5-3-2010

Field Data Acquisition Unit

Benjamin Connelly
Mehmet Bacak
Sami Alhajjar

Follow this and additional works at: http://opus.ipfw.edu/etcs_seniorproj_engineering

Opus Citation
Benjamin Connelly, Mehmet Bacak, and Sami Alhajjar (2010). Field Data Acquisition Unit.
http://opus.ipfw.edu/etcs_seniorproj_engineering/37

This Senior Design Project is brought to you for free and open access by the School of Engineering, Technology and Computer Science Design Projects at Opus: Research & Creativity at IPFW. It has been accepted for inclusion in Engineering Senior Design Projects by an authorized administrator of Opus: Research & Creativity at IPFW. For more information, please contact admin@lib.ipfw.edu.
Indiana University-Purdue University Fort Wayne
Department of Engineering

ECE 406

Senior Design Project

*Report #2*

Project Title: Field Data Acquisition Unit

Team Members: Benjamin Connelly

Mehmet Bacak

Sami Alhajjar

Advisor: Dr. Hossein Oloomi

Date: May 3, 2010
# Table of Contents

Acknowledgement

Abstract 1

Section I: Detailed Description of the Conceptual Design 3

Section II: The Building Process 17

Section III: Testing: Procedures and Results 24

Section IV: Evaluations and Recommendations 35

Conclusions 38

References 39

Appendix
Acknowledgements

I would like to thank my family, especially my parents for their support throughout my college career and this project in particular. I would also like to thank my girlfriend Aimee for her support and understanding as I have put in many long days working on this project. I would like to thank Dr. Hossein Oloomi for his guidance and insight throughout this project. Thanks are also due to WaterFurnace International for the chance to work on this project and to Mark Adams, WaterFurnace Intl. research engineer for his support, guidance, and help in this whole project. I would also like to thank Bipom Electronics and their Chief Technology Officer, Vitaliy Avramenko, for their help on this project. Finally I would like to thank the IPFW engineering department for helping me to develop my abilities and guide me on my path to graduation. Thank you.

Benjamin A. Connelly

I would like to thank first my parents. They have been supportive of me during my education. I couldn’t be at the point that I am today without them. I wish my mother, who had passed away in summer of 2009, could see that I am working on a project like that. Even though she is not alive, I want to thank my mother for her support and guidance that she showed during my education life. I would like to thank to my fiancé Hande for her patience, understanding and support that she has shown during this busy designing period. I would like to thank Dr. Oloomi for his advice during this project. He was the one who was helping us to make the right choices and avoid the mistakes. Without his advice and help this important project would be very difficult for us. I would like to thank Mark Adams and WaterFurnace International. Mr. Adams had provided us many information and opportunity during the project. He couldn’t be more understanding and nice to us. WaterFurnace International has given us the freedom on design and idea. Also, WaterFurnace had provided us numerous opportunities that we can use on the project. I would like to thank my best friend, Gurbuz for finding and letting me know about IPFW, without him I can’t be living this education life and experience. I want to thank so many times to my group members, Benjamin A. Connelly and Sami Alhajjar. I believe, they are the best team members that I can ever meet. Thanks to them for turning this project into enjoyable and easier than it is.

Thank you.

Mehmet Bacak

I would like to thank my family for the support they have given to me throughout my college education. They were the ones who gave me the right advice and the motivation to make sure I completed my college degree. Also I would like to thank Dr. H. Oloomi for his advice and help throughout this design phase. He was one of our first options to ask for help, without him we would never have reached this point on our design. Furthermore I would like to thank the rest of my group, Ben Connelly and Mehmet Bacak for the help they provided and making this project as enjoyable as possible. I also would like to thank WaterFurnace international and Mark Adams for giving us the opportunity to work on their project. Without the help of Mr. Adams this project will be far from finishing so thank you all for the help. I would also to thank Bipom Electronics for all of their help.

Sami Alhajjar
Abstract

This paper is written as a continuation of the fall 2009 semester of work on the field data acquisition unit senior design project performed in conjunction with WaterFurnace International. The project is the result of work done by senior design students in the Indiana University-Purdue University School of Engineering, Technology, and Computer Science. This document is a written record of the process of building a final prototype based on the problem statement and chosen conceptual design of the previous semester. The proposal of the previous semester was to build a web-based data acquisition unit to monitor several temperature and pressure sensors, record the data, and publish the data to a website.

This report reviews the selected conceptual design of the previous semester, a brief description of the building process and any changes made to the conceptual design, testing results, the groups evaluations and recommendations, and a final conclusion to this project. Information will be provided for all of our processes, our reasoning, and final conclusions. All schematics, research, and calculations are provided for reference and understanding.
Section I:

Detailed Description of the Conceptual Design
The schematic on the next page shows how the peripheral board is going to be interfaced with the WebCatPlus internet board. The peripheral board is going to include two 16-to-1 multiplexers (74150) that will be the connection points for the 12 temperature and 8 pressure sensors. The sensors are on the left and they will be connected to multiplexers’ input ports. We will be using Danfoss model AKS 21A temperature sensors and Danfoss model NSK-BE050I-U009 pressure sensors. The multiplexer addressing will be done by the microcontroller of WebCatPlus internet board (LPC2138). The addressing port of multiplexers will be connected to expansion port of internet board. The expansion board is connected to one of input/output ports of LPC2138. Addressing will be done by microcontroller program. The analog signal that is passed through from the multiplexer to the microcontroller is going to go to an analog terminal of the WebCatPlus board. The WebCatPlus board will accept this analog input and convert it to digital information for storage and transmission. We will use three analog input/output pins to accept the analog signals from the sensors and fault signals. The fault signal will be used to instruct the microcontroller to begin reading the sensor inputs and there will be two analog input/outputs dedicated to sensor readings. One input/output for each bank of sensors. The LPC2138 is going to control everything including the data collection and conversion and the web page updating. The analog terminal is connected to microcontroller analog-to-digital converter (ADC) unit. The rest of the pins of microcontroller are used for other features. The converted digital data will be sent to memory and the internet microcontroller (ENC28J60) via serial communication port. Also, while receiving data from website the same port will be used, through the ENC28J60. All this converting, storing, sending and receiving process will be done internally on WebCatPlus MMC-1 board shown in the schematics in the appendix and in Figure 12 and Figure 14.
Figure 1: Conceptual design schematic for field data acquisition unit.
Figure 14, above, shows the components of the WebCatPlus MMC-1 internet board and its’ interfacing with external devices. First of all the board has its own power supply which is providing 3.3, 5, and 3 volts output voltage to supply the components. The reason of three different level voltage output is the internet microcontroller’s (ENC28J60) supply voltage is 3.3 Volt while main microcontroller’s (LCP2138) is 5 Volt. The RS232 is a tool for debugging computer connection with peripheral devices using a COM port. The RS232 can send and receive data through the COM port. The RS232 micro controller is used to communicate the Internet board with a pc. By using that microcontroller the board is became able to communicate with a local computer via COM port. With RS232 the user can download the program the microcontroller with connecting the board to computer through the COM port. The ATTINY2313 works parallel to RS232. The duty of the ATTINY2313 is to hold the program of the main microcontroller. The RS232 will get information from a PC via the COM port, and transmit that data to the ATTINY2313 to execute the programming process. By doing that the internet board will be programmed with a user friendly program loaded on computer.
To connect our analog peripheral boards, there is an analog terminal. This terminal is where we will interface the multiplexers with the microcontroller. Through the analog terminal, a selected analog signal will be transmitted to microcontrollers analog to digital converter (ADC) unit. This terminal can be assigned as either input or output. For digital peripheral boards, there is an expansion connector. The pins on the expansion board can be assigned as either input or output by the microcontroller program. In case of a digital output is needed, that is where to get it. The internet board is ready to be connected with an LCD display or keypad. The necessary circuit design is done on the board and the connection points are designated with arrows in Figure 15. The LCD and keypad are sold separately. In case of future need, those components can be connected to enter values or display the values. The clock operation crystals are used for the microcontrollers clock signals. The crystals are perfect circuit components for clocking signals. By using the crystals, then all components of the circuit work in good order.

Figure 3: Physical board layout of Mini-Max/ARM-E microcontroller board used in WebCatPlus.
Conceptual Hand Calculations

Resolution

This is the determination of the resolution of our ADC versus the ranges of voltage for the sensors and the ranges of pressure and temperature.

- 10-bit ADC -- (N=10)
- 0-5V range for temperature sensors -- (V_{fst}=5)
- 0.5-4.5V range for pressure sensors -- (V_{fsp}=4)
- -20° - 300° F temperature range -- (T_s=320)
- 0 - 700 psi pressure range -- (T_s=700)

Temperature Resolution:

\[ Q = \frac{V_{fst}}{2^N - 1} \]

\[ Q = \frac{5}{2^{10} - 1} \]

\[ Q = 0.00489 \text{ } V/\text{step} \]

\[ 320 \times 0.00489 = 1.56 \text{ } \textdegree/\text{step} \]

Pressure Resolution:

\[ Q = \frac{V_{fst}}{2^N - 1} \]

\[ Q = \frac{4}{2^{10} - 1} \]

\[ Q = 0.00391 \text{ } V/\text{step} \]

\[ 700 \times 0.00391 = 2.737 \text{ } \text{psi}/\text{step} \]
Sampling Rates

Three sampling rates are chosen for examination in this portion. They are one hertz, three hertz and five hertz. These are the options that we plan to build in as predetermined sampling rates offered for this project. They are hereafter referred to as Rate A, Rate B, and Rate C respectively in the following calculations.

