Accessible Weight Scale

Team 2
Eric Bernstein
Maria Elescano
Julie Rosario
Matthew Veilleux

Contact:
Rehabilitation Engineering Research Center on Accessible Medical Instrumentation
National Student Design Competition
John Enderle
860-486-5521
Table of Contents

Abstract ................................................................................................................................. 3
Introduction ............................................................................................................................. 3
Statement of need .................................................................................................................... 4
Product Research ................................................................................................................... 5
  -Market ............................................................................................................................. 5
  -Similar Projects .............................................................................................................. 6
  -Patent Opportunities ..................................................................................................... 7
Constraints ............................................................................................................................ 8
Health and Safety: ............................................................................................................... 10
First Design: ......................................................................................................................... 11
Second Design: .................................................................................................................... 13
Third Design: ....................................................................................................................... 14
Optimal Design: .................................................................................................................... 15
Optimal Design: Overview .................................................................................................... 15
Optimal Design: Objective ................................................................................................. 15
Mechanical Design .............................................................................................................. 15
Mechanical Analysis ........................................................................................................... 16
  -Rail Analysis ................................................................................................................... 16
  -Ramp Analysis .............................................................................................................. 21
Additional Parts .................................................................................................................. 25
Amplifiers and Filtering ....................................................................................................... 30
  Instrumentation Amplifier and Low Pass Filter ................................................................. 30
  Voltage Subtraction ......................................................................................................... 31
  Final Amplifier Setup ....................................................................................................... 32
Rechargeable Battery Circuitry ............................................................................................. 33
  Batteries ........................................................................................................................... 33
  Rechargeable Circuit Design: .......................................................................................... 33
  Low Battery Indicator Design: ........................................................................................ 36
Power Supply Design: ......................................................................................................... 39
Wireless Testing: .................................................................................................................. 41
Rail Microcontroller Electronics .......................................................................................... 42
Display Microcontroller Electronics .................................................................................... 45
Software ............................................................................................................................... 48
  Rail Program Flow ........................................................................................................... 48
  Display Program Flow ..................................................................................................... 50
Printed Circuit Boards ......................................................................................................... 53
Final Product ........................................................................................................................ 57
Calibration and Testing ....................................................................................................... 59
Final Budget ......................................................................................................................... 60
Conclusion ............................................................................................................................ 60
References ............................................................................................................................ 61
Acknowledgements ............................................................................................................. 62
Appendix A: Display Microcontroller Code ........................................................................ 62
Appendix B: Rail Microcontroller Code .............................................................................. 114

2
Abstract

Many people are required to closely monitor their weight in order to track the progress of certain medical conditions. Diabetes patients must know their weight to determine insulin dosages and renal failure patients must know their weight to make sure they are not retaining excess fluids. However, patients who are restricted to wheelchairs are not able to use the same scales that normal people use to obtain this information. Therefore, a scale capable of accommodating a full wheelchair is necessary for these patients to remain healthy. To measure this weight, two rails are required with high resolution and a high maximum load. Also, since many times a nurse or doctor reading the scale is not on the actual scale, a wireless display could allow reading of the weight either on the scale or some distance away from the scale. An accessible wireless wheelchair scale would benefit many patients and clinicians.

Introduction

An accessible weight scale that can be used either at home by handicapped persons or at a health care facility would be of great use to many patients. The weight scale must be accessible to patients who suffer from a variety of disabilities, such as patients with limited movement of the extremities, general frailty, paraplegics, and wheelchair users. Patients suffering from renal failure, heart failure, diabetes, multiple sclerosis, or others who need to monitor their weight regularly can also benefit from this device.

To better serve the needs of these patients, the accessible wheelchair scale designed in this project has several features to aid the handicapped. Our scale design uses two lightweight and easily transported rails to accommodate a wheelchair. A ramp allows easy wheelchair access to the scale platform, the dual rails permit adjustment of the rail distance for different size wheelchairs, a carpet and wood trim around an aluminum core add aesthetic quality to the scale without sacrificing strength, and two beam load cells on each side produce a variable difference voltage for measuring the patients weight. The actual patient weight is averaged over two seconds and transmitted to a wireless display which determines the desired user functions and
calculates the weight. The display incorporates a large display and keypad to better serve handicapped individuals. Functions of the display unit include tarring the scale, setting a wheelchair weight to subtract from the current reading, and a two user memory to store thirty data values each. Additionally, the rail and display electronics include an integrated rechargeable NiMH battery system with a low battery indicator to eliminate the need for users to replace batteries, instead when the low battery light is activated the device must be plugged in for three hours to be recharged to full capacity. This scales versatile design makes it ideal for either home or hospital use and the many features can better help patients monitor and control their weight or treatment methods over time.

Statement of need

Many clients suffer from paralysis, renal failure, heart failure, multiple sclerosis, stroke, diabetes, and heart attack, and need to monitor their weight regularly. The following is a description of these conditions and why weight monitoring is important:

- **Renal failure** is a loss of the ability of the kidneys to excrete wastes, concentrate urine, and conserve electrolytes. Those suffering from kidney failure need to measure their weight often because their body may be retaining excess fluid.

- **Heart failure** is a condition where the heart cannot pump enough blood throughout the body. The weakening of the heart’s pumping ability causes buildup of fluid in the feet, ankles, and legs. Weighing oneself is necessary because a sudden weight gain could mean extra fluid building up in the body.
• *Multiple sclerosis* (MS) is a chronic, energy-taxing, debilitating disease that affects the brain and spinal cord. Those with this illness weigh themselves often since weight gain is common in people who are less active, since fewer calories are burned.

• A *stroke* is a complication that affects the blood vessels that supply blood to the brain. Being overweight increases the chance of developing high blood pressure, heart disease, atherosclerosis and diabetes — all of which increase stroke risk.

• *Diabetes* is an illness where either the body does not produce enough insulin or the cells ignore the insulin. Insulin is necessary for the body to be able to use sugar. Being overweight or obese is a leading risk factor for developing type 2 diabetes.

• A *heart attack* occurs when the blood supply to part of the heart muscle itself is severely reduced or stopped. Maintaining appropriate body weight improves cardiovascular health.

**Product Research**

- **Market**

  Market research shows the price range for similar weight scale designs to be between $1,500 and $3,000.¹ Many of the scales found were offered from Detecto Medical Supplies. Some of the common features are AC or battery power, a LCD, and portability. To compete with the other products on the market, our scale should accommodate as much of these features as possible.

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>Cost</th>
<th>Features</th>
<th>Max Weight</th>
<th>Rail Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detecto</td>
<td>6550</td>
<td>2860.99</td>
<td>LCD display, Keypad</td>
<td>800lb</td>
<td>28”Wx28”L</td>
</tr>
</tbody>
</table>

¹ [www.itinscales.com/phealth1.htm](http://www.itinscales.com/phealth1.htm)
<table>
<thead>
<tr>
<th></th>
<th>Tare, Battery Powered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detecto 25495</td>
<td>1254.93 Mechanical, dual sided balance, removable ramp, 350lb 30”Wx 26”L</td>
</tr>
<tr>
<td>Tanita PWC 620</td>
<td>Not listed Battery or AC power source, foldable, wheels for easy mobility, tare, adjustable leveling feet, 440lb 40”Wx36.25”Dx4”H</td>
</tr>
<tr>
<td>Health O Meter</td>
<td>2450KL 1495 Rail, motion sensing weighing technology, foldable, 600lb Small</td>
</tr>
<tr>
<td>Detecto 475</td>
<td>865.49 Lift-away arms and footrest, portability, transport, 350lb Chair Scale: N/A</td>
</tr>
</tbody>
</table>

-Similar Projects

Two similar products were found in the *NSF Engineering Senior Design Projects to Aid Persons with Disabilities*. The first is “A Scale for Weighing a Client While in the Wheelchair” (1999). This design is that of a portable scale, with a cost of only $300. It has two LED displays that come from two bathroom scales that are mechanically attached to the rail. For this scale, assistance is needed because a calculator is used to compute the user’s weight, and the weight of the wheelchair must be known.

The second design is “Visual Output Weight Scale for Wheelchair Users” (2004). This device uses load cells, which allows for more accurate measurements. It too, is a portable weight scale, with an estimated cost of $1,155. It allows the user to calibrate the weight of the wheelchair, or manually enter its weight. No assistance is needed to use this scale.
-Patent Opportunities

There is great opportunity for obtaining a patent for our design, since only one patent for a similar design was found after searching the United States Patent and Trademark Office. The patent for this device, Wheelchair Portable Scale Apparatus, was filed on March 19, 2003, and the patent number is D489,279.

Engineering Standards

With all mechanical designs the engineer should keep in mind a large factor of safety. Components can not just be designed to minimally support the loads specified in the product description, rather they must go a certain factor above this depending on the application. Furthermore, this device is for use by humans with disabilities and as such it must also adhere to special medical requirements. The Americans With Disabilities Act (ADA) states that ramps for wheelchairs need to have a slope of less than 10 degrees. Further health considerations will be expressed in following sections. The National Society of Professional Engineers ® provides a good guideline for product design and fabrication within ethical constraints. It is entitled the NSPE Code of Ethics for Engineers. A few fundamental rules include:

“1. Hold paramount the safety, health and welfare of the public.

2. Perform services only in areas of their competence.

3. Issue public statements only in an objective and truthful manner.

4. Act for each employer or client as faithful agents or trustees.

5. Avoid deceptive acts.

6. Conduct themselves honorably, responsibly, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.”

These laws were kept in mind during the entire design process.

---

2 http://www.nspe.org/ethics/eh1-code.asp
Constraints

The constraints for this project fall into several categories: economic, environmental, sustainability, manufacturability, ethical, health and safety, social, and political. There are several scales on the market already, so economic concerns played a large role in the design. The scale stands out because it is wireless and has desirable electronic components with a high degree of accuracy. The power consumption is low to reduce operating costs and the product itself should cost no more than $2000. To minimize the harmful environmental effects that are inherent in any manufacturing process, the amount of silicon and batteries used is as small as possible. Silicon waste is quickly becoming a concern as computers are replaced every few years and typically are not recycled. Battery disposal has long been detrimental to the environment because the materials are toxic, so rechargeable batteries will be used in the completed scale. Power consumption also becomes an issue; the scale will turn off automatically and the electronic components absorb as little power as possible. The materials were selected for their durability weight and cost. They are light and strong enough to support the load without deforming too much but are also cost efficient. Selection of recyclable materials limits the impact the scale will have on the environment at the end of its operational lifetime. The timeline for this project is relatively short and manufacturing the designed parts will no doubt take a good portion of the time. The parts therefore are designed with a consideration for fabrication time. Handicapped individuals necessitate a design in which there side stops to prevent the wheelchair
from falling off the scale. Social and political factors do not pose any foreseeable design constraints or issues at this time.
Health and Safety:

In any project design involving human interaction safety is of utmost importance. Special consideration was taken to ensure that the scale will be safe and reliable even for someone living alone and confined to a wheelchair. The scale will have no exposed electrical work of any kind and fail-safes designed to prevent electrical shock will be incorporated into the circuitry. In addition wireless standards for hospitals must be followed. The American Society for Healthcare Engineering and the American Hospital Association working with the Federal Communications Commission established a dedicated frequency spectrum for all health care wireless telemetry. Since the scale is designed to permit hospital use the electromagnetic interference will be kept to a minimum to prevent the malfunctioning of other medical equipment.

Welded joints and other joined mechanical components have to be tightened to specifications and solid. To the aim of general safety for the operator, wheelchair stops are incorporated into the design. The stops prevent the chair from rolling off the scale resulting in injury to the user. The ramp leading to the scale will also have carpet to prevent the wheels of the chair from slipping during ascent. As mentioned previously the ramp must also adhere to the minimum inclination guidelines established by the ADA.

Batteries, though usually safe, can pose hazards if not cased properly due to the acid inside. Again, good connections must be made to prevent overheating of the battery and surrounding casing.
First Design:

Differences - This design uses an alternative miniature button load cell (figure 1) to measure the load at four corner locations on the plate. Other major differences include lack of wireless capabilities and a thinner plate to decrease cost.

Figure 1.0

Advantage - The advantage of this type of cell is the low profile that creates a smaller ramp angle and will generate less material costs. One platform and four small plates used instead of a two large plates.

Disadvantages – The miniature button load cells are extremely expensive and the benefits are outweighed by the cost.

Figure 1.1
Second Design:

Differences - The second design incorporates a wireless component to transfer the output from the load cells to the display.

Advantages - Wireless information transference allows the display to move freely up to 100 feet away from the scale. A wireless transfer would prove useful for physicians wishing to record stored weight values and adds aesthetic value to the scale by removing unnecessary wires. Less exposed wires increases the general safety of the scale as well.

Disadvantages – Miniature button load cells and wireless make for high costs.

Design II

Figure 1.2
**Third Design:**

Differences – Thin film load cells (figure) attached to a large central column. No wireless.

![Figure 1.3](image)

Advantages – Low cost. Thin film load cells are approximately ¼ of the price of a miniature button load cell.

Disadvantages – Central column may make the scale unstable. Two plates will increase cost needlessly.

**Design III**

![Figure 1.4](image)
Optimal Design:

Overview
The optimum design is a combination of designs III and II. While a wireless LCD display will be expensive, the cost can be alleviated by using four custom load cells that can support the weight of the scale to provide stability and obviate the need for a second support plate.

Objective
The scale design has several critical components. The device will have two rails resting on four load cells. The load cells convert the mechanical force into an electrical signal. The electronic signal, from the measuring device, is then interpreted by a microcontroller, which decodes the desired user input from a keypad. Finally, the weight value will be displayed on an LCD for interpretation by the user.

The ramp, also made out of aluminum alloy, has a small angle in order to facilitate access to the scale. The rails and ramp are covered with carpet in order to both increase the coefficient of friction between the rail and the wheelchair and increase the aesthetic quality of the scale. Stops along the sides of the rail are made from aluminum to prevent bending of the rail and covered in mahogany stained wood to cover the welds connecting the different aluminum pieces.

Mechanical Design
The weight scale will be designed to sustain a maximum weight of 800 lb; for the design we will be using a factor of safety of 1.2, which means that the maximum weight we will be using for the calculations is 960 lb. The scale rails will have dimensions large enough to accommodate a standard adult sized wheelchair 10”x36”. The scale ramp will have a small angle of inclination for easy access. The stops, to avoid rolling accidents, will be placed around the rail of the scale. The following is a side view of the weight scale.

---

4 http://www.usdoj.gov/crt/ada/descript/reg3a/figA3ds.htm
Wheelchair Weight Scale

Figure 1.5
Note: Stops will be around the three sides of the rail but the side stops have been left out to show position of the load cells.

Mechanical Analysis

-Rail Analysis
As mentioned before, the rails will be one quarter inch thick aluminum 6061 plate.
Figure 1.6

Plate:
\[ t_p = .25 \text{ in} \]
\[ L_p = 36 \text{ in} \]
\[ W_p = 10 \text{ in} \]
For a plate or sheet the flexural rigidity is

\[ D = \frac{Et^3}{12(1-\nu^2)}, \]

where \( E \) is the modulus of elasticity and \( t \) is the thickness of the cross section. The flexural rigidity for a narrow beam is equal to \( EI \), where \( I \) is the moment of inertia. For a rectangular cross section of one unit width this equation becomes \( Et^3/12 \). A plate therefore, manifests greater stiffness than a narrow beam by a factor of \( 1/(1-\nu^2) \) or about 10\%.\(^5\) Therefore, for the scale rails, a narrow beam will be used as an approximation for the calculation of the bending stress, since the plate is stiffer than the beam.

**Approximation with a beam:**

\(^5\) Ugural and Fenster. *Advanced Strength and Applied Elasticity.*
Figure 1.8

W = Weight of Rails = 50 lb
P = Weight of Patient + Weight of Wheelchair = 600 lb
R = Reactions at strain gages
M1 = Moment about point 1
Fy = Force in the y-direction

Summation of Forces and Moments:

\[ 17 \text{ in} = 1.42 \text{ ft} \quad 34 \text{ in} = 2.83 \text{ ft} \]

\[ \sum F_y = P + W - R_1 - R_2 = 0 \quad \Rightarrow R_1 + R_2 = 650 \text{ lb} \]

\[ \sum M_1 = 650 \times 1.42 - R_2 \times 2.83 = 0 \quad \Rightarrow R_2 = 325 \text{ lb} \quad \Rightarrow R_1 = 325 \text{ lb} \]

Calculation of Stress:

To calculate the stress a small area of on unit width will be used.
σ = Bending Stress

\[ M = \text{Max Moment} = (\text{Load}/2) \times \text{Arm} \]

\[ I = \text{Moment of Inertia} \]

\[ c = \text{Largest Distance from Neutral Axis} \]

Figure 1.9

c = \frac{0.2}{2} = 0.1 \text{ in}

\[ I = 1 \times 0.2^3/12 = 6.667 \times 10^{-4} \text{ in}^4 \]

\[ M = (600 \text{ lb}/2) \times 18 \text{ in} = 5400 \text{ lb}\cdot\text{in} \]

\[ \sigma = \frac{Mc}{I} = \frac{5525 \times 0.1}{6.667 \times 10^{-4}} = 828,709 \text{ psi} \]
-Ramp Analysis

The angle of inclination of the ramp will be equal to $10^\circ$ for easy access to the scale; with the height and the angle we can calculate the bottom length of the ramp.

\[ L_r = \frac{h_r}{\tan \theta} = \frac{3.232}{\tan(10^\circ)} = 18.33 \text{ in} \]
To ensure a safe ramp, bending and frictional forces will be calculated using a worse case scenario. In this case let’s assume that the whole body weight ($P$) is located at the center of the ramp as indicated in the figure below.

\[
\begin{align*}
\alpha & = 85.24^\circ \\
P & = 600 \sin \alpha = 598 \text{ lb} \\
P_x & = 600 \cos \alpha = 49.8 \text{ lb}
\end{align*}
\]

\[F_s = \text{Frictional Force}\]

**Figure 2.1**

**Summation of Forces and Moments:**

\[\sum F_x = P_x - F_s = 0; \quad \Rightarrow \quad F_s = P_x = 49.8 \text{ lb}\]

The frictional force is equal to the normal force times the coefficient of friction ($\mu$); therefore, we have that the required coefficient of friction for a static ramp is:

\[
\mu = \frac{F_s}{P_y} = \frac{49.8}{600} = 0.0825.
\]

**Calculation of Stress:**
\[ c = \frac{0.25}{2} = 0.125 \text{ in} \]
\[ I = \frac{1 \times 0.25^3}{12} = 0.0013 \text{ in} \]
\[ M = \frac{698 \text{ lb/2}}{2} \times 4.01 \text{ in} = 2802 \text{ lb*in} \]
Aluminum Alloy 6061 rectangular bar with a yield strength of 35,000 psi, will be used for the ramp.

