive a single byte

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char receiveByte( void )

{

unsigned char data, status;

while(!(UCSRA & (1<<RXC))); // Wait for incomming data

status = UCSRA;

data = UDR;

return(data);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit a single byte

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void transmitByte( unsigned char data )

{

while ( !(UCSRA & (1<<UDRE)) )

; /\* Wait for empty transmit buffer \*/

UDR = data; /\* Start transmition \*/

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit hex format data

//first argument indicates type: CHAR, INT or LONG

//Second argument is the data to be displayed

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void transmitHex( unsigned char dataType, unsigned long data )

{

unsigned char count, i, temp;

unsigned char dataString[] = "0x ";

if (dataType == CHAR) count = 2;

if (dataType == INT) count = 4;

if (dataType == LONG) count = 8;

for(i=count; i>0; i--)

{

temp = data % 16;

if((temp>=0) && (temp<10)) dataString [i+1] = temp + 0x30;

else dataString [i+1] = (temp - 10) + 0x41;

data = data/16;

}

transmitString (dataString);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit a string in Flash

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void transmitString\_F(char\* string)

{

while (pgm\_read\_byte(&(\*string)))

transmitByte(pgm\_read\_byte(&(\*string++)));

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit a string in RAM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void transmitString(unsigned char\* string)

{

while (\*string)

transmitByte(\*string++);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Functions for SD Raw Data Tramsfer \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to initialize the SD/SDHC card in SPI mode

//Arguments : none

//return : unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_init(void)

{

unsigned char i, response, SD\_version;

unsigned int retry=0 ;

for(i=0;i<10;i++)

SPI\_transmit(0xff); //80 clock pulses spent before sending the first command

SD\_CS\_ASSERT;

do

{

response = SD\_sendCommand(GO\_IDLE\_STATE, 0); //send 'reset & go idle' command

retry++;

if(retry>0x20)

return 1; //time out, card not detected

}while(response != 0x01);

SD\_CS\_DEASSERT;

SPI\_transmit (0xff);

SPI\_transmit (0xff);

retry = 0;

SD\_version = 2; //default set to SD compliance with ver2.x;

//this may change after checking the next command

do

{

response = SD\_sendCommand(SEND\_IF\_COND,0x000001AA); //Check power supply status, mendatory for SDHC card

retry++;

if(retry>0xfe)

{

TX\_NEWLINE;

SD\_version = 1;

cardType = 1;

break;

} //time out

}while(response != 0x01);

retry = 0;

do

{

response = SD\_sendCommand(APP\_CMD,0); //CMD55, must be sent before sending any ACMD command

response = SD\_sendCommand(SD\_SEND\_OP\_COND,0x40000000); //ACMD41

retry++;

if(retry>0xfe)

{

TX\_NEWLINE;

return 2; //time out, card initialization failed

}

}while(response != 0x00);

retry = 0;

SDHC\_flag = 0;

if (SD\_version == 2)

{

do

{

response = SD\_sendCommand(READ\_OCR,0);

retry++;

if(retry>0xfe)

{

TX\_NEWLINE;

cardType = 0;

break;

} //time out

}while(response != 0x00);

if(SDHC\_flag == 1) cardType = 2;

else cardType = 3;

}

//SD\_sendCommand(CRC\_ON\_OFF, OFF); //disable CRC; deafault - CRC disabled in SPI mode

//SD\_sendCommand(SET\_BLOCK\_LEN, 512); //set block size to 512; default size is 512

return 0; //successful return

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to send a command to SD card

//Arguments : unsigned char (8-bit command value)

// & unsigned long (32-bit command argument)

//return : unsigned char; response byte

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_sendCommand(unsigned char cmd, unsigned long arg)

{

unsigned char response, retry=0, status;

//SD card accepts byte address while SDHC accepts block address in multiples of 512

//so, if it's SD card we need to convert block address into corresponding byte address by

//multipying it with 512. which is equivalent to shifting it left 9 times

//following 'if' loop does that

if(SDHC\_flag == 0)

if(cmd == READ\_SINGLE\_BLOCK ||

cmd == READ\_MULTIPLE\_BLOCKS ||

cmd == WRITE\_SINGLE\_BLOCK ||

cmd == WRITE\_MULTIPLE\_BLOCKS ||

cmd == ERASE\_BLOCK\_START\_ADDR||

cmd == ERASE\_BLOCK\_END\_ADDR )

{

arg = arg << 9;

}

SD\_CS\_ASSERT;

SPI\_transmit(cmd | 0x40); //send command, first two bits always '01'

SPI\_transmit(arg>>24);

SPI\_transmit(arg>>16);

SPI\_transmit(arg>>8);

SPI\_transmit(arg);

if(cmd == SEND\_IF\_COND) //it is compulsory to send correct CRC for CMD8 (CRC=0x87) & CMD0 (CRC=0x95)

SPI\_transmit(0x87); //for remaining commands, CRC is ignored in SPI mode

else

SPI\_transmit(0x95);

while((response = SPI\_receive()) == 0xff) //wait response

if(retry++ > 0xfe) break; //time out error

if(response == 0x00 && cmd == 58) //checking response of CMD58

{

status = SPI\_receive() & 0x40; //first byte of the OCR register (bit 31:24)

if(status == 0x40) SDHC\_flag = 1; //we need it to verify SDHC card

else SDHC\_flag = 0;

SPI\_receive(); //remaining 3 bytes of the OCR register are ignored here

SPI\_receive(); //one can use these bytes to check power supply limits of SD

SPI\_receive();

}

SPI\_receive(); //extra 8 CLK

SD\_CS\_DEASSERT;

return response; //return state

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to erase specified no. of blocks of SD card

//Arguments : none

//return : unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_erase (unsigned long startBlock, unsigned long totalBlocks)

{

unsigned char response;

response = SD\_sendCommand(ERASE\_BLOCK\_START\_ADDR, startBlock); //send starting block address

if(response != 0x00) //check for SD status: 0x00 - OK (No flags set)

return response;

response = SD\_sendCommand(ERASE\_BLOCK\_END\_ADDR,(startBlock + totalBlocks - 1)); //send end block address

if(response != 0x00)

return response;

response = SD\_sendCommand(ERASE\_SELECTED\_BLOCKS, 0); //erase all selected blocks

if(response != 0x00)

return response;

return 0; //normal return

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to read a single block from SD card

//return : unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_readSingleBlock(unsigned long startBlock)

{

unsigned char response;

unsigned int i, retry=0;

response = SD\_sendCommand(READ\_SINGLE\_BLOCK, startBlock); //read a Block command

if(response != 0x00) return response; //check for SD status: 0x00 - OK (No flags set)

SD\_CS\_ASSERT;

retry = 0;

while(SPI\_receive() != 0xfe) //wait for start block token 0xfe (0x11111110)

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;} //return if time-out

for(i=0; i<512; i++) //read 512 bytes

buffer[i] = SPI\_receive();

SPI\_receive(); //receive incoming CRC (16-bit), CRC is ignored here

SPI\_receive();

SPI\_receive(); //extra 8 clock pulses

SD\_CS\_DEASSERT;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to write to a single block of SD card

//return : unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_writeSingleBlock(unsigned long startBlock)

{

unsigned char response;

unsigned int i, retry=0;

response = SD\_sendCommand(WRITE\_SINGLE\_BLOCK, startBlock); //write a Block command

if(response != 0x00) return response; //check for SD status: 0x00 - OK (No flags set)

SD\_CS\_ASSERT;

SPI\_transmit(0xfe); //Send start block token 0xfe (0x11111110)

for(i=0; i<512; i++) //send 512 bytes data

SPI\_transmit(buffer[i]);

SPI\_transmit(0xff); //transmit dummy CRC (16-bit), CRC is ignored here

SPI\_transmit(0xff);

response = SPI\_receive();

if( (response & 0x1f) != 0x05) //response= 0xXXX0AAA1 ; AAA='010' - data accepted

{ //AAA='101'-data rejected due to CRC error

SD\_CS\_DEASSERT; //AAA='110'-data rejected due to write error

return response;

}

while(!SPI\_receive()) //wait for SD card to complete writing and get idle

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;}

SD\_CS\_DEASSERT;

SPI\_transmit(0xff); //just spend 8 clock cycle delay before reasserting the CS line

SD\_CS\_ASSERT; //re-asserting the CS line to verify if card is still busy

while(!SPI\_receive()) //wait for SD card to complete writing and get idle

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;}

SD\_CS\_DEASSERT;

return 0;

}

#ifndef FAT\_TESTING\_ONLY

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function : to read multiple blocks from SD card & send every block to UART

//return : unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_readMultipleBlock (unsigned long startBlock, unsigned long totalBlocks)

{

unsigned char response;

unsigned int i, retry=0;

retry = 0;

response = SD\_sendCommand(READ\_MULTIPLE\_BLOCKS, startBlock); //write a Block command

if(response != 0x00) return response; //check for SD status: 0x00 - OK (No flags set)

SD\_CS\_ASSERT;

while( totalBlocks )

{

retry = 0;

while(SPI\_receive() != 0xfe) //wait for start block token 0xfe (0x11111110)

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;} //return if time-out

for(i=0; i<512; i++) //read 512 bytes

buffer[i] = SPI\_receive();

SPI\_receive(); //receive incoming CRC (16-bit), CRC is ignored here

SPI\_receive();

SPI\_receive(); //extra 8 cycles

TX\_NEWLINE;

transmitString\_F(PSTR(" --------- "));

TX\_NEWLINE;

for(i=0; i<512; i++) //send the block to UART

{

if(buffer[i] == '~') break;

transmitByte ( buffer[i] );

}

TX\_NEWLINE;

transmitString\_F(PSTR(" --------- "));

TX\_NEWLINE;

totalBlocks--;

}

SD\_sendCommand(STOP\_TRANSMISSION, 0); //command to stop transmission

SD\_CS\_DEASSERT;

SPI\_receive(); //extra 8 clock pulses

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to receive data from UART and write to multiple blocks of SD card

