# A Digital Implementation of Power System Metering and Protection

Andreas Joachim Schmitt

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Electrical Engineering

Jaime De La ReeLopez, Chair

Virgilio A. Centeno

Robert P. Broadwater

December 8th 2014

Blacksburg, Virginia

Keywords: Power Systems Communications, Power Systems Management, Substation Automation

A Digital Implementation of Power System Metering and Protection

Andreas Joachim Schmitt

## ABSTRACT

The work presents an entirely digital system which has several benefits as compared to the systems that are deployed currently. Utilizing digital capabilities to a much greater extent than is currently used within the power system allows for various improvements upon the current system. One such improvement is the ease of configuring and using the system. Each device can easily alter its functionality through a user interface, and the addition of devices is as easy as plugging it in. Additionally, the burden on the transformer due to the increase in the number of devices is nullified. The information remains accurate and unchanged, even when new devices are added to the system. The entire system conforms to the IEC 61850 standard, such that it adheres to the requirements of the actual power system.

# Acknowledgements

I would like to thank all of those who have given me support during my research. I would like to thank my parents and my brothers for everything they have given me, my friends for all of the help they have given me, and my professors for supporting me throughout the duration of my work.

# Table of Contents

# List of Figures

# List of Abbreviations

**GPS** - Global Positioning System

**ADC** - Analog-to-Digital Converter

**IEC** – International Electrotechnical Commission

**NMEA** – National Marine Electronics Association

**UART** – Universal Asynchronous Receiver/Transmitter

**Mbps** – Megabit per second

**UDP** – User Datagram Protocol

**DFT** – Discrete Fourier Transform

**GUI** – Graphical User Interface

# 1 Introduction

The use of digital components within the power system continues to increase rapidly. With this ever increasing number of digital devices, the capabilities of the system continue to grow along with it. However, there is a great hesitancy towards relying on an entirely digital system that is preventing digital capabilities from being utilized to their full extent. These concerns arise from a variety of different areas including the security and latency inherent to digital systems. In addition to this reluctance of adopting an entirely digital system, redundancy concerns are also holding back the capabilities of devices which are in operation today.

One of the major factors that is different when using a digital system is the presence of a digital latency in the system. Particularly, when discussing protection schemes, the devices need to act as quickly as possible. In an analog system, the measurement device is a physical component that moves and trips when something has occurred. However, in a digital system, data needs to be measured, recorded and compared, and after all of that is done a signal needs to be transmitted and received by the component which performs the physical protection. Each of these processes has an associated time that they take to perform, and the accumulation of these delays is a cause of concern. However, with technology improving drastically from when such schemes were first created, this latency can be insignificant if the system is designed properly.

Another area that can be drastically improved is the number of devices that are present or required in a system to be able to maintain all of the functionality that is desired. Currently, in the power system every scheme and functionality has its own associated device. Yet, each of these devices has the same hardware, a microprocessor. With each scheme requiring a separate device, in addition to having multiple devices for each scheme to ensure redundancy in the system, the number of devices in a typical power system is extremely large. If the devices are already comprised of microprocessors, why assign them only one specific task. Microprocessors are continuously growing in computational power, so why limit their

capabilities by keeping the same functionality as when they were first starting to be implemented. If every microprocessor handles multiple tasks, which is easily accomplishable with modern technology, the number of devices required in a system is much lower.

## 1.1 Project Overview and Hardware Profile

The project presented implements an entirely digital system which uses digital capabilities to a much greater extent in order to improve on current metering and protection systems. The concept behind this system is that data is gathered by one device and then broadcast to a network. Once there is data available on the network, any device that wants to use that data needs to simply be plugged in and it will have access to this data without a loss of accuracy. To demonstrate this functionality, an example device that is capable of multiple protection schemes is implemented to show the capability of the system. The overview of the system can be seen in Figure 1.1.

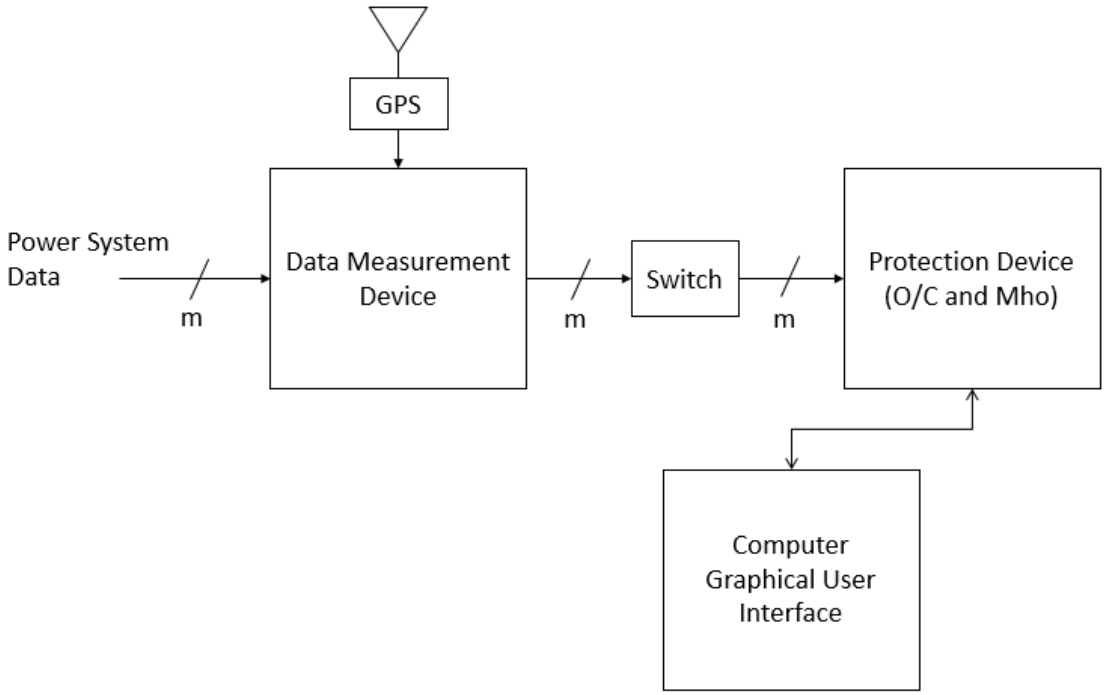


**Figure 1.1: Project Overview**

The system first takes data from the power system from sources such as transmission lines or buses. This is done by connecting a device to an instrument transformer which is connected to the power system. This device then converts this analog data from the power system to digital and assigns a timestamp to the data using a GPS time synchronized receiver to allow for comparison between points across the physically vast power system. Once the data has been completely converted, it is broadcasted to a communications network which conforms to the IEC 61850 standard. The communications network runs on the UDP protocol over a switched Ethernet system.

The second device, which is then connected to this network, is a protection device which is capable of doing, but is not limited to, any phasor based protection scheme. The two protection schemes that are implemented are an over-current protection scheme and a distance relay protection scheme. This device runs with an associated graphical user interface that configures the device and displays its outputs to increase the ease of usage. The methodology behind the functionality of these devices and the communication network are discussed in this work.

Before discussion of the project inner workings can be done, the hardware specifications need to be detailed. The specific hardware was selected for a variety of reasons, including but not limited to, cost, availability, and most importantly functionality. The most important hardware component is the microprocessor since that is the basis of the project. All other components were selected in consideration of the selected microprocessor.

The board that was selected is the Cerebot 32MX7$^{TM}$ Board made by Digilent, Inc. which can be seen below in Figure 1.2 [2]. The board comes with a PIC32MX795F512L processor.

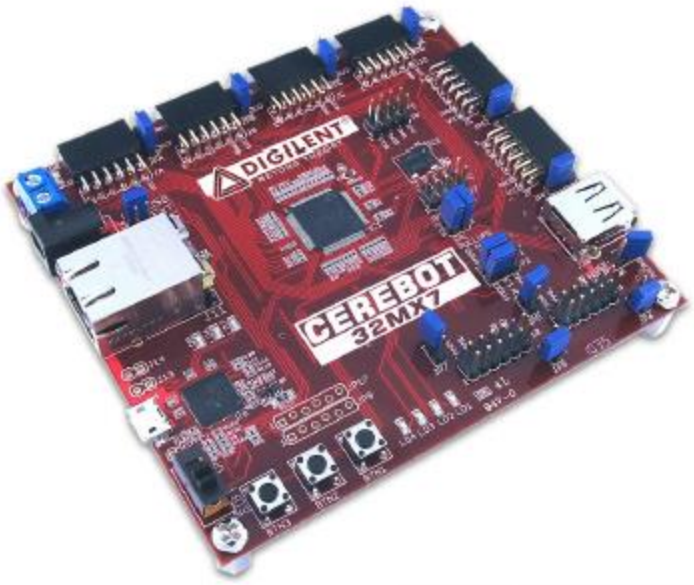


**Figure 1.2: Cerebot 32MX7 Board,**
**http://www.digilentinc.com/Products/Detail.cfm?Prod=CEREBOT32MX7, Used under fair use, 2014.**

The important specifications of this board are as follows:

- 32 Bit, 80 MHz Microprocessor
- 8 MHz External Oscillator
- 512 KB Internal Program Flash Memory
- 128 KB Internal SRAM Memory
- 10 bit Successive Approximation ADC
- 10/100 Mbps Ethernet Connector
- 6 Peripheral Connectors

While these some of these specifications have low capabilities, the board meets all of the requirements for this specific project. However, the processor speed can be an issue due to the computational requirements of some of the algorithms. If higher computational power is desired, then a different board would have to be used.

The GPS device that was used in conjunction with the microprocessor is the Digilent PmodGPS™ [3]. The PmodGPS comes equipped with a GlobalTop Gms-u1LP GPS antenna.

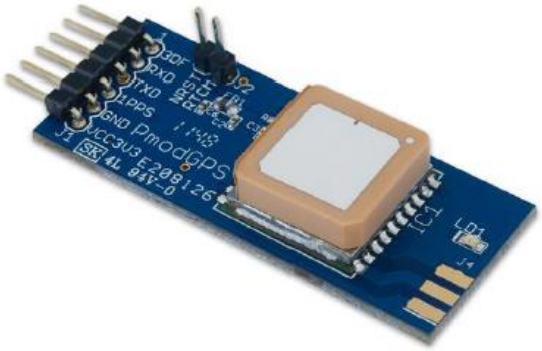


**Figure 1.3: Digilent PmodGPS, https://www.digilentinc.com/Products/Detail.cfm?NavPath=2,401,1038&Prod=PMOD-GPS, Used under fair use, 2014.**

Other equipment that was used in various places are as follows:

- TP-LINK TL-SF1008D 8-port 10/100Mbps Desktop Switch
- Several Cat6 Ethernet Cables
- Passive Circuit Components(resistors/capacitors/wires)
- LM324n Operational Amplifier

# 2 Data Measurement Device

The first requirement of this system is to measure values. In the power system this means connecting a transformer to desired locations so that it is both safe and possible to measure the values. The values in the power system are extremely high, which makes it difficult for measurement devices to read these values. Therefore, they need to be brought down with an instrument transformer to levels suitable for measurement. Many of the inaccuracies that are present while measuring quantities occur in these transformers. These transformers have an associated error with them which can be very large when compared to other components of the measuring process.

One of the causes that is a source of error is the burden associated with loading the transformers. The purpose of the transformer is to allow for measurement to be possible; however, in order to measure a transmission line, the transformer needs to be directly connected to it. This connection creates another path for the current to flow which can be seen as a parallel impedance in the circuit as depicted in Figure 2.1. Therefore, the impedance at the secondary of the transformer must be as large as possible to not influence the values in the line.

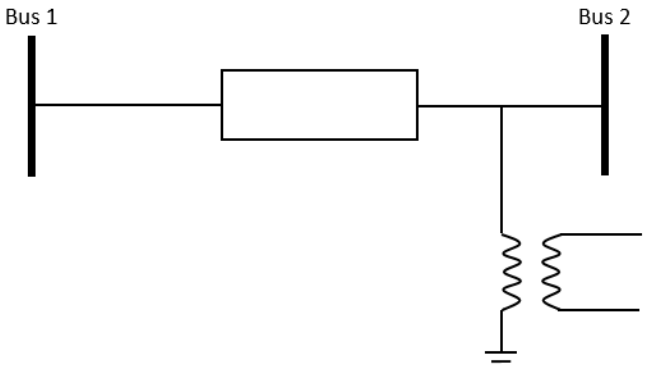


**Figure 2.1: Transformer Connection**

The burden on a transformer decreases the secondary impedance when more devices are added to the system. Every additional device attached to the transformer will decrease the

impedance of the transformer, which will decrease the accuracy of measurement. The load decreases with each device since devices are added in parallel when connected to the transformer as shown in Figure 2.2.

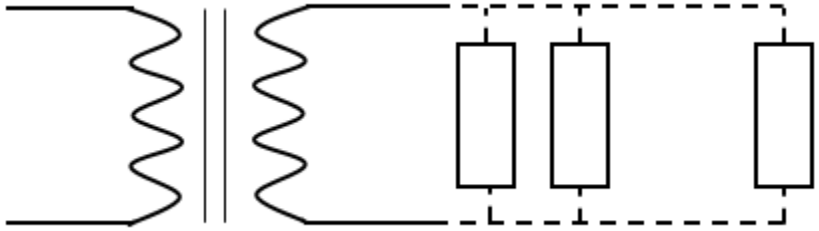

**Figure 2.2: Loading Effect**

However, as technology keeps developing more and more of these loads are using digital components. This means that each device connected to the transformer has its own analog to digital conversion processes. If each device is using digital data anyway, why not simply connect one device that measures the data, converts it to digital and then broadcasts the data for any device to use. There would now be only a single device attached to the transformer rather than multiple, which would greatly mitigate the burden associated with the load on the transformer.

This is precisely what the first device of this system does. Specifically, there are two major functions that this device must accomplish before the data can be broadcast. The first requirement is that the data must be converted to digital quantities so that it is even possible to send the data to other devices over a communications network. The other is to assign a synchronized timestamp to the data, such that comparison between multiple devices and their data is possible. Once these two processes have been done, the data is able to be sent and used by other devices in the system.

## 2.1 Analog-to-Digital Conversion

The Digilent Cerebot 32MX7 board comes with a 10 bit analog to digital converter. The input voltage range of this ADC can be selected; however, it has a maximum of 0-5 volts, with a

default of 0-3.3 volts. The maximum conversion rate of this ADC is 1 Msps [2]. Since extra voltage levels are required to be input to select a different input voltage range, the default range of 0-3.3 volts was used. The voltage range and number of bits are what determine accuracy of the ADC.

The major source of error during the analog to digital conversion process is the quantization error. Since there are only a set of specific, quantized digital outputs, there are small ranges for analog inputs that will have the same output. The analog input can only be approximated to the closest quantization level. Quantization and the possible errors associated with it are depicted in Figure 2.3 below.

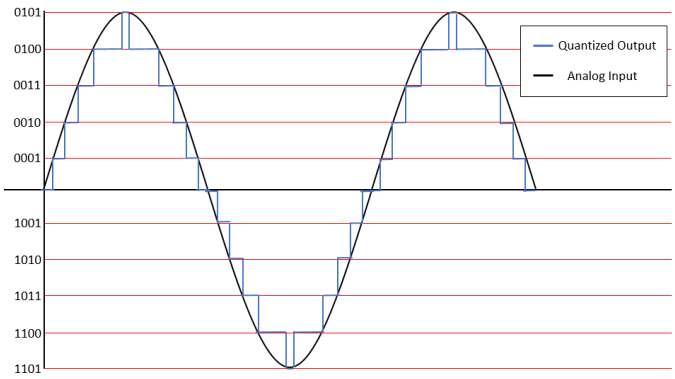


**Figure 2.3: Quantization Plot**

It can be seen from this graph that with an increase of quantization levels comes a reduction in the quantization error. The total number of quantization levels in a specific range determines the maximum possible error in a data point. The number of quantization levels is determined by the number of bits. Higher voltage ranges will require more quantization levels to maintain the same amount of accuracy. Accuracy can be thought of in terms of the resolution of the ADC. The resolution is the maximum deviation of input voltage that will still give the same digital output. The resolution of an ADC is calculated with the following equation:

$$\boldsymbol{Resolution(volts): Q = \frac{V_{MAX} - V_{MIN}}{2^N - 1}}$$

In this equation, N is equal to the number of bits. Since the ADC of the Cerebot 32MX7 is a 10 bit ADC with a voltage range of 0-3.3 volts, the resolution of the ADC is:

$$Q = \frac{V_{MAX} - V_{MIN}}{2^N - 1} = \frac{3.3 - 0}{2^{10} - 1} = \frac{3.3}{1023} = 0.00322\ Volts$$

To give a better perspective, if the input to the ADC was a 550 kV line there could be a difference of $537.11$ volts with no difference in the output value. While this error is small ($.1\%$) when operating at the maximum voltage range, it can have a significant impact if the ADC is receiving input values much lower than the maximum range. This becomes an issue when trying to do protection. Metering devices can be configured to have the rated values be roughly equal to the maximum range, yet the values that can be seen under fault conditions can vary significantly when compared to normal operating conditions. Therefore, data need to be configured separately depending on the purpose that it will ultimately be used for. Data used for protection purposes will have a much smaller input voltage range, causing larger inaccuracies during normal operating conditions.

### 2.1.1 Sampling

The process of analog to digital conversion is done through sampling. Sampling, as the name suggests, is the process of taking samples of an input signal and recording that value as a digital output. There are various methods for sampling including successive approximation, integration, flash, and sigma delta. The sampling method that is used by the ADC on the Cerebot 32MX7 board is successive approximation [4].

Successive approximation is the most common analog to digital conversion technique for mid-to-high resolution ADCs [5]. This method searches through all possible discrete outputs using a binary search algorithm until the best matching case is found. This works by checking the median value of the possible outputs. If the input is larger than that value, values lower than the median are disregarded and vice versa. The ADC then checks the median case again of the new range of possible outputs. If it is lower, then the higher values are discarded and vice versa. This process is repeated until the closest matching value to the input value is obtained.

When sampling an input signal, the issue of aliasing is always present. Aliasing is the distortion of the original signal due to the overlap of the original frequency spectrum with the spectrum of the images created through sampling. Assuming an input signal with the frequency spectrum shown in Figure 2.4.

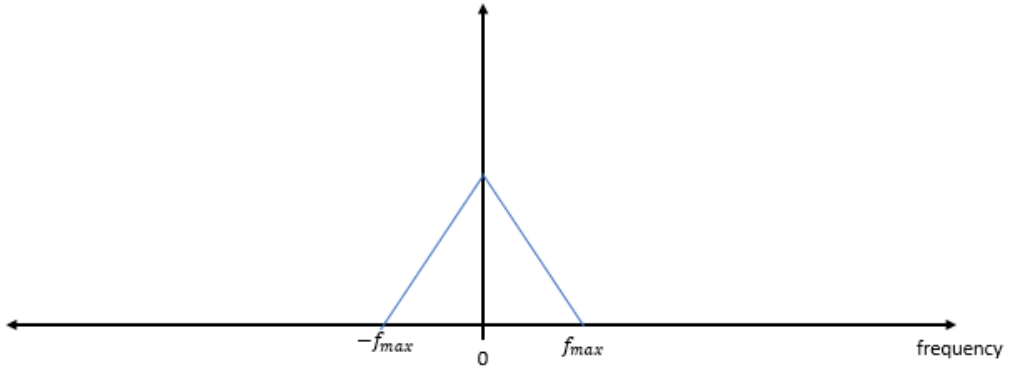


**Figure 2.4: Frequency Spectrum of an Input Signal**

The processes of digitizing the signal then causes a repetition of the frequency spectrum at multiples of the sampling frequency creating images of the original spectrum as shown in Figure 2.5.

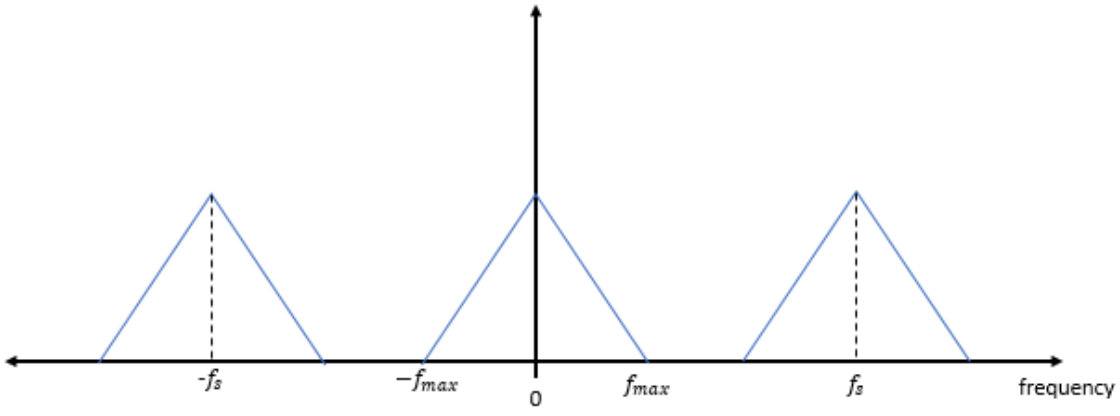


**Figure 2.5: Frequency Spectrum of a Sampled Signal**

This repetition of the frequency spectrum can cause issues if the system is not designed properly. It can be seen that if the sampling frequency is too small, the frequency spectrums will overlap causing a distortion of the initial signal as shown in Figure 2.6.

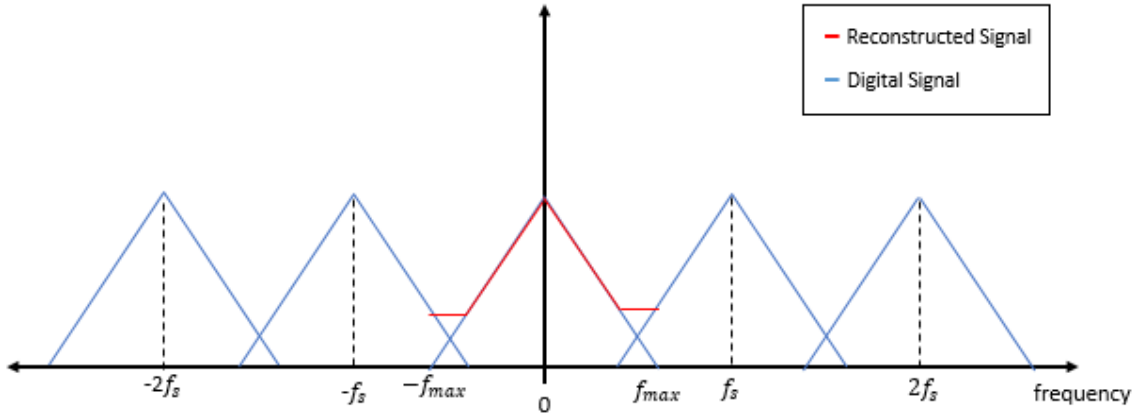


**Figure 2.6: Aliasing Example**

This overlap of the frequency spectrums is called aliasing. There are two things that are done to prevent aliasing. The first is to ensure that the sampling frequency is high enough. The Nyquist Sampling Theorem states that to ensure that aliasing does not occur, the sampling frequency must be greater than twice the highest frequency present in the signal. The second is to place what is called an anti-aliasing filter on the input signal. This filter is a low-pass filter which has a cut-off frequency at the maximum frequency of the signal. This will prevent any distortions due to the presence of high frequency signals created by sampling. Since the sampling rate of this system is set to 720 samples per second, the highest frequency signals that can be measured are 360 Hz and the highest the cut-off frequency of the anti-aliasing filter can be is also 360 Hz. This will allow for the measurement of the fundamental frequency (60 Hz), up to the $6^{th}$ Harmonic accurately.

Another factor which effects the quality of the sampling is the accuracy of the oscillator used. When a sampling rate of 720 Hz is stated, this assumes that the samples are at equal intervals, however due to oscillator frequency drift that may not be the case. As oscillators grow in age, the frequency at which they operate slowly starts to drift from the initial frequency. Additionally, the ADC is triggered after a specific number of oscillations has been

reached, but the oscillator may be slow or fast which means this time is not always accurate. The oscillator of this board has a possible error of ±100 parts per million. This means that for every million oscillations, there is a possible error of 100. While this may seem to be an insignificant error, over a long period of time it will accrue and cause major errors. This is fixed through the use of GPS time synchronized pulse from the GPS receiver, which is discussed later. Every time this pulse is received, the device software will force the sampling to re-align itself such that the accumulated oscillator offset, due to both the inaccuracies and frequency drift, is reset.

### 2.1.2 Front-End Circuit

There are several functions that the front-end circuit of this system must be able to do. Besides containing the anti-aliasing filter which was discussed previously, the front end circuit needs to convert the signal to match the range of the ADC while at the same time ensuring that the microprocessor is not damaged. Since the ADC only accepts inputs between 0 and 3.3 volts, the input needs first be brought to match the ADC voltage range. Typically, an extra step-down transformer needs to be at the front of the circuit which brings the voltage down to 3.3 volts peak-to-peak. Once the value has the correct voltage range of 3.3 volts peak-to-peak, it needs to be given a DC offset of 1.65 volts so that the range is between 0 and 3.3 volts.