**-Stops**

As mentioned before stops will be used to prevent rolling accidents. They will be welded around the rails. Aluminum Alloy 6061 ¼ inch sheet metal remnants from the rail plate stock were used to create the side stops. The welding material between metals and ceramics has a yield strength range of 44,000 to 116,000 psi, and an ultimate strength range of 51,000 to 150,000 psi. The welding product, Dura-FIX Rod, is made of a zinc base that welds metals, and it has a tensile strength of 47,000 psi, a compression strength of 75,000 psi, and a shear strength of 34,000 psi.
Additional Parts

-Hinges

Figure 3.0

The hinge will be simple with holes drilled to connect it to the rails and ramp.
The final rails are very close to the ground. The safety stops will be welded on but the ramp will be attached with hinges to give it a greater degree of freedom.
Load cells

-Strain Gauges

The use of load cells in electronic weight systems is nearly ubiquitous. At its most basic form, a load cell is simply a force transducer that converts a load into an electrical signal via a strain gauge. The strain gauge consists of a thin metallic foil that has been bonded to a dielectric layer. Dielectric materials transmit electrical force using induction rather than conduction. They do not make good conductors but can support an electrostatic field. When a force is applied to a strain gauge the resistance of the gauge changes proportionally. If a voltage is applied during loading the change in resistance will alter the output voltage in a linear manner and the output can be used to calculate the applied force. However, the strain gauge is a delicate thin piece of wire and it cannot be deformed directly without failing. It must be mounted to a strain element using an adhesive. The shape of the strain element can vary; typically beams, rings, or columns are used depending on the function. The adhesives used and the mounting method will have the greatest effect on the quality of the load cell. If there is not a good bond between the two it will introduce errors into the calculations. The strain element and gauge are usually housed in a metal casing to prevent damage during use.

A Wheatstone-Bridge is the only internal electrical component of the load cell. Each of the four legs is connect to a separate strain gauge and when an input voltage is applied the gain in the output becomes proportional to the load. The Wheatstone-bridge also serves to make the voltage output semi-linear.

In addition to the cell there are some peripheral components that allow the device to interface with a computer. The most common addition is an analog to digital converter that
allows the cell to communicate directly with the computer. There may also be indicators, extra cables, printers and scoreboards that are used with the cell. \(^6\)

There are myriad types of load cells on the market but for the aforementioned design the cost must be kept to a minimum so a custom load cell was suggested by SMD sensors.\(^7\) SMD already makes a product line designed for direct integration into a patient’s wheelchair. Based on these specifications, we already know that the load cells can handle human weights and our major task was designing a device that mounted the device for use in a new method. With a cost of only $95, using these cells will reduce the overall cost of the project by several hundred dollars when compared to button cells. The specifications of this cell are as follows:

---

\(^6\) [http://www.measurespec.com/tips/principles.htm](http://www.measurespec.com/tips/principles.htm)

\(^7\) [http://www.smdsensors.com/detail_pgs/s100.htm](http://www.smdsensors.com/detail_pgs/s100.htm)
Figure 3.2
This model created in Autodesk Inventor was utilized in the final scale model for appropriate sizing. Four beam load cells were required with two mounted on either end of each rail. The screws at the end of each cell were used to provide the four points of contact between the scale and the ground to provide stability.

**Amplifiers and Filtering**

For our rail scale we used an instrumentation amplifier and a low pass filter, as well as a voltage subtraction circuit. The instrumentation amplifier circuit provides an output based on the difference between the two input signals from the load cells, times a scale factor. A potentiometer is used to permit adjusting the scale factor or gain of the circuit. The low pass filter passes the signal from the load cell at frequencies lower than the cutoff frequency from the input to the output. Finally, the voltage subtraction circuit removes the offset from the output of the instrumentation amplifier, and makes our output approximately equal to zero volts for the no load condition.
The instrumentation amplifier and the low pass filter were built using a single chip, the AMP04 from Analog Devices. This device is a single instrumentation amplifier designed to work over a +5 volt to a +15 volt supply range. This chip is very accurate, has low power consumption, wide input voltage range, and an excellent gain performance. The following is a schematic of the AMP04 being used as an amplifier and low pass filter:

![Schematic of AMP04](image)

The gain is set by the external resistor $R_{Gain}$ and can vary from 1 to 1000.

$$Gain = 100 \, k\Omega/R_{GAIN}$$

From the above equation we have decided to use a potentiometer equal to 1 kΩ for the gain to be between 100 and 1000.

The cutoff frequency $f_{LP}$ for the low pass filter can be determined from the following equation:

$$f_{LP} = \frac{1}{2\pi (100 \, k\Omega) C_{EXT}}$$

From this equation we have decided on a cutoff frequency of 16 Hz, with a capacitor of 0.1 µF.

**Voltage Subtraction**

As stated earlier the voltage subtraction circuit is used to remove the offset from the output of the instrumentation amplifier. For our design we will be using it to make the output from each load cell equal to 0 to 5 volts for corresponding loads of 0 to 300 lbs (or 0 to 136 kgs). The following is a schematic for the voltage subtraction circuit:
All four resistors will be equal to 10 kΩ. The output voltage from this circuit can be shown to be equal to the following equation:

\[ V_o = \frac{R_1}{R_1 + R_2} * \frac{R_2 + R_3}{R_2} * V_l - \frac{R_4}{R_2} V_2 \]

**Final Amplifier Setup**

For the voltage subtraction circuit we used the MCP619 because it has four opamps; two will be used for the voltage subtraction circuit and one will be used for the low battery indicator circuit. The middle pin of a 100 kΩ potentiometer will go into the V2 input of the voltage subtraction circuit; as for the other two pins, one will go to the 8V power supply and the other will go to ground. The instrumentation amplifier circuit output, from the AMP04, will go into the V1 input of the voltage subtraction. There are four of these amplifier circuits, one for each load cell. The final amplifier schematic will look as follows:
Rechargeable Battery Circuitry

Batteries
After extensive research and after careful review of our specifications and requirements, we decided to use Nickel Metal Hydride (NiMH) Batteries for the following reasons:

- High energy density, for longer time between charges and longer running time;
- Rapid charge;
- Excellent life cycle compared to other rechargeable batteries; &
- Excellent discharge profile.

The batteries chosen had a nominal voltage capacity of 1.2V per cell and were standard AA battery size.

Nickel Metal Hydride

<table>
<thead>
<tr>
<th>Dimensions (with Tube) (mm)</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimensions (with Tube) (mm)</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nominal Voltage</th>
<th>1.2V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discharge Capacity*</td>
<td>Average**</td>
</tr>
<tr>
<td>Rated (Min.)</td>
<td>1500mAh</td>
</tr>
<tr>
<td>Approx. Internal impedance at 1000Hz at charged state.</td>
<td>23mΩ</td>
</tr>
<tr>
<td>Charge</td>
<td>Standard</td>
</tr>
<tr>
<td>Rapid</td>
<td>1550mA</td>
</tr>
<tr>
<td>Rapid</td>
<td>1500mA</td>
</tr>
<tr>
<td>Charge</td>
<td>Standard</td>
</tr>
<tr>
<td>Rapid</td>
<td>0°C to 45°C</td>
</tr>
<tr>
<td>Rapid</td>
<td>6°C to 40°C</td>
</tr>
<tr>
<td>Discharge</td>
<td>-10°C to 65°C</td>
</tr>
<tr>
<td>Storage</td>
<td>&lt; 1 years</td>
</tr>
<tr>
<td>Storage</td>
<td>&lt; 3 months</td>
</tr>
<tr>
<td>Storage</td>
<td>&lt; 1 month</td>
</tr>
</tbody>
</table>

Rechargeable Circuit Design:

Our device has a rechargeable circuit incorporated. As such, the user should never have to remove the batteries from the electronic boxes. To recharge the batteries the user would just have to plug the power cord into the device. Component MAX712CPE from Maxim Integrated
Products was used. The following is the recharging circuit built:

![Recharging Circuit Diagram]

The following calculations were done in order to obtained the values of \( R_1 \) and \( R_{sense} \) (In accordance to the component’s data sheet):

\[
R_1 = \frac{\text{Minimum wallcube voltage} - 5V}{5mA} = \frac{11.1 - 5}{0.005} = 1220\Omega \approx 1.3K\Omega
\]

The MAX712CPE fast-charge NiMH batteries from a DC source at least 1.5V higher than the maximum battery voltage. Since we have 8 batteries of 1.2V each, our minimum wall-cube voltage equals 11.1 V. However, for our design we are using a 12V DC source.

\[
I_{fast} = \frac{\text{Battery's Capacity (mAh)}}{\text{Charge time (hr)}} = \frac{1550}{1.2} = 1291.67mA \approx 1.29A
\]

\[
R_{sense} = \frac{.25V}{I_{fast}} = \frac{.25}{1.29} = .1935\Omega
\]

The value used for \( R_{sense} \) was \(.56\Omega \) (3W), which gave us a re-charging time of 3.5 hours as shown:

\[
R_{sense} = .56\Omega = \frac{.25V}{I_{fast}} \Rightarrow I_{fast} = .446A = \frac{1.55Ah}{Charg e\ time} \Rightarrow Charge\ time = 3.472hr \approx 3.5hr
\]

In order to test the recharging circuit, 4 batteries (4.8V) were totally drained with a 50\Omega\ resistor
for 16.15 hours:

\[ I = \frac{4.8V}{50\Omega} = 96mA \]

(I is more than the entire device’s current draw)

\[ \text{Time before total depletion} = \frac{\text{Discharge Capacity (mAh)}}{I} = \frac{1550}{96} = 16.15hr \]

The drained batteries were then charged with the recharging circuit for about 3.5 hours. In order to test the battery’s discharge capacity, the depleted batteries were connected to a digital multimeter and the following Labview program, which recorded the battery’s voltage value over time.

It took about 13 hours before the batteries depleted as shown below, the expected capacity was approximately 15 hours, which could be explained by the batteries requiring slightly more charge time that indicated:
Low Battery Indicator Design:

A low battery indicator has been implemented into our design. A 2.5 V shunt voltage regulator (LT1431IZ from Linear Technology) compared with resistor bridge from battery supply voltage is used. The following is the circuit built:

When the operational amplifier receives less than 2.5 V from the resistor bridge, the LED light will turn on, indicating that the batteries are low and that they need to be recharged. This would occur when the voltage value of the batteries reaches approximately 9.35V, according to the following calculations:
According to the battery’s discharge characteristics the nominal battery voltage value drops to 1.15V. For 8 batteries that would be 9.2V, but in order to enact a safety factor we have designed it for 9.35V.
The following was the circuit built for the 2.5V voltage regulator:

![Circuit Diagram](image)

The value of $R_L$ was determined by trial and error. The circuit was tested with different values for $R_L$, but the best output obtained was for $R_L = 1\, \Omega$ as shown:

The resistor values of the 100K$\Omega$ potentiometer was preset to: $R_1 = 72.3k\Omega$ and $R_2 = 27.7k\Omega$. Which would give us the 2.59V as shown by the following equation:

\[
Vo = \frac{R_2}{R_1 + R_2} \times V_{in} \Rightarrow 2.59V = \frac{100K\Omega - R_1}{R_1 + 100k\Omega - R_1} \times 9.35V \Rightarrow R_1 = 72.3K\Omega
\]
\[ R_2 = 100K\Omega - R_1 \Rightarrow R_2 = 27.7k\Omega \]

Once the circuit was built and the potentiometer tuned, the cutoff voltage was 9.36V.

**Power Supply Design:**

The recommended voltage input value for the load cells is 10 Volts. However, it can receive a voltage input ranging from 1V to a maximum of 20V. Our first option was to build a 10V linear regulator to provide the required voltage. However, after research was done, such component required a minimum input of 15V, which will require a minimum of 13 (1.2V) NiMH batteries for each rail. To minimize the number of batteries needed, we decided to build a 10V DC/DC converter to supply the required voltage to the load cells; however, after extensive research we were unable to find a 10V DC/DC converter. But, found a 12V DC/DC converter instead. Component TPS67341P from Texas Instruments was used. TPS6734 requires a minimum voltage input of 3.75V to supply 12V. Therefore, 4 (1.2V) NiMH batteries for rail were required. The 12V DC/DC converter was built, but the output had a lot of noise and it was changing over time, as such, no DC signal was obtained. The output from the 12V DC/DC converter was compared with the output of an 8 Volt Linear Regulator (MC78T08 from Motorola) and the following was obtained: (The 8V linear regulator required an input voltage of at least 9.6V or 8 (1.2V) NiMH batteries.)

As shown the output from the 8V Linear Regulator has negligible amount of noise compared to the output from the 12V DC/DC converter, which had a 1.5V difference between peaks. Therefore, the 8V linear regulator was chosen to reduce the power supply noise which could affect the load cell signal and subsequent weight calculations. In addition, the following factors were considered when making our decision:
Provides constant DC output voltage;
Contains circuitry that continuously holds the output voltage at the design value regardless of changes in load current or input voltage;
Better efficiency, device life longer;
No ripple;
Fast response time to load; &
Fewer external components needed.

The following is the 8V linear regulator circuit built and used in our device, input into each load cell:

A 5V input was needed for proper use of the wireless, the amplifier design, and for the use of the LCD’s. As such, a 5V linear regulator was used. (LM323 from On Semiconductor) The component required a minimum input voltage of 7.5V. Therefore a minimum of 7 batteries would be needed for the design. However, because the battery holders we were using hold four batteries, we used 8 1.2 V NiMH batteries. The component was tested, and the following result was obtained:
As shown, the output signal had no noise, allowing for better efficiency. The following is the 5V linear regulator circuit built and used in our device:

**Simplified Application**

![Simplified Application Diagram]

**Wireless Testing:**

The Linx SC-PA series allows bidirectional transmission of USART data and is also pre-certified by the. The wireless component was tested by giving an input of 5V to pin 5, 7, 8, 9, and 10.
A function generator was connected to pin 9 so that we can be certain that the period of the square wave transmitted was less than 3.5ms, as stated on the component’s data sheet. The frequency was adjusted and a period of 220us was given. The output obtained from the receiver was the same as the input from the transmitter, thus proving the functionality of the wireless transceiver for our design.

**Rail Microcontroller Electronics**
A PIC16F737 microcontroller was selected for this device for a variety of reasons. It contains eleven on board A/D converters which allows for multiple load cells if required. An internal oscillator block simplifies oscillator selection to a simple programmable code. A USART interface allows the processor to access a computer through a serial port when the proper RS-232 interface chip is selected. The SPI bus permits expansion of the device by allowing it to communicate with peripheral devices. And the multitude of general purpose I/O ports allows for an interface to the keypad and LCD display. Additionally, the 16F737 is equipped with Microchips low power nanowatt technology to limit the power dissipation and increase electrical life when operating on battery power.
After careful consideration, digital transmission was selected over possible wireless transmission of the output from the strain gauges because it greatly reduces error rates in the transmission of the data. As a result, a microcontroller will be required to convert the analog output into a digital form, and then transmit that data over the wireless link. The output from the strain gauge amplifiers will be connected to the A/D converter ports. Since there are ten free A/D converter ports on the microcontroller, we can accommodate data collection from up to ten strain gauges or load cells. The ICD ports will remain free for in-circuit debugging and programming using the Microchip ICD 2.

Analysis of the wireless devices on the market led us to the conclusion that Linx Technologies produced the best products to meet our specifications. Although we will most likely require single directional data transmission, the Linx SC-PA series allows bidirectional transmission of USART data. This series also comes pre-certified by the FCC as long as it is used with certain antennas. This feature drastically simplifies signal power consideration. Also, the device has an integrated antenna connector to fit many of Linx’s antenna products. For antenna selection, we had to decide between an internal helical and an external whip antenna. After careful consideration we decided to use a whip antenna because the helical antenna detunes badly when in proximity of large metal objects. Since our entire scale is a large metal object, this will create a worst case interference condition. The whip antenna also detunes, but placing it closer to the rails does not have as much of an interfering effect. The major disadvantage of the whip antenna is that since it is external, it may be damaged by unexpected use of the scale.

With the exception of the USART data lines, this wireless device only requires three other inputs and a single output. The RSSI line, or the received signal strength
indicator line, can be used to evaluate the presence of noise affecting the wireless transmission. Using an A/D converter port the microcontroller can determine if the signal strength is providing an inadequate signal. User notification of this information can indicate that the receiver and transmitter must be moved closer together. A power down line allows the microcontroller to control current usage in the device by shutting it down when it is not needed. The RX and TX enable lines are opposite of each other and must be set appropriately during receive or transmission cycles. Finally, if the signal power is too high, an extra resistor can connect the LVLADJ line to ground in order to reduce signal power by up to 5 dB. Using this device, data can be transmitted wirelessly in the same manner as if the two microcontrollers were connected using a serial port.

Display Microcontroller Electronics
Figure 4.0
The display electronics of the scale will consist of another PIC16F737 connected to a keypad, a series of 7 segment LCD displays, and another wireless transceiver. To receive the wireless signal, we used the same Linx antenna and receiver combination that was used for the rail electronics. Since the signal is only being transmitted between two PICs, it does not require a MAX 232 at either the transmitting or receiving end. Because of the amount of noise inherent in wireless transmission, some sort of error checking algorithm will need to be implemented in the software design to prevent spurious signals.

