Automotive Rollover Detection and Early Warning Device Part Two

A Major Qualifying Project Report
Completed in Partial Fulfillment of Requirements for the
Bachelor of Science Degree in
Electrical and Computer Engineering at

Worcester Polytechnic Institute, Worcester, MA

Report Submitted by:

Kimberly Chinkidjakarn _____

Stefan Mrkic _____

Jonathan Vo _____

July 9, 2010

Report Submitted to the Faculty and Advisor:

Professor Robert Labonte, Major Advisor _____

Abstract

The greatest danger that drivers of off-road vehicles face is the risk of a vehicle rollover. Our goal was to create a device to effectively notify drivers of off-road vehicles when they are in danger of rolling over. By using a two axis accelerometer to measure the left to right tilt of the vehicle, we created a device to warn drivers both visually and audibly about their risk of rollover. Two banks of LEDs, each corresponding to either a left or right tilt, along with a speaker that sounds at an increasing frequency as the severity of the tilt increases, provide the driver with sufficient warning about when he or she is in danger of rolling over. The MSP430 microcontroller is used in the device to determine the angles that trigger the speaker and the lighting of each LED. Testing has shown that the device provides clear, effective notification about a vehicle's chance of rollover; however, additional features such as a second accelerometer to measure the front to back tilt of a vehicle could be added to increase the product's functionality.

Acknowledgements

We would like to thank everyone who helped us to complete our Major Qualifying Project.

First, we would like to thank our advisor, Professor Labonte, for taking time out of his summer to help us to not only formulate the idea for our project, but also for providing guidance and support throughout the design process.

We would also like to thank Tom Angelotti for helping us to find the many components we used in our device in the ECE shop.

Without the strong engineering foundation that we have developed during our time at WPI, we would not have had the knowledge to complete this project, so we would like to extend a big thank you to all of the ECE professors who we have had during our undergraduate careers.

Lastly, we would like to thank our families for supporting us during the time we spent on this project, and throughout our WPI experience. We would not have been able to do it without their help and encouragement.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

Executive Summary

Drivers of off-road vehicles face the danger of rollover every time that they take their vehicles onto uneven terrain. Since off-roaders most often drive older vehicles, there are not many built-in features, such as Electronic Stability Control (ESC), to help prevent a rollover. Rather than invest in aftermarket ESC, a device that can warn a driver when he or she is driving dangerously is a more affordable and practical way to help drivers to prevent a rollover.

The device that we have created effectively notifies drivers when they are dangerously close to rolling over. Our product, unlike the competition, provides drivers with an accurate, yet intuitive, warning system that allows the driver to see whether his or her vehicle is tilted at an angle that increases the risk of rollover. The design of our product is based on product requirements that were formulated after market research was conducted. This research helped us to determine the characteristics that we wanted to incorporate into our device. After testing our product on a model of an off-road vehicle, our product was shown to respond correctly to the tilting of a vehicle and provide the driver with clear notification about how close he or she is to rolling over based on the angle at which the vehicle is tilted.

1.0 Introduction

The purpose of the Major Qualifying Project (MQP) is to provide students with the opportunity to work through an engineering design project from beginning to end. The first step in the design process was to clearly define our project. Our project is a continuation of the Automotive Rollover Detection and Early Warning Device project completed this past school year, so we began by brainstorming how we could improve the existing device. Whereas the existing product is designed to work in vehicles driving under normal conditions, we decided to create a rollover warning and detection system for off-road vehicles in an effort to create a device that could more accurately determine a driver's chance of rollover. Defining the project allowed us to determine our target market and appropriate market and background research was then conducted. After performing this research, it was possible for us to determine the essential features of our design. We decided that our product must provide both an audio and visual warning for the driver, be portable, and be easy to install and use. Once these design characteristics were established, the initial construction of our prototype began. Construction, testing, and debugging of our product eventually led to the construction of a final prototype that can be demonstrated on the testing apparatus that we have created. Our final product demonstrates proof of concept and could be used as is, however there are a number of features that can be incorporated into the product in the future to add to its functionality.

1.1 Problem Statement

Both recreational and competitive off-roaders face the risk of a possible rollover every time they participate in the activity. This type of accident has the potential to not only damage the vehicle involved, but can also cause serious injury and even death. The

creation of a rollover warning and detection device designed specifically for off-roading will hopefully provide users with an effective way of knowing when they are close to a rollover and will allow them to correct their driving appropriately.

2.0 Methodology

To achieve our goal of constructing a device that can warn drivers of off-road vehicles when they are driving at an angle that leaves them dangerously close to a rollover, the following steps were taken:

- Research about off-roading and vehicle rollovers was conducted
- Existing devices designed to warn off-road drivers of their risk of rollover were examined
- A prototype was constructed that had the features we determined to be essential based on our market research
- The prototype was tested on the testing rig that we constructed
- A final working design was created

2.1 Market Research

General background research about off-roading was conducted to learn more about the sport and to determine if there truly was a need for a product such as the one that we hoped to create. It was then important to research existing products similar to the device we wanted to construct to determine characteristics that would be desirable in our device. After looking at the different products, it was possible for us to decide which aspects of the existing products would be important to include in our design. We were also able to determine from this research what improvements and additional features we could add to our product to make it competitive in this niche market.

2.2 Construction of a Prototype

Prior to creating a final product, it was necessary to test each of the systems that make up the functional block diagram of our product. Every system was tested and debugged individually before being integrated to create a functional, breadboarded prototype. This prototype, although aesthetically unattractive, was a proof of concept and allowed us to ensure that each portion of our design worked properly both individually and together.

2.3 Development of a Final Design

The breadboarded version of our device was turned into a device that could easily be installed into an off-road vehicle. The final product met not only the functional requirements that we had developed, but also met the requirements of being portable and easy to use. Although the device is still a prototyped version of an off-road rollover warning and detection system, it could be used by off-road drivers to effectively warn the driver when he or she is at risk of a rollover.

3.0 Background Information

Prior to the construction of a prototype, it was important to conduct background research. First, we reviewed the rollover warning and detection system that was completed by Erik Ryll, Tyler Golightly, and Shannon Leary during the past school year. Additional research consisted of learning more about the sport of off-roading and investigating existing products similar to the one that we hoped to build. It was also important to determine the best way to calculate an off-road vehicle's risk of rollover. Lastly, it was necessary to gain knowledge about how an accelerometer works since this

was the main component in our design and would be the part of our circuit that would determine the amount a vehicle is tilting.

3.1 The Prior Team's Project

The team who completed the original automobile rollover warning and detection system allowed us to have access to their device and report. Figure 1 shows the completed project, and Figure 2 shows the schematic of the device.

Figure 1-The First Team's Completed Device[1]

[1] Ryll, Erik, Tyler Golightly, and Shannon Leary. "Automotive Rollover Detection and Early Warning Device." 29 April 2010. p. 38

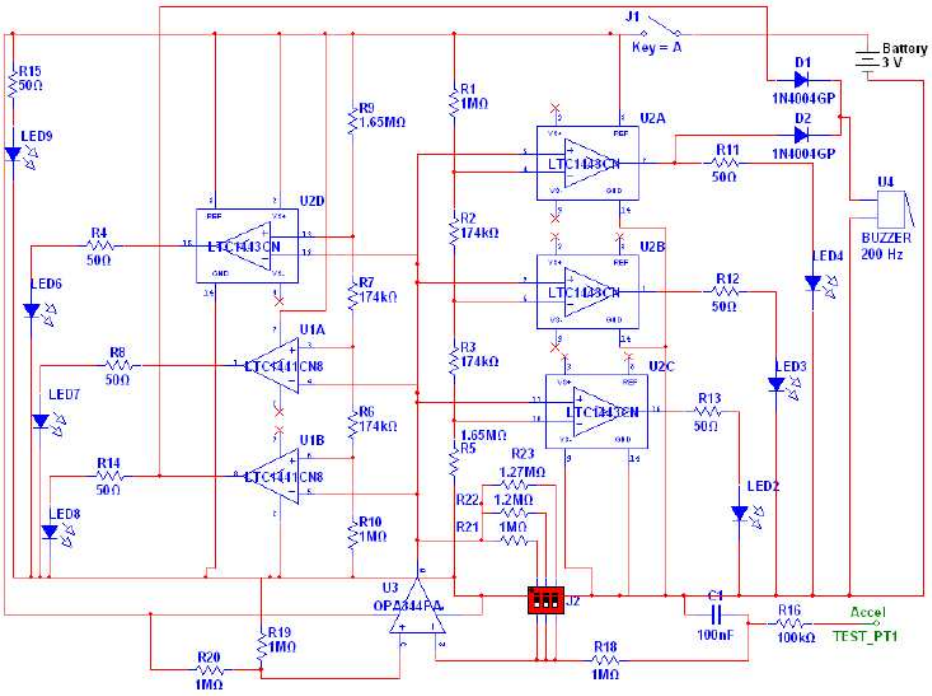


Figure 2-The First Team's Schematic[2]

The completed product was marketed to the drivers of average vehicles. The design involved the use of six LEDs to warn the driver of a vehicle if he or she was driving dangerously. A buzzer sounded when the driver was at an angle that triggered the final LED, indicating a very high danger level. The final product is small, portable, and relatively low cost; all of which are characteristics that would make the device attractive to customers. By looking at the internal circuitry of the device, seen in Figure 2, it is possible to see that the design uses only analog components. The LEDs are controlled by comparators, which decide the severity of the tilt based on the voltage produced by the accelerometer. Another aspect of the design that should be noted is that it is battery powered.

[2] Ryll, p. 35

In the team's report, several recommendations regarding how the product can be improved were made. It is suggested in the report that a charging circuit be added in order for the batteries to last longer; calculations showed that the batteries would need to be replaced often if the device was turned on every time a driver was in his or her car. To improve the way that the device warns the driver, it was recommended that a sound alarm be incorporated to beep at every level of danger, not just for the final LED. The group also suggested that this detection system be integrated into something that a driver may already have in the car, such as a GPS.

Reviewing the work of the prior team provided us with information about the device, and also helped us to begin the design process. Taking the recommendations of the group into account, we decided to make our device plug into the cigarette lighter of a car, eliminating the problem of battery life. We also decided that having a speaker that would sound accordingly based on the level of rollover risk would be an important addition to the design. After looking at the schematic of their device, we decided that rather than use comparators, we would use a microcontroller to control the LEDs, allowing us to more easily adjust when the driver would be warned by our device. Lastly, we decided to alter our target market; we decided to design our product for drivers of off-road vehicles rather than for drivers of common vehicles. The smaller market allowed us to create a device that could calculate the risk of rollover more precisely than if the device had been marketed to consumers who could place the device in vehicles ranging from sedans to pick-up trucks.

3.2 Dangers of Off-Roading

The greatest danger that a driver faces when off-roading is the risk of rollover.[3] One of the main reasons that this is so is because of the types of vehicles that are used in off-roading; off-roaders tend to drive vehicles that have a short wheel-base and a high center of gravity. These vehicles have up to five times the rollover rate of a typical passenger vehicle. Furthermore, when these vehicles are driven in rural areas, such as those frequented by off-roaders, the rate of rollover increases. It has been documented that rollovers are a greater problem when drivers are in a rural area than when they are in an urban setting.[4]

The terrain on which off-roading occurs is a major reason why rollovers are such a large problem. People drive off-road on a variety of surfaces; however research has shown that all of these surfaces raise the chance of a rollover by destabilizing a vehicle. Off-road surfaces, whether they are soft like grass or hard like a dirt surface, are greater initiators of rollovers than the pavement used on highways. In a study conducted in the UK, 31% of the rollovers investigated involved off-road vehicles. This is a significantly higher percentage than that of sports cars, which were involved in the second most number of rollovers. Sports cars were involved in less than 20% of the rollover accidents investigated.[5]

---

[3] McNulty, Kevin. "4x4 Trucks Rollover Accidents Preparation - Rollover Reality: Always Prepare For The Worst Before Hitting the Trail." Peterson's 4Wheel & Off-Road. April 2009. Accessed 7 June 2010. <http://www.4wheeloffroad.com/features/131_0904_4x4_trucks_rollover_accident_preparation/index.html>

[4] Richardson, Shane. "Engineering Solutions. Rollover: One of the Road Safety Problems that is not being addressed." Road Safety Toward 2010, pp. 48-50. Accessed 15 June 2010. <http://www.dvexperts.net/pubs/rollover.pdf>

[5] Cuerden, Richard, Rebecca Cookson, and David Richards. "Car Rollover Mechanisms and Injury Outcome." p. 2. Accessed 15 June 2010. < http://www-nrd.nhtsa.dot.gov/pdf/esv/esv21/09-0481.pdf >

It is not uncommon for drivers of off-road vehicles to be involved in multiple rollover accidents. Some, such as competitive off-roader Brad Lovell, have been involved in excess of two dozen rollovers. Rolling over an off-road vehicle is so common that it is included as one of the statistics that these competitive drivers list in their biographies.[6] Although two dozen rollovers may sound extreme, it can be deduced that if an experienced driver can be involved in this many rollovers, recreational off-roaders most likely are involved in these types of accidents as well. Less experienced drivers are unlikely to have the skills and knowledge required to protect them in this type of accident, making the rollovers even more dangerous.

3.3 Market Research

After deciding that we were going to construct a rollover warning and detection system designed specifically for off-road vehicles, we needed to determine if a market for our product existed, and if so, we needed to learn more about the market that we were targeting. To do this, we conducted a large amount of internet research and also spoke to someone who is involved in the sport of off-roading to determine if there was interest in a device such as the one we were going to create.

3.3.1 Research Method

We began our market research by joining an internet forum for people involved in off-roading. This allowed us to learn more about the sport. Internet research was also conducted to see if there were products designed specifically to warn drivers of off-road vehicles about their risk of rollover.

[6] "Stats and Info." Lovell Rock Racing. Accessed 7 June 2010. <http://lovellrockracing.com/stats_info.php>

In addition to internet research, we also spoke in person with a girl who participates in off-roading. From this conversation, we hoped to see if there was interest in a rollover warning and detection device by people in the off-roading community. Both the internet research and interview provided us with integral information about the market that we are gearing our product toward.

3.3.2 Research Results

After we conducted our research, we found that there was definitely a market for our product. We found that there are similar products for sale that are meant to warn off-road drivers about their risk of rollover. The first example, which is sold for $24.50, can be seen in Figure 3.

Figure 3- Dash Mounted Inclinometer[7]

The second example, which is more of a high end device priced at $179, can be seen in Figure 4.

[7] "Off-Road Inclinometer." Wheeler's Off-Road, Inc. Accessed 6 June 2010. <http://www.wheelersoffroad.com/inclinosam.htm>

Figure 4- RAD-1 Digital Audio Inclinometer[8]

Each of the two types of rollover detection and warning devices have positive and negative aspects. The Dash Mounted Inclinometer is inexpensive and easy to use; however, there is no audio warning that goes along with the device. This device probably would not effectively gain the attention of the driver, so the driver would not know that he or she was in danger of rolling over until it was too late. The more expensive RAD-1 Digital Audio Inclinometer has many of the features that we wanted to include in our design. It has both an audio and visual warning system to gain the driver's attention. It also is small, and seems to be easy to use. The device has the capability to measure a left or right tilt (roll), as well as a front or back tilt (pitch), and can also be programmed by the user. The main drawback to this device is that the display only shows the number of degrees that the car is tilting. The average user of a device may not have an appreciation for tilt angles, and therefore may not know the critical angle that will result in a rollover. Ideally, we wanted the product that we created to have the best aspects of the above devices.

Internet research also yielded information about the most popular cars used in off-roading. We found that the Hummer H1, Land Rover Discovery, Jeep Wrangler, Toyota Tacoma, and the Chevy Tracker were the most commonly used vehicles. From our

[8]" RAD-1 Digital Audio Inclinometer." Accessed 6 June 2010. <http://rad1.com/>

background research about rollovers, we knew that cars with a short wheel-base and high center of gravity were the most likely to rollover. This eliminated the Hummer from the list of cars that we would be targeting. We also knew that the Land Rover has many built in features that help to prevent rollovers, so we also decided that we would not target drivers of those cars. Knowing that Jeep Wranglers are very popular vehicles, we decided that the calculations that we would be making during the design and construction of our device would be based on this vehicle.

After deciding that our device would target drivers of the Jeep Wrangler, we interviewed a woman who owns a lifted Wrangler and participates in off-roading. When asked whether or not she would be interested in a rollover warning and detection system for her off-road vehicle, she said, “something like that would be awesome. I really need something to give me some warning ‘cause I flipped my old jeep. I know that my friends would also like to use it." This further proved that there was a market for our product.

It is clear from our research that as the number of SUVs sold has increased, so has the number of people who go off-roading. As the popularity of off-roading continues to grow, it is important that drivers have tools to help them prevent dangerous, and potentially fatal, rollovers.

3.3.3 Product Requirements

The research that we conducted produced the following product requirements:

Affordable

Based on the market research that was conducted, we determined that in order for our product to be competitive, it would have to be reasonably inexpensive. The high end products that are similar to the one that we have constructed cost $179. Since our

product possesses many of the features of the more expensive inclinometer, it is imperative that we be able to sell our product for under $200. Although the cost of our prototype may cause the profit margin to seem small if the device is sold for this amount, mass production would allow for the price of the components to decrease, lowering the overall construction cost of the device.

Intuitive

We felt that it was important that our device be intuitive to use. Drivers of off-road vehicles may not have the technical knowledge to install a complicated warning system, so it was necessary to make sure that our product could be easily understood by the layperson. We achieved this goal by designing the product so that there is a minimum amount of set-up required by the driver.

Portable

Drivers of off-road vehicles may have more than one car that they like to take off-road; therefore, it was important to make our device portable so that it could be moved from vehicle to vehicle. Our device is not something that has to be permanently installed in a vehicle, and it can be easily dismantled and assembled. This allows the user to take the device with them no matter which car they decide to drive.

Effective Notification System

It was critical that our device have an effective warning system that would catch the attention of the driver. In addition to a visual warning system in the form of LEDs that indicate the severity of the tilt, an audio system was installed to warn the driver. A driver probably would not notice the lighting of a single LED on its own; however having

a speaker sound in conjunction with the lighting of the LED will cause the driver to realize when there is a change in the degree of tilt of the vehicle.