The next component of the circuit is the anti-aliasing filter. This filter is a low-pass filter with a cut-off frequency of 360 Hz. For this project, a third-order active Butterworth filter was used. This design was chosen to have a flat magnitude pass-band with a steep drop-off in order to reduce high frequency interference. The choice of using an active filter was due to the fact that active filters have a very high input impedance which will reduce the loading effect on the transformer, and the practicality of using an operational amplifier over purely passive components.

The final component is the protection of the microprocessor. The microprocessor can only handle values between 0 and 5 volts before damage is done to the circuit [4]. Therefore, before the signal is input to the microprocessor, it must be ensured that the voltage is within this range. The entire front end circuit and the associated equations can be seen in Figure 2.7.

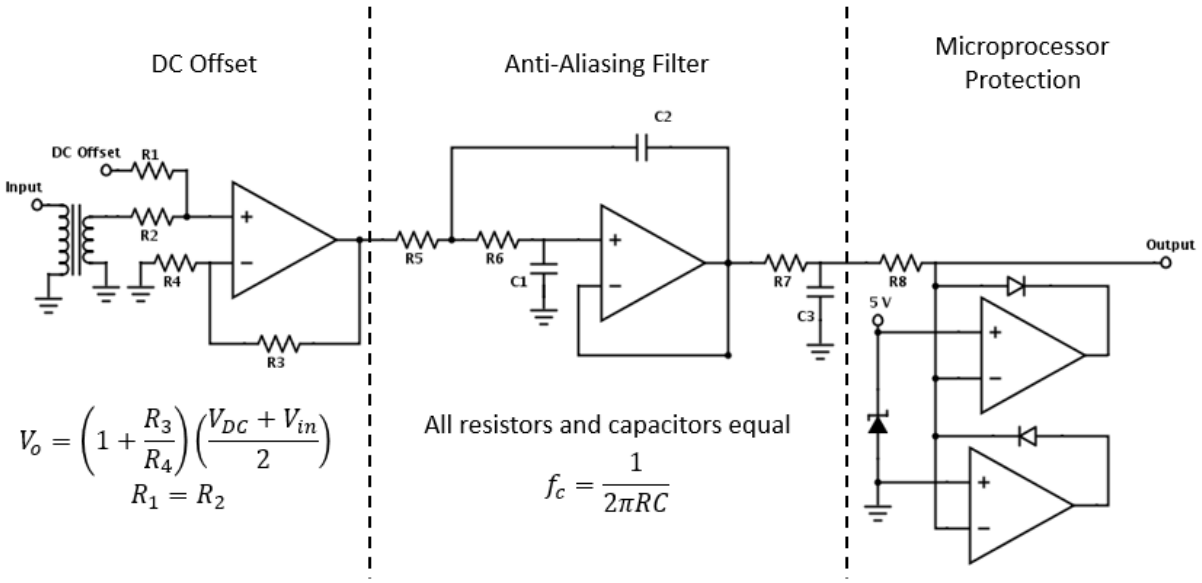


**Figure 2.7: Front-End Circuit**

The magnitude and phase response of this circuit is shown in Figure 2.8. From this plot it can be seen that there is an error associated with the front end circuit. While the magnitude response is constant through the pass band, and more importantly at 60 Hz, the phase has an offset.

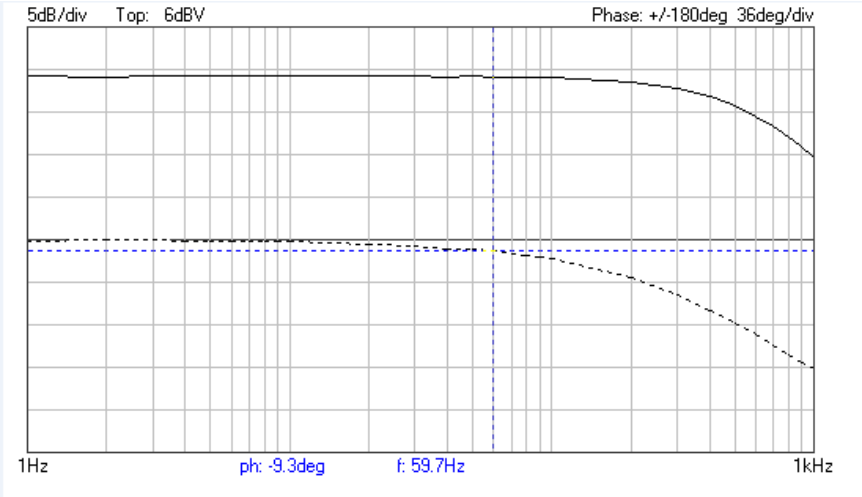


**Figure 2.8: Magnitude and Phase Response**

Specifically, at 60 Hz the phase has a constant offset of about -9.5°. However, this is to be expected for the project specifications given. With a sampling rate of 720 Hz, the highest the cut-off frequency can be is 360 Hz. In a Butterworth filter, the phase will start to change a decade before the cut-off frequency. Thus, given a cut-off frequency of 360 Hz, the phase will start changing at 36 Hz, which means there will be a difference at 60 Hz. While there is an error associated with this filter, it is a constant error. This means that the error in both phase and magnitude can easily be accounted for, making this not a big issue for the system.

There are a few ways to account for this error. If higher computational power microprocessors are available, the sampling frequency can be increased. This has many benefits, one of which is that the phase begins to decay after 60 Hz. Other solutions would be phase shift the value back to the original value. This can be complicated and will cause errors if not careful, but it is possible. In the current system, the phase offset is only accounted for at 60 Hz since this is the fundamental frequency.

## 2.2 GPS Time Synchronization

When working with the power system, the physical size of the system is always an issue. It is not easy to coordinate between locations and comparing data becomes more difficult. This is due to both the local differences at each locations, and the time it takes to send data between locations. Therefore it is required that data be time-stamped for it to be useful. This is done through the use of a GPS receiver. The GPS receiver has two functionalities to it. The first is that it receives a 1 pulse-per-second signal to ensure that any device receiving a GPS signal is synchronized with all other devices that operate using a GPS signal. The second is to receive the actual time data.

The pulse-per-second signal on the Digilent PmodGPS™ functions as shown in Figure 2.9 [3].

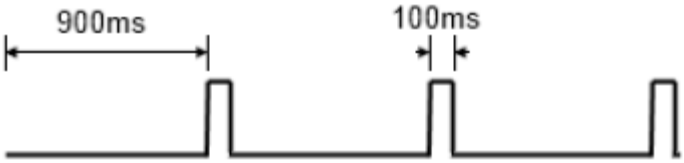


**Figure 2.9: Pulse per Second Diagram, http://www.digilentinc.com/Data/Products/PMOD-GPS/PmodGPS_rm.pdf, Used under fair use, 2014.**

As can be seen, the pulse stays low for 900ms and high for 100ms. This is to ensure that devices are given enough time to recognize that the pulse has gone high since there are potential delays in all systems. The time data is recorded using the NMEA protocols and then sent serially to the Digilent 32MX7 board using a UART communication system where it is decoded. Upon reception of the pulse, the new time data is updated. This means that the time data is only updated once per second, however this is not an issue. Each data point that is sent out by this device will have both the time stamp each second, and a counter indicating how many data points have been sent since the last second so that each data point has a precise time-stamp associated with it.

# 3 Communication System

The IEC 61850 standard details the communication network requirements of a network within the power system. The IEC 61850 is very extensive in its requirements; however, the important points are as follows [1]:

- Provide Description of Capabilities of Device
- High-Speed Inter-Device Communication(Guaranteed Delivery Times)
- High Availability
- Support for Voltage/Current sampled data and File Transfers
- Configurability
- Independent of Specific Vendors
- Security

In order to meet the high speed and availability requirements, the IEC selected switched Ethernet to be the standard for all communication systems. Ethernet transmissions have a very high transmission rate, which allows for large amounts of data to be sent quickly. This then allows for a variety of possibilities in what information can be sent over the network so that all of the other key points can be met.

The issue with Ethernet is that in order for devices to be practical and useful, transmission protocols need to be used. Protocols define how the data is transmitted and received and there are a wide variety that function using Ethernet. While it may seem easy to simply chose one of these protocols for the standard, this presents many issues. The whole reason for the different protocols is that they all have different benefits and consequences. Some focus on the speed of data transmission, while others may focus on the success rate of the data transmission and more. This means that to limit the communication to a specific protocol may limit the capabilities of certain systems. Along the same lines, legacy devices were made using the protocol that suited them. If the new standard were to define a specific

protocol, either all devices with a differing protocol would need to be changed or they would become unusable.

As a result, the transmission architecture of the IEC 61850 focuses on “abstracting” the device capabilities and data such that it is independent of underlying protocols [1]. This abstracting is done through the creation of a data object which can then be mapped to any existing protocols. The object name defined by the 61850 standard can be seen in Figure 3.1.

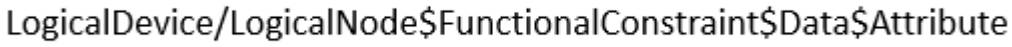


**Figure 3.1: IEC 61850 Object Name**

The defined object name contains all of the information that a device needs to communicate with other devices. The LogicalDevice/LogicalNode information that is sent is the type of device that is communicating, such as a relay or a synchrophasor and the physical location within the power system associated with the data. The $FunctionalConstraint defines the attributes of the data such as status attributes, description attributes, or others. The $Data is the actual data being sent by the device. The $Attribute is the actual value of the functional constraint. Existing protocols can then be used as long as they send the specified object name, which is also down by the presented system.

Outside of defining the specific transmission protocol, the 61850 standard also states testing procedures to ensure that devices conform to the standard and requirements for devices to contain information detailing their configuration.

## 3.1 Project Network

Once the data is in the correct form and has an associated time stamp, it needs to be broadcast out to a network. The IEC 61850 standard states that this communications system should be a switched Ethernet network [1]. As the name suggests, a switched Ethernet network makes use of a switch that each device will connect to via an Ethernet connection. The switch

acts to facilitate the transmission between multiple devices. A typical example of a switched Ethernet network can be seen in Figure 3.2.

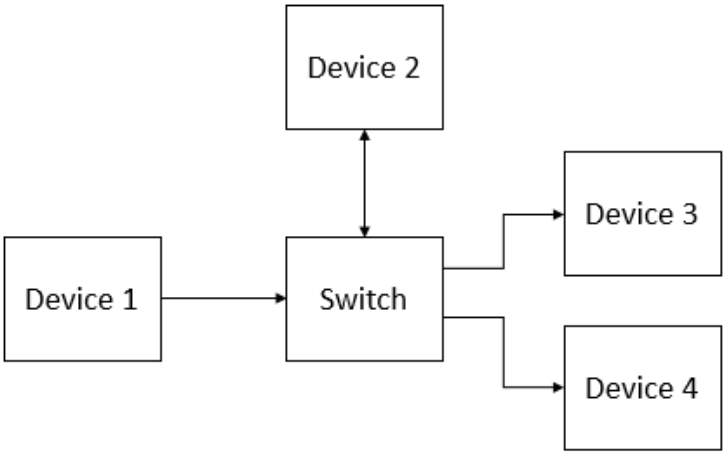


**Figure 3.2: Switched Ethernet Network**

There are several sources of latency inherent to a switched Ethernet network [6]. The first source of latency in this network is the store and forward latency. This latency comes from the switch storing the transmission packet completely before sending it out again. The store and forward latency is proportional to the packet size and inversely proportional to the bit rate of the network.

$$L_{SF} = \frac{frame\ size}{bit\ rate}$$

While this delay can be high for large packet sizes, the amount of data being sent in this system will always be less than the minimum packet size. Using a 100 Mbps Ethernet connection and assuming the minimum packet size of 64 Bytes, the store and forward latency will be 0.5 microseconds.

The next two sources of latency are due to the materials in the network. The switch fabric is the material that implements the store and forward functionality and will have an associated latency, typically in the order of several microseconds. The other source of latency is the actual transmission of the data across the line. This latency can become very large if the

distance that the data needs to travel is very large. However, the distances within a local area network, which this system is, are so small that this delay is negligible.

The last source of latency in a switched Ethernet network is the queueing latency. Unlike the other sources of latency, this one is less certain and constant. To prevent collision between multiple transmitting packets, the switch will queue the packets for the store and forward function. This delay can become significant when the network is heavily loaded, but is not typically an issue in a lightly loaded system. Every device added to the system will increase this load by increasing the amount of packets present in the network; thus, care needs to be taken when considering adding additional devices.

In addition to the delay of the switched Ethernet configuration, the transmission protocol will have an inherent latency. When constructing a system like this for use in the power system, speed is a big issue. Devices that perform protection schemes need to operate as quickly as possible such that extreme consequences are prevented. Therefore, when selecting the transmission protocol of the system, the deciding factor will be its associated latency. Naturally, the quickest transmission method would be to use raw Ethernet since that has the lowest amount of overhead; however, this is impractical for real use since it severely limits what devices are capable of being used, since each device would have to be made specifically for this system.

However, UDP/IP is not significantly slower than raw Ethernet. In a single switch network, the difference in latency between raw Ethernet and UDP is in the order of microseconds [7]. UDP is a connectionless transmission system, meaning that the data pertaining to the transmission and routing of the information is contained within the packet. There are no handshaking mechanisms in UDP, which means that there is no guarantee that the data arrives. However, data quality is checked with checksums. Other protocols, such as TCP require that an acknowledge signal is sent on each successful transmission forcing the system to wait for a successful transmission before then next packet it sent. This will cause the entire

system to be slower than had this not been done. By foregoing this, the overhead of the transmission in a UDP system is minimized to allow for low latency transmissions. In this specific system, data is sent quickly enough and the significance of an individual data point is low. Thus, the possibility of missing data points is not an issue while the speed provided by UDP makes it ideal. Additionally, each data point is sent twice simultaneously to significantly reduce the possibility of a missed data point.

# 4 Protection Device

Now that there is a functional communications network that is constantly being updated with new data, any device that wants to receive this data is able to by simply connecting to the network. What is actually done with the data will depend on the configuration of the device and what its intended purposes are. In power systems, there are various applications for which this data can be used for. One such application is for protection purposes. There are many different protection schemes that account for a variety of different system conditions.

A very large portion of these schemes use phasors to determine how to respond to specific conditions. This means that once a phasor value has been obtained, any of these protection schemes can be deployed. Currently in the field, each protection scheme will have a dedicated device. Redundancy then requires that there be multiple of each of these devices. With many different protection schemes, each deploying multiple devices across the extremely vast number of buses within the power system, this number of devices becomes extremely large. This may have been a necessity in the past when processors were much slower, but with modern day processors this is simply a waste of processing power. Given that a phasor has already been calculated, why not have a device that can do multiple protection schemes?

This would solve many problems, including the availability of certain protection schemes in certain locations, the cost of installing new schemes and the total number of devices in the system. Hidden failures would be much easier to diagnose. This device implements several phasor based protection schemes; therefore, before anything can be done with the data, the phasor needs to be calculated. A phasor represents the signal with both a magnitude and phase angle.

Instead of propagating forward in time, a sinusoid can be thought of as rotating around in a circle at a rate of the frequency of the signal in radians per second as shown in Figure 4.1.

The magnitude of the phasor representing this signal is the radius of this circle and the phase angle is the angle of the phasor when time is equal to zero. In order to be able to compare phasors, there needs to be a reference for the phase angle. This reference for phase angles is defined with the cosine function, such that $\cos(0) = 0°$ and $\cos\left(\frac{\pi}{2}\right) = 90°$. Therefore, given an input signal of $Xcos(\omega t + \phi)$, the phasor would be $X\angle\phi$.

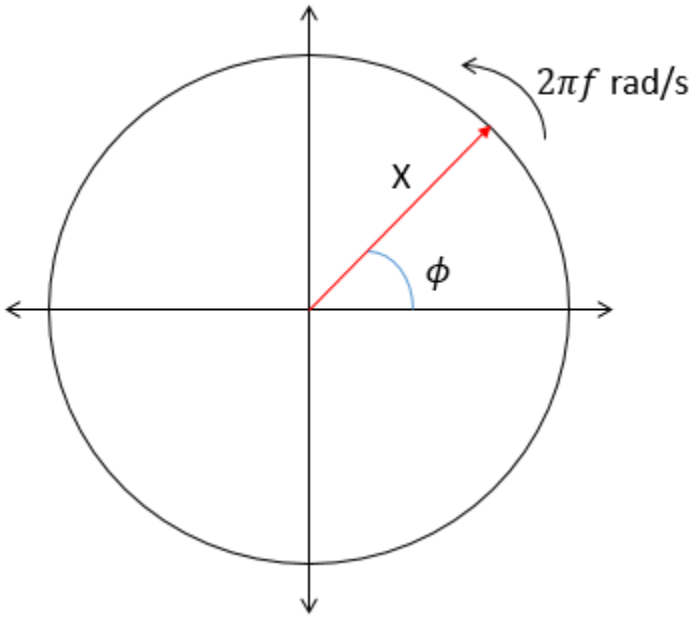


**Figure 4.1: Rotating Phasor Diagram**

## 4.1 Phasor Calculation

The method in which phasors are typically calculated is the Fourier transform. Every periodic time domain signal can be said to be made up of sinusoids of varying magnitude and frequency. The Fourier transform works by taking these periodic time domain signals and converting them into an equivalent frequency domain signal, which is represented as a spectrum of the magnitudes of present sinusoids using the equation shown below.

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi tf}\,dt$$

Each frequency in this spectrum then has an associated phasor. In an ideal power system, the signal on each line is a single sinusoid at the fundamental frequency, yet harmonics are present in the system for a wide range of reasons. While these harmonics will have an

impact on the system, the fundamental frequency and its associated phasor give the most information about that state of the system. In the United States, the ideal operating frequency of the power system is at 60 Hz.

### 4.1.1 DFT and recursive phasor calculations

The calculation of a phasor given sampled input data makes use of the discrete Fourier transform. As the name suggests, the DFT operates using discrete quantities rather than a continuous function to calculate the Fourier transform. This is accomplished by taking a certain number of data points and then calculating the DFT on that set of data. The number of data points in each calculation is determined by the window size and the sampling rate of the system. Typically, the window size of each calculation is a multiple of cycles that the sinusoid has gone through. An example which uses a window size of one cycle is shown below in Figure 4.2. The presented system computes the DFT with a one cycle window size.

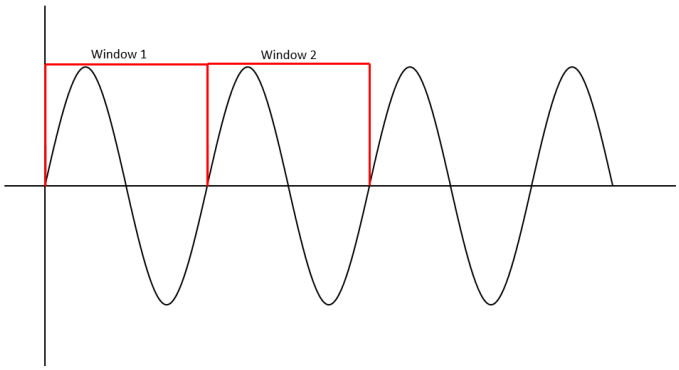


**Figure 4.2: One-Cycle DFT Window**

Given a specific window size of N data points, the DFT function for the fundamental frequency is as follows:

$$X\angle\phi = \frac{2}{N}\sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi n}{N}}$$

Using a sampling rate of 720 Hz, given that the frequency of the signal is 60 Hz, a window size of one cycle will have 12 data points. This makes the equation for this system as follows:

$$X\angle\phi = \frac{2}{12}\sum_{n=0}^{11} x(n)e^{-j\frac{2\pi n}{12}}$$

However, since the microprocessor cannot handle imaginary numbers this equation needs to be altered. This can be done through the use of Euler's formula which states that:

$$e^{jx} = \cos(x) + jsin(x)$$

Substituting that into equation DFT equation, the result is:

$$X\angle\phi = \frac{2}{12}\sum_{n=0}^{11} x(n)\left[\cos\left(\frac{2\pi n}{12}\right) - jsin\left(\frac{2\pi n}{12}\right)\right]$$

This equation can then be split into a real and imaginary component.

$$A = \frac{2}{12}\sum_{n=0}^{11} x(n)\cos\left(\frac{2\pi n}{12}\right)$$

$$B = -\frac{2}{12}\sum_{n=0}^{11} x(n)\sin\left(\frac{2\pi n}{12}\right)$$

With these two components, the phasor then can be found as:

$$X = \sqrt{A^2 + B^2}$$

$$\phi = \tan^{-1}\left(\frac{B}{A}\right)$$

However, care needs to be taken when calling an inverse tangent function on a microprocessor. The function does not take the quadrant of A and B into consideration when executing the arctangent function. For example, if both values of A and B are negative, the arctangent will be computed on a positive value. The microprocessor will then determine that the angle is in the first quadrant as if both A and B are positive, yet this is not the case. Therefore, the quadrature of the phasor needs to be checked and accounted for so that errors do not arise.

While this function is appropriate for the calculation of a single phasor, it is not ideal for the system. Since this device is being used for protection, values need to be checked as often and quickly as possible. That means that for each new sample received, a new phasor needs to be calculated. However, the DFT is a very computational intensive function; therefore, the

microprocessor used in this system cannot handle calling the function that often. In order to alleviate the strain on the processor, a recursive function is used. Given a sliding window as depicted in Figure 4.3, the phasor can be found in each new window based off of the previous window [8].

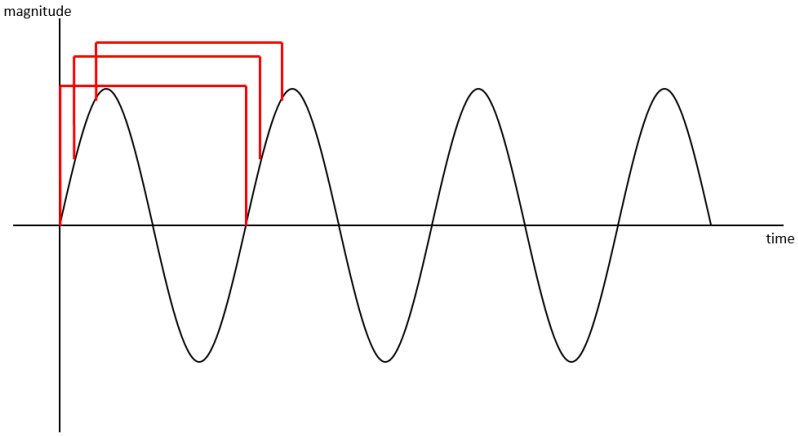


**Figure 4.3: Sliding Data Window**

The equation that governs the recursive function is as follows:

$$X(r+1) = \big(x(N+r+1) - x(r+1)\big)e^{-j\frac{2\pi}{N}(r-1)} + X(r)$$

Where:

$$N = number\ of\ data\ points\ in\ window$$

$$r = the\ number\ of\ phasor\ being\ calculated$$

$$X = phasor$$

$$x(n) = n^{th}\ data\ point$$

For example:

$$X(0) = the\ initial\ phasor\ calculated\ using\ the\ DFT$$

Therefore:

$$X(1) = \big(x(13) - x(1)\big)e^{-j\left(\frac{2\pi}{12}\right)(0)} + X(0)$$

$$X(2) = \big(x(14) - x(2)\big)e^{-j\left(\frac{2\pi}{12}\right)(1)} + X(1)$$

Another way to think about this equation is that it takes the difference in values between the equivalent data points of subsequent windows and combines that with the

previously found phasor. Naturally, the $e^{-j\frac{2\pi}{N}(r-1)}$ needs to be split into cosine and sine terms just like previously so that the microprocessor can actually compute the function. This is done as follows:

$$A_{new} = \big(x(N+r+1) - x(r+1)\big)\cos\left(\frac{2\pi}{N}(r-1)\right) + A_{old}$$

$$B_{new} = -\big(x(N+r+1) - x(r+1)\big)\sin\left(\frac{2\pi}{N}(r-1)\right) + B_{old}$$

The phasor is then found again just like before using the real and imaginary term. Each new phasor now only requires the calling of one sine and one cosine function rather than twelve of each. This may not seem like that important of a process, however if the sampling rate ever increases the recursive function will always only take one cosine and one sine function while the normal DFT will continue to increase the number of calls to these functions. Additionally, to avoid possible persistent errors with subsequent calls of the recursive function for long periods of time, the protection device will recalculate the DFT every second so that the initial phasor for the recursive function resets and is accurate.

### 4.1.2 Off-Nominal Frequency Calculations

These equations however assume that the system frequency is a constant 60 Hz which is not the case in the real world. During normal operations, system frequency is allowed to deviate slightly from 60 Hz, but controls prevent a large deviation from nominal outside of collapse events. This becomes an issue when calculating the phasor since the number of bits in a window, N, is determined by the sampling rate and the signal frequency.

$$N = \frac{f_{sample}}{f_{sys}}$$

At the operating conditions of 60 Hz and a sampling rate of 720 Hz, N becomes an integer of 12. However, if the system frequency is now slightly off 60 Hz, N will not be an integer. While the phasor calculations will still function with N as a non-integer, the error will increase proportionally to the deviation from the normal N. One method to combat this error is to track the frequency and then resample the data, such that N is always an integer.