The keypad is a standard blank legend sixteen button keypad offered by Grayhill. A self-adhesive legend will be used to indicate the desired functions of each key. A clear plastic cover will also protect the keypad from possible environmental damage. When a key on the pad is pressed, two pins of the key pad are connected. Using a series of pull-up resistors and a simple software algorithm, the microcontroller can decode which key was pressed. When each of four of the PIC outputs are pulsed high, the PIC will listen to four input lines. If a key is pressed, when its corresponding column is pulsed, its corresponding row will be pulled high. Successive scanning of the keys in this manner will allow the user to change the microcontroller program flow.

Reading the final output weight can pose a problem for people who are either far away from the display or have poor vision. For this reason, we decided to use extra large 2.21” seven segment LCD displays. Although this eliminates the character functionality of a character LCD, it allows for a significant increase in display size. Since the only major features we wish to accommodate are data scrolling and subtracting an input weight, the sacrifice of characters in the display is negligible. Since each character requires seven input lines, a special driver chip will be needed for each character. Using
the CD4543B chip by Texas Instruments, only a single latch line is required for each character along with four data bus lines. Also, since the chips are driving an LCD instead of a series of LEDs, the phase line of the chips and the backplane of the LCD must be pulsed with a square wave input. Using this technique, only eight data lines are required to clock in all of the proper character values to the four numerical characters required by our display and four of those lines could be multiplexed with the keypad.

Software

Rail Program Flow

Each rail microcontroller is responsible for transmitting the raw load cell data to the display for analysis. Based on our advisors recommendations, the rail was designed to only transmit a signal when the weight changed by a significant amount. Therefore, after the initialization of the scale, the rail only begins a transmit cycle when weight on one of the load cells exceeds ten pounds. Once the measuring cycle is initiated, a two second
delay begins in order to allow the weight to settle. Next the measuring cycle begins by averaging the A/D conversion values over two seconds to reduce the effects of noise. A two second averaging time corresponds to approximately three thousand A/D conversion values which is more than enough to remove much of the higher frequency noise components. The averaged A/D conversion value must then be transferred to the display. Since two load cells provide ten bits of usable data from two different scales, an eight byte wireless data signal was designed for rails. After careful consideration, a single byte message send structure was selected because it minimized the potential for interference affecting the results for all the bytes of the load cell data. The first bit of every byte is used to identify which of the two rails is transmitting. The next two bits identify which of the four bytes for each load cell is being transmitted. Finally the lower five bits contain the data from the load cells, either the high or low bit of the load cell A/D conversion for each cell. The final data structure is outlined below.

Serial send data structure

<table>
<thead>
<tr>
<th></th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte 1</td>
<td>Rail ID</td>
<td>0</td>
<td>0</td>
<td>AN1 average lower five bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 2</td>
<td>0</td>
<td>1</td>
<td>AN1 average upper five bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 3</td>
<td>1</td>
<td>0</td>
<td>AN2 average lower five bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 4</td>
<td>1</td>
<td>1</td>
<td>AN2 average upper five bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Once the data bytes have been determined, the data is sent over the wireless link using the proper combinations of setup lines for the transceiver and the USART transmit port. To avoid signal interference, individual bytes are transmitted in 10 millisecond intervals. Since the data bytes are sent fifteen times to fill the display transmit buffer, the total data send time is approximately half a second. After the data has been sent, the rail
reverts back into a standby mode until a ten pound change is again detected to start a new transmit cycle.

Display Program Flow

The display program actually calculates the final weight based on the load cell outputs and the specified user parameters decoded from the keypad. When a wireless message from a rail is received, an error checking algorithm performs a simple parity...
check on the message. If a signal was invalid, the entire message will be thrown away and the processor will wait until the next message is received. Once a valid message is received, it is sorted and stored into an array of values containing nine total receive values for the given byte. When the load cell data is needed, a bit wise averaging process determines the final load cell value used in the weight calculations. As a result, four incorrect signals can be detected by the wireless link for each byte before an incorrect value is used in the weight calculations. This feature is important since the scale only transmits data when the weight changes by tens pounds.

The main loop of the display microcontroller simply calculates and displays the current weight value then checks to determine if any key on the keypad has been depressed. Selection between pounds and kilograms is accomplished using a selector switch, but since the stored values do not scale when units switch is changed, the switch is intended to be set prior to initializing the system. Three main functions are available using the keypad. First, zeroing the scale is accomplished by simply depressing the “TARE” button on the keypad. The zero value can be set for either nothing on the scale or to an initial wheelchair weight. Next, a wheelchair weight can be entered into the system for subtraction from subsequent measurements. To enter the wheelchair weight, the user first must depress the “SUB” button, at which point the display should start flashing. The user then enters the desired weight to be subtracted followed by depressing the “SUB” button a second time. The entered weight will always be subtracted unless another value is entered or the system is powered down.

The final keypad function is memory access and retrieval. Since many individuals must be able to track their weight over time, our scale incorporates a two user memory
with thirty weight storage values each, translating into a full month of data. By default user 1 is automatically selected at start up, however user 2 is selected by pressing “SRCH”, “2”, and “SRCH”. To store any data value currently displayed, the user must simply verify that the proper user is selected and press the “ENT” key. Memory access is obtained by pressing the “SRCH” key. The most recent data for the currently selected user is displayed automatically, but earlier data can be accessed by pressing the “DWN” key while later data is accessed by pressing the “UP” key. During memory search operations, the memory light remains on to indicate that the current value is not the actual value on the scale. It is important to understand that the stored memory values are stored directly to the microprocessor registers which clear when the device is powered down. Therefore the device batteries must be recharged before it turns off to avoid losing the stored memory values.

The display code uses all of the data registers available of the microcontroller and nearly 75% of the available program memory. The program code was specifically designed to facilitate easy modifications to program execution to affect the major features of the final project. For example, rather than transmission when a weight is loaded, one line of the rail code can be changed for continuous transmission of the current weight value. Also, the units of the scale can be altered by simply changing a multiplier found in the weight calculation algorithm. Finally, the display originally included a low wireless signal strength indicator which was eventually abandoned because since the rails only transmit intermittently, the indicator ended up always being enabled. As a result, an extra LED is already connected to the microprocessor for further expansion of the project to display additional information.
Printed Circuit Boards

When designing the printed circuit boards we used the ExpressSCH and ExpressPCB programs. The ExpressSCH is used to draw the circuit schematic, and this schematic can be linked to ExpressPCB, and it will guide you through the wiring process. ExpressPCB is used for the actual design of the circuit board.

For our project, we have designed two circuit boards, one for the LCD display and one for the rail amplifiers. The following are the schematics for the LCD and the rail circuits:
As mentioned before, the schematics were linked to the ExpressPCB program to design the circuit boards. The following are the circuit board designs for the rail and LCD circuits:
Packaging

Once the circuit boards were designed we started looking for the appropriate electronics boxes. We ordered three Serpac Electronics Enclosures from Digikey, one for
the LCD Display and two for the rail circuits. The following are the pictures and datasheets for the boxes (the first is the rail box, the second is the LCD box):

**Products: S-Series**

152

<table>
<thead>
<tr>
<th>MODEL</th>
<th>LENGTH</th>
<th>WIDTH</th>
<th>HEIGHT</th>
<th>DRAWINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>152</td>
<td>5.630&quot;</td>
<td>3.250&quot;</td>
<td>2.010&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Click to view circuit board diagram ([350/332, 450/432](#))

**FEATURES**

- Two-piece standard design
- Circuit board mounting bosses
- Wall mount option (select models)
- Assembly with four screws
- Customization available
Final Product

Once the printed circuit boards arrived, the electronics were soldered onto the boards, the boxes were machined, and the final device was assembled. Several issues were discovered once all of the components were finally assembled. Calibration of the load cell amplifiers proved difficult to maintain over the entire area of the rail. Therefore a known weight of approximately 190 pounds was calibrated at the center of each of the rails. If the weight is placed at the far edge of each of the rails, an approximately ten pound difference can develop. Calibrating the scale for a center point load seems to be the most logical calibration because that should typically be the location of a subject’s center of mass. Another issue is that the rails should have been slightly wider to allow
more mobility of the patient in the wheelchair. As designed, patients without finer control over their wheelchairs movement may require assistance to get on the scale.
Calibration and Testing

Three healthy individuals were used to properly calibrate the scale to display a desired weight. For initial calibration purposes, an individual stepped in the center of a single rail and the gain of the two amplifiers associated with the rail were adjusted to produce the necessary weight. Once the proper weight was obtained at the center of the scale the weight was positioned at either end of the scale and the gain was adjusted to produce a value as close to the actual weight as possible. However, as stated earlier, at the best calibration point, an accurate reading at the center of the scale would translate into approximately a five to ten pound lighter reading at either end of the rail. Since increasing the gains for the edges of the rail would bring the center out of calibration, the center was chosen as the primary calibration point since this would most likely be where the individual’s would be focused. The tables below demonstrate the results obtained from three individuals standing on both of the rails. Actual weights were obtained on a scale assumed to be properly calibrated which is currently used for patient weights in the University of Connecticut infirmary.

Patient 1
Actual Weight = 187.0 lbs

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>Scale Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>186.5</td>
</tr>
<tr>
<td>2</td>
<td>185.0</td>
</tr>
<tr>
<td>3</td>
<td>188.0</td>
</tr>
<tr>
<td>4</td>
<td>186.0</td>
</tr>
<tr>
<td>5</td>
<td>187.5</td>
</tr>
<tr>
<td>6</td>
<td>187.0</td>
</tr>
<tr>
<td>7</td>
<td>189.5</td>
</tr>
<tr>
<td>8</td>
<td>186.5</td>
</tr>
<tr>
<td>9</td>
<td>187.0</td>
</tr>
<tr>
<td>10</td>
<td>185.5</td>
</tr>
</tbody>
</table>
Patient 2
Actual Weight = 164.5

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>Scale Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>163.0</td>
</tr>
<tr>
<td>2</td>
<td>164.5</td>
</tr>
<tr>
<td>3</td>
<td>163.5</td>
</tr>
<tr>
<td>4</td>
<td>166.0</td>
</tr>
<tr>
<td>5</td>
<td>165.0</td>
</tr>
</tbody>
</table>

Patient 3
Actual Weight = 138.5

<table>
<thead>
<tr>
<th>Trial Number</th>
<th>Scale Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>139.5</td>
</tr>
<tr>
<td>2</td>
<td>137.0</td>
</tr>
<tr>
<td>3</td>
<td>139.0</td>
</tr>
<tr>
<td>4</td>
<td>138.5</td>
</tr>
<tr>
<td>5</td>
<td>138.0</td>
</tr>
</tbody>
</table>

Final Budget

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digi-Key</td>
<td>$242.32</td>
</tr>
<tr>
<td>Allied Electronics</td>
<td>$100.11</td>
</tr>
<tr>
<td>Yarde Metals</td>
<td>$298.00</td>
</tr>
<tr>
<td>Strain Measurement Devices</td>
<td>$394.27</td>
</tr>
<tr>
<td>Digi-Key</td>
<td>$342.02</td>
</tr>
<tr>
<td>Newark</td>
<td>$29.98</td>
</tr>
<tr>
<td>Digi-Key</td>
<td>$99.01</td>
</tr>
<tr>
<td>Digi-Key</td>
<td>$73.70</td>
</tr>
<tr>
<td>PCB Express</td>
<td>$175.26</td>
</tr>
<tr>
<td>Digi-Key</td>
<td>$64.08</td>
</tr>
<tr>
<td>PCB Express</td>
<td>$59.00</td>
</tr>
<tr>
<td>Home Depot</td>
<td>$78.56</td>
</tr>
</tbody>
</table>

Total $1956.31

While we recognize that the cost of the scale is slightly higher than desired, the features of the scale still make it highly competitive for its price in the current wheelchair scale market as illustrated by the Detecto scale researched in the Product Research section of this report.

Conclusion
Development of a wireless wheelchair rail scale is the next step in wheelchair scale technology. The scale designed in this project is much more aesthetically designed than many scales available on the market and thus is an ideal feature in a patient’s home. Separation of the rails and display components allow the patient to reposition the system as desired to better suit the scales environment without needing to run wires along walls or floors which could impede a wheelchair patient’s movement. The wheelchair subtraction and memory storage features of the scale better allow a patient to use the scale without assistance from others. A rechargeable battery system further improves the systems benefits because the user no longer needs to open up the electronics cases when the batteries are fully discharged, instead they can simply temporarily plug the system in to restore full operating capacity. It is our understanding that the features of this rail scale make it highly competitive on the current market and would enhance the lives of wheelchair bound patients who constantly need to monitor their weight.

References

www.mcmaster.com
www.smdsensors.com
www.omega.com
http://www.nspe.org/ethics/eh1-code.asp
Acknowledgements

We would like to thank Dr. Enderle and Chris Liebler.

Appendix A: Display Microcontroller Code

list P=PIC16F737, F=INHX8M, C=160, N=77, ST=OFF, MM=OFF, R=DEC, X=OFF
#include P16F737.inc
#include math.inc
__CONFIG  _CONFIG1, _CP_OFF & _CCP2_RC1 & _DEBUG_OFF & _VBOR_2_0 & _BOREN_1 & _MCLR_OFF & _PWRTE_OFF & _WDT_OFF & _INTRC_IO
__CONFIG  _CONFIG2, _BORSEN_1 & _IESO_OFF & _FCMEN_OFF

;Equates
Bank0 equ H'20'

;Variables

cblock  H'7C'
W_temp
STATUS_temp
PCLATH_temp
dende

cblock  H'20'
BCDhigh
BCDlow
BCDtemp
Binaryhigh
Binarylow
LCDhighBCD
LCDlowBCD
BCDOUT
Programflags
Shiftcount
mathtemp
mathtemp2
keystate
keyvalue
pressedserialsendbyte
serialsendtemp
paritycalc
bitaddloop
bitaddloopprecieve
serialrecievebyte
serialrecievetemp

;BCD output of binary to BCD conversion
;binary input of binary to BCD conversion
;BCD input for sending to the LCD display
;bits of this variable can be used for error flags, see defines
;number of left shifts in binary to BCD algorithm
;upper nibble is key that was pressed, lower is key that is
serialrecievetempcalc
paritytest
receiveparity
rail1AN1low
rail1AN1high
rail1AN2low
rail1AN2high
rail2AN1low
rail2AN1high
rail2AN2low
rail2AN2high
onecountR1AN1B10
onecountR1AN1B32
onecountR1AN1B54
onecountR1AN1B76
onecountR1AN1B98
onecountR1AN2B10
onecountR1AN2B32
onecountR1AN2B54
onecountR1AN2B76
onecountR1AN2B98
onecountR2AN1B10
onecountR2AN1B32
onecountR2AN1B54
onecountR2AN1B76
onecountR2AN1B98
onecountR2AN2B10
onecountR2AN2B32
onecountR2AN2B54
onecountR2AN2B76
onecountR2AN2B98
iterR1AN1L
iterR1AN1H
iterR1AN2L
iterR1AN2H
iterR2AN1L
iterR2AN1H
iterR2AN2L
iterR2AN2H
calcR1AN1L
calcR1AN1H
calcR1AN2L
calcR1AN2H
calcR2AN1L
calcR2AN1H
calcR2AN2L
calcR2AN2H
unitsmultiplierlow
unitsmultiplierhigh
weightsumlow
weightsumhigh
finalbinaryweighthigh
finalbinaryweightlow
zerosettinglow
zerosettinghigh
subtractweightlow
subtractweighthigh
endc

cblock H'AO'
loadcellwirelessdata:80
endc

cblock H'120'
usernumber
oldestdatapointeruser1
oldestdatapointeruser2
displaydatapointeruser1
displaydatapointeruser2
user1data:60
endc

cblock H'190'
usernumberbank3
oldestdatapointeruser1bank3
oldestdatapointeruser2bank3
displaydatapointeruser1bank3
displaydatapointeruser2bank3
user2data:60
endc

;Defines
#define BCDA PORTC,4    ;LCD port definitions, alternate electrical wiring is accommodated by changing these values only
#define BCDB PORTA,5
#define BCDC PORTA,6
#define BCDD PORTA,7
#define LATx000 PORTA,3
#define LAT0x00 PORTA,2
#define LAT00x0 PORTA,1
#define LAT000x PORTB,4
#define Blanking PORTC,3
#define BCDoverflow Programflags,0 ;program error flags, overflow if binary value has more than four digits
#define flashcycle Programflags,1 ;displays current on or off cycle for LCD blinking
#define keypressed Programflags,2 ;indicate whether a key is pressed
#define keyconfirmed Programflags,3 ;confirm that a key was pressed, necessary for debouncing
#define newkeyvalue Programflags,4
#define keyrow1 PORTC,4    ;keypad port definitions
#define keyrow2 PORTA,5
#define keyrow3 PORTA,6
#define keyrow4 PORTA,7
#define keycol1 PORTB,3
#define keycol2 PORTB,2
#define keycol3 PORTB,1
#define keycol4 PORTB,0
#define serialTX9 TXSTA,0
#define wirelesspwd PORTC,0 ;logic low powers down transceiver
#define wirelessRXenable PORTC,1
#define wirelessTXenable PORTC,2
#define maintainwirelessenable Programflags,5
#define validdatareceived Programflags,6
#define newdataR1AN1L railupdate,0
#define newdataR1AN1H railupdate,1
#define newdataR1AN2L railupdate,2
#define newdataR1AN2H railupdate,3
#define newdataR2AN1L railupdate,4
#define newdataR2AN1H railupdate,5
#define newdataR2AN2L railupdate,6
#define newdataR2AN2H railupdate,7
#define lbskgselect PORTA,4 ; set = lbs, clear = kg
#define user1sel usernumber,0
#define memoryled PORTB,5
#define user1led PORTB,7
#define user2led PORTB,6
#define lowsignalwirelessled PORTC,5
#define rdAN0 B'00000001',ADCON0
#define startAD ADCON0,2
#define offAD ADCON0,0