//return: unsigned char; will be 0 if no error,

// otherwise the response byte will be sent

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char SD\_writeMultipleBlock(unsigned long startBlock, unsigned long totalBlocks)

{

unsigned char response, data;

unsigned int i, retry=0;

unsigned long blockCounter=0, size;

response = SD\_sendCommand(WRITE\_MULTIPLE\_BLOCKS, startBlock); //write a Block command

if(response != 0x00) return response; //check for SD status: 0x00 - OK (No flags set)

SD\_CS\_ASSERT;

TX\_NEWLINE;

transmitString\_F(PSTR(" Enter text (End with ~): "));

TX\_NEWLINE;

while( blockCounter < totalBlocks )

{

i=0;

do

{

data = receiveByte();

if(data == 0x08) //'Back Space' key pressed

{

if(i != 0)

{

transmitByte(data);

transmitByte(' ');

transmitByte(data);

i--;

size--;

}

continue;

}

transmitByte(data);

buffer[i++] = data;

if(data == 0x0d)

{

transmitByte(0x0a);

buffer[i++] = 0x0a;

}

if(i == 512) break;

}while (data != '~');

TX\_NEWLINE;

transmitString\_F(PSTR(" ---- "));

TX\_NEWLINE;

SPI\_transmit(0xfc); //Send start block token 0xfc (0x11111100)

for(i=0; i<512; i++) //send 512 bytes data

SPI\_transmit( buffer[i] );

SPI\_transmit(0xff); //transmit dummy CRC (16-bit), CRC is ignored here

SPI\_transmit(0xff);

response = SPI\_receive();

if( (response & 0x1f) != 0x05) //response= 0xXXX0AAA1 ; AAA='010' - data accepted

{ //AAA='101'-data rejected due to CRC error

SD\_CS\_DEASSERT; //AAA='110'-data rejected due to write error

return response;

}

while(!SPI\_receive()) //wait for SD card to complete writing and get idle

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;}

SPI\_receive(); //extra 8 bits

blockCounter++;

}

SPI\_transmit(0xfd); //send 'stop transmission token'

retry = 0;

while(!SPI\_receive()) //wait for SD card to complete writing and get idle

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;}

SD\_CS\_DEASSERT;

SPI\_transmit(0xff); //just spend 8 clock cycle delay before reasserting the CS signal

SD\_CS\_ASSERT; //re assertion of the CS signal is required to verify if card is still busy

while(!SPI\_receive()) //wait for SD card to complete writing and get idle

if(retry++ > 0xfffe){SD\_CS\_DEASSERT; return 1;}

SD\_CS\_DEASSERT;

return 0;

}

#endif

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Functions for i2c Communication \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//TWI initialize

// bit rate:18 (freq: 100Khz @16MHz)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void twi\_init(void)

{

TWCR= 0x00; //disable twi

TWBR= 0x12; //set bit rate

TWSR= 0x00; //set prescale

//TWCR= 0x44; //enable twi

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to start i2c communication

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_start(void)

{

TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //Send START condition

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//START condition has been transmitted

if ((TWSR & 0xF8) == START) //Check value of TWI Status Register

return(0);

else

return(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function for repeat start condition

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_repeatStart(void)

{

TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //Send START condition

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//START condition has been transmitted

if ((TWSR & 0xF8) == REPEAT\_START) //Check value of TWI Status Register

return(0);

else

return(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit address of the slave

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_sendAddress(unsigned char address)

{

unsigned char STATUS;

if((address & 0x01) == 0)

STATUS = MT\_SLA\_ACK;

else

STATUS = MR\_SLA\_ACK;

TWDR = address;

TWCR = (1<<TWINT)|(1<<TWEN); //Load SLA\_W into TWDR Register. Clear TWINT bit

//in TWCR to start transmission of address

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//SLA+W has been transmitted, and

//ACK/NACK has been received.

if ((TWSR & 0xF8) == STATUS) //Check value of TWI Status Register

return(0);

else

return(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to transmit a data byte

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_sendData(unsigned char data)

{

TWDR = data;

TWCR = (1<<TWINT) |(1<<TWEN); //Load SLA\_W into TWDR Register. Clear TWINT bit

//in TWCR to start transmission of data

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//data has been transmitted, and

//ACK/NACK has been received.

if ((TWSR & 0xF8) != MT\_DATA\_ACK) //Check value of TWI Status Register

return(1);

else

return(0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to receive a data byte and send ACKnowledge

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_receiveData\_ACK(void)

{

unsigned char data;

TWCR = (1<<TWEA)|(1<<TWINT)|(1<<TWEN);

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//data has been received

if ((TWSR & 0xF8) != MR\_DATA\_ACK) //Check value of TWI Status Register

return(ERROR\_CODE);

data = TWDR;

return(data);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to receive the last data byte (no acknowledge from master

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char i2c\_receiveData\_NACK(void)

{

unsigned char data;

TWCR = (1<<TWINT)|(1<<TWEN);

while (!(TWCR & (1<<TWINT))); //Wait for TWINT flag set. This indicates that the

//data has been received

if ((TWSR & 0xF8) != MR\_DATA\_NACK) //Check value of TWI Status Register

return(ERROR\_CODE);

data = TWDR;

return(data);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to end the i2c communication

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void i2c\_stop(void)

{

TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO); //Transmit STOP condition

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Functions for EEPROM Read/Write on i2c Communication \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char time[10]; //xxam:xx:xx;

unsigned char date[12]; //xx/xx/xxxx;

unsigned char day;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to set initial address of the RTC for subsequent reading / writing

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_setStartAddress(void)

{

unsigned char errorStatus;

errorStatus = i2c\_start();

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC start1 failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_sendAddress(DS1307\_W);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC sendAddress1 failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_sendData(0x00);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write-2 failed.."));

i2c\_stop();

return 1;

}

i2c\_stop();

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to read RTC registers and store them in buffer rtc\_register[]

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_read(void)

{

unsigned char errorStatus, i, data;

errorStatus = i2c\_start();

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC start1 failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_sendAddress(DS1307\_W);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC sendAddress1 failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_sendData(0x00);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write-1 failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_repeatStart();

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC repeat start failed.."));

i2c\_stop();

return 1;

}

errorStatus = i2c\_sendAddress(DS1307\_R);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC sendAddress2 failed.."));

i2c\_stop();

return 1;

}

for(i=0;i<7;i++)

{

if(i == 6) //no Acknowledge after receiving the last byte

data = i2c\_receiveData\_NACK();

else

data = i2c\_receiveData\_ACK();

if(data == ERROR\_CODE)

{

//transmitString\_F(PSTR("RTC receive failed.."));

i2c\_stop();

return 1;

}

rtc\_register[i] = data;

}

i2c\_stop();

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to form time string for sending it to LCD & UART

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_getTime(void)

{

unsigned char error;

error = RTC\_read();

if(error) return 1;

RTC\_read();

time[8] = 0x00; //NIL

time[7] = (SECONDS & 0x0f) | 0x30; //seconds(1's)

time[6] = ((SECONDS & 0x70) >> 4) | 0x30; //seconds(10's)

time[5] = ':';

time[4] = (MINUTES & 0x0f) | 0x30;

time[3] = ((MINUTES & 0x70) >> 4) | 0x30;

time[2] = ':';

time[1] = (HOURS & 0x0f) | 0x30;

time[0] = ((HOURS & 0x30) >> 4) | 0x30;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to form date string for sending it to LCD & UART

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_getDate(void)

{

unsigned char error;

error = RTC\_read();

if(error) return 1;

date[11] = 0x00;

date[10] = 0x00;

date[9] = (YEAR & 0x0f) | 0x30;

date[8] = ((YEAR & 0xf0) >> 4) | 0x30;

date[7] = '0';

date[6] = '2';

date[5] = '/';

date[4] = (MONTH & 0x0f) | 0x30;

date[3] = ((MONTH & 0x10) >> 4) | 0x30;

date[2] = '/';

date[1] = (DATE & 0x0f) | 0x30;

date[0] = ((DATE & 0x30) >> 4) | 0x30;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to display time on LCD and send it to PC (thru UART)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_displayTime(void)

{

unsigned char error;

error = RTC\_getTime();

if(error) return 1;

TX\_NEWLINE;

transmitString\_F(PSTR("Time:"));

transmitString(time);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to display date on LCD and send it to PC (UART)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_displayDate(void)

{

unsigned char error;

error = RTC\_getDate();

if(error) return 1;

TX\_NEWLINE;

transmitString\_F(PSTR("Date:"));

transmitString(date);

RTC\_displayDay();

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to get the string for day

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RTC\_displayDay(void)

{

transmitString\_F(PSTR(" Day: "));

switch(DAY)

{

case 0:transmitString\_F(PSTR("Sunday"));

break;

case 1:transmitString\_F(PSTR("Monday"));

break;

case 2:transmitString\_F(PSTR("Tuesday"));

break;

case 3:transmitString\_F(PSTR("Wednesday"));

break;

case 4:transmitString\_F(PSTR("Thursday"));

break;

case 5:transmitString\_F(PSTR("Friday"));

break;

case 6:transmitString\_F(PSTR("Saturday"));

break;

default: transmitString\_F(PSTR("Unknown"));

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to update buffer rtc\_register[] for next writing to RTC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RTC\_updateRegisters(void)

{

SECONDS = ((time[6] & 0x07) << 4) | (time[7] & 0x0f);

MINUTES = ((time[3] & 0x07) << 4) | (time[4] & 0x0f);

HOURS = ((time[0] & 0x03) << 4) | (time[1] & 0x0f);

DAY = date[10];

DATE = ((date[0] & 0x03) << 4) | (date[1] & 0x0f);

MONTH = ((date[3] & 0x01) << 4) | (date[4] & 0x0f);

YEAR = ((date[8] & 0x0f) << 4) | (date[9] & 0x0f);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to write new time in the RTC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_writeTime(void)

{

unsigned char errorStatus, i;

errorStatus = i2c\_start();