Rate A

- Sampling rate of 1 Hz
- 20 sensors to be sampled
- 2.44 ms ADC conversion time (taken from technical manual)

\[
\frac{1 \text{s}}{20 \text{ sensor}} = 0.05 \text{s/sensor}
\]

\[
0.05 \text{s/sensor} - 0.00244 \text{s/conversion} = 0.0476 \text{s/sensor}
\]

Total conversion time per 20 sensor sweep:

\[
20 \text{ sensor} \times 0.00244 \text{s/conversion} = 0.0488 \text{s}
\]

- There will be one second between consecutive readings from the same sensor.
Rate B

- Sampling rate of 3 Hz
- 20 sensors to be sampled
- 2.44 ms ADC conversion time (taken from technical manual)

\[
\frac{1 \text{ s}}{(3 \text{ readings})(20 \text{ sensors})} = 0.0167 \frac{\text{s}}{\text{sensor}}
\]

\[
0.0167 \frac{\text{s}}{\text{sensor}} - 0.00244 \frac{\text{s}}{\text{conversion}} = 0.01423 \frac{\text{s}}{\text{sensor}}
\]

Total conversion time per 20 sensor sweep:

\[
20 \text{ sensor} \times 0.00244 \frac{\text{s}}{\text{conversion}} = 0.0488 \text{ s}
\]

Time between consecutive readings from same sensor:

\[
19 \times 0.01423 \frac{\text{s}}{\text{sensor}} = 0.27037 \text{ s}
\]

- There will be 0.27037 seconds between consecutive readings from the same sensor.

Rate C

- Sampling rate of 5 Hz
- 20 sensors to be sampled
- 2.44 ms ADC conversion time (taken from technical manual)

\[
\frac{1 \text{ s}}{(5 \text{ readings})(20 \text{ sensors})} = 0.01 \frac{\text{s}}{\text{sensor}}
\]

\[
0.0167 \frac{\text{s}}{\text{sensor}} - 0.00244 \frac{\text{s}}{\text{conversion}} = 0.00756 \frac{\text{s}}{\text{sensor}}
\]

Total conversion time per 20 sensor sweep:

\[
20 \text{ sensor} \times 0.00244 \frac{\text{s}}{\text{conversion}} = 0.0488 \text{ s}
\]

Time between consecutive readings from same sensor:

\[
19 \times 0.0756 \frac{\text{s}}{\text{sensor}} = 0.14364 \text{ s}
\]

- There will be 0.14364 seconds between consecutive readings from the same sensor.
**Memory Size Calculations**

The calculations given here are based upon a sample file provided by WaterFurnace Intl. The file that was provided to us was an example of data collection from a field test unit. It was purely to estimate the file size that we should be prepared to handle if the data was not downloaded or erased regularly.

**File Specifications:**

- 1350 data points
- data taken every 5 seconds
- file is .csv format
- total file size is 550.6 kB

\[
1350 \text{ samples} \times 5 \frac{s}{\text{sample}} = 6750 \text{ s}
\]

\[
(6750 \text{ s}) \left(\frac{1 \text{ min}}{60 \text{ s}}\right) \left(\frac{1 \text{ hr}}{60 \text{ min}}\right) = 1.875 \text{ hrs of operation}
\]

\[
\frac{550.6 \text{ kB}}{1350 \text{ samples}} = 0.41 \frac{\text{kB}}{\text{sample}}
\]

Based on sampling rates A, B, and C, using 1.875 hours of operation:

**Rate A:**

\[
(6750 \text{ s}) \left(1 \frac{\text{sample}}{s}\right) = 6750 \text{ samples}
\]

**Rate B:**

\[
(6750 \text{ s}) \left(3 \frac{\text{sample}}{s}\right) = 20250 \text{ samples}
\]

**Rate C:**

\[
(6750 \text{ s}) \left(5 \frac{\text{sample}}{s}\right) = 33750 \text{ samples}
\]
Memory requirement for file size based on sampling rate and number of samples:

Rate A: \[(6750 \text{ samples}) (0.41 \frac{kB}{\text{sample}}) = 2767.5 \text{ kB}\]

Rate B: \[(20250 \text{ samples}) (0.41 \frac{kB}{\text{sample}}) = 8302.5 \text{ kB}\]

Rate C: \[(33750 \text{ samples}) (0.41 \frac{kB}{\text{sample}}) = 13837.5 \text{ kB}\]

So the memory size requirements for one cycle of 1.875 hours of operation are:

Rate A: 2.8 MB
Rate B: 8.3 MB
Rate C: 13.84 MB

Now, assuming a five minute start up and shut down time (10 minutes total time):

\[10 \text{ min} \times \left( \frac{1 \text{ hr}}{60 \text{ min}} \right) = 0.167 \text{ hr}\]

\[1.875 + 0.167 = 2.042 \text{ hr}\]

\[\frac{24}{2.042} = 11.75 \text{ cycles possible per day}\]

Assuming most possible cycles per day, 12 cycles:

Rate A: \[(12 \text{ cycles})(2.8 \text{ MB}) = 33.6 \frac{\text{MB}}{\text{day}}\]

Rate B: \[(12 \text{ cycles})(8.3 \text{ MB}) = 99.6 \frac{\text{MB}}{\text{day}}\]

Rate C: \[(12 \text{ cycles})(13.84 \text{ MB}) = 166.08 \frac{\text{MB}}{\text{day}}\]
Assuming only 6 cycles per day:

Rate A: \[(6 \text{ cycles})(2.8 \text{ MB}) = 16.8 \frac{\text{MB}}{\text{day}}\]

Rate B: \[(6 \text{ cycles})(8.3 \text{ MB}) = 49.8 \frac{\text{MB}}{\text{day}}\]

Rate C: \[(6 \text{ cycles})(13.84 \text{ MB}) = 83.04 \frac{\text{MB}}{\text{day}}\]

Assuming 4 cycles per day:

Rate A: \[(4 \text{ cycles})(2.8 \text{ MB}) = 11.2 \frac{\text{MB}}{\text{day}}\]

Rate B: \[(4 \text{ cycles})(8.3 \text{ MB}) = 33.2 \frac{\text{MB}}{\text{day}}\]

Rate C: \[(4 \text{ cycles})(13.84 \text{ MB}) = 55.36 \frac{\text{MB}}{\text{day}}\]

A two hour cycle is more than most furnaces run at one time, so it would be more realistic to plan on a cycle of 10 or 20 minutes. Both cases will be assumed to have two hours of total run time per day and will be figured for each sampling rate. These calculations are shown below.

**20 Minute Case:**

\[20 \text{ minutes} \times \left(\frac{1 \text{ hour}}{60 \text{ minutes}}\right) = 0.333 \text{ hour}\]

\[(0.333 \text{ hour}) \left(\frac{60 \text{ minutes}}{1 \text{ hour}}\right) \left(\frac{60 \text{ s}}{1 \text{ minute}}\right) = 1198.8 \text{ s}\]

Rate A: \[(1198.8 \text{ s}) \left(\frac{1 \text{ sample}}{1 \text{ s}}\right) = 1198.8 \text{ samples}\]

\[(1198.8 \text{ samples}) \times \left(0.41 \frac{kB}{\text{sample}}\right) = 491.51 kB/\text{cycle}\]

\[\left(\frac{2 \text{ hr}}{1 \text{ hour}}\right) \left(\frac{60 \text{ min}}{1 \text{ hr}}\right) = 6 \text{ cycles}\]

\[(6 \text{ cycles}) \times \left(491.51 \frac{kB}{\text{cycle}}\right) = 2949.06 \text{ kB}\]

Rate A requires 2949.06 kB for one day of six 20 minute cycles.
Rate B: 

\[
(1198.8 \, s) \left( \frac{3 \, \text{samples}}{1 \, s} \right) = 3596.4 \, \text{samples}
\]

\[
(3596.4 \, \text{samples}) \times \left( 0.41 \, \frac{\text{kB}}{\text{sample}} \right) = 1474.524 \, \text{kB/cycle}
\]

\[
(2 \, \text{hr}) \left( \frac{60 \, \text{minutes}}{1 \, \text{hr}} \right) \left( \frac{1 \, \text{cycle}}{20 \, \text{minutes}} \right) = 6 \, \text{cycles}
\]

\[
(6 \, \text{cycles}) \times \left( 1474.524 \, \frac{\text{kB}}{\text{cycle}} \right) = 8847.144 \, \text{kB}
\]

Rate B requires 8847.144 kB for one day of six 20 minute cycles.

Rate C: 

\[
(1198.8 \, s) \left( \frac{5 \, \text{samples}}{1 \, s} \right) = 5994 \, \text{samples}
\]

\[
(5994 \, \text{samples}) \times \left( 0.41 \, \frac{\text{kB}}{\text{sample}} \right) = 2457.54 \, \text{kB/cycle}
\]

\[
(2 \, \text{hr}) \left( \frac{60 \, \text{minutes}}{1 \, \text{hr}} \right) \left( \frac{1 \, \text{cycle}}{20 \, \text{minutes}} \right) = 6 \, \text{cycles}
\]

\[
(6 \, \text{cycles}) \times \left( 2457.54 \, \frac{\text{kB}}{\text{cycle}} \right) = 14745.24 \, \text{kB}
\]

Rate C requires 14745.24 kB for one day of six 20 minute cycles.
10 Minute Case:

\[ 10 \text{ minutes} \times \left( \frac{1 \text{ hour}}{60 \text{ minutes}} \right) = 0.167 \text{ hour} \]

\[ (0.167 \text{ hour}) \left( \frac{60 \text{ minutes}}{1 \text{ hour}} \right) \left( \frac{60 \text{ s}}{1 \text{ minute}} \right) = 600 \text{ s} \]

Rate A:

\[ (600 \text{ s}) \left( \frac{1 \text{ sample}}{1 \text{ s}} \right) = 600 \text{ samples} \]

\[ (600 \text{ samples}) \times \left( 0.41 \frac{\text{kB}}{\text{sample}} \right) = 246 \text{ kB/cycle} \]

\[ (2 \text{ hr}) \left( \frac{60 \text{ minutes}}{1 \text{ hr}} \right) \left( \frac{1 \text{ min}}{10 \text{ minute/cycle}} \right) = 12 \text{ cycles} \]

\[ (12 \text{ cycles}) \times \left( 246 \frac{\text{kB}}{\text{cycle}} \right) = 2952 \text{ kB} \]

Rate A requires 2952 kB for one day of 12 ten minute cycles.