; Macros
MOVLF macro literal, dest
movlw literal
movwf dest
endm

MOVFF macro source, dest
movf source, W
movwf dest
endm

BANK1 macro
bcf STATUS,RP1
bsf STATUS,RP0
endm

BANK0 macro
bcf STATUS,RP1
bcf STATUS,RP1
endm

ADD1616 macro AARG0, AARG1, BARG0, BARG1, CARG0, CARG1 ; AARG0 and BARG0 are the lowest bytes
movf BARG0,W
addwf AARG0, W
movwf CARG0
movf BARG1, W
btfsc STATUS,C
incf BARG1, W
addwf AARG1, W
movwf CARG1
endm

SUB16 macro AARGBH, AARGBL, BARGBH, BARGBL, CARGBH, CARGBL
movf BARGBL, 0
subwf AARGBL,0
movwf CARGBL
movf BARGBH,0
btfss STATUS,C
incfsz BARGBH,0
subwf AARGBH,0
movwf CARGBH
endm

; Vectors
org H'000'
nop
goto Mainprog
org H'004' ; Interrupt vector
goto IntService

Mainprog
  call Initialize
  call enableserialrecieve
  ; call BinarytoBCD
  ; call Write7seg

Mainloop
  call keypad
  btfsc newkeyvalue
  call decodecommand
  call finalizeloadcellvalues
  call calcweight
  MOVFF finalbinaryweightlow, Binarylow
  MOVFF finalbinaryweighthigh, Binaryhigh
  call BinarytoBCD
  call RoundBCD
  MOVFF BCDhigh, LCDhighBCD
  MOVFF BCDlow, LCDlowBCD
  call Write7seg
  call Lowwirelessnesssignaltest
  goto Mainloop

Lowwirelessnesssignaltest
  MOVFLF rdAN0
  movlw d'15'
  movwf mathtemp
  waitSetAN0
  decfsz mathtemp, F
  goto waitSetAN0
  bsf startAD
  nop
  ADwaitAN0
btfsc ADCON0,GO_DONE
  goto ADwaitAN0

MOVFF ADRESH,mathtemp
  bcf offAD

  movlw B'00110011' ; set this value for minimum voltage for low signal strength
  subwf mathtemp,W
  btfsc STATUS,C
  goto lowsignal
  bcf lowsignalwirelessled
  return

  lowsignal
  bsf lowsignalwirelessled
  return

RoundBCD
  MOVFF BCDlow,mathtemp
  bcf mathtemp,7
  bcf mathtemp,6
  bcf mathtemp,5
  bcf mathtemp,4

  movlw H'00'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto rounddown

  movlw H'01'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto rounddown

  movlw H'02'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto rounddown

  movlw H'03'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto roundfive

  movlw H'04'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto roundfive

  movlw H'05'
  subwf mathtemp,W
  btfsc STATUS,Z
  goto roundfive
movlw H'06'
subwf mathtemp,W
btfsc STATUS,Z
goto roundfive

movlw H'07'
subwf mathtemp,W
btfsc STATUS,Z
goto roundfive

movlw H'08'
subwf mathtemp,W
btfsc STATUS,Z
goto roundup

movlw H'09'
subwf mathtemp,W
btfsc STATUS,Z
goto roundup

rounddown
bcf  BCDlow,0
bcf  BCDlow,1
bcf  BCDlow,2
bcf  BCDlow,3
return

roundfive
bsf  BCDlow,0
bcf  BCDlow,1
bsf  BCDlow,2
bcf  BCDlow,3
return

roundup
bcf  BCDlow,0
bcf  BCDlow,1
bcf  BCDlow,2
bcf  BCDlow,3

MOVFF BCDlow,mathtemp
swapf mathtemp,F
incf mathtemp,F
movlw H'0A'
subwf mathtemp,W
btfsc STATUS,Z
goto roundtens

bcf  BCDlow,4
bcf  BCDlow,5
bcf  BCDlow,6
bcf  BCDlow,7

swapf mathtemp,F
movf mathtemp,W
iorwf  BCDlow,F
return

roundtens
bcf    BCDlow,4
bcf    BCDlow,5
bcf    BCDlow,6
bcf    BCDlow,7
MOVFF BCDhigh,mathtemp
bcf    mathtemp,4
bcf    mathtemp,5
bcf    mathtemp,6
bcf    mathtemp,7
incf    mathtemp,F
movlw    H'0A'
subwf    mathtemp,W
btsc    STATUS,Z
goto    roundhundreds
bcf    BCDhigh,0
bcf    BCDhigh,1
bcf    BCDhigh,2
bcf    BCDhigh,3
movf    mathtemp,W
iorwf    BCDhigh,F
return

roundhundreds
bcf    BCDhigh,0
bcf    BCDhigh,1
bcf    BCDhigh,2
bcf    BCDhigh,3
MOVFF BCDhigh,mathtemp
swapf    mathtemp,F
incf    mathtemp,F
movlw    H'0A'
subwf    mathtemp,W
btsc    STATUS,Z
goto    overload
bcf    BCDhigh,4
bcf    BCDhigh,5
bcf    BCDhigh,6
bcf    BCDhigh,7
swapf    mathtemp,F
movf    mathtemp,W
iorwf    BCDhigh,F
return
overload
    MOVLF H'99',BCDlow
    MOVLF H'99',BCDhigh
    return

;stop
    ; nop
    ; nop
    ; nop
    ; nop
    ; nop
    ; goto Mainloop

; testcalc
    ; nop
    ; nop
    ; call finalizeloadcellvalues
    ; nop
    ; nop
    ; nop
    ; call calcweight
    ; nop
    ; nop
    ; call BinarytoBCD
    ; MOVFF finalbinaryweightlow,Binarylow
    ; MOVFF finalbinaryweighthigh,Binaryhigh
    ; call BinarytoBCD
    ; MOVFF B'10011000',BCDhigh
    ; MOVFF B'01110110',BCDlow
    ; MOVFK B'10001111',Binarylow
    ; MOVFK B'00000000',Binaryhigh
    ; call BinarytoBCD
    ; MOVFF BCDhigh,LCDhigh
    ; MOVFF BCDlow,LCDlow
    ; call Write7seg
    ; goto Mainloop

    MOVFLB'10111000',LCDhigh
    MOVFLB'01011011',LCDlow
    MOVFLB'10001111',Binarylow
    MOVFLB'00000000',Binaryhigh
    call BinarytoBCD
    MOVFF BCDhigh,LCDhigh
    MOVFF BCDlow,LCDlow
    call Write7seg
; call testkeypad
; call keypad
; btfsc newkeyvalue
; call sendkey
; call Looptime
; goto Mainloop

Initialize

bcf STATUS,RP1
bsf STATUS,RP0
MOVLF B'01000010',OSCCON ; select 1 MHz internal RC oscillator frequency and select
internal oscillator
oscwait    ; wait until oscillator stabilizes
BTFSS OSCCON, 2
    goto oscwait
MOVLF B'00001110',ADCON1 ; enable one a/d converter on AN0 and remainder I/O
MOVLF B'11100000',OPTION_REG
MOVLF B'00010001',TRISA
MOVLF B'00001111',TRISB
MOVLF B'10000000', TRISC
MOVLF B'11100100',TXSTA
MOVLF B'00100000',PIE1

bcf STATUS,RP0

MOVLF B'01110000',RCSTA ; must enable serial port for send or receive
MOVLF 25,SPBRG
MOVLF B'11000000',INTCON

bcf Blanking ; turn on LCD output
MOVLF 255,PR2 ; timer 2 controls the blinking rate of the LCD's
MOVLF B'01111111',T2CON
MOVLF 0,Programflags
MOVLF 0,keystate
clrf PORTB
clrf PORTA
bcf keypressed
bcf keyconfirmed
bcf newkeyvalue
MOVLF 0,LCDhighBCD
MOVLF 0,LCDlowBCD
MOVLF 0,Binarylow
MOVLF 0,Binaryhigh
MOVLF 'B8',unitsmultiplierlow
MOVLF '0B',unitsmultiplierhigh
MOVLF 0,iterR1AN1L
MOVLF 0,iterR1AN1H
MOVLF 0,iterR1AN2L
MOVLF 0,iterR1AN2H
MOVLF 0,iterR2AN1L
MOVLF 0,iterR2AN1H
MOVLF 0,iterR2AN2L
MOVLF 0,iterR2AN2H
MOVLF 0, zerosettinglow
MOVLF 0, zerosettinghigh
MOVLF 0, subtractweightlow
MOVLF 0, subtractweighthigh

MOVLF 0, onecountR1AN1B10
MOVLF 0, onecountR1AN1B32
MOVLF 0, onecountR1AN1B54
MOVLF 0, onecountR1AN1B76
MOVLF 0, onecountR1AN1B98
MOVLF 0, onecountR1AN2B10
MOVLF 0, onecountR1AN2B32
MOVLF 0, onecountR1AN2B54
MOVLF 0, onecountR1AN2B76
MOVLF 0, onecountR1AN2B98
MOVLF 0, onecountR2AN1B10
MOVLF 0, onecountR2AN1B32
MOVLF 0, onecountR2AN1B54
MOVLF 0, onecountR2AN1B76
MOVLF 0, onecountR2AN1B98
MOVLF 0, onecountR2AN2B10
MOVLF 0, onecountR2AN2B32
MOVLF 0, onecountR2AN2B54
MOVLF 0, onecountR2AN2B76
MOVLF 0, onecountR2AN2B98

MOVLF H'A0', FSR

clearloadcelldata

MOVLF 0, INDF
incfsz FSR, F

goto clearloadcelldata

bcf usersel
bsf user1led
bcf user2led
bsf STATUS, RP1
bcf STATUS, RP0

MOVLF H'25', oldestdatapointeruser1
MOVLF H'25', displaydatapointeruser1
MOVLF H'95', oldestdatapointeruser2
MOVLF H'95', displaydatapointeruser2

bsf STATUS, IRP

MOVLF H'25', FSR

clearbank2

MOVLF 0, INDF

incf FSR, F

movf FSR, W

sublw H'60'

btfs STATUS, Z


goto clearbank2

bsf STATUS, IRP
MOVLF H'95',FSR

clearbank3
  MOVLF 0,INDF
  incf  FSR,F
  movf  FSR,W
  sublw H'D0'
  btfss STATUS,Z
  goto clearbank3

  bcf STATUS,IRP
  bcf STATUS,RP1
  bcf STATUS,RP0

  return

; LCD test program, put in Mainloop
LCDtest
  MOVLF H'12',LCDhighBCD
  MOVLF H'34',LCDlowBCD
  call Write7seg
  goto Mainloop

; displaykey
  ; MOVFF keyvalue,Binarylow
  ; swapf Binarylow,F
  ; call BinarytoBCD
  ; MOVFF BCDhigh,LCDDhighBCD
  ; MOVFF BCDlow,LCDDlowBCD
  ; call Write7seg
  ; bcf newkeyvalue
  ; return

; sendkey
  ; MOVFF keyvalue,serialsendbyte
  ; call serialsend
  ; bcf newkeyvalue
  ; return

; testkeypad
  ; bcf keyrow1
  ; bcf keyrow2
  ; bcf keyrow3
  ; bcf keyrow4
  ; bsf keyrow1
  ; nop
  ; nop
  ; nop
  ; clrf PORTC
  ; return
finalizeloadcellvalues

bcf INTCON,GIE

MOVLFO,calcR1AN1L
MOVLFO,calcR1AN1H
MOVLFO,calcR1AN2L
MOVLFO,calcR1AN2H
MOVLFO,calcR2AN1L
MOVLFO,calcR2AN1H
MOVLFO,calcR2AN2L
MOVLFO,calcR2AN2H

movlw B'00001111'
andwf onecountR1AN1B10,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,0

MOVFF onecountR1AN1B10,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,1

movlw B'00001111'
andwf onecountR1AN1B32,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,2

MOVFF onecountR1AN1B32,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,3

movlw B'00001111'
andwf onecountR1AN1B54,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,4

MOVFF onecountR1AN1B54,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,5
movlw B'00001111'
andwf onecountR1AN1B76,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,6

MOVFF onecountR1AN1B76,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN1L,7

movlw B'00001111'
andwf onecountR1AN1B98,W
sublw 4
btfss STATUS,C
bsf calcR1AN1H,0

MOVFF onecountR1AN1B98,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN1H,1

movlw B'00001111'
andwf onecountR1AN2B10,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,0

MOVFF onecountR1AN2B10,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,1

movlw B'00001111'
andwf onecountR1AN2B32,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,2

MOVFF onecountR1AN2B32,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,3

movlw B'00001111'
andwf onecountR1AN2B54,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,4

MOVFF onecountR1AN2B54,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,5

movlw B'00001111'
andwf onecountR1AN2B76,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,6

MOVFF onecountR1AN2B76,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN2L,7

movlw B'00001111'
andwf onecountR1AN2B98,W
sublw 4
btfss STATUS,C
bsf calcR1AN2H,0

MOVFF onecountR1AN2B98,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR1AN2H,1

movlw B'00001111'
andwf onecountR2AN1B10,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,0

MOVFF onecountR2AN1B10,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,1

movlw B'00001111'
andwf onecountR2AN1B32,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,2

MOVFF onecountR2AN1B32,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,3

movlw B'00001111'
andwf onecountR2AN1B54,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,4

MOVFF onecountR2AN1B54,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,5

movlw B'00001111'
andwf onecountR2AN1B76,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,6

MOVFF onecountR2AN1B76,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN1L,7

movlw B'00001111'
andwf onecountR2AN1B98,W
sublw 4
btfs STATUS,C
bsf calcR2AN1H,0

MOVFF onecountR2AN1B98,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN1H,1

movlw B'00001111'
andwf onecountR2AN2B10,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,0

MOVFF onecountR2AN2B10,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,1

movlw B'00001111'
andwf onecountR2AN2B32,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,2

MOVFF onecountR2AN2B32,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,3

movlw B'00001111'
andwf onecountR2AN2B54,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,4

MOVFF onecountR2AN2B54,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfs STATUS,C
bsf calcR2AN2L,5

movlw B'00001111'
andwf onecountR2AN2B76,W
sublw 4
btfss STATUS,C
bsf calcR2AN2L,6

MOVFF onecountR2AN2B76,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR2AN2L,7

movlw B'00001111'
andwf onecountR2AN2B98,W
sublw 4
btfss STATUS,C
bsf calcR2AN2H,0

MOVFF onecountR2AN2B98,mathtemp
swapf mathtemp,F
movlw B'00001111'
andwf mathtemp,W
sublw 4
btfss STATUS,C
bsf calcR2AN2H,1

bsf INTCON,GIE
return
calcweight
nop
calcweight
nop
MOVLF 0,weightsumlow
MOVLF 0,weightsumhigh
bcf INTCON,GIE
ADD1616
calcR1AN1L,calcR1AN1H,calcR1AN2L,calcR1AN2H,weightsumlow,weightsumhigh
ADD1616
weightsumlow,weightsumhigh,calcR2AN1L,calcR2AN1H,weightsumlow,weightsumhigh
ADD1616
weightsumlow,weightsumhigh,calcR2AN2L,calcR2AN2H,weightsumlow,weightsumhigh

MOVFF weightsumlow,AARGB1
MOVFF weightsumhigh,AARGB0
call unitsselect
MOVFF unitsmultiplierlow,BARGB1
MOVFF unitsmultiplierhigh,BARGB0
bcf PCLATH,4
bsf PCLATH,3
CALL FXM1616U
bcf PCLATH,3
MOVLF H'FFFF',BARGB1
MOVLF H'03',BARGB0
bsf PCLATH,3
CALL FXD3216U
bcf PCLATH,3
MOVFF AARG3,finalbinaryweightlow
MOVFF AARG2,finalbinaryweighthigh
bsf INTCON,GIE
;      nop
;      nop

;SUB16 macro AARGH,AARGBL,BARGBH,BARGBL,CARGBH,CARGBL
SUB16
    finalbinaryweighthigh,finalbinaryweightlow,zerosettinghigh,zerosettinglow,finalbinaryweighthigh
    ,finalbinaryweightlow
    ;      nop
    ;      nop
    ;      movf zerosettinglow,0
    ;      subwf finalbinaryweightlow,0
    ;      movwf finalbinaryweightlow
    ;      movf zerosettinghigh,0
    ;      btfss STATUS,C
    ;      incfsz zerosettinghigh,0
    ;      subwf finalbinaryweighthigh,0
    ;      movwf finalbinaryweighthigh
    ;      movf zerosettinglow,0
    ;      subwf finalbinaryweightlow,0
    ;      movwf finalbinaryweightlow
    ;      btfss STATUS,C
    ;      goto calcweightlessthanzero
    ;      nop
    ;      nop
SUB16
    finalbinaryweighthigh,finalbinaryweightlow,subtractweighthigh,subtractweightlow,finalbinaryweighthigh
    ,finalbinaryweighthigh
    ;      nop
    ;      nop
    ;      btfss STATUS,C
    ;      goto calcweightlessthanzero
    ;      nop
    ;      nop
    return
    calcweightlessthanzero
    MOVLF 0,finalbinaryweightlow
    MOVLF 0,finalbinaryweighthigh
    return
    decodecommand
    movlw H'0C'
    subwf keyvalue,W
    btfsc STATUS,Z
    goto TAREcommand
    movlw H'0E'
subwf keyvalue,W
btfsc STATUS,Z
goto SetSubtractweight

movlw H'0B'
subwf keyvalue,W
btfsc STATUS,Z
goto Searchmem

movlw H'0F'
subwf keyvalue,W
btfsc STATUS,Z
goto Storemem

bcf newkeyvalue
return

unitsselect
btfsc lbskgselect
goto lbsselect
goto kgselect

lbsselect
MOVLF H'B8',unitsmultiplierlow
MOVLF H'0B',unitsmultiplierhigh
return

kgselect
MOVLF H'51',unitsmultiplierlow
MOVLF H'05',unitsmultiplierhigh
return

TAREcommand
bcf newkeyvalue
call finalizeloadcellvalues
MOVLF 0,weightsumlow
MOVLF 0,weightsumhigh
ADD1616
calcR1AN1L,calcR1AN1H,calcR1AN2L,calcR1AN2H,weightsumlow,weightsumhigh
ADD1616
weightsumlow,weightsumhigh,calcR2AN1L,calcR2AN1H,weightsumlow,weightsumhigh
ADD1616
weightsumlow,weightsumhigh,calcR2AN2L,calcR2AN2H,weightsumlow,weightsumhigh
bcf INTCON,GIE
MOVFF weightsumlow,AARGB1
MOVFF weightsumhigh,AARGB0
call unitsselect
MOVFF unitsmultiplierlow,BARGB1
MOVFF unitsmultiplierhigh,BARGB0
bsf PCLATH,3
CALL FXM1616U
bcf PCLATH,3
MOVLF H'FF',BARGB1
MOVLF H'03',BARGB0
bsf PCLATH,3
CALL FXD3216U
bcf PCLATH,3
MOVFF AARGB3,finalbinaryweightlow
MOVFF AARGB2,finalbinaryweighthigh
bsf INTCON,GIE
MOVFF finalbinaryweightlow,zerosettinglow
MOVFF finalbinaryweighthigh,zerosettinghigh
goto Mainloop