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC start1 failed.."));

i2c\_stop();

return(1);

}

errorStatus = i2c\_sendAddress(DS1307\_W);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC sendAddress1 failed.."));

i2c\_stop();

return(1);

}

errorStatus = i2c\_sendData(0x00);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write-1 failed.."));

i2c\_stop();

return(1);

}

for(i=0;i<3;i++)

{

errorStatus = i2c\_sendData(rtc\_register[i]);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write time failed.."));

i2c\_stop();

return(1);

}

}

i2c\_stop();

return(0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to write new date in the RTC

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_writeDate(void)

{

unsigned char errorStatus, i;

errorStatus = i2c\_start();

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC start1 failed.."));

i2c\_stop();

return(1);

}

errorStatus = i2c\_sendAddress(DS1307\_W);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC sendAddress1 failed.."));

i2c\_stop();

return(1);

}

errorStatus = i2c\_sendData(0x03);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write-1 failed.."));

i2c\_stop();

return(1);

}

for(i=3;i<7;i++)

{

errorStatus = i2c\_sendData(rtc\_register[i]);

if(errorStatus == 1)

{

//transmitString\_F(PSTR("RTC write date failed.."));

i2c\_stop();

return(1);

}

}

i2c\_stop();

return(0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to update RTC time by entering it at hyper terminal

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_updateTime(void)

{

unsigned char data;

TX\_NEWLINE;

transmitString\_F(PSTR("Enter Time in 24h format(hh:mm:ss):"));

data = receiveByte(); //receive hours

transmitByte(data);

if(data < 0x30 || data > 0x32)

goto TIME\_ERROR;

time[0]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto TIME\_ERROR;

time[1]= data;

transmitByte(':');

if(((time[1] & 0x0f) + ((time[0] & 0x03)\*10)) > 23)

goto TIME\_ERROR;

data = receiveByte(); //receive minutes

transmitByte(data);

if(data < 0x30 || data > 0x35)

goto TIME\_ERROR;

time[3]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto TIME\_ERROR;

time[4]= data;

transmitByte(':');

data = receiveByte(); //receive seconds

transmitByte(data);

if(data < 0x30 || data > 0x35)

goto TIME\_ERROR;

time[6]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto TIME\_ERROR;

time[7]= data;

RTC\_updateRegisters();

data = RTC\_writeTime();

TX\_NEWLINE;

if(data == 0)

{

transmitString\_F(PSTR("Time Updated sucessfully!"));

return 0;

}

else

{

transmitString\_F(PSTR("Time Update Failed.."));

return 1;

}

TIME\_ERROR:

TX\_NEWLINE;

transmitString\_F(PSTR("Invalid Entry.."));

return 1;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to update RTC date by entering it at hyper terminal

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char RTC\_updateDate(void)

{

unsigned char data;

TX\_NEWLINE;

transmitString\_F(PSTR("Enter Date (dd/mm/yy):"));

data = receiveByte(); //receive date

transmitByte(data);

if(data < 0x30 || data > 0x33)

goto DATE\_ERROR;

date[0]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto DATE\_ERROR;

date[1]= data;

if(((date[1] & 0x0f) + ((date[0] & 0x03)\*10)) > 31)

goto DATE\_ERROR;

transmitByte('/');

date[2] = '/';

data = receiveByte(); //receive month

transmitByte(data);

if(data < 0x30 || data > 0x31)

goto DATE\_ERROR;

date[3]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto DATE\_ERROR;

date[4] = data;

if(((date[4] & 0x0f) + ((date[3] & 0x03)\*10)) > 12)

goto DATE\_ERROR;

transmitByte('/');

date[5] = '/';

date[6] = '2'; //year is 20xx

date[7] = '0';

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto DATE\_ERROR;

date[8]= data;

data = receiveByte();

transmitByte(data);

if(data < 0x30 || data > 0x39)

goto DATE\_ERROR;

date[9]= data;

TX\_NEWLINE;

transmitString\_F(PSTR("Enter Day (Sunday:0, Monday:1...) (0-6):"));

data = receiveByte(); //receive Day of the week

transmitByte(data);

if(data < 0x30 || data > 0x36)

goto DATE\_ERROR;

date[10] = data & 0x0f;

RTC\_updateRegisters();

data = RTC\_writeDate();

TX\_NEWLINE;

if(data == 0)

{

transmitString\_F(PSTR("Date Updated sucessfully!"));

return 0;

}

else

{

transmitString\_F(PSTR("Date Update Failed.."));

return 1;

}

DATE\_ERROR:

TX\_NEWLINE;

transmitString\_F(PSTR("Invalid Entry.."));

return 1;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function to get RTC date & time in FAT32 format

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char getDateTime\_FAT(void)

{

unsigned char mth, dt, hr, min, sec, error;

unsigned int yr;

error = RTC\_read();

if(error) return 1;

yr = (YEAR & 0xf0) >> 4;

yr = (yr \* 10)+(YEAR & 0x0f);

yr = yr+2000;

yr = yr - 1980;

dateFAT = yr;

mth = (MONTH & 0xf0) >> 4;

mth = (mth \* 10)+(MONTH & 0x0f);

dateFAT = (dateFAT << 4) | mth;

dt = (DATE & 0xf0) >> 4;

dt = (dt \* 10)+(DATE & 0x0f);

dateFAT = (dateFAT << 5) | dt;

hr = (HOURS & 0xf0) >> 4;

hr = (hr \* 10)+(HOURS & 0x0f);

timeFAT = hr;

min = (MINUTES & 0xf0) >> 4;

min = (min \* 10)+(MINUTES & 0x0f);

timeFAT = (timeFAT << 6) | min;

sec = (SECONDS & 0xf0) >> 4;

sec = (sec \* 10)+(SECONDS & 0x0f);

sec = sec / 2; //FAT32 fromat accepts dates with 2sec resolution (e.g. value 5 => 10sec)

timeFAT = (timeFAT << 5) | sec;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Routines for FAT32 Implimentation of SD Card \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to read data from boot sector of SD card, to determine important

//parameters like bytesPerSector, sectorsPerCluster etc.

//Arguments: none

//return: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char getBootSectorData (void)

{

struct BS\_Structure \*bpb; //mapping the buffer onto the structure

struct MBRinfo\_Structure \*mbr;

struct partitionInfo\_Structure \*partition;

unsigned long dataSectors;

unusedSectors = 0;

SD\_readSingleBlock(0);

bpb = (struct BS\_Structure \*)buffer;

if(bpb->jumpBoot[0]!=0xE9 && bpb->jumpBoot[0]!=0xEB) //check if it is boot sector

{

mbr = (struct MBRinfo\_Structure \*) buffer; //if it is not boot sector, it must be MBR

if(mbr->signature != 0xaa55) return 1; //if it is not even MBR then it's not FAT32

partition = (struct partitionInfo\_Structure \*)(mbr->partitionData);//first partition

unusedSectors = partition->firstSector; //the unused sectors, hidden to the FAT

SD\_readSingleBlock(partition->firstSector);//read the bpb sector

bpb = (struct BS\_Structure \*)buffer;

if(bpb->jumpBoot[0]!=0xE9 && bpb->jumpBoot[0]!=0xEB) return 1;

}

bytesPerSector = bpb->bytesPerSector;

//transmitHex(INT, bytesPerSector); transmitByte(' ');

sectorPerCluster = bpb->sectorPerCluster;

//transmitHex(INT, sectorPerCluster); transmitByte(' ');

reservedSectorCount = bpb->reservedSectorCount;

rootCluster = bpb->rootCluster;// + (sector / sectorPerCluster) +1;

firstDataSector = bpb->hiddenSectors + reservedSectorCount + (bpb->numberofFATs \* bpb->FATsize\_F32);

dataSectors = bpb->totalSectors\_F32

- bpb->reservedSectorCount

- ( bpb->numberofFATs \* bpb->FATsize\_F32);

totalClusters = dataSectors / sectorPerCluster;

//transmitHex(LONG, totalClusters); transmitByte(' ');

if((getSetFreeCluster (TOTAL\_FREE, GET, 0)) > totalClusters) //check if FSinfo free clusters count is valid

freeClusterCountUpdated = 0;

else

freeClusterCountUpdated = 1;

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to calculate first sector address of any given cluster

//Arguments: cluster number for which first sector is to be found

//return: first sector address

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long getFirstSector(unsigned long clusterNumber)

{

return (((clusterNumber - 2) \* sectorPerCluster) + firstDataSector);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: get cluster entry value from FAT to find out the next cluster in the chain

//or set new cluster entry in FAT

//Arguments: 1. current cluster number, 2. get\_set (=GET, if next cluster is to be found or = SET,

//if next cluster is to be set 3. next cluster number, if argument#2 = SET, else 0

//return: next cluster number, if if argument#2 = GET, else 0

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long getSetNextCluster (unsigned long clusterNumber,

unsigned char get\_set,

unsigned long clusterEntry)

{

unsigned int FATEntryOffset;

unsigned long \*FATEntryValue;

unsigned long FATEntrySector;

unsigned char retry = 0;

//get sector number of the cluster entry in the FAT

FATEntrySector = unusedSectors + reservedSectorCount + ((clusterNumber \* 4) / bytesPerSector) ;

//get the offset address in that sector number

FATEntryOffset = (unsigned int) ((clusterNumber \* 4) % bytesPerSector);

//read the sector into a buffer

while(retry <10)

{ if(!SD\_readSingleBlock(FATEntrySector)) break; retry++;}

//get the cluster address from the buffer

FATEntryValue = (unsigned long \*) &buffer[FATEntryOffset];

if(get\_set == GET)

return ((\*FATEntryValue) & 0x0fffffff);

\*FATEntryValue = clusterEntry; //for setting new value in cluster entry in FAT

SD\_writeSingleBlock(FATEntrySector);

return (0);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to get or set next free cluster or total free clusters in FSinfo sector of SD card