Rate B:

\[ (600 \text{ s}) \left( \frac{3 \text{ sample}}{1 \text{ s}} \right) = 1800 \text{ samples} \]

\[ (1800 \text{ samples}) \times \left( 0.41 \frac{\text{kB}}{\text{sample}} \right) = 738 \text{ kB/cycle} \]

\[ (2 \text{ hr}) \left( \frac{60 \text{ minutes}}{1 \text{ hr}} \right) \left( \frac{1 \text{ min}}{10 \text{ minute/cycle}} \right) = 12 \text{ cycles} \]

\[ (12 \text{ cycles}) \times \left( 738 \frac{\text{kB}}{\text{cycle}} \right) = 8856 \text{ kB} \]

Rate B requires 8856 kB for one day of 12 ten minute cycles.
Rate C: \[(600 \text{ s}) \left(\frac{5 \text{ sample}}{1 \text{ s}}\right) = 3000 \text{ samples}\]

\[(3000 \text{ samples}) \ast \left(0.41 \frac{kB}{\text{sample}}\right) = 1230 kB/cycle\]

\[\frac{(2 \text{ hr})(60 \text{ minute})}{1 \text{ hr}} \frac{10 \text{ minute/cycle}}{1 \text{ cycle}} = 12 \text{ cycles}\]

\[(12 \text{ cycles}) \ast \left(1230 \frac{kB}{\text{cycle}}\right) = 14760 kB\]

Rate C requires 14760 kB for one day of 12 ten minute cycles.
Section II: The Building Process
Building the Prototype

The prototype design and build phase of the project began with the software programming portion of the project. The first thing that was attempted was to get the WebCatPlus MMC-1 board ready for communication over the internet. This is where the first problems were encountered in the prototype construction phase. The board was found to be incapable of communication with the program provided by Bipom Electronics, WebCatPlus Publisher, and it was necessary to remove the SD memory card and load the configuration files manually through a SD memory card reader slot on one of the group members laptop computers. The board also required a complete erasing of its dataflash memory to enable it to read the configuration files from the memory card instead of the dataflash. The board was then pinged to make sure that the internet communication was functional with the result of a successful pinging. The pinging and online capabilities were initially tested using a network patch cable to connect it to a laptop computer due to the inability to access an internet connection in the engineering building at IPFW or at WaterFurnace International. Both of these sites required a password to log into a secure network connection and at this point the WebCatPlus MMC-1 does not have the ability to log into a secure, password protected internet connection.

The next step in programming the main microcontroller was to determine the commands used to read values from the analog input channels and how they are converted. These instructions were found in examples provided with the Micro IDE program provided by Bipom Electronics. Now that the values were able to be read from the analog inputs, through the analog-to-digital converter, it was necessary to determine how the multiplexers in the design would be addressed. The WebCatPlus board has two sets of general purpose input/output pins that can be used for connection to an LCD display or keypad, but are used for multiplexer addressing in this application. A resource was found for general purpose IO pin utilization for flashing LEDs using the LPC 2138 microcontroller and this code was consulted as a reference to determine how the commands were implemented. The output pins for the keypad connection were connected through wires to a bank of LEDs to test this function and it was found to be operational. The LEDs would blink as described in the tutorial and we replicated these commands to further our addressing of the multiplexers.

Once the multiplexer addressing was determined, storage of the incoming values became the focus. The values had to be stored so that they could be retrieved in a file and also published to the website. It was at this point that the website design also began. The website was created using MS Office FrontPage 2003 by Mehmet Bacak. It includes buttons to change the sampling rate, hyperlinks to view the text files where the data is stored, and a two lists of the current values for temperature and pressure readings. Some difficulty was encountered when
trying to implement code to have the website refresh itself without any external commands. However, a code was developed that would allow the website to perform this function by itself with a delay that could be set to any amount of time desired. Also when designing the website embedded commands are used to publish the time and date of the real-time clock. It was also decided that the embedded commands would be the best way to change the sampling rate. The embedded commands can instruct the WebCatPlus board to perform various actions such as display the current time and date or change the value of the general purpose IO pins. To set the sampling rate from the website a button on the main web page is clicked and an active server page is used to send the command to the WebCatPlus microcontroller. The command changes the setting of pins on Port 0, either to input or output. This idea was easy to integrate into the website code, but more difficult to integrate into the microcontroller code. There are 32 pins each on Port 0 and Port 1 and the register where the value of each pin (0 for input or 1 for output) is a 32 bit register. The code developed to set the sampling rate was only checking for the value of four of those bits or pin, but all of the values were changing as those were changed from the website. This problem was solved by incorporating a logic AND into the code to select the four bits that were to be examined and effectively zeroing out all others that were of no interest for the process. These four bits of change allowed for the nine different combinations required for the designed nine different combinations of sampling rates.

The file writing of the main microcontroller did offer several problems that were overcome through trial and error and communication with Vitaliy Avramenko at Bipom Electronics technical support. The provided example programs were searched for any predefined functions that would allow us to write the incoming voltages to .txt files created on the memory card. The functions should allow us to create, open, and write to the files any time data is collected. Functions were found in another program that could be implemented in our own code, but they did not function with the fprintf, fopen, and fclose commands that are standard in most C language libraries. There were fat_write, fat_open, and fat_close commands that took their places and required that a string of characters be written instead of numerical values. Therefore, the incoming data had to be converted and stored in another character array using the sprint command, then written to the file. It was also found that if a file was closed after writing valuable instruction processing time was taken up and the file would be written over every time it was reopened. Reopening the files also used up valuable processor time and if the new values were not stored after the old ones, the previous data would be lost and a file would be left without anything to compare the new and old data. It was these problems that led to the decision to not close the files every time they were written to.
Now that the program was able to address the multiplexers, read in data, convert the data, and store this data to memory it was decided that the program for accomplishing these tasks be configured to also include code that would make the board functional over the internet. Another search of the provided example programs ensued and a program was found that did just that portion of the code. This program was found to initialize the IP address, configurations for all FTP settings and run the Ethernet socket to provide internet content. The code that was found was modified so that it would not run the Ethernet socket at all times, only when called upon. Introduction of this portion of code into our own development did raise some problems as that it reconfigured our IO pin settings to a previous state. This problem was solved simply by moving a line of code from one spot to another, just below the recently added internet initialization code.

When this point in the programming had been reached the decision was made to construct the prototype on the breadboard and incorporate the multiplexers, signal conditioning op amps, the voltage regulation circuit, and the WebCatPlus board. Several problems were realized very quickly on this first assembly of all the circuit sections. These problems included not having a negative power supply to power the operational amplifiers, no way to test the voltage regulation circuit with a 24V AC signal, inaccurate gain across the signal conditioning op amps, unaccounted for resistance inherent in the multiplexers, and also a problem with adding the voltage divider circuit to simulate the thermistor temperature sensor.

The first problem that was overcome was the negative power supply for the op amps. The solution that was arrived at after much research was to use the MAX764 converting DC-DC inverter according to the circuit schematic shown below. A positive five volt signal is available from either the voltage regulation circuit or from the WebCatPlus board itself. With a positive voltage to drive this circuit a negative voltage was made available for use with the op amps. This five volt source also drove the positive rail of the op amps and ensured a more predictable behavior from the signal conditioning portion of the prototype.
The signal conditioning portion of our prototype also introduced other problems to be overcome. The second of which was an inconsistent gain that did not match the common equation for determining the output gain for an inverting op amp configuration. Even with resistors that are specified at 0.1% tolerance there were still errors, most likely due to the added resistance of the multiplexers. To correct this error in gain potentiometers were added so that the resistance across the feedback resistor could be varied and the gain could be observed using a Tektronix multimeter in the lab. This allowed the best ability to produce the desired gain across the signal conditioning.

The problem of testing the voltage regulation circuit shown below was overcome very quickly as it was testable with any voltage greater than 5V. Simply applying a 20V DC power source to the input of the circuit produced a 5V output. A test of voltages greater than 5V showed that the output was always regulated down to the desired five volts.

**Figure 4:** Schematic of MAX764 voltage inverting circuit for negative power supply to op amps.

**Figure 5:** Circuit schematic of the voltage regulation portion of circuitry.
The most daunting problem faced came at the end of the building process. There was an unexplained loss of signal through the multiplexer when the voltage divider circuit was applied to the multiplexer input. It was the addition of the voltage divider that caused this problem. The circuit was tested down to the component level to determine what was causing this problem. It was found to be the increased impedance load of the voltage divider circuit. Under the direction of Dr. Oloomi, our project advisor, a unity gain buffer op amp was added to isolate the impedances and shield them from each other. Once this idea was implemented the signal flowed correctly across all pins of the multiplexers. The solution did affect the gain across the signal conditioning circuit as that the unity gain buffer was not an exact one-to-one ratio. That was quickly resolved using the potentiometers as mentioned earlier in a previous problem.

An outlined picture of the final prototype is provided on the bottom of this page as a reference to show what portions of the circuit are being explained. There is also a detailed schematic in the attached Appendix at the end of this report.