SetSubtractweight
bcf newkeyvalue
MOVLF 0,subtractweightlow
MOVLF 0,subtractweighthigh
nop
nop
call enableLCDflash

Entersubtractloop
MOVFF subtractweighthigh,LCDbighigh
MOVFF subtractweightlow,LCDbilow

call Write7seg
call keypad
btfsc newkeyvalue
call decodesubtract
goto Entersubtractloop

Endsubtractloop
bcf newkeyvalue
call disableLCDflash
call convertBCDtobinary
goto Mainloop

;ADD1616 macro AARG0,AARG1,BARG0,BARG1,CARG0,CARG1 ;AARG0 and BARG0 are the lowest bytes
 ; Input:  8 bit unsigned fixed point multiplicand in AARG0
 ;        8 bit unsigned fixed point multiplier in BARG0
 ; Use:   CALL FXM0808U
 ; Input:  16 bit unsigned fixed point multiplicand in AARG0,AARG1
 ;        16 bit unsigned fixed point multiplier in BARG0,BARG1
 ; Use:   CALL FXM1616U
 ; Output: 32 bit unsigned fixed point product in AARG0,AARG1,AARG2,AARG3

convertBCDtobinary
nop
nop
MOVFF subtractweighthigh,BCDhigh
MOVFF subtractweightlow,BCDlow
MOVLF 0,subtractweighthigh
MOVLF 0,subtractweightlow
MOVFF BCDlow,Binarylow
bcf       Binarylow,4
bcf       Binarylow,5
bcf       Binarylow,6
bcf       Binarylow,7
MOVFF Binarylow,subtractweightlow
MOVFF BCDlow,Binarylow
swapf    Binarylow,F
bcf       Binarylow,4
bcf       Binarylow,5
bcf       Binarylow,6
bcf       Binarylow,7
MOVFF Binarylow,AARGB0
MOVLF 10,BARGB0
bsf       PCLATH,3
CALL    FXM0808U
bcf       PCLATH,3
ADD1616   subtractweightlow,subtractweighthigh,AARGB1,AARGB0,subtractweightlow,subtractweighthigh
MOVFF BCDhigh,Binaryhigh
bcf       Binaryhigh,4
bcf       Binaryhigh,5
bcf       Binaryhigh,6
bcf       Binaryhigh,7
MOVFF Binaryhigh,AARGB0
MOVLF 100,BARGB0
bsf       PCLATH,3
CALL    FXM0808U
bcf       PCLATH,3
ADD1616   subtractweightlow,subtractweighthigh,AARGB1,AARGB0,subtractweightlow,subtractweighthigh
MOVFF BCDhigh,Binaryhigh
swapf    Binaryhigh,F
bcf       Binaryhigh,4
bcf       Binaryhigh,5
bcf       Binaryhigh,6
bcf       Binaryhigh,7
MOVFF Binaryhigh,AARGB1
MOVLF 0,AARGB0
MOVLFH'03',BARGB0
MOVLFH'E8',BARGB1
bsf       PCLATH,3
CALL    FXM1616U
bcf       PCLATH,3
ADD1616   subtractweightlow,subtractweighthigh,AARGB3,AARGB2,subtractweightlow,subtractweighthigh
nop
nop
nop
return

decodesubtract
bcf       newkeyvalue
movlw H'00'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractone

movlw H'01'
subwf keyvalue,W
btfsc STATUS,Z
goto subtracttwo

movlw H'02'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractthree

movlw H'03'
subwf keyvalue,W
btfsc STATUS,Z
goto Endsubtractloop

movlw H'04'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractfour

movlw H'05'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractfive

movlw H'06'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractsix

movlw H'07'
subwf keyvalue,W
btfsc STATUS,Z
goto Endsubtractloop

movlw H'08'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractseven

movlw H'09'
subwf keyvalue,W
btfsc STATUS,Z
goto subtracteight

movlw H'0A'
subwf keyvalue,W
btfsc STATUS,Z
goto subtractnine

movlw H'0B'
subwf    keyvalue,W
btfsc    STATUS,Z
goto     Endsubtractloop

movlw    H'0C'
subwf    keyvalue,W
btfsc    STATUS,Z
goto     subtractzero

movlw    H'0D'
subwf    keyvalue,W
btfsc    STATUS,Z
goto     Endsubtractloop

movlw    H'0E'
subwf    keyvalue,W
btfsc    STATUS,Z
goto     Endsubtractloop

movlw    H'0F'
subwf    keyvalue,W
btfsc    STATUS,Z
goto     Endsubtractloop

subtractswap
  swapf   subtractweighthigh,F
  bcf     subtractweighthigh,0
  bcf     subtractweighthigh,1
  bcf     subtractweighthigh,2
  bcf     subtractweighthigh,3
  btfsc   subtractweightlow,4
  bsf     subtractweighthigh,0
  btfsc   subtractweightlow,5
  bsf     subtractweighthigh,1
  btfsc   subtractweightlow,6
  bsf     subtractweighthigh,2
  btfsc   subtractweightlow,7
  bsf     subtractweighthigh,3
  swapf   subtractweightlow,F
  bcf     subtractweightlow,0
  bcf     subtractweightlow,1
  bcf     subtractweightlow,2
  bcf     subtractweightlow,3
  btfsc   mathtemp,0
  bsf     subtractweightlow,0
  btfsc   mathtemp,1
  bsf     subtractweightlow,1
  btfsc   mathtemp,2
  bsf     subtractweightlow,2
  btfsc   mathtemp,3
  bsf     subtractweightlow,3
  goto    Entersubtractloop
subtractone
    MOVLF H'01',mathtemp
    goto subtractswap

subtracttwo
    MOVLF H'02',mathtemp
    goto subtractswap

subtractthree
    MOVLF H'03',mathtemp
    goto subtractswap

subtractfour
    MOVLF H'04',mathtemp
    goto subtractswap

subtractfive
    MOVLF H'05',mathtemp
    goto subtractswap

subtractsix
    MOVLF H'06',mathtemp
    goto subtractswap

subtractseven
    MOVLF H'07',mathtemp
    goto subtractswap

subtracteight
    MOVLF H'08',mathtemp
    goto subtractswap

subtractnine
    MOVLF H'09',mathtemp
    goto subtractswap

subtractzero
    MOVLF H'00',mathtemp
    goto subtractswap

;     oldestdatapointeruser1
;     oldestdatapointeruser2
;     displaydatapointeruser1
;     displaydatapointeruser2

Searchmem
    bsf memoryled
    bsf STATUS,IRP
    bsf STATUS,RP1
    bcf STATUS,RP0
    MOVFF oldestdatapointeruser1,displaydatapointeruser1
    MOVFF oldestdatapointeruser2,displaydatapointeruser2
    movlw H'02'
subwf displaydatapointeruser1,F
 movlw H'02'
 subwf displaydatapointeruser2,F
 movf displaydatapointeruser1,W
 sublw H'23'
 btfsc STATUS,Z
 call resettoendofuser1mem
 movf displaydatapointeruser2,W
 sublw H'93'
 btfsc STATUS,Z
 call resettoendofuser2mem
 btfss usersel
 goto Displayuser1mem
 goto Displayuser2mem

resettoendofuser1mem
 MOVLF H'5F',displaydatapointeruser1
 return
resettoendofuser2mem
 MOVLF H'CF',displaydatapointeruser2
 return

Displayuser1mem
 bsf STATUS,IRP
 bsf STATUS,RP1
 bcf STATUS,RP0
 MOVFF displaydatapointeruser1,FSR
 movf INDF,W
 bcf STATUS,RP1
 movwf LCDlowBCD
 incf FSR,F
 movf INDF,W
 movwf LCDhighBCD
 call Write7seg

call keypad
btfs newkeyvalue
 call decodememory
 goto Displayuser1mem

Displayuser2mem
 bsf STATUS,IRP
 bsf STATUS,RP1
 bcf STATUS,RP0
 MOVFF displaydatapointeruser2,FSR
 movf INDF,W
 bcf STATUS,RP1
 movwf LCDlowBCD
 incf FSR,F
 movf INDF,W
movwf LCDhighBCD  
call Write7seg  
call keypad  
btfsc newkeyvalue  
call decodememory  
goto Displayuser2mem

decodememory  
bcf newkeyvalue

movlw H'00'  
subwf keyvalue,W  
btfsc STATUS,Z  
goto selectuser1

movlw H'01'  
subwf keyvalue,W  
btfsc STATUS,Z  
goto selectuser2

movlw H'03'  
subwf keyvalue,W  
btfsc STATUS,Z  
goto movetomorerecentsave

movlw H'07'  
subwf keyvalue,W  
btfsc STATUS,Z  
goto movetolatersave

movlw H'0B'  
subwf keyvalue,W  
btfsc STATUS,Z  
goto Exitmemorysearch

return

selectuser1  
bcf user2led  
bsf user1led  
bsf STATUS,RP1  
bcf usersel  
bcf STATUS,RP1  
goto Displayuser1mem

selectuser2  
bsf user2led  
bcf user1led  
bsf STATUS,RP1  
bsf usersel  
bcf STATUS,RP1  
goto Displayuser2mem

movetomorerecentsave  
bsf STATUS,IRP
bsf STATUS,RP1
bcf STATUS,RP0
btfss usersel
goto movetomorerecentsaveuser1
goto movetomorerecentsaveuser2

movetomorerecentsaveuser1
  movf displaydatapointeruser1,W
  addlw H'02'
  movwf displaydatapointeruser1

  subwf oldestdatapointeruser1,W
  btfsc STATUS,Z
  call mostrecentdatauser1

  movf displaydatapointeruser1,W
  sublw H'61'
  btfsc STATUS,Z
  call resettobeginuser1

bcf STATUS,RP1
return

mostrecentdatauser1
  movlw H'02'
  subwf displaydatapointeruser1,F
return

resettobeginuser1
  MOVLFH'25',displaydatapointeruser1
return

movetomorerecentsaveuser2
  movf displaydatapointeruser2,W
  addlw H'02'
  movwf displaydatapointeruser2

  subwf oldestdatapointeruser2,W
  btfsc STATUS,Z
  call mostrecentdatauser2

  movf displaydatapointeruser2,W
  sublw H'D1'
  btfsc STATUS,Z
  call resettobeginuser2

bcf STATUS,RP1
return
mostrecentdatauser2
    movlw H'02'
    subwf displaydatapointeruser2,F
    return
resettobeginuser2
    MOVLF H'95',displaydatapointeruser2
    return

movetolatersave
    bsf STATUS,IRP
    bsf STATUS,RP1
    bcf STATUS,RP0
    btfs usersel
    goto movetolatersaveuser1
    goto movetolatersaveuser2

movetolatersaveuser1
    movf displaydatapointeruser1,W
    subwf oldestdatapointeruser1,W
    btfs STATUS,Z
    goto endofmovetolatersaveuser1
    movlw H'02'
    subwf displaydatapointeruser1,F
    movf displaydatapointeruser1,W
    sublw H'23'
    btfs STATUS,Z
    call resettoendofuser1mem
endofmovetolatersaveuser1
    bcf STATUS,RP1
    return

movetolatersaveuser2
    movf displaydatapointeruser2,W
    subwf oldestdatapointeruser2,W
    btfs STATUS,Z
    goto endofmovetolatersaveuser2
    movlw H'02'
    subwf displaydatapointeruser2,F
Storememuser2
  MOVFF oldestdatapointeruser2,FSR
  bcf STATUS,RP1
  bcf STATUS,RP0
  movf LCDlowBCD,W
  bsf STATUS,IRP
  movwf INDF
  incf FSR,F
  movf LCDhighBCD,W
  movwf INDF
  bsf STATUS,RP1
  movf oldestdatapointeruser2,W
  addlw H'02'
  movwf oldestdatapointeruser2
  movf oldestdatapointeruser2,W
  sublw H'61'
  btfsc STATUS,Z
  call resetoldesttobeginuser2
  bcf STATUS,RP1
  bcf STATUS,IRP
  goto Mainloop

resetoldesttobeginuser2
  MOVLF H'25',oldestdatapointeruser2
  return

keypad
  movf keystate,w
  sublw 0
  btfsc STATUS,Z
goto  KEYSTATE0

movf  keystate,w
sublw 1
btfsc STATUS,Z
  goto  KEYSTATE1

movf  keystate,w
sublw 2
btfsc STATUS,Z
  goto  KEYSTATE2

;  movf  keystate,w
;  sublw 3
;  btfsc STATUS,Z
;  goto  KEYSTATE0

;  movf  keystate,w
;  sublw 4
;  btfsc STATUS,Z
;  goto  KEYSTATE0

KEYSTATE0
  call  SCANKEYS
  btfss keypressed
    return
  incf  keystate,F
  swapf  keyvalue,F
  return

KEYSTATE1
  call  SCANKEYS
  btfss keyconfirmed
    decf  keystate,F
    btfsc keyconfirmed
      incf  keystate,F
      swapf  keyvalue,F
    return

KEYSTATE2
  call  SCANKEYS
  btfsc keypressed
    bsf  newkeyvalue
    bcf  keypressed
    bcf  keyconfirmed
    swapf  keyvalue,F
    MOVLF 0,keystate
    return
SCANKEYS

```
movlw B'11110000' ;set lower nibble to zero without affecting upper nibble
andwf keyvalue,F
bcf keyrow1
bcf keyrow2
bcf keyrow3
bcf keyrow4
bsf keyrow1
btfsc keycol1
  goto R1C1
btfsc keycol2
  goto R1C2
btfsc keycol3
  goto R1C3
btfsc keycol4
  goto R1C4
bcf keyrow1

bsf keyrow2
btfsc keycol1
  goto R2C1
btfsc keycol2
  goto R2C2
btfsc keycol3
  goto R2C3
btfsc keycol4
  goto R2C4
bcf keyrow2

bsf keyrow3
btfsc keycol1
  goto R3C1
btfsc keycol2
  goto R3C2
btfsc keycol3
  goto R3C3
btfsc keycol4
```
goto R3C4
bcf keyrow3
bsf keyrow4
btfsc keycol1
goto R4C1
btfsc keycol2
goto R4C2
btfsc keycol3
goto R4C3
btfsc keycol4
goto R4C4
bcf keyrow4

bcf keypressed

return

R1C1 ;row 1 column 1 pressed
movlw B'00000000'
iorwf keyvalue,F
goto debouncepress

R1C2
movlw B'00000001'
iorwf keyvalue,F
goto debouncepress

R1C3
movlw B'00000010'
iorwf keyvalue,F
goto debouncepress

R1C4
movlw B'00000011'
iorwf keyvalue,F
goto debouncepress

R2C1 ;row 2 column 1 pressed
movlw B'00000100'
iorwf keyvalue,F
goto debouncepress

R2C2
movlw B'00000101'
iorwf keyvalue,F
goto debouncepress

R2C3
movlw B'00000110'
iorwf keyvalue,F
goto debouncepress
R2C4
movlw B'00000111'
iorwf keyvalue,F
goto debouncepress

R3C1 ;row 3 column 1 pressed
movlw B'00001000'
iorwf keyvalue,F
goto debouncepress

R3C2
movlw B'00001001'
iorwf keyvalue,F
goto debouncepress

R3C3
movlw B'00001010'
iorwf keyvalue,F
goto debouncepress

R3C4
movlw B'00001011'
iorwf keyvalue,F
goto debouncepress

R4C1 ;row 4 column 1 pressed
movlw B'00001100'
iorwf keyvalue,F
goto debouncepress

R4C2
movlw B'00001101'
iorwf keyvalue,F
goto debouncepress

R4C3
movlw B'00001110'
iorwf keyvalue,F
goto debouncepress

R4C4
movlw B'00001111'
iorwf keyvalue,F
goto debouncepress

debouncepress
btsc keypressed
goto SAMEKEYDEPRESSED
bsf keypressed
return

SAMEKEYDEPRESSED
MOVFF keyvalue,mathtemp
MOVFF keyvalue, mathtemp2
movlw B'00001111'
andwf mathtemp, F
swapf mathtemp2, F
movlw B'00001111'
andwf mathtemp2, W
subwf mathtemp, w
btfsc STATUS, Z
bsf keyconfirmed
return