Another method to account for the differences caused by an off-nominal frequency uses a 4-parameter taylor series [9]. This method uses a one-cycle, boxcar window which makes it ideal for the presented system. This algorithm uses the currently calculated phasor and the previously calculated phasor, to correct the phasor for off-nominal frequencies. The final equation that is presented from this paper, and which is used to account for off-nominal frequencies is:

$$X_{true}(r) = X(r) - j\frac{X(r) - X(r-1)}{2Nsin\left(\frac{2\pi}{N}\right)}$$

This equation needs to be split into real and imaginary again in order for the microprocessor to be able to process the calculations.

$$A_{Final} + jB_{Final} = (A_r + jB_r) - j\frac{A_r + jB_r - A_{r-1} - jB_{r-1}}{2Nsin\left(\frac{2\pi}{N}\right)}$$

$$A_{Final} + jB_{Final} = \left[A_r + \left(\frac{B_r}{2Nsin\left(\frac{2\pi}{N}\right)}\right) - \left(\frac{B_{r-1}}{2Nsin\left(\frac{2\pi}{N}\right)}\right)\right] + j\left[B_r - \left(\frac{A_r}{2Nsin\left(\frac{2\pi}{N}\right)}\right) + \left(\frac{A_{r-1}}{2Nsin\left(\frac{2\pi}{N}\right)}\right)\right]$$

$$A_{Final} = A_r + \frac{B_r - B_{r-1}}{2Nsin\left(\frac{2\pi}{N}\right)}$$

$$\mathrm{B_{Final}} = \mathrm{B_r} + \frac{\mathrm{A_{r\text{-}1}} - \mathrm{A_r}}{2\mathrm{Nsin}\left(\frac{2\pi}{\mathrm{N}}\right)}$$

The increase in accuracy will depend on the system conditions, but has been shown to drastically increase the accuracy during many normal dynamic system events and off-nominal frequency conditions [9].

## 4.2 Transient Monitoring

In order to be able to make decisions based on the phasor value, the accuracy of the phasor calculation needs to be validated. This is done through the process of transient monitoring [10]. Given a calculated phasor, the signal can then be reconstructed and compared to the initial data that was used to calculate it. This can be seen in Figure 4.4.

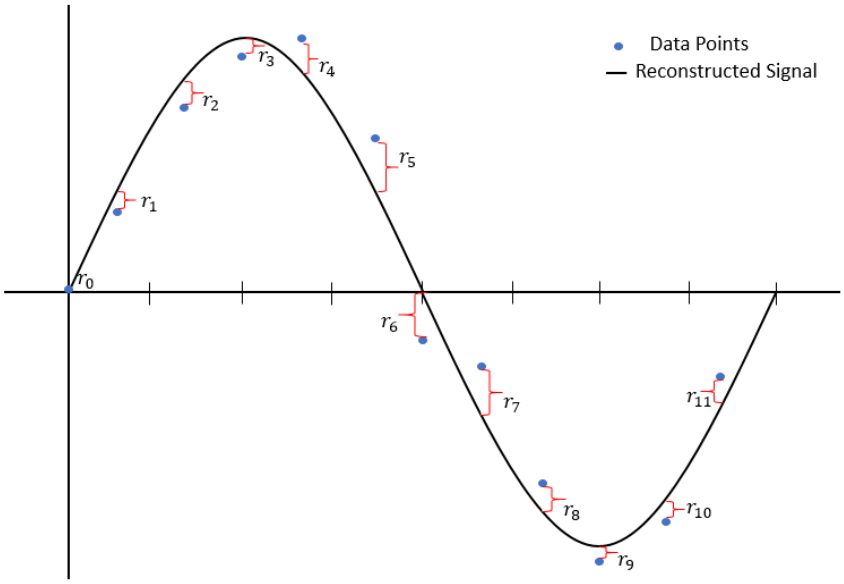


**Figure 4.4: Transient Monitoring**

To think of this mathematically, given a phasor $X\angle\phi$, the reconstructed signal would be $Xcos(\frac{2\pi n}{N}+\phi)$. Each data point $x(n)$, can then be compared to the corresponding data point in $Xcos(\frac{2\pi n}{N}+\phi)$. For example, in this system the comparisons would be as follows:

$$x(0) \leftrightarrow Xcos(\phi)$$

$$x(1) \leftrightarrow Xcos\left(\frac{\pi}{6}+\phi\right)$$

$$x(2) \leftrightarrow Xcos\left(\frac{2\pi}{6}+\phi\right), and\ so\ forth$$

These differences are then squared in order to avoid cancellation between negative and positive differences and then summed. The resultant value can then be used to determine if the phasor was calculated correctly. If the value is large, then the phasor calculation was wrong and the value cannot be used. If the value is small, then the phasor can be used for further processes. This quality check ensures that decisions are made based on a correct phasor, which will prevent false tripping.

One area which is affected by this is the instant a fault occurs. Each new data point that is seen by the window will be under a fault condition, but all of the older data points will be under normal operating conditions. This will cause the phasor to be higher than normal

operating conditions, but lower than the fault conditions which will cause the value as a result of the transient monitoring to be very large. In these circumstances, the device will not trip since the phasor value is not deemed to be of high quality. The device will not trip until an entire cycle under fault conditions is read by the window. This is done intentionally to avoid tripping due to noise or short transient events which could decay very quickly and the system could be back in normal operating conditions. However, this also means that the device will not trip until an entire cycle of the fault condition is under the phasor calculation window.

## 4.3 Protection Schemes

### 4.3.1 Over-Current Protection

Over-Current protection is one of the more straight forward protection schemes. When a fault occurs in the system, the current in the line will rapidly increase in magnitude. An over-current protection device tracks the value of the current in the line and trips when it has gone above a specified value. Since the phasor measures the magnitude of the signal in the line, this becomes a simple comparison that the microprocessor can do. This device deploys over-current protection by checking the magnitude of the phasor against a set value, ensuring that the phasor is correct by checking the value obtained from transient monitoring, and then tripping when the appropriate conditions are met.

### 4.3.2 Distance/Impedance Relay

The distance relay protection scheme is more complicated than the over-current scheme, but as a result also provides more capabilities. Distance relays not only determine when a fault has occurred, but also have the added benefit of giving an estimate of the location of the fault in the transmission line. This protection scheme takes both voltage and current as an input, and uses those quantities to compute an impedance. If we consider a transmission line with an impedance of $R + jX$, when a fault occurs in that line this impedance will be split at the location of the fault as seen in Figure 4.5.

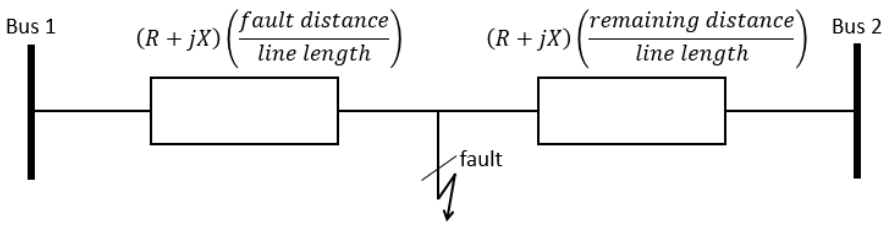


**Figure 4.5: Transmission Line Fault**

The impedance of the line now seen from Bus 1 will only be the first portion of the line which terminates at the fault. Using this information, the calculated impedance of the line can be checked against the known impedance of the line to determine if a fault has occurred. Since the impedance of the line depends on the phase angles, it is typically depicted as a circle of the possible values. Therefore, if the value of the calculated impedance falls within the circle, it is considered as tripping. Typically, in a scheme such as this one there are multiple limits to determine the severity of the fault. The outer limit serves more as a warning that something may be occurring, while the inner limit is used when a serious fault has occurred. A typical set-up of this can be seen in Figure 4.6.

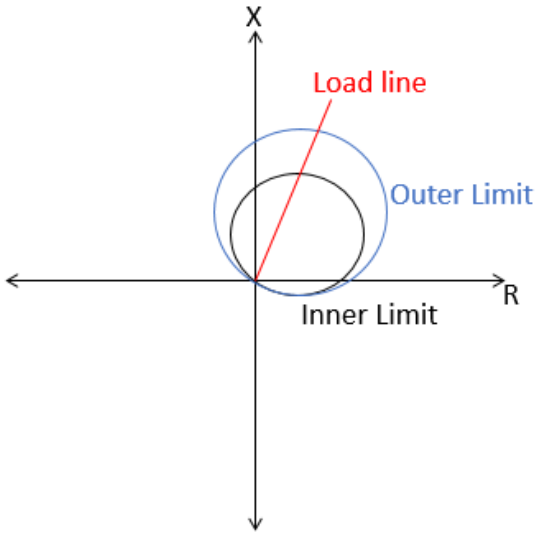


**Figure 4.6: Distance Relay Configuration**

In addition to the two limits, distance relays generally have multiple levels of protection such that there is redundancy in the system from multiple buses. Level one usually checks for

80% of the line impedance and trips instantly. Level two checks for 120%, meaning it covers some of the subsequent transmission lines and has a delay associated with it so that it only trips if the level one check failed to trip at either bus. The use of a level three is a debated topic, but acts as a level two would but with more delay and a higher coverage area.

The location of the fault is also determined using the impedance calculated. The ratio of calculated impedance and normal line impedance is equivalent to the ratio of the fault distance from the bus and the total line length. Since the length and impedance of the transmission line is known under normal operating conditions, and the calculated impedance is found, the only unknown is the fault distance. In this way, the location of the fault can be estimated making repairs to the line much quicker and easier.

## 4.4 Graphical User Interface

In order to increase the ease of usage and the functionality of the protection device, a GUI was created in conjunction with it. When the GUI is first opened, the window shown in Figure 4.7 will appear.

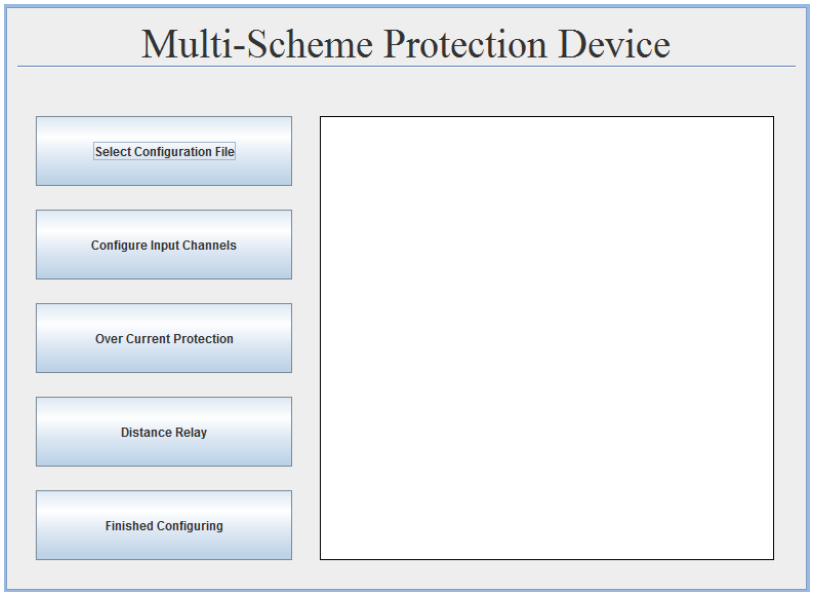


**Figure 4.7: GUI Main Window**

From there, the buttons go in order of the configuration. If a section has already been configured or the configuration doesn't need to be changed, then that portion of the process

can be skipped. The first step is to select the configuration file which to device gets configured from. The configuration of the device was done through the use of a separate file as to create a buffer between the GUI and the device. If the GUI malfunctions, the hard code of the device is not affected. The file is selected through the “Select Configuration File” button, which opens up the window shown in Figure 4.8.

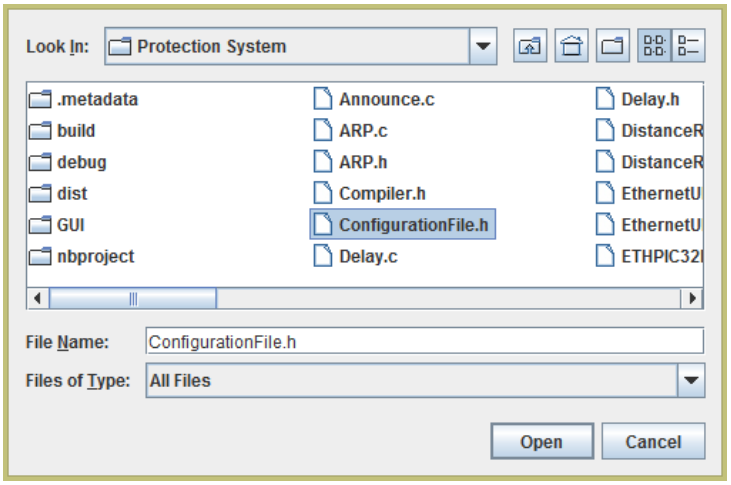


**Figure 4.8: GUI Configuration File Selection**

After the correct configuration file has been selected, the functionality of the device can be set. The “Configure Input Channels” button allows the selection of which data the protection device will operate on. This configuration includes selecting the relevant input channels, the ratings of the input channels and the type of data that is being received (Voltage or Current) as can be seen in Figure 4.9. To maintain the per-unit value of the channel, the rating field should either be set to 1 or left un-altered.

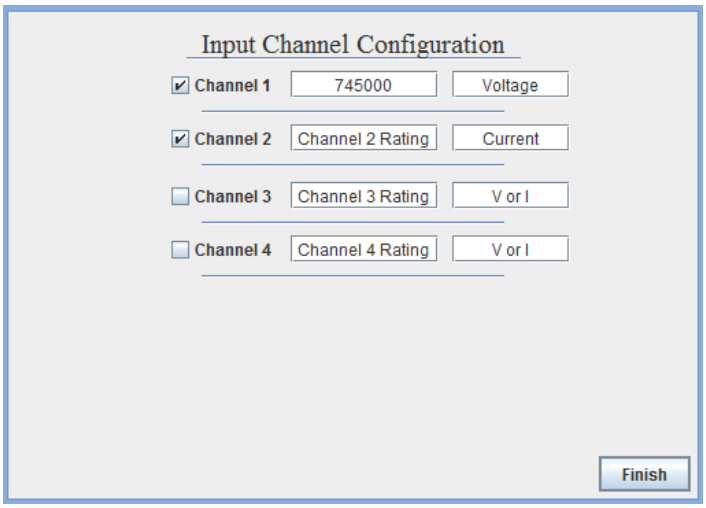


**Figure 4.9: GUI Input Channel Configuration**

The final step of the configuration is to set up the protection schemes. The two possible schemes are an overcurrent scheme which is set up with the "Over Current Protection" button, and the distance relay protection scheme which is set up with the "Distance Relay" button. Opening the over current protection configuration will result in the window shown in Figure 4.10.

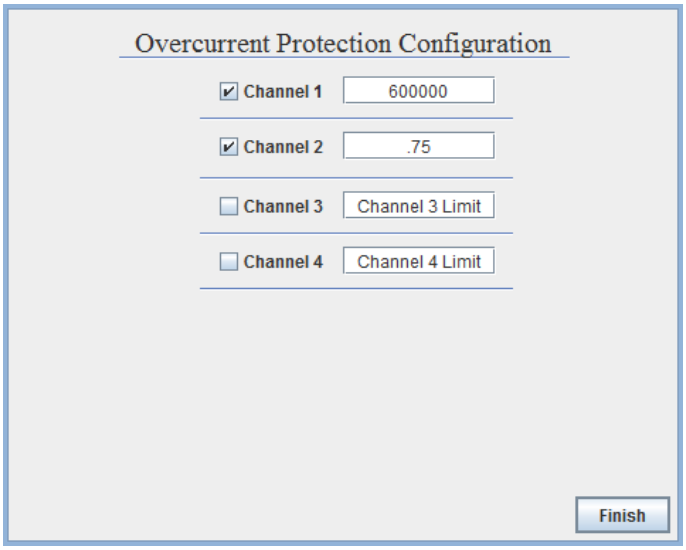


**Figure 4.10: GUI Over Current Configuration**

In this window, the channels which operate an over current protection scheme are selected as well as the limit value, which once the phasor has gone above causes the device to send a trip signal. The window which is opened by the distance relay button is shown in Figure

4.11. Similarly, this window allows for the selection upon which two channels the distance relay operates and the relevant limits to the schemes operation.

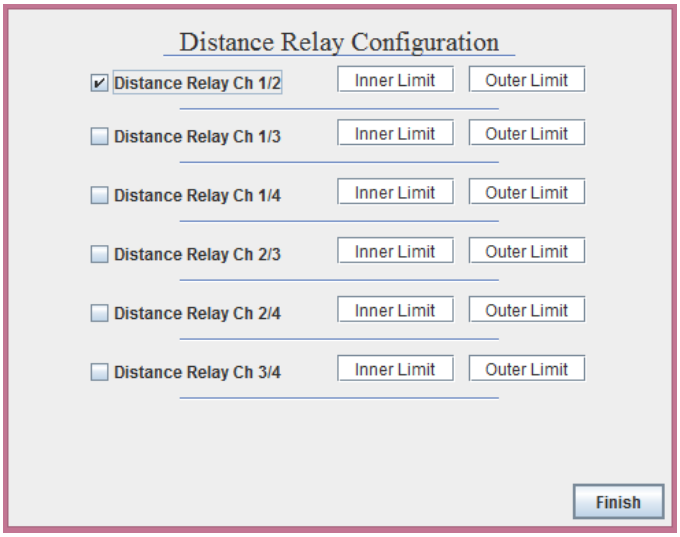


**Figure 4.11: GUI Distance Relay Configuration**

Once the desired configuration of the device has been attained, the "Finished Configuring" button opens up the device output window as shown in Figure 4.12.

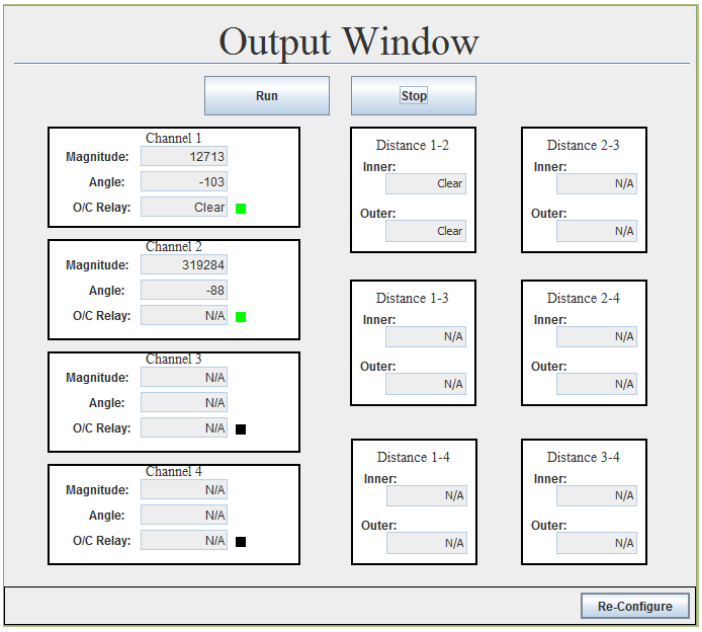


**Figure 4.12: GUI Output Window**

This window displays the output produced by the protection device. Every channel that is selected as an input channel has its magnitude and phase displayed. If an over current relay was selected for a specific channel, the O/C Relay text field displays the current condition of the

system, either clear condition of fault condition. The distance relays similarly display if the condition of the system is clear or in a fault condition. This allows for the observation of both the values upon which the protection device is making its decisions and what decisions are being made. If there is a desire to reconfigure the device, the “Re-Configure” button can be selected. However, the device needs to stop operating while it is being re-configured.

There are several benefits to having the device function through the use of a GUI. The GUI allows for the user to configure the device without having to go into the hard code and sort through each line of code to change the device to their desired settings. Instead, the user can easily switch what type of protection schemes are in operation and where they are operating through the GUI.

Additionally, the user does not have to physically go to each device to change their settings. Simply by sitting at their computer, the user can change the settings of each device in the system through the GUI with a few mouse clicks. The task of configuring the system is now much simpler and less time consuming. Rather than go to each device individually and altering their settings, the entire system can be re-configured from the computer.

Another functionality of the GUI is to track and display the output of the protection device. This allows for the operator to see the decisions the device is making and the values from which the device is making these decisions.

# 5 Conclusion

The system presented in this paper has shown the functionality of an entirely digital system for power system metering and protection. There are many benefits associated with the use of a digital system, such as the one presented, in comparison to the typical system in use currently.

With the use of a single device that digitizes the data and broadcasts it to the network connected to the instrument transformers, the effect of the burden on these instrument transformer is mitigated. Currently, any device that wants to be added to the system and access the information needs to connect itself directly to the instrument transformer. This decreases the overall load impedance of the transformer which increases the transformer error. This error will get transferred to the values that are used by the device. With this system, whenever a device wants to be added, it simply needs to be plugged into the communications network to gain access to the data and the accuracy of the data will not change.

When a system is entirely digital it is also much easier to configure. Traditionally, every device that needs to be changed has to be done so manually by working on those specific devices individually. Digital devices have the benefit of being configurable from a computer, and if the entire system is digital, then everything can be configured from the computer. This makes re-configuring and updating the system much simpler and less time consuming.

Currently, every new device or protection scheme that is added to the system has its own dedicated microprocessor. If each scheme has multiple devices for redundancy and considering the size of the power system, this means that there are a huge number of devices. However, all of these devices have the same hardware, a microprocessor. If these microprocessors are implemented with multiple functionalities then the number of devices is reduced drastically. With so many benefits associated with an entirely digital system, the widespread usage of such systems is inevitable.

# References


[1] R. E. Mackiewicz, "Overview of IEC 61850 and Benefits," IEEE, 2006.

[2] Digilent Inc., *Cerebot 32MX7 Board Referene Manual,* 2011.

[3] Digilent Inc., "PmodGPS Reference Manual," 2012.

[4] Microchip Technology Inc., *PIC32MX5XX/6XX/7XX Family Data Sheet,* 2013.

[5] Maxim Integrated Products, Inc., "Understanding SAR ADCs: Their Architecture and Comparison with Other ADCs," Oct 02, 2001.

[6] RuggedCom Inc., "Latency on a Switched Ethernet Network," 2008.

[7] L. A. H. G. R. V. J. Ricardo Alexsandro de Medeiros Valentim, "Performance Analysis of Protocols: UDP and RAW Ethernet to Real-Time Networks".

[8] A. Phadke, J. Thorp and M. Adamiak, "A New Measurement Technique for Tracking Voltage Phasors, Local System Frequency, and Rate of Change of Frequency," *IEEE Transactions on Power Apparatus and Systems,* 1983.

[9] W. Premerlani, B. Kasztenny and M. Adamiak, "Development and Implementation of a Synchrophasor Estimator Capable of Measurements Under Dynamic Conditions," *IEEE Transactions on Power Delivery,* 2008.

[10] A. Phadke and J. Thorp, Computer Relaying for Power Systems, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, 2009.

# A. Appendix

The Ethernet network code was modifying from the Microchip MLA standard functions. The key functions are included in this appendix.