**Final Prototype:**

[Diagram of the final prototype]

**Figure 6:** Picture of the final prototype built on a development board.
Figure 7: Final circuit schematic for the prototype design.
Section III:

Testing and Results
List of parameters

This project must do and contain the following concepts

- Must be able to read, record, and transmit or transfer data for up to 12 different temperature points and up to eight different pressures
- Must be able to read the temperatures value with the range of -10 to 275 degrees Fahrenheit and a pressure value of zero to 700 PSI
- Must be able to detect a 24V fault signal presence and absence to start the reading
- Must be able to sample at different time of a giving sampling time, for example once every second
- Must be able to support a website and publish the data to it
- The website should be easy to use, easy to understand and follow without any problems.
- Must have a graphical interface that will be easy to use. Therefore a software program was written to collect, store, retrieve, and transmit data.
- The data should be output in a real-time and in a file format (.csv) or equivalent
- The system must be safe to connect to WaterFurnace system and send and receive data.
The Procedure of the Testing Process

As far as the testing process there were two kinds of tests that took place. One took place at the engineering building and the other one at WaterFurnace. However the majority of the testing took place at the engineering building, where there was equipment available such as a PC (Computer), multimeter, power supply, oscilloscope and probes. The power supply was used to simulate all signals including the fault signal, temperature sensor voltage divider circuit, and incoming DC voltage of the pressure sensor. A multimeter was used to check for AC or DC voltages, and resistance. Also the multimeter was used to check and see if there is voltage present on the circuit once built, which was very important because a high voltage on the circuit which will cause some damages to the electrical components used. Therefore the multimeter was a very helpful piece of equipment to have in the lab because with the type of project being performed required that it be checked that the voltage being applied a tolerable voltage to the board. Oscilloscopes were frequently used to troubleshoot electronic equipment which is not functioning correctly. Furthermore the oscilloscope to give a better view of what the problem is and what signals were coming through. As the oscilloscope has the ability to display a signal graphically which will make it more clear and understandable of what is the system going through. Finally with the equipment available in the engineering buildings lab we were able to finish the project successfully without any problems.

The second kind of test took place at WaterFurnace. There was not a lot testing done there because most of the testing equipments could be accomplished at the engineering buildings lab. However, WaterFurnace did provide the actual pressure sensors and the temperature sensors when testing occurred there. The temperature sensors were placed in a tank of oil and two pressure sensors were threaded onto nozzles where a pressurized tank provided pressure. The pressure was set up by the pressure gage where it could be increased and decreased to watch the values change. After the setup was completed the program was run to see if it was going to read the sensors. As a result it turned out that the final prototype was successfully working without any problems on the second trip to WaterFurnace International. The first test process to occur there showed some flaws in the programming and circuit design that required addressing.
Data

The data obtained in performing this project was downloaded in .txt file form and is displayed in the following screen shot figures. Several versions of the data obtained are present in this section of the report. The first few sets of data show the incorrect results obtained from having incorrect conversion formulas and a circuit loading problem. In the calculations section of this paper, the derivations of our conversion equations are available.

The first two figures shown below, Figure 8 and Figure 9, are examples of the files that are created on the memory card strictly for use by the website. These two files are how the website publishes the most current data for viewing on the main page.

**Figure 8:** WEBTEMP file for displaying most current readings of temperature sensors on webpage.

**Figure 9:** WEBPRES file for displaying most current readings of pressure sensors on webpage.

**Figure 10:** The picture of temperature file with correct data
The above figure is showing a piece of temperature file. The sample values on the each column are related to a different sensor. As it can see on the picture, there are zeros on some columns. The reason for that the program was written in a way that it will put the zeros on the channels if they are connected to ground in other word if the channels are not in use it will be zero.

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>Format</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 11**: The picture of pressure file with correct data

The above figure is showing a piece of pressure file. The sample values on each column are related to a different sensor. As it can see on the picture, there are negative values on some columns. The columns which have negative values are grounded in other word they are not in use.

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>Format</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 12**: the picture of temperature file with wrong temperature values

The above picture is showing temperature while the system was working incorrectly. The system was putting negative values on file. The values were not only negative they were also way off then they supposed to be. By going over the conversion equation and finding out that it was incorrect. The equation that used was for Kelvin instead of Fahrenheit.
Figure 13: The picture of pressure file with wrong sampling values

The above picture is showing a piece of pressure file while the system was working incorrectly. The system was putting random values on the channels which are not even connected. To come over this problem some delay had been places on the programming right after the setting the addressing. Also, grounding the channels, which are not in use, helped to solve the problem.
Calculations and Analysis

Resolution

This is the determination of the resolution of our ADC versus the ranges of voltage for the sensors and the ranges of pressure and temperature.

- 10-bit ADC -- (N=10)
- 0-3V range for temperature sensors -- (V_{fst}=3)
- 1/3-3V range for pressure sensors -- (V_{fsp}=8/3)
- -10° - 275° F temperature range -- (T_s=285)
- 0 - 725psi pressure range -- (T_s=725)

Temperature Resolution:

\[ Q = \frac{V_{fst}}{2^N - 1} \]

\[ Q = \frac{3}{2^{10} - 1} \]

\[ Q = 0.002933 \ V/\text{step} \]

285 \times 0.002933 = 0.8359 \ degree/\text{step}

Pressure Resolution:

\[ Q = \frac{V_{fsp}}{2^N - 1} \]

\[ Q = \frac{8}{3} \times \frac{3}{2^{10} - 1} \]

\[ Q = 0.002607 \ V/\text{step} \]

725 \times 0.002607 = 1.8901 \ psi/\text{step}
The form of the Steinhart-Hart equation for modeling a thermistors resistance based on temperature is:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3$$

In this equation, $T$ is in Kelvin. We chose three different temperature point from the data sheet provided by WaterFurnace, and record the temperature and corresponding resistance. The three points we picked are -10°F, 55°F, and 275°F.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Resistance</th>
</tr>
</thead>
<tbody>
<tr>
<td>T=-10°F</td>
<td>R=118050Ω</td>
</tr>
<tr>
<td>T=55°F</td>
<td>R=17435Ω</td>
</tr>
<tr>
<td>T=275°F</td>
<td>R=265Ω</td>
</tr>
</tbody>
</table>

Now convert the temperature value to Kelvin using the following equation

$$K = \frac{5}{9}(-F - 32) + 273$$

F=-10°F

$$K = \frac{5}{9}(-10 + 459.67) \left(\frac{5}{9}\right)$$

$$K = 249.817K$$

F=55°F

$$K = \frac{5}{9}(55 + 459.67) \left(\frac{5}{9}\right)$$

$$K = 285.9277K$$

F=275°F

$$K = \frac{5}{9}(275 + 459.67) \left(\frac{5}{9}\right)$$

$$K = 408.15K$$

Now we calculate the values to the Steinhart-Hart equation to solve for $A$, $B$, and $C$ by solving three simultaneous equations. The results are as follows

A=0.0011260551
B=0.0002346182
C=0.0000000859
**Pressure Sensor Conversion Formula**

The pressure sensors used in this project output a voltage in the range of 0.5V and 4.5V. However, the analog terminals are limited by a circuit component to a voltage maximum of 3V. Fortunately the pressure sensor has a linear output versus the pressure that it is sensing. Shown below is a table from the Danfoss data sheet for NSK pressure sensors showing the relationship between voltage and pressure.

![Graph of pressure to voltage ratio for NSK pressure sensor.](image)

To scale the voltage down to an acceptable voltage for the WebCatPlus boards analog inputs the upper and lower limits of voltage were multiplied by 2/3 (coming from 3.0/4.5). This two-thirds would be our gain across the signal conditioning op amps. The graph and table on the following page show the conversion performed in MS Excel and the graph used to find a trendline equation for use in our conversions.
Table 1: Pressure vs. Voltage after conversion for use in main program

<table>
<thead>
<tr>
<th>Pressure</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.333333</td>
</tr>
<tr>
<td>362.5</td>
<td>1.67</td>
</tr>
<tr>
<td>725</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 16: Graph of conversion and trendline used to find conversion from voltage to pressure.
## Final Price List for Prototype

### Prototype Circuit Section

<table>
<thead>
<tr>
<th>Op amp signal conditioning</th>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>Price for Quantity Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six LM741</td>
<td>LM741CNFS</td>
<td>6</td>
<td>$0.49/1</td>
<td>$2.94</td>
</tr>
<tr>
<td>zener diode</td>
<td>1N4733ADICT-ND</td>
<td>3</td>
<td>$0.46</td>
<td>$1.38</td>
</tr>
<tr>
<td>Potentiometer 10K</td>
<td><em>Radio Shack</em></td>
<td>2</td>
<td>$1.49</td>
<td>$2.98</td>
</tr>
<tr>
<td>Resistors 10k ±0.1%</td>
<td>10KADCT-ND</td>
<td>16</td>
<td>$0.62</td>
<td>$9.92</td>
</tr>
<tr>
<td>Resistors 1K ±5%</td>
<td><em>Radio Shack</em></td>
<td>1</td>
<td>$1.00/5</td>
<td>$1.00</td>
</tr>
<tr>
<td>Resistors 1.5K ±5%</td>
<td><em>Radio Shack</em></td>
<td>1</td>
<td>$1.00/5</td>
<td>$1.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Power Supply (-5V)</th>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>Price for Quantity Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverting chip (MAX764)</td>
<td>MAX764CSA+-ND</td>
<td>1</td>
<td>$6.10</td>
<td>$6.10</td>
</tr>
<tr>
<td>Inductor</td>
<td>M9989-ND</td>
<td>1</td>
<td>$1.13</td>
<td>$1.13</td>
</tr>
<tr>
<td>Schottky Diode</td>
<td>1N5817FSCT-ND</td>
<td>1</td>
<td>$0.54</td>
<td>$0.54</td>
</tr>
<tr>
<td>Capacitor: 120µf, Dielectric</td>
<td>P10270-ND</td>
<td>1</td>
<td>$0.32</td>
<td>$0.32</td>
</tr>
<tr>
<td>Capacitor: 68µf, Dielectric</td>
<td>P10292-ND</td>
<td>1</td>
<td>$0.28</td>
<td>$0.28</td>
</tr>
<tr>
<td>Capacitor: 10µf, Ceramic</td>
<td>BC1148CT-ND</td>
<td>1</td>
<td>$0.06</td>
<td>$0.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AC to DC Voltage Regulator</th>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>Price for Quantity Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>7805 voltage regulator</td>
<td>KA7805ETU-ND</td>
<td>1</td>
<td>$0.66</td>
<td>$0.66</td>
</tr>
<tr>
<td>Diode</td>
<td>1N5400-E3/54GICT-ND</td>
<td>5</td>
<td>$0.60</td>
<td>$3.00</td>
</tr>
<tr>
<td>Capacitor: 0.1µf, Ceramic</td>
<td>BC1160CT-ND</td>
<td>1</td>
<td>$0.06</td>
<td>$0.06</td>
</tr>
<tr>
<td>Capacitor: 4700µf, Dielectric</td>
<td>565-1091-ND</td>
<td>1</td>
<td>$2.21</td>
<td>$2.21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multiplexers</th>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>Price for Quantity Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplexer</td>
<td>ADG406BNZ-ND</td>
<td>2</td>
<td>$8.78/1</td>
<td>$17.56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Microcontroller Board</th>
<th>Quantity</th>
<th>Price</th>
<th>Price for Quantity Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebCatPlus MMC-1</td>
<td>1</td>
<td>$179</td>
<td>$179.00</td>
</tr>
</tbody>
</table>

**TOTAL** $230.14

* - Before added cost of sensors
Section IV:

Evaluation and Recommendations
Evaluation

After we were done with designing our project and made our evaluation, we realized that we have met our requirements and parameters. We accomplished what we had proposed at the beginning of design progress.