//Arguments: 1.flag:TOTAL\_FREE or NEXT\_FREE,

// 2.flag: GET or SET

// 3.new FS entry, when argument2 is SET; or 0, when argument2 is GET

//return: next free cluster, if arg1 is NEXT\_FREE & arg2 is GET

// total number of free clusters, if arg1 is TOTAL\_FREE & arg2 is GET

// 0xffffffff, if any error or if arg2 is SET

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long getSetFreeCluster(unsigned char totOrNext, unsigned char get\_set, unsigned long FSEntry)

{

struct FSInfo\_Structure \*FS = (struct FSInfo\_Structure \*) &buffer;

unsigned char error;

SD\_readSingleBlock(unusedSectors + 1);

if((FS->leadSignature != 0x41615252) || (FS->structureSignature != 0x61417272) || (FS->trailSignature !=0xaa550000))

return 0xffffffff;

if(get\_set == GET)

{

if(totOrNext == TOTAL\_FREE)

return(FS->freeClusterCount);

else // when totOrNext = NEXT\_FREE

return(FS->nextFreeCluster);

}

else

{

if(totOrNext == TOTAL\_FREE)

FS->freeClusterCount = FSEntry;

else // when totOrNext = NEXT\_FREE

FS->nextFreeCluster = FSEntry;

error = SD\_writeSingleBlock(unusedSectors + 1); //update FSinfo

}

return 0xffffffff;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to get DIR/FILE list or a single file address (cluster number) or to delete a specified file

//Arguments: #1 - flag: GET\_LIST, GET\_FILE or DELETE #2 - pointer to file name (0 if arg#1 is GET\_LIST)

//return: first cluster of the file, if flag = GET\_FILE

// print file/dir list of the root directory, if flag = GET\_LIST

// Delete the file mentioned in arg#2, if flag = DELETE

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struct dir\_Structure\* findFiles (unsigned char flag, unsigned char \*fileName)

{

unsigned long cluster, sector, firstSector, firstCluster, nextCluster;

struct dir\_Structure \*dir;

unsigned int i;

unsigned char j;

cluster = rootCluster; //root cluster

while(1)

{

firstSector = getFirstSector (cluster);

for(sector = 0; sector < sectorPerCluster; sector++)

{

SD\_readSingleBlock (firstSector + sector);

for(i=0; i<bytesPerSector; i+=32)

{

dir = (struct dir\_Structure \*) &buffer[i];

if(dir->name[0] == EMPTY) //indicates end of the file list of the directory

{

if((flag == GET\_FILE) || (flag == DELETE))

transmitString\_F(PSTR("File does not exist!"));

return 0;

}

if((dir->name[0] != DELETED) && (dir->attrib != ATTR\_LONG\_NAME))

{

if((flag == GET\_FILE) || (flag == DELETE))

{

for(j=0; j<11; j++)

if(dir->name[j] != fileName[j]) break;

if(j == 11)

{

if(flag == GET\_FILE)

{

appendFileSector = firstSector + sector;

appendFileLocation = i;

appendStartCluster = (((unsigned long) dir->firstClusterHI) << 16) | dir->firstClusterLO;

fileSize = dir->fileSize;

return (dir);

}

else //when flag = DELETE

{

TX\_NEWLINE;

transmitString\_F(PSTR("Deleting.."));

TX\_NEWLINE;

TX\_NEWLINE;

firstCluster = (((unsigned long) dir->firstClusterHI) << 16) | dir->firstClusterLO;

//mark file as 'deleted' in FAT table

dir->name[0] = DELETED;

SD\_writeSingleBlock (firstSector+sector);

freeMemoryUpdate (ADD, dir->fileSize);

//update next free cluster entry in FSinfo sector

cluster = getSetFreeCluster (NEXT\_FREE, GET, 0);

if(firstCluster < cluster)

getSetFreeCluster (NEXT\_FREE, SET, firstCluster);

//mark all the clusters allocated to the file as 'free'

while(1)

{

nextCluster = getSetNextCluster (firstCluster, GET, 0);

getSetNextCluster (firstCluster, SET, 0);

if(nextCluster > 0x0ffffff6)

{transmitString\_F(PSTR("File deleted!"));return 0;}

firstCluster = nextCluster;

}

}

}

}

else //when flag = GET\_LIST

{

TX\_NEWLINE;

for(j=0; j<11; j++)

{

if(j == 8) transmitByte(' ');

transmitByte (dir->name[j]);

}

transmitString\_F (PSTR(" "));

if((dir->attrib != 0x10) && (dir->attrib != 0x08))

{

transmitString\_F (PSTR("FILE" ));

transmitString\_F (PSTR(" "));

displayMemory (LOW, dir->fileSize);

}

else

transmitString\_F ((dir->attrib == 0x10)? PSTR("DIR") : PSTR("ROOT"));

}

}

}

}

cluster = (getSetNextCluster (cluster, GET, 0));

if(cluster > 0x0ffffff6)

return 0;

if(cluster == 0)

{transmitString\_F(PSTR("Error in getting cluster")); return 0;}

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: if flag=READ then to read file from SD card and send contents to UART

//if flag=VERIFY then functions will verify whether a specified file is already existing

//Arguments: flag (READ or VERIFY) and pointer to the file name

//return: 0, if normal operation or flag is READ

// 1, if file is already existing and flag = VERIFY

// 2, if file name is incompatible

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char readFile (unsigned char flag, unsigned char \*fileName)

{

struct dir\_Structure \*dir;

unsigned long cluster, byteCounter = 0, fileSize, firstSector;

unsigned int k;

unsigned char j, error;

error = convertFileName (fileName); //convert fileName into FAT format

if(error) return 2;

dir = findFiles (GET\_FILE, fileName); //get the file location

if(dir == 0)

return (0);

if(flag == VERIFY) return (1); //specified file name is already existing

cluster = (((unsigned long) dir->firstClusterHI) << 16) | dir->firstClusterLO;

fileSize = dir->fileSize;

TX\_NEWLINE;

TX\_NEWLINE;

while(1)

{

firstSector = getFirstSector (cluster);

for(j=0; j<sectorPerCluster; j++)

{

SD\_readSingleBlock(firstSector + j);

for(k=0; k<512; k++)

{

transmitByte(buffer[k]);

if ((byteCounter++) >= fileSize ) return 0;

}

}

cluster = getSetNextCluster (cluster, GET, 0);

if(cluster == 0) {transmitString\_F(PSTR("Error in getting cluster")); return 0;}

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to convert normal short file name into FAT format

//Arguments: pointer to the file name

//return: 0, if successful else 1.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned char convertFileName (unsigned char \*fileName)

{

unsigned char fileNameFAT[11];

unsigned char j, k;

for(j=0; j<12; j++)

if(fileName[j] == '.') break;

if(j>8) {transmitString\_F(PSTR("Invalid fileName..")); return 1;}

for(k=0; k<j; k++) //setting file name

fileNameFAT[k] = fileName[k];

for(k=j; k<=7; k++) //filling file name trail with blanks

fileNameFAT[k] = ' ';

j++;

for(k=8; k<11; k++) //setting file extention

{

if(fileName[j] != 0)

fileNameFAT[k] = fileName[j++];

else //filling extension trail with blanks

while(k<11)

fileNameFAT[k++] = ' ';

}

for(j=0; j<11; j++) //converting small letters to caps

if((fileNameFAT[j] >= 0x61) && (fileNameFAT[j] <= 0x7a))

fileNameFAT[j] -= 0x20;

for(j=0; j<11; j++)

fileName[j] = fileNameFAT[j];

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to create a file in FAT32 format in the root directory if given

// file name does not exist; if the file already exists then append the data

//Arguments: pointer to the file name

//return: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void writeFile (unsigned char \*fileName)

{

unsigned char j, data, error, fileCreatedFlag = 0, start = 0, appendFile = 0, sectorEndFlag = 0, sector=0;

unsigned int i, firstClusterHigh=0, firstClusterLow=0; //value 0 is assigned just to avoid warning in compilation

struct dir\_Structure \*dir;

unsigned long cluster, nextCluster, prevCluster, firstSector, clusterCount, extraMemory;

j = readFile (VERIFY, fileName);

if(j == 1)

{

transmitString\_F(PSTR(" File already exists, appending data.."));

appendFile = 1;

cluster = appendStartCluster;

clusterCount=0;

while(1)

{

nextCluster = getSetNextCluster (cluster, GET, 0);

if(nextCluster == EOF) break;

cluster = nextCluster;

clusterCount++;

}

sector = (fileSize - (clusterCount \* sectorPerCluster \* bytesPerSector)) / bytesPerSector; //last sector number of the last cluster of the file

start = 1;

// appendFile();

// return;

}

else if(j == 2)

{

return; //invalid file name

}

else

{

TX\_NEWLINE;

transmitString\_F(PSTR(" Creating File.."));

cluster = getSetFreeCluster (NEXT\_FREE, GET, 0);

if(cluster > totalClusters)

cluster = rootCluster;

cluster = searchNextFreeCluster(cluster);

if(cluster == 0)

{

TX\_NEWLINE;

transmitString\_F(PSTR(" No free cluster!"));

return;

}

getSetNextCluster(cluster, SET, EOF); //last cluster of the file, marked EOF

firstClusterHigh = (unsigned int) ((cluster & 0xffff0000) >> 16 );

firstClusterLow = (unsigned int) ( cluster & 0x0000ffff);

fileSize = 0;

}

while(1)

{

if(start)

{

start = 0;

startBlock = getFirstSector (cluster) + sector;

SD\_readSingleBlock (startBlock);

i = fileSize % bytesPerSector;

j = sector;

}

else

{

startBlock = getFirstSector (cluster);

i=0;

j=0;

}

TX\_NEWLINE;

transmitString\_F(PSTR(" Enter text (end with ~):"));

do

{

if(sectorEndFlag == 1) //special case when the last character in previous sector was '\r'

{

transmitByte ('\n');

buffer[i++] = '\n'; //appending 'Line Feed (LF)' character

fileSize++;

}

sectorEndFlag = 0;

data = receiveByte();

if(data == 0x08) //'Back Space' key pressed

{

if(i != 0)