BinarytoBCD ; takes 16 bit binary value from
Binaryhigh and Binarylow and converts it into a four digit BCD format spanning BCDhigh and BCDlow
MOVLF B'00000000', BCDhigh ; reset BCD to zero
MOVLF B'00000000', BCDlow
bcf BCDoverflow ; clear overflow flag
MOVLF 15, Shiftcount ; initialize 15 shift and add cycles

shiftaddcycle
call Leftshift
call Add3test
decfsz Shiftcount, F
goto shiftaddcycle ; repeat if necessary
call Leftshift ; final shift operation
MOVFF BCDhigh, LCDhighBCD ; move BCD output to input for LCD write algorithm
MOVFF BCDlow, LCDlowBCD
btfsc BCDoverflow ; check if the necessary digits exceeded four
call displayoverflow
return

displayoverflow ; sets the LCD to maximum value and flashes the display
MOVLF B'10011001', LCDhighBCD
MOVLF B'10011001', LCDlowBCD
return

Leftshift ; 32-bit left shift algorithm
bcf STATUS, C
rlf Binarylow, 1
rlf Binaryhigh, 1
rlf BCDlow, 1
rlf BCDhigh, 1
btfsc STATUS, C
bsf BCDoverflow
return

Add3test
MOVFF BCDlow, mathtemp ; tenths digit add 3 test
movf mathtemp, W ; binary arithmetic assumes integers,
therefore output has been multiplied by ten

andlw B'00001111'
movwf mathtemp
movlw B'00000101'
subwf mathtemp,W
btfsc STATUS,C
call BCDlowlowadd3

MOVFF BCDlow,mathtemp ;ones digit add 3 test
swapf mathtemp,F
movf mathtemp,W
andlw B'00001111'
movwf mathtemp
movlw B'00000101'
subwf mathtemp,W
btfsc STATUS,C
call BCDlowhighadd3

MOVFF BCDhigh,mathtemp ;tens digit add 3 test
movf mathtemp,W
andlw B'00001111'
movwf mathtemp
movlw B'00000101'
subwf mathtemp,W
btfsc STATUS,C
call BCDhighlowadd3

MOVFF BCDhigh,mathtemp ;hundreds digit add 3 test
swapf mathtemp,F
movf mathtemp,W
andlw B'00001111'
movwf mathtemp
movlw B'00000101'
subwf mathtemp,W
btfsc STATUS,C
call BCDhighhighadd3

return

BCDlowlowadd3 ;add 3 to tenths digit BCD location
movlw B'000000111'
advf mathtemp,F
bcf BCDlow,0
bcf BCDlow,1
bcf BCDlow,2
bcf BCDlow,3
btfsc mathtemp,0
bsf BCDlow,0
btfsc mathtemp,1
bsf BCDlow,1
btfsc mathtemp,2
bsf BCDlow,2
btfsc mathtemp,3
bsf BCDlow,3
return
BCDlowhighadd3

;add 3 to ones digit BCD

location

movlw B'00000011'
addwf mathtemp,F
bcf BCDlow,4
bcf BCDlow,5
bcf BCDlow,6
bcf BCDlow,7
btfs mathtemp,0
bsf BCDlow,4
btfs mathtemp,1
bsf BCDlow,5
btfs mathtemp,2
bsf BCDlow,6
btfs mathtemp,3
bsf BCDlow,7
return

BCDhighlowadd3

;add 3 to tens digit BCD

location

movlw B'00000011'
addwf mathtemp,F
bcf BCDhigh,0
bcf BCDhigh,1
bcf BCDhigh,2
bcf BCDhigh,3
btfs mathtemp,0
bsf BCDhigh,0
btfs mathtemp,1
bsf BCDhigh,1
btfs mathtemp,2
bsf BCDhigh,2
btfs mathtemp,3
bsf BCDhigh,3
return

BCDhighhighadd3

;add 3 to hundreds digit

location

movlw B'00000011'
addwf mathtemp,F
bcf BCDhigh,4
bcf BCDhigh,5
bcf BCDhigh,6
bcf BCDhigh,7
btfs mathtemp,0
bsf BCDhigh,4
btfs mathtemp,1
bsf BCDhigh,5
btfs mathtemp,2
bsf BCDhigh,6
btfs mathtemp,3
bsf BCDhigh,7
return
Write7seg  ;sets LCD output to the four numbers given by
LCDlowBCD and LCDhighBCD
  bcf    LAT000x  ;lower four bits of LCDlowBCD are sent to
  bcf    LAT00x0  ;upper four bits of LCDlowBCD are sent to
  bcf    LAT0x00  ;lower four bits of LCDhighBCD are sent to
  bcf    LATx000  ;upper four bits of LCDhighBCD are sent to
  MOVFF LCDlowBCD,BCDOUT
  call    BCDBUSSET
  bcf    LAT000x
  bcf    LAT00x0
  MOVFF LCDlowBCD,BCDOUT
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  call    BCDBUSSET
  bcf    LAT00x0
  bcf    LAT0x00
  MOVFF LCDhighBCD,BCDOUT
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  rrf      BCDOUT,1
  call    BCDBUSSET
  bcf    LATx000
  bcf    LATx000
  return
BCDBUSSET  ;sets the BCD data bus to the value of the lower four
bits of BCDOUT
  bcf    BCDA
  bcf    BCDB
  bcf    BCDC
  bcf    BCDD
  btfs    BCDOUT,0
  bcf    BCDA
  btfs    BCDOUT,1
  bcf    BCDB
  btfs    BCDOUT,2
  bcf    BCDC
  btfs    BCDOUT,3
  bcf    BCDD
  return
Looptime  ;keep looping until timer 2 overflows
  btfs  PIR1,TMR2IF
  goto    Looptime
btssel BCDoverflow  
call flash  
bcf PIR1,TMR2IF  
return  

flash ; flashes the LCD display, however when flashing is turned off 
the BCDoverflow flag must be cleared along with the blanking output to reenable the display  
nop  
nop  
btfss flashcycle  
bsf Blanking  
btfssel flashcycle  
bcf Blanking  
movlw B'00000010'  
xorwf Programflags,F  
return  

serialsend  
bsf RCSTA,SPEN  
bsf wirelesspwd  
bcf wirelessRXenable  
bsf wirelessTXenable ; must add timer  

for switching times  
; MOVLF B'10101100',serialsendbyte  
MOVLF 7,bitaddloop  
MOVLF 0,paritycalc  

MOVFF serialsendbyte,serialsendtemp  
movf serialsendtemp,W  
xorwf paritycalc,F  

bitaddcycle  
bcf STATUS,C  
rrf serialsendtemp,F  
movf serialsendtemp,W  
xorwf paritycalc,F  
decfsz bitaddloop,F  
goto bitaddcycle  

btssel paritycalc,0  
call setTX9 ; txsta bit 0  
btfss paritycalc,0  
call clearTX9  

MOVFF serialsendbyte,TXREG  

bsf STATUS,RP0  

TXwait  
btssel TXSTA,TRMT  
goto TXwait  
bcf STATUS,RP0  
nop  
nop  
bcf wirelessTXenable  
bsf wirelessRXenable  
btfssel maintainwirelessenable  
call disablewirelesstransmit
return
disablewirelesstransmit
  bcf  wirelesspwd
  bcf  RCSTA,SPEN
return

setTX9
  bsf  STATUS,RP0
  bsf  TXSTA,0
  bcf  STATUS,RP0
return
clearTX9
  bsf  STATUS,RP0
  bcf  TXSTA,0
  bcf  STATUS,RP0
return

enableserialrecieve
  bsf  RCSTA,SPEN
  bsf  wirelesspwd
  bsf  wirelessRXenable
  bsf  maintainwirelessenable
return
disableserialrecieve
  bcf  RCSTA,SPEN
  bcf  wirelessRXenable
  bcf  wirelesspwd
  bcf  maintainwirelessenable
return

enableLCDflash
  nop
  bsf  STATUS,RP0
  bsf  PIE1,TMR1IE
  bcf  STATUS,RP0
  MOVLF B'00110001',T1CON
  MOVLF B'11100000',TMR1H
  MOVLF 0,TMR1L
  nop
return
disableLCDflash
  bsf  STATUS,RP0
  bcf  PIE1,TMR1IE
  bcf  STATUS,RP0
  bcf  T1CON,TMR1ON
  bcf  Blanking
return
IntService
    movwf  W_temp
    swapf  STATUS,W
    clrf STATUS
    movwf STATUS_temp
    MOVFF PCLATH,PCLATH_temp

Poll
    btfs PIR1,RCIF
    goto serialrecieveswitchpage
    btfs PIR1, TMR1IF
    goto flashLCD
    MOVFF PCLATH_temp,PCLATH
    swapf  STATUS_temp,W
    movwf  STATUS
    swapf  W_temp,F
    swapf  W_temp,W
    retfie

flashLCD
    nop
    nop
    btfs flashcycle
    goto flashLCDon
    goto flashLCDoff

flashLCDon
    MOVLF B'11000000',TMR1H
    MOVLF 0,TMR1L
    bcf  Blanking
    bsf  flashcycle
    bcf  PIR1, TMR1IF
    goto Poll

flashLCDoff
    MOVLF B'11100000',TMR1H
    MOVLF 0,TMR1L
    bsf  Blanking
    bcf  flashcycle
    bcf  PIR1, TMR1IF
    goto Poll

serialrecieveswitchpage
    bsf  PCLATH,3
    goto serialrecieve

serialrecieve
    bcf  PIR1,RCIF
    bcf  receiveparity,0
    btfs RCSTA,RX9D
    bsf  receiveparity,0
MOVFF RCREG, serialrecievetemp
btfs serialrecievetemp, 7
call stopprog

MOVLF 7, bitaddloprecieve
MOVLF 0, paritytest

MOVFF serialrecievetemp, serialrecievetempcalc
movf serialrecievetempcalc, W
xorwf paritytest, F

receivebitaddcycle
bcf STATUS, C
rrf serialrecievetempcalc, F
movf serialrecievetempcalc, W
xorwf paritytest, F
decfsz bitaddloprecieve, F
goto receivebitaddcycle

movf paritytest, W
xorwf receiveparity, F
btfs receiveparity, 0
goto validrecieve
goto invalidrecieve

returnfromserial
bcf PCLATH, 3
goto Poll

invalidrecieve
bcf RCSTA, CREN
MOVFF RCREG, serialrecievetemp
MOVFF RCREG, serialrecievetemp
bsf RCSTA, CREN
goto returnfromserial

validrecieve
MOVFF serialrecievetemp, serialrecievebyte
; bsf validdaterecieved
nop
nop

bcf RCSTA, CREN
call decodeserialdata
MOVFF RCREG, serialrecievetemp
MOVFF RCREG, serialrecievetemp
bsf RCSTA, CREN

goto returnfromserial

stopprog
nop
nop
return
decode serial data
  btfs serial receive byte, 7
  goto rail1 serial data decode
  goto rail2 serial data decode

rail1 serial data decode
  btfs serial receive byte, 6
  goto rail1 serial byte 1 or 2
  goto rail1 serial byte 3 or 4

rail1 serial byte 1 or 2
  btfs serial receive byte, 5
  goto rail1 serial byte 1
  goto rail1 serial byte 2

rail1 serial byte 3 or 4
  btfs serial receive byte, 5
  goto rail1 serial byte 3
  goto rail1 serial byte 4

rail1 serial byte 1
  MOVFF iterR1AN1L, AARGBO
  MOVLF 8, BARGBO
  CALL FXM0808U
  movlw H'A0'
  addlw 0
  addwf AARGB1, W
  movwf FSR
  incf iterR1AN1L, F
  movlw 9
  subwf iterR1AN1L, W
  btfsc STATUS, Z
  call reset iter R1AN1L

  btfs INDF, 0
  decf onecount R1AN1B10, F
  swapf onecount R1AN1B10, F
  btfs INDF, 1
  decf onecount R1AN1B10, F
  swapf onecount R1AN1B10, F
  btfs INDF, 2
  decf onecount R1AN1B10, F
  swapf onecount R1AN1B10, F
  btfs INDF, 3
  decf onecount R1AN1B10, F
  swapf onecount R1AN1B10, F
  btfs INDF, 4
  decf onecount R1AN1B54, F

  MOVFF serial receive byte, INDF
  btfsc INDF, 0
  incf onecount R1AN1B10, F
  swapf onecount R1AN1B10, F

; point to oldest value out of ten
; subtract the oldest byte from the running ones count
; move the new data value into the list
; add the new byte to the running ones count
btfsc   INDF,1
incf   onecountR1AN1B10,F
swapf  onecountR1AN1B10,F
btfsc   INDF,2
incf   onecountR1AN1B32,F
swapf  onecountR1AN1B32,F
btfsc   INDF,3
incf   onecountR1AN1B32,F
swapf  onecountR1AN1B32,F
btfsc   INDF,4
incf   onecountR1AN1B54,F

nop
nop

return

resetiterR1AN1L
    MOVLF 0,iterR1AN1L
    return

rail1serialbyte2
    MOVFF iterR1AN1H,AARGB0 ;point to oldest value out of ten
    MOVLF 8,BARGB0
    CALL    FXM0808U
    movlw H'A0'
    addlw 1
    addwf AARGB1,W
    movwf FSR
    incf   iterR1AN1H,F
    movlw 9
    subwf iterR1AN1H,W
    btfsc   STATUS,Z
    call    resetiterR1AN1H
    swapf  onecountR1AN1B54,F ;subtract the oldest byte from the running ones count
    btfsc   INDF,0
    decf   onecountR1AN1B54,F
    swapf  onecountR1AN1B54,F
    btfsc   INDF,1
    decf   onecountR1AN1B76,F
    swapf  onecountR1AN1B76,F
    btfsc   INDF,2
    decf   onecountR1AN1B76,F
    swapf  onecountR1AN1B76,F
    btfsc   INDF,3
    decf   onecountR1AN1B98,F
    swapf  onecountR1AN1B98,F
    btfsc   INDF,4
    decf   onecountR1AN1B98,F
    swapf  onecountR1AN1B98,F

    MOVFF serialreceivebyte,INDF ;move the new data value into the list
    swapf  onecountR1AN1B54,F ;add the new byte to the running ones count
btfsc INDF,0
inef onecountR1AN1B54,F
swapf onecountR1AN1B54,F
btfsc INDF,1
inef onecountR1AN1B76,F
swapf onecountR1AN1B76,F
btfsc INDF,2
inef onecountR1AN1B76,F
swapf onecountR1AN1B76,F
btfsc INDF,3
inef onecountR1AN1B98,F
swapf onecountR1AN1B98,F
btfsc INDF,4
inef onecountR1AN1B98,F
swapf onecountR1AN1B98,F

return

resetiterR1AN1H
MOVLF0,iterR1AN1H
return

rail1serialbyte3
MOVFF iterR1AN2L,AARGB0 ;point to oldest value out of ten
MOVLF8,BARGB0
CALL FXM0808U
movlw H'0'
addlw 2
addwf AARGB1,W
movwf FSR
inef iterR1AN2L,F
movlw 9
subwf iterR1AN2L,W
btfsc STATUS,Z
call resetiterR1AN2L

btfsc INDF,0 ;subtract the oldest byte from the running ones count
decf onecountR1AN2B10,F
swapf onecountR1AN2B10,F
btfsc INDF,1
decf onecountR1AN2B10,F
swapf onecountR1AN2B10,F
btfsc INDF,2
decf onecountR1AN2B10,F
swapf onecountR1AN2B10,F
btfsc INDF,3
decf onecountR1AN2B32,F
swapf onecountR1AN2B32,F
btfsc INDF,4
decf onecountR1AN2B32,F

MOVFF serialrecievebyte,INDF ;move the new data value into the list
btfsc INDF,0 ;add the new byte to the running ones count
inef onecountR1AN2B10,F
swapf onecountR1AN2B10,F
btfs INDF,1
incf onecountR1AN2B10,F
swapf onecountR1AN2B10,F
btfs INDF,2
incf onecountR1AN2B32,F
swapf onecountR1AN2B32,F
btfs INDF,3
incf onecountR1AN2B32,F
swapf onecountR1AN2B32,F
btfs INDF,4
incf onecountR1AN2B54,F

return

resetiterR1AN2L
    MOVLF 0,iterR1AN2L
    return

rail1serialbyte4
    MOVFF iterR1AN2H,AARGB0 ;point to oldest value out of ten
    MOVLF 8,BARGB0
    CALL FXM0808U
    movlw H'A0'
    addlw 3
    addwf AARGB1,W
    movwf FSR
    incf iterR1AN2H,F
    movlw 9
    subwf iterR1AN2H,W
    btfsc STATUS,Z
    call resetiterR1AN2H
    swapf onecountR1AN2B54,F ;subtract the oldest byte from the running ones count
    btfs INDF,0
decf onecountR1AN2B54,F
    swapf onecountR1AN2B54,F
    btfs INDF,1
decf onecountR1AN2B76,F
    swapf onecountR1AN2B76,F
    btfs INDF,2
decf onecountR1AN2B76,F
    swapf onecountR1AN2B76,F
    btfs INDF,3
decf onecountR1AN2B98,F
    swapf onecountR1AN2B98,F
    btfs INDF,4
decf onecountR1AN2B98,F
    swapf onecountR1AN2B98,F

    MOVFF serialrecievebyte,INDF ;move the new data value into the list
    swapf onecountR1AN2B54,F
    btfs INDF,0 ;add the new byte to the running ones count
incf onecountR1AN2B54,F
swapf onecountR1AN2B54,F
btfsc INDF,1
incf onecountR1AN2B76,F
swapf onecountR1AN2B76,F
btfsc INDF,2
incf onecountR1AN2B76,F
swapf onecountR1AN2B76,F
btfsc INDF,3
incf onecountR1AN2B98,F
swapf onecountR1AN2B98,F
btfsc INDF,4
incf onecountR1AN2B98,F
return

resetiterR1AN2H
MOVLF 0,iterR1AN2H
return

rail2serialdatadecode;
  btfss serialrecievebyte,6
goto rail2serialbyte1or2
goto rail2serialbyte3or4

rail2serialbyte1or2
  btfss serialrecievebyte,5
goto rail2serialbyte1
goto rail2serialbyte2

rail2serialbyte3or4
  btfss serialrecievebyte,5
goto rail2serialbyte3
goto rail2serialbyte4

rail2serialbyte1
  MOVFF iterR2AN1L,AARGB0 ;point to oldest value out of ten
  MOVLF 8,BARGB0
  CALL FXM0808U
  movlw H'A0'
  addlw 4
  addwf AARGB1,W
  movwf FSR
  incf iterR2AN1L,F
  movlw 9
  subwf iterR2AN1L,W
  btfsc STATUS,Z
  call resetiterR2AN1L

  btfsc INDF,0
  decf onecountR2AN1B10,F
  swapf onecountR2AN1B10,F
  btfsc INDF,1
  decf onecountR2AN1B10,F
swapf onecountR2AN1B10,F  
btfsc INDF,2  
decf onecountR2AN1B32,F  
swapf onecountR2AN1B32,F  
btfsc INDF,3  
decf onecountR2AN1B32,F  
swapf onecountR2AN1B32,F  
btfsc INDF,4  
decf onecountR2AN1B54,F  