Aesthetically Pleasing

When a product is sold to consumers, it is necessary for it to look attractive. Customers are naturally drawn to products that are aesthetically pleasing, so it was important for the components of the device to be enclosed in a manner that would make the product more visually appealing to customers. A product can be fully functional, but if it is not encased properly, it will not appeal to consumers.

3.4 Determining a Driver's Risk of Rollover

The risk of rollover is related to the center of gravity of the vehicle. By determining the center of gravity, it is possible to calculate the angle at which a vehicle will rollover. The device that we constructed has pre-set angles that correspond to the lighting of a specific LED. The standard measure of a vehicle's risk of rollover is the Standard Stability Factor (SSF). This measurement determines how top-heavy a vehicle is: the higher the calculated the SSF, the less likely a vehicle is to rollover. The SSF is found using Equation 1.

$$SSF = T/2H$$

Equation 1-Calculation of the SSF

In this equation, T is the track width of the vehicle, and H is the height of the center of gravity. The track width is the distance between the centers of the right and left tires along the axle, while the center of gravity is measured in a laboratory to determine the height above the ground of the vehicle's mass.[9] Although the SSF is how the government rates rollover risk, it is not applicable to our project. The SSF assumes that the vehicle is

[9] "Motorvehicle Rollover-FAQs," Accessed 28 May 2010. <http://buzzmotors.com/rolloverfaqs.html>

on a flat, smooth road, which is clearly not the conditions under which our product will be used.[10] Therefore, we calculated the pre-set values in our device by following the procedure below. This process to determine a rollover angle was designed specifically for off-road Jeeps.

In order to calculate the angle at which an off-road vehicle will rollover, the following parameters are needed:

- LWB: The length of the wheelbase (measured from axle center to axle center)
- LTB: The track width (measured from outer tire edge to outer tire edge)
- HF1: The height of the center of the front axle hub from the level ground
- HF2: The height of the center of the front axle hub from the ground while the front is elevated at least 24 inches
- Wt: The total weight of the vehicle
- WF: The weight on the front axle when the vehicle is level
- WP: The weight on the passenger side
- WR1: The weight on the rear axle when the vehicle is level
- WR2: The weight on the rear axle when the front of the vehicle is elevated

First, it is necessary to calculate the wheelbase center of gravity (WBCG). This is where the center of gravity lies in relation to the wheelbase of the vehicle. Equation 2 shows how to find where the WBCG is located, in inches, behind the vehicle's front axle.

$$WBCG = (1-(WF/Wt))*LWB$$

Equation 2-Wheelbase Center of Gravity

[10] Hall, Jonathan, and Jerry Magraw. "Modeling Vehicle Rollover." The Physics Teacher. Volume 43 (2005), p. 51. Accessed 28 May 2010. <http://www.suu.edu/faculty/penny/RolloverPaper/Modeling%20vehicle%20rollover(Magraw-Hill%20paper).pdf>

Second, the wheel track center of gravity (WTCG) can be calculated using the track width, weight on the passenger side, and the total weight of the Jeep. Equation 3 shows how to calculate the location of the WTCG in inches from the outer tire edge on the passenger side of the vehicle.

$$WTCG = (1-(WP/Wt))*LTB$$

Equation 3- Wheel Track Center of Gravity

Next, the height of the vehicle's center of gravity (HTCG) can be determined. This calculation is a bit more involved than the prior two center of gravity calculations. It requires the height difference between the front axle when it is level and when it is elevated (HFd), the length of the wheelbase when the vehicle is elevated (LWBn), and the difference in the weight of the rear axle when the vehicle is level and when the front is elevated (WRd). Equations 4, 5, and 6 show the calculations for HFd, LWBn, and WRd respectively. Equation 7 shows how these values can be used to calculate the height of the center of gravity.

$$HFd = HF2 - HF1$$

Equation 4- Height Difference Between the Front Axle Level and Elevated

$$LWBn = \sqrt{LWB^2 - HFd^2}$$

Equation 5- Length of the Shortened Wheelbase when Elevated

$$WRd = WR2 - WR1$$

Equation 6- Weight Added to the Rear Axle

$$HTCG = HF1 + ((WRd * LWB * LWBn) / (Wt * HFd))$$

Equation 7- Height of the Center of Gravity

The wheelbase center of gravity, the wheel track center of gravity, and the height of the center of gravity are the x, y, and z coordinates respectively of where the center of

gravity is located in the vehicle. This is the point where the average weight of the vehicle is located. The vehicle will rollover if this point is no longer within the support structure formed by the tires of the vehicle; therefore the above values can be used to calculate the rollover angles. For our purposes, we were concerned solely with the calculation of the driver side and passenger side rollover angles. The document also shows how the rearward and forward rollover angles can be calculated, but our device is not meant to warn a driver of these angles. Equation 8 shows how to calculate, in degrees, the driver side rollover angle (DSROA).

$$DSROA = Degrees\ (arctan\ ((LTB - WTCG) / HTCG))$$

Equation 8- Driver Side Rollover Angle

Equation 9 demonstrates how the passenger side rollover angle (PSROA) can be calculated in degrees.

$$PSROA = Degrees\ (arctan\ (WTCG/HTCG))$$

Equation 9-Passenger Side Rollover Angle

The driver side and passenger side rollover angles are the critical angles that we used in our device as the point when the vehicle will roll either to the left or to the right. The paper explaining these calculations expresses that these angles are accurate, but do not necessarily take into account when off-road conditions cause the tires to "tuck" under the wheels of the vehicle. This phenomenon causes the track width of the vehicle to shorten and the height of the center of gravity to become lower. These factors will decrease the

angle at which a vehicle will rollover.[11] In order to compensate for this, we built in a safety factor into our calculated angles.

The pre-set values that we calculated will give the driver an accurate, but not exact, indication of when his or her vehicle will rollover; however we also programmed the device so that the user can enter specific parameters that can be used to calculate the exact critical rollover angle for their vehicle. At this time, it is not possible for the calculation to be completed using the user-entered parameters because the compiler that we are using to write our code has a size limit that allows for the parameters to be entered, but will not allow for the calculation of the angle to be done. The microcontroller itself has the capacity required to calculate the angle, so in the future, it should be possible to include this aspect in our design. By allowing the driver to enter specific parameters, drivers will be able to either install our device and use the pre-set values, or take the time to customize the device for their vehicles in order to provide a more accurate rollover angle.

3.5 Accelerometer Information

An accelerometer is a device that is used to measure acceleration. To understand how an accelerometer works, it is important to define acceleration: acceleration is the change in velocity with respect to a unit of time. The common units of acceleration are ft/s^2 or m/s^2; the specifications of accelerometers, however, are given in a unit of "g." A "g" is a unit of acceleration that is equal to the Earth's gravity at sea level (32 ft/s^2 or 9.8 m/s^2). An accelerometer outputs a voltage that is proportional to the acceleration, or the change in velocity with respect to time, that has been measured.

---

[11] "Determine Center of Gravity, and Roll-Over Angles on Your Vehicle." Jeepaholics Anonymous. 13 Jan 2002. Accessed 28 May 2010. <http://www.jeepaholics.com/tech/cog/>

There are several types of accelerometers. The following are the most common types of accelerometers along with a brief description of how they work:

- Capacitive: A metal beam or micromachined feature produces a capacitance that changes proportionally to the acceleration
- Piezoelectric: A piezoelectric crystal, which produces an electric field in response to an applied mechanical stress, is mounted to a mass. A voltage is produced that is then converted to acceleration
- Piezoresistive: A beam or micromachined feature has a resistance that changes with acceleration
- Hall Effect: Motion is converted to an electrical signal by the sensing of changing magnetic fields. The voltage produced is proportional to the acceleration
- Magnetoresistive: The resistivity of a material changes in the presence of a magnetic field. The resistivity is proportional to the acceleration
- Heat Transfer: The location of a heated mass is tracked during acceleration by the sensing of temperature

After reviewing the different types of accelerometers, we decided to choose a capacitive accelerometer. This type of accelerometer has a high level of accuracy, but is relatively low cost.[12] Specifically, we decided to purchase an accelerometer based on the Analog Devices ADXL320. This accelerometer measures static acceleration, such as gravity, which allows it to be used to measure tilt. It uses polysilicon springs to suspend a polysilicon surface-micromachined structure over a silicon wafer. The springs provide resistance against acceleration forces. A differential capacitor, made up of independent

[12] "Accelerometers and How they Work." Texas Instruments. Accessed 7 June 2010. <http://www2.usfirst.org/2005comp/Manuals/Acceler1.pdf>

fixed plates attached to the moving object, measures movement in the structure. The plates of the capacitor are driven by 180° out-of-phase square waves. Acceleration causes the capacitor to become unbalanced, producing a square wave with amplitude that is proportional to the acceleration of the object. Demodulation methods are then used to determine the direction of acceleration.[13]

3.6 Design Approach

The functional block diagram outlining the overall structure of the design of our product can be seen in Figure 5.

[13] "ADXL320." Analog Devices, p. 11. Accessed 24 June 2010. <http://www.dimensionengineering.com/datasheets/ADXL320.pdf>

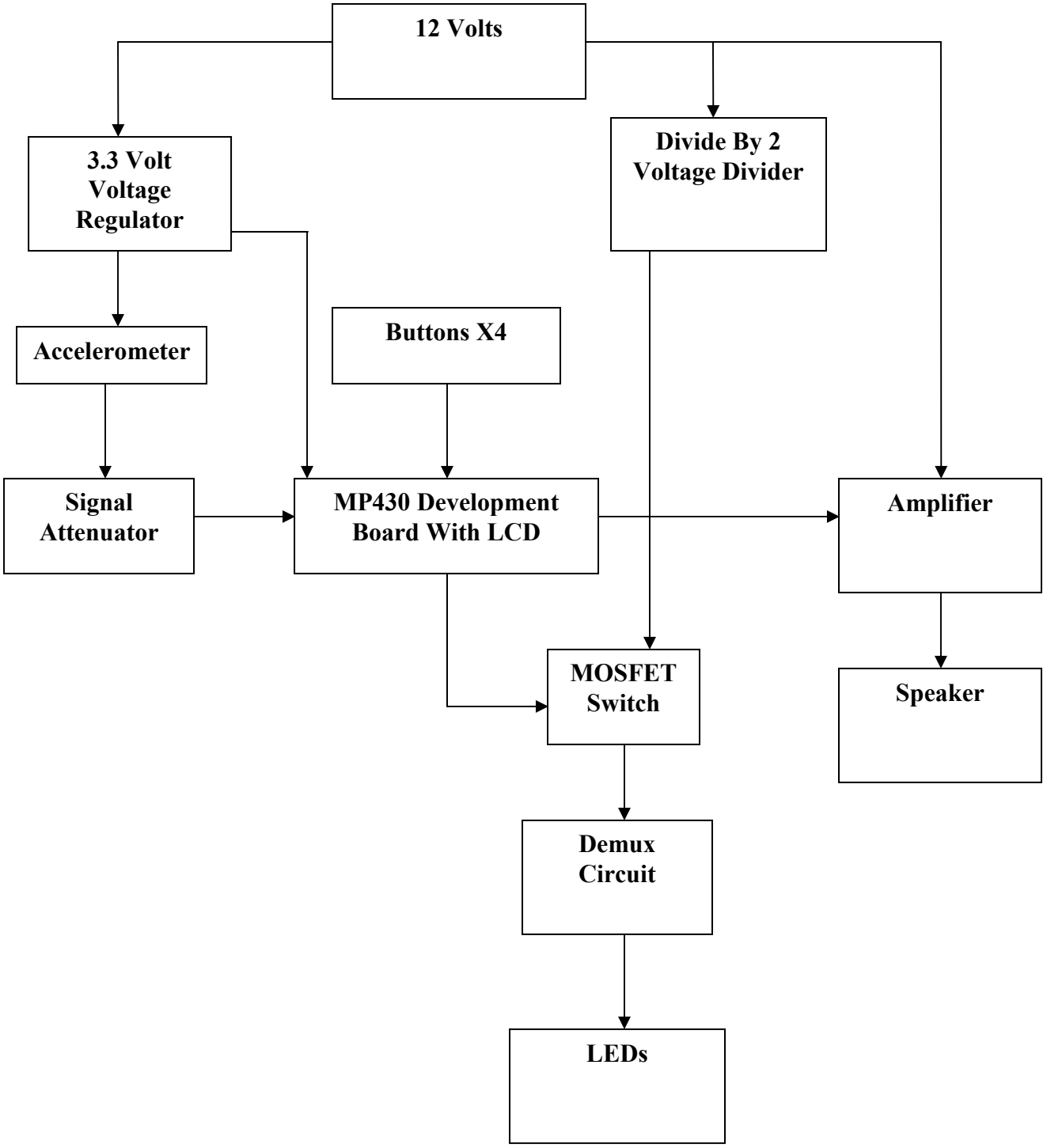


Figure 5-Functional Block Diagram

The 12 volt power supply, which is supplied by the car battery and is accessed via the vehicle's cigarette lighter, provides the power for the system. A voltage regulator is used to provide the MSP430 development board and the accelerometer with a constant voltage of 3.3 volts. The output of the accelerometer is attenuated before connecting to the analog to digital converter (ADC) of the MSP430 board. The buttons of the MSP430 can be used by the driver to enter the parameters of his or her car, which can then be used to calculate the exact angle when the vehicle is in danger of a rollover. The MSP430 itself is used to control the LEDs and speaker. A detailed description of each portion of the diagram follows.

3.6.1 Voltage Regulator

The voltage regulator is the module in our design that converts the voltage coming in from the car battery to a voltage that is usable by the MSP430 and accelerometer. The car battery provides 12 volts, however the MSP430 requires an input between 1.8 and 3.6 volts. To perform this conversion, a constant 3.3V voltage regulator with a high current limit was chosen.

It was decided that a voltage regulator with an output of 3.3 volts would be a good choice for our design because the output not only falls within the acceptable input range of the MSP430 and the accelerometer, but it is one of the standard voltages used in logic applications. The regulator that was chosen for our design was the LT1084CT, seen in Figure 6.

Figure 6-Voltage Regulator[14]

The voltage regulator has three pins: a voltage input, a voltage output, and ground. The 12 volt power supply is connected to Vin, while Vout will provide a steady 3.3 volt supply.

3.6.2 Accelerometer

The accelerometer is the part of the design that determines if the vehicle is tilting, and whether it is tilting to either the left or the right. The DEACCM5G was chosen because it is compatible with the microcontroller that we chose to use in our design and already has protection circuitry built in to protect the accelerometer. This accelerometer can be seen in Figure 7.

[14] "LT1084CT-3.3" Digikey. Accessed 19 June 2010. <http://media.digikey.com/photos/Linear%20Tech%20Photos/LT1084CT-3.3%23PBF.JPG>

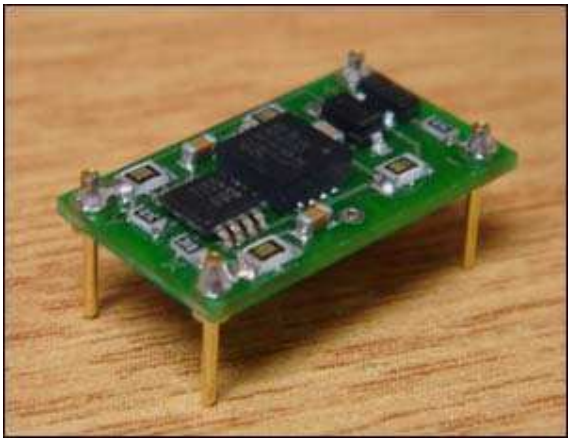


Figure 7- Accelerometer[15]

It also draws little current and has a low operating voltage, making it a low power device. This accelerometer is a two axis device, with X and Y outputs. We currently are using just one output to identify a left or right tilt; in the future, it will be possible to use the other output to measure a front to back tilt.

3.6.3 Voltage Signal Attenuator

The voltage signal attenuator converts the voltage signal produced by the accelerometer to a voltage that the ADC of the MSP430 development board can handle. When given an input of 3.3 volts, the accelerometer can produce a maximum voltage of 1.85 volts. The ADC of the MSP430, however, can handle an input ranging from 0V-1.5V. By configuring the attenuator to divide the voltage signal in half, the output will be within the range that can be handled by the ADC. Equation 10 was used to determine the resistor values needed to achieve this functionality.

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in}$$

Equation 10-Attenuator Equation

[15] "DE-ACCM5G." Dimension Engineering .Accessed 19 June 2010. <http://www.dimensionengineering.com/DE-ACCM5G.htm>

Both R1 and R2 were chosen to be 20k ohm. An op-amp connected in the unity gain configuration was added to the voltage divider to ensure that the attenuated value would remain constant. The op-amp that we used for this function, the LM358, was chosen because it can operate on a rail as low as 3 volts. The output of this circuit is connected directly to the ADC of the MSP430 board.

3.6.4 Voltage Divider

The LEDs require less voltage to operate than the input voltage value of 12 volts. It was decided that the LEDs would be powered using 6 volts because it is not difficult to divide the input voltage in half; a simple voltage divider is all that is needed. A voltage divider was constructed using two 100 ohm resistors with a power rating of 1 watt. The value of the resistors was found by using the voltage attenuator equation above. It was important to use resistors with a higher power rating because the value that we chose for the resistors was low. Low resistor values were needed in order to provide enough current to the LEDs, but in order to ensure that a minimal amount of heat was dissipated, a high power rating was necessary.

3.6.5 MSP430 and LCD

To prototype our device, we used the readily available MSP430 development kit. This was chosen because in addition to being an easy way to obtain a working LCD display for a prototype, we have experience with the MSP430 from working with it in a prior class. The development board can be seen in Figure 8.

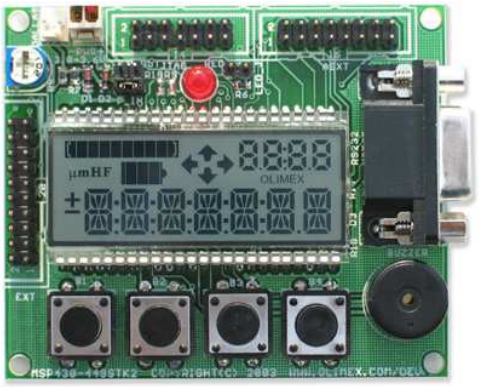


Figure 8- MSP430 Development Board[16]

It is possible to write directly to the LCD screen using the software code, and it is not necessary to do extensive testing on this module given that it is fully integrated. Using the MSP430 to control the LEDs also allows for the turn-on thresholds to be set and altered more easily than if we had performed the same function using an analog approach, such as through the use of comparators.