{

transmitByte(data);

transmitByte(' ');

transmitByte(data);

i--;

fileSize--;

}

continue;

}

transmitByte(data);

buffer[i++] = data;

fileSize++;

if(data == '\r') //'Carriege Return (CR)' character

{

if(i == 512)

sectorEndFlag = 1; //flag to indicate that the appended '\n' char should be put in the next sector

else

{

transmitByte ('\n');

buffer[i++] = '\n'; //appending 'Line Feed (LF)' character

fileSize++;

}

}

if(i >= 512) //though 'i' will never become greater than 512, it's kept here to avoid

{ //infinite loop in case it happens to be greater than 512 due to some data corruption

i=0;

error = SD\_writeSingleBlock (startBlock);

j++;

if(j == sectorPerCluster) {j = 0; break;}

startBlock++;

}

}while (data != '~');

if(data == '~')

{

fileSize--; //to remove the last entered '~' character

i--;

for(;i<512;i++) //fill the rest of the buffer with 0x00

buffer[i]= 0x00;

error = SD\_writeSingleBlock (startBlock);

break;

}

prevCluster = cluster;

cluster = searchNextFreeCluster(prevCluster); //look for a free cluster starting from the current cluster

if(cluster == 0)

{

TX\_NEWLINE;

transmitString\_F(PSTR(" No free cluster!"));

return;

}

getSetNextCluster(prevCluster, SET, cluster);

getSetNextCluster(cluster, SET, EOF); //last cluster of the file, marked EOF

}

getSetFreeCluster (NEXT\_FREE, SET, cluster); //update FSinfo next free cluster entry

error = getDateTime\_FAT(); //get current date & time from the RTC

if(error) { dateFAT = 0; timeFAT = 0;}

if(appendFile) //executes this loop if file is to be appended

{

SD\_readSingleBlock (appendFileSector);

dir = (struct dir\_Structure \*) &buffer[appendFileLocation];

dir->lastAccessDate = 0; //date of last access ignored

dir->writeTime = timeFAT; //setting new time of last write, obtained from RTC

dir->writeDate = dateFAT; //setting new date of last write, obtained from RTC

extraMemory = fileSize - dir->fileSize;

dir->fileSize = fileSize;

SD\_writeSingleBlock (appendFileSector);

freeMemoryUpdate (REMOVE, extraMemory); //updating free memory count in FSinfo sector;

TX\_NEWLINE;

transmitString\_F(PSTR(" File appended!"));

TX\_NEWLINE;

return;

}

//executes following portion when new file is created

prevCluster = rootCluster; //root cluster

while(1)

{

firstSector = getFirstSector (prevCluster);

for(sector = 0; sector < sectorPerCluster; sector++)

{

SD\_readSingleBlock (firstSector + sector);

for(i=0; i<bytesPerSector; i+=32)

{

dir = (struct dir\_Structure \*) &buffer[i];

if(fileCreatedFlag) //to mark last directory entry with 0x00 (empty) mark

{ //indicating end of the directory file list

//dir->name[0] = EMPTY;

//SD\_writeSingleBlock (firstSector + sector);

return;

}

if((dir->name[0] == EMPTY) || (dir->name[0] == DELETED)) //looking for an empty slot to enter file info

{

for(j=0; j<11; j++)

dir->name[j] = fileName[j];

dir->attrib = ATTR\_ARCHIVE; //settting file attribute as 'archive'

dir->NTreserved = 0; //always set to 0

dir->timeTenth = 0; //always set to 0

dir->createTime = timeFAT; //setting time of file creation, obtained from RTC

dir->createDate = dateFAT; //setting date of file creation, obtained from RTC

dir->lastAccessDate = 0; //date of last access ignored

dir->writeTime = timeFAT; //setting new time of last write, obtained from RTC

dir->writeDate = dateFAT; //setting new date of last write, obtained from RTC

dir->firstClusterHI = firstClusterHigh;

dir->firstClusterLO = firstClusterLow;

dir->fileSize = fileSize;

SD\_writeSingleBlock (firstSector + sector);

fileCreatedFlag = 1;

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR(" File Created! "));

freeMemoryUpdate (REMOVE, fileSize); //updating free memory count in FSinfo sector

}

}

}

cluster = getSetNextCluster (prevCluster, GET, 0);

if(cluster > 0x0ffffff6)

{

if(cluster == EOF) //this situation will come when total files in root is multiple of (32\*sectorPerCluster)

{

cluster = searchNextFreeCluster(prevCluster); //find next cluster for root directory entries

getSetNextCluster(prevCluster, SET, cluster); //link the new cluster of root to the previous cluster

getSetNextCluster(cluster, SET, EOF); //set the new cluster as end of the root directory

}

else

{

transmitString\_F(PSTR("End of Cluster Chain"));

return;

}

}

if(cluster == 0) {transmitString\_F(PSTR("Error in getting cluster")); return;}

prevCluster = cluster;

}

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to search for the next free cluster in the root directory

// starting from a specified cluster

//Arguments: Starting cluster

//return: the next free cluster

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long searchNextFreeCluster (unsigned long startCluster)

{

unsigned long cluster, \*value, sector;

unsigned char i;

startCluster -= (startCluster % 128); //to start with the first file in a FAT sector

for(cluster =startCluster; cluster <totalClusters; cluster+=128)

{

sector = unusedSectors + reservedSectorCount + ((cluster \* 4) / bytesPerSector);

SD\_readSingleBlock(sector);

for(i=0; i<128; i++)

{

value = (unsigned long \*) &buffer[i\*4];

if(((\*value) & 0x0fffffff) == 0)

return(cluster+i);

}

}

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to display total memory and free memory of SD card, using UART

//Arguments: none

//return: none

//Note: this routine can take upto 15sec for 1GB card (@1MHz clock)

//it tries to read from SD whether a free cluster count is stored, if it is stored

//then it will return immediately. Otherwise it will count the total number of

//free clusters, which takes time

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void memoryStatistics (void)

{

unsigned long freeClusters, totalClusterCount, cluster;

unsigned long totalMemory, freeMemory;

unsigned long sector, \*value;

unsigned int i;

totalMemory = totalClusters \* sectorPerCluster / 1024;

totalMemory \*= bytesPerSector;

TX\_NEWLINE;

TX\_NEWLINE;

transmitString\_F(PSTR("Total Memory: "));

displayMemory (HIGH, totalMemory);

freeClusters = getSetFreeCluster (TOTAL\_FREE, GET, 0);

//freeClusters = 0xffffffff;

if(freeClusters > totalClusters)

{

freeClusterCountUpdated = 0;

freeClusters = 0;

totalClusterCount = 0;

cluster = rootCluster;

while(1)

{

sector = unusedSectors + reservedSectorCount + ((cluster \* 4) / bytesPerSector) ;

SD\_readSingleBlock(sector);

for(i=0; i<128; i++)

{

value = (unsigned long \*) &buffer[i\*4];

if(((\*value)& 0x0fffffff) == 0)

freeClusters++;;

totalClusterCount++;

if(totalClusterCount == (totalClusters+2)) break;

}

if(i < 128) break;

cluster+=128;

}

}

if(!freeClusterCountUpdated)

getSetFreeCluster (TOTAL\_FREE, SET, freeClusters); //update FSinfo next free cluster entry

freeClusterCountUpdated = 1; //set flag

freeMemory = freeClusters \* sectorPerCluster / 1024;

freeMemory \*= bytesPerSector ;

TX\_NEWLINE;

transmitString\_F(PSTR(" Free Memory: "));

displayMemory (HIGH, freeMemory);

TX\_NEWLINE;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: To convert the unsigned long value of memory into

// text string and send to UART

//Arguments: 1. unsigned char flag. If flag is HIGH, memory will be displayed in KBytes, else in Bytes.

// 2. unsigned long memory value

//return: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void displayMemory (unsigned char flag, unsigned long memory)

{

unsigned char memoryString[] = " Bytes"; //19 character long string for memory display

unsigned char i;

for(i=12; i>0; i--) //converting freeMemory into ASCII string

{

if(i==5 || i==9)

{

memoryString[i-1] = ',';

i--;

}

memoryString[i-1] = (memory % 10) | 0x30;

memory /= 10;

if(memory == 0) break;

}

if(flag == HIGH) memoryString[13] = 'K';

transmitString(memoryString);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: to delete a specified file from the root directory

//Arguments: pointer to the file name

//return: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void deleteFile (unsigned char \*fileName)

{

unsigned char error;

error = convertFileName (fileName);

if(error) return;

findFiles (DELETE, fileName);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Function: update the free memory count in the FSinfo sector.