## A.1 Measurement Device Code

### A.1.1 Main Function

```
#define THIS_IS_STACK_APPLICATION

#include "TCPIP.h"
#include "Main.h"
#include "ADC.h"
#include "UARTGPS.h"

extern int sprintf(char *, const char *, ...);
unsigned int Channel_1;
unsigned int Channel_2;
unsigned int Channel_3;
unsigned int Channel_4;
unsigned int offset;
double convertedvalue;
int GPS_BAUD = 9600;

// Declare AppConfig structure and some other supporting stack variables
APP_CONFIG AppConfig;
static unsigned short wOriginalAppConfigChecksum;    // Checksum of the ROM defaults for AppConfig
BYTE AN0String[8];

int main(void)
{

   //Initialize the LAN8720 PHY
   TRISE = 0;
   mPORTEClearBits(BIT_9);
   DelayMs(100);
   mPORTESetBits(BIT_9);


   char adcoutput[20];
   char speedCommand[] = "$PMTK251,57600*2C\r\n";
   char updateRate[] = "$PMTK300,200,0,0,0,0*2F\r\n";
   char utc_time[17];
   int sample_count=1;
   int check=0;
   int GetTimeCounter=3599;
   int fs = 720;

   unsigned int TimerValue;

   // Initialize application specific hardware
   InitializeBoard();
   initADC();
//       //sets up UART
   init_UART_JE(UART1,GPS_BAUD);
   GPS_BAUD = 57600;
   SendUARTData(updateRate,sizeof(updateRate),UART1);
   SendUARTData(speedCommand,sizeof(speedCommand),UART1);
   UARTSetDataRate(UART1, GetPeripheralClock(),GPS_BAUD);
   while(UARTReceivedDataIsAvailable(UART1))//Clears out the Uart receive buffer
```

```
  {
    UARTGetDataByte(UART1);
  }


  // Initialize stack-related hardware components that may be
  // required by the UART configuration routines
  TickInit();

  // Initialize Stack and application related NV variables into AppConfig.
  InitAppConfig();

  // Initialize core stack layers (MAC, ARP, TCP, UDP) and
  // application modules (HTTP, SNMP, etc.)
  StackInit();

  ReadGPS_UARTData(utc_time);
  while(1)
  {
    if(GetTimeCounter==1800)
    {
      //get time value
      ReadGPS_UARTData(utc_time);
      GetTimeCounter=0;
    }
    else
    {
      UpdateTime(utc_time);
    }
    while((!mPORTDReadBits(BIT_15)||check==1))
    {
      if(sample_count!=fs+1)
      {
        while(! mAD1GetIntFlag() )
        {
        // wait for the first conversion to complete so there will be vaild data in ADC result registers
        }

        offset = 8 * ((~ReadActiveBufferADC10() & 0x01));// determine which buffer is idle and create an offset
        Channel_1 = ReadADC10(offset);// read the result of conversion from the idle buffer
        Channel_2 = ReadADC10(offset+1);
        Channel_3 = ReadADC10(offset+2);
        Channel_4 = ReadADC10(offset+3);//check this one to ensure proper offset

        TimerValue=ReadCoreTimer();
//        TimerValue=TimerValue+0x00000003;
//        WriteCoreTimer(TimerValue);

        SendEthernetUDP(1,Channel_1,utc_time,sample_count);
        SendEthernetUDP(2,Channel_2,utc_time,sample_count);
        SendEthernetUDP(3,Channel_3,utc_time,sample_count);
        SendEthernetUDP(4,Channel_4,utc_time,sample_count);
        // This task performs normal stack task including checking
        // for incoming packet, type of packet and calling
        // appropriate stack entity to process it.
        StackTask();

        sample_count=sample_count+1;

//        if(mPORTDReadBits(BIT_15)&&check==0)
//        {
//          WriteTimer3(0x0000);
//        }
        if(!mPORTDReadBits(BIT_15))
        {
          check=0;
```

```
            }

            mAD1ClearIntFlag();
          }


        }
        WriteTimer3(0x0d05);
        sample_count=1;
        GetTimeCounter=GetTimeCounter+1;
        check=1;
      }
}

/******************************************************************************
 Function:
   static void InitializeBoard(void)

 Description:
   This routine initializes the hardware.  It is a generic initialization
   routine for many of the Microchip development boards, using definitions
   in HardwareProfile.h to determine specific initialization.

 Precondition:
   None

 Parameters:
   None - None

 Returns:
   None

 Remarks:
   None
 *****************************************************************************/
static void InitializeBoard(void)
{
   // LEDs
   LED0_TRIS = 0;
   LED1_TRIS = 0;
   LED2_TRIS = 0;
   LED3_TRIS = 0;
   LED4_TRIS = 0;
   LED5_TRIS = 0;
   LED6_TRIS = 0;
   LED7_TRIS = 0;
   LED_PUT(0x00);

   INTEnableSystemMultiVectoredInt();

   // Enable optimal performance
   SYSTEMConfigPerformance(GetSystemClock());
   mOSCSetPBDIV(OSC_PB_DIV_8);              // Use 1:1 CPU Core:Peripheral clocks

   // Disable JTAG port so we get our I/O pins back, but first
   // wait 50ms so if you want to reprogram the part with
   // JTAG, you'll still have a tiny window before JTAG goes away.
   // The PIC32 Starter Kit debuggers use JTAG and therefore must not
   // disable JTAG.
   DelayMs(50);
   #if !defined(__MPLAB_DEBUGGER_PIC32MXSK) && !defined(__MPLAB_DEBUGGER_FS2)
      DDPCONbits.JTAGEN = 0;
   #endif
   LED_PUT(0x00);              // Turn the LEDs off
```

```
   CNPUESET = 0x00098000;      // Turn on weak pull ups on CN15, CN16, CN19 (RD5, RD7, RD13), which is connected to buttons on PIC32
Starter Kit boards
}

/*********************************************************************
 * Function:        void InitAppConfig(void)
 *
 * PreCondition:    MPFSInit() is already called.
 *
 * Input:           None
 *
 * Output:          Write/Read non-volatile config variables.
 *
 * Side Effects:    None
 *
 * Overview:        None
 *
 * Note:            None
 ********************************************************************/
// MAC Address Serialization using a MPLAB PM3 Programmer and
// Serialized Quick Turn Programming (SQTP).
// The advantage of using SQTP for programming the MAC Address is it
// allows you to auto-increment the MAC address without recompiling
// the code for each unit.  To use SQTP, the MAC address must be fixed
// at a specific location in program memory.  Uncomment these two pragmas
// that locate the MAC address at 0x1FFF0.  Syntax below is for MPLAB C
// Compiler for PIC18 MCUs. Syntax will vary for other compilers.
//#pragma romdata MACROM=0x1FFF0
static ROM BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1, MY_DEFAULT_MAC_BYTE2, MY_DEFAULT_MAC_BYTE3,
MY_DEFAULT_MAC_BYTE4, MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
//#pragma romdata

static void InitAppConfig(void)
{
  while(1)
  {
    // Start out zeroing all AppConfig bytes to ensure all fields are
    // deterministic for checksum generation
    memset((void*)&AppConfig, 0x00, sizeof(AppConfig));

    AppConfig.Flags.bIsDHCPEnabled = TRUE;
    AppConfig.Flags.bInConfigMode = TRUE;
    memcpypgm2ram((void*)&AppConfig.MyMACAddr, (ROM void*)SerializedMACAddress, sizeof(AppConfig.MyMACAddr));

    AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 | MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
    AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
    AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 | MY_DEFAULT_MASK_BYTE2<<8ul | MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;
    AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
    AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 | MY_DEFAULT_GATE_BYTE2<<8ul | MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;
    AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 | MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul  | MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
    AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 | MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul  | MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;

    // Load the default NetBIOS Host Name
    memcpypgm2ram(AppConfig.NetBIOSName, (ROM void*)MY_DEFAULT_HOST_NAME, 16);
    FormatNetBIOSName(AppConfig.NetBIOSName);

    // Compute the checksum of the AppConfig defaults as loaded from ROM
    wOriginalAppConfigChecksum = CalcIPChecksum((BYTE*)&AppConfig, sizeof(AppConfig));
    break;
  }
}
```

## A.1.2 ADC

```
#include <plib.h>
#include "ADC.h"
#include "HardwareProfile.h"

void initADC()
{
	CloseADC10();

	// Setup timer 3 to interrupt once per second at interrupt priority 7.
	// Assuming a processor frequency of 80 MHz, peripheral bus frequency set
	// to 1/8 of the processor frequency, and a timer prescale of 256, we get
	// 80,000,000 / (8 * 256) = 39062 ticks.

    //OpenTimer23(T2_ON | T2_PS_1_1 | T2_32BIT_MODE_ON,0x00013333);

	OpenTimer3(T3_ON | T3_SOURCE_INT | T3_PS_1_1, 6945);
	//ConfigIntTimer3(T3_INT_OFF | T3_INT_PRIOR_7);

	//***********************************************
	//CONFIGURE THE ADC

	//consider turning auto-sampling off and then sampling when the interrupt occurs
	#define PARAM1 ADC_FORMAT_SIGN_INT16 | ADC_CLK_TMR | ADC_AUTO_SAMPLING_ON

	//sets reference voltages for adc(Vmax to Vmin), doesnt scan input channels and only allows for one input
	//buffer is set as 2 8 byte words rather than 1 16 byte word
	#define PARAM2 ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF | ADC_SAMPLES_PER_INT_2 |
ADC_ALT_BUF_ON | ADC_ALT_INPUT_OFF

	//ADC_SAMPLE_TIME sets the delay time before next sample (look into conversion clock select bits)
	#define PARAM3 ADC_CONV_CLK_SYSTEM | ADC_SAMPLE_TIME_15

	//sets AN2(JA-01),AN3(JA-02),AN4(JA-03),AN10(JA-10) as an analog input
    //sets ground and Vref as ranges(default)
	#define PARAM5 ENABLE_ALL_ANA

	//scanning is turned off
	#define PARAM4 SKIP_SCAN_ALL
	//***********************************************

	SetChanADC10(ADC_CH0_POS_SAMPLEA_AN2 | ADC_CH0_NEG_SAMPLEA_NVREF | ADC_CH0_POS_SAMPLEB_AN3 |
ADC_CH0_NEG_SAMPLEB_NVREF);
	ConfigIntADC10(ADC_INT_PRI_3 | ADC_INT_SUB_PRI_3 | ADC_INT_OFF);
	OpenADC10(PARAM1,PARAM2,PARAM3,PARAM4,PARAM5);

	EnableADC10();
}

static void __ISR(_TIMER_3_VECTOR, ipl7) _Timer3Handler(void)
{
	// Clear the interrupt flag.
	mT3ClearIntFlag();
}
```

# A.2 Communication System Code

## A.2.1 Ethernet Send Function

```
#define __EthernetUDP_C

#include "TCPIPConfig.h"
#include "TCPIP.h"

#define PORT            30000

extern NODE_INFO remoteNode;

void SendEthernetUDP(int channel,unsigned int value,char time[],int sample_counter)
{
        UDP_SOCKET          MySocket;
    int i=0;

    //PORT=30000+channel;

        if(!MACIsLinked())  // Check for link before blindly opening and transmitting (similar to DHCP case)
                return;

        // Open a UDP socket for outbound broadcast transmission
    MySocket = UDPOpenEx(0,UDP_OPEN_IP_ADDRESS,55056,30000+channel);
    //0x0101FEA9

        // Abort operation if no UDP sockets are available
        // If this ever happens, incrementing MAX_UDP_SOCKETS in
        // StackTsk.h may help (at the expense of more global memory
        // resources).
        if(MySocket == INVALID_UDP_SOCKET)
                return;

        // Make certain the socket can be written to
        while(!UDPIsPutReady(MySocket));

    unsigned char ADCvalue[2]; // or char, but unsigned char is better
    char Timevalue[10];
    unsigned char SampleCountvalue[2];

    ADCvalue[0] = value >> 8 & 0xFF;
    ADCvalue[1] = value     & 0xFF;

    for(i=0;i<sizeof(Timevalue);i++)
    {
       Timevalue[i]=time[i+7];
    }

//      SampleCountvalue[0] = sample_counter >>  24 & 0xFF;
//      SampleCountvalue[1] = sample_counter >>  16 & 0xFF;
//      SampleCountvalue[2] = sample_counter >>  8 & 0xFF;
//      SampleCountvalue[3] = sample_counter      & 0xFF;

    SampleCountvalue[0] = sample_counter >>  8 & 0xFF;
    SampleCountvalue[1] = sample_counter      & 0xFF;

    //UDPPut((BYTE*)DataType);
    UDPPutArray((BYTE*)ADCvalue,sizeof(ADCvalue));
    UDPPutArray((BYTE*)Timevalue,sizeof(Timevalue));
    UDPPutArray((BYTE*)SampleCountvalue,sizeof(SampleCountvalue));

        // Send the packet
        UDPFlush();
```

```
		// Close the socket so it can be used by other modules
		UDPClose(MySocket);
}

double ConvertValue(unsigned int value,double factor,int sample)
{
  double cv;
  double temp;
  if(CHECK_BIT(value, 10))//negative number (try to find better statement
  {
    temp=(value-65024)*.0032289628180039;
    cv=temp-(1.65)*(factor/1.65);
  }
  else//positive numbers
  {
    cv=(value*.0032289628180039)*(factor/1.65);
  }
  return cv;
}
```

## A.2.2 Ethernet Receive Function

```
#define __EthernetUDP_C

#include "TCPIPConfig.h"
#include "TCPIP.h"
#include "ConfigurationFile.h"

void SendEthernetUDPChannel(int channel,double PhasorMag,double PhasorAng,double Residual,int OCTrip)
{
		UDP_SOCKET		MySocket;
    int i=0;

    //PORT=30000+channel;

		if(!MACIsLinked())  // Check for link before blindly opening and transmitting (similar to DHCP case)
			return;

		// Open a UDP socket for outbound broadcast transmission
    MySocket = UDPOpenEx(0,UDP_OPEN_IP_ADDRESS,2860,20000+channel);

		// Abort operation if no UDP sockets are available
		// If this ever happens, incrementing MAX_UDP_SOCKETS in
		// StackTsk.h may help (at the expense of more global memory
		// resources).
		if(MySocket == INVALID_UDP_SOCKET)
			return;

		// Make certain the socket can be written to
		while(!UDPIsPutReady(MySocket));

    unsigned char PhasorMagVal[4]; // or char, but unsigned char is better
    unsigned char PhasorAngVal[4];
    unsigned char ResidualVal[4];
    unsigned char OCTripVal[1];



    PhasorMagVal[0] = (int)PhasorMag >> 24 & 0xFF;
    PhasorMagVal[1] = (int)PhasorMag >> 16 & 0xFF;
    PhasorMagVal[2] = (int)PhasorMag >>  8 & 0xFF;
    PhasorMagVal[3] = (int)PhasorMag      & 0xFF;

    PhasorAngVal[0] = (int)PhasorAng >> 24 & 0xFF;
    PhasorAngVal[1] = (int)PhasorAng >> 16 & 0xFF;
```

```
	PhasorAngVal[2] = (int)PhasorAng >>  8 & 0xFF;
	PhasorAngVal[3] = (int)PhasorAng      & 0xFF;

	ResidualVal[0] = (int)Residual >> 24 & 0xFF;
	ResidualVal[1] = (int)Residual >> 16 & 0xFF;
	ResidualVal[2] = (int)Residual >>  8 & 0xFF;
	ResidualVal[3] = (int)Residual       & 0xFF;

	OCTripVal[0] = OCTrip & 0xFF;


	//UDPPut((BYTE*)DataType);
	UDPPutArray((BYTE*)PhasorMagVal,sizeof(PhasorMagVal));
	UDPPutArray((BYTE*)PhasorAngVal,sizeof(PhasorAngVal));
	UDPPutArray((BYTE*)ResidualVal,sizeof(ResidualVal));
	UDPPutArray((BYTE*)OCTripVal,sizeof(OCTripVal));

		// Send the packet
		UDPFlush();

		// Close the socket so it can be used by other modules
		UDPClose(MySocket);
}

void SendEthernetUDPDistance(int channel,int DTripI,int DTripO,double ImpMag,double ImpAng)
{
		UDP_SOCKET		MySocket;
	int i=0;

	//PORT=30000+channel;

		if(!MACIsLinked())  // Check for link before blindly opening and transmitting (similar to DHCP case)
			return;

		// Open a UDP socket for outbound broadcast transmission
	MySocket = UDPOpenEx(0,UDP_OPEN_IP_ADDRESS,2860,40000+channel);

		// Abort operation if no UDP sockets are available
		// If this ever happens, incrementing MAX_UDP_SOCKETS in
		// StackTsk.h may help (at the expense of more global memory
		// resources).
		if(MySocket == INVALID_UDP_SOCKET)
			return;

		// Make certain the socket can be written to
		while(!UDPIsPutReady(MySocket));

	unsigned char DTripIVal[1];
	unsigned char DTripOVal[1];
	unsigned char ImpMagVal[4];
	unsigned char ImpAngVal[4];

	DTripIVal[0] = DTripI & 0xFF;

	DTripOVal[0] = DTripO & 0xFF;

	ImpMagVal[0] = (int)ImpMag >> 24 & 0xFF;
	ImpMagVal[1] = (int)ImpMag >> 16 & 0xFF;
	ImpMagVal[2] = (int)ImpMag >>  8 & 0xFF;
	ImpMagVal[3] = (int)ImpMag       & 0xFF;

	ImpAngVal[0] = (int)ImpAng >> 24 & 0xFF;
	ImpAngVal[1] = (int)ImpAng >> 16 & 0xFF;
	ImpAngVal[2] = (int)ImpAng >>  8 & 0xFF;
	ImpAngVal[3] = (int)ImpAng       & 0xFF;
```

```
    //UDPPut((BYTE*)DataType);
    UDPPutArray((BYTE*)DTripIVal,sizeof(DTripIVal));
    UDPPutArray((BYTE*)DTripOVal,sizeof(DTripOVal));
    UDPPutArray((BYTE*)ImpMagVal,sizeof(ImpMagVal));
    UDPPutArray((BYTE*)ImpAngVal,sizeof(ImpAngVal));

        // Send the packet
        UDPFlush();

        // Close the socket so it can be used by other modules
        UDPClose(MySocket);
}


void ReceiveEthernetUDP(BYTE value[],int channel,int *success)
{
        static enum {
                RECEIVE_HOME = 0,
                RECEIVE_LISTEN,
        } RECEIVE_STATE = RECEIVE_HOME;

        static UDP_SOCKET    MySocket;
        BYTE                receive_data[14];

        switch(RECEIVE_STATE)
        {
                case RECEIVE_HOME:
                        // Open a UDP socket for inbound and outbound transmission
                        // Since we expect to only receive broadcast packets and
                        // only send unicast packets directly to the node we last
                        // received from, the remote NodeInfo parameter can be anything
                        //MySocket = UDPOpen(ANNOUNCE_PORT, NULL, ANNOUNCE_PORT);
                        MySocket = UDPOpenEx(0,UDP_OPEN_SERVER,30000+channel, 30000+channel);

                        if(MySocket == INVALID_UDP_SOCKET)
                                return;
                        else
                                RECEIVE_STATE++;
                        break;

                case RECEIVE_LISTEN:
                        // Do nothing if no data is waiting
                        if(!UDPIsGetReady(MySocket))
                                return;

                        UDPGetArray(&receive_data[0], 14);
          *success=1;
          int i;
          for(i=0;i<14;i++)
          {
            value[i]=receive_data[i];
          }
          UDPDiscard();
          UDPClose(MySocket);
          RECEIVE_STATE=RECEIVE_HOME;

        }

}

void ReceiveEthernetUDP2(BYTE value[],int channel)
{
        static enum {
                RECEIVE_HOME = 0,
                RECEIVE_LISTEN,
```

```
	} RECEIVE_STATE = RECEIVE_HOME;

	static UDP_SOCKET	MySocket;
	BYTE		receive_data[14];

	switch(RECEIVE_STATE)
	{
		case RECEIVE_HOME:
			// Open a UDP socket for inbound and outbound transmission
			// Since we expect to only receive broadcast packets and
			// only send unicast packets directly to the node we last
			// received from, the remote NodeInfo parameter can be anything
			//MySocket = UDPOpen(ANNOUNCE_PORT, NULL, ANNOUNCE_PORT);
			MySocket = UDPOpenEx(0,UDP_OPEN_SERVER,30000+channel, 30000+channel);

			if(MySocket == INVALID_UDP_SOCKET)
				return;
			else
				RECEIVE_STATE++;
			break;

		case RECEIVE_LISTEN:
			// Do nothing if no data is waiting
			if(!UDPIsGetReady(MySocket))
				return;

			UDPGetArray(&receive_data[0], 14);
		  int i;
		  for(i=0;i<14;i++)
		  {
		    value[i]=receive_data[i];
		  }
		  UDPDiscard();
		  UDPClose(MySocket);
		  RECEIVE_STATE=RECEIVE_HOME;

	}

}


void DecipherMessage(BYTE value[],unsigned int *ADCval,int *time,int *samp_counter)
{
  //ADC is first 2 bytes
  *ADCval = (value[0]<<8) | (value[1]);
  //counter is final 2 bytes
  *samp_counter = ((value[12]<<8)) | value[13];
  //time is middle 10 bytes
  //*time=(((value[2]-0x30)*10)+(value[3]-0x30))*3600;//add hours
  //*time=*time+((((value[4]-0x30)*10)+(value[5]-0x30))*60);//minutes
  *time=((((value[6]-0x30)*10)+(value[7]-0x30)));//seconds
}
```