3.6.6 MOSFET Switch

A switch was needed in order to change between the two banks of LEDs as the vehicle tilts from left to right. In our design, two MOSFETs are connected in a NAND gate configuration. The MOSFETs are controlled by the signal received from the MSP430.

3.6.7 Demultiplexer, Inverter, and OR Gate

The demultiplexer (demux), inverter, and OR gate are placed in between the MSP430 and the LEDs. The demux sends in a binary value from the MSP430 to the LEDs. By using this method to control the LEDs, we only had to use three lines to control the LEDs rather than the eight lines that we would have had to use if the MSP430

[16] "MPS430F449 STARTER-KIT DEVELOPMENT BOARD." Accessed 19 June 2010. <http://www.techtoys.com.hk/MSP430/MSP430-449STK2/MSP430-449STK2.htm>

connected directly to the LEDs. An inverter is involved in the demux circuit because the demux produces a logic high when a logic low is necessary and vice versa. Each bank of LEDs also has an OR gate that ensures that when an LED is lit, the LEDs that had been previously lit in the row remain on as well. The OR gate allows for multiple LEDs to be lit at the same time, providing a more effective warning system.

3.6.8 LED Output

Each LED is connected to the power rail through a resistor at their anode and a 2N3904 transistor at the cathode. Each transistor is controlled by the MSP430 and allows for control of when each LED will be lit. The LED driving circuit can be seen in Appendix C.

3.6.9 Speaker and Amplifier

The speaker that is used in our design is from an old, dismantled computer speaker. The speaker and its amplifier circuit are connected to the same connection as the buzzer that is integrated into the MSP430 development board. We decided to use a speaker rather than the buzzer to produce a louder noise. In addition to the speaker, we decided to use the amplifier circuit that was also inside of the computer speaker because of the built-in knob that can control the volume level. There is also a built-in power button that we have used as the power button for the entire device.

3.6.10 Prototype

Once the blocks of the system described above were constructed and tested, they were combined to create a functional, breadboarded prototype of our design. It was important to have a functional prototype before creating the final product. A picture of the prototype can be seen in Figure 9.

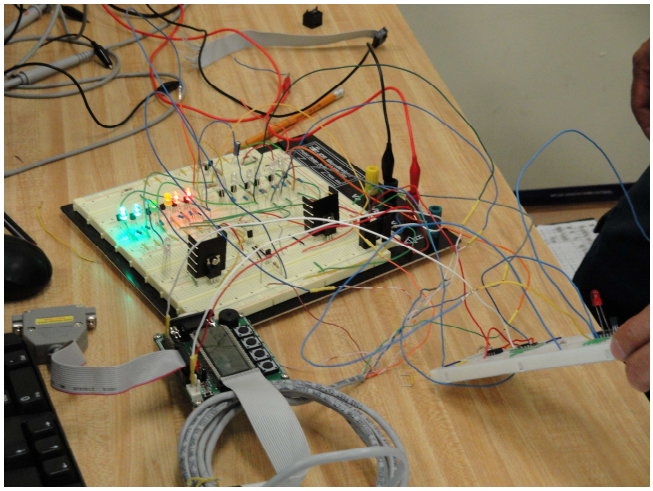


Figure 9- Functional, Breadboarded Prototype

4.0 Product Results

The completed, enclosed prototype can be seen in Figure 10, while a picture of the interior of the final prototype can be seen in Figure 11.

Figure 10- Completed Prototype

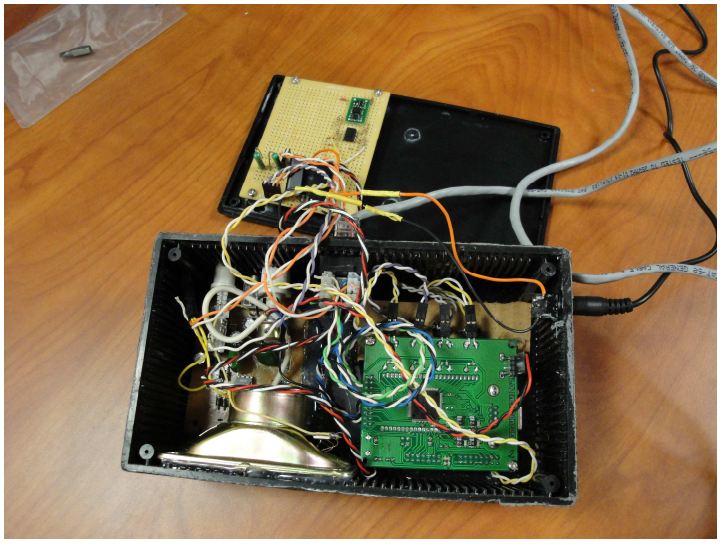


Figure 11-Interior of Completed Prototype

The finished product consists of two boxes connected by a standard CAT5 8 cable. The main box is placed on the floor of the vehicle between the driver and passenger seats, while the LED display box is placed on the dashboard. Having the LED display box separate from the main box will allow the user of the product to choose where to place the LEDs. The LEDs provide a visual warning for the driver in a place where the driver can clearly see the change in the number of LEDs that are lit. The speaker that provides the audio warning is inside the main box and is connected to an amplifier with a knob that allows the user to change the volume level accordingly.

The main box also houses the MSP430 development board. The LCD screen can be seen by the user of the device; however the other parts of the board are enclosed. We installed four external buttons that are connected to the integrated buttons of the MSP430. These buttons are easier to press than the integrated buttons, while also giving the product a more finished look. Inside the main box, there is also the accelerometer, voltage divider, voltage regulator, and signal attenuator circuit.

The LED display enclosure contains the LEDs along with the MOSFET switch, demux circuit, and LED driver circuit. We chose to use a total of fourteen LEDs; each bank consists of four green and three red LEDs. The green LEDs indicate that there is a tilt, but the vehicle is not yet at a potentially dangerous angle. When lit, the red LEDs indicate to the driver that he or she should take action to correct the angle at which the car is driving. The final LED in each bank blinks, indicating immediate danger. The colors of LEDs were chosen because of their common associations; from driving a car everyday, people have learned to associate a green light with safety, while a red light is seen as indicating that a driver should stop because there is a potential danger. The speaker sounds once the third green LED is turned on and increases in frequency as the sequential LEDs are lit. We decided to have the speaker sound at this point because it will be natural for the vehicle to have a bit of a tilt since it will be driving off-road in uneven terrain. Only when the tilt begins to reach a point of being dangerous does the speaker sound.

Our final product is reliable. Since it is able to run off of the power provided by a car battery, the consumer does not have to worry about the product not functioning because of depleted batteries. In addition, we made an effort to design the product to have a minimal amount of components; the less complicated the circuitry, the less likely the device is to have a part that malfunctions. To increase reliability, it was also important for us to choose components that were not likely to be damaged easily. For instance, the accelerometer that we chose has built-in protection circuitry so that the accelerometer is not easily damaged, and the voltage regulator that we chose has a maximum voltage and current range that far exceeds what our device uses.

In order to test and demonstrate our completed device, we constructed a testing rig. This rig models a Jeep Wrangler. The parts for the rig were taken from a Jeep in a local scrap yard. A photo of the Jeep's exterior can be seen in Figure 12, and a picture of the interior can be seen in Figure 13.

Figure 12-The Exterior of the Jeep

Figure 13- The Interior of the Jeep

The finished testing rig works by having a linear actuator tip the base, which models half of a Jeep Wrangler. It effectively demonstrates how our device works, while also giving

a visual as to how far an off-road vehicle has to tip before it rolls over. The completed rig can be seen in Figure 14.

Figure 14- Completed Testing Rig

The rig was used to test our device; however it also could potentially be used to demonstrate our product to consumers. If we took our product to a trade show, for example, it would be a convenient way to show our device to potential customers.

5.0 Cost Analysis

The cost of our prototype is about $158. The most expensive component that we purchased was the MSP430 development board. Although the price of this component remains the same even if 1000 units are produced, the price of many of the components decreases. If 1000 units were to be made, the total cost of the unit would decrease to around $128. A comparison of the price of our prototype and the cost of one unit if we were to make 1000 of our device can be seen in Table 1.

Part	Quantity	Unit Price	Total	Price if 1000 Units are Produced		Total
MSP430 Development Board	1	$54.95	$54.95		$54.95	$54.95
Voltage Regulator LT1084CT	1	$8.00	$8.00		$4.45	$4.45
Op-Amp LM358P	1	$0.48	$0.48		$0.17	$0.17
Accelerometer	1	$22.00	$22.00		$22.00	$22.00
RJ45 Terminal Block	1	$4.24	$4.24		$2.65	$2.65
RJ45 Connector	1	$0.62	$0.62		$0.31	$0.31
CAT5 8 conductor Cable	3 ft	$0.36	$0.36		$0.30	$0.30
SN74LS138N decoder (7 bit)	1	$0.70	$0.70		$0.30	$0.30
HD74LS32P (OR)	2	$0.57	$0.57		$0.21	$0.42
SN74LS14N (inverter)	1	$0.52	$0.52		$0.20	$0.20
100k Resistor 1/4 Watt	3	$0.22	$0.66		$0.09	$0.27
100k Resistor 1 Watt	2	$2.21	$4.42		$0.77	$1.54
2N3904 transistor	20	$0.10	$2.00		$0.04	$0.80
MOSFET IRF510	2	$0.79	$1.58		$0.51	$1.02
5226 Diode	1	$0.04	$0.04		$0.01	$0.01
10k Resistor 1/4 Watt	19	$0.15	$2.85		$0.01	$0.19
470 Resistor 1/4 Watt	14	$0.11	$1.54		$0.05	$0.70
LEDs	15	$0.15	$2.25		$0.08	$1.20
Speaker and Amplifer Circuit	1	$5.00	$5.00		$5.00	$5.00
Prototype Board	2	$5.00	$10.00		$5.00	$10.00
Cigarette Adapter (12 V)	1	$14.95	$14.95		$8.95	$8.95
2mm DC plug	1	$0.72	$0.72		$0.40	$0.40
External Buttons	4	$3.41	$13.64		$1.46	$5.84
Enclosure	1	$5.00	$5.00		$5.00	$5.00
LED Enclosure	1	$1.00	$1.00		$1.00	$1.00
TOTAL			**$158.09**			**$127.67**

Table 1- Prototype Cost vs. Cost of 1000 Units

If we were to sell 1000 units at a price of $180, we would make a profit of $52.33 per unit, and a total profit of $52330. Selling the product for $180 allows us to be competitive within the market. As mentioned in the market research section of the report, a product that has many of the features of our device sells for $179.

It is important to note that this profit estimate does not take into account manufacturing and production costs. The cost of labor, as well as the cost for any

equipment that would be necessary to produce the product efficiently, would lower our profit amount per unit. It is difficult to estimate the total manufacturing cost; however, some of the cost could be off-set by choosing to use less expensive parts in our device. For example, in reality, if we were to manufacture our product, we would probably not use the MSP430 development board. It would not be necessary to purchase the complete integrated board; we could buy a separate chip and LCD screen, which would be significantly less expensive than the development board. This would lower the cost to produce our device, allowing us to make a greater profit per unit and off-set some of the manufacturing costs.

The cost that was incurred during our project also included the cost of the testing rig that we constructed. The total cost of this device was $399.54. This cost, however, was a one-time expense, so our overall profit for 1000 units would not be affected.

6.0 Recommendations

The prototype that we have created can be used as it is currently designed; however there are some additions that could be made to add to the product's functionality. One addition that we could make would be to have the device measure front to back tilt as well as the right to left tilt that the device currently measures. In order to add this feature, it would be necessary to exploit the second voltage output of our dual axis accelerometer. A small amount of additional coding would have to be done for the MSP430, but it would be very similar to the code that has already been written. There are additional pins available on the MSP43 development board to add this feature as well.

We could also add batteries to the device to allow it to function without being plugged in to the cigarette lighter of the vehicle. If the user could choose to operate the device off of batteries, it would also be desirable to add a charging circuit so that battery life would not be a large concern.

We would like to make our device so that the user can enter the specific parameters of their cars so that the rollover angle that is calibrated will be more accurate. The MSP430 has the ability to do this calculation, and the code for the calculation has already been written. Due to the size constraint of the program that we are using to write the code, however, this function can not be implemented at this time. If we were able to find another program that would allow us to compile all of our code, or were able to find a way to work around the use of the arctan function, which calculates the angle, then this addition could be easily made to our product.

7.0 Conclusion

We have successfully created a device that will warn drivers of off-road vehicles how close they are to a rollover. By incorporating both a visual and audio warning into our device, we have ensured that the driver will be made aware of how close he or she is to rolling over. Our product has built upon the device created by the previous team; the modifications that have been made make the device more accurate and have increased the effectiveness of the warning system. The final product that we have created meets all of the product requirements that we established at the beginning of the project; the device is intuitive to use, portable, provides an effective notification, and is aesthetically pleasing. Although additional features could be added to our device, the product currently provides drivers of off-road vehicles with a relatively affordable, effective way of identifying if

they are in danger of rolling over. As market research showed, there is a need and desire for a product such as the one we have created. It is our hope that widespread use of a device like ours will help to decrease the serious rollover accidents experienced by off-road drivers.

Appendix A: Works Cited

"Accelerometers and How they Work." Texas Instruments. 7 June 2010. <http://www2.usfirst.org/2005comp/Manuals/Acceler1.pdf>

"ADXL320." Analog Devices. 24 June 2010. <http://www.dimensionengineering.com/datasheets/ADXL320.pdf>

Cuerden, Richard, Rebecca Cookson, and David Richards. "Car Rollover Mechanisms and Injury Outcome." 15 June 2010. < http://www-nrd.nhtsa.dot.gov/pdf/esv/esv21/09-0481.pdf >

"DE-ACCM5G." Dimension Engineering . 19 June 2010. <http://www.dimensionengineering.com/DE-ACCM5G.htm>

"Determine Center of Gravity, and Roll-Over Angles on Your Vehicle." Jeepaholics Anonymous. 13 Jan 2002. 28 May 2010. <http://www.jeepaholics.com/tech/cog/>

Hall, Jonathan, and Jerry Magraw. "Modeling Vehicle Rollover." The Physics Teacher. Volume 43 (2005), p. 51. 28 May 2010. <http://www.suu.edu/faculty/penny/RolloverPaper/Modeling%20vehicle%20rollover(Magraw-Hill%20paper).pdf>

"LT1084CT-3.3" Digikey. 19 June 2010. <http://media.digikey.com/photos/Linear%20Tech%20Photos/LT1084CT-3.3%23PBF.JPG>

McNulty, Kevin. "4x4 Trucks Rollover Accidents Preparation - Rollover Reality: Always Prepare For The Worst Before Hitting the Trail." Peterson's 4Wheel & Off-Road. April 2009. 7 June 2010.

<http://www.4wheeloffroad.com/features/131_0904_4x4_trucks_rollover_accident_preparation/index.html>

"Motorvehicle Rollover-FAQs," 28 May 2010. <http://buzzmotors.com/rolloverfaqs.html>

"MSP430x44x Datasheet." Texas Instruments. 26 June 2010. <http://ece.wpi.edu/courses/ece2801smj/msp430f449_datasheet.pdf>

"MPS430F449 STARTER-KIT DEVELOPMENT BOARD." 19 June 2010. <http://www.techtoys.com.hk/MSP430/MSP430-449STK2/MSP430-449STK2.htm>

"MSP430 x4xx User's Guide." Texas Instruments. 26 June 2010. <http://ece.wpi.edu/courses/ece2801smj/MSP430x4xxUserGuide.pdf>

"Off-Road Inclinometer." Wheeler's Off-Road, Inc. 6 June 2010. <http://www.wheelersoffroad.com/inclinosam.htm>

"Olimex Schematic." Olimex LTD. 26 June 2010. <http://ece.wpi.edu/courses/ece2801smj/msp430-449stk2-a.PDF>

" RAD-1 Digital Audio Inclinometer." 6 June 2010. <http://rad1.com/>

Richardson, Shane. "Engineering Solutions. Rollover: One of the Road Safety Problems that is not being addressed." Road Safety Toward 2010, pp. 48-50. 15 June 2010. <http://www.dvexperts.net/pubs/rollover.pdf>

Ryll, Erik, Tyler Golightly, and Shannon Leary. "Automotive Rollover Detection and Early Warning Device." 29 April 2010.

"Stats and Info." Lovell Rock Racing. 7 June 2010. <http://lovellrockracing.com/stats_info.php>

Appendix B: Weekly Summaries

Week One Summary
May 27, 2010

Our main objective for this week was to clearly define our project. Since we are continuing the Automotive Rollover Detection and Early Warning Device project completed this past school year, we began the week by brainstorming about additional features that could be added to the existing device. Also this week, we wanted to determine the market that our product would target so that we could customize our device to meet its specific needs.

After reading the final report describing the current device, we decided to change our target market. The current product targets all drivers of common vehicles. We feel that it is difficult to create a device that can predict an automobile's chance of rollover accurately for a variety of different vehicles. This fact has caused us to target drivers of off-road vehicles who frequently drive in difficult terrain. Initial market research has included the joining of an online forum about off-roading. This has led to the determination of the most popular vehicles used: the Hummer H1, Land Rover Discovery, Jeep Wrangler, Toyota Tacoma, and the Chevy Tracker. By limiting our market, we can create a product that more accurately measures the risk of rollover. Additional market research will need to be conducted about these specific vehicles before proceeding.

Based on the description and schematic of the existing product, we came up with several changes that we would like to make to the device. The first change that we thought to make is to add an audio warning component to the product in addition to the LEDs that the device currently uses. Although the current device has a buzzer that sounds upon reaching the most dangerous level, we believe that a buzzer that gradually increases in either volume or frequency would be an addition that would better get the attention of the driver. Secondly, we think that adding more LEDs to the design would help to make the driver more aware of his or her dangerous driving. We picture having two LED arrays, one on either side of the box containing the device. In addition to the changes that we want to make in how the driver is warned by the device, we think that changing the power supply from external batteries to the power available from the cigarette lighter of a car would be an improvement that we could make to the design. This is an easy change that would eliminate the concern about battery life and the need for the customer to frequently change the batteries of the device. Lastly, we feel that it would be beneficial to use a microcontroller, specifically the MSP430, rather than the comparators used in the current device. A microcontroller would allow us to more easily adjust the voltage thresholds that would determine when the driver would be warned by the LEDs and buzzer.