MOVFF serialrecievebyte,INDF  ;move the new data value into the list

btfsc INDF,0  ;add the new byte to the running ones count

incf onecountR2AN1B10,F
swapf onecountR2AN1B10,F
btfscl INDF,1
incf onecountR2AN1B10,F
swapf onecountR2AN1B10,F
btfscl INDF,2
incf onecountR2AN1B32,F
swapf onecountR2AN1B32,F
btfscl INDF,3
incf onecountR2AN1B32,F
swapf onecountR2AN1B32,F
btfscl INDF,4
incf onecountR2AN1B54,F

return

resetiterR2AN1L
  MOVLF 0,iterR2AN1L
  return

rail2serialbyte2
  MOVFF iterR2AN1H,AARGB0  ;point to oldest value out of ten
  MOVLF 8,BARGB0
  CALL FXM0808U
  movlw H'A0'
  addlw 5
  addwf AARGB1,W
  movwf FSR
  incf iterR2AN1H,F
  movlw 9
  subwf iterR2AN1H,W
  btfsc STATUS,Z
  call resetiterR2AN1H
  swapf onecountR2AN1B54,F  ;subtract the oldest byte from the running ones count
  btfscl INDF,0
  decf onecountR2AN1B54,F
  swapf onecountR2AN1B54,F
  btfscl INDF,1
  decf onecountR2AN1B76,F
  swapf onecountR2AN1B76,F
  btfscl INDF,2
decf onecountR2AN1B76,F  ;move the new data value into the list
swapf onecountR2AN1B76,F
btfsc INDF,3
decf onecountR2AN1B98,F
swapf onecountR2AN1B98,F
btfsc INDF,4
decf onecountR2AN1B98,F
swapf onecountR2AN1B98,F

MOVFF serialrecivebyte,INDF  ;add the new byte to the running ones count
swapf onecountR2AN1B54,F  
btsc INDF,0
incf onecountR2AN1B54,F
bstsc INDF,1
incf onecountR2AN1B54,F
bstsc INDF,2
incf onecountR2AN1B54,F
bstsc INDF,3
incf onecountR2AN1B54,F
bstsc INDF,4
incf onecountR2AN1B54,F
return

resetiterR2AN1H
MOVLF 0,iterR2AN1H
return

rail2serialbyte3
MOVFF iterR2AN2L,AARGB0  ;point to oldest value out of ten
MOVLF 8,BARGB0
CALL FXM0808U
movlw H'A0'
addlw 6
addwf AARGB1,W
movwf FSR
incf iterR2AN2L,F
movlw 9
subwf iterR2AN2L,W
btfsc STATUS,Z
    call resetiterR2AN2L
btsc INDF,0  ;subtract the oldest byte from the running ones count
decf onecountR2AN2B10,F
swapf onecountR2AN2B10,F
btsc INDF,1
decf onecountR2AN2B10,F
swapf onecountR2AN2B10,F
btsc INDF,2
decf onecountR2AN2B32,F
swapf onecountR2AN2B32,F
btsc INDF,3
decf onecountR2AN2B32,F
swapf onecountR2AN2B32,F
btsc INDF,4
decf onecountR2AN2B54,F

MOVFF serialrecievebyte,INDF ; move the new data value into the list

btsc INDF,0 ; add the new byte to the running ones count
incf onecountR2AN2B10,F
swapf onecountR2AN2B10,F
btsc INDF,1
incf onecountR2AN2B10,F
swapf onecountR2AN2B10,F
btsc INDF,2
incf onecountR2AN2B32,F
swapf onecountR2AN2B32,F
btsc INDF,3
incf onecountR2AN2B32,F
swapf onecountR2AN2B32,F
btsc INDF,4
incf onecountR2AN2B32,F

resetiterR2AN2L
MOVLF 0,iterR2AN2L
return

rail2serialbyte4
MOVFF iterR2AN2H,AARGB0 ; point to oldest value out of ten
MOVLF 8,BARGB0
CALL FXM0808U
movlw H'A0'
addlw 7
addwf AARGB1,W
movwf FSR
incf iterR2AN2H,F
movlw 9
subwf iterR2AN2H,W
btfsc STATUS,Z
call resetiterR2AN2H

swapf onecountR2AN2B54,F ; subtract the oldest byte from the running ones count
btsc INDF,0
decf onecountR2AN2B54,F
swapf onecountR2AN2B54,F
btsc INDF,1
decf onecountR2AN2B76,F
swapf onecountR2AN2B76,F
btsc INDF,2
decf onecountR2AN2B76,F
swapf onecountR2AN2B76,F
btsc  INDF,3
decf  onecountR2AN2B98,F
swapf onecountR2AN2B98,F
btsc  INDF,4
decf  onecountR2AN2B98,F
swapf onecountR2AN2B98,F

MOVFF serialreceivebyte,INDF ; move the new data value into the list

swapf onecountR2AN2B54,F ; add the new byte to the running ones count
btsc  INDF,0
incf  onecountR2AN2B54,F
swapf onecountR2AN2B54,F
btsc  INDF,1
incf  onecountR2AN2B76,F
swapf onecountR2AN2B76,F
btsc  INDF,2
incf  onecountR2AN2B76,F
swapf onecountR2AN2B76,F
btsc  INDF,3
incf  onecountR2AN2B98,F
swapf onecountR2AN2B98,F
btsc  INDF,4
incf  onecountR2AN2B98,F
swapf onecountR2AN2B98,F
return

resetiterR2AN2H
MOVLF 0,iterR2AN2H
return

************************************************************************************
; 8x8 Bit Unsigned Fixed Point Multiply 8x8 -> 16
; Input: 8 bit unsigned fixed point multiplicand in AARGB0
; 8 bit unsigned fixed point multiplier in BARGB0
; Use: CALL FXM0808U
; Output: 16 bit unsigned fixed point product in AARGB0,AARGB1
; Result: AARG <-- AARG x BARG
; Max Timing: 1+70+2 = 73 clks
; Min Timing: 1+53 = 54 clks
; PM: 1+19+1 = 21              DM: 4
FXM0808U CLRFB ACCB1
UMUL0808L
RETLW 0x00

************************************************************************************
; 16x16 Bit Unsigned Fixed Point Multiply 16x16 -> 32
; Input: 16 bit unsigned fixed point multiplicand in AARGB0,AARGB1
; 16 bit unsigned fixed point multiplier in BARGB0,BARGB1
; Use: CALL FXM1616U
; Output: 32 bit unsigned fixed point product in AARGB0,AARGB1,AARGB2,AARGB3
; Result: AARG <-- AARG x BARG
; Max Timing: 6+248+2 = 256 clks
; Min Timing: 6+101 = 107 clks
; PM: 6+51+1 = 58   DM: 9
FXM1616U    CLRF    ACCB2
           CLRF    ACCB3
           MOVF    AARGB0,W
           MOVWF    TEMPB0
           MOVF    AARGB1,W
           MOVWF    TEMPB1
           UMUL1616L
           RETLW    0x00

;***********************************************************************************
; 32/16 Bit Unsigned Fixed Point Divide 32/16 -> 32.16
; Input:  32 bit unsigned fixed point dividend in AARGB0,AARGB1,AARGB2,AARGB3
; 16 bit unsigned fixed point divisor in BARGB0,BARGB1
; Use:    CALL    FXD3216U
; Output: 32 bit unsigned fixed point quotient in AARGB0,AARGB1,AARGB2,AARGB3
; 16 bit unsigned fixed point remainder in REMB0,REMB1
; Result: AARG, REM <- AARG / BARG
; Max Timing: 2+699+2 = 703 clks
; Max Timing: 2+663+2 = 667 clks
; PM: 2+240+1 = 243   DM: 9
FXD3216U    CLRF    REMB0
           CLRF    REMB1
           UDIV3216L
           RETLW    0x00

End

Appendix B: Rail Microcontroller Code

list  P=PIC16F737, F=INHX8M, C=160, N=77, ST=OFF, MM=OFF, R=DEC, X=OFF
#include P16F737.inc
#include math.inc
__CONFIG    _CONFIG1, _CP_OFF & _CCP2_RC1 & _DEBUG_OFF & _VBOR_2_0 &
            _BOREN_1 & _MCLR_OFF & _PWRTE_OFF & _WDT_OFF & _INTRC_IO
            _CONFIG    _CONFIG2, _BORSEN_1 & _IESO_OFF & _FCMEN_OFF

;Equates
Bank0 equ H'20'

;Variables
cblock Bank0
ADCHARGETIME
ADAN2low
ADAN2high

114
ADAN1low
ADAN1high
ADAN2lowhold
ADAN2highhold
ADAN1lowhold
ADAN1highhold

ADAN2lowdatasend
ADAN2highdatasend
ADAN1lowdatasend
ADAN1highdatasend

serialsendbyte
serialsendtemp
paritycalc
bitaddloop
bitaddlooprecieve
serialrecievebyte
serialrecievetemp
serialrecievetempcalc
paritytest
W_temp
STATUS_temp
recieveparity
Programflags
timer1high
timer1low
countdown
progstate
Platformprogflags
mathtemph
mathtempl
mathtemph2
mathtempl2
ADAN1total0
ADAN1total1
ADAN1total2
ADAN1iter0
ADAN1iter1
ADAN2total0
ADAN2total1
ADAN2total2
ADAN2iter0
ADAN2iter1
ADAN1ave0
ADAN1ave1
ADAN2ave0
ADAN2ave1
remaindertemphigh
remaindertemplow
ADAN1lowerlock
ADAN1upperlock
ADAN2lowerlock
ADAN2upperlock
repeatsenditerator
LEDADAN1L
LEDADAN1H
LEDADAN2L
LEDADAN2H

delaystartiter
endc

;Defines
#define BCDA PORTC,4          ;LCD port definitions, alternate electrical
  wiring is accommodated by changing these values only
#define rdAN0 B'00000001',ADCON0
#define rdAN1 B'000001001',ADCON0
#define rdAN2 B'000100001',ADCON0
#define startAD ADCON0,2
#define offAD ADCON0,0
#define serialTX9 TXSTA,0
#define wirelesspwd PORTC,0    ;logic low powers down transceiver
#define wirelessRXenable PORTC,1
#define wirelessTXenable PORTC,2
#define maintainwirelesssenable Programflags,5
#define validdatarecieved Programflags,6
#define withinrange Platformprogflags,0

;Macros
MOVLF macro literal,dest
  movlw literal
  movwf dest
endm

MOVFF macro source,dest
  movf source,W
  movwf dest
endm

BANK1 macro
  bcf STATUS,RP1
  bsf STATUS,RP0
endm

BANK0 macro
  bcf STATUS,RP1
  bcf STATUS,RP1
endm

ADD2416 macro AARG0,AARG1,AARG2,BARG0,BARG1 ;AARG0 and BARG0 are the
  lowest bytes
  movf BARG0,W
  addwf AARG0,F
  btfsc STATUS,C
  incf BARG1,F
  movf BARG1,W
  addwf AARG1,F
  btfsc STATUS,C
  incf AARG2,F
endm
INC16 macro lowbyte, highbyte
  inef lowbyte,F
  btfs STATUS,Z
  inef highbyte,F
endm

SUB16 macro AARGBH,AARGBL,BARGBH,BARGBL,CARGBH,CARGBL
  movf BARGBL,0
  subwf AARGBL,0
  movwf CARGBL
  movf BARGBH,0
  btfs STATUS,C
  incfsz BARGBH,0
  subwf AARGBH,0
  movwf CARGBH
endm

;Vectors
org H'000'
  nop
  goto Mainprog
org H'004' ;Interrupt vector
  goto IntService

Mainprog
  call Initialize
  ; call enableserialrecieve
  ; call starttimer1

Mainloop
  movf progstate,w ;this was moved here because the execution of Average is time
      sensitive
  sublw 3
  btfs STATUS,Z
  goto Average
  movf progstate,w
  sublw 0
  btfs STATUS,Z
  goto Standby
  movf progstate,w
  sublw 1
  btfs STATUS,Z
  goto Delaystart
  movf progstate,w
  sublw 2
  btfs STATUS,Z
goto StartAverage

movf progstate, w
sublw 4
btfsc STATUS, Z
goto Calc

movf progstate, w
sublw 5
btfsc STATUS, Z
goto Sendloadcell

movf progstate, w
sublw 6
btfsc STATUS, Z
goto Repeatserialsend

; call serialsend
; nop
; nop
; nop
; btsc validdata received
; bcf validdata received
; call tenmscount
; call ADconvertAN0
; nop
; nop
; call ADconvertAN1
; nop
; nop
; goto Mainloop

; tenmscount
; MOVLF 0, countdown1
; tenmswait
; decsz countdown1, f
; goto tenmswait
; btss PORTA, 2
; goto sethigh
; bcf PORTA, 2
; rettimer1
; return
; sethigh
; bsf PORTA, 2
; goto rettimer1

Initialize

; bcf STATUS, RP1 ; switch to Bank1
bsf STATUS, RP0
MOVLF B'11000010', OSCCON ; select 1 MHz internal RC oscillator frequency and select
internal oscillator
MOVLF 0, OSCTUNE ; wait until oscillator stabilizes
BTFSS OSCCON, 2
goto oscwait
MOVLF B'10001100', ADCON1 ; enable one a/d converter on AN0 and remainder I/O
MOVLF B'00010111', TRISA
MOVLF B'11000000', TRISB
MOVLF B'10001000', TRISC
MOVLF B'00100000', PIE1
bcf STATUS, RP0
MOVLF B'01110000', RCSTA ; must enable serial port for send or receive
MOVLF 25, SPBRG
MOVLF B'11000000', INTCON
MOVLF 0, Programflags
MOVLF 0, ADAN2high
MOVLF 0, ADAN2low
MOVLF 0, ADAN1high
MOVLF 0, ADAN1low
MOVLF 0, ADAN2highhold
MOVLF 0, ADAN2lowhold
MOVLF 0, ADAN1highhold
MOVLF 0, ADAN1lowhold
MOVLF 0, timer1high
MOVLF 0, timer1low
MOVLF 0, progstate
nop
nop
bcf wirelessTXenable
bcf wirelessRXenable
bcf wirelesspwd
nop
nop

; bcf PORTA, 2

return

Standby
    call ADconvertAN1 ; must have added 17 to hold value
    call ADconvertAN2
    nop
    nop
    ; MOVLF 0, ADAN1highhold
    ; MOVLF 100, ADAN1lowhold
    ; MOVLF 0, ADAN1high
    ; MOVLF 66, ADAN1low
    SUB16 ADAN1highhold, ADAN1lowhold, ADAN1high, ADAN1low, mathtemph, mathtempl
    btfss STATUS, C
goto outofrange
    MOVLF 0, ADAN1high
MOVLF 35, ADAN1low
; 2*17+1 sets acceptable range for change in A/D due to noise, change for different noise rejection
    SUB16 mathtemph, mathtempl, ADAN1high, ADAN1low, mathtemph2, mathtempl2
btfrsc STATUS,C
goto outofrange

SUB16 ADAN2highhold,ADAN2lowhold,ADAN2high,ADAN2low,mathtemph,mathtempl
btfrsc STATUS,C
goto outofrange
MOVLF 0,ADAN2high
MOVLF 35,ADAN2low
/>.2^17+1 sets acceptable range for change in A/D due to noise, change for
different noise rejection
SUB16 mathtemph,mathtempl,ADAN2high,ADAN2low,mathtemph2,mathtempl2
btfrsc STATUS,C
goto outofrange
call loadcellLED

goto Mainloop

outofrange
nop
nop
MOVLF 0,delaystartiter
incf progstate,F
call starttimer1
goto Mainloop

loadcellLED
btfrsc PORTA,4
goto displayAN1
goto displayAN2

displayAN1
bcf PORTC,4
bcf PORTC,5
bcf PORTB,0
bcf PORTB,1
bcf PORTB,2
bcf PORTB,3
bcf PORTB,4
bcf PORTB,5
bcf PORTA,3
bcf PORTA,5

btfrsc LEDADAN1L,0
bsf PORTC,4
btfrsc LEDADAN1L,1
bsf PORTC,5
btfrsc LEDADAN1L,2
bsf PORTB,0
btfrsc LEDADAN1L,3
bsf PORTB,1
btfrsc LEDADAN1L,4
bsf PORTB,2
btfrsc LEDADAN1L,5
bsf PORTB,3


120
btfsc LEDADAN1L,6
bsf PORTB,4
btfsc LEDADAN1L,7
bsf PORTB,5
btfsc LEDADAN1H,0
bsf PORTA,3
btfsc LEDADAN1H,1
bsf PORTA,5

return

displayAN2
bcf PORTC,4
bcf PORTC,5
bcf PORTB,0
bcf PORTB,1
bcf PORTB,2
bcf PORTB,3
bcf PORTB,4
bcf PORTB,5
bcf PORTA,3
bcf PORTA,5

btfsc LEDADAN2L,0
bsf PORTC,4
btfsc LEDADAN2L,1
bsf PORTC,5
btfsc LEDADAN2L,2
bsf PORTB,0
btfsc LEDADAN2L,3
bsf PORTB,1
btfsc LEDADAN2L,4
bsf PORTB,2
btfsc LEDADAN2L,5
bsf PORTB,3
btfsc LEDADAN2L,6
bsf PORTB,4
btfsc LEDADAN2L,7
bsf PORTB,5
btfsc LEDADAN2H,0
bsf PORTA,3
btfsc LEDADAN2H,1
bsf PORTA,5

return

Delaystart
; incf progstate,F
; goto Mainloop
incf delaystartiter,F
btfsc PIR1,TMR1IF
goto proceed
goto Mainloop
proceed
  bcf    T1CON,TMR1ON
  bcf    PIR1,TMR1IF
  incf   progstate,F
  goto   Mainloop