// Whenever a file is deleted or created, this function will be called

// to ADD or REMOVE clusters occupied by the file

//Arguments: #1.flag ADD or REMOVE #2.file size in Bytes

//return: none

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void freeMemoryUpdate (unsigned char flag, unsigned long size)

{

unsigned long freeClusters;

//convert file size into number of clusters occupied

if((size % 512) == 0) size = size / 512;

else size = (size / 512) +1;

if((size % 8) == 0) size = size / 8;

else size = (size / 8) +1;

if(freeClusterCountUpdated)

{

freeClusters = getSetFreeCluster (TOTAL\_FREE, GET, 0);

if(flag == ADD)

freeClusters = freeClusters + size;

else //when flag = REMOVE

freeClusters = freeClusters - size;

getSetFreeCluster (TOTAL\_FREE, SET, freeClusters);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## SystemDemo.c

//System Demo

//Authors: Dana Schultz, Katie Shea

//Created: April 9, 2012

//Revised: April 12, 2012

//Version: 2

//Uses Atmel ATmega1284 microcontroller to demonstrate the components of the system

//TFT LCD Connections

//Pin 1 - Read - Active Low (Pin 6)

//Pin 2 - Write - Active Low (Pin 5)

//Pin 20 - Register Select - 1=data, 0=command (Pin 4)

//Pin 21 - Chip Select - Active Low (Pin 15)

//Pin 19 - PORTD[5] - Reset - Active Low (Pin 16)

//Pin 22-29 [Port C]- Data (Pin 7-14)

//Touchscreen Connections

//Pin 5 - SS (AL) - Slave Select (Pin 15 - Chip Select)

//Pin 6 - MOSI - Master Out Slave In (Pin 14 - Din)

//Pin 7 - MISO - Master In Slave Out (Pin 12 - Dout)

//Pin 8 - SCK - SPI Clock (Pin 16)

//Pin 16 - INT0 - Ext. Interrupt (Pin 11) - Drops low for touch

//Encoder Connections

//Pin 17 - PORTD[4] - Right A (Level Shifter Pin 2) - Falling Edge Interrupt (INT1)

//Pin 18 - PORTD[5] - Right B (Level Shifter Pin 4)

//Pin 3 - PORTB[2] - Left A (Level Shifter Pin 6) - Falling Edge Interrupt (INT2)

//Pin 4 - PORTB[3] - Left B (Level Shifter Pin 10)

//Ultrasonic Sensor Connections

//Pin 40 - PORTA[0] - Left 1

//Pin 39 - PORTA[1] - Left 2

//Pin 38 - PORTA[2] - Left 3

//Pin 37 - PORTA[3] - Left 4

//Pin 36 - PORTA[4] - Right 1

//Pin 35 - PORTA[5] - Right 2

//Pin 34 - PORTA[6] - Right 3

//Pin 33 - PORTA[7] - Right 4

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Include Header Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iom1284pv.h>

#include <macros.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Function Declarations \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void);

void Initialize\_LCDPorts(void);

void Initialize\_LCDController(void);

void WriteCommand\_LCD(unsigned char);

void WriteData\_LCD(unsigned char);

void CommandWrite\_LCD(unsigned char,unsigned char);

void SendPixel\_LCD(unsigned char, unsigned char, unsigned char);

void WindowSet\_LCD(unsigned int,unsigned int,unsigned int,unsigned int);

void Monochrome\_LCD(unsigned char, unsigned char, unsigned char);

//void Square\_LCD(unsigned char, unsigned char, unsigned char);

void VariableRectangle\_LCD(unsigned int, unsigned int, unsigned int, unsigned int,

unsigned char, unsigned char, unsigned char);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touchscreen Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void);

void Initialize\_TouchPorts(void);

void Initialize\_InterruptINT0(void);

void Initialize\_SPI(void);

void Transmit\_SPI(char);

void HandleTouch\_ISR(void);

char GetX(void);

char GetY(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void Initialize\_Encoder(void);

void Initialize\_EncoderPorts(void);

void Initialize\_EncoderInterrupts(void);

void RightWheel\_ISR(void);

void LeftWheel\_ISR(void);

void UpdatePosition\_ISR(void);\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void);

void ReadUltrasonic\_ISR(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void);

void Delay\_ms(unsigned int num);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Demonstration Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Ultrasonic\_DrawWheelchair(void);

void Ultrasonic\_DrawSensors(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//INT0

#pragma interrupt\_handler HandleTouch\_ISR:2

//INT1

//#pragma interrupt\_handler RightWheel\_ISR:3

//INT2

//#pragma interrupt\_handler LeftWheel\_ISR:4

//Timer/Counter2 Overflow

//#pragma interrupt\_handler UpdatePosition\_ISR:12

//Timer/Counter0 Overflow

#pragma interrupt\_handler ReadUltrasonic\_ISR:19

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Global Variable Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Touchscreen Variables

unsigned int Touch\_X;

unsigned int Touch\_Y;

//Encoder Variables

/\*//Bottom left corner (SW corner) of the map equates to position (0,0)

unsigned int X\_position;

unsigned int Y\_position;

//Counters for encoder pulses

char pulses\_LW;

char pulses\_RW;

char lastpulses\_LW;

char lastpulses\_RW;

//Direction variables

char direction\_LW;

char direction\_RW;

char orientation;\*/

//Ultrasonic Sensor Data Variables

unsigned int Left1, Left2, Left3, Left4;

unsigned int Right1, Right2, Right3, Right4;

char SensorCounter;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Keyword Definitions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Wheel rotation direction definitions

#define NOMOTION 0

#define CLOCKWISE 1

#define COUNTERCLOCKWISE 2

//Left Wheel - CLOCKWISE = reverse motion, COUNTERCLOCKWISE = forward motion

//Right Wheel - CLOCKWISE = forward motion, COUNTERCLOCKWISE = reverse motion

//Wheelchair orientation definitions

#define NORTH 0

#define SOUTH 1

#define EAST 2

#define WEST 3

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

Initialize\_LCD();

Initialize\_Touchscreen();

//Initialize\_Encoder();

Initialize\_Ultrasonic();

Ultrasonic\_DrawWheelchair();

while(1)

{

Ultrasonic\_DrawSensors();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*LCD Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initializes LCD Ports and Controller and writes screen to blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCD(void)

{

Initialize\_LCDPorts(); //Initialize ports for LCD

Initialize\_LCDController(); //Initialize SSD1963 LCD Controller

Monochrome\_LCD(0x00,0x33,0xff); //Set LCD to medium blue

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize uC ports for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDPorts(void)

{

DDRC = 0xff; //Set Port C for output

DDRB |= 0x03; //Set Port B[0:1] for output

DDRD |= 0xe0; //Set Port D[5:7] for output

PORTC = 0x00; //Initialize data lines

PORTB |= 0x03; //Initialize command lines, set all command lines to 1

PORTD |= 0xe0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SSD1963 controller for TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_LCDController(void)

{

PORTD &= 0xdf; //RESET = 0

Delay\_ms(5);

PORTD |= 0x20; //RESET = 1

Delay\_ms(500);

WriteCommand\_LCD(0x01); //Software Reset

WriteCommand\_LCD(0x01);

WriteCommand\_LCD(0x01);

Delay\_ms(50);

CommandWrite\_LCD(0xe0,0x01); //START PLL

CommandWrite\_LCD(0xe0,0x03); //LOCK PLL

WriteCommand\_LCD(0xb0); //SET LCD MODE SET TFT 18Bits MODE

WriteData\_LCD(0x0c); //SET TFT MODE - 18-BIT, DISABLE COLOR DEPTH

//ENHANCEMENT,ENABLE TFT FRC,LATCH=RISING,

//LLINE POLARITY=AL,LFRAME POLARITY=AL

WriteData\_LCD(0x80); //SET TFT MODE

WriteData\_LCD(0x01); //SET horizontal size=320-1 HighByte

WriteData\_LCD(0x3f); //SET horizontal size=320-1 LowByte

WriteData\_LCD(0x00); //SET vertical size=240-1 HighByte

WriteData\_LCD(0xef); //SET vertical size=240-1 LowByte

WriteData\_LCD(0x00); //SET even/odd line RGB seq.=RGB

CommandWrite\_LCD(0xf0,0x00); //SET pixel data interface format=8bit

CommandWrite\_LCD(0x3a,0x60); //SET R G B format = 6 6 6

WriteCommand\_LCD(0xe6); //SET PCLK freq=6.4MHz; pixel clock frequency

WriteData\_LCD(0x00);

WriteData\_LCD(0xe7);

WriteData\_LCD(0x4f);

WriteCommand\_LCD(0xb4); //SET Horizontal Period

WriteData\_LCD(0x01); //SET HSYNC Total 440

WriteData\_LCD(0xb8);

WriteData\_LCD(0x00); //SET HBP 68

WriteData\_LCD(0x44);

WriteData\_LCD(0x0f); //SET VBP 16=15+1

WriteData\_LCD(0x00); //SET Hsync pulse start position

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET Hsync pulse subpixel start position

WriteCommand\_LCD(0xb6); //SET Vertical Period

WriteData\_LCD(0x01); //SET Vsync total 265=264+1

WriteData\_LCD(0x08);

WriteData\_LCD(0x00); //SET VBP=19

WriteData\_LCD(0x13);

WriteData\_LCD(0x07); //SET Vsync pulse 8=7+1

WriteData\_LCD(0x00); //SET Vsync pulse start position

WriteData\_LCD(0x00);

WriteCommand\_LCD(0x2a); //SET column address

WriteData\_LCD(0x00); //SET start column address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x01); //SET end column address=319

WriteData\_LCD(0x3f);

WriteCommand\_LCD(0x2b); //SET page address

WriteData\_LCD(0x00); //SET start page address=0

WriteData\_LCD(0x00);

WriteData\_LCD(0x00); //SET end page address=239

WriteData\_LCD(0xef);

WriteCommand\_LCD(0x29); //SET display on

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write single command to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteCommand\_LCD(unsigned char command)

{

PORTB |= 0x01; //RD = 1

PORTC = command;

PORTD &= 0xbf; //RS = 0 for command

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write data to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriteData\_LCD(unsigned char data)

{

PORTB |= 0x01; //RD = 1

PORTC = data;

PORTD |= 0x40; //RS = 1 for data

PORTB &= 0xfd; //WR = 0

PORTD &= 0x7f; //CS = 0

PORTD |= 0x80; //CS = 1

PORTB |= 0x02; //WR = 1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Write command with accompanying parameter to TFT LCD \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CommandWrite\_LCD(unsigned char REG,unsigned char VALUE)

{

WriteCommand\_LCD(REG);

WriteData\_LCD(VALUE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sends data for individual pixel to LCD - red, green, blue \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SendPixel\_LCD(unsigned char red, unsigned char green, unsigned char blue)

{

WriteData\_LCD(red);

WriteData\_LCD(green);

WriteData\_LCD(blue);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets current window on TFT LCD controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WindowSet\_LCD(unsigned int s\_x,unsigned int e\_x,unsigned int s\_y,unsigned int e\_y)

{

WriteCommand\_LCD(0x2a); //SET page address

WriteData\_LCD((s\_x)>>8); //SET start page address=0

WriteData\_LCD(s\_x);

WriteData\_LCD((e\_x)>>8); //SET end page address=319

WriteData\_LCD(e\_x);