The first goal was collecting data from a number of sensors, twelve temperature and eight pressure sensors. Our designed unit is able to collect analog data from that many sensors. Moreover, the requirement that our system can be developed to add more sensors and only make minor changes to the program. Our chosen sensors are able to take measurements in the ranges that WaterFurnace required. Another requirement was an ability to change the sampling rate over the Internet without visiting the testing site. Our designed website has ability to give the option to the user to change the sampling rate to a predefined sampling rate. As WaterFurnace required our website is giving three different sampling rate options, in nine combinations, to the user, once every second, once every three seconds and once every five seconds. On the website side, we designed a website which is user friendly. There is not a lot of data to put on website. The main page has buttons for the nine different combinations of sampling rates. The main page also shows the current or latest readings from the sensors and divides them in two groups; one row for pressure and one for temperature. Also when users click on hyperlinks, they are able to see the entire sampling files and copy them to their computer. The requirement about not starting sampling until a fault signal is present has also been met. The process of sampling and recording does not start until the fault signal is received from the geothermal systems main control board. Moreover, the unit is checking for signal during the process and if the is signal lost, it stops sampling and puts an error message on the file such as "Signal Lost." When system put the error message it stamps the real-time value as well. Thus the user will be able to know when the system stopped and started to work again. The accuracy of the sampling was another thing that WaterFurnance strictly stated. Our system samples the signal in an accuracy which falls within the range tolerated. Therefore the reading values are accurately showing up on the sampling files. The memory where the readings are recorded had to be easily readable and able to be deleted. On this step, we decided to use a standard multimedia card. The user is able to take out the card, plug in to any laptop take a copy and delete file. On memory side of our system is very easy to use and work. Also another advantage of the memory card is it is developable or updateable. It is a standard memory card so they can get it from anywhere conveniently and in a wide range of memory capacities. Another requirement of WaterFurnace was the budget. We had been given a five hundred dollar budget to complete the project and develop a working prototype. We have designed a working prototype within the budget, we did not go over the budget. The requirement of budget has successfully met too. The time
limitation was completing the project by the last week of classes in May. By the first week of May we have completed the design of the project and our prototype is working well.

**Recommendations**

After we completed the design, we sat down and thought about the mistakes we made and how they made the project more difficult during the designing and building process.

First, and the biggest problem we have had, was the setting the gain of amplifiers. Some various reasons we had difficulty to obtain the gain we desired. The reasons for that were:

1) The resistances were appearing on the switches of the multiplexer and when it was combined the input resistance of amplifier they were acting as a fixed resistor. For this reason we had some difficulties to set the gain.

2) To not be able to get the same voltage on output of amplifier made us suspect other problems such as conversion equations. We had many times to check the correctness of the conversion equations.

3) On temperature sensors we design a voltage divider to reduce the voltage into a harmless level for board. But the voltage divider was giving us a loading problem. The output of voltage divider was not the same when it was connected to channels of multiplexer with when it was disconnected. To solve this issue we had to add unity gain feedback buffers to our signal conditioning part of system.

4) During the project we should not forget that we are working as a team and notify other members of the team about the changes we have made. If had done this, a part of general purpose input/output pins on PORT1 would not be damaged and unusable.

5) On programing we had to be more careful about the brackets and punctuation. One missing or extra bracket made us look over entire program to find the mistake.

6) While building the prototype on breadboard, we had to be more organized for placing the components and the wires to avoid short circuits and misconnections.
7) While deciding the circuit components, we had to be observant of the parameters. The information on data sheets give many tips about the working principle of the components.

8) Predefined functions and codes in programming can be a problem as much as they are useful. In case of needing of predefined function, the first thing to do must be to learn the idea behind the function and how to use it.
Conclusion

The remote web-based field data acquisition (FielDAQ) system is a project that should design a system that will monitor several pressure and temperature sensors around the unit, and store the information from each sensor and make it readily available through a website. This project required a lot of research, reading and planning to be able to have a chance to complete it all. Therefore as we promised we were able to finish this project by meeting the entire requirement and the specification that we mentioned in the problem statement. As a result this project reads, monitors and publishes all the pertinent data to the website.

In conclusion this project was a challenging project; however it was a good learning experience for us. This project gave us a lot of characteristics on how to approach an engineering problem and how to solve such problems. This project made us see what are the steps to solve an engineering problem; by going through planning, improvement, development, testing, and implementation. Finally with this project we should be ready to solve any problem we encounter when we join work force because of the experience we have gained.
References

1. General Purpose IO example for flashing LEDs using LPC 2138:

   http://www.8051projects.net/lofiversion/t4128/Arm-Development---GPIO-tutorial-.html

2. C Language functions library

   http://www.cppreference.com/wiki/c/all

3. Unity Gain Buffer Op Amp Circuit Design

   http://www.facstaff.bucknell.edu/mastascu/elessonshtml/OpAmps/OpAmp3Note1Buffer.html

4. Danfoss NSK pressure sensor datasheet

   http://www.ra.danfoss.com/TechnicalInfo/Literature/Manuals/01/DKRCC.PD.SE0.C1.22.pdf

5. MAX764 Example Circuit Diagram

   http://jkx.larsen-b.com/photos/Electronic/MAX_764.png

6. Reading a .txt file to a .asp page

   http://www.w3schools.com/asp/asp_ref_textstream.asp

7. Monitoring a .txt file on a website

   http://support.microsoft.com/kb/300982

8. Automatic website page refreshing

   http://www.quackit.com/javascript/javascript_refresh_page.cfm

9. Voltage regulation circuit diagram example

Appendix
Final Circuit Schematic:
Main Microcontroller Code:

```c
#include <stdio.h>   ///////////// All header files to be included
#include <stdlib.h>
#include <string.h>
#include <BipomTypes.h>
#include <init.h>
#include <uart.h>
#include <uartInt.h>
#include <BipomFunc.h>
#include <i2c.h>
#include <Ethernet.h>
#include <critical.h>
#include <uip\include\uip.h>
#include <ipx.h>
#include <fat\include\fat.h>
#include <fat\drivers\mmc_disk_driver.h>
#include <fat\drivers\df_disk_driver.h>
#include <fat\drivers\ram1_disk_driver.h>
#include <network\include\sockets.h>
#include <fat\include\filesys\fs.h>
#include <network\include\CommonTag.h>
#include <network\Tag.h>
#include <network\include\CommonCommand.h>
#include <network\command.h>
#include <network\include\ipxbuffer.h>
#include <network\include\ipxserver.h>
#include <network\include\httpserver.h>
#include <network\include\ftpserver.h>
#include <network\include\telnetserver.h>
#include <fat\filesys\uart0Loader.h>
#include <network\include\config.h>
#include <math.h>
#define WEBCAT_WDT_TIMEOUT 30
#define WEBCAT_WDT_RESET 10
#define INCLUDE_FTP_SERVER
#define delay 3000;
#define File_1 "WFTempData.txt"
#define File_2 "WFPresData.txt"
#define File_3 "WEBTEMP.txt"
#define File_4 "WEBPRES.txt"
#define A  0.0011255095
#define B  0.0002347322
#define C  0.0000000854
#define R  95000
#define V  5.0
```
int main (void)
{

////////////////////////////////////////////////////////////////////
///     Declaration of all variables used in program  ///
////////////////////////////////////////////////////////////////////
UBYTE WDT_curSecond = 60;
UBYTE WDT_cntSecond = 0;
UBYTE channel;
UBYTE Tchan;
UBYTE Pchan;
UINT val;
UINT Tval;
UINT Pval;
FILE *WFdat1;
SOCKET s;

tVolume* mmc;
tFile *file1;
tFile *file2;
tFile *file3;
tFile *file4;

cchar message1[256];
cchar message2[256];
cchar RTClock[32];
cchar CheckIOPIN[32];
cchar *lostmsg = "*** Signal Lost ***";
cchar* fileName = "WFTempData.txt";
cchar* fileName1 = "WFPresData.txt";
cchar* fileName2 = "WEBTEMP.txt";
cchar* fileName3 = "WEBPRES.txt";
float RT;
float Temperature;
float voltage;
float Tvoltage;
float Pvoltage;
float j1;
float j2;
float j3;
float j4;
float j5;
float j6;
float j7;
float j8;
float j9;
float j10;
float j11;
float j12;
float p1;
float p2;
float p3;
float p4;
float p5;
float p6;
float p7;
float p8;
float k;
unsigned int m;
unsigned int n;
unsigned int d;
unsigned int j;
unsigned int p;
unsigned int z;
unsigned int x;
unsigned int delayset;
int Tdelay;
int Pdelay;

////////////////////////////////////////////////////////////////////////
///// IO pins set for output for Multiplexer addressing and ADC channels set for /////
///// Temperature or pressure bank of sensors.
/////  
////////////////////////////////////////////////////////////////////////
Initialize();
d= 0x00060000;
j=1;
n=0;
m=0;
IOSET0 = 0x00000000;
IODIR1 = 0x00FF0000;
channel= AN0_INPUT;
Tchan= AN4_INPUT;
Pchan = AN2_INPUT;