The main problem that we encountered this week involved the MSP430. When we attempted to use it in AK113, we received driver errors on each computer that we tried. It seems that after the computers were upgraded to Windows 7, the serial ports on the computers were not functioning properly. For the time being, we are using the chip on a personal computer, but have been told that by the end of the week, the problem should be resolved.

Although we do not yet have a formal schedule, we believe that we are making good progress. Having clearly defined our project within a week is a good step towards completing the project successfully within the allotted amount of time.

Week Two Summary
June 3, 2010

This week, we had several goals. We wanted to continue doing market research to see if there is a sufficient market for our product. In addition, research about how to calculate the thresholds that would determine the level of warning a driver would receive was conducted. Using this information, it was possible to make progress in the programming of the MSP430. Further consideration was then given to a system level block diagram of our product. After working on the design and functionality of our device, we also decided upon the best way that we will be able to test the prototype.

After getting our project idea approved, it was important to continue market research regarding the need for rollover detection and warning devices designed for off-road vehicles. We found that because off-road vehicles, such as the Jeep Wrangler, have a higher center of gravity than standard cars, they have a higher tendency to rollover. Speaking with a girl who drives a Jeep that she takes off-road further showed that there is a need, and market, for our product. She said that she had previously rolled over her Jeep and would have liked to have had a device to warn her that she was travelling at a dangerous angle. The girl also said that her friends, who are also involved in off-roading, would appreciate a warning device as well. Internet research also was useful in showing that there are a large number of people who go off-road who would like a rollover detection and warning device. We even found people discussing on forums how they were trying to build a device that does exactly what we hope our product will be able to do; warn a driver when they begin to drive at too dangerous of an angle.

Deciding how to calculate when a vehicle has reached a dangerous angle and the driver needs to be warned was the next step that was taken this week. Research showed that the standard method of calculating a vehicle's risk of rollover, the static stability factor (SSF), would not be adequate for our purposes. The SSF assumes vehicles to be driving on a flat, smooth road; clearly, people using our device would not be driving under these conditions. We were able to find a clearly explained way of how to calculate the center of gravity and related rollover angles of a Jeep that is to be taken off-road. These findings enabled us to determine the vehicle parameters that would be necessary to include in the programming of the microcontroller to ensure that the driver would be warned appropriately about the different levels of danger.

Once we determined the way in which the thresholds would be calculated, a large amount of progress was made in the programming of the MSP430. The MSP430 is now able to read in a voltage. Eventually, this voltage will come from the accelerometer and/or the power supply, but for now, the voltage is from the lab power supply. The MSP430 has also been programmed to control seven LEDs off of port two. These LEDs light up sequentially based on the input voltage. Control of the buttons has also been programmed to run a menu driven system. This menu allows the user to enter the appropriate parameters needed to calculate the vehicle's risk of rollover without an additional keypad or being connected to a computer. The menu feature is very useful in

that it will allow the parameters to be custom entered for each vehicle. Currently, the device has the ability to store information for up to three vehicles. We may design the device so that it has stored values that we estimate to be accurate for the majority of off-roading vehicles based on measurements that we will take. These measurements will most likely be found using a scale model of a Jeep Wrangler.

Once again, the main problem that we encountered this week involved IAR Kickstart. Since we are using the free version of IAR because the lab computers are not working properly, the program has a size limitation; when programming in C, the program can not exceed 4KB. This limits our use of certain functions within the program. It was possible to work around this problem by avoiding the use of these functions, but if the program becomes more complicated, it may be difficult to stay within this size constraint. If this becomes a problem, we have been told by Professor Jarvis and Bob Brown that there are certain computers that have yet to be converted to Windows 7, so it would most likely be possible to continue writing the program, just on a school computer.

With the programming of the MSP430 progressing, it was possible for us to draw an initial system block diagram. This is attached. Given that we now have an idea about how our device will function, we are going to begin researching the different components that we will need. We have already ordered an accelerometer that we think is appropriate for our device. Further component research will occur next week.

In addition to working on the device, we discussed how we were going to test its functionality. Since we do not have access to an actual off-road vehicle, we have decided that the best way to test our product will be to build a rig modeling a Jeep Wrangler. This will be done by purchasing old Jeep parts and building an appropriate platform that can be moved mechanically to a wide range of angles. This will allow us to see when the driver will be warned, and if the warning is adequate.

We have decided upon the tasks that must be completed in order to construct the device and successfully complete the MQP. This is outlined in the Gantt chart that is provided. The tasks were divided evenly and were assigned based on our known strengths and weaknesses as a group. We are currently on schedule, and hope to continue making sufficient progress to complete the project by July.

Week Three Summary
June 10, 2010

The main goals that we had this week were to perform component research and to begin breadboarding the system. We also began to discuss how the device would be enclosed. In addition to working on the actual prototype, progress was made on the construction of the rig that will be used to test and demonstrate our product.

After creating the block diagram last week, we realized that we would need to take the 12 volt power supply and decrease it to a dependable 3.3 volt supply in order to safely power the MSP430 development board. We chose to use a voltage regulator to do this. Initially, we chose the LM1086, a 3.3V voltage regulator. When testing the circuit with the LEDs from our ECE kit, the voltage regulator worked fine; however when these LEDs were replaced with the brighter green and red LEDs that we will actually use in our design, the voltage regulator began to get hot as the circuit was drawing too much

current. To solve this problem, we have decided to use an adjustable voltage regulator with a higher current rating. The component that we chose has a rating of 5A, which should be more than sufficient for our device. At this time, we simply need to determine the appropriate resistor values to add to the circuit to ensure that the output of the voltage regulator is 3.3 volts.

When we replaced the LEDs that we had been using with the brighter LEDs that we will use in our product, we realized that we may be able to eliminate the LED driver circuitry; the LEDs seem to be sufficiently bright without the addition of a transistor. Eliminating this circuitry would not only improve the reliability of our product by completely eliminating a block from our diagram, but it will also save space within our enclosure.

We decided that although we have two banks of LEDs in the design of our device, we would like to control them off of one line. This decision was made after a good deal of debate. Even though we felt that it would be easier to control the LEDs with two lines, having one line will save space in the design and make the product look better aesthetically. Having one line, however, meant that we would have to find a way to switch between the two banks of LEDs. Figuring out how to do so posed one of the biggest problems that we faced this week. At first, we thought that a multiplexer would be a good choice, but we then realized that a multiplexer was a big chip to use when we were only going to have to decide between one bank or the other. After consulting Professor O'Rourke, we decided to use a relay switch. The relay is now working, so it seems that at this time, we will be able to use one line.

The audio system of our device was also tested this week. We decided to dismantle an old computer speaker and use the amplifier, speaker, and knobs in our design. The rectangular speaker will fit well within the enclosure that we have chosen to use. In order to ensure that the sound can be heard, we have decided to drill holes in the enclosure in front of the speaker.

Also, in regards to the enclosure, we discovered a flaw in how we had decided to enclose the MSP430 board. The wires that we have been using while testing our device come up above the LCD screen, meaning that the LCD screen would not be able to be as close to the top of the enclosure as we would like it to be when mounted inside. When trying to determine how to fix this problem, we thought about bending the pins, but decided that they were too fragile. We also thought that it might be possible to solder the pins under the board, but then realized that it would be easy to short out a pin because of how precise we would have to be when soldering. We eventually decided that the best solution would be to make a ribbon cable, which would have a height that was very close to that of the LCD screen. We made a ribbon cable for the pins and now just need to test that it is functioning the same way as the individual wires that we have been using.

The testing rig that is being constructed is well under way at this time. We have purchased the appropriate parts from a local scrap yard. So far, the dashboard, which includes a steering wheel, has been constructed. This has determined the width that the base will be, along with the height the rig must be raised by the actuator to ensure that it will be able to tilt at a large enough angle to set off the device. The rig should be completed in about a week.

Unfortunately, the accelerometer has not yet arrived. In order to move forward with our design, it is important for this part to arrive so that we can test that our prototype

will react properly when provided with the accelerometer input. Once this is tested, it will be possible to begin to decide how all of the components will be inserted in the enclosure. Aside from this, however, we are on schedule and the project is progressing well.

Week Four Summary
June 17, 2010

The main goal this week was to breadboard our design and produce a working prototype. During this process, we encountered a number of challenges that we had to solve by modifying or adding blocks to our functional block diagram. In addition to breadboarding our system, we began to think of the best way to enclose the device. We also worked to make progress on the testing rig and the final report.

We continued to breadboard the different parts of our product. The accelerometer finally arrived, so we began the week by measuring the maximum voltage that it can produce. We found that the accelerometer can produce a maximum voltage of 1.85 volts. This value is greater than the maximum voltage that the ADC of the MSP430 can handle, so it was necessary to insert a voltage signal attenuator between the accelerometer and the ADC. This circuit consists of a voltage divider made up of two 20k ohm resistors (in order to cut the voltage from the accelerometer in half) and an op-amp in the unity gain configuration to keep the value steady. This circuit ensures that the ADC will never receive too much voltage.

Testing of our circuit also showed that the relay switch that we intended to use to switch between the two LED banks would not work in our design. The relay switch is current driven, and not enough current was being produced by the MSP430 to cause it to switch. To fix this problem, we decided to use two MOSFETs two create a switch. This approach, because it is not current driven, works well.

We decided to insert a demux in between the MSP430 and the LEDs. The demux sends in a binary value from the MSP430. Using this method to control the LEDs takes only three lines as opposed to the eight lines that we had before. This frees up other lines to function as ground, a 3.3V rail, and a 12V rail which will make it easier to transfer our design on to a PCB board. There are also additional lines that can be used in the future to not only measure tilts from the left and right, but also from the front and back. The demux circuit also involves an inverter because the demux alone was producing a logic high when we needed a logic low and vice versa. The inverter easily corrects this problem.

While we worked during the week, we also realized the LED driving circuit was still necessary. Initially, we thought that we would be able to eliminate this aspect of the design because the LEDs were sufficiently bright, but we found that the transistor was necessary to control when to turn on each LED. The transistors are controlled by the MSP430.

Once the breadboarded design was shown to work, we began to discuss how to enclose the LED display that will be mounted on the dashboard of the vehicle. The challenge that we are facing is that the enclosure must be relatively slim, but it must be able to contain the MOSFET switch, LED driver circuits, and the LEDs. We are currently working to see if it can fit in an enclosure that we purchased and will have to determine if a different size enclosure is necessary.

The testing rig is mostly completed. The dashboard and steering wheel are already mounted and the seat just needs to be screwed on to the base. The actuator that will control the tilting of the rig has been ordered and will hopefully be installed by next week.

The final report was started. During the week, research was performed into the effect of off-road driving on rollovers. The report has been outlined and several sections have been written. As the project progresses, it will be possible to write more of the report.

By the end of the week, a functional prototype was created that demonstrated proof of concept; however we did have to make modifications to our original design. Luckily, these changes did not set us back; the timeline outlined in the Gantt chart is still accurate. We continue to be on schedule and the project is beginning to come together. Now, we will work towards enclosing the device, completing the testing rig, and writing the final report.

Week Five Summary
June 24, 2010

This week, we continued to transfer our breadboarded prototype to two PCB boards. We also worked on the fabrication of the enclosures for the main box and the LED display. Progress was also made on the final report. During this week, we made a good amount of progress, but also encountered a few problems.

We have moved everything from our breadboard onto PCB boards. The board that will be located in the main box consists of the voltage regulator, signal attenuator, and accelerometer. The PCB board that will be placed in the enclosure that will be mounted on the dashboard of the car consists of the LEDs and LED driver circuit, as well as the demux, inverter, and "OR" gate. The boards each connect to the MSP430 via jumpers. The speaker and its corresponding amplifier circuit have also been wired and will be connected by a jumper to the MSP430 board. The soldered circuits have been shown to work and measurements have been taken to ensure that the correct voltage values are produced from each functional block.

Fabrication of the two enclosures has started. The main enclosure has been constructed. Four external buttons have been put on the top of the box and will be able to connect to the buttons of the MSP430. The MSP430 will be mounted internally, and the LCD screen will be slightly sunken-in from the top of the box. An old MSP430 board was used to make the cutting measurements. The speaker, amplifier circuit, and the PCB board containing the accelerometer will also be placed in this box. We have already been able to fit all of theses parts into the box successfully. Once we are sure of the functionality of our device, we will permanently mount these pieces. We purchased a wooden box from a craft store to use as the LED display enclosure. The enclosure that we had wanted to use turned out to be slightly too small for our design, so we chose a box is taller. Since it is wood, and it will also be easier to drill holes for the LEDs.

A good amount of progress was made on the final report during this week. The majority of the introduction and background information has been written. It will probably be possible to begin the section describing the final product this weekend. The other main part that still needs to be written is the cost analysis section. We have

compiled a list of components, and simply need to do research on their cost individually versus the cost of the components when purchased in bulk. We have not calculated our final product cost as of yet, but this should be determined by our next meeting.

Although a lot was accomplished this week, we did run into a few problems. One of the problems that we encountered involved the actuator that we will be using to control the tilting of our testing rig. When it arrived, the actuator did not work. After speaking with technical support, we returned the actuator and the company will be sending us a new one that should be arriving soon. We also encountered a problem with the ADC of the MSP430. It is not functioning as it should, so we will most likely purchase a replacement board. We have thoroughly gone through the wiring of our circuit, and have determined that everything is functioning correctly, so the problem that we are experiencing is definitely related to the development board. Uploading our code and connecting our circuitry to a new MSP430 board is not difficult, so this problem should not have an effect our timetable.

Overall, this week was very productive. We are pleased with the soldering and fabrication progress that we have made. Despite the problems that we experienced, we continue to be on schedule and the project is beginning to come together. Once the actuator arrives, we hope to complete the testing rig. Next week, we would like to continue the fabrication process, make progress on the report, and finish the testing device.

Week Six Summary
July 1, 2010

Our goal this week was to work towards completing our project. To achieve this, we worked on finishing our prototype and the testing rig. We also worked on the final report.

At this time, our final prototype is complete. This week, we finished both the main enclosure, and the LED enclosure. The replacement MSP430 development board arrived, and we were able to fit all of the components into the main box. The bottom of the box is screwed on, so we can still access all of the components easily. The wooden box that we used for the LED enclosure was also completed. The box was painted black to match the main box, and holes were drilled for each of the LEDs. The components can still be accessed via a sliding panel on the box. This allows for access to the components if necessary.

The testing apparatus has also been completed. Earlier this week, a new linear actuator arrived. This actuator works properly. It has been mounted to the rig that we had already built. To power the actuator, we are using a charger that provides the needed voltage and current to operate the device.

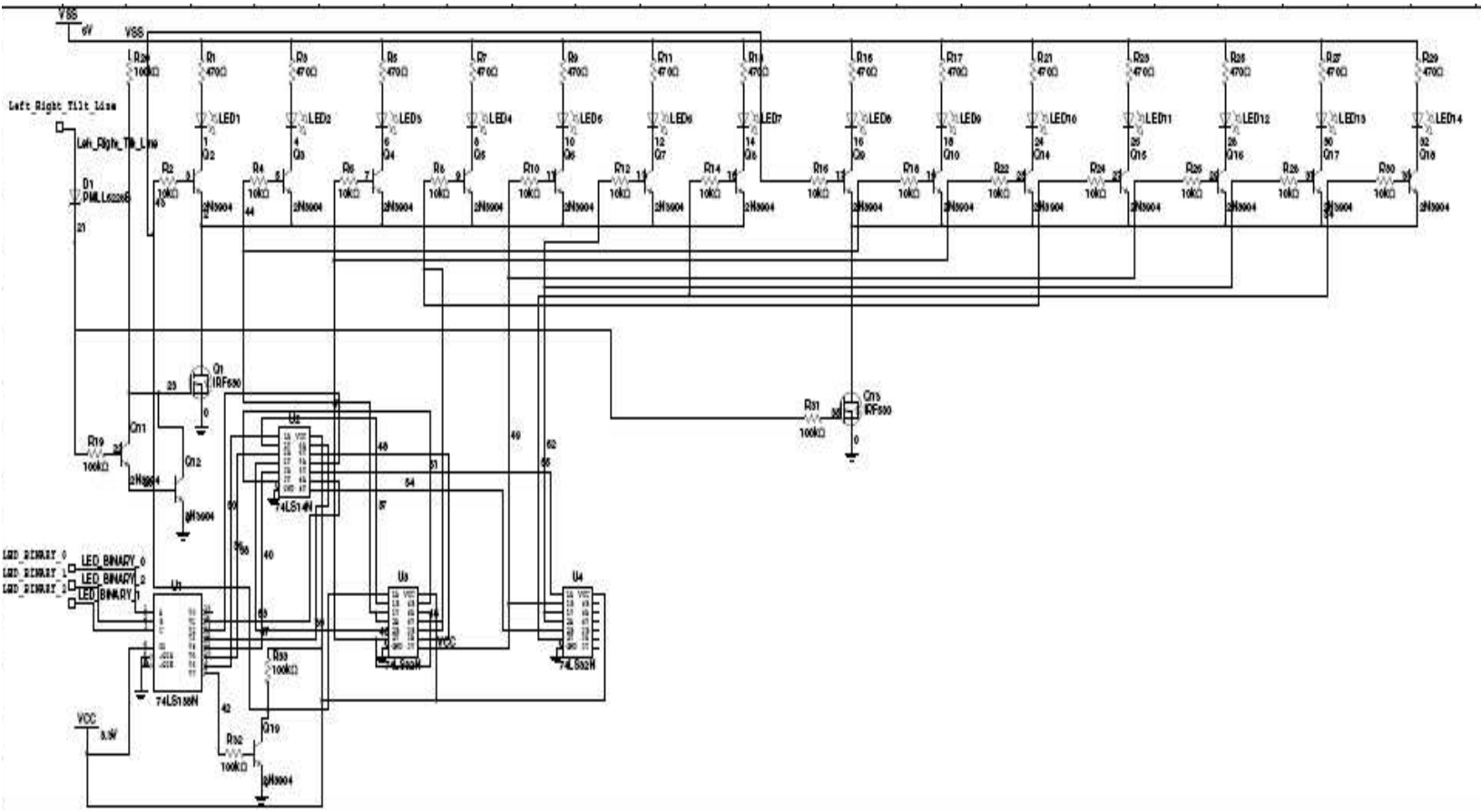
A good amount of progress has also been made on the final report. The writing of the paper is pretty much complete; it is now only necessary to add additional pictures of the prototype and schematics. Once the schematics are complete, they can be added to the appendices.