# A.3 Protection Device Code

## A.3.1 Main Function

```
#define THIS_IS_STACK_APPLICATION

#include "TCPIP.h"
#include "Main.h"
#include "Phasor.h"
#include "LCD.h"
#include <math.h>
```

```
#include "ConfigurationFile.h"
#include "OverCurrent.h"
#include "DistanceRelay.h"



extern int sprintf(char *, const char *, ...);

const char clearDisplay[] = "j";  // Clears the LCD and homes the cursor.
const char newLine[] = "1;0H";
char DisplayString[20];           // Used to format the elapsed time.

// Declare AppConfig structure and some other supporting stack variables
APP_CONFIG AppConfig;
static unsigned short wOriginalAppConfigChecksum;    // Checksum of the ROM defaults for AppConfig
BYTE AN0String[8];

int main(void)
{
   static DWORD t = 0;
   BYTE Channel_1[14],Channel_2[14],Channel_3[14],Channel_4[14];
   unsigned int ADCval1,ADCval2,ADCval3,ADCval4;
   int time1,time2,time3,time4;
   int counter1,counter2,counter3,counter4;
   double Mag1,Mag2,Mag3,Mag4;
   double Ang1,Ang2,Ang3,Ang4;
   double Residual1,Residual2,Residual3,Residual4;
   double Data_Array1[12]={0};
   double Data_Array2[12]={0};
   double Data_Array3[12]={0};
   double Data_Array4[12]={0};
   double Time_Array1[12]={0};
   double Time_Array2[12]={0};
   double Time_Array3[12]={0};
   double Time_Array4[12]={0};
   int Recursive_Counter1=0;
   int Recursive_Counter2=0;
   int Recursive_Counter3=0;
   int Recursive_Counter4=0;
   int Initial_Phasor1=0;
   int Initial_Phasor2=0;
   int Initial_Phasor3=0;
   int Initial_Phasor4=0;
   double Real_Last1=0;
   double Real_Last2=0;
   double Real_Last3=0;
   double Real_Last4=0;
   double Imag_Last1=0;
   double Imag_Last2=0;
   double Imag_Last3=0;
   double Imag_Last4=0;
   int TripCounter1=0;
   int TripCounter2=0;
   int TripCounter3=0;
   int TripCounter4=0;
   int DelayCounter1=0;
   int DelayCounter2=0;
   int DelayCounter3=0;
   int DelayCounter4=0;
   int OCTrip1,OCTrip2,OCTrip3,OCTrip4;
   int DTripI12,DTripI13,DTripI14,DTripI23,DTripI24,DTripI34;
   int DTripO12,DTripO13,DTripO14,DTripO23,DTripO24,DTripO34;
   double ImpMag12,ImpMag13,ImpMag14,ImpMag23,ImpMag24,ImpMag34;
   double ImpAng12,ImpAng13,ImpAng14,ImpAng23,ImpAng24,ImpAng34;
   int success=0;
```

```
  //Initialize the LAN8720 PHY
  TRISE = 0;
  mPORTEClearBits(BIT_9);
  DelayMs(100);
  mPORTESetBits(BIT_9);

  initLCD();

  // Initialize application specific hardware
  InitializeBoard();

  // Initialize stack-related hardware components that may be
  // required by the UART configuration routines
  TickInit();

  // Initialize Stack and application related NV variables into AppConfig.
  InitAppConfig();

  // Initialize core stack layers (MAC, ARP, TCP, UDP) and
  // application modules (HTTP, SNMP, etc.)
  StackInit();


  while(1)
  {
    // This task performs normal stack task including checking
    // for incoming packet, type of packet and calling
    // appropriate stack entity to process it.
    StackTask();


    #if defined(Channel1)
      ReceiveEthernetUDP(Channel_1,1,&success);
    #endif
    #if defined(Channel2)
      ReceiveEthernetUDP2(Channel_2,2);
    #endif
    #if defined(Channel3)
      ReceiveEthernetUDP2(Channel_3,3);
    #endif
    #if defined(Channel4)
      ReceiveEthernetUDP2(Channel_4,4);
    #endif

    if(success==1)
    {

      #if defined(Channel1)
        DecipherMessage(Channel_1,&ADCval1,&time1,&counter1);

CalcPhasor(Data_Array1,Time_Array1,&Mag1,&Ang1,ADCval1,time1,counter1,&Residual1,CF01,&Recursive_Counter1,&Initial_Phasor1,&Real_
Last1,&Imag_Last1);
      #endif
      #if defined(Channel2)
        DecipherMessage(Channel_2,&ADCval2,&time2,&counter2);

CalcPhasor(Data_Array2,Time_Array2,&Mag2,&Ang2,ADCval2,time2,counter2,&Residual2,CF2,&Recursive_Counter2,&Initial_Phasor2,&Real_L
ast2,&Imag_Last2);
      #endif
      #if defined(Channel3)
        DecipherMessage(Channel_3,&ADCval3,&time3,&counter3);

CalcPhasor(Data_Array3,Time_Array3,&Mag3,&Ang3,ADCval3,time3,counter3,Residual3,CF3,Recursive_Counter3,Initial_Phasor3,Real_Last3,Im
ag_Last3);
      #endif
      #if defined(Channel4)
```

```
        DecipherMessage(Channel_4,&ADCval4,&time4,&counter4);

CalcPhasor(Data_Array4,Time_Array4,&Mag4,&Ang4,ADCval4,time4,counter4,Residual4,CF4,Recursive_Counter4,Initial_Phasor4,Real_Last4,Im
ag_Last4);
      #endif

      #if defined(OCChannel1)
        #if defined(Channel1)
          if(OverCurrentProt(Mag1,Residual1,OCLimit01,&TripCounter1))
            OCTrip1=1;
          else if(OverCurrentDelay(Mag1,Residual1,OCLimit01,&DelayCounter1))
            OCTrip1=1;
          else
            OCTrip1=0;
        #endif
      #endif
      #if defined(OCChannel2)
        #if defined(Channel2)
          if(OverCurrentProt(Mag2,Residual2,OCLimit2,&TripCounter2))
            OCTrip2=1;
          else if(OverCurrentDelay(Mag2,Residual2,OCLimit2,&DelayCounter2))
            OCTrip2=1;
          else
            OCTrip2=0;
        #endif
      #endif
      #if defined(OCChannel3)
        #if defined(Channel3)
          if(OverCurrentProt(Mag3,Residual3,OCLimit3,&TripCounter3))
            OCTrip3=1;
          else if(OverCurrentDelay(Mag3,Residual3,OCLimit3,&DelayCounter3))
            OCTrip3=1;
          else
            OCTrip3=0;
        #endif
      #endif
      #if defined(OCChannel4)
        #if defined(Channel4)
          if(OverCurrentProt(Mag4,Residual4,OCLimit4,&TripCounter4))
            OCTrip4=1;
          else if(OverCurrentDelay(Mag4,Residual4,OCLimit4,&DelayCounter4))
            OCTrip4=1;
          else
            OCTrip4=0;
        #endif
      #endif


      #if defined(Distance12)
        #if defined(Voltage121)
            if(DistanceRelayInner(Mag1,Ang1,Residual1,Mag2,Ang2,Residual2,DLimitInner12,&ImpMag12,&ImpAng12))
              DTripI12=1;
            else if(DistanceRelayOuter(Mag1,Ang1,Residual1,Mag2,Ang2,Residual2,DLimitOuter12,&ImpMag12,&ImpAng12))
              DTripO12=1;
            else
            {
              DTripI12=0;
              DTripO12=0;
            }
            SendEthernetUDPDistance(12,DTripI12,DTripO12,ImpMag12,ImpAng12);
        #endif
        #if defined(Voltage122)
            if(DistanceRelayInner(Mag2,Ang2,Residual2,Mag1,Ang1,Residual1,DLimitInner12,&ImpMag12,&ImpAng12))
            {
              DTripI12=1;
            }
```

```
        else if(DistanceRelayOuter(Mag2,Ang2,Residual2,Mag1,Ang1,Residual1,DLimitOuter12,&ImpMag12,&ImpAng12))
        {
          DTripO12=1;
        }
        else
        {
          DTripI12=0;
          DTripO12=0;
        }
        SendEthernetUDPDistance(12,DTripI12,DTripO12,ImpMag12,ImpAng12);
    #endif
  #endif
  #if defined(Distance13)
    #if defined(Voltage131)
        if(DistanceRelayInner(Mag1,Ang1,Residual1,Mag3,Ang3,Residual3,DLimitInner13,&ImpMag13,&ImpAng13))
        {
          DTripI13=1;
        }
        else if(DistanceRelayOuter(Mag1,Ang1,Residual1,Mag3,Ang3,Residual3,DLimitOuter13,&ImpMag13,&ImpAng13))
        {
          DTripO13=1;
        }
        else
        {
          DTripI13=0;
          DTripO13=0;
        }
        SendEthernetUDPDistance(13,DTripI13,DTripO13,ImpMag13,ImpAng13);
    #endif
    #if defined(Voltage133)
        if(DistanceRelayInner(Mag3,Ang3,Residual3,Mag1,Ang1,Residual1,DLimitInner13,&ImpMag13,&ImpAng13))
        {
          DTripI13=1;
        }
        else if(DistanceRelayOuter(Mag3,Ang3,Residual3,Mag1,Ang1,Residual1,DLimitOuter13,&ImpMag13,&ImpAng13))
        {
          DTripO13=1;
        }
        else
        {
          DTripI13=0;
          DTripO13=0;
        }
        SendEthernetUDPDistance(13,DTripI13,DTripO13,ImpMag13,ImpAng13);
    #endif
  #endif
  #if defined(Distance14)
    #if defined(Voltage141)
        if(DistanceRelayInner(Mag1,Ang1,Residual1,Mag4,Ang4,Residual4,DLimitInner14,&ImpMag14,&ImpAng14))
        {
          DTripI14=1;
        }
        else if(DistanceRelayOuter(Mag1,Ang1,Residual1,Mag4,Ang4,Residual4,DLimitOuter14,&ImpMag14,&ImpAng14))
        {
          DTripO14=1;
        }
        else
        {
          DTripI14=0;
          DTripO14=0;
        }
        SendEthernetUDPDistance(14,DTripI14,DTripO14,ImpMag14,ImpAng14);
    #endif
    #if defined(Voltage144)
        if(DistanceRelayInner(Mag4,Ang4,Residual4,Mag1,Ang1,Residual1,DLimitInner14,&ImpMag14,&ImpAng14))
        {
```

```
            DTripI14=1;
        }
        else if(DistanceRelayOuter(Mag4,Ang4,Residual4,Mag1,Ang1,Residual1,DLimitOuter14,&ImpMag14,&ImpAng14))
        {
            DTripO14=1;
        }
        else
        {
            DTripI14=0;
            DTripO14=0;
        }
        SendEthernetUDPDistance(14,DTripI14,DTripO14,ImpMag14,ImpAng14);
    #endif
#endif
#if defined(Distance23)
    #if defined(Voltage232)
        if(DistanceRelayInner(Mag2,Ang2,Residual2,Mag3,Ang3,Residual3,DLimitInner23,&ImpMag23,&ImpAng23))
        {
            DTripI23=1;
        }
        else if(DistanceRelayOuter(Mag2,Ang2,Residual2,Mag3,Ang3,Residual3,DLimitOuter23,&ImpMag23,&ImpAng23))
        {
            DTripO23=1;
        }
        else
        {
            DTripI23=0;
            DTripO23=0;
        }
        SendEthernetUDPDistance(23,DTripI23,DTripO23,ImpMag23,ImpAng23);
    #endif
    #if defined(Voltage233)
        if(DistanceRelayInner(Mag3,Ang3,Residual3,Mag2,Ang2,Residual2,DLimitInner23,&ImpMag23,&ImpAng23))
        {
            DTripI23=1;
        }
        else if(DistanceRelayOuter(Mag3,Ang3,Residual3,Mag2,Ang2,Residual2,DLimitOuter23,&ImpMag23,&ImpAng23))
        {
            DTripO23=1;
        }
        else
        {
            DTripI23=0;
            DTripO23=0;
        }
        SendEthernetUDPDistance(23,DTripI23,DTripO23,ImpMag23,ImpAng23);
    #endif
#endif
#if defined(Distance24)
    #if defined(Voltage242)
        if(DistanceRelayInner(Mag2,Ang2,Residual2,Mag4,Ang4,Residual4,DLimitInner24,&ImpMag24,&ImpAng24))
        {
            DTripI24=1;
        }
        else if(DistanceRelayOuter(Mag2,Ang2,Residual2,Mag4,Ang4,Residual4,DLimitOuter24,&ImpMag24,&ImpAng24))
        {
            DTripO24=1;
        }
        else
        {
            DTripI24=0;
            DTripO24=0;
        }
        SendEthernetUDPDistance(24,DTripI24,DTripO24,ImpMag24,ImpAng24);
    #endif
    #if defined(Voltage244)
```

```
        if(DistanceRelayInner(Mag4,Ang4,Residual4,Mag2,Ang2,Residual2,DLimitInner24,&ImpMag24,&ImpAng24))
        {
          DTripI24=1;
        }
        else if(DistanceRelayOuter(Mag4,Ang4,Residual4,Mag2,Ang2,Residual2,DLimitOuter24,&ImpMag24,&ImpAng24))
        {
          DTripO24=1;
        }
        else
        {
          DTripI24=0;
          DTripO24=0;
        }
        SendEthernetUDPDistance(24,DTripI24,DTripO24,ImpMag24,ImpAng24);
    #endif
  #endif
  #if defined(Distance34)
    #if defined(Voltage343)
        if(DistanceRelayInner(Mag3,Ang3,Residual3,Mag4,Ang4,Residual4,DLimitInner34,&ImpMag34,&ImpAng34))
        {
          DTripI34=1;
        }
        else if(DistanceRelayOuter(Mag3,Ang3,Residual3,Mag4,Ang4,Residual4,DLimitOuter34,&ImpMag34,&ImpAng34))
        {
          DTripO34=1;
        }
        else
        {
          DTripI34=0;
          DTripO34=0;
        }
        SendEthernetUDPDistance(34,DTripI34,DTripO34,ImpMag34,ImpAng34);
    #endif
    #if defined(Voltage344)
        if(DistanceRelayInner(Mag4,Ang4,Residual4,Mag3,Ang3,Residual3,DLimitInner34,&ImpMag34,&ImpAng34))
        {
          DTripI34=1;
        }
        else if(DistanceRelayOuter(Mag4,Ang4,Residual4,Mag3,Ang3,Residual3,DLimitOuter34,&ImpMag34,&ImpAng34))
        {
          DTripO34=1;
        }
        else
        {
          DTripI34=0;
          DTripO34=0;
        }
        SendEthernetUDPDistance(34,DTripI34,DTripO34,ImpMag34,ImpAng34);
    #endif
  #endif

  #if defined(Channel1)
    SendEthernetUDPChannel(1,Mag1,Ang1,Residual1,OCTrip1);
  #endif
  #if defined(Channel2)
    SendEthernetUDPChannel(2,Mag2,Ang2,Residual2,OCTrip2);
  #endif
  #if defined(Channel3)
    SendEthernetUDPChannel(3,Mag3,Ang3,Residual3,OCTrip3);
  #endif
  #if defined(Channel4)
    SendEthernetUDPChannel(4,Mag4,Ang4,Residual4,OCTrip4);
  #endif

  success=0;
}
```

```
      if(TickGet() - t >= TICK_SECOND/2ul)
      {
         t = TickGet();
         LED0_IO ^= 1;
         getDisplayString(DisplayString,Mag1);
         lcdInstruction(clearDisplay);
         lcdString(DisplayString);
         lcdInstruction(newLine);
         getDisplayString(DisplayString,Mag2);
         lcdString(DisplayString);
      }

   }
}

/*********************************************************************
 Function:
   static void InitializeBoard(void)

 Description:
   This routine initializes the hardware.  It is a generic initialization
   routine for many of the Microchip development boards, using definitions
   in HardwareProfile.h to determine specific initialization.

 Precondition:
   None

 Parameters:
   None - None

 Returns:
   None

 Remarks:
   None
 ********************************************************************/
static void InitializeBoard(void)
{
   // LEDs
   LED0_TRIS = 0;
   LED1_TRIS = 0;
   LED2_TRIS = 0;
   LED3_TRIS = 0;
   LED4_TRIS = 0;
   LED5_TRIS = 0;
   LED6_TRIS = 0;
   LED7_TRIS = 0;
   LED_PUT(0x00);

   INTEnableSystemMultiVectoredInt();

   // Enable optimal performance
   SYSTEMConfigPerformance(GetSystemClock());
   mOSCSetPBDIV(OSC_PB_DIV_8);              // Use 1:1 CPU Core:Peripheral clocks

   // Disable JTAG port so we get our I/O pins back, but first
   // wait 50ms so if you want to reprogram the part with
   // JTAG, you'll still have a tiny window before JTAG goes away.
   // The PIC32 Starter Kit debuggers use JTAG and therefore must not
   // disable JTAG.
   DelayMs(50);
   #if !defined(__MPLAB_DEBUGGER_PIC32MXSK) && !defined(__MPLAB_DEBUGGER_FS2)
      DDPCONbits.JTAGEN = 0;
```

```
  #endif
  LED_PUT(0x00);              // Turn the LEDs off

  CNPUESET = 0x00098000;      // Turn on weak pull ups on CN15, CN16, CN19 (RD5, RD7, RD13), which is connected to buttons on PIC32
Starter Kit boards
}

/*********************************************************************
 * Function:       void InitAppConfig(void)
 *
 * PreCondition:    MPFSInit() is already called.
 *
 * Input:          None
 *
 * Output:         Write/Read non-volatile config variables.
 *
 * Side Effects:    None
 *
 * Overview:       None
 *
 * Note:           None
 ********************************************************************/
// MAC Address Serialization using a MPLAB PM3 Programmer and
// Serialized Quick Turn Programming (SQTP).
// The advantage of using SQTP for programming the MAC Address is it
// allows you to auto-increment the MAC address without recompiling
// the code for each unit.  To use SQTP, the MAC address must be fixed
// at a specific location in program memory.  Uncomment these two pragmas
// that locate the MAC address at 0x1FFF0.  Syntax below is for MPLAB C
// Compiler for PIC18 MCUs. Syntax will vary for other compilers.
//#pragma romdata MACROM=0x1FFF0
static ROM BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1, MY_DEFAULT_MAC_BYTE2, MY_DEFAULT_MAC_BYTE3,
MY_DEFAULT_MAC_BYTE4, MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
//#pragma romdata

static void InitAppConfig(void)
{
  while(1)
  {
    // Start out zeroing all AppConfig bytes to ensure all fields are
    // deterministic for checksum generation
    memset((void*)&AppConfig, 0x00, sizeof(AppConfig));

    AppConfig.Flags.bIsDHCPEnabled = TRUE;
    AppConfig.Flags.bInConfigMode = TRUE;
    memcpypgm2ram((void*)&AppConfig.MyMACAddr, (ROM void*)SerializedMACAddress, sizeof(AppConfig.MyMACAddr));

    AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 | MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
    AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
    AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 | MY_DEFAULT_MASK_BYTE2<<8ul | MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;
    AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
    AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 | MY_DEFAULT_GATE_BYTE2<<8ul | MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;
    AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 | MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul  | MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
    AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 | MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul  | MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;

    // Load the default NetBIOS Host Name
    memcpypgm2ram(AppConfig.NetBIOSName, (ROM void*)MY_DEFAULT_HOST_NAME, 16);
    FormatNetBIOSName(AppConfig.NetBIOSName);

    // Compute the checksum of the AppConfig defaults as loaded from ROM
    wOriginalAppConfigChecksum = CalcIPChecksum((BYTE*)&AppConfig, sizeof(AppConfig));
```

```
    break;
  }
}

void getDisplayString(char string[],double dub)
{
        sprintf(string,"%f",dub);
}
```

## A.3.2 Phasor Calculation Code

```
#include "Phasor.h"
#include "HardwareProfile.h"
#include <math.h>

double fsys=60.0;

void CalcPhasor(double Data_Array[],double Time_Array[],double *Mag, double *Ang,unsigned int ADCval,int Time, int samp_counter,double
*Residual,double ConversionFactor,int *Recursive_Counter,int *Initial_Phasor,double *Real_Last,double *Imag_Last)
{
  int N = 12;
  double diffFactor=2.0*N*sin((2.0*M_PI)/N);
  int i;
  double Data_Point,Prev_Data_Point,Real,Imag,TimeMag,TempR,TempI,diff,tempTime,tempReal,tempImag,AngTemp;
  int tempCount;

  Prev_Data_Point=Data_Array[0];
  for(i=0;i<N-1;i++)
  {
    Data_Array[i]=Data_Array[i+1];
    Time_Array[i]=Time_Array[i+1];
  }
  Data_Point=ConvertADC(ADCval,ConversionFactor);
  i=N-1;
  Data_Array[i]=Data_Point;
  tempTime=Time;
  Time_Array[i]=tempTime+((samp_counter-1)/(N*60.0))-.000431;//.00136574

  if(samp_counter==N)
    *Initial_Phasor=0;


  //dft
  Real=Imag=tempReal=tempImag=0;
  double factor = 2.0/N;
  if(Data_Array[0]!=0&&*Initial_Phasor==0)
  {
    //initial phasor
    for(i=0;i<N;i++)
    {
      tempReal=tempReal+(factor*Data_Array[i]*cos((2*M_PI*i)/N));
      tempImag=tempImag+(-factor*Data_Array[i]*sin((2*M_PI*i)/N));
    }

    Real=tempReal+((tempImag-TempI)/diffFactor);
    Imag=tempImag+((TempR-tempReal)/diffFactor);
    *Real_Last=Real;
    *Imag_Last=Imag;
    *Recursive_Counter=0;
    *Initial_Phasor=1;
  }
  else if(*Initial_Phasor==1)
  {
    diff=Data_Point-Prev_Data_Point;
    TempR = *Real_Last;
```

```
      TempI = *Imag_Last;
      tempCount = *Recursive_Counter;
      tempReal= TempR + (factor*diff*cos((2*M_PI*tempCount)/N));
      tempImag= TempI + (-factor*diff*sin((2*M_PI*tempCount)/N));

      Real=tempReal+((tempImag-TempI)/diffFactor);
      Imag=tempImag+((TempR-tempReal)/diffFactor);

      *Real_Last=Real;
      *Imag_Last=Imag;
      *Recursive_Counter=*Recursive_Counter+1;
   }
   *Mag=sqrt((Real*Real)+(Imag*Imag));
   AngTemp=(atan(fabs(Imag)/fabs(Real))*(180/M_PI));
   if(Imag>0&&Real<0)
      *Ang=180-(AngTemp);
   else if(Imag<0&&Real<0)
      *Ang=(AngTemp)-180;
   else if(Imag<0&&Real>0)
      *Ang=-AngTemp;
   else
      *Ang=AngTemp;

   *Ang=*Ang+9.3;
   if(*Ang>180)
      *Ang=-180+(*Ang-180);
   //29.5


   TimeMag=(*Mag);//*sqrt(2);
   *Residual=0;
   double x,restemp,actemp,average;
   restemp=0;
   for(i=0;i<N;i++)
   {
      x=TimeMag*cos((2*M_PI*fsys*Time_Array[i])+((*Ang)*(M_PI/180)));
      if(x<5)
         x=5;
      restemp=restemp+(100*fabs((Data_Array[i]-x)/x));//*((Data_Array[i]-x)/x);
   }
   *Residual=restemp;

}

double ConvertADC(unsigned int ADCval,double ConversionFactor)
{
   double Converted_Value;
   double temp;
   if(CHECK_BIT(ADCval, 10))//negative number (try to find better statement
   {
      temp=(ADCval-65024)*.0032289628180039;
      Converted_Value=(temp-1.65)*(ConversionFactor/1.65);
   }
   else//positive numbers
   {
      Converted_Value=(ADCval*.0032289628180039)*(ConversionFactor/1.65);
   }
   return Converted_Value;
}
```

## A.3.3 Over-Current Protection

```
#include "OverCurrent.h"
#include <math.h>
```

```
int OverCurrentProt(double Mag,double Residual,double Limit,int *TripCounter)
{
  if(Mag>=Limit&&Residual<=1000)
    *TripCounter=*TripCounter+1;
  else
    *TripCounter=0;


  if(*TripCounter>=3)
    return 1;
  else
    return 0;
}

int OverCurrentDelay(double Mag,double Residual,double Limit,int *DelayCounter)
{
  if(Mag>=Limit&&Residual<=1000)
    *DelayCounter=*DelayCounter+1;
  else
    *DelayCounter=0;


  if(*DelayCounter>=216)
    return 1;
  else
    return 0;

}
```

## A.3.4 Distance Relay

```
#include "DistanceRelay.h"
#include <math.h>

double tempImpMag=0;
double tempImpAng=0;

int DistanceRelayOuter(double Mag1,double Ang1,double Residual1,double Mag2,double Ang2,double Residual2,double Limit,double
*ImpMag,double *ImpAng)
{
  int DisCheck=0;
  if(Residual1<=400&&Residual2<=400)
  {
    tempImpMag=Mag1/Mag2;
    tempImpAng=Ang1-Ang2;
    *ImpMag=tempImpMag;
    *ImpAng=tempImpAng;
    if(tempImpMag<=Limit)
      DisCheck=1;
    else
      DisCheck=0;
  }
  return DisCheck;
}


int DistanceRelayInner(double Mag1,double Ang1,double Residual1,double Mag2,double Ang2,double Residual2,double Limit,double
*ImpMag,double *ImpAng)
{
  int DisCheck=0;
  if(Residual1<=400&&Residual2<=400)
  {
    tempImpMag=Mag1/Mag2;
    tempImpAng=Ang1-Ang2;
```

```
      *ImpMag=tempImpMag;
      *ImpAng=tempImpAng;
      if(tempImpMag<=Limit)
         DisCheck=1;
      else
         DisCheck=0;
   }
   return DisCheck;
}
```

## A.3.5 Configuration File

```
#ifndef ConfigurationFile_H
#define     ConfigurationFile_H

#define Channel1
#define Channel2
//#define Channel3
//#define Channel4
//#define Channel5
//#define Channel6
//#define Channel7
//#define Channel8
//#define Channel9
//#define Channel10
//#define Channel11
//#define Channel12
//#define Channel13
//#define Channel14
//#define Channel15
//#define Channel16

#if defined(Channel1)
            #define CF01 50000
#endif
#if defined(Channel2)
            #define CF2 745000
#endif
#if defined(Channel3)
            #define CF3 3
#endif
#if defined(Channel4)
            #define CF4 4
#endif
#if defined(Channel5)
   #define CF5 220000.0
#endif
#if defined(Channel6)
   #define CF6 220000.0
#endif
#if defined(Channel7)
   #define CF7 220000.0
#endif
#if defined(Channel8)
   #define CF8 220000.0
#endif
#if defined(Channel9)
   #define CF9 220000.0
#endif
#if defined(Channel10)
            #define CF10 100000.0
#endif
#if defined(Channel11)
            #define CF11 1000000.0
#endif
```

```
#if defined(Channel12)
        #define CF12 10000.0
#endif
#if defined(Channel13)
        #define CF13 635465.4
#endif
#if defined(Channel14)
        #define CF14 35685367
#endif
#if defined(Channel15)
        #define CF15 35683567
#endif
#if defined(Channel16)
        #define CF16 2347345
#endif

#define OCChannel1
//#define OCChannel2
//#define OCChannel3
//#define OCChannel4
//#define OCChannel5
//#define OCChannel6
//#define OCChannel7
//#define OCChannel8
//#define OCChannel9
//#define OCChannel10
//#define OCChannel11
//#define OCChannel12
//#define OCChannel13
//#define OCChannel14
//#define OCChannel15
//#define OCChannel16

#if defined(OCChannel1)
        #define OCLimit01 40000
#endif
#if defined(OCChannel2)
        #define OCLimit2 130000
#endif
#if defined(OCChannel3)
        #define OCLimit3 3
#endif
#if defined(OCChannel4)
        #define OCLimit4 4
#endif
#if defined(OCChannel5)
  #define OCLimit5 220000.0
#endif
#if defined(OCChannel6)
  #define OCLimit6 220000.0
#endif
#if defined(OCChannel7)
  #define OCLimit7 220000.0
#endif
#if defined(OCChannel8)
  #define OCLimit8 220000.0
#endif
#if defined(OCChannel9)
  #define OCLimit9 220000.0
#endif
#if defined(OCChannel10)
  #define OCLimit10 220000.0
#endif
#if defined(OCChannel11)
  #define OCLimit11 220000.0
#endif
```

```
#if defined(OCChannel12)
   #define OCLimit12 220000.0
#endif
#if defined(OCChannel13)
   #define OCLimit13 220000.0
#endif
#if defined(OCChannel14)
   #define OCLimit14 220000.0
#endif
#if defined(OCChannel15)
   #define OCLimit15 220000.0
#endif
#if defined(OCChannel16)
   #define OCLimit16 220000.0
#endif

#define Distance12
//#define Distance13
//#define Distance14
//#define Distance23
//#define Distance24
//#define Distance34

#if defined(Distance12)
   ////////////#define Voltage121
   #define Voltage122
               #define DLimitInner12 5
               #define DLimitOuter12 8
#endif
#if defined(Distance13)
   #define Voltage131
   ////#define Voltage133
   #define DLimitInner13 3
               #define DLimitOuter13 4
#endif
#if defined(Distance14)
   #define Voltage141
   ////#define Voltage144
               #define DLimitInner14 5
               #define DLimitOuter14 6
#endif
#if defined(Distance23)
   #define Voltage232
   ////////////////#define Voltage233
               #define DLimitInner23 7
               #define DLimitOuter23 8
#endif
#if defined(Distance24)
   #define Voltage242
   ////////////////#define Voltage244
               #define DLimitInner24 9
               #define DLimitOuter24 10
#endif
#if defined(Distance34)
   #define Voltage343
   //#define Voltage344
               #define DLimitInner34 11
               #define DLimitOuter34 12
#endif

#endif      /* ConfigurationFile_H */
#ifndef ConfigurationFile_H
#define     ConfigurationFile_H

#define Channel1
#define Channel2
```

```
//#define Channel3
//#define Channel4
//#define Channel5
//#define Channel6
//#define Channel7
//#define Channel8
//#define Channel9
//#define Channel10
//#define Channel11
//#define Channel12
//#define Channel13
//#define Channel14
//#define Channel15
//#define Channel16

#if defined(Channel1)
        #define CF01 50000
#endif
#if defined(Channel2)
        #define CF2 745000
#endif
#if defined(Channel3)
        #define CF3 3
#endif
#if defined(Channel4)
        #define CF4 4
#endif
#if defined(Channel5)
  #define CF5 220000.0
#endif
#if defined(Channel6)
  #define CF6 220000.0
#endif
#if defined(Channel7)
  #define CF7 220000.0
#endif
#if defined(Channel8)
  #define CF8 220000.0
#endif
#if defined(Channel9)
  #define CF9 220000.0
#endif
#if defined(Channel10)
        #define CF10 100000.0
#endif
#if defined(Channel11)
        #define CF11 1000000.0
#endif
#if defined(Channel12)
        #define CF12 10000.0
#endif
#if defined(Channel13)
        #define CF13 635465.4
#endif
#if defined(Channel14)
        #define CF14 35685367
#endif
#if defined(Channel15)
        #define CF15 35683567
#endif
#if defined(Channel16)
        #define CF16 2347345
#endif

#define OCChannel1
//#define OCChannel2
```

```
//#define OCChannel3
//#define OCChannel4
//#define OCChannel5
//#define OCChannel6
//#define OCChannel7
//#define OCChannel8
//#define OCChannel9
//#define OCChannel10
//#define OCChannel11
//#define OCChannel12
//#define OCChannel13
//#define OCChannel14
//#define OCChannel15
//#define OCChannel16

#if defined(OCChannel1)
        #define OCLimit01 40000
#endif
#if defined(OCChannel2)
        #define OCLimit2 130000
#endif
#if defined(OCChannel3)
        #define OCLimit3 3
#endif
#if defined(OCChannel4)
        #define OCLimit4 4
#endif
#if defined(OCChannel5)
  #define OCLimit5 220000.0
#endif
#if defined(OCChannel6)
  #define OCLimit6 220000.0
#endif
#if defined(OCChannel7)
  #define OCLimit7 220000.0
#endif
#if defined(OCChannel8)
  #define OCLimit8 220000.0
#endif
#if defined(OCChannel9)
  #define OCLimit9 220000.0
#endif
#if defined(OCChannel10)
  #define OCLimit10 220000.0
#endif
#if defined(OCChannel11)
  #define OCLimit11 220000.0
#endif
#if defined(OCChannel12)
  #define OCLimit12 220000.0
#endif
#if defined(OCChannel13)
  #define OCLimit13 220000.0
#endif
#if defined(OCChannel14)
  #define OCLimit14 220000.0
#endif
#if defined(OCChannel15)
  #define OCLimit15 220000.0
#endif
#if defined(OCChannel16)
  #define OCLimit16 220000.0
#endif

#define Distance12
//#define Distance13
```

```
//#define Distance14
//#define Distance23
//#define Distance24
//#define Distance34

#if defined(Distance12)
  //////////////#define Voltage121
  #define Voltage122
        #define DLimitInner12 5
        #define DLimitOuter12 8
#endif
#if defined(Distance13)
  #define Voltage131
  ////#define Voltage133
  #define DLimitInner13 3
        #define DLimitOuter13 4
#endif
#if defined(Distance14)
  #define Voltage141
  ////#define Voltage144
        #define DLimitInner14 5
        #define DLimitOuter14 6
#endif
#if defined(Distance23)
  #define Voltage232
  //////////////////#define Voltage233
        #define DLimitInner23 7
        #define DLimitOuter23 8
#endif
#if defined(Distance24)
  #define Voltage242
  //////////////////#define Voltage244
        #define DLimitInner24 9
        #define DLimitOuter24 10
#endif
#if defined(Distance34)
  #define Voltage343
  //#define Voltage344
        #define DLimitInner34 11
        #define DLimitOuter34 12
#endif

#endif      /* ConfigurationFile_H */
```