////////////////////////////////////////////////////////////////////////
///// Files created on MMC storage
////////////////////////////////////////////////////////////////////////
mmc_fat_mount();
mmc=mmc_get_volume();
switch(mmc->FatType);
fat_open(mmc, File_1, O_CREAT | O_WRONLY, 0, 0, &file1);
fat_open(mmc, File_2, O_CREAT | O_WRONLY, 0, 0, &file2);
fat_open(mmc, File_3, O_CREAT | O_WRONLY, 0, 0, &file3);
fat_open(mmc, File_4, O_CREAT | O_WRONLY, 0, 0, &file4);

sprintf(RTClock, "%i:%i:%i-%i/%i/%i," , HOUR, MIN, SEC, MONTH, DOM, YEAR);
fat_write(file1, RTClock, strlen(RTClock));
fat_write(file2, RTClock, strlen(RTClock));

///////////////////////////////////////////////////////////////////////
///////////  Initialize system and initialize internet connection///////
///////////////////////////////////////////////////////////////////////

CCR = 0x11;
uart0Init(B115200, UART_8N1, UART_FIFO_8);
LoadHardDrive();
if(ResetSoftWDT(WEBCAT_WDT_TIMEOUT))
{
    uart0Puts("\n\rERROR: UNKNOWN HARDWARE");
    for(;;);
}

mmc_fat_mount();
df_fat_mount();
ram1_fat_mount();

CFG.isDebugMessage = CONFIG_YES;

CFG.hostIP[0] = 192;
CFG.hostIP[1] = 168;
CFG.hostIP[2] = 0;
CFG.hostIP[3] = 8;

CFG.gateIP[0] = 192;
CFG.gateIP[1] = 168;
CFG.gateIP[2] = 0;
CFG.gateIP[3] = 9;

CFG.netMask[0] = 255;
CFG.netMask[1] = 255;
CFG.netMask[3] = 0;

CFG.listenHttpPort = 80;

CFG.baudRate = 115200;
CFG.parity = 0;
//CFG.isLCD = CONFIG_YES;

//CFG.isKEYPAD = CONFIG_YES;

CFG.user[0] = 0;
CFG.password[0] = 0;
CFG.isProtected = PROTECTED;

CFG.ExternalTags = NULL;

CFG.ExternalCommands = NULL;

CFG.ftpPort = 21;
CFG.ftpDataPort = 2121;
CFG.ftpLogin[0] = 0;
CFG.ftpPassword[0] = 0;

GetConfigParameters(&CFG);

uart0IntConfig(UART_BAUD(CFG.baudRate), PARITY_BIT[CFG.parity], UART_FIFO_14);

Initialize_ETHERNET();
Initialize_IPX();
Initialize_IPXBUFFER(ETHERNET_BUFFER);

Initialize_SOCKET();

Reset_SOCKET(&s);
s.family = TCP_TIMER_SOCKET;
s.pApplication = TIMER_APP_CALL;
Register_SOCKET(&s);

Reset_SOCKET(&s);
s.family = AF_IPX_SOCKET;
memset(s.dNetwork.v, 0, sizeof(IPX_NET));
memset(s.sNetwork.v, 0, sizeof(IPX_NET));
s.port = DEFAULT_TX_SOCKET;
s.sPort= DEFAULT_RX_SOCKET;
s.pApplication = IPX_APP_CALL;
Register_SOCKET(&s);

Reset_SOCKET(&s);
s.family = AF_INET_SOCKET;
s.port = CFG.listenHttpPort;
s.type = SOCK_STREAM_SOCKET;
s.mode = LISTEN_MODE_SOCKET;
s.pApplication = HTTP_APP_CALL;
Register_SOCKET(&s);
HTTP_INIT();
TAG_INIT();
COMMAND_INIT();

Reset_SOCKET(&s);
s.family = AF_INET_SOCKET;
s.port = 23;
s.type = SOCK_STREAM_SOCKET;
s.mode = LISTEN_MODE_SOCKET;
s.pApplication = TELNET_APP_CALL;
Register_SOCKET(&s);

#ifdef INCLUDE_FTP_SERVER
Reset_SOCKET(&s);
s.family = AF_INET_SOCKET;
s.port = CFG.ftpPort;
s.type = SOCK_STREAM_SOCKET;
s.mode = LISTEN_MODE_SOCKET;
s.pApplication = FTP_APP_CALL;
Register_SOCKET(&s);

Reset_SOCKET(&s);
s.family = AF_INET_SOCKET;
s.port = CFG.ftpDataPort;
s.type = SOCK_STREAM_SOCKET;
s.mode = LISTEN_MODE_SOCKET;
s.pApplication = FTP_DATA_CALL;
Register_SOCKET(&s);

char buf[32] = {0};
sprintf(buf, "%d.%d.%d.%d", CFG.hostIP[0],CFG.hostIP[1],CFG.hostIP[2],CFG.hostIP[3]);
FTP_INIT(buf, CFG.ftpDataPort);
#endif

IP4_ADDR(CFG.hostIP[0],CFG.hostIP[1],CFG.hostIP[2],CFG.hostIP[3]);
DR_ADDR(CFG.gateIP[0],CFG.gateIP[1],CFG.gateIP[2],CFG.gateIP[3]);
NETMASK(CFG.netMask[0],CFG.netMask[1],CFG.netMask[2],CFG.netMask[3]);

if (strstr(CFG.user,"ANONYMOUS"))
    if (strstr(CFG.password,"GUEST"))
        

CFG.isProtected = NON_PROTECTED;

switch (CFG.parity)
{
    case 0: uart0Puts("NONE");break;
    case 1: uart0Puts("EVEN");break;
    case 2: uart0Puts("ODD");break;
    default: uart0Puts("UNKNOWN");break;
}

#ifdef INCLUDE_FTP_SERVER
#endif

///////////////////////////////////////////////////////////////////////////////////////
//////// Beginning of program for sampling and publishing data to website /////////
///////////////////////////////////////////////////////////////////////////////////////

IODIR1 = 0x00FF0000;
goto CheckSignal;
delayset=0x00000000;
IOCLR0=0x00E80000;
// goto CheckDelete;

CheckSampling:

//goto CheckDelete;
there:

delayset=IOPIN0;
delayset = (delayset & 0x00E80000);

if (delayset==0x00000000) //T1 P1
{
    m=0;

goto TSampling;

tr1:

goto PSampling;

tr2:

    m=0;
goto CheckSampling;
}
if (delayset==0x00080000) //T1 P3
{
  m=2;

  tr3:
  tr4:
  tr5:

  goto TSampling;
  tr6:
  goto PSampling;
  tr7:
    m=2;
  goto CheckSampling;
}
if (delayset==0x00200000) //T1 P5
{
  m=7;

  tr8:
  tr9:
  tr10:
  tr11:
  tr12:

  goto TSampling;
  tr13:
  goto PSampling;
  tr14:
    m=7;
  goto CheckSampling;
}
if (delayset==0x00280000) //T3 P1
{
  m=14;

  tr15:
  tr16:
  tr17:

  goto PSampling;
  tr18:
  goto TSampling;
  tr19:
    goto CheckSampling;
}
if (delayset==0x00400000) //T3 P3
{
    m=19;
    tr20:
    tr21:
    tr22:
    tr23:
        x=50000;
        goto PublishWebSite;
    tr24:
        goto TSampling;
    tr25:
        goto PSampling;
    tr26:
        goto CheckSampling;
}  //T3 P5
    if (delayset==0x00480000)  
    {  
        m=26;
        tr27:
        tr28:
        tr29:
        tr30:
            x=50000;
            goto PublishWebSite;
        tr31:
            goto TSampling;
        tr32:
        tr33:
            x=50000;
            goto PublishWebSite;
        tr34:
            goto PSampling;
        tr35:
            goto TSampling;
        tr36:
        tr37:
        tr38:
        tr39:
            x=50000;
            goto PublishWebSite;
        tr40:
            goto TSampling;
        tr41:
            goto PSampling;
        tr42:
        tr43:
x=50000;
goto PublishWebSite;
tr44:
    goto TSampling;
tr45:
tr46:
tr47:
tr48:
    x=50000;
goto PublishWebSite;
tr49:
    goto TSampling;
tr50:
    goto PSampling;
tr51:
    goto CheckSampling;
}
if (delayset==0x00600000) //T5 P1
{
    m=51;
tr52:
tr53:
tr54:
tr55:
    goto PSampling;
tr56:
    goto PSampling;
tr57:
    goto PSampling;
tr58:
    goto CheckSampling;
}
if (delayset==0x00680000) //T5 P3
{
    m=57;
tr581:
tr59:
tr60:
tr61:
    x=50000;
goto PublishWebSite;
tr62:
    goto PSampling;
tr63:
tr64:
    x=50000;
goto PublishWebSite;
tr65:
goto TSampling;

tr66:
goto PSampling;

tr67:  
tr68:  
tr69:  
tr70:  
  
x=50000;
goto PublishWebSite;

tr71:  
tr72:  
tr73:  
tr74:  
  
x=50000;
goto PublishWebSite;

tr75:  
tr76:  
tr77:  
tr78:  
tr79:  
  
x=50000;
goto PublishWebSite;

tr80:  
tr81:  
tr82:  
   
goto CheckSampling;
  
} 

if (delayset==0x00C00000)    //T5 P5 
{ 
  m=82;

tr83:  
tr84:  
tr85:  
tr86:  
  
x=100000;
goto PublishWebSite;

tr87:  

tr88:  
goto TSampling;

tr89:
goto CheckSampling;
}

////////////////////////////////////////////////////////////////////
///////////////Temperature Sampling Portion Code ///////////////////
////////////////////////////////////////////////////////////////////

TSampling:

IOSET1 = 0x00100000;       /// Set Port1 to correct address
delayUs(50);
Tval = GetADC(Tchan);       /// Get voltage from analog port
Tvoltage = (((float)Tval) * VREF)/ 1023.0; /// Convert voltage to Temperature
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)   /// Check if the channel is grounded
{ j1=0;}
else
j1=Temperature;   /// Record the temperature to 'J1' variable
IOCLR1 = 0x00100000;