We have encountered a couple problems when creating the schematics. The first is the size of the schematic. It has been difficult to draw the schematics so that they are readable, and can be put into the report. When the schematics are added to the report, we

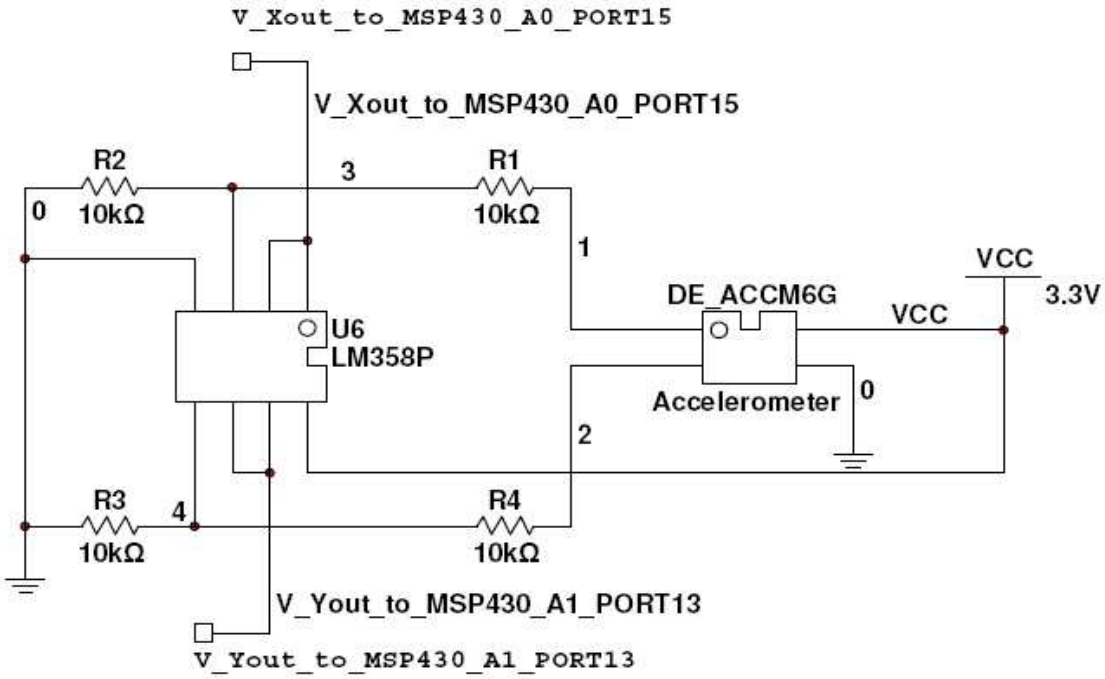
will just have to try to adjust the size of the picture so that it will fit well on the page and can be easily read. The other problem that we have encountered is finding a component in Multisim that is equivalent to the op-amp chip that we used in our design. There does not seem to be a component with the same pinout in the Multisim library. If we are unable to find a component that matches our op-amp, a solution that we have considered is to find a component that looks similar and just label it to match the component number of our op-amp.

Our project is nearly complete. We have a working prototype, and our testing rig is now working. The final report is close to completion, and we feel that we have met the MQP criteria outlined by the ECE department. We are currently slightly ahead of schedule, and hope to have the entire project completed by the end of next week.