# A.4 GUI Code

## A.4.1 Main Window

```
import java.awt.EventQueue;


import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFileChooser;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.File;
import javax.swing.JCheckBox;
import com.jgoodies.forms.layout.FormLayout;
import com.jgoodies.forms.layout.ColumnSpec;
import com.jgoodies.forms.factories.FormFactory;
```

```
import com.jgoodies.forms.layout.RowSpec;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JSeparator;
import javax.swing.JPanel;
import java.awt.Color;
import javax.swing.border.LineBorder;
import javax.swing.ImageIcon;


public class MainWindow {

	private JFrame frame;
	private static String FilePath;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					MainWindow window = new MainWindow();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public MainWindow()
	{
		initialize();
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.getContentPane().setEnabled(false);
		frame.setBounds(100, 100, 795, 604);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		InputConfiguration inputConfigWindow = new InputConfiguration();
		OverCurrentConfiguration OverCurrConfigWindow = new OverCurrentConfiguration();
		DistanceRelayConfiguration DistRelayConfigWindow = new DistanceRelayConfiguration();
		OutputWindow OutWindow = new OutputWindow();




		JButton btnConfigureOverCurrent = new JButton("Over Current Protection");
		btnConfigureOverCurrent.setBounds(28, 287, 250, 68);
		btnConfigureOverCurrent.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				OverCurrConfigWindow.setVisible(true);
			}
		});
		frame.getContentPane().setLayout(null);
```

```
btnConfigureOverCurrent.setEnabled(false);
frame.getContentPane().add(btnConfigureOverCurrent);

JButton btnDistanceRelay = new JButton("Distance Relay");
btnDistanceRelay.setBounds(28, 378, 250, 68);
btnDistanceRelay.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		DistRelayConfigWindow.setVisible(true);
	}
});
btnDistanceRelay.setEnabled(false);
frame.getContentPane().add(btnDistanceRelay);

JButton btnConfigInput = new JButton("Configure Input Channels");
btnConfigInput.setBounds(28, 196, 250, 68);
btnConfigInput.setEnabled(false);
btnConfigInput.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		inputConfigWindow.setVisible(true);
		btnConfigureOverCurrent.setEnabled(true);
		btnDistanceRelay.setEnabled(true);
	}
});
frame.getContentPane().add(btnConfigInput);


JButton btnNewButton = new JButton("Finished Configuring");
btnNewButton.setEnabled(false);
btnNewButton.setBounds(28, 469, 250, 68);
btnNewButton.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		OutWindow.setVisible(true);
	}
});
frame.getContentPane().add(btnNewButton);

JButton btnSelectConfigurationFile = new JButton("Select Configuration File");
btnSelectConfigurationFile.setBounds(28, 105, 250, 68);
btnSelectConfigurationFile.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		JFileChooser chooser = new JFileChooser();
	   chooser.setCurrentDirectory(new File(".h"));
	   int r = chooser.showOpenDialog(new JFrame());
	   if (r == JFileChooser.APPROVE_OPTION) {
	     FilePath = chooser.getSelectedFile().getPath();
	   }

		btnConfigInput.setEnabled(true);
		btnNewButton.setEnabled(true);
	}
});
frame.getContentPane().add(btnSelectConfigurationFile);


JLabel lblNewLabel = new JLabel("Multi-Scheme Protection Device");
lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 40));
lblNewLabel.setBounds(88, 11, 602, 47);
frame.getContentPane().add(lblNewLabel);

JSeparator separator = new JSeparator();
separator.setBounds(10, 56, 759, 2);
```

```
                frame.getContentPane().add(separator);

                JPanel panel = new JPanel();
                panel.setBorder(new LineBorder(new Color(0, 0, 0)));
                panel.setBackground(Color.WHITE);
                panel.setBounds(305, 105, 443, 432);
                frame.getContentPane().add(panel);
                panel.setLayout(null);

                JLabel lblNewLabel_1 = new JLabel("");
                lblNewLabel_1.setHorizontalAlignment(SwingConstants.CENTER);
                lblNewLabel_1.setIcon(new ImageIcon("C:\\Users\\aschm_000\\Desktop\\Folder\\Thesis\\Protection
System\\GUI\\GUI\\transmission_tower.jpg"));
                lblNewLabel_1.setBounds(10, 11, 423, 410);
                panel.add(lblNewLabel_1);

        }

        public static String FilePath()
        {
                return FilePath;
        }
}
```

## A.4.2 Input Selection

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import java.awt.Font;

import javax.swing.AbstractButton;
import javax.swing.SwingConstants;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.JSeparator;

import java.awt.Color;

import javax.swing.JTable;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;

public class InputConfiguration extends JDialog {

        private final JPanel contentPanel = new JPanel();
```

```
	private JTextField txtChannelRating;
	private JTextField txtChannelRating_3;
	private JTextField txtChannelRating_2;
	private JTextField txtChannelRating_1;
	private double Rating1;
	private double Rating2;
	private double Rating3;
	private double Rating4;
	boolean Ch1 = false;
	boolean Ch2 = false;
	boolean Ch3 = false;
	boolean Ch4 = false;
	boolean Voltage1 = false;
	boolean Voltage2 = false;
	boolean Voltage3 = false;
	boolean Voltage4 = false;

	ArrayList<String> lines = new ArrayList<String>();
String line = null;
private JTextField txtVOrI;
private JTextField txtVOrI_1;
private JTextField textField_1;
private JTextField textField_2;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		try {
			InputConfiguration dialog = new InputConfiguration();
			dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
			dialog.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Create the dialog.
	 */
	public InputConfiguration() {
		setBounds(100, 100, 519, 394);
		getContentPane().setLayout(new BorderLayout());
		contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
		getContentPane().add(contentPanel, BorderLayout.CENTER);
		contentPanel.setLayout(null);

		JLabel lblInputChannelConfiguration = new JLabel("Input Channel Configuration");
		lblInputChannelConfiguration.setHorizontalAlignment(SwingConstants.CENTER);
		lblInputChannelConfiguration.setFont(new Font("Times New Roman", Font.PLAIN, 20));
		lblInputChannelConfiguration.setBounds(141, 11, 221, 24);
		contentPanel.add(lblInputChannelConfiguration);

		JSeparator separator_1 = new JSeparator();
		separator_1.setBounds(130, 33, 244, 2);
		contentPanel.add(separator_1);

		JCheckBox chckbxChannel = new JCheckBox("Channel 1");
		chckbxChannel.setBounds(115, 42, 85, 23);
		contentPanel.add(chckbxChannel);

		txtChannelRating = new JTextField();
		txtChannelRating.setHorizontalAlignment(SwingConstants.CENTER);
		txtChannelRating.setText("Channel 1 Rating");
		txtChannelRating.setBounds(206, 43, 108, 20);
		contentPanel.add(txtChannelRating);
```

```
txtChannelRating.setColumns(10);

JSeparator separator = new JSeparator();
separator.setBounds(141, 72, 221, 2);
contentPanel.add(separator);

JCheckBox chckbxChannel_1 = new JCheckBox("Channel 2");
chckbxChannel_1.setBounds(115, 81, 85, 23);
contentPanel.add(chckbxChannel_1);

JSeparator separator_2 = new JSeparator();
separator_2.setBounds(141, 111, 221, 2);
contentPanel.add(separator_2);

JCheckBox chckbxChannel_2 = new JCheckBox("Channel 3");
chckbxChannel_2.setBounds(115, 123, 85, 23);
contentPanel.add(chckbxChannel_2);

JSeparator separator_3 = new JSeparator();
separator_3.setBounds(141, 153, 221, 2);
contentPanel.add(separator_3);

JCheckBox chckbxChannel_3 = new JCheckBox("Channel 4");
chckbxChannel_3.setBounds(115, 162, 85, 23);
contentPanel.add(chckbxChannel_3);

JSeparator separator_4 = new JSeparator();
separator_4.setBounds(141, 192, 221, 2);
contentPanel.add(separator_4);

txtChannelRating_3 = new JTextField();
txtChannelRating_3.setText("Channel 2 Rating");
txtChannelRating_3.setHorizontalAlignment(SwingConstants.CENTER);
txtChannelRating_3.setColumns(10);
txtChannelRating_3.setBounds(206, 82, 108, 20);
contentPanel.add(txtChannelRating_3);

txtChannelRating_2 = new JTextField();
txtChannelRating_2.setText("Channel 3 Rating");
txtChannelRating_2.setHorizontalAlignment(SwingConstants.CENTER);
txtChannelRating_2.setColumns(10);
txtChannelRating_2.setBounds(206, 124, 108, 20);
contentPanel.add(txtChannelRating_2);

txtChannelRating_1 = new JTextField();
txtChannelRating_1.setText("Channel 4 Rating");
txtChannelRating_1.setHorizontalAlignment(SwingConstants.CENTER);
txtChannelRating_1.setColumns(10);
txtChannelRating_1.setBounds(206, 163, 108, 20);
contentPanel.add(txtChannelRating_1);

txtVOrI = new JTextField();
txtVOrI.setText("V or I");
txtVOrI.setHorizontalAlignment(SwingConstants.CENTER);
txtVOrI.setBounds(324, 43, 86, 20);
contentPanel.add(txtVOrI);
txtVOrI.setColumns(10);

txtVOrI_1 = new JTextField();
txtVOrI_1.setText("V or I");
txtVOrI_1.setHorizontalAlignment(SwingConstants.CENTER);
txtVOrI_1.setColumns(10);
txtVOrI_1.setBounds(324, 82, 86, 20);
contentPanel.add(txtVOrI_1);

textField_1 = new JTextField();
```

```
		textField_1.setText("V or I");
		textField_1.setHorizontalAlignment(SwingConstants.CENTER);
		textField_1.setColumns(10);
		textField_1.setBounds(324, 124, 86, 20);
		contentPanel.add(textField_1);

		textField_2 = new JTextField();
		textField_2.setText("V or I");
		textField_2.setHorizontalAlignment(SwingConstants.CENTER);
		textField_2.setColumns(10);
		textField_2.setBounds(324, 163, 86, 20);
		contentPanel.add(textField_2);
		{
			JPanel buttonPane = new JPanel();
			buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
			getContentPane().add(buttonPane, BorderLayout.SOUTH);
			{
				JButton okButton = new JButton("Finish");
				okButton.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent arg0)
					{
						if(chckbxChannel.isSelected()==true)
						{
							if(txtChannelRating.getText().contains("Channel"))
								Rating1=1;
							else

Rating1=Double.parseDouble(txtChannelRating.getText());
							Ch1 = true;
							System.out.print(txtVOrI.getText());
							if(txtVOrI.getText().contains("V"))
								Voltage1=true;
							else
								Voltage1=false;

						}
						else
						{
							Ch1 = false;
						}
						if(chckbxChannel_1.isSelected()==true)
						{
							if(txtChannelRating_3.getText().contains("Channel"))
								Rating2=1;
							else

Rating2=Double.parseDouble(txtChannelRating_3.getText());
							Ch2 = true;
							System.out.print(txtVOrI_1.getText());
							if(txtVOrI_1.getText().contains("V"))
								Voltage2=true;
							else
								Voltage2=false;
						}
						else
						{
							Ch2 = false;
						}
						if(chckbxChannel_2.isSelected()==true)
						{
							if(txtChannelRating_2.getText().contains("Channel"))
								Rating3=1;
							else

Rating3=Double.parseDouble(txtChannelRating_2.getText());
							Ch3 = true;
```

```
                                        System.out.print(textField_1.getText());
                                        if(textField_1.getText().contains("V"))
                                                Voltage3=true;
                                        else
                                                Voltage3=false;
                                }
                                else
                                {
                                        Ch3 = false;
                                }
                                if(chckbxChannel_3.isSelected()==true)
                                {
                                        if(txtChannelRating_1.getText().contains("Channel"))
                                                Rating4=1;
                                        else

        Rating4=Double.parseDouble(txtChannelRating_1.getText());
                                        Ch4 = true;
                                        if(textField_2.getText().contains("V"))
                                                Voltage4=true;
                                        else
                                                Voltage4=false;
                                }
                                else
                                {
                                        Ch4 = false;
                                }

                                try
                                {
                                        File ConfigFile = new File(MainWindow.FilePath());
                                        FileReader fr = new FileReader(ConfigFile);
                                        BufferedReader ConfigFileReader = new BufferedReader(fr);

                                        while ((line = ConfigFileReader.readLine()) != null)
                        {
                                                if(Ch1==true)
                                                {
                                                        if (line.contentEquals("//#define
Channel1"))
                                line = line.replace("//#define Channel1", "#define Channel1");
                                                        if (line.contains("#define CF01"))
                                                        {
                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                String f =
formatter.format(Rating1);
                                                                String temp2 = "\t #define CF01
";
                                                                temp2=temp2.concat(f);
                                                                line = temp2;
                                                        }
                                                }
                                                else
                                                {
                                                        if (line.contentEquals("#define Channel1"))
                                        line = line.replace("#define Channel1", "//#define Channel1");
                                                }
                                                if(Ch2==true)
                                                {
                                                        if (line.contentEquals("//#define
Channel2"))
                                line = line.replace("//#define Channel2", "#define Channel2");
                                                        if (line.contains("#define CF2"))
                                                        {
```

```
                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                String f =
formatter.format(Rating2);
                                                String temp2 = "\t #define CF2 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define Channel2"))
                        line = line.replace("#define Channel2", "//#define Channel2");
                                }
                                if(Ch3==true)
                                {
                                        if (line.contentEquals("//#define
Channel3"))
                        line = line.replace("//#define Channel3", "#define Channel3");
                                        if (line.contains("#define CF3"))
                                        {
                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                String f =
formatter.format(Rating3);
                                                String temp2 = "\t #define CF3 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define Channel3"))
                        line = line.replace("#define Channel3", "//#define Channel3");
                                }
                                if(Ch4==true)
                                {
                                        if (line.contentEquals("//#define
Channel4"))
                        line = line.replace("//#define Channel4", "#define Channel4");
                                        if (line.contains("#define CF4"))
                                        {
                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                String f =
formatter.format(Rating4);
                                                String temp2 = "\t #define CF4 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define Channel4"))
                        line = line.replace("#define Channel4", "//#define Channel4");
                                }

                                if(Voltage1==true)
                                {
                                        if (line.contains("//#define Voltage121"))
                                                line = line.replace("//#define
Voltage121", "#define Voltage121");
                                        if (line.contains("//#define Voltage131"))
                                                line = line.replace("//#define
Voltage131", "#define Voltage131");
                                        if (line.contains("//#define Voltage141"))
```

```
                                        line = line.replace("//#define
Voltage141", "#define Voltage141");
                                if (line.contains("#define Voltage122"))
                                        line = line.replace("#define
Voltage122", "//#define Voltage122");
                                if (line.contains("#define Voltage133"))
                                        line = line.replace("#define
Voltage133", "//#define Voltage133");
                                if (line.contains("#define Voltage144"))
                                        line = line.replace("#define
Voltage144", "//#define Voltage144");
                        }
                        else if(Voltage2==true)
                        {
                                if (line.contains("//#define Voltage122"))
                                        line = line.replace("//#define
Voltage122", "#define Voltage122");
                                if (line.contains("//#define Voltage232"))
                                        line = line.replace("//#define
Voltage232", "#define Voltage232");
                                if (line.contains("//#define Voltage242"))
                                        line = line.replace("//#define
Voltage242", "#define Voltage242");
                                if (line.contains("#define Voltage121"))
                                        line = line.replace("#define
Voltage121", "//#define Voltage121");
                                if (line.contains("#define Voltage233"))
                                        line = line.replace("#define
Voltage233", "//#define Voltage233");
                                if (line.contains("#define Voltage244"))
                                        line = line.replace("#define
Voltage244", "//#define Voltage244");
                        }
                        else if(Voltage3==true)
                        {
                                if (line.contains("//#define Voltage133"))
                                        line = line.replace("//#define
Voltage133", "#define Voltage133");
                                if (line.contains("//#define Voltage233"))
                                        line = line.replace("//#define
Voltage233", "#define Voltage233");
                                if (line.contains("//#define Voltage343"))
                                        line = line.replace("//#define
Voltage343", "#define Voltage343");
                                if (line.contains("#define Voltage131"))
                                        line = line.replace("#define
Voltage131", "//#define Voltage131");
                                if (line.contains("#define Voltage232"))
                                        line = line.replace("#define
Voltage232", "//#define Voltage232");
                                if (line.contains("#define Voltage344"))
                                        line = line.replace("#define
Voltage344", "//#define Voltage344");
                        }
                        else if(Voltage4==true)
                        {
                                if (line.contains("//#define Voltage144"))
                                        line = line.replace("//#define
Voltage144", "#define Voltage144");
                                if (line.contains("//#define Voltage244"))
                                        line = line.replace("//#define
Voltage244", "#define Voltage244");
                                if (line.contains("//#define Voltage344"))
                                        line = line.replace("//#define
Voltage344", "#define Voltage344");
                                if (line.contains("#define Voltage141"))
```

```
                                                                                line = line.replace("#define
Voltage141", "//#define Voltage141");
                                                                                if (line.contains("#define Voltage242"))
                                                                                line = line.replace("#define
Voltage242", "//#define Voltage242");
                                                                                if (line.contains("#define Voltage343"))
                                                                                line = line.replace("#define
Voltage343", "//#define Voltage343");
                                                                                }
                                                lines.add(line);
                                                }
                                                fr.close();
                                                ConfigFileReader.close();

                                                FileWriter fw = new FileWriter(ConfigFile);
                                                BufferedWriter ConfigFileWriter = new BufferedWriter(fw);
                                                for(String s : lines)
                                                ConfigFileWriter.write(s+"\n");
                                                ConfigFileWriter.flush();
                                                ConfigFileWriter.close();
                                                fw.close();
                                                }
                                                catch (Exception Ex)
                                                {
                                                Ex.printStackTrace();
                                                }

                                                InputConfiguration.this.setVisible(false);
                                                }
                                        });
                                        okButton.setActionCommand("OK");
                                        buttonPane.add(okButton);
                                        getRootPane().setDefaultButton(okButton);
                                }
                        }

        }
        public boolean getCheck(int i)
        {
                if(Ch1==true)
                        return true;
                else
                        return false;
        }

        public boolean getVoltage(int i)
        {
                if(Voltage1==true)
                        return true;
                else
                        return false;
        }
}
```

## A.4.3 Over Current Configuration

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import java.awt.event.ActionListener;
```

```
import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import java.awt.Font;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;

import javax.swing.JSeparator;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.SwingConstants;


public class OverCurrentConfiguration extends JDialog {

	private final JPanel contentPanel = new JPanel();
	private JTextField txtChannelLimit_1;
	private JTextField txtChannelLimit;
	private JTextField txtChannelLimit_2;
	private JTextField txtChannelLimit_3;
	private double Limit1;
	private double Limit2;
	private double Limit3;
	private double Limit4;
	boolean Ch1 = false;
	boolean Ch2 = false;
	boolean Ch3 = false;
	boolean Ch4 = false;

	ArrayList<String> lines = new ArrayList<String>();
  String line = null;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		try {
			OverCurrentConfiguration dialog = new OverCurrentConfiguration();
			dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
			dialog.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Create the dialog.
	 */
	public OverCurrentConfiguration() {
		setBounds(100, 100, 488, 413);
		getContentPane().setLayout(new BorderLayout());
		contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
		getContentPane().add(contentPanel, BorderLayout.CENTER);
		contentPanel.setLayout(null);

			JLabel lblNewLabel = new JLabel("Overcurrent Protection Configuration");
			lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));
			lblNewLabel.setBounds(89, 11, 294, 24);
			contentPanel.add(lblNewLabel);
```

```
        JSeparator separator = new JSeparator();
        separator.setBounds(78, 33, 318, 2);
        contentPanel.add(separator);

        txtChannelLimit_1 = new JTextField();
        txtChannelLimit_1.setText("Channel 1 Limit");
        txtChannelLimit_1.setHorizontalAlignment(SwingConstants.CENTER);
        txtChannelLimit_1.setColumns(10);
        txtChannelLimit_1.setBounds(236, 47, 108, 20);
        contentPanel.add(txtChannelLimit_1);

        JSeparator separator1 = new JSeparator();
        separator1.setBounds(135, 76, 221, 2);
        contentPanel.add(separator1);

        txtChannelLimit = new JTextField();
        txtChannelLimit.setText("Channel 2 Limit");
        txtChannelLimit.setHorizontalAlignment(SwingConstants.CENTER);
        txtChannelLimit.setColumns(10);
        txtChannelLimit.setBounds(236, 86, 108, 20);
        contentPanel.add(txtChannelLimit);

        txtChannelLimit_2 = new JTextField();
        txtChannelLimit_2.setText("Channel 3 Limit");
        txtChannelLimit_2.setHorizontalAlignment(SwingConstants.CENTER);
        txtChannelLimit_2.setColumns(10);
        txtChannelLimit_2.setBounds(236, 128, 108, 20);
        contentPanel.add(txtChannelLimit_2);

        JSeparator separator2 = new JSeparator();
        separator2.setBounds(135, 157, 221, 2);
        contentPanel.add(separator2);

        txtChannelLimit_3 = new JTextField();
        txtChannelLimit_3.setText("Channel 4 Limit");
        txtChannelLimit_3.setHorizontalAlignment(SwingConstants.CENTER);
        txtChannelLimit_3.setColumns(10);
        txtChannelLimit_3.setBounds(236, 167, 108, 20);
        contentPanel.add(txtChannelLimit_3);

        JSeparator separator3 = new JSeparator();
        separator3.setBounds(135, 196, 221, 2);
        contentPanel.add(separator3);

        JSeparator separator4 = new JSeparator();
        separator4.setBounds(135, 118, 221, 2);
        contentPanel.add(separator4);

        JCheckBox chckbxNewCheckBox = new JCheckBox("Channel 1");
        chckbxNewCheckBox.setBounds(145, 46, 85, 23);
        contentPanel.add(chckbxNewCheckBox);


        JCheckBox chckbxChannel = new JCheckBox("Channel 2");
        chckbxChannel.setBounds(145, 85, 85, 23);
        contentPanel.add(chckbxChannel);

        JCheckBox chckbxChannel_1 = new JCheckBox("Channel 3");
        chckbxChannel_1.setBounds(145, 127, 85, 23);
        contentPanel.add(chckbxChannel_1);

        JCheckBox chckbxChannel_2 = new JCheckBox("Channel 4");
        chckbxChannel_2.setBounds(145, 166, 85, 23);
        contentPanel.add(chckbxChannel_2);
{
```

```
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
getContentPane().add(buttonPane, BorderLayout.SOUTH);
{
	JButton okButton = new JButton("Finish");
	okButton.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e)
		{
			if(chckbxNewCheckBox.isSelected()==true)
			{
				Limit1=Double.parseDouble(txtChannelLimit_1.getText());
				Ch1 = true;

			}
			else
			{
				Ch1 = false;
			}
			if(chckbxChannel.isSelected()==true)
			{
				Limit2=Double.parseDouble(txtChannelLimit.getText());
				Ch2 = true;
			}
			else
			{
				Ch2 = false;
			}
			if(chckbxChannel_1.isSelected()==true)
			{
				Limit3=Double.parseDouble(txtChannelLimit_2.getText());
				Ch3 = true;
			}
			else
			{
				Ch3 = false;
			}
			if(chckbxChannel_2.isSelected()==true)
			{
				Limit4=Double.parseDouble(txtChannelLimit_3.getText());
				Ch4 = true;
			}
			else
			{
				Ch4 = false;
			}

			try
			{
				File ConfigFile = new File(MainWindow.FilePath());
				FileReader fr = new FileReader(ConfigFile);
				BufferedReader ConfigFileReader = new BufferedReader(fr);

				while ((line = ConfigFileReader.readLine()) != null)
	{
					if(Ch1==true)
					{
						if (line.contentEquals("//#define
OCChannel1"))
	line = line.replace("//#define OCChannel1", "#define OCChannel1");
						if (line.contains("#define OCLimit01"))
						{
							NumberFormat formatter = new
DecimalFormat("##########.###");
							String f =
formatter.format(Limit1);
```

```
                                                String temp2 = "\t #define
OCLimit01 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define
OCChannel1"))
                        line = line.replace("#define OCChannel1", "//#define OCChannel1");
                                }

                                if(Ch2==true)
                                {
                                        if (line.contentEquals("//#define
OCChannel2"))
                line = line.replace("//#define OCChannel2", "#define OCChannel2");
                                        if (line.contains("#define OCLimit2"))
                                        {
                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                String f =
formatter.format(Limit2);
                                                String temp2 = "\t #define
OCLimit2 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define
OCChannel2"))
                        line = line.replace("#define OCChannel2", "//#define OCChannel2");
                                }

                                if(Ch3==true)
                                {
                                        if (line.contentEquals("//#define
OCChannel3"))
                line = line.replace("//#define OCChannel3", "#define OCChannel3");
                                        if (line.contains("#define OCLimit3"))
                                        {
                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                String f =
formatter.format(Limit3);
                                                String temp2 = "\t #define
OCLimit3 ";
                                                temp2=temp2.concat(f);
                                                line = temp2;
                                        }
                                }
                                else
                                {
                                        if (line.contentEquals("#define
OCChannel3"))
                        line = line.replace("#define OCChannel3", "//#define OCChannel3");
                                }

                                if(Ch4==true)
                                {
                                        if (line.contentEquals("//#define
OCChannel4"))
                line = line.replace("//#define OCChannel4", "#define OCChannel4");
```

```
                                                                        if (line.contains("#define OCLimit4"))
                                                                        {
                                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                                String f =
formatter.format(Limit4);
                                                                                String temp2 = "\t #define
OCLimit4 ";
                                                                                temp2=temp2.concat(f);
                                                                                line = temp2;
                                                                        }
                                                                }
                                                                else
                                                                {
                                                                        if (line.contentEquals("#define
OCChannel4"))
                                                        line = line.replace("#define OCChannel4", "//#define OCChannel4");
                                                                }
                                                  lines.add(line);
                                                }
                                                                fr.close();
                                                                ConfigFileReader.close();

                                                                FileWriter fw = new FileWriter(ConfigFile);
                                                                BufferedWriter ConfigFileWriter = new BufferedWriter(fw);
                                                                for(String s : lines)
                                                   ConfigFileWriter.write(s+"\n");
                                                ConfigFileWriter.flush();
                                                ConfigFileWriter.close();
                                                fw.close();
                                                        }
                                                        catch (Exception Ex)
                                                        {
                                                                Ex.printStackTrace();
                                                        }
                                                        OverCurrentConfiguration.this.setVisible(false);
                                                    }
                                        });
                                        okButton.setActionCommand("Finish");
                                        buttonPane.add(okButton);
                                        getRootPane().setDefaultButton(okButton);
                                }
                        }
                }

}
```