IOSET1 = 0x00200000;       /// Set Port1 to correct address
delayUs(50);
Tval = GetADC(Tchan);       /// Get voltage from analog port
Tvoltage = (((float)Tval) * VREF)/ 1023.0; /// Convert voltage to Temperature
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)    /// Check if the channel is grounded
{ j2=0;}
else
j2=Temperature;   /// Record the temperature to 'J2' variable
IOCLR1 = 0x00200000;

IOSET1 = 0x00300000;       /// Set Port1 to correct address
delayUs(50);
Tval = GetADC(Tchan);       /// Get voltage from analog port
Tvoltage = (((float)Tval) * VREF)/ 1023.0; /// Convert voltage to Temperature
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)    /// Check if the channel is grounded
{ j3=0;}
else
j3=Temperature;   /// Record the temperature to 'J3' variable
IOCLR1 = 0x00300000;
IOSET1 = 0x00400000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
    { j4=0; }
else
    j4=Temperature;
IOCLR1 = 0x00400000;

IOSET1 = 0x00500000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
    { j5=0; }
else
    j5=Temperature;
IOCLR1 = 0x00500000;

IOSET1 = 0x00600000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
    { j6=0; }
else
    j6=Temperature;
IOCLR1 = 0x00600000;

IOSET1 = 0x00700000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
    { j7=0; }
else
    j7=Temperature;
{ j7=0; }
else
j7=Temperature;
IOCLR1 = 0x00700000;

IOSET1 = 0x00800000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
{ j8=0; }
else
j8=Temperature;
IOCLR1 = 0x00800000;

IOSET1 = 0x00900000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
{ j9=0; }
else
j9=Temperature;
IOCLR1 = 0x00900000;

IOSET1 = 0x00A00000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
{ j10=0; }
else
j10=Temperature;
IOCLR1 = 0x00A00000;

IOSET1 = 0x00B00000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
	{j11=0;}
else
	j11=Temperature;
IOCLR1 = 0x00B00000;

IOSET1 = 0x00C00000;
delayUs(50);
Tval = GetADC(Tchan);
Tvoltage = (((float)Tval) * VREF)/ 1023.0;
RT = logf((Tvoltage*R)/(V-Tvoltage));
Temperature = 1/(A+B*(RT)+C*(RT*RT*RT));
Temperature = ((Temperature)*(1.8))-459.67;
if (Temperature<-10)
	{j12=0;}
else
	j12=Temperature;
IOCLR1 = 0x00C00000;

////////////////////////////////////////////////////////////////////////////////////////
///// Write the variables to Temperature file /////
////////////////////////////////////////////////////////////////////////////////////////

fat_write(file1, "\n", 2);
sprintf(RTClock, "%i:%i:%i", HOUR, MIN, SEC);
fat_write(file1, RTClock, strlen(RTClock));
sprintf (message1, "%8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f", j1, j2, j3, j4, j5, j6, j7, j8, j9, j10, j11, j12);
fat_write(file1, message1, strlen(message1));
fat_write(file3, message1, strlen(message1));
fat_close(file3);
fat_open(mmc,File_3, O_WRONLY, 0,0, &file3);
x=46500;  // Set the 'x' variable to complete the sampling cycle to one second

////////////////////////////////////////////////////////////////////////////////////////
///// Check for fault signal again /////
////////////////////////////////////////////////////////////////////////////////////////

val=GetADC(channel);
voltage = (((float)val) * VREF)/ 1023.0;
k=0.5;
IOCLR1=0x00FF0000;
if (voltage <= k) // Check for fault signal again ///
{
    goto FaultLost;
}
goto PublishWebSite;

////////////////////////////////////////////////////////////////////
/////////////// Pressure Sampling Portion Code /////////////////////
////////////////////////////////////////////////////////////////////

PSampling:

p = 0x00000000;
for( p < 0x00900000; p=p+0x00100000;) // increment addressing IO pins //
{
    IOSET1=p; // Set Port1 to correct address
delayUs(50);
Pval = GetADC(Pchan); // Get voltage from analog port
Pvoltage = (((float)Pval) * VREF)/ 1023.0; // Convert voltage to Pressure
Pvoltage = (270.2702703)*(Pvoltage-(0.33333333));

switch(p)
{
    case 0x100000:
        p1=Pvoltage; // Record the pressure to 'P1' variable
        IOCLR1 = p; // Clear the pins on Port1
        break;
    case 0x200000:
        p2=Pvoltage; // Record the pressure to 'P2' variable
        IOCLR1 = p; // Clear the pins on Port1
        break;
    case 0x300000:
        p3=Pvoltage;
        IOCLR1 = p;
        break;
    case 0x400000:
        p4=Pvoltage;
        IOCLR1 = p;
        break;
    case 0x500000:
        p5=Pvoltage;
        IOCLR1 = p;
        break;
}

// End of Pressure Sampling Portion Code
case 0x600000:
p6=Pvoltage;
IOCLR1 = p;
break;

case 0x700000:
p7=Pvoltage;
IOCLR1 = p;
break;

case 0x800000:
p8=Pvoltage;
IOCLR1 = p;
break;
}
}

///////////////////////////////////////////////////
////// Write the variables to Pressure file ///////
///////////////////////////////////////////////////

fat_write(file2, ",\r\n", 2);
sprintf(RTClock,"%i:%i:%i," , HOUR, MIN, SEC);
fat_write(file2, RTClock, strlen(RTClock));
sprintf (message2, "%8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f, %8.3f", p1, p2, p3, p4, p5, p6 ,p7, p8);
fat_write(file2, message2, strlen(message2));
fat_write(file4, message2, strlen(message2));
fat_close(file4);
fat_open(mmc,File_4, O_WRONLY, 0,0, &file4);
x=60000; // Set the 'x' variable to complete the sampling cycle to one second

////////////////////////////////////////////////////////////////////
/////////////// Check for fault signal again ///////////////////////
////////////////////////////////////////////////////////////////////

val=GetADC(channel);
voltage = (((float)val) * VREF)/ 1023.0;
k=0.5;
IOCLR1=0x00FF0000;
if (voltage <= k)
{
   goto FaultLost;
}
goto PublishWebSite;

///////////////////////////////////////////////////////////////////
///////// Publish the website for a while //////////////////////////
///////////////////////////////////////////////////////////////////

///////// according 'x' variable //////////////////////
PublishWebSite:
    n=0;        // Reset the 'n' to zero for next time
here:
    if (n<x)
    {
        Run_SOCKET();
        if (WDT_curSecond !SEC)
        {
            WDT_curSecond = SEC;
            if (++WDT_cntSecond >= WEBCAT_WDT_RESET)
            {
                ResetSoftWDT(WEBCAT_WDT_TIMEOUT);
                WDT_cntSecond =0;
            }
        }
        n=n+1;
        if (n<=x);
    {goto here;}
    }
    goto Cases;

FaultLost:
    fat_write(file1, "\n", 2);
    fat_write(file2, "\n", 2);
    sprintf(RTClock,"%i:%i:%i", HOUR, MIN, SEC);
    fat_write(file1, RTClock, strlen(RTClock));
    fat_write(file2, RTClock, strlen(RTClock));
    fat_write(file1, "\n", 2);
    fat_write(file2, "\n", 2);
    fat_write(file1, lostmsg, strlen(lostmsg));
    fat_write(file2, lostmsg, strlen(lostmsg));
    fat_write(file1, "\n", 2);
    fat_write(file2, "\n", 2);
    fat_close(file3);
    fat_open(mmc,File_3, O_WRONLY, 0,0, &file3);
    fat_close(file4);
    fat_open(mmc,File_4, O_WRONLY, 0,0, &file4);
CheckSignal:

val=GetADC(channel);
voltage = (((float)val) * VREF)/ 1023.0;
k=0.5;
IOCLR1=0x00FF0000;
if (voltage <= k)     ///// Check for fault signal again ////
{
    IOCLR1=0x00FF0000;
    Run_SOCKET();
    if (WDT_curSecond != SEC)      /// Keep publishing website
    {
        WDT_curSecond = SEC;
        if(++WDT_cntSecond >= WEBCAT_WDT_RESET)    /// Keep publishing website
        {
            ResetSoftWDT(WEBCAT_WDT_TIMEOUT);
            WDT_cntSecond =0;
        }
    }
    goto CheckSignal;       ///// If signal lost go back to check signal /////
}
goto CheckSampling;       ///// If signal presents go to check sampling rate /////