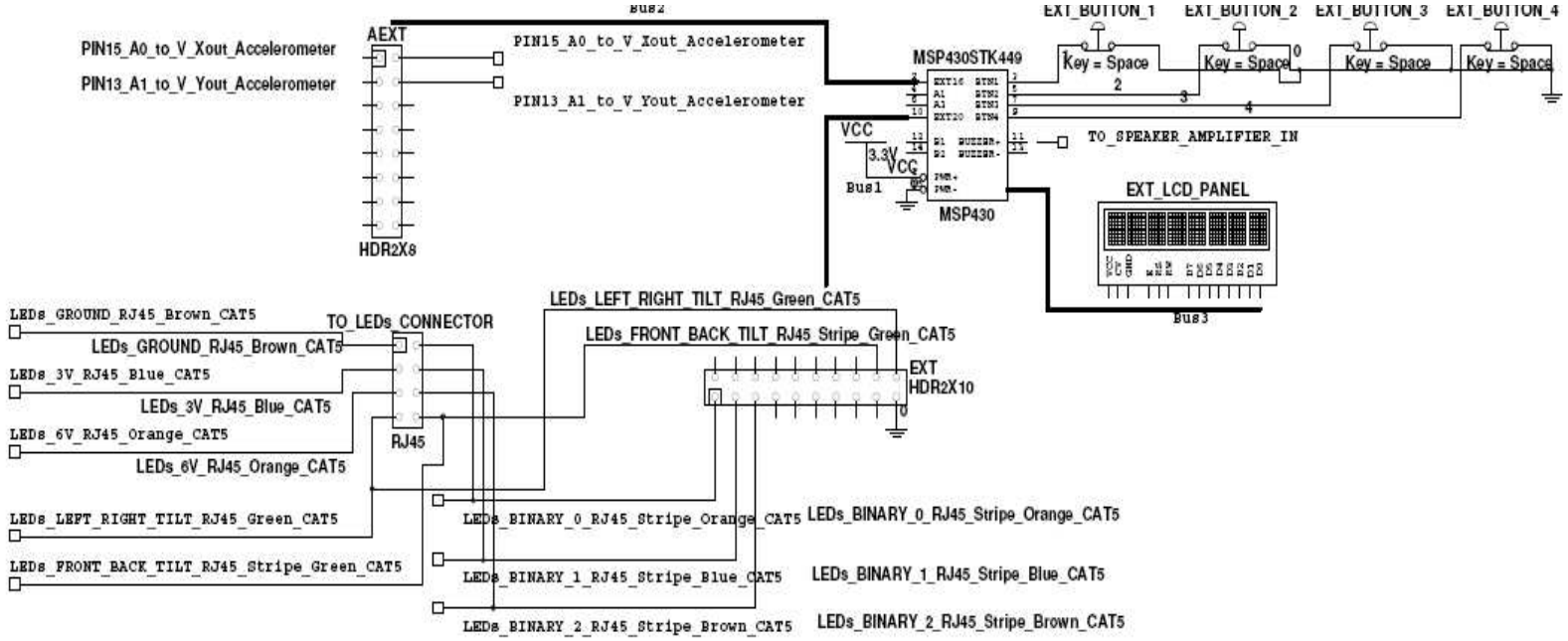
Appendix C: Schematics



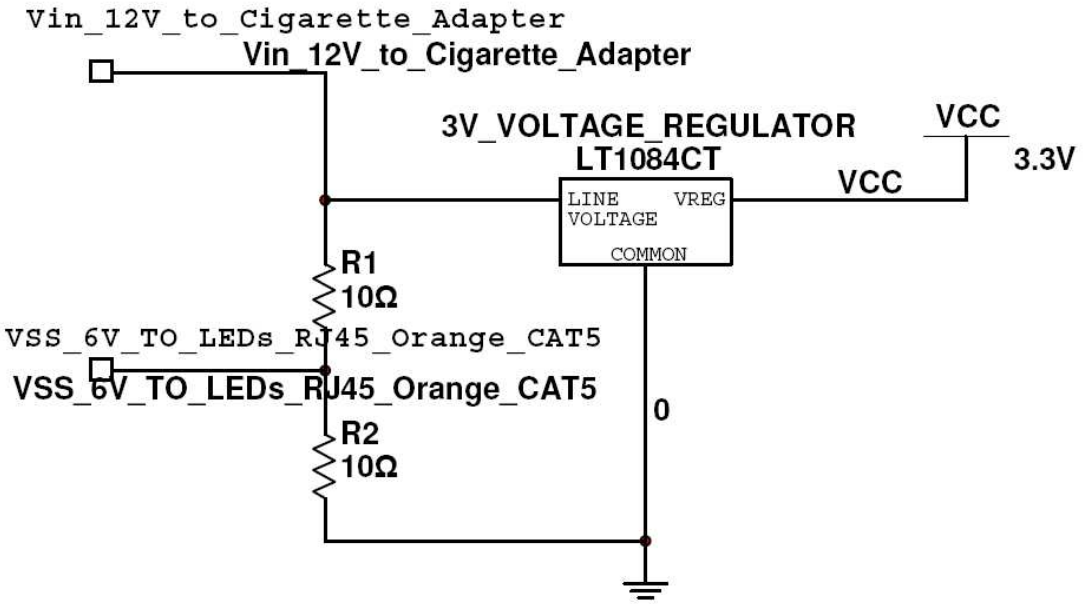
LED Driver Circuit



Accelerometer Circuit



MSP430 Schematic



Power Regulator Circuit

Appendix D: MSP430 Code

```
#include <msp430x44x.h>      // Chip definitions for msp430F449
#include <string.h>          // for some string functions
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <in430.h>
#include <signal.h>  // interrupt (ADC12_VECTOR)

//****************************************** GLOBAL VARIABLES*************************************
char *LCD = LCDMEM;          // pointer to LCD Memory Segments. Pretty cool, got this idea from
        TI-website

//****************************************** LCD CONSTANTS ***************************************
#define    a     (0x80)    // definitions for LCD seegments on the Olimex LCD. 4-Mux operation is
        assumed
#define    b     (0x40)    // For more details on 4-Mux operation, gather your LCD datasheet,
#define    c     (0x20)    // TI's MSP430F449 User Guide (look for LCD Controller, then 4-Mux),
#define    d     (0x01)    // and MSP-449STK-2 schematic. You will need ALL these 3 when
        defining
#define    e     (0x02)    // each number or character. Remember, the Olimex LCD doesn't use a
        LCD driver!
#define    f     (0x08)    // You tell the LCD what characters to display. It's very time consuming!!
#define    g     (0x04)
#define    h     (0x10)

#define        B1              P3IN & BIT4      //B1 - P4.4
#define        B2              P3IN & BIT5      //B2 - P4.5
#define        B3              P3IN & BIT6      //B3 - P4.6
#define        B4              P3IN & BIT7      //B4 - P4.7
#define        STATUS_LED      BIT3             //STATUS_LED - P1.3
#define        TIME            200
#define        TIME_S          200
#define        BUZ1_ON         P1OUT |= BIT0    //P4.2
#define        BUZ1_OFF        P1OUT &= ~BIT0   //P4.2
#define        BUZ2_ON         P1OUT |= BIT2    //P4.3
#define        BUZ2_OFF        P1OUT &= ~BIT2   //P4.3
#define        DALLAS          P2IN & BIT7      //P2.7 - DALLAS
#define        CR              0x0d
#define        LF              0x0
#define        LED_OFF         P1OUT |= BIT3
#define        LED_ON          P1OUT &= ~BIT3
#define        CLK_CNT         (0x2000)         // what is this value? 32768
#define        BUTTON_1        0x
#define        MAX_MENU_LEVEL    3

#define        MAX_MENU_OPTIONS   8   // 10 options per menu level
#define        MAX_OPDESC_LEN     10   // 10 characters
#define        LWB_AI          0    // Length of wheel base - array index corresponse to the
        operational value of each operation parameter such as vehicle wheel base, weight,
        length and height
#define        LTB_AI          1    // Length of Track - outer tire edge to outer tire edge
#define        HF1_AI          2    // Height of Front Axle Hub (Center) from ground level
```

```
#define       HF2_AI          3   // Height of Front Axle Hub (Center) from ground level while
        Front is elevated
#define       WF_AI           4   // Weight on Front Axle at level
#define       WP_AI           5   // Weight on passenger side
#define       WR1_AI          6   // Weight on Rear Axle with vehicle level
#define       WR2_AI          7   // Weight on Rear Axle when Front of Vehicle is elevated

#define       MAX_MAIN_MENU_OPS  3   // maximum number of options at the main menu level
#define       MENU_SETUP_OP      0   // Setup option # 0
#define       MENU_SAVE_OP       1   // Setup option # 1
#define       MENU_LOAD_OP       2   // Setup option # 2

// ******* Public - Global Application Variables *****************
//
// Needs to be global in this example. Otherwise, the
// compiler removes it because it is not used

static int ADC_Results_A0; // Channel A0 ADC Reading
static int ADC_Results_A1; // Channel A0 ADC Reading
static int ADC_Results_A2; // Channel A0 ADC Reading

unsigned char hitKey=0;
unsigned char buttons = 0; // current buttons pressed from port 3
unsigned char LastBtnPressed = 0; // Last set of buttons pressed
unsigned long SecLstBtnPressed; // number of seconds since last button pressed
unsigned long SecBtnHeld; // number of seconds button held down.

unsigned int ADCresults[300]; // 300 samples
unsigned int resultsindex = 0; // looping thru 300 samples
char paused = 0;
unsigned int Vin_Max = 81; // Max Angle 81= 1020 @ level - 1101 @ 45 deg  ----- 4096 for 3.6 V
unsigned int Disp_Channel = 0; // 0, left , 1 right , 2 front panel
int X_Angle = 0; // Horizontal Tilt Angle
int DriverSide_Angle = 0; // in degree
int Passenger_Angle  = 0;

//
// config menu level and menu option within each level
// use menu level and options to store  user configurable parameters such as
unsigned int Menu_Level = 0;  // 1 - 10
unsigned int Menu_Option = 0; // 1 - 10 or more for each
float OPValue[MAX_MENU_OPTIONS]; // store the actual values in float array - 5000 lbs for
        example Max. number of options - 2D arrays can be used to store many more options
char  OPDesc[MAX_MENU_OPTIONS][MAX_OPDESC_LEN]; // description of setting array
        elemement i.e. LENGTH, PASSENGER WEIGHT max 10 Char or each
//
// saved vehicle profiles
//
float VH0_OPValue[MAX_MENU_OPTIONS]; // Vehicle 0 - current values at start up. if these are
        not blank
float VH1_OPValue[MAX_MENU_OPTIONS]; // Vehicle 1
float VH2_OPValue[MAX_MENU_OPTIONS]; // Vehicle 2
float VH3_OPValue[MAX_MENU_OPTIONS]; // Vehicle 3

//
// main menu - SETUP , LOAD, SAVE
```

```
// Setup - change current settings
// Save - save current settings to Vehicle1,2,3,4 ( up to 4
// Load - load settings from saved vehicle profile
//
char MainMenu[MAX_MAIN_MENU_OPS][MAX_OPDESC_LEN]; // SETUP,SAVE,LOAD
unsigned int Main_Menu_Option = 0;

// ****************************************** FUNCTION DECLARATIONS ********************************
void clearLCD(void);                        // Clears LCD memory segments so that LCD is blank
void initLCD(void);                        // Setup code to interface LCD with MSP430F449
void shortDelay(int dSpeed);                    // Loop Delay for a reasonable time. Adjustable delay.
void writeLetter(int position,char letter);     // displays a single character on the LCD
void writeWord(const char *word, int repeat_times); // displays words upto 7 characters on LCD.
        Can also
                                    // display decimal numbers passed as text
void writeNumber(int long number);                // displays integer numbers on the LCD
void writeFloat(float number, int decimals);        // display floating with specified decimal digits
//void writeSentence(const char *word, int scrollForever);   // enable this function if you want to
        display complete
                                        // sentences only. The sentences will scroll from right to

void LEDdisplayHex(unsigned char num);  // displays hex code for digit num

void Init_IO( void ); // initialize IO ports LED / Buzzer
void Init_ADC12( void ); // initialize ADC12 interrupt handler
void Init_Port_02( void );// initialize port 2 for output jv 2010-05-27 18:46
void BUZZER (void);
void STATUS_LED_ON (void); // turn on status led
void STATUS_LED_OFF (void);
void Blink_LED (void);  // blink 5 times
void Disp_LED_Meter ( unsigned int Meter_Value, int Disp_Channel ); // turn on leds on channel
        2 0 - 7 LEDs

//void Delay ( int a);

// ADC functions
void Get_ADC12_A0 (void);
void Get_ADC12_A1 (void);

void getButtons(void); // get button value and store in
void itoa(int input, char* output);

void timerBSetup(void);
void ADC12setup(void);
void startTimerB (void);
void stopTimerB (void);
void runtimerb(void);


void init_sys(void);      // MSP430 Initialization routine
void Init_Main_Menu( void );  // initialize main menu array - SETUP,SAVE,LOAD
void Init_OPSettings_Arrays ( void );// initialize operational settings arrays
//void setupKeypad(void);   // Keypad initialization
//void getKeys(void);       // Read from ketpad
void Disp_OPSettings ( int Op_No ); // display both description and value / flashing
void Disp_OPValue ( int Op_No ); // display selected value / no flashing
```

```
void Inc_OPValue( int Op_No);   // increase Operational Value by 1
void Dec_OPValue( int Op_No);   // decrease Store Value by 1
void Select_OpSetting (void);  // select operational settings
void Change_OpSetting (void); // change selected option

void Init_Main_Menu (void);
void Select_MainMenu (void);  // main menu selection

void buzzerOn(char period);    // turns buzzer on
void buzzerOff(void);   // turns buzzer off
void alarm_buzzer(char Meter_Value);


// ****************************************** MAIN FUNCTION ************************************************
void main(void)
{

   unsigned int count = 0;
   unsigned int Port3_Out =0;
   unsigned int Volt_Meter_Value = 0;
   float Tmp_No = 0 ; // temp number



   WDTCTL = WDTPW + WDTHOLD;   // Stop watchdog timer  (better stop this fellow!)
   init_sys();               // Initialize the MSP430


   //shortDelay(2);             // you can remove this delay if you want
  //P1DIR |= 0x08;              // Set P1.3 to output direction to run LED
   Init_IO();

   Init_Port_02();
   //P3DIR |= 0XFF; // all channels to output on port 3

   // jv 2010-06-12 06:59 PM
   // try to initialize port 3

   //
   // here are the proper settings for port 3 which is shared with buttons in higher bits
   // and available on ext connector under lower bits bit 0 - 3
   // bit 4 - 7 - connected to button 1 - button 4
   // P3DIR = 0X00 // set all port 3 pins to input
   // i = P3IN; // Read in the port 3 pins
   // i &= 0x01  // Mask upper 7 bits to look at pin 0
   //           // i now equals the value of port 3 pin 0

   // P3DIR = 0XFF // set all port 3 pins to output
   // P3OUT |= 0x01 set port 3  pin 0 to 1
   //

   //


   //P3DIR = 0X00; // all channels to input on port 3
```

```
  P3DIR = 0x0F; // only lower bits 0-3 for output and upper bits for input ( KEYS )
  //P3SEL = 0x00;  // input
  //P3OUT = 0x00;

  // Start timerb
  runtimerb();
  Init_Main_Menu();
  Init_OPSettings_Arrays();

  //ADC12setup();


 // LCD Test Code Goes Here ----------------

     //  writeLetter(1,'^');  // zero battery life
     //  shortDelay(5);


  // See Turorial on bottom of this file on how to use the LCD properly

  Blink_LED(); // works as of 2010-06-12
  writeWord("R90",1);
  shortDelay(2);

  STATUS_LED_ON();
  alarm_buzzer(5); // works
  //alarm_buzzer(6);
  //alarm_buzzer(7); // this works fine
  buzzerOff();   // this works too
  //STATUS_LED_OFF();
  count=1;
  Disp_Channel =1;
  while (1) {
     //
     // cycle count
     //
    count++;
    if (count>64000)
    {
      count = 0;
    }


       //for (i = 1; i<=20000;i++) {  // repeats words two times
       //writeWord("*",1);// show this word once
       //writeWord("READY52",1);  // show this word once

       //writeNumber(count);
       //shortDelay(1);


       //
       // jv this port works
       // port 3 works best - it has 4 pins - pin 1,3,5,7
       //
```

```
//P3DIR = 0X0F; // 0xFF all channels to output - 0x0F only 4 channels out - this affects
   button input
//P3OUT = 0X0F | i ;// turn on bits on port 3 - bit 0 of port 3 on - Which is the 2nd pin on EXT
   connector

//
// turn on all channels on port 2
//
 // P2OUT = i; // this works
 //Disp_LED_Meter(
//P3OUT |= 0XFF; // all bits turns on PIN 6
//Port3_Out |=0x00FF

Get_ADC12_A0();
//shortDelay(1); // short delay is needed if sampling continuously - if display results remove
   it
writeNumber(ADC_Results_A0);

//Get_ADC12_A1();
//shortDelay(1);
//writeNumber(ADC_Results_A1);


//
// jv 2010-05-29 05:55 pm
// show the voltage value
// from 0 - 3.6 volts
// 0 - 4046 int max
//


//
// for Accelerometer reading values
// Dimension Engineering
// @ level 1.63 V - 2030-2033 Dec
// Passenger tilt angle upto 90 deg - 1.410 V, @ 45 Deg 1850 (dec) @ 90 1759
// Driver side tilt angle upto 90 deg - 1.851 V
//



//
// convert voltage reading to angle
//


if (ADC_Results_A0>1025)  // driverside tilt 10 degrees
{
  Disp_Channel=1;

}
else
{
  Disp_Channel=0;

}
```

```
//
// switch led bank
//
hitKey = P3IN; // read in current value from port 3
//writeNumber(hitKey);
//shortDelay(1);
if (Disp_Channel==1)
{
   hitKey = 0x0F & hitKey; // get only lower bits 0-3 - strip away the upper bits
   hitKey = hitKey | BIT3; // save all bits - but toggle bit 3 of port 3

}
else
{
 //writeNumber(hitKey);
 //shortDelay(2);

 // 1011 - x0B - turn bit3 off
 // 1111 - x0F

 //hitKey = 0x0F & hitKey; // get only lower bits 0-3 - strip away the upper bits

 hitKey = 0x00; // 0x0B & hitKey; // get only lower bits 0-3 - strip away the upper bits
 //hitKey = hitKey | BIT3; // save all bits - but toggle bit 3 of port 3
}

//writeNumber(hitKey);
//shortDelay(1);

P3OUT = hitKey;



Volt_Meter_Value = abs(1015 - ADC_Results_A0);
Volt_Meter_Value = (Volt_Meter_Value * 7 / Vin_Max);


//
// convert to radian to degree
//

//writeNumber(Volt_Meter_Value);


Disp_LED_Meter(Volt_Meter_Value,Disp_Channel);
alarm_buzzer(Volt_Meter_Value);

//Tmp_No = asin(ADC_Results_A0);

//X_Angle = Tmp_No * 180 / 3.1415926;



//shortDelay(1);
//STATUS_LED_OFF();
```

```
 //P3DIR = 0XFF; // all channels to output
 //P3OUT &= ~0xFF; // turn off all bits
 //BUZZER();

 //writeWord("READY. ",1);}  // show this word once also
 //writeSentence("MSP430F449 REPORTING FOR SERVICE!", 0);

 //
 // test button press
 // jv 2010-05-29 07:06 PM
 getButtons();


 // Update state
//if (count > 60 * (0x8000/CLK_CNT))
//  state = 0x0;
 //
 // check to see if entering setup mode
 // via Button 4
 //

if (!(buttons & BIT4))
{
  // Button 1 - UP
  //clearLCD();
  //writeWord("B1",1);
  //shortDelay(1);
  //Menu_Option--;
  //if ( Menu_Option < 0 )
  //{
  //  Menu_Option = MAX_MENU_OPTIONS -1;
  //}

  //
  // test with buttons and turn on channel 0 - 3 of port 3 ONLY
  //
  hitKey = P3IN;

  //writeNumber(hitKey);
  //shortDelay(1);

  hitKey = 0x0F & hitKey; // get only lower bits 0-3 - strip away the upper bits

  //writeNumber(hitKey);
  //shortDelay(2);


  //
  // & - AND bit operation
  // | - OR
  // ^ - XOR
  // ~ - Complement
  // ! = NOT
  //
```

```
  //hitKey = hitKey ^ BIT0; // save all bits - but toggle bit 0 of port 3
  hitKey = hitKey ^ BIT3; // save all bits - but toggle bit 3 of port 3

  //writeNumber(hitKey);
  //shortDelay(2);

  P3OUT = hitKey;

  //Disp_LED_Meter(0,0); // clear port2 to reset


  /* dont need to toggle any more
  // 2010-06-15 12:26 pm tv
  if (Disp_Channel==0)
  {
     Disp_Channel=1;

  }
  else
  {
     Disp_Channel=0;
     //
     // read in current port state
     // and turn off only channel 0
     //
     //P3OUT = 0x00; // & hitKeys
  }
  */

}
else if (!(buttons & BIT5))
{
  // Button 2 - Down
  //clearLCD();
  //writeWord("B2",1);
  //shortDelay(20);
  //Menu_Option++;
  //if ( Menu_Option > MAX_MENU_OPTIONS)
  //{
  //  Menu_Option = 0;
  //}

}
else if (!(buttons & BIT6))
{
  //button 3 left; decrease value by 1
  //clearLCD();
  //writeWord("B3",1);
  //shortDelay(20);
  //Dec_OPValue( Menu_Option );



}
else if (!(buttons & BIT7))
{
```

```
          //button 4 right; increase value by 1
          //clearLCD();
          Select_MainMenu();
        }



        //Disp_OPSettings( Menu_Option );
        //Menu_Option++;
        //shortDelay(5);
/*
       // ----------------------------------------
       // test menu option text display
       //
       if (Menu_Option  <MAX_MENU_OPTIONS)
       {
         //writeNumber(Menu_Option);
         Disp_OPSettings( Menu_Option );
         Menu_Option++;
         shortDelay(5);

       }
       else
       {
         Menu_Option =0; // reset
       }
*/
}


   LPM3;                    // enter low power mode 3


}

//////////////////////////////////////
//
// 2010-06-07 jv
// Driver side roll over angle DSROA
// DSROA = DEGREES ( ATAN (( LTB - WTCG)/ HTCG)) - driver side roll over angle
// PSROA = DEGREES ( ATAN ( WTCG/HTCG)) - passenger roll over angle
// RWROA = DEGREES ( ATAN ((LWB-WBCG)/HTCG)) Rearward Roll Over Angle
// FWROA = DEGREES ( ATAN ( WBCG/HTCG)) Forward Roll Over Angle
//
// HTCG -> JEEP HEIGHT Center of Gravity-> HF1 + ((WRd * LWB * LWBn) / (Wt * HFd))
//
// WRd = WR2 - WR1 = Weight added to rear axle (lbs)
// HFd = HF2 - HF1 = Height diffrence between front axle level and
// LWBn = SQRT ( LWB ^ 2 - HFd ^ 2) = lenght of the shortened wheelbase elevated
// WBCG = (1 - (WF/Wt))*LWB - wheel base center of gravity behind front axle.
// WTCG = (1 - (WF ???? * LTB - wheel track center of gravity from the passenger side outer tire
          edge

// i would compute the DSROA once a the start of the trip
// compare the actual tilt angle with this max allowed range and compute a scale range from 0 - 7
          to
```

```
// indicate severity.
//

//
// ********* enter Main Menu selection ********************
// jv 2010-06-01 08:01 AM
// enter SETUP - to change current settings
// SAVE to save current settings to specific veh # 1 ( up to 4 )
// LOAD saved vehicle profile to current settings
//
void Select_MainMenu (void)
{
  //
  // stay in the setup up until exit
  // or no key press for while
  //
  char tmp_str[MAX_OPDESC_LEN];
  int idle_time =0;
  char done = 0;
  Main_Menu_Option=0; // default to SETUP

  writeWord("MENU",1);
  shortDelay(10);

  while (!done)
  {
     // show current menu option
     strcpy(tmp_str,MainMenu[Main_Menu_Option]);
     writeWord(tmp_str,1); // description
     shortDelay(3);


        getButtons();
        if (buttons != 0xF0)
        {
          // a key was press reset idle_time
          idle_time = 0;
        }


        // Update state
       //if (count > 60 * (0x8000/CLK_CNT))
       //  state = 0x0;
        //
        // check to see if entering setup mode
        // via Button 4
        //

       if (!(buttons & BIT4))
       {
         // Button 1 - UP
         //clearLCD();
         //writeWord("B1",1);
         //shortDelay(20);
         Main_Menu_Option++;
         if ( Main_Menu_Option >= MAX_MAIN_MENU_OPS)
```

```
  {
    Main_Menu_Option = 0;
  }

}
else if (!(buttons & BIT5))
{
  // Button 2 - Down
  //clearLCD();
  //writeNumber(Menu_Option);
  //shortDelay(10);

  // menu_option is unsigned int
  if ( Main_Menu_Option == 0 )
  {
    Main_Menu_Option = MAX_MAIN_MENU_OPS - 1;
  }
  else
  {
    Main_Menu_Option--;
  }

}
else if (!(buttons & BIT6))
{
  //button 3 left; decrease value by 1
  //clearLCD();

  //
  // exit setup and return to ready mode
  //
  writeWord("READY",1);
  shortDelay(5);
  done =1;


}
else if (!(buttons & BIT7))
{
  //button 4 right; increase value by 1
  //clearLCD();
  //writeWord("B4",1);
  //shortDelay(20);
  //Dec_OPValue( Menu_Option );

  //
  // depending on which menu option selected setup,save,load
  //
  //Change_OpSetting();

  if (Main_Menu_Option==MENU_SETUP_OP)
  {
    Select_OpSetting();
  }
  else if (Main_Menu_Option==MENU_SAVE_OP)
  {
```

```
          //Save_OpSetting();
         }
         else if (Main_Menu_Option==MENU_LOAD_OP)
         {
          //Load_OpSetting();
         }



       }




        //
       // auto time out if idle for a minute or so
       //

        //idle_time++;

        //if (idle_time>1000)
        //{
        //  done=1; //break; // done
        //}



  } // while (1)
}



//
// ********* enter setup mode ********************
// jv 2010-05-30 07:02 PM
// allow users to view and change current operation settings
void Select_OpSetting (void)
{
  //
  // stay in the setup up until exit
  // or no key press for while
  //
  int idle_time =0;
  char done = 0;
  writeWord("SETUP",1);
  shortDelay(10);

  while (!done)
  {
       Disp_OPSettings( Menu_Option );


        getButtons();
        if (buttons != 0xF0)
        {
          // a key was press reset idle_time