WriteCommand\_LCD(0x2b); //SET column address

WriteData\_LCD((s\_y)>>8); //SET start column address=0

WriteData\_LCD(s\_y);

WriteData\_LCD((e\_y)>>8); //SET end column address=239

WriteData\_LCD(e\_y);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Monochrome\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x013f,0x0000,0x00ef);

WriteCommand\_LCD(0x2c);

for(x=0;x<=240;x++)

{

for(y=0;y<=320;y++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Sets every pixel on LCD to same color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void Square\_LCD(unsigned char datred, unsigned char datgreen, unsigned char datblue)

{

unsigned int x,y;

WindowSet\_LCD(0x0000,0x0028,0x0000,0x0028);

WriteCommand\_LCD(0x2c);

for(y=0;y<=40;y++)

{

for(x=0;x<=40;x++)

{

SendPixel\_LCD(datred,datgreen,datblue);

}

}

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Writes rectangle in selected area and color \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void VariableRectangle\_LCD(unsigned int startX, unsigned int endX, unsigned int startY,

unsigned int endY, unsigned char red, unsigned char green,

unsigned char blue)

{

unsigned int sizeX = endX - startX;

unsigned int sizeY = endY - startY;

unsigned int x,y;

WindowSet\_LCD(startX, endX, startY, endY);

WriteCommand\_LCD(0x2c);

for(y=0;y<=sizeY;y++)

{

for(x=0;x<=sizeX;x++)

{

SendPixel\_LCD(red,green,blue);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Touch Controller Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Touchscreen(void)

{

Initialize\_TouchPorts();

Initialize\_SPI();

Initialize\_InterruptINT0();

Touch\_X = 0;

Touch\_Y = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize PORTD for Touch Controller \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_TouchPorts(void)

{

DDRD &= 0xfb; //Set PORTD[2] for input - external interrupt INT0

PORTD &= ~0x04; //Disable pullup resistor PD2

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize External Interrupt INT0 - touch detected \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_InterruptINT0()

{

EIMSK = 0x00; //disable INT0

EICRA = 0x02; //set INT0 for falling edge trigger

TCCR0A = 0x00; //normal waveform generation mode

TCCR0B = 0x01; //no prescaling of clock source

EIMSK = 0x01; //enable INT0

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize SPI for ADS7843 Communication \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_SPI(void)

{

DDRB |= 0xb0; //Set SS (PB5), MOSI (PB6), and SCK (PB8) for output

SPCR = 0x53; //0x57 //Configure SPI Control Register

//SPIE = 0; SPE = 1; DORD = 0; MSTR = 1; CPOL = 0

//CPHA = 0; SPR1 = 1; SPR0 = 1

SPSR |= 0x01; //SPI2X = 1, doubles SPI clock speed when MSTR=1

PORTB |= 0x10; //Set SS low to activate ADS7843 as slave

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmits one byte of data via SPI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Transmit\_SPI(char data)

{

SPDR = data; //Transmit data

while(!(SPSR & (1<<SPIF))); //Wait for transmission to finish

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Interrupt Handler for INT0 - Touch on touchscreen \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HandleTouch\_ISR()

{

int i;

Monochrome\_LCD(0x00, 0x00, 0x00);

Touch\_X = 0;

Touch\_Y = 0;

//Collects touch coor data 8 times

for(i=0;i<8;i++)

{

Touch\_X += GetX();

Touch\_Y += GetY();

}

//Average 8 samples

Touch\_X = Touch\_X >> 3;

Touch\_Y = Touch\_Y >> 3;

EIFR = 0x01; //Clear interrupt flag

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines X coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetX(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b11011000); //X coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Determines Y coordinate of touch \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char GetY(void)

{

PORTB &= 0xef; //Enable slave

Transmit\_SPI(0b10011000); //Y coor, 8 bits, differential mode, PENIRQ enabled

Transmit\_SPI(0x00); //Receive coordinate from ADS 7843

PORTB |= 0x10; //Disable slave

return SPDR;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Encoder Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void Initialize\_Encoder(void)

{

X\_position = 512;

Y\_position = 512;

orientation = NORTH;

pulses\_RW = 0;

pulses\_LW = 0;

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

Initialize\_EncoderPorts();

Initialize\_EncoderInterrupts();

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Ports Associated with Encoders \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void Initialize\_EncoderPorts(void)

{

PORTD &= 0xe7; //Sets PORTD[3:4] for input - Right Wheel

PORTB &= 0xf3; //Sets PORTB[2:3] for input - Left Wheel

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize Encoder Interrupts INT1 and INT2 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void Initialize\_EncoderInterrupts(void)

{

//Initialize INT1 and INT2

EIMSK &= 0xf9; //disable INT1 and INT2

EICRA &= 0xeb; //set INT1 and INT2 for falling edge trigger

EICRA |= 0x28;

EIMSK |= 0x06; //enable INT1 and INT2

//Initialize Timer/Counter0 Overflow Interrupt

TCCR2B = 0x04; //Prescaler of 1/64 - will result in interrupt every 2.05 ms

TIMSK2 = 0x01; //Enable Timer/Counter2 Overflow Interrupt

asm("SEI"); //global interrupt enable

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Right Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void RightWheel\_ISR(void)

{

pulses\_RW++; //Increment Pulse Counter

//Determine direction of motion

if((PIND & 0x10) == 1)

{

direction\_RW = COUNTERCLOCKWISE;

}

else

{

direction\_RW = CLOCKWISE;

}

EIFR = 0x02; //Clear interrupt flag

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ISR for sensed falling edge on signal A from encoder on Left Wheel \*/

/\*Determines direction of motion, updates position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void LeftWheel\_ISR(void)

{

pulses\_LW++; //Increment Pulse Counter

//Determine direction of motion

if((PINB & 0x08) == 1)

{

direction\_RW = COUNTERCLOCKWISE;

}

else

{

direction\_RW = CLOCKWISE;

}

EIFR = 0x04; //Clear interrupt flag

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timed ISR - occurs every 2.05 ms \*/

/\*Uses directions and pulses from encoders to determine wheelchair position \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*void UpdatePosition\_ISR(void)

{

int pulses;

if((pulses\_RW == lastpulses\_RW) || (pulses\_LW == lastpulses\_LW))

{

//Wheelchair is not moving

direction\_LW = NOMOTION;

direction\_RW = NOMOTION;

}

else if((pulses\_RW < 3) && (pulses\_LW < 3))

{

//Not enough pulses to update position

}

else if((direction\_LW == CLOCKWISE) && (direction\_RW == CLOCKWISE))

{

//Turning Left - change of orientation will be handled in code for turning

}

else if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == COUNTERCLOCKWISE))

{

//Turning Right - change of orientation will be handled in code for turning

}

else

{

//Check for greater number of pulses

if(pulses\_LW > pulses\_RW)

{

pulses = pulses\_LW;

}

else

{

pulses = pulses\_RW;

}

//Update pulse counters

if(pulses == 3)

{

pulses\_RW = 0;

pulses\_LW = 0;

}

else

{

pulses\_RW -= 3;

pulses\_LW -= 3;

}

//Moving forward

if((direction\_LW == COUNTERCLOCKWISE) && (direction\_RW == CLOCKWISE))

{

if(orientation == NORTH)

{

Y\_position++;

}

else if(orientation == SOUTH)

{

Y\_position--;

}

else if(orientation == EAST)

{

X\_position++;

}

else

{

X\_position--;

}

}

else //Moving in reverse

{

if(orientation == NORTH)

{

Y\_position--;

}

else if(orientation == SOUTH)

{

Y\_position++;

}

else if(orientation == EAST)

{

X\_position--;

}

else

{

X\_position++;

}

}

}

//Store current pulse counts

lastpulses\_LW = pulses\_LW;

lastpulses\_RW = pulses\_RW;

}\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Ultrasonic Sensor Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize ports, interrupts and variables associated with ultrasonic sensors \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_Ultrasonic(void)

{

//Initialize Port

DDRA = 0x00;

//Initialize Variables

SensorCounter = 0;

Left1 = 0;

Left2 = 0;

Left3 = 0;

Left4 = 0;

Right1 = 0;

Right2 = 0;

Right3 = 0;

Right4 = 0;

//Initialize ADC

ADMUX = 0x00; //Select Channel 0 for dummy conversion

ADCSRA = 0xc3; //Enable ADC, single conversion,

//prescaler of 8 for accurate results

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

//Initialize Timer/Counter0 Interrupt

TCCR0B = 0x04; //Prescaler of 1/256 - will result in interrupt every 8.19 ms

TIMSK0 = 0x01; //Enable Timer/Counter0 Overflow Interrupt

asm("SEI"); //global interrupt enable

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Timer/Counter0 ISR - occurs every 8.19 ms \*/

/\*Initiates ATD Conversion and stores results \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadUltrasonic\_ISR(void)

{

//Temporary data storage variables

unsigned int datalow;

unsigned int datahigh;

//Perform ATD Conversion

ADCSRA |= 0x40; //Start single conversion

while(!(ADCSRA & 0x10)); //Wait for conversion to finish

ADCSRA |= 0x10; //Clear conversion ready flag

datalow = ADCL; //Read low data register first

datahigh = ADCH << 8; //Read high data register

switch(SensorCounter)

{

case 1:

Left1 = datahigh | datalow; //Store data for sensor Left1

ADMUX = 0x01; //Select Channel 1 for next conversion

break;

case 2:

Left2 = datahigh | datalow; //Store data for sensor Left2

ADMUX = 0x02; //Select Channel 2 for next conversion

break;

case 3:

Left3 = datahigh | datalow; //Store data for sensor Left3

ADMUX = 0x03; //Select Channel 3 for next conversion

break;

case 4:

Left4 = datahigh | datalow; //Store data for sensor Left4

ADMUX = 0x04; //Select Channel 4 for next conversion

break;

case 5:

Right1 = datahigh | datalow; //Store data for sensor Right1

ADMUX = 0x05; //Select Channel 5 for next conversion

break;

case 6:

Right2 = datahigh | datalow; //Store data for sensor Right2

ADMUX = 0x06; //Select Channel 6 for next conversion

break;

case 7:

Right3 = datahigh | datalow; //Store data for sensor Right3

ADMUX = 0x07; //Select Channel 7 for next conversion

break;

default:

Right4 = datahigh | datalow; //Store data for sensor Right4

ADMUX = 0x00; //Select Channel 0 for next conversion

SensorCounter = 0;

break;

}

SensorCounter++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Delay Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*125us delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_125us(void)

{

unsigned int i;

for (i=0; i<138; i++);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*num ms delay based on an 8MHz clock \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Delay\_ms(unsigned int num)

{

int i;

for (i=0; i<(num\*8); i++)

{

Delay\_125us();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Demonstration Functions \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Ultrasonic\_DrawWheelchair(void)

{

Monochrome\_LCD(0x00, 0x00, 0x00);

//Draw berry rectangle 80X140 pixels in center of screen

VariableRectangle\_LCD(120, 200, 50, 190, 0xff, 0xff, 0xff);

}

void Ultrasonic\_DrawSensors(void)

{

//Draw Sensor Left1

if((20<= Left1) && (Left1 < 30))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left1) && (Left1 < 40))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left1) && (Left1 < 50))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left1) && (Left1 < 60))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left1) && (Left1 < 70))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left1) && (Left1 < 80))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left1) && (Left1 < 90))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left1) && (Left1 < 100))

{

VariableRectangle\_LCD(120, 150, 10, 40, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(120, 150, 10, 40, 0xff, 0xff, 0xff);

}

//Draw Sensor Left2

if((20<= Left2) && (Left2 < 30))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left2) && (Left2 < 40))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left2) && (Left2 < 50))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left2) && (Left2 < 60))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left2) && (Left2 < 70))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left2) && (Left2 < 80))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left2) && (Left2 < 90))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left2) && (Left2 < 100))

{

VariableRectangle\_LCD(80, 110, 50, 80, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(80, 110, 50, 80, 0xff, 0xff, 0xff);

}

//Draw Sensor Left3

if((20<= Left3) && (Left3 < 30))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left3) && (Left3 < 40))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left3) && (Left3 < 50))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left3) && (Left3 < 60))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left3) && (Left3 < 70))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left3) && (Left3 < 80))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left3) && (Left3 < 90))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left3) && (Left3 < 100))

{

VariableRectangle\_LCD(80, 110, 160, 190, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(80, 110, 160, 190, 0xff, 0xff, 0xff);

}

//Draw Sensor Left4

if((20<= Left4) && (Left4 < 30))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left4) && (Left4 < 40))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left4) && (Left4 < 50))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left4) && (Left4 < 60))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left4) && (Left4 < 70))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left4) && (Left4 < 80))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left4) && (Left4 < 90))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left4) && (Left4 < 100))

{

VariableRectangle\_LCD(120, 150, 200, 230, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(120, 150, 200, 230, 0xff, 0xff, 0xff);

}

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0xff, 0xff);

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0xff, 0xff);

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0xff, 0xff);

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0xff, 0xff);

//Draw Sensor Right1

/\*if((20<= Left1) && (Left1 < 30))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left1) && (Left1 < 40))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left1) && (Left1 < 50))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left1) && (Left1 < 60))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left1) && (Left1 < 70))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left1) && (Left1 < 80))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left1) && (Left1 < 90))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left1) && (Left1 < 100))

{

VariableRectangle\_LCD(170, 200, 10, 40, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(170, 200, 10, 40, 0xff, 0xff, 0xff);

}\*/

//Draw Sensor Right2

/\*if((20<= Left1) && (Left1 < 30))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left1) && (Left1 < 40))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left1) && (Left1 < 50))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left1) && (Left1 < 60))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left1) && (Left1 < 70))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left1) && (Left1 < 80))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left1) && (Left1 < 90))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left1) && (Left1 < 100))

{

VariableRectangle\_LCD(210, 240, 50, 80, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(210, 240, 50, 80, 0xff, 0xff, 0xff);

}\*/

//Draw Sensor Right3

/\*if((20<= Left1) && (Left1 < 30))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left1) && (Left1 < 40))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left1) && (Left1 < 50))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left1) && (Left1 < 60))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left1) && (Left1 < 70))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left1) && (Left1 < 80))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left1) && (Left1 < 90))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left1) && (Left1 < 100))

{

VariableRectangle\_LCD(210, 240, 160, 190, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(210, 240, 160, 190, 0xff, 0xff, 0xff);

}\*/

//Draw Sensor Right4

/\*if((20<= Left1) && (Left1 < 30))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0x33, 0x66); //coral

}

else if((30<= Left1) && (Left1 < 40))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0xcc, 0x66); //light orange

}

else if((40<= Left1) && (Left1 < 50))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0xff, 0x00); //yellow

}

else if((50<= Left1) && (Left1 < 60))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0x33, 0xff, 0xcc); //aquamarine

}

else if((60<= Left1) && (Left1 < 70))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0x00, 0xcc, 0x66); //blue

}

else if((70<= Left1) && (Left1 < 80))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xcc, 0x66, 0xff); //violet

}

else if((80<= Left1) && (Left1 < 90))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0x66, 0xcc); //hot pink

}

else if((90<= Left1) && (Left1 < 100))

{

VariableRectangle\_LCD(170, 200, 200, 230, 0x66, 0xff, 0x33); //lime green

}

else

{

VariableRectangle\_LCD(170, 200, 200, 230, 0xff, 0xff, 0xff);

}\*/

}

## USART\_Test.c

//USARTTest

//Authors: Dana Schultz

//Created: March 31, 2012

//Revised: March 31, 2012

//Version: 1

//Uses two Atmel ATmega164 microcontrollers communicating via USART0

//Both uC transmit byte of data, receive data, and output received data to PORTA

//Include header files

#include <iom164pv.h>

#include <macros.h>

//Function Declarations

void Initialize\_USART0(void);

void USART0\_Transmit(char data);

char USART0\_Receive(void);

void main(void)

{

DDRA = 0xff; //Configure Port A for output

PORTA = 0x00;

Initialize\_USART0();

while(1)

{

USART0\_Transmit(0b10110110);

PORTA = USART0\_Receive();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Initialize USART0 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Initialize\_USART0(void)

{

UCSR0A = 0x20; //TXC0,U2X0, and MPCM0 are off.

UCSR0B = 0x18; //Enable Transmitter and Receiver. NO INTERRUPTS

UCSR0C = 0x06; //Asynchronous, 1 stop bit, parity disabled, and 8-bit data\*/

//Set baud to 9600 for 1 MHz clock (0x06) 8MHz (0x33)

UBRR0H = 0x00;

UBRR0L = 0x33;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmit one byte of data via USART \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void USART0\_Transmit(char data)

{

while((UCSR0A & 0x20) == 0x00); //Wait empty transmit buffer

UDR0 = data; //Send data

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Transmit one byte of data via USART \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char USART0\_Receive(void)

{

while((UCSR0A & 0x80) == 0x00); //Wait data to be received

return UDR0;

}

# Appendix B: Parts

## Cost Breakdown

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Supplier | Description | Part Number | Price | Quantity | Totals |
| Digikey | Voltage Regulator 3.3V | LM1085IT-3.3-ND | $2.29 | 3 | $6.87 |
|  | Voltage Regulator 5 V | LM1085IT-5.0-ND | $2.29 | 3 | $6.87 |
|  | Voltage Regulator 5 V | LM78L05ACZFS-ND | $0.48 | 3 | $1.44 |
|  | Female Serial Connector | 6E17C-009S-AJ-120-ND | $5.10 | 10 | $51.00 |
|  | Aluminum Box 200.2x120.19x84.51mm | HM441-ND | $20.40 | 1 | $20.40 |
|  | Level Shifter | MC14504BCPGOS-ND | $0.99 | 4 | $3.96 |
|  | Differential Amplifier | INA2134PA-ND | $4.76 | 4 | $19.04 |
|  | Aluminum Chassis | 377-1023-ND | $29.20 | 1 | $29.20 |
|  | 5.7" TFT LCD with Touchscreen | NHD‐5.7‐320240WFB‐CTXI#‐T‐1 | $85.00 | 1 | $85.00 |
|  | TI Four Wire Resistive Touch Controller | LM8300 | $11.35 | 1 | $11.35 |
|  | PCB Thruhole Adapter | NHD-FFC20-1 | $9.50 | 1 | $9.50 |
|  | Touchscreen Controller | ADS7843 | $5.32 | 1 | $5.32 |
| New Haven Display | LED Driver | NHD-5.7B | $10.00 | 1 | $10.00 |
| Backlight Connector | B 3B-XH-A | $2.00 | 1 | $2.00 |
|  | 20 POS FFC |  | $2.00 | 1 | $2.00 |
|  | 20-Pin IDC Interface Cable |  | $5.50 | 1 | $5.50 |
| Sparkfun | Level Shifter | BOB-08745 | $1.95 | 10 | $19.50 |
| Adafruit | MicroSD card breakout board |  | $15.00 | 1 | $15.00 |
|  |  |  |  |  | $303.95 |

## Additional Parts

These parts were purchased by previous research groups. They were used in our project but did not factor into our costs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Supplier | Description | Part Number | Price | Quantity | Totals |
| Digikey | Male-Male serial cables | AK129-2-R | 3.437 | 10 | 34.37 |
|  | Female serial connectors | 6E17C-009S-AJ-120 | 4.032 | 17 | 68.544 |
| Sparkfun | Ultrasonic Range Finder - XL - Maxsonar EZ3 | MB1330 | 44.96 | 10 | 449.6 |
| Automation Direct | Mechanical Shaft Encoder | TRD-S360-VD | 85.99 | 2 | 171.98 |
|  | Couplings | MCGL16-6-635 | 17.99 | 2 | 35.98 |
|  |  |  |  |  | 760.474 |