Cases:
m=m+1;
if (m==1)
    {goto tr1;}
if (m==2)
    {goto tr2;}
if (m==3)
    {goto tr3;}
if (m==4)
    {goto tr4;}
if (m==5)
    {goto tr5;
if (m==6) 
    {goto tr7;}
if (m==7) 
    {goto tr8;}
if (m==8) 
    {goto tr9;}
if (m==9) 
    {goto tr10;}
if (m==10) 
    {goto tr11;}
if (m==11) 
    {goto tr12;}
if (m==12) 
    {goto tr13;}
if (m==13) 
    {goto tr14;}
if (m==14) 
    {goto tr15;}
if (m==15) 
    {goto tr16;}
if (m==16) 
    {goto tr17;}
if (m==17) 
    {goto tr18;}
if (m==18) 
    {goto tr19;}
if (m==19) 
    {goto tr20;}
if (m==20) 
    {goto tr21;}
if (m==21) 
    {goto tr22;}
if (m==22) 
    {goto tr23;}
if (m==23) 
    {goto tr24;}
if (m==24) 
    {goto tr25;}
if (m==25) 
    {goto tr26;}
if (m==26) 
    {goto tr27;}
if (m==27) 
    {goto tr28;}
if (m==28) 
    {goto tr29;}
if (m==29) 
    {goto tr30;}
if (m==30)
    {goto tr31;}
if (m==31)
    {goto tr32;}
if (m==32)
    {goto tr33;}
if (m==33)
    {goto tr34;}
if (m==34)
    {goto tr35;}
if (m==35)
    {goto tr36;}
if (m==36)
    {goto tr37;}
if (m==37)
    {goto tr38;}
if (m==38)
    {goto tr39;}
if (m==39)
    {goto tr40;}
if (m==40)
    {goto tr41;}
if (m==41)
    {goto tr42;}
if (m==42)
    {goto tr43;}
if (m==43)
    {goto tr44;}
if (m==44)
    {goto tr45;}
if (m==45)
    {goto tr46;}
if (m==46)
    {goto tr47;}
if (m==47)
    {goto tr48;}
if (m==48)
    {goto tr49;}
if (m==49)
    {goto tr50;}
if (m==50)
    {goto tr51;}
if (m==51)
    {goto tr52;}
if (m==52)
    {goto tr53;}
if (m==53)
    {goto tr54;}

if (m==54)
    {goto tr55;}
if (m==55)
    {goto tr56;}
if (m==56)
    {goto tr57;}
if (m==57)
    {goto tr58;}
if (m==58)
    {goto tr581;}
if (m==59)
    {goto tr59;}
if (m==60)
    {goto tr60;}
if (m==61)
    {goto tr61;}
if (m==62)
    {goto tr63;}
if (m==63)
    {goto tr64;}
if (m==64)
    {goto tr65;}
if (m==65)
    {goto tr66;}
if (m==66)
    {goto tr67;}
if (m==67)
    {goto tr68;}
if (m==68)
    {goto tr69;}
if (m==69)
    {goto tr70;}
if (m==70)
    {goto tr71;}
if (m==71)
    {goto tr72;}
if (m==72)
    {goto tr73;}
if (m==73)
    {goto tr74;}
if (m==74)
    {goto tr75;}
if (m==75)
    {goto tr76;}
if (m==76)
    {goto tr77;}
if (m==77)
    {goto tr78;}

if (m==78) {
goto tr79;
}
if (m==79) {
goto tr80;
}
if (m==80) {
goto tr81;
}
if (m==81) {
goto tr82;
}
if (m==82) {
goto tr83;
}
if (m==83) {
goto tr84;
}
if (m==84) {
goto tr85;
}
if (m==85) {
goto tr86;
}
if (m==86) {
goto tr87;
}
if (m==87) {
goto tr88;
}
if (m==88) {
goto tr89;
}
if (m==89) {
    m=82;
    goto tr89;
}

Code of Main Web Page for Website:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<!-- HEAD SECTION -->
<head>
<title> Online DAQ from IPFW </title>
<link href="../style.css" type="text/css" rel="STYLESHEET">
<script language="JavaScript">
    <!--
    function FP_swapImg() {//v1.0
        var doc=document, args=arguments, elm,n; doc.$imgSwaps=new Array(); for(n=2; n<args.length; n+=2) { elm=FP_getObjectById(args[n]); if(elm) { doc.$imgSwaps[doc.$imgSwaps.length]=elm; elm.$src=elm.src; elm.src=args[n+1]; } }
    }
    function FP_preloadImgs() {//v1.0
```
```
var d=document,a=arguments; if(!d.FP_imgs) d.FP_imgs=new Array();
for(var i=0; i<a.length; i++) { d.FP_imgs[i]=new Image; d.FP_imgs[i].src=a[i]; }
}

function FP_getObjectByID(id,o) {//v1.0
var c,el,els,f,m,n; if(!o)o=document; if(o.getElementById) el=o.getElementById(id);
else if(o.layers) c=o.layers; else if(o.all) el=o.all[id]; if(el) return el;
if(o.id==id || o.name==id) return o; if(o.childNodes) c=o.childNodes; if(c)
for(n=0; n<c.length; n++) { el=FP_getObjectByID(id,c[n]); if(el) return el; }
f=o.forms; if(f) for(n=0; n<f.length; n++) { els=f[n].elements;
for(m=0; m<els.length; m++) { el=FP_getObjectByID(id,els[n]); if(el) return el; }
}
return null;
}
// -->
</script>
<script type="text/JavaScript">
<!--[if!v1.0]
function timedRefresh(timeoutPeriod) {
    setTimeout("location.reload(true);",timeoutPeriod);
}
// -->
</script>
</head>
<!-- INITIALIZE ADC VALUES SECTION -->
<script language="javascript">
    var adc0, adc1, adc2, adc3;
    adc0 = ^WCADC:0^;
    adc1 = ^WCADC:1^;
    adc2 = ^WCADC:2^;
    adc3 = ^WCADC:3^;
</script>
<!-- BODY SECTION -->
<BODY onload="FP_preloadImgs(/*url*/'button28.jpg', /*url*/'button29.jpg'); JavaScript:timedRefresh(15000)"
    onunload="FP_preloadImgs(/*url*/'button28.jpg', /*url*/'button29.jpg', /*url*/'button2C.jpg',
       /*url*/'button2D.jpg', /*url*/'button37.jpg', /*url*/'button38.jpg', /*url*/'button26.jpg',
       /*url*/'button30.jpg', /*url*/'button31.jpg', /*url*/'button32.jpg', /*url*/'button33.jpg',
       /*url*/'button34.jpg', /*url*/'button35.jpg', /*url*/'button39.jpg', /*url*/'button40.jpg',
       /*url*/'button41.jpg', /*url*/'button3B.jpg', /*url*/'button3C.jpg')"
DESIGNTIMESP="-1" designtimesp="-1">
    <P align="center"><font size="7">
        WaterFurnance &amp; IPFW Engineering</font></P>
    <P align="center"><font size="4">To Change the Sampling Rate Please Pick a Sampling Rate Combination from Below...</font></P>
    <P align="center" style="margin-top: 0; margin-bottom: 0"><a style="text-decoration: none"
        href="T1P1.asp"><IMG id=img1 onmouseup="FP_swapImg(0,0,/*id*/'img1',/*url*/'button28.jpg')"
        onmouseover="FP_swapImg(1,0,/*id*/'img1',/*url*/'button28.jpg')"
        onmouseout="FP_swapImg(0,0,/*id*/'img1',/*url*/'button27.jpg')"</a></P>
Click link to download Temperature file; <a href="WFTempData.txt">Temperature File</a>

Click link to download Pressure file; <a href="WFPresData.txt">Pressure File</a>

Current time and date is ^WCHOUR:^WCMIN:^WCSEC:^WCMONTH/^WCYEAR^ and the Most Recent Measurements are given below:
Temp#1
Temp#2
Temp#3
Temp#4
Temp#5
Temp#6
Temp#7
Temp#8
Temp#9
Temp#10
Temp#11
Temp#12

Press#1
Press#2
Press#3
Press#4
Press#5
Press#6
Press#7
Press#8

Press#7
Press#8
Example of Sampling Rate Change Web Page Code:

```html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>^#OUT=P0</title>
<script language="JavaScript">
<!--
function FP_swapImg() {//v1.0
  var doc=document,args=arguments,elm,n; doc.$imgSwaps=new Array(); for(n=2; n<args.length; n+=2) { elm=FP_getObjectByld(args[n]); if(elm) { doc.$imgSwaps[doc.$imgSwaps.length]=elm; elm.$src=elm.src; elm.src=args[n+1]; } }
}

function FP_preloadImgs() {//v1.0
  var d=document,a=arguments; if(!d.FPimgs) d.FPimgs=new Array();
  for(var i=0; i<a.length; i++) { d.FPimgs[i]=new Image; d.FPimgs[i].src=a[i]; }
}

function FP_getObjectByld(id,o) {//v1.0
  var c,el,els,f,m,n; if(!o) o=document; if(o.getElementById) el=o.getElementById(id);
  else if(o.layers) c=o.layers; else if(o.all) el=o.all[id]; if(!el) return null;
  if(o.id==id || o.name==id) return o; if(o.childNodes) c=o.childNodes; if(c)
  for(n=0; n<c.length; n++) { el=FP_getObjectByld(id,c[n]); if(el) return el; }
  f=o.forms; if(f) for(n=0; n<f.length; n++) { els=f[n].elements;
  for(m=0; m<els.length; m++) { el=FP_getObjectByld(id,els[n]); if(el) return el; }
  return null;
  }
  // -->
</script>
</head>
<body onload="FP_preloadImgs('url/*button24.jpg', 'url/*button25.jpg')">
<H1 align="center">SAMPLING RATES HAVE BEEN CHANGED SUCCESSFULLY</H1>
<p align="center">CURRENT RATES ARE; </p>
<p align="center">TEMPERATURE =<font size="6" color="#FF0000"> EVERY SECOND</font></p>
<p align="center">PRESSURE =<font color="#FF0000"> </font>
<font size="6" color="#FF0000">EVERY FIVE SECOND</font></p>
<p align="center">TO GO TO MAIN PAGE CLICK ON BUTTON BELOW...</p>
<p align="center"><a href="index.asp">
<img border="0" id="img1" src="button23.jpg" height="20" width="100" alt="MAIN PAGE"
onmouseover="FP_swapImg(1,0,/*id*/'img1',/*url*/'button24.jpg')"
onmouseout="FP_swapImg(0,0,/*id*/'img1',/*url*/'button23.jpg')"
onmousedown="FP_swapImg(1,0,/*id*/'img1',/*url*/'button25.jpg')"
onmouseup="FP_swapImg(0,0,/*id*/'img1',/*url*/'button24.jpg')" fp-style="fp-btn: Embossed Capsule 5" fp-title="MAIN PAGE"></a></p>
</body>
</html>
```
^#OUT=P0.19^ Configure P0.19 as OUTPUT
^#OUT=P0.21^ Configure P0.21 as OUTPUT
^#OUT=P0.22^ Configure P0.22 as OUTPUT
^#OUT=P0.23^ Configure P0.23 as OUTPUT

^#CLR=P0.19^ Falling ON #19
^#SET=P0.21^ RISING ON #21
^#CLR=P0.22^ Falling ON #22
^#CLR=P0.23^ Falling ON #23
//-->
</body>
</html>