```

```
   idle_time = 0;
 }


 // Update state
//if (count > 60 * (0x8000/CLK_CNT))
//  state = 0x0;
 //
 // check to see if entering setup mode
 // via Button 4
 //

if (!(buttons & BIT4))
{
  // Button 1 - UP
  //clearLCD();
  //writeWord("B1",1);
  //shortDelay(20);
  Menu_Option++;
  if ( Menu_Option >= MAX_MENU_OPTIONS)
  {
    Menu_Option = 0;
  }

}
else if (!(buttons & BIT5))
{
  // Button 2 - Down
  //clearLCD();
  //writeNumber(Menu_Option);
  //shortDelay(10);

  // menu_option is unsigned int
  if ( Menu_Option == 0 )
  {
    Menu_Option = MAX_MENU_OPTIONS - 1;
  }
  else
  {
    Menu_Option--;
  }

}
else if (!(buttons & BIT6))
{
  //button 3 left; decrease value by 1
  //clearLCD();

  //
  // exit setup and return to ready mode
  //
  writeWord("EXIT",1);
  shortDelay(5);
  done =1;
```

```
        }
        else if (!(buttons & BIT7))
        {
          //button 4 right; increase value by 1
          //clearLCD();
          //writeWord("B4",1);
          //shortDelay(20);
          //Dec_OPValue( Menu_Option );

          Change_OpSetting();
        }



         //
        // auto time out if idle for a minute or so
        //

         //idle_time++;

         //if (idle_time>1000)
         //{
         //  done=1; //break; // done
         //}



  } // while (1)
}

//
// ********* enter setup mode ********************
// jv 2010-05-30 07:02 PM
// allow users to view and change current operation settings
void Change_OpSetting (void)
{
  //
  // stay in the setup up until exit
  // or no key press for while
  //
  int idle_time =0;
  char done = 0;


  while (!done)
  {
        Disp_OPValue( Menu_Option );
        getButtons();
        if (buttons != 0xF0)
        {
          // a key was press reset idle_time
          idle_time = 0;
        }
```

```
 // Update state
//if (count > 60 * (0x8000/CLK_CNT))
//  state = 0x0;
 //
 // check to see if entering setup mode
 // via Button 4
 //

if (!(buttons & BIT4))
{
  // Button 1 - UP
  //clearLCD();
  //writeWord("B1",1);
  //shortDelay(20);
  Inc_OPValue( Menu_Option );

}
else if (!(buttons & BIT5))
{
  // Button 2 - Down
  //clearLCD();
  //writeWord("B2",1);
  //shortDelay(20);
  Dec_OPValue( Menu_Option );
}
else if (!(buttons & BIT6))
{
  //button 3 left; decrease value by 1
  //clearLCD();
  //writeWord("B3",1);
  //shortDelay(20);
  //Dec_OPValue( Menu_Option );
  done=1;


}
else if (!(buttons & BIT7))
{
  //button 4 right; increase value by 1
  //clearLCD();
  //writeWord("B4",1);
  //shortDelay(20);

  //done=1; // exit
  //Inc_OPValue( Menu_Option );
}




 //idle_time++;

 //if (idle_time>500)
 //{
 //  done=1; // done
```

```
        //}


  } // while (1)
}

// ************************************************ display floating number *********************************

void writeFloat(float number, int decimals)
{
  //char Int_str[9];
  char Dec_str[9];
  char tmp_str[MAX_OPDESC_LEN];
  int Int_No; // Integer whole number portion
  int Dec_No; // decimal portion
  int Ten_Power, Tmp_No; // Number to multiply 10,100,1000 after the decimal

  Int_No = truncf(number);

  Ten_Power = pow(10,decimals);

  Tmp_No = (number - Int_No)*Ten_Power;

  //Dec_No = truncf( Tmp_No);

  //
  // convert integer to str
  //
  itoa(Int_No,tmp_str);
  itoa(Dec_No,Dec_str);

  strcat(tmp_str,".");
  strcat(tmp_str,Dec_str);

  //sprintf(tmp_str, "%f", number);


  writeWord(tmp_str,1);

}


// ************************ init main menu desc ***************
// jv 2010-06-01 07:54 AM
//
void Init_Main_Menu( void )  // initialize main menu array - SETUP,SAVE,LOAD
{
  strcpy(MainMenu[MENU_SETUP_OP],"SETUP"); //
  strcpy(MainMenu[MENU_SAVE_OP],"SAVE"); //
  strcpy(MainMenu[MENU_LOAD_OP],"LOAD"); //

}
// ************************** init OPSettings and OPDesc arrays  *****************
//
void Init_OPSettings_Arrays ( void )
```

```
{
  //
  // set default vehicle characteristic values here
  //

  //
  // jv 2010-06-07
  // init values only if these are 0's otherwise keep what's there.
  //
  if (OPValue[LWB_AI]<=0)
  {
    OPValue[LWB_AI]=95; // Length of wheel base - 95 in.
    OPValue[LTB_AI]=65; // Length of Track - 65 in.
    OPValue[HF1_AI]=17; // in. Height of Front Axle Hub (Center) from ground level
    OPValue[HF2_AI]=42; // in. Height of Front Axle Hub (Center) from ground level while Front is
          elevated
    OPValue[WF_AI] =2600; //  Lbs. Weight on Front Axle at level
    OPValue[WP_AI] =2100; //  Lbs. Weight on passenger side
    OPValue[WR1_AI]=2200; //  Weight on Rear Axle with vehicle level
    OPValue[WR2_AI]=2415; //  Weight on Rear Axle when Front of Vehicle is elevated
  }
  //
  // now define the description of each value
  // to show to end user
  //
  strcpy(OPDesc[LWB_AI],"LWB"); //
  strcpy(OPDesc[LTB_AI],"LTB"); //
  strcpy(OPDesc[HF1_AI],"HF1"); //
  strcpy(OPDesc[HF2_AI],"HF2"); //
  strcpy(OPDesc[WF_AI],"WF"); //
  strcpy(OPDesc[WP_AI],"WP"); //
  strcpy(OPDesc[WR1_AI],"WR1"); //
  strcpy(OPDesc[WR2_AI],"WR2"); //
}
//*****************************************************************************


void Inc_OPValue( int Op_No)
{
  int val;
  val = OPValue[Op_No];
  val ++;
  OPValue[Op_No]=val;
}

void Dec_OPValue( int Op_No)
{
  int val;
  val = OPValue[Op_No];
  val --;
  if (val<0)
  {
    val=0;
  }
  OPValue[Op_No]=val;
}
```

```
//
// display op value no flashing
//
void Disp_OPValue ( int Op_No )
{
  int val;
  //int len;
  //char tmp_str[MAX_OPDESC_LEN];
  //char num_str[5];
  val = OPValue[Op_No];
  //strcpy(tmp_str,OPDesc[Op_No]);

  //writeWord(tmp_str,1); // description
  //shortDelay(3);

  writeNumber(val); // show value
}

// ************* show selected OP option value **************************
void Disp_OPSettings ( int Op_No )
{
  int val;
  //int len;
  char tmp_str[MAX_OPDESC_LEN];
  //char num_str[5];
  val = OPValue[Op_No];
  strcpy(tmp_str,OPDesc[Op_No]);

  writeWord(tmp_str,1); // description
  shortDelay(3);

  writeNumber(val); // show value

  shortDelay(3);

  //writeFloat(val,1);
  //
  // combine desc and value
  //
  //itoa(val,num_str); // convert the integer number into string
  //len = strlen(num_str); // for int value the len is always 5 xxx + /0

  //strcat(tmp_str,":");
  //strcat(tmp_str,num_str);
  //writeWord(num_str,1);
  //writeNumber(len);
  //writeWord(tmp_str,1);
}

//************* show current option value **************************


//******************** getButtons() *************************/
void getButtons(void)
{
  //
```

```
  // get button pressed via port 3
  //
  //unsigned int port3_value = 0;
  //char tmp_char[5];
  //char tmp_str[20] = "button" ;



  //LastBtnPressed = buttons; // save previous buttons value - having this in a loop can cause
          slow ADC
  buttons= P3IN; // read in new value
  // jv 2010-05-30 09:47 am - raw value from inport 3
  // button1 224 (dec) - 0xE0 hex - 1110 0000 bin
  // button2 208 (dec) - 0xD0 hex - 1101 0000 bin
  // button3 176 (dec) - 0xB0 hex - 1011 0000 bin
  // button4 112 (dec) - 0x70 hex - 0111 0000 bin
  // none pressed 240  - 0xF0 hex - 1111 0000 bin
  // all pressed  0    - 0x00 hex - 0000 0000 bin
  // B3 + B4      48   - 0x30 hex - 0011 0000 bin
  // B1 + B2      192  - 0xC0 hex - 1100 0000 bin
  // B1 + B3      160  - 0xA0 hex - 1010 0000 bin
  // B2 + B4      80   - 0x50 hex - 0101 0000 bin

  // convert 123 to string [tmp_char]
  //itoa(port3_value, tmp_char);
  //tmp_str="BUTTON";
  //strcat(tmp_str,tmp_char);

  //writeNumber(buttons);
  //shortDelay(2);
  //buttons = 0;
/*
  if (!(buttons & BIT4))
    buttons |= BIT0;
  if (!(buttons & BIT5))
    buttons |= BIT1;
  if (!(buttons & BIT6))
    buttons |= BIT2;
  if (!(buttons & BIT7))
    buttons |= BIT3;
*/
  //
  // reset number of seconds since last button pressed
  //


//  if (buttons != 0xF0)
//  {
    //
    // at least one button was pressed
    //
    //writeNumber(buttons);


    //
    // this test works
```

```
    // jv 2010-05-30 13:06 PM
    //
 /*
        if (!(buttons & BIT4))
        {
          //state = SOLAR;
          //clearLCD();
          writeWord("B1",1);
          //shortDelay(20);
        }
        else if (!(buttons & BIT5))
        {
          //state = WIND;
          //clearLCD();
          writeWord("B2",1);
          //shortDelay(20);
        }
        else if (!(buttons & BIT6))
        {
          //state = BATTERY;
          //clearLCD();
          writeWord("B3",1);
          //shortDelay(20);
        }
        else if (!(buttons & BIT7))
        {
          //state = 0x0;
          //clearLCD();
          writeWord("B4",1);
          //shortDelay(20);
        }

 */
//  } // if buttons pressed
//  else
//  {

    //
    // no button was pressed
    //
    //SecBtnHeld=0;  // number of seconds button held down.
    //SecLstBtnPressed=0; // number of seconds since last button pressed
    //LastBtnPressed =

    //LastBtnPressed =
    //clearLCD();
//  } // else no button pressed


  //writeWord(tmp_str,1);
  //shortDelay(5);
  //clearLCD();
}

//
// buzzzer alarm 0-7
```

```
//
void alarm_buzzer(char Meter_Value)
{
  //int i=0;
  unsigned int delay;



  if (Meter_Value>=7)
  {
    buzzerOn(17);
  }
  else if (Meter_Value>4)
  {
  //writeNumber(delay);
    //for (i = 1; i<=10;i++)
    //{

  /// buzzeron for 15-20 good values
  // 1-10 too weak
  // 25 > too long of a beep - it drags - no sense of urgent.
  //
      delay=7-Meter_Value;
      buzzerOn(17);
      shortDelay(delay);

      if (Meter_Value<7)
      {
          buzzerOff(); // constant on - flat-liner alarm
      }
    //}
  }
  else
  {
    buzzerOff();
  }
}


//
// display LED meter
// jv 2010-05-29 05:42 PM
// activate 0 - 7 led's
// using int value from 0 to 7
//
void Disp_LED_Meter ( unsigned int Meter_Value, int Disp_Channel )
{
    //
    // Disp_Channel - which display channel left,right,front back etc..
    //

    //
    // set number of LED's on based on value from 0 to 7
    // where 0 all off, 7 all on
    //P2OUT = 0x00;
 /* this does not work as expected
```

```
    switch(Meter_Value)
    {
      case 0: P2OUT =0;
      case 1: P2OUT =(BIT0);
      case 2: P2OUT =(BIT1|BIT0); // 2 LEDS
      case 3: P2OUT =(BIT2|BIT1|BIT0); //
      case 4: P2OUT =(BIT3|BIT2|BIT1|BIT0);
      case 5: P2OUT =(BIT4|BIT3|BIT2|BIT1|BIT0);
      case 6: P2OUT =(BIT5|BIT4|BIT3|BIT2|BIT1|BIT0);
      case 7: P2OUT =(BIT6|BIT5|BIT4|BIT3|BIT2|BIT1|BIT0);
    }
*/

    //
  // jv 2010-06-10 02:41 pm
   // simulate left or right channel using port 3 pin 3
  //
  //Disp_Channel = rand()%10;

/*
  //
  // 2010-06-14 12:20 PM jv
  // switch display LED channel depending on the tilt
  // if 1 then tilt toward the driver side
  // if 0 then tilt toward passenger side
  //
  if (Disp_Channel>0)
  {

    //
    // turn on por 3
    // pin 3
    //

        //
        // test with buttons and turn on channel 0 - 3 of port 3 ONLY
        //
        hitKey = P3IN;

        //writeNumber(hitKey);
        //shortDelay(2);

        hitKey = 0x0F & hitKey; // get only lower bits 0-3 - strip away the upper bits

        //writeNumber(hitKey);
        //shortDelay(2);


        //
        // & - AND bit operation
        // | - OR
        // ^ - XOR
        // ~ - Complement
        // ! = NOT
        //
```

```
        //hitKey = hitKey ^ BIT0; // save all bits - but toggle bit 0 of port 3

        //
        // turn bit 3 ON
        //
        hitKey = hitKey | BIT3; // save all bits - but toggle bit 3 of port 3

        //writeNumber(hitKey);
        //shortDelay(1);

        P3OUT = hitKey;


  }
  else
  {
        hitKey = P3IN;

        //writeNumber(hitKey);
        //shortDelay(2);

        // 1011 - x0B - turn bit3 off
        // 1111 - x0F

        hitKey = 0x0B & hitKey; // get only lower bits 0-3 - strip away the upper bits
        //hitKey = hitKey | BIT3; // save all bits - but toggle bit 3 of port 3
        P3OUT = hitKey;

  }

  shortDelay(1);
*/

  // 2010-06-22 jv
  // new method simply send the binary value to port via 3 bits - 3 channels'
/*
  if (Meter_Value>=7)
  {
    P2OUT = Meter_Value;

  }
  else
  {
    P2OUT = 0x00;
  }
*/



  //* this is the old method where we turn on each port channel to control LED

    if (Meter_Value>=7)
    {
      if (P2IN == 0x07)
      {
        P2OUT = 0x06; // all 7
```

```
      }
      else
      {
        P2OUT = 0x07;  // toggle between 6 and 7 to make 7 blink
      }
      //
      // make this blink
      //

    }
    else if (Meter_Value==6)
    {
      P2OUT = 0x06; // 6 led's
    }
    else if (Meter_Value==5)
    {
      P2OUT = 0x05; // 5 led's
    }
    else if (Meter_Value==4)
    {
      P2OUT = 0x04; // 4 led's
    }
    else if (Meter_Value==3)
    {
      P2OUT = 0x03; // 3 led's
    }
    else if (Meter_Value==2)
    {
      P2OUT = 0x02; // 2 led's
    }
    else if (Meter_Value==1)
    {
      P2OUT = 0x01; // 1 led's
    }
    else
    {
      P2OUT = 0x00; // 0 led's
    }
    //P2OUT = 0x07;

}



//** set port 2 as output
// jv 2010-05-27 18:46
void Init_Port_02( void )
{
    P2DIR = 0xFF ; // set port 2 to output 0x01 ; 0x00 for input
    P2SEL = 0x00 ; // turn all channels under port 2 to off.
}



// ** init i/o port
// IO port ini JV 2010-05-27
```

```
void Init_IO( void )
{
  P1SEL=BIT5;                                   //p1.5 is 32768Hz
  P1DIR=BIT5 | BIT3 | BIT0 | BIT2;                    //BUZ,LED are outputs
}

//
// *** init ADC12 Conversion
//
void Init_ADC12( void )
{
  // Initialize ADC
  P6SEL |= BIT2|BIT1|BIT0;                 // Enable P6.0 channel input
  P6DIR &= ~(BIT2|BIT1|BIT0);
  ADC12CTL0 = ADC12ON+SHT0_8+REFON+MSC;     // ADC12 on, extend sampling time
                                  // to avoid overflow of results
  ADC12CTL1 = SHP+CONSEQ_3;                // Single channel, multiple conversion
                                  // sample and hold pulse mode
  ADC12IE = BIT0;                          // Enable ADC12IFG.0

  ADC12MCTL0 = SREF_1+INCH_0;              // ref+=1.5V, channel = A0
  ADC12MCTL1 = SREF_1+INCH_1;              // ref+=1.5V, channel = A1
  ADC12MCTL2 = SREF_1+INCH_2+EOS;          // ref+=1.5V, channel = A2, last

  for (int i=0; i<0x3600; i++)               // Delay for reference start-up
  {
  }

  ADC12CTL0 |= ENC;                        // Enable conversions

  _EINT();                                 // Enable interrupts
  ADC12CTL0 |= ADC12SC;                    // Start conversion


}
//
// *** To avoid ADC12_VECTOR not defined error include <signal.h>
// jv 2010-05-30 12:50 PM
//
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)
{
  ADC_Results_A0 = ADC12MEM0;                      // Move results
  ADC_Results_A1 = ADC12MEM1;
  ADC_Results_A2 = ADC12MEM2;
}


void Blink_LED (void)
{
  unsigned int i = 0;

  for (i = 1; i<=5;i++) {  // repeats words two times
  //writeWord("*",1);// show this word once
  STATUS_LED_ON();
  shortDelay(2);
```

```
  STATUS_LED_OFF();
  shortDelay(2);
  }
}

// ********************************************** numberScroll ***********************************************
void writeNumber(int long number)   // A cool function that moves number right to left
{
  unsigned int i;                  // dummy variable
  unsigned int digit;               // dummy digit
  char Letter;


  clearLCD();

  for (i=1; i<=9; i++)              // Extract each digit in number, put in an integer array, and count
        total length also
  {
    digit = number%10;              // digit = the least significant character obtained from number for
        display
    number = number/10;              // remove the least significant character from number

    switch(digit)                  // pass on the right char value to writeLetter function
    {
       case 0:  Letter = '0'; writeLetter(i,Letter); break;
       case 1:  Letter = '1'; writeLetter(i,Letter); break;
       case 2:  Letter = '2'; writeLetter(i,Letter); break;
       case 3:  Letter = '3'; writeLetter(i,Letter); break;
       case 4:  Letter = '4'; writeLetter(i,Letter); break;
       case 5:  Letter = '5'; writeLetter(i,Letter); break;
       case 6:  Letter = '6'; writeLetter(i,Letter); break;
       case 7:  Letter = '7'; writeLetter(i,Letter); break;
       case 8:  Letter = '8'; writeLetter(i,Letter); break;
       case 9:  Letter = '9'; writeLetter(i,Letter); break;
    }

    if (number == 0)               // when the number has finally been reduced to zero
       break;                     // break so that LCD doesn't diplay leading zeroes. E.g 234 instead of
         0000234
  }
  shortDelay(2);                   // remove this delay if you update numbers regularly
}



// ********************************************** shortDelay ***********************************************
void shortDelay(int dSpeed)  // a very easy to code delay which keeps the processor busy for a
        small duration of time
{
    unsigned int iDelay = 0;
    unsigned int kDelay = 0;
    for (kDelay = 1;kDelay < dSpeed * 5 ;kDelay++)   // kDelay value can be changed from 10 - 50
    {
      iDelay = 8000;                  // Do not make iDelay more than 40000. Change kDelay instead
      do (iDelay--);
```

```
    while (iDelay != 0);
   }
}

// ********************************************** initLCD ************************************************
void initLCD(void)   // initialize the various registers for LCD to work (code obtained from sample
        demos of MSP430F449)
{
  FLL_CTL0 = XCAP18PF;              //set load capacitance for 32k xtal
  // Initialize LCD driver (4Mux mode)
  LCDCTL = LCDSG0_7 + LCD4MUX + LCDON; // 4mux LCD, segs16-23 = outputs
  BTCTL  = BT_fLCD_DIV128;           // set LCD frame freq = ACLK
  P5SEL  = 0xFC;                    // set Rxx and COM pins for LCD
}

// ********************************************** clearLCD ***********************************************
void clearLCD(void) // makes the LCD blank
{  // clear LCD memory to clear display
  unsigned int iLCD;
  for (iLCD =0; iLCD<20; iLCD++)  // clears all 20 LCD memory segments
  {
    LCD[iLCD] = 0;
  }
}

// ********************************************** writeLetter ********************************************
void writeLetter(int position,char letter) // writes a single character on the LCD. User can specify
        position as well
{
  // DO NOT PLAY WITH THE CODE BELOW ------------------------------------------------------------------
        ------------
  if (position == 1)  {  position = position + 6; } // this is position adjustment for compatibility.
  else if (position == 2 || position == 3 || position == 4 || position == 5 || position == 6 || position
        == 7)
  {  position = ((position * 2) - 1) + 6; }  // adjust position
  // ---------------------------------------------------------------------------------------------------

  switch(letter)
  {
    // letter  // LCDM7                    // LCDM8                    // End
    case 'A':  LCD[position-1] = a + b + c + e;     LCD[position] = b + c + g;       break;
    case 'B':  LCD[position-1] = c + h + e;         LCD[position] = b + c + g;       break;
    case 'C':  LCD[position-1] = a + h;             LCD[position] = b + c;           break;
    case 'D':  LCD[position-1] = b + c + h + e;     LCD[position] = c + g;           break;
    case 'E':  LCD[position-1] = a + h + e;         LCD[position] = b + c + g;       break;
    case 'F':  LCD[position-1] = a;                 LCD[position] = b + c + g;       break;
    case 'G':  LCD[position-1] = a + c + h + e;     LCD[position] = b + c;           break;
    case 'H':  LCD[position-1] = b + c + e;         LCD[position] = b + c + g;       break;
    case 'I':  LCD[position-1] = a + h + f;         LCD[position] = d;               break;
    case 'J':  LCD[position-1] = b + h + c;         LCD[position] = c;               break;
    case 'K':  LCD[position-1] = d + g;             LCD[position] = b + c + g;       break;