## A.4.4 Distance Relay Configuration

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import javax.swing.JLabel;

import java.awt.Font;
import java.io.BufferedReader;
import java.io.BufferedWriter;
```

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;

import javax.swing.JSeparator;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.SwingConstants;


public class DistanceRelayConfiguration extends JDialog {

	private final JPanel contentPanel = new JPanel();
	private JTextField txtInnerLimit;
	private JTextField txtOuterLimit;
	private JTextField textField;
	private JTextField textField_1;
	private JTextField textField_2;
	private JTextField textField_3;
	private JTextField textField_4;
	private JTextField textField_5;
	private JTextField textField_6;
	private JTextField textField_7;
	private JTextField textField_8;
	private JTextField textField_9;

	private double ILimit12;
	private double OLimit12;
	private double ILimit13;
	private double OLimit13;
	private double ILimit14;
	private double OLimit14;
	private double ILimit23;
	private double OLimit23;
	private double ILimit24;
	private double OLimit24;
	private double ILimit34;
	private double OLimit34;
	boolean Ch12 = false;
	boolean Ch13 = false;
	boolean Ch14 = false;
	boolean Ch23 = false;
	boolean Ch24 = false;
	boolean Ch34 = false;

	ArrayList<String> lines = new ArrayList<String>();
  String line = null;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		try {
			DistanceRelayConfiguration dialog = new DistanceRelayConfiguration();
			dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
			dialog.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	/**
	 * Create the dialog.
```

```
 */
public DistanceRelayConfiguration() {
		setBounds(100, 100, 500, 416);
		getContentPane().setLayout(new BorderLayout());
		contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
		getContentPane().add(contentPanel, BorderLayout.CENTER);
		contentPanel.setLayout(null);

			JLabel lblNewLabel = new JLabel("Distance Relay Configuration");
			lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 20));
			lblNewLabel.setBounds(125, 11, 233, 24);
			contentPanel.add(lblNewLabel);

			JSeparator separator = new JSeparator();
			separator.setBounds(113, 33, 257, 2);
			contentPanel.add(separator);

			JCheckBox chckbxDistanceRelayCh12 = new JCheckBox("Distance Relay Ch 1/2");
			chckbxDistanceRelayCh12.setBounds(56, 42, 178, 23);
			contentPanel.add(chckbxDistanceRelayCh12);

			txtInnerLimit = new JTextField();
			txtInnerLimit.setHorizontalAlignment(SwingConstants.CENTER);
			txtInnerLimit.setText("Inner Limit");
			txtInnerLimit.setBounds(240, 41, 86, 20);
			contentPanel.add(txtInnerLimit);
			txtInnerLimit.setColumns(10);

			txtOuterLimit = new JTextField();
			txtOuterLimit.setText("Outer Limit");
			txtOuterLimit.setHorizontalAlignment(SwingConstants.CENTER);
			txtOuterLimit.setColumns(10);
			txtOuterLimit.setBounds(336, 41, 86, 20);
			contentPanel.add(txtOuterLimit);

			JSeparator separator1 = new JSeparator();
			separator1.setBounds(125, 72, 233, 2);
			contentPanel.add(separator1);

			JCheckBox chckbxDistanceRelayCh13 = new JCheckBox("Distance Relay Ch 1/3");
			chckbxDistanceRelayCh13.setBounds(56, 81, 178, 23);
			contentPanel.add(chckbxDistanceRelayCh13);

			textField = new JTextField();
			textField.setText("Inner Limit");
			textField.setHorizontalAlignment(SwingConstants.CENTER);
			textField.setColumns(10);
			textField.setBounds(240, 80, 86, 20);
			contentPanel.add(textField);

			textField_1 = new JTextField();
			textField_1.setText("Outer Limit");
			textField_1.setHorizontalAlignment(SwingConstants.CENTER);
			textField_1.setColumns(10);
			textField_1.setBounds(336, 80, 86, 20);
			contentPanel.add(textField_1);

			JSeparator separator2 = new JSeparator();
			separator2.setBounds(125, 111, 233, 2);
			contentPanel.add(separator2);

			JSeparator separator3 = new JSeparator();
			separator3.setBounds(125, 154, 233, 2);
			contentPanel.add(separator3);

			JCheckBox chckbxDistanceRelayCh14 = new JCheckBox("Distance Relay Ch 1/4");
```

```
chckbxDistanceRelayCh14.setBounds(56, 124, 178, 23);
contentPanel.add(chckbxDistanceRelayCh14);

textField_2 = new JTextField();
textField_2.setText("Inner Limit");
textField_2.setHorizontalAlignment(SwingConstants.CENTER);
textField_2.setColumns(10);
textField_2.setBounds(240, 123, 86, 20);
contentPanel.add(textField_2);

textField_3 = new JTextField();
textField_3.setText("Outer Limit");
textField_3.setHorizontalAlignment(SwingConstants.CENTER);
textField_3.setColumns(10);
textField_3.setBounds(336, 123, 86, 20);
contentPanel.add(textField_3);

JSeparator separator4 = new JSeparator();
separator4.setBounds(125, 197, 233, 2);
contentPanel.add(separator4);

JCheckBox chckbxDistanceRelayCh23 = new JCheckBox("Distance Relay Ch 2/3");
chckbxDistanceRelayCh23.setBounds(56, 167, 178, 23);
contentPanel.add(chckbxDistanceRelayCh23);

textField_4 = new JTextField();
textField_4.setText("Inner Limit");
textField_4.setHorizontalAlignment(SwingConstants.CENTER);
textField_4.setColumns(10);
textField_4.setBounds(240, 166, 86, 20);
contentPanel.add(textField_4);

textField_5 = new JTextField();
textField_5.setText("Outer Limit");
textField_5.setHorizontalAlignment(SwingConstants.CENTER);
textField_5.setColumns(10);
textField_5.setBounds(336, 166, 86, 20);
contentPanel.add(textField_5);

JSeparator separator5 = new JSeparator();
separator5.setBounds(125, 240, 233, 2);
contentPanel.add(separator5);

JCheckBox chckbxDistanceRelayCh24 = new JCheckBox("Distance Relay Ch 2/4");
chckbxDistanceRelayCh24.setBounds(56, 210, 178, 23);
contentPanel.add(chckbxDistanceRelayCh24);

textField_6 = new JTextField();
textField_6.setText("Inner Limit");
textField_6.setHorizontalAlignment(SwingConstants.CENTER);
textField_6.setColumns(10);
textField_6.setBounds(240, 209, 86, 20);
contentPanel.add(textField_6);

textField_7 = new JTextField();
textField_7.setText("Outer Limit");
textField_7.setHorizontalAlignment(SwingConstants.CENTER);
textField_7.setColumns(10);
textField_7.setBounds(336, 209, 86, 20);
contentPanel.add(textField_7);

JSeparator separator6 = new JSeparator();
separator6.setBounds(125, 283, 233, 2);
contentPanel.add(separator6);

JCheckBox chckbxDistanceRelayCh34 = new JCheckBox("Distance Relay Ch 3/4");
```

```
		chckbxDistanceRelayCh34.setBounds(56, 253, 178, 23);
		contentPanel.add(chckbxDistanceRelayCh34);

		textField_8 = new JTextField();
		textField_8.setText("Inner Limit");
		textField_8.setHorizontalAlignment(SwingConstants.CENTER);
		textField_8.setColumns(10);
		textField_8.setBounds(240, 252, 86, 20);
		contentPanel.add(textField_8);

		textField_9 = new JTextField();
		textField_9.setText("Outer Limit");
		textField_9.setHorizontalAlignment(SwingConstants.CENTER);
		textField_9.setColumns(10);
		textField_9.setBounds(336, 252, 86, 20);
		contentPanel.add(textField_9);
{
		JPanel buttonPane = new JPanel();
		buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
		getContentPane().add(buttonPane, BorderLayout.SOUTH);
		{
			JButton okButton = new JButton("Finish");
			okButton.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e)
				{
					if(chckbxDistanceRelayCh12.isSelected()==true)
					{
						ILimit12=Double.parseDouble(txtInnerLimit.getText());
						OLimit12=Double.parseDouble(txtOuterLimit.getText());
						Ch12 = true;

					}
					else
					{
						Ch12 = false;
					}

					if(chckbxDistanceRelayCh13.isSelected()==true)
					{
						ILimit13=Double.parseDouble(textField.getText());
						OLimit13=Double.parseDouble(textField_1.getText());
						Ch13 = true;

					}
					else
					{
						Ch13 = false;
					}

					if(chckbxDistanceRelayCh14.isSelected()==true)
					{
						ILimit14=Double.parseDouble(textField_2.getText());
						OLimit14=Double.parseDouble(textField_3.getText());
						Ch14 = true;

					}
					else
					{
						Ch14 = false;
					}

					if(chckbxDistanceRelayCh23.isSelected()==true)
					{
						ILimit23=Double.parseDouble(textField_4.getText());
						OLimit23=Double.parseDouble(textField_5.getText());
						Ch23 = true;
```

```
}
else
{
	Ch23 = false;
}

if(chckbxDistanceRelayCh24.isSelected()==true)
{
	ILimit24=Double.parseDouble(textField_6.getText());
	OLimit24=Double.parseDouble(textField_7.getText());
	Ch24 = true;

}
else
{
	Ch24 = false;
}

if(chckbxDistanceRelayCh34.isSelected()==true)
{
	ILimit34=Double.parseDouble(textField_8.getText());
	OLimit34=Double.parseDouble(textField_9.getText());
	Ch34 = true;

}
else
{
	Ch34 = false;
}
try
{
	File ConfigFile = new File(MainWindow.FilePath());
	FileReader fr = new FileReader(ConfigFile);
	BufferedReader ConfigFileReader = new BufferedReader(fr);

	while ((line = ConfigFileReader.readLine()) != null)
{
		if(Ch12==true)
		{
			if (line.contentEquals("//#define
Distance12"))
    line = line.replace("//#define Distance12", "#define Distance12");
			if (line.contains("#define DLimitInner12"))
			{
				NumberFormat formatter = new
DecimalFormat("##########.###");
				String f =
formatter.format(ILimit12);
				String temp2 = "\t #define
DLimitInner12 ";
				temp2=temp2.concat(f);
				line = temp2;
			}
			if (line.contains("#define DLimitOuter12"))
			{
				NumberFormat formatter = new
DecimalFormat("##########.###");
				String f =
formatter.format(OLimit12);
				String temp2 = "\t #define
DLimitOuter12 ";
				temp2=temp2.concat(f);
				line = temp2;
			}
		}
```

```
                                                                else
                                                                {
                                                                        if (line.contentEquals("#define
Distance12"))
                                                        line = line.replace("#define Distance12", "//#define Distance12");
                                                                }

                                                                if(Ch13==true)
                                                                {
                                                                        if (line.contentEquals("//#define
Distance13"))
                                                line = line.replace("//#define Distance13", "#define Distance13");
                                                                        if (line.contains("#define DLimitInner13"))
                                                                        {
                                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                                String f =
formatter.format(ILimit13);
                                                                                String temp2 = "\t #define
DLimitInner13 ";
                                                                                temp2=temp2.concat(f);
                                                                                line = temp2;
                                                                        }
                                                                        if (line.contains("#define DLimitOuter13"))
                                                                        {
                                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                                String f =
formatter.format(OLimit13);
                                                                                String temp2 = "\t #define
DLimitOuter13 ";
                                                                                temp2=temp2.concat(f);
                                                                                line = temp2;
                                                                        }
                                                                }
                                                                else
                                                                {
                                                                        if (line.contentEquals("#define
Distance13"))
                                                        line = line.replace("#define Distance13", "//#define Distance13");
                                                                }

                                                                if(Ch14==true)
                                                                {
                                                                        if (line.contentEquals("//#define
Distance14"))
                                                line = line.replace("//#define Distance14", "#define Distance14");
                                                                        if (line.contains("#define DLimitInner14"))
                                                                        {
                                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                                String f =
formatter.format(ILimit14);
                                                                                String temp2 = "\t #define
DLimitInner14 ";
                                                                                temp2=temp2.concat(f);
                                                                                line = temp2;
                                                                        }
                                                                        if (line.contains("#define DLimitOuter14"))
                                                                        {
                                                                                NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                                String f =
formatter.format(OLimit14);
                                                                                String temp2 = "\t #define
DLimitOuter14 ";
```

```
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                        }
                                                        else
                                                        {
                                                                if (line.contentEquals("#define
Distance14"))
                                                        line = line.replace("#define Distance14", "//#define Distance14");
                                                        }

                                                        if(Ch23==true)
                                                        {
                                                                if (line.contentEquals("//#define
Distance23"))
                                                line = line.replace("//#define Distance23", "#define Distance23");
                                                                if (line.contains("#define DLimitInner23"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                        String f =
formatter.format(ILimit23);
                                                                        String temp2 = "\t #define
DLimitInner23 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                                if (line.contains("#define DLimitOuter23"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                        String f =
formatter.format(OLimit23);
                                                                        String temp2 = "\t #define
DLimitOuter23 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                        }
                                                        else
                                                        {
                                                                if (line.contentEquals("#define
Distance23"))
                                                        line = line.replace("#define Distance23", "//#define Distance23");
                                                        }

                                                        if(Ch24==true)
                                                        {
                                                                if (line.contentEquals("//#define
Distance24"))
                                                line = line.replace("//#define Distance24", "#define Distance24");
                                                                if (line.contains("#define DLimitInner24"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                        String f =
formatter.format(ILimit24);
                                                                        String temp2 = "\t #define
DLimitInner24 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                                if (line.contains("#define DLimitOuter24"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
```

```
                                                                        String f =
formatter.format(OLimit24);
                                                                        String temp2 = "\t #define
DLimitOuter24 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                        }
                                                        else
                                                        {
                                                                if (line.contentEquals("#define
Distance24"))
                                                line = line.replace("#define Distance24", "//#define Distance24");
                                                        }

                                                        if(Ch34==true)
                                                        {
                                                                if (line.contentEquals("//#define
Distance34"))
                                        line = line.replace("//#define Distance34", "#define Distance34");
                                                                if (line.contains("#define DLimitInner34"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                        String f =
formatter.format(ILimit34);
                                                                        String temp2 = "\t #define
DLimitInner34 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                                if (line.contains("#define DLimitOuter34"))
                                                                {
                                                                        NumberFormat formatter = new
DecimalFormat("##########.###");
                                                                        String f =
formatter.format(OLimit34);
                                                                        String temp2 = "\t #define
DLimitOuter34 ";
                                                                        temp2=temp2.concat(f);
                                                                        line = temp2;
                                                                }
                                                        }
                                                        else
                                                        {
                                                                if (line.contentEquals("#define
Distance34"))
                                                line = line.replace("#define Distance34", "//#define Distance34");
                                                        }

                                        lines.add(line);
                                }
                                                fr.close();
                                                ConfigFileReader.close();

                                                FileWriter fw = new FileWriter(ConfigFile);
                                                BufferedWriter ConfigFileWriter = new BufferedWriter(fw);
                                                for(String s : lines)
                                        ConfigFileWriter.write(s+"\n");
                                ConfigFileWriter.flush();
                                ConfigFileWriter.close();
                                fw.close();
                                        }
                                        catch (Exception Ex)
                                        {
                                                Ex.printStackTrace();
```

```
                                                }

                                                DistanceRelayConfiguration.this.setVisible(false);
                                        }
                                });
                                okButton.setActionCommand("OK");
                                buttonPane.add(okButton);
                                getRootPane().setDefaultButton(okButton);
                        }
                }
        }

}
```

## A.4.5 Output Window

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Panel;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.SocketException;
import java.util.ArrayList;
import java.util.Observable;

import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.Timer;
import javax.swing.border.BevelBorder;
import javax.swing.border.EmptyBorder;
import javax.swing.border.LineBorder;


public class OutputWindow extends JDialog {

        private PropertyChangeSupport changes = new PropertyChangeSupport(this);
        private final JPanel contentPanel = new JPanel();
        private JTextField Mag1TxtField;
        private JTextField Ang1TxtField;
        private JTextField OC1TxtField;
        boolean running = false;
        private int port1 = 20001;
        private int Mag1;
        private int Ang1;
        private int OCTrip1;
        private String Mag1s;
        private String Ang1s;
        private int port2 = 20002;
        private int Mag2;
        private int Ang2;
```

```
        private int OCTrip2;
        private String Mag2s;
        private String Ang2s;
        private int port3 = 20003;
        private int Mag3;
        private int Ang3;
        private int OCTrip3;
        private String Mag3s;
        private String Ang3s;
        private int port4 = 20004;
        private int Mag4;
        private int Ang4;
        private int OCTrip4;
        private String Mag4s;
        private String Ang4s;
        private int port12 = 40012;
        private int DITrip12;
        private int DOTrip12;
        private int port13 = 40013;
        private int DITrip13;
        private int DOTrip13;
        private int port14 = 40014;
        private int DITrip14;
        private int DOTrip14;
        private int port23 = 40023;
        private int DITrip23;
        private int DOTrip23;
        private int port24 = 40024;
        private int DITrip24;
        private int DOTrip24;
        private int port34 = 40034;
        private int DITrip34;
        private int DOTrip34;
        private JTextField OC2TxtField;
        private JTextField OC3TxtField;
        private JTextField Ang3TxtField;
        private JTextField Mag3TxtField;
        private JTextField OC4TxtField;
        private JTextField Ang4TxtField;
        private JTextField Mag4TxtField;
        private boolean Channel1=false;
        private boolean Channel2=false;
        private boolean Channel3=false;
        private boolean Channel4=false;
        private boolean D12=false;
        private boolean D13=false;
        private boolean D14=false;
        private boolean D23=false;
        private boolean D24=false;
        private boolean D34=false;
        Timer timer;

        ArrayList<String> lines = new ArrayList<String>();
String line = null;
private JTextField D12ITxtField;
private JTextField D12OTxtField;
private JTextField Mag2TxtField;
private JTextField Ang2TxtField;
private JTextField D14ITxtField;
private JTextField D14OTxtField;
private JTextField D13ITxtField;
private JTextField D13OTxtField;
private JTextField D23ITxtField;
private JTextField D23OTxtField;
private JTextField D24ITxtField;
private JTextField D24OTxtField;
```

```
private JTextField D34ITxtField;
private JTextField D34OTxtField;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		try {
			OutputWindow dialog = new OutputWindow();
			dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
			dialog.setVisible(true);
		} catch (Exception e) {
			e.printStackTrace();
		}

	}

	/**
	 * Create the dialog.
	 */
	public OutputWindow() {

		setBounds(100, 100, 700, 650);
		getContentPane().setLayout(new BorderLayout());
		contentPanel.setBorder(new EmptyBorder(1, 1, 1, 1));
		getContentPane().add(contentPanel, BorderLayout.CENTER);
		contentPanel.setLayout(null);

			JLabel lblNewLabel = new JLabel("Output Window");
			lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
			lblNewLabel.setFont(new Font("Times New Roman", Font.PLAIN, 40));
			lblNewLabel.setBounds(5, 11, 673, 47);
			contentPanel.add(lblNewLabel);

			JPanel panel = new JPanel();
			panel.setBorder(new BevelBorder(BevelBorder.RAISED, new Color(0, 0, 0), new Color(0, 0, 0), Color.BLACK,
Color.BLACK));
			panel.setBackground(Color.WHITE);
			panel.setBounds(43, 119, 250, 100);
			contentPanel.add(panel);
			panel.setLayout(null);

			JLabel lblMagnitude = new JLabel("Magnitude:");
			lblMagnitude.setBounds(18, 22, 64, 14);
			panel.add(lblMagnitude);

			JLabel lblAngle = new JLabel("Angle:");
			lblAngle.setBounds(41, 47, 41, 14);
			panel.add(lblAngle);

			Mag1TxtField = new JTextField();
			Mag1TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
			Mag1TxtField.setEditable(false);
			Mag1TxtField.setBounds(92, 19, 86, 20);
			panel.add(Mag1TxtField);
			Mag1TxtField.setColumns(10);

			Ang1TxtField = new JTextField();
			Ang1TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
			Ang1TxtField.setEditable(false);
			Ang1TxtField.setBounds(92, 44, 86, 20);
			panel.add(Ang1TxtField);
			Ang1TxtField.setColumns(10);

			JLabel lblChannel = new JLabel("Channel 1");
			lblChannel.setFont(new Font("Times New Roman", Font.PLAIN, 14));
```

```
lblChannel.setHorizontalAlignment(SwingConstants.CENTER);
lblChannel.setBounds(82, 4, 86, 14);
panel.add(lblChannel);

JLabel lblTripCondition = new JLabel("O/C Relay:");
lblTripCondition.setHorizontalAlignment(SwingConstants.TRAILING);
lblTripCondition.setBounds(10, 72, 72, 14);
panel.add(lblTripCondition);

OC1TxtField = new JTextField();
OC1TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
OC1TxtField.setEditable(false);
OC1TxtField.setBounds(92, 69, 86, 20);
panel.add(OC1TxtField);
OC1TxtField.setColumns(10);

Panel Trip1Color = new Panel();
Trip1Color.setBackground(Color.GREEN);
Trip1Color.setBounds(186, 76, 10, 10);
panel.add(Trip1Color);

JPanel panel1 = new JPanel();
panel1.setBorder(new BevelBorder(BevelBorder.RAISED, new Color(0, 0, 0), new Color(0, 0, 0), Color.BLACK,
Color.BLACK));
panel1.setBackground(Color.WHITE);
panel1.setBounds(43, 230, 250, 100);
contentPanel.add(panel1);
panel1.setLayout(null);

Panel Trip2Color = new Panel();
Trip2Color.setBackground(Color.GREEN);
Trip2Color.setBounds(186, 72, 10, 10);
panel1.add(Trip2Color);