StartAverage

  MOVLF 0,ADAN1total0
  MOVLF 0,ADAN1total1
  MOVLF 0,ADAN1total2
  MOVLF 0,ADAN1iter0
  MOVLF 0,ADAN1iter1
  MOVLF 0,ADAN2total0
  MOVLF 0,ADAN2total1
  MOVLF 0,ADAN2total2
  MOVLF 0,ADAN2iter0
  MOVLF 0,ADAN2iter1
  incf   progstate,F
  call   starttimer1
  goto   Mainloop

Average

;ADD2416  macro  AARG0,AARG1,AARG2,BARG0,BARG1 ;AARG0 and BARG0 are the
;INC16    macro  lowbyte, highbyte
    call   ADconvertAN1
    call   ADconvertAN2
    ADD2416  ADAN1total0,ADAN1total1,ADAN1total2,ADAN1low,ADAN1high
    INC16   ADAN1iter0,ADAN1iter1
    ADD2416  ADAN2total0,ADAN2total1,ADAN2total2,ADAN2low,ADAN2high
    INC16   ADAN2iter0,ADAN2iter1
    btfsc  PIR1,TMR1IF
    incf   progstate,F
    ; nop
    ; call   ADconvertAN1
    ; call   ADconvertAN2
    ; MOVFF ADAN1high,ADAN1highhold
    ; MOVFF ADAN1low,ADAN1lowhold
    ; movlw  17
    ; addwf  ADAN1lowhold,F
    ; btfsc  STATUS,C
    ; incf   ADAN1highhold,F
    ; MOVFF ADAN2high,ADAN2highhold
    ; MOVFF ADAN2low,ADAN2lowhold
    ; movlw  17
    ; addwf  ADAN2lowhold,F
    ; btfsc  STATUS,C
    ; incf   ADAN2highhold,F
    ; nop
    goto   Mainloop

Calc

  bcf    T1CON,TMR1ON
  bcf    PIR1,TMR1IF
MOVFF ADAN1total0,AARGB2 ;change to appropriate values
MOVFF ADAN1total1,AARGB1
MOVFF ADAN1total2,AARGB0
nop
nop
MOVFF ADAN1iter0,BARGB1
MOVFF ADAN1iter1,BARGB0
nop
nop
CALL FXD2315U
MOVFF AARGB2,ADAN1lowhold
MOVFF AARGB1,ADAN1highhold

MOVFF REMB1,remaindertemplow
MOVFF REMB0,remaindertemphigh
MOVFF ADAN1iter0,AARGB1
MOVFF ADAN1iter1,AARGB0
MOVLF 2,BARGB0
call FXD1608U

movf remaindertemplow,0
subwf AARGB1,0
movf remaindertemphigh,0
btfss STATUS,C
incfsz remaindertemphigh,0
subwf AARGB0,0
btfss STATUS,C
call roundupAN1

nop
nop
nop
;accomodate remainder rounding

MOVFF ADAN2total0,AARGB2 ;change to appropriate values
MOVFF ADAN2total1,AARGB1
MOVFF ADAN2total2,AARGB0
MOVFF ADAN2iter0,BARGB1
MOVFF ADAN2iter1,BARGB0
CALL FXD2315U
MOVFF AARGB2,ADAN2lowhold
MOVFF AARGB1,ADAN2highhold

MOVFF REMB1,remaindertemplow
MOVFF REMB0,remaindertemphigh
MOVFF ADAN2iter0,AARGB1
MOVFF ADAN2iter1,AARGB0
MOVLF 2,BARGB0
call FXD1608U
movf remaindertemplow,0
subwf AARGB1,0
movf remaindertemphigh,0
btfss STATUS,C
incfsz remaindertemphigh,0
subwf AARGB0,0
btfss STATUS,C
call roundupAN2

MOVFF ADAN1lowhold,ADAN1lowdatasend
MOVFF ADAN1highhold,ADAN1highdatasend
MOVFF ADAN2lowhold,ADAN2lowdatasend
MOVFF ADAN2highhold,ADAN2highdatasend
nop
nop
nop
MOVFF ADAN1lowhold,LEDADAN1L
MOVFF ADAN1highhold,LEDADAN1H
MOVFF ADAN2lowhold,LEDADAN2L
MOVFF ADAN2highhold,LEDADAN2H
nop
nop
nop
call loadcellLED
nop
nop
nop
incf progstate,F
goto Mainloop

roundupAN1
incf ADAN1lowhold,F
btfsc STATUS,C
incf ADAN1highhold,F
btfsc ADAN1highhold,2
call maxscaleAN1
return

maxscaleAN1
MOVLF B'00000111',ADAN1highhold
MOVLF B'11111111',ADAN1lowhold
return

roundupAN2
incf ADAN2lowhold,F
btfsc STATUS,C
incf ADAN2highhold,F
btfsc ADAN2highhold,2
call maxscaleAN2
return

maxscaleAN2
MOVLFB'00000111',ADAN2highhold
MOVLFB'11111111',ADAN2lowhold
return

Sendloadcell
bcf STATUS,C
btfs ADAN1lowdatasend,7
bsf STATUS,C
rlf ADAN1highdatasend,F
bcf STATUS,C
btfs ADAN1lowdatasend,6
bsf STATUS,C
rlf ADAN1highdatasend,F
bcf STATUS,C
btfs ADAN1lowdatasend,5
bsf STATUS,C
rlf ADAN1highdatasend,F
bsf ADAN1lowdatasend,7 ;change this bit for different rails
bcf ADAN1lowdatasend,6
bcf ADAN1lowdatasend,5
bsf ADAN1highdatasend,7
bcf ADAN1highdatasend,6
bsf ADAN1highdatasend,5
bcf STATUS,C
btfs ADAN2lowdatasend,7
bsf STATUS,C
rlf ADAN2highdatasend,F
bcf STATUS,C
btfs ADAN2lowdatasend,6
bsf STATUS,C
rlf ADAN2highdatasend,F
bcf STATUS,C
btfs ADAN2lowdatasend,5
bsf STATUS,C
rlf ADAN2highdatasend,F
bsf ADAN2lowdatasend,7
bsf ADAN2lowdatasend,6
bcf ADAN2lowdatasend,5
bsf ADAN2highdatasend,7
bsf ADAN2highdatasend,6
bsf ADAN2highdatasend,5
MOVFF ADAN1lowdatasend,ADAN1lowerlock
MOVFF ADAN1highdatasend,ADAN1upperlock
MOVFF ADAN2lowdatasend,ADAN2lowerlock
MOVFF ADAN2highdatasend,ADAN2upperlock
; nop
; nop
; nop
; call wirelesstransmitpowerup

MOVFF ADAN1lowerlock,serialsendbyte
; nop
; nop
; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call loadcellLED
; nop
; nop

;sendwaitset1
; btfss PORTC,3
; goto sendwaitset1
;sendwaitclear1
; btfsc PORTC,3
; goto sendwaitclear1

; nop
; call serialsend
; nop
; nop
; MOVFF ADAN1upperlock,serialsendbyte
; nop
; nop

; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call loadcellLED

;sendwaitset2
; btfss PORTC,3
; goto sendwaitset2
;sendwaitclear2
; btfsc PORTC,3
; goto sendwaitclear2

; nop
; call serialsend
; MOVFF ADAN2lowerlock,serialsendbyte
; nop
; nop

; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call loadcellLED
;sendwaitset3
;    btfss PORTC,3
;    goto sendwaitset3
;sendwaitclear3
;    btfsc PORTC,3
;    goto sendwaitclear3

;    nop
    call serialsend
    MOVFF ADAN2upperlock,serialsendbyte
;    nop
;    nop

;    MOVFF serialsendbyte,LEDADAN1L
;    MOVLF B'00000011',LEDADAN1H
;    call loadcellLED

;sendwaitset4
;    btfss PORTC,3
;    goto sendwaitset4
;sendwaitclear4
;    btfsc PORTC,3
;    goto sendwaitclear4

;    nop
    call serialsend

;    nop

;the calculated average
;    call ADconvertAN1
;    call ADconvertAN2
;    MOVFF ADAN1high,ADAN1highhold
;    MOVFF ADAN1low,ADAN1lowhold
movlw 17
addwf ADAN1lowhold,F
btfscc STATUS,C
inff ADAN1highhold,F
;    MOVFF ADAN2high,ADAN2highhold
;    MOVFF ADAN2low,ADAN2lowhold
movlw 17
addwf ADAN2lowhold,F
btfscc STATUS,C
inff ADAN2highhold,F
;    nop
inff progstate,F

goto Mainloop

;change the hold value to
Repeatserialsend
MOVLF 15,repeatsenditerator

iteratesend
MOVFF ADAN1lowerlock,serialsendbyte
; nop
; nop
; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call  loadcellLED

;sendwaitset5
; btfs PORTC,3
; goto  sendwaitset5
;sendwaitclear5
; btfsc PORTC,3
; goto  sendwaitclear5

; nop
call  serialsend
MOVFF ADAN1upperlock,serialsendbyte
; nop
; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call  loadcellLED

;sendwaitset6
; btfs PORTC,3
; goto  sendwaitset6
;sendwaitclear6
; btfsc PORTC,3
; goto  sendwaitclear6

; nop
; nop
call  serialsend
MOVFF ADAN2lowerlock,serialsendbyte
; nop
; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call  loadcellLED

;sendwaitset7
; btfs PORTC,3
; goto  sendwaitset7
;sendwaitclear7
; btfsc PORTC,3
; goto  sendwaitclear7

; nop
; nop
call  serialsend
MOVFF ADAN2upperlock,serialsendbyte
; nop
; MOVFF serialsendbyte,LEDADAN1L
; MOVLF B'00000011',LEDADAN1H
; call loadcellLED

;sendwaitset8
; btfss PORTC,3
; goto sendwaitset8
;sendwaitclear8
; btfsc PORTC,3
; goto sendwaitclear8

; nop
; nop
call serialsend
decfsz repeatsenditerator,F
goto iteratesend
; call wirelesstransmitpowerdown
MOVLF0,progstate
goto Mainloop

ADconvertAN2
MOVLFrdAN2
movlw d'15'
movwf ADCHARGETIME
waitSetAN2
decfsz ADCHARGETIME,F
goto waitSetAN2
bsf startAD
nop
ADwaitAN2
btfsc ADCON0,GO_DONE
goto ADwaitAN2

MOVFF ADRESH,ADAN2high
bsf STATUS,RP0
movf ADRESL,W
bcf STATUS,RP0
movwf ADAN2low

cbf offAD

return

ADconvertAN1
MOVLFrdAN1
movlw d'15'
movwf ADCHARGETIME
waitSetAN1
decfsz ADCHARGETIME,F
goto waitSetAN1
bsf startAD
nop
ADwaitAN1
    btfs ADCON0,GO_DONE
    goto ADwaitAN1

MOVFF ADRESH,ADAN1high
bsf STATUS,RP0
movf ADRESL,W
bcf STATUS,RP0
movwf ADAN1low

bcf offAD
return

serialsend
    bsf RCSTA,SPEN
    bsf wirelesspwd
    bcf wirelessRXenable
    bsf wirelessTXenable
    ;must add timer
for switching times
    MOVLF 0,countdown ;9 ms wait time
    TX3mswait1
        decfsz countdown,f
goto TX3mswait1
    MOVLF 0,countdown
    TX3mswait2
        decfsz countdown,f
goto TX3mswait2
    MOVLF 0,countdown
    TX3mswait3
        decfsz countdown,f
goto TX3mswait3
    MOVLF 7,bitaddloop
    MOVLF 0,paritycalc
    MOVFF serialsendbyte,serialsendtemp
    movf serialsendtemp,W
    xorwf paritycalc,F
    bitaddcycle
        bcf STATUS,C
        rrf serialsendtemp,F
        movf serialsendtemp,W
        xorwf paritycalc,F
        decfsz bitaddloop,F
goto bitaddcycle

    btfs paritycalc,0
    call setTX9 ;txsta bit 0
    btfs paritycalc,0
    call clearTX9

MOVFF serialsendbyte,TXREG
bsf STATUS,RP0
TXwait
btfss TXSTA,TRMT
goto TXwait
bcf STATUS,RP0

bcf wirelessTXenable
bsf wirelessRXenable
btfss maintainwirelessenable
call disablewirelesstransmit

return
disablewirelesstransmit
bcf wirelesspwd
bcf RCSTA,SPEN
return

;wirelesstransmitpowerup
; bsf RCSTA,SPEN
; bsf wirelesspwd
; bcf wirelessRXenable
; bsf wirelessTXenable
;
; MOVLF 0,countdown ;9 ms wait time
;TX3mswait1
; decfsz countdown,f
; goto TX3mswait1
; MOVLF 0,countdown
;TX3mswait2
; decfsz countdown,f
; goto TX3mswait2
; MOVLF 0,countdown
;TX3mswait3
; decfsz countdown,f
; goto TX3mswait3
;
; return

;wirelesstransmitpowerdown
; bcf wirelessTXenable
; bsf wirelessRXenable
; bcf wirelesspwd
; bcf RCSTA,SPEN
; return
setTX9
  bsf STATUS,RP0
  bsf TXSTA,0
  bcf STATUS,RP0
  return

clearTX9
  bsf STATUS,RP0
  bcf TXSTA,0
  bcf STATUS,RP0
  return

enableserialrecieve
  bsf RCSTA,SPEN
  bsf wirelesspwd
  bsf wirelessRXenable
  bsf maintainwirelessenable
  MOVLF 0,countdown ;9 ms wait time

RX3mswait1
  decfsz countdown,f
  goto RX3mswait1
  MOVLF 0,countdown

RX3mswait2
  decfsz countdown,f
  goto RX3mswait2
  MOVLF 0,countdown

RX3mswait3
  decfsz countdown,f
  goto RX3mswait3

return
disableserialrecieve
  bcf RCSTA,SPEN
  bcf wirelessRXenable
  bcf wirelesspwd
  bcf maintainwirelessenable
  return

IntService
  movwf W_temp
  swapf STATUS,W
  movwf STATUS_temp
  bcf STATUS,RP0

Poll
  btfs PIR1,RCIF
  goto serialrecieve
  btfs PIR1,TMR1IF
  goto timer1reset
  swapf STATUS_temp,W
  movwf STATUS
  swapf W_temp,F
  swapf W_temp,W
serialrecieve
  bcf PIR1,RCIF
  bcf receiveparity,0
  btfs RCSTA,RXD
  bsf receiveparity,0
  MOVFF RCREG,serialrecievetemp

  MOVLF 7,bitaddlooprecieve
  MOVLF 0,paritytest

  MOVFF serialrecievetemp,serialrecievetempcalc
  movf serialrecievetempcalc,W
  xorwf paritytest,F

recievebitaddcycle
  bcf STATUS,C
  rrf serialrecievetempcalc,F
  movf serialrecievetempcalc,W
  xorwf paritytest,F
  decfsz bitaddlooprecieve,F
  goto recievebitaddcycle

  movf paritytest,W
  xorwf receiveparity,F
  btfs receiveparity,0
  call invalidrecieve ;txsta bit 0
  btfs receiveparity,0
  call validrecieve
  goto Poll

invalidrecieve ;check, does anything need to go here
  return

validrecieve
  MOVFF serialrecievetemp,serialrecievebyte
  bsf validdatareceived
  return

starttimer1
  MOVL F B'00110001',T1CON
  MOVL 0,TMR1H
  MOVL 0,TMR1L
  ; bsf STATUS,RP0
  ; bsf PIE1,TMR1IE
  ; bcf STATUS,RP0
  return

timer1reset
  bcf PIR1,TMR1IF
  MOVL F 0,TMR1H
  MOVL F 0,TMR1L
  btfs PORTA,2
  goto sethigh
bcf PORTA,2
rettimer1
    goto Poll

sethigh
    bsf PORTA,2
    goto rettimer1

;************************************************************************************
;       23/15 Bit Unsigned Fixed Point Divide 23/15 -> 23.15
;       Input:  23 bit unsigned fixed point dividend in AARGB0,AARGB1,AARGB2
;               15 bit unsigned fixed point divisor in BARGB0,BARGB1
;       Use:    CALL    FXD2315U
;       Output: 23 bit unsigned fixed point quotient in AARGB0,AARGB1,AARGB2
;               15 bit unsigned fixed point remainder in REMB0,REMB1
;       Result: AARG, REM  <--  AARG / BARG
;       Max Timing:     2+403+2 = 407 clks
;       Min Timing:     2+375+2 = 379 clks
;       PM: 2+117+1 = 120               DM: 7
FXD2315U
    CLRF            REMB0
    CLRF            REMB1
    UDIV2315L
    RETLW           0x00

;************************************************************************************
;       16/8 Bit Unsigned Fixed Point Divide 16/8 -> 16.08
;       Input:  16 bit unsigned fixed point dividend in AARGB0,AARGB1
;               8 bit unsigned fixed point divisor in BARGB0
;       Use:    CALL    FXD1608U
;       Output: 16 bit unsigned fixed point quotient in AARGB0,AARGB1
;               8 bit unsigned fixed point remainder in REMB0
;       Result: AARG, REM  <--  AARG / BARG
;       Max Timing:     1+291+2 = 294 clks
;       Min Timing:     1+227+2 = 230 clks
;       PM: 1+39+1 = 41               DM: 7
FXD1608U
    CLRF          REMB0
    UDIV1608L
    RETLW          0x00

end