    case 'L':  LCD[position-1] = h;                 LCD[position] = b + c ;          break;
    case 'M':  LCD[position-1] = b + c + g;         LCD[position] = b + c + f;       break;
    case 'N':  LCD[position-1] = b + c + d;         LCD[position] = b + c + f;       break;
    case 'O':  LCD[position-1] = a + b + c + h;     LCD[position] = b + c;           break;
    case 'P':  LCD[position-1] = a + b + e;         LCD[position] = b + c + g;       break;
```

```
    case 'Q':  LCD[position-1] = a + b + c + h + d;  LCD[position] = b + c;           break;
    case 'R':  LCD[position-1] = a + b + d + e;      LCD[position] = b + c + g;       break;
    case 'S':  LCD[position-1] = a + c + h + e;      LCD[position] = b + g;           break;
    case 'T':  LCD[position-1] = a + f + b;          LCD[position] = d + b;           break;
    case 'U':  LCD[position-1] = b + c + h;          LCD[position] = b + c;           break;
    case 'V':  LCD[position-1] = g;                  LCD[position] = b + c + e;       break;
    case 'W':  LCD[position-1] = b + c + d;          LCD[position] = b + c + e;       break;
    case 'X':  LCD[position-1] = d + g;              LCD[position] = e + f;           break;
    case 'Y':  LCD[position-1] = b + c + h + e;      LCD[position] = f;               break;
    case 'Z':  LCD[position-1] = a + h + g;          LCD[position] = e;               break;
    //
    // add special char's jv 2010-05-30
    //
    case ':':  LCD[position-1] = f;                  LCD[position] = d+b;             break;


    // number  // LCDM7                          // LCDM8                        // END
    case '0':  LCD[position-1] = a + b + c + h;      LCD[position] = b + c;           break;
    case '1':  LCD[position-1] = b + c;                                               break;
    case '2':  LCD[position-1] = a + b + e + h;      LCD[position] = c + g;           break;
    case '3':  LCD[position-1] = a + b + c + e + h;  LCD[position] = g;               break;
    case '4':  LCD[position-1] = b + c + e;          LCD[position] = b + g;           break;
    case '5':  LCD[position-1] = a + c + h + e;      LCD[position] = b + g;           break;
    case '6':  LCD[position-1] = a + c + h + e;      LCD[position] = b + c + g;       break;
    case '7':  LCD[position-1] = a + b + c;                                           break;
    case '8':  LCD[position-1] = a + b + c + e + h;  LCD[position] = b + c + g;       break;
    case '9':  LCD[position-1] =a + b + c + e ;      LCD[position] = b + g;           break;

    // others
    case '.':                                LCD[position] = h;              break;  // decimal point
    case '^':                                LCDM2 = c;                      break;  // top arrow
    case '!':                                LCDM2 = a;                      break;  // bottom arrow
    case '>':                                LCDM2 = b;                      break;  // right arrow
    case '<':                                LCDM2 = h;                      break;  // left arrow
    case '+':                                LCDM20= a;                      break;  // plus sign
    case '-':                                LCDM20= h;                      break;  // minus sign
    case '&':                                LCDM2 = d;                      break;  // zero battery
    case '*':                                LCDM2 = d + f;                  break;  // low battery
    case '(':                                LCDM2 = d + f + g;              break;  // medium battery
    case ')':                                LCDM2 = d + e + f + g;          break;  // full battery */
  }
}

// ********************************** writeSentence*************************************************************
/*
void writeSentence(const char *word, int scrollForever)  // writes out an entire sentence scrolling it
        right to left
                                        // sentences must be in upper case
{
  unsigned int strLength = 0;         // variable to store length of the sentence
  unsigned int i;                     // dummy variable
  unsigned int j;                     // dummy variable
  unsigned int k;                     // dummy variable
  char letter_list[75];               // keeps track of characters. Sentence can have upto 74 characters.
        Do not make too large
  unsigned int position_list[75];     // keeps track of position of the characters
```

```
unsigned int marker = 0;            // keeps track of index of the last being displayed character
unsigned int dispCount = 0;         // keep count of how many characters are being displayed
unsigned int flag = 1;              // a normal flag that defines if process should be stopped

strLength = strlen(word);           // get the length of the sentence

for (i = 1; i <= strLength; i++)    // put each character in string in a special array called letter_list
{
  letter_list[strLength - i + 1] = word[i-1];
  position_list[i] = 0;             // also, place their relative position values in an array called
       position_list
}

marker = strLength;                 // marker takes the position of the last character
position_list[marker] = 1;          // Set the marker of this position to 1
dispCount++;                        // dispCount = 1; meaning we start display with 1 character

do                                  // Digit Shifter with light + delay
{
  shortDelay(2);                    // take a short break so that the user can see the changes
  clearLCD();                       // since we are gonna update LCD soon, clear the LCD first
  P1OUT ^= 0x08;                    // toggle the pin connected to LED to that we can see it blinking

  for (k = marker; k >= marker - dispCount + 1; k--)   // display the first frame
  {
     writeLetter(position_list[k],letter_list[k]);
  }

  if (dispCount < 7)                // update frame count (characters to be displayed during next
       frame)
    { dispCount++; }

  for (i = marker; i >= marker - dispCount + 1;i--) // shift the relative position values
  {
     position_list[i] = position_list[i] + 1;
     if (position_list[i] == 8)
     {   marker--;                          // shift the marker value

       if (marker - dispCount + 1 <= 0)
       {
          marker = dispCount;               // make sure marker never goes less than zero
       }

     }
  }

  if (position_list[1] == 2)                // when marker has hit maximum index
    {
        if (scrollForever == 0)
        {  flag = 0; clearLCD();  }         // adjust flag value to repeat or not
        marker = strLength;                 // reset marker to original
        dispCount = 1;                      // display length of frame to 1
        for (j = 1; j <= 50; j++)           // reset position_list to original
        {
          if (marker != j)
          { position_list[j] = 0; }
```

```
            else if (marker == j)
            { position_list[j] = 1; }            // set position list of marker character to 1
          }
        }
   } while (flag == 1);                          // function repeats forever if flag remains 1

}
*/
// ********************************************** writeSentence**********************************************
void writeWord(const char *word, int repeat_times)  // displays a word (upto 7 characters) for
          specified number of times
                                        // words must be in upper case
{
  unsigned int strLength = 0;   // variable to store length of word
  unsigned int i;              // dummy variable
  unsigned int k;               // dummy variable

  strLength = strlen(word);     // get the length of word now
  clearLCD();
  for (k = 1; k <= repeat_times; k++)  // repeat display
  {
     for (i = 1; i <= strLength; i++) // display word
     {
       writeLetter(strLength - i + 1,word[i-1]);  // displays each letter in the word
     }

     shortDelay(1);
     // * flashing on the display
     /* jv 2010-05-30 06:50 PM
     shortDelay(3);       // software delay
     clearLCD();          // clears the LCD
     shortDelay(3);       // software delay
     */
  }

}

// ** delay ***
//void Delay (int a)                        //9+a*12 cycles
//{
//  int l;
//  for (l=0 ; l != a; l++);
//}

// ******* turn on buzzer *********
void BUZZER (void)
{
  BUZ1_OFF;
  BUZ2_ON;
  shortDelay(20);
  //Delay(40);
  BUZ2_OFF;
  BUZ1_ON;
  shortDelay(20);
  //Delay(40);
}
```

```
// ********* turn on Status LED *******
void STATUS_LED_ON (void)
{
        P1OUT &= ~STATUS_LED;               //switch on status_led

}

// ********* turn on Status LED *******
void STATUS_LED_OFF (void)
{
    P1OUT |= STATUS_LED;           //switch off status led
}


// ******* use interupt to perform ADC from A0 using 12-bit resolution
// see sample code from fet440_adc12_07.c
// under C:\_WPI\MQP\Sample_Code\slac019j\MSP430F43x, MSP430F44x Code Examples\C
// 2010-05-27 jv
//

// **********************
// initiate reading ADC data from A0 channel
//
void Get_ADC12_A0 (void)
{
  WDTCTL = WDTPW+WDTHOLD;                 // Stop watchdog timer
  P6SEL |= 0x01;                        // Enable A/D channel A0
  ADC12CTL0 = ADC12ON+SHT0_15;            // Turn on ADC12, set sampling time
  ADC12CTL1 = SHP;                        // Use sampling timer, set mode
  ADC12IE = 0x01;                       // Enable ADC12IFG.0
  ADC12CTL0 |= ENC;                       // Enable conversions
  //_EINT();                            // Enable interrupts

  //while(1)
  //{
    ADC12CTL0 |= ADC12SC;                 // Start conversion
    //_BIS_SR(LPM0_bits);                 // Enter LPM0
  //}

    //ADC12CTL0 |= ADC12SC;                 // Start conversion
    while ((ADC12IFG & BIT0)==0);
  ADC_Results_A0 = ADC12MEM0;     // Move results, IFG is cleared
  //_BIC_SR_IRQ(LPM0_bits);         // Clear LPM0

  //* delay
  //shortDelay(1);

}

// **********************
// initiate reading ADC data from A1 channel
//
void Get_ADC12_A1 (void)
{
  WDTCTL = WDTPW+WDTHOLD;                 // Stop watchdog timer
```

```
  P6SEL |= 0x02;                          // Enable A/D channel A1
  ADC12CTL0 = ADC12ON+SHT0_15;            // Turn on ADC12, set sampling time
  ADC12CTL1 = SHP;                        // Use sampling timer, set mode
  ADC12IE = 0x02;                        // Enable ADC12IFG.1
  ADC12CTL0 |= ENC;                       // Enable conversions
  //_EINT();                             // Enable interrupts

  //while(1)
  //{
    ADC12CTL0 |= ADC12SC;                 // Start conversion
    //_BIS_SR(LPM0_bits);                 // Enter LPM0
  //}

    //ADC12CTL0 |= ADC12SC;                 // Start conversion
    while ((ADC12IFG & BIT1)==0);
  ADC_Results_A1 = ADC12MEM0;      // Move results, IFG is cleared
  //_BIC_SR_IRQ(LPM0_bits);         // Clear LPM0

  //* delay
  //shortDelay(1);

}


void LEDdisplayHex(unsigned char num)
{
  unsigned char   tmp_num;

  tmp_num = (~num)<<4;
  P2OUT = tmp_num & 0xF0;
}

/*******************TimerB*********************************/
void timerBSetup(void)
{
  // TBSSEL_1 selects ACLK, 32,768Hz
  // CNTL_0
  // ID_0 sets division to 0, 32,768 tics per second
  // MC_0 turns timer off
  TBCTL = TBSSEL_1 + CNTL_0 + ID_0 + MC_0;

  // Max counter value = 327
  TBCCR0 = 32768;

  // enable interrupts
  TBCCTL0 = CCIE;
  _BIS_SR(GIE);

  TBR = 0; // Zero Timer
}

void ADC12setup(){
  //SHTO_6 sets sample and hold time to 128
  //REFON turns reference generator on
  //REF2_5 sets the reference voltage to 2.5V
  //ADC12ON turn the ADC on
```

```
  ADC12CTL0 = SHT0_6 + REFON + REF2_5V + ADC12ON;
  //Sets SAMPCON signal source to be the sampling timer
  ADC12CTL1 = SHP;
  //Sets the first memory register to store result from A0 and the reference to be Vref+ adn VR- =
          AVss
  ADC12MCTL0 = INCH_0 + SREF_1;
}

void startTimerB(void)
{

  TBCTL |= MC_1; // start timer
}

void stopTimerB(void)
{
  TBCTL = TBSSEL_1 + CNTL_0 + ID_0 + MC_0;
}

// Interrupt for Timer B


#pragma vector=TIMERB0_VECTOR
__interrupt void Timer_B0(void)
{

  char str[6];// tmp char string to show voltage reading
  if( ! paused ){

    //Enables and starts the conversion
    ADC12CTL0 |= ADC12SC + ENC;
    while (ADC12CTL1 & 0x01);
    ADCresults[resultsindex] = ADC12MEM0; // store voltage reading in array
  }else{
    ADCresults[resultsindex] = 0;
  }
  clearLCD();
  itoa(ADCresults[resultsindex]*.61035, str); // converting from int voltage reading to actual
          voltage.
  writeWord(str,1);

  resultsindex++;

  if (resultsindex >= 300) resultsindex = 0; // reset back to beginning of list


}


// ***** What is this function???? *****
// converting int to string upto 4 digits max
// zero leading so 45 -> 0045
//
void itoa(int dec, char* s)
{
  //sprintf(s,"%04d",dec);
```

```
  s[0] = '0' + (dec/1000);
  dec = (dec % 1000);
  s[1] = '0' + (dec/100);
  dec = (dec % 100);
  s[2] = '0' + (dec/10);
  dec = (dec % 10);
  s[3] = '0' + dec;
  s[4] = '\0';
}

/*
// ***** What is this function???? *****
// converting int to string upto 4 digits max
// zero leading so 45 -> 0.045
//
void itoa(long unsigned int dec, char* s) {
  s[0] = '0' + (dec/1000);
  dec = (dec % 1000);
  s[1] = '.';
  s[2] = '0' + (dec/100);
  dec = (dec % 100);
  s[3] = '0' + (dec/10);
  dec = (dec % 10);
  s[4] = '0' + dec;
  s[5] = '\0';
}

related using sprint(buf
 char *itoa(int i)
 {
    char buf[20];
    sprintf(buf,"%d",i);
    return buf;
 }


*/


/*
/****************** setupKeypad() ***********************/
/*
void setupKeypad(void)
{
  P1DIR |= BIT7|BIT6|BIT5;       // P1.7-5 Column Select output
  P1SEL &= ~(BIT7|BIT6|BIT5);    // P1.7-5 I/O Function
  P1OUT |= BIT7|BIT6|BIT5;       // P1.7-5 High

  P2DIR &= ~(BIT3|BIT2|BIT1|BIT0);    // P2.3-0 Row Select input
  P2SEL &= ~(BIT3|BIT2|BIT1|BIT0);    // P2.3-0 I/O Function
}
*/

/****************** getKeys() ***********************/
/*
```

```
void getKeys(void)
{
  unsigned char lookup[]={'1','4','7','+','2','5','8','0','3','6','9','#'};
  unsigned char KY[12];
  unsigned char RW=0x01;
  unsigned char CL=0x20;
  for(int i=0; i<12; i++)
  {
    P1OUT &= ~CL;                // Set current Column
    KY[i] = (P2IN&0x0F)==(~RW&0x0F);
    if(KY[i]==1)                 // Key was pressed
    {
      hitKey = lookup[i];        // Gets key value
      break;
    }
    else
      hitKey=0;                  // no key was pressed
    RW <<=1;                     // Goto next row
    if(RW>0x08)
    {
      P1OUT |= BIT7|BIT6|BIT5;    // P1.7-5 High
      RW=0x01;                   // Goto 1rst Row
      CL <<=1;                   // Goto next column
    }
  }
  if (~(P3IN | ~BIT4)) {
    hitKey = '1';
  } else if (~(P3IN | ~BIT5)) {
    hitKey = '2';
  } else if (~(P3IN | ~BIT6)) {
    hitKey = '3';
  } else if (~(P3IN | ~BIT7)) {
    hitKey = '4';
  }
  P1OUT |= BIT7|BIT6|BIT5;    // P1.7-5 High
}

*/

/******************* initSys() ****************************/
void init_sys(void)
{
  initLCD();                 // Setup LCD for work
  clearLCD();                // Clear LCD display
  //setupKeypad();           // Setup Keypad ports
  STATUS_LED_OFF;
  SecLstBtnPressed=0;
}



// This function sets up timerb to count 0.04 second intervals (25 Hz).
void runtimerb(void)
{
  // Use ACLK, 16 bit, clock division 1, up mode.
  TBCTL = TBSSEL_1 + CNTL_0 + ID_0 + MC_1;
```

```
  TBCCR0 = CLK_CNT;      // Count to 32768 (1 sec)
  TBCCTL0 = CCIE;       // Enable interrupts
}


/***************** buzzerOn() ************************/
void buzzerOn(char period)
{
  FLL_CTL0 |= XCAP10PF;                 // Configure load caps
  P1DIR |= BIT2|BIT0;                  // P1.2,0 output
  P1SEL &= ~BIT2;                      // P1.2 I/O option
  P1OUT &= ~BIT2;                       // P1.2 output = 0
  P1SEL |= BIT0;                      // P1.0 TA0 option
  CCTL0 = OUTMOD_7;                     // CCR0 reset/set
  CCR0 = period;                       // PWM Period
  TACTL = TASSEL_1 + MC_1 + ID_0;       // ACLK, up mode, 1 divider
}


/***************** buzzerOff() ************************/
void buzzerOff(void)
{
  TACTL = MC_0;                   // Stop Timer
  P1DIR |= BIT2|BIT0;             // P1.2,0 output
  P1SEL &= ~(BIT2|BIT0);          // P1.2,0 I/O option
  P1OUT &= ~(BIT2|BIT0);          // P1.2,0 output = 0
}


// **********************************
// button tests
//
    //
    // this test works
    // jv 2010-05-30 13:06 PM
    //
 /*
        if (!(buttons & BIT4))
        {
          //state = SOLAR;
          //clearLCD();
          writeWord("B1",1);
          //shortDelay(20);
        }
        else if (!(buttons & BIT5))
        {
          //state = WIND;
          //clearLCD();
          writeWord("B2",1);
          //shortDelay(20);
        }
        else if (!(buttons & BIT6))
        {
          //state = BATTERY;
          //clearLCD();
          writeWord("B3",1);
```

```
        //shortDelay(20);
      }
      else if (!(buttons & BIT7))
      {
        //state = 0x0;
        //clearLCD();
        writeWord("B4",1);
        //shortDelay(20);
      }

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


/* MSL90: MSP430F449 LCD Driver Code for MSP-449STK-2 Starter Kit from Olimex.com
 ---------------------------------------------------------------------------------------------------------------------
* Author Details   :  Muneem Shahriar
                      Electrical Engineering & Mathematics (Senior)
                      Texas Tech University, Lubbock, TX, USA
                      Email: muneem.shahriar@ttu.edu

* Last Updated      : July 22,2004

* Software:
          Version  : 2.0
          ID       : MSL90
          License  : Freeware
          Features : Has functions that allows the user to:
                      (a) Display integers numbers on LCD
                      (b) Display decimal numbers on LCD (in string form)
                      (c) Display words (upto 7 characters) on LCD
                      (d) Display sentences on LCD by making them scroll right to left
                      (e) Can also display battery life and arrows on LCD

* ---------------------------------------------------------------------------------------------------------------------

* Software requirements:
  1. Microsoft Windows (tested on XP). Website: http://www.microsoft.com
  2. IAR Embedded Workbench (Regular, not Professional). This has the C compiler
     and C-spy debugger to get the code all running. Make sure you select MSP430F449
     in the chip description under Project>>Options>>C-spy
     Website: http://www.iar.com

* Hardware Requirements:
  1. MSP-449STK-2 Starter Kit. This kit can be purchased from http://www.sparkfun.com
     This kit basically comes with a Olimex LCD built-in the kit. Sadly, no software was available
     when I purchased it, so I wrote this file as a driver program.

* Disclaimer:
  This code is specific to the MSP-449STK-2 Starter Kit. I CANNOT be held
  responsible if the code below is tested on other standard kits. If the code
  below is not working on MSP-449STK-2, contact me. All my codes are tested thoroughly.

  Have a starter kit different from MSP-449STK-2 and using Olimex LCD? Let me know!

---------------------------------------------------------------------------------------------------------------------*/
```

```
// ************************************************************************************************************
// This is a freeware, but if you find this software very useful, please consider donating
// me through paypal at muneemonline@hotmail.com
// Remember, I am NOT affiliated with Olimex, Texas Instruments or Sparkfun Electronics.
// ************************************************************************************************************
```