OC2TxtField = new JTextField();
OC2TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
OC2TxtField.setEditable(false);
OC2TxtField.setColumns(10);
OC2TxtField.setBounds(92, 65, 86, 20);
panel1.add(OC2TxtField);

JLabel lblChannel_1 = new JLabel("Channel 2");
lblChannel_1.setHorizontalAlignment(SwingConstants.CENTER);
lblChannel_1.setFont(new Font("Times New Roman", Font.PLAIN, 14));
lblChannel_1.setBounds(82, 0, 86, 14);
panel1.add(lblChannel_1);

JLabel label_1 = new JLabel("Magnitude:");
label_1.setBounds(18, 18, 64, 14);
panel1.add(label_1);

JLabel label_2 = new JLabel("Angle:");
label_2.setBounds(41, 43, 41, 14);
panel1.add(label_2);

JLabel label_3 = new JLabel("O/C Relay:");
label_3.setHorizontalAlignment(SwingConstants.TRAILING);
label_3.setBounds(10, 68, 72, 14);
panel1.add(label_3);

Mag2TxtField = new JTextField();
Mag2TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
Mag2TxtField.setEditable(false);
Mag2TxtField.setColumns(10);
Mag2TxtField.setBounds(92, 15, 86, 20);
panel1.add(Mag2TxtField);
```

```
                        Ang2TxtField = new JTextField();
                        Ang2TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                        Ang2TxtField.setEditable(false);
                        Ang2TxtField.setColumns(10);
                        Ang2TxtField.setBounds(92, 40, 86, 20);
                        panel1.add(Ang2TxtField);

                        JPanel panel2 = new JPanel();
                        panel2.setBorder(new BevelBorder(BevelBorder.RAISED, new Color(0, 0, 0), new Color(0, 0, 0), Color.BLACK,
Color.BLACK));
                        panel2.setBackground(Color.WHITE);
                        panel2.setBounds(43, 341, 250, 100);
                        contentPanel.add(panel2);
                        panel2.setLayout(null);

                        Panel Trip3Color = new Panel();
                        Trip3Color.setBackground(Color.GREEN);
                        Trip3Color.setBounds(186, 72, 10, 10);
                        panel2.add(Trip3Color);

                        OC3TxtField = new JTextField();
                        OC3TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                        OC3TxtField.setEditable(false);
                        OC3TxtField.setColumns(10);
                        OC3TxtField.setBounds(92, 65, 86, 20);
                        panel2.add(OC3TxtField);

                        Ang3TxtField = new JTextField();
                        Ang3TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                        Ang3TxtField.setEditable(false);
                        Ang3TxtField.setColumns(10);
                        Ang3TxtField.setBounds(92, 40, 86, 20);
                        panel2.add(Ang3TxtField);

                        Mag3TxtField = new JTextField();
                        Mag3TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                        Mag3TxtField.setEditable(false);
                        Mag3TxtField.setColumns(10);
                        Mag3TxtField.setBounds(92, 15, 86, 20);
                        panel2.add(Mag3TxtField);

                        JLabel lblChannel_2 = new JLabel("Channel 3");
                        lblChannel_2.setHorizontalAlignment(SwingConstants.CENTER);
                        lblChannel_2.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                        lblChannel_2.setBounds(82, 0, 86, 14);
                        panel2.add(lblChannel_2);

                        JLabel label_4 = new JLabel("Magnitude:");
                        label_4.setBounds(18, 18, 64, 14);
                        panel2.add(label_4);

                        JLabel label_5 = new JLabel("Angle:");
                        label_5.setBounds(41, 43, 41, 14);
                        panel2.add(label_5);

                        JLabel label_6 = new JLabel("O/C Relay:");
                        label_6.setHorizontalAlignment(SwingConstants.TRAILING);
                        label_6.setBounds(10, 68, 72, 14);
                        panel2.add(label_6);

                        JPanel panel3 = new JPanel();
                        panel3.setBorder(new BevelBorder(BevelBorder.RAISED, new Color(0, 0, 0), new Color(0, 0, 0), Color.BLACK,
Color.BLACK));
                        panel3.setBackground(Color.WHITE);
                        panel3.setBounds(43, 452, 250, 100);
```

```
                contentPanel.add(panel3);
                panel3.setLayout(null);

                Panel Trip4Color = new Panel();
                Trip4Color.setBackground(Color.GREEN);
                Trip4Color.setBounds(186, 72, 10, 10);
                panel3.add(Trip4Color);

                OC4TxtField = new JTextField();
                OC4TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                OC4TxtField.setEditable(false);
                OC4TxtField.setColumns(10);
                OC4TxtField.setBounds(92, 65, 86, 20);
                panel3.add(OC4TxtField);

                Ang4TxtField = new JTextField();
                Ang4TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                Ang4TxtField.setEditable(false);
                Ang4TxtField.setColumns(10);
                Ang4TxtField.setBounds(92, 40, 86, 20);
                panel3.add(Ang4TxtField);

                Mag4TxtField = new JTextField();
                Mag4TxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                Mag4TxtField.setEditable(false);
                Mag4TxtField.setColumns(10);
                Mag4TxtField.setBounds(92, 15, 86, 20);
                panel3.add(Mag4TxtField);

                JLabel lblChannel_3 = new JLabel("Channel 4");
                lblChannel_3.setHorizontalAlignment(SwingConstants.CENTER);
                lblChannel_3.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                lblChannel_3.setBounds(82, 0, 86, 14);
                panel3.add(lblChannel_3);

                JLabel label_7 = new JLabel("Magnitude:");
                label_7.setBounds(18, 18, 64, 14);
                panel3.add(label_7);

                JLabel label_8 = new JLabel("Angle:");
                label_8.setBounds(41, 43, 41, 14);
                panel3.add(label_8);

                JLabel label_9 = new JLabel("O/C Relay:");
                label_9.setHorizontalAlignment(SwingConstants.TRAILING);
                label_9.setBounds(10, 68, 72, 14);
                panel3.add(label_9);

                JSeparator separator = new JSeparator();
                separator.setBounds(10, 56, 668, 2);
                contentPanel.add(separator);

                JPanel panel_1 = new JPanel();
                panel_1.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
                panel_1.setBackground(Color.WHITE);
                panel_1.setBounds(342, 119, 125, 125);
                contentPanel.add(panel_1);
                panel_1.setLayout(null);

                JLabel Distance_12 = new JLabel("Distance 1-2");
                Distance_12.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                Distance_12.setHorizontalAlignment(SwingConstants.CENTER);
                Distance_12.setBounds(10, 11, 105, 14);
                panel_1.add(Distance_12);
```

```
JLabel lblInner = new JLabel("Inner:");
lblInner.setHorizontalAlignment(SwingConstants.TRAILING);
lblInner.setBounds(10, 32, 35, 14);
panel_1.add(lblInner);

JLabel lblNewLabel_1 = new JLabel("Outer:");
lblNewLabel_1.setHorizontalAlignment(SwingConstants.TRAILING);
lblNewLabel_1.setBounds(10, 78, 35, 14);
panel_1.add(lblNewLabel_1);

D12ITxtField = new JTextField();
D12ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D12ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D12ITxtField.setEditable(false);
D12ITxtField.setBounds(35, 46, 80, 21);
panel_1.add(D12ITxtField);
D12ITxtField.setColumns(10);

D12OTxtField = new JTextField();
D12OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D12OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D12OTxtField.setEditable(false);
D12OTxtField.setColumns(10);
D12OTxtField.setBounds(35, 92, 80, 22);
panel_1.add(D12OTxtField);

JPanel panel_2 = new JPanel();
panel_2.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
panel_2.setBackground(Color.WHITE);
panel_2.setBounds(343, 427, 125, 125);
contentPanel.add(panel_2);
panel_2.setLayout(null);

JLabel label = new JLabel("Distance 1-4");
label.setHorizontalAlignment(SwingConstants.CENTER);
label.setFont(new Font("Times New Roman", Font.PLAIN, 14));
label.setBounds(10, 11, 105, 14);
panel_2.add(label);

JLabel label_12 = new JLabel("Inner:");
label_12.setHorizontalAlignment(SwingConstants.TRAILING);
label_12.setBounds(10, 32, 35, 14);
panel_2.add(label_12);

D14ITxtField = new JTextField();
D14ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D14ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D14ITxtField.setEditable(false);
D14ITxtField.setColumns(10);
D14ITxtField.setBounds(35, 46, 80, 21);
panel_2.add(D14ITxtField);

JLabel label_13 = new JLabel("Outer:");
label_13.setHorizontalAlignment(SwingConstants.TRAILING);
label_13.setBounds(10, 78, 35, 14);
panel_2.add(label_13);

D14OTxtField = new JTextField();
D14OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D14OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D14OTxtField.setEditable(false);
D14OTxtField.setColumns(10);
D14OTxtField.setBounds(35, 92, 80, 22);
panel_2.add(D14OTxtField);
```

```
JPanel panel_3 = new JPanel();
panel_3.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
panel_3.setBackground(Color.WHITE);
panel_3.setBounds(342, 270, 125, 125);
contentPanel.add(panel_3);
panel_3.setLayout(null);

JLabel label_10 = new JLabel("Distance 1-3");
label_10.setHorizontalAlignment(SwingConstants.CENTER);
label_10.setFont(new Font("Times New Roman", Font.PLAIN, 14));
label_10.setBounds(10, 11, 105, 14);
panel_3.add(label_10);

JLabel label_11 = new JLabel("Inner:");
label_11.setHorizontalAlignment(SwingConstants.TRAILING);
label_11.setBounds(10, 32, 35, 14);
panel_3.add(label_11);

D13ITxtField = new JTextField();
D13ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D13ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D13ITxtField.setEditable(false);
D13ITxtField.setColumns(10);
D13ITxtField.setBounds(35, 46, 80, 21);
panel_3.add(D13ITxtField);

JLabel label_14 = new JLabel("Outer:");
label_14.setHorizontalAlignment(SwingConstants.TRAILING);
label_14.setBounds(10, 78, 35, 14);
panel_3.add(label_14);

D13OTxtField = new JTextField();
D13OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D13OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D13OTxtField.setEditable(false);
D13OTxtField.setColumns(10);
D13OTxtField.setBounds(35, 92, 80, 22);
panel_3.add(D13OTxtField);

JPanel panel_4 = new JPanel();
panel_4.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
panel_4.setBackground(Color.WHITE);
panel_4.setBounds(511, 119, 125, 125);
contentPanel.add(panel_4);
panel_4.setLayout(null);

JLabel lblDistance = new JLabel("Distance 2-3");
lblDistance.setHorizontalAlignment(SwingConstants.CENTER);
lblDistance.setFont(new Font("Times New Roman", Font.PLAIN, 14));
lblDistance.setBounds(10, 11, 105, 14);
panel_4.add(lblDistance);

JLabel label_16 = new JLabel("Inner:");
label_16.setHorizontalAlignment(SwingConstants.TRAILING);
label_16.setBounds(10, 32, 35, 14);
panel_4.add(label_16);

D23ITxtField = new JTextField();
D23ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D23ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D23ITxtField.setEditable(false);
D23ITxtField.setColumns(10);
D23ITxtField.setBounds(35, 46, 80, 21);
panel_4.add(D23ITxtField);
```

```
                JLabel label_17 = new JLabel("Outer:");
                label_17.setHorizontalAlignment(SwingConstants.TRAILING);
                label_17.setBounds(10, 78, 35, 14);
                panel_4.add(label_17);

                D23OTxtField = new JTextField();
                D23OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                D23OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
                D23OTxtField.setEditable(false);
                D23OTxtField.setColumns(10);
                D23OTxtField.setBounds(35, 92, 80, 22);
                panel_4.add(D23OTxtField);

                JPanel panel_5 = new JPanel();
                panel_5.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
                panel_5.setBackground(Color.WHITE);
                panel_5.setBounds(511, 270, 125, 125);
                contentPanel.add(panel_5);
                panel_5.setLayout(null);

                JLabel lblDistance_1 = new JLabel("Distance 2-4");
                lblDistance_1.setHorizontalAlignment(SwingConstants.CENTER);
                lblDistance_1.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                lblDistance_1.setBounds(10, 11, 105, 14);
                panel_5.add(lblDistance_1);

                JLabel label_18 = new JLabel("Inner:");
                label_18.setHorizontalAlignment(SwingConstants.TRAILING);
                label_18.setBounds(10, 32, 35, 14);
                panel_5.add(label_18);

                D24ITxtField = new JTextField();
                D24ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                D24ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
                D24ITxtField.setEditable(false);
                D24ITxtField.setColumns(10);
                D24ITxtField.setBounds(35, 46, 80, 21);
                panel_5.add(D24ITxtField);

                JLabel label_19 = new JLabel("Outer:");
                label_19.setHorizontalAlignment(SwingConstants.TRAILING);
                label_19.setBounds(10, 78, 35, 14);
                panel_5.add(label_19);

                D24OTxtField = new JTextField();
                D24OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
                D24OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
                D24OTxtField.setEditable(false);
                D24OTxtField.setColumns(10);
                D24OTxtField.setBounds(35, 92, 80, 22);
                panel_5.add(D24OTxtField);

                JPanel panel_6 = new JPanel();
                panel_6.setBorder(new BevelBorder(BevelBorder.LOWERED, new Color(0, 0, 0), new Color(0, 0, 0),
Color.BLACK, Color.BLACK));
                panel_6.setBackground(Color.WHITE);
                panel_6.setBounds(511, 427, 125, 125);
                contentPanel.add(panel_6);
                panel_6.setLayout(null);

                JLabel lblDistance_2 = new JLabel("Distance 3-4");
                lblDistance_2.setHorizontalAlignment(SwingConstants.CENTER);
                lblDistance_2.setFont(new Font("Times New Roman", Font.PLAIN, 14));
                lblDistance_2.setBounds(10, 11, 105, 14);
```

```
panel_6.add(lblDistance_2);

JLabel label_20 = new JLabel("Inner:");
label_20.setHorizontalAlignment(SwingConstants.TRAILING);
label_20.setBounds(10, 32, 35, 14);
panel_6.add(label_20);

D34ITxtField = new JTextField();
D34ITxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D34ITxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D34ITxtField.setEditable(false);
D34ITxtField.setColumns(10);
D34ITxtField.setBounds(35, 46, 80, 21);
panel_6.add(D34ITxtField);

JLabel label_21 = new JLabel("Outer:");
label_21.setHorizontalAlignment(SwingConstants.TRAILING);
label_21.setBounds(10, 78, 35, 14);
panel_6.add(label_21);

D34OTxtField = new JTextField();
D34OTxtField.setHorizontalAlignment(SwingConstants.TRAILING);
D34OTxtField.setFont(new Font("Tahoma", Font.PLAIN, 11));
D34OTxtField.setEditable(false);
D34OTxtField.setColumns(10);
D34OTxtField.setBounds(35, 92, 80, 22);
panel_6.add(D34OTxtField);



int delay = 250; //milliseconds
 ActionListener taskPerformer = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
            if(Channel1)
                                        ReceiveData1();
                                    else
                                    {
                                        Mag1TxtField.setText("N/A");
                                        Ang1TxtField.setText("N/A");
                                        OC1TxtField.setText("N/A");
                                        Trip1Color.setBackground(Color.BLACK);
                                    }
                                    if(Channel2)
                                        ReceiveData2();
                                    else
                                    {
                                        Mag2TxtField.setText("N/A");
                                        Ang2TxtField.setText("N/A");
                                        OC2TxtField.setText("N/A");
                                        Trip2Color.setBackground(Color.BLACK);
                                    }
                                    if(Channel3)
                                        ReceiveData3();
                                    else
                                    {
                                        Mag3TxtField.setText("N/A");
                                        Ang3TxtField.setText("N/A");
                                        OC3TxtField.setText("N/A");
                                        Trip3Color.setBackground(Color.BLACK);
                                    }
                                    if(Channel4)
                                        ReceiveData4();
                                    else
                                    {
                                        Mag4TxtField.setText("N/A");
                                        Ang4TxtField.setText("N/A");
```

```
                                        OC4TxtField.setText("N/A");
                                        Trip4Color.setBackground(Color.BLACK);
                                }
                                if(D12)
                                        ReceiveD12();
                                else
                                {
                                        D12ITxtField.setText("N/A");
                                        D12OTxtField.setText("N/A");
                                }
                                if(D13)
                                        ReceiveD13();
                                else
                                {
                                        D13ITxtField.setText("N/A");
                                        D13OTxtField.setText("N/A");
                                }
                                if(D14)
                                        ReceiveD14();
                                else
                                {
                                        D14ITxtField.setText("N/A");
                                        D14OTxtField.setText("N/A");
                                }
                                if(D23)
                                        ReceiveD23();
                                else
                                {
                                        D23ITxtField.setText("N/A");
                                        D23OTxtField.setText("N/A");
                                }
                                if(D24)
                                        ReceiveD24();
                                else
                                {
                                        D24ITxtField.setText("N/A");
                                        D24OTxtField.setText("N/A");
                                }
                                if(D34)
                                        ReceiveD34();
                                else
                                {
                                        D34ITxtField.setText("N/A");
                                        D34OTxtField.setText("N/A");
                                }
    }
 };

 timer = new Timer(250, taskPerformer);

JButton btnNewButton = new JButton("Run");
btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent arg0)
        {
                setRunning(true);
                getConfiguration();
                timer.start();

        }
});
btnNewButton.setBounds(198, 69, 124, 39);
contentPanel.add(btnNewButton);

JPanel buttonPane = new JPanel();
buttonPane.setBorder(new LineBorder(new Color(0, 0, 0)));
buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
```

```
                        getContentPane().add(buttonPane, BorderLayout.SOUTH);
                        {
                                JButton okButton = new JButton("Re-Configure");
                                okButton.addActionListener(new ActionListener() {
                                        public void actionPerformed(ActionEvent e)
                                        {
                                                timer.stop();
                                                OutputWindow.this.setVisible(false);
                                        }
                                });
                                okButton.setActionCommand("OK");
                                buttonPane.add(okButton);
                                getRootPane().setDefaultButton(okButton);
                        }

                        JButton btnStop = new JButton("Stop");
                        btnStop.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e)
                                {
                                        timer.stop();
                                }
                        });
                        btnStop.setBounds(343, 69, 124, 39);
                        contentPanel.add(btnStop);

        }

        public void getConfiguration()
        {
                try{
                File ConfigFile = new File(MainWindow.FilePath());
                FileReader fr = new FileReader(ConfigFile);
                BufferedReader ConfigFileReader = new BufferedReader(fr);

                while ((line = ConfigFileReader.readLine()) != null)
{
                        if (line.contentEquals("#define Channel1"))
                                Channel1=true;

                        if (line.contentEquals("#define Channel2"))
                                Channel2=true;

                        if (line.contentEquals("#define Channel3"))
                                Channel3=true;

                        if (line.contentEquals("#define Channel4"))
                                Channel4=true;

                        if (line.contentEquals("#define Distance12"))
                                D12=true;

                        if (line.contentEquals("#define Distance13"))
                                D13=true;

                        if (line.contentEquals("#define Distance14"))
                                D14=true;

                        if (line.contentEquals("#define Distance23"))
                                D23=true;

                        if (line.contentEquals("#define Distance24"))
                                D24=true;

                        if (line.contentEquals("#define Distance34"))
                                D34=true;
```

```
    }
                fr.close();
                ConfigFileReader.close();
                }
                catch (Exception Ex)
                {
                        Ex.printStackTrace();
                }
        }

        public void ReceiveData1()
        {
                //while(running)
                //{
                DatagramSocket socket1;


                try {
                        socket1 = new DatagramSocket(port1);
                                DatagramPacket packet1 = new DatagramPacket(new byte[13],13);
                                try {
                                        socket1.receive(packet1);
                                        byte[] data = packet1.getData();
                                        Mag1=((data[0]&0xFF)<<24) | ((data[1]&0xFF)<<16) | ((data[2]&0xFF)<<8) |
((data[3]&0xFF));
                                        Ang1=((data[4]&0xFF)<<24) | ((data[5]&0xFF)<<16) | ((data[6]&0xFF)<<8) |
((data[7]&0xFF));
                                        OCTrip1=data[12]&0xFF;
                                        Mag1s=Mag1+"";
                                        Ang1s=Ang1+"";
                                        Mag1TxtField.setText(Mag1s);
                                        Ang1TxtField.setText(Ang1s);
                                        if(OCTrip1==0)
                                        {
                                                //Trip1Color.setBackground(Color.GREEN);
                                                OC1TxtField.setText("Clear");
                                        }
                                        else if(OCTrip1==1)
                                        {
                                                //Trip1Color.setBackground(Color.RED);
                                                OC1TxtField.setText("Trip");
                                        }
                                        else
                                        {
                                                //Trip1Color.setBackground(Color.BLACK);
                                                OC1TxtField.setText("N/A");
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                socket1.close();
                } catch (SocketException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                }


        //}
        }

        public void ReceiveData2()
        {
                DatagramSocket socket2;

                try {
```

```
                socket2 = new DatagramSocket(port2);
                        DatagramPacket packet2 = new DatagramPacket(new byte[13],13);
                        try {
                                socket2.receive(packet2);
                                byte[] data = packet2.getData();
                                Mag2=((data[0]&0xFF)<<24) | ((data[1]&0xFF)<<16) | ((data[2]&0xFF)<<8) |
((data[3]&0xFF));
                                Ang2=((data[4]&0xFF)<<24) | ((data[5]&0xFF)<<16) | ((data[6]&0xFF)<<8) |
((data[7]&0xFF));
                                OCTrip2=data[12]&0xFF;
                                Mag2s=Mag2+"";
                                Ang2s=Ang2+"";
                                Mag2TxtField.setText(Mag2s);
                                Ang2TxtField.setText(Ang2s);
                                if(OCTrip2==0)
                                {
                                        //Trip2Color.setBackground(Color.GREEN);
                                        OC2TxtField.setText("Clear");
                                }
                                else if(OCTrip2==1)
                                {
                                        //Trip2Color.setBackground(Color.RED);
                                        OC2TxtField.setText("Trip");
                                }
                                else
                                {
                                        //Trip2Color.setBackground(Color.BLACK);
                                        OC2TxtField.setText("N/A");
                                }
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                socket2.close();
                } catch (SocketException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                }


        }

        public void ReceiveData3()
        {
                DatagramSocket socket3;

                try {
                        socket3 = new DatagramSocket(port3);
                                DatagramPacket packet3 = new DatagramPacket(new byte[13],13);
                                try {
                                        socket3.receive(packet3);
                                        byte[] data = packet3.getData();
                                        Mag3=((data[0]&0xFF)<<24) | ((data[1]&0xFF)<<16) | ((data[2]&0xFF)<<8) |
((data[3]&0xFF));
                                        Ang3=((data[4]&0xFF)<<24) | ((data[5]&0xFF)<<16) | ((data[6]&0xFF)<<8) |
((data[7]&0xFF));
                                        OCTrip3=data[12]&0xFF;
                                        Mag3s=Mag3+"";
                                        Ang3s=Ang3+"";
                                        Mag3TxtField.setText(Mag3s);
                                        Ang3TxtField.setText(Ang3s);
                                        if(OCTrip3==0)
                                        {
                                                //Trip3Color.setBackground(Color.GREEN);
                                                OC3TxtField.setText("Clear");
                                        }
```

```
                                        else if(OCTrip3==1)
                                        {
                                                //Trip3Color.setBackground(Color.RED);
                                                OC3TxtField.setText("Trip");
                                        }
                                        else
                                        {
                                                //Trip3Color.setBackground(Color.BLACK);
                                                OC3TxtField.setText("N/A");
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        socket3.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }

        }

        public void ReceiveData4()
        {
                DatagramSocket socket4;
                try {
                        socket4 = new DatagramSocket(port4);
                                DatagramPacket packet4 = new DatagramPacket(new byte[13],13);
                                try {
                                        socket4.receive(packet4);
                                        byte[] data = packet4.getData();
                                        Mag4=((data[0]&0xFF)<<24) | ((data[1]&0xFF)<<16) | ((data[2]&0xFF)<<8) |
((data[3]&0xFF));
                                        Ang4=((data[4]&0xFF)<<24) | ((data[5]&0xFF)<<16) | ((data[6]&0xFF)<<8) |
((data[7]&0xFF));
                                        OCTrip4=data[12]&0xFF;
                                        Mag4s=Mag4+"";
                                        Ang4s=Ang4+"";
                                        Mag4TxtField.setText(Mag4s);
                                        Ang4TxtField.setText(Ang4s);
                                        if(OCTrip4==0)
                                        {
                                                //Trip4Color.setBackground(Color.GREEN);
                                                OC4TxtField.setText("Clear");
                                        }
                                        else if(OCTrip4==1)
                                        {
                                                //Trip4Color.setBackground(Color.RED);
                                                OC4TxtField.setText("Trip");
                                        }
                                        else
                                        {
                                                //Trip4Color.setBackground(Color.BLACK);
                                                OC4TxtField.setText("N/A");
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        socket4.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
        }
```

```
	public void ReceiveD12()
	{
		DatagramSocket socket12;
		try {
			socket12 = new DatagramSocket(port12);
				DatagramPacket packet12 = new DatagramPacket(new byte[10],10);
				try {
					socket12.receive(packet12);
					byte[] data = packet12.getData();
					//Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
					//Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
					DITrip12=data[0]&0xFF;
					DOTrip12=data[1]&0xFF;;
					if(DITrip12==0)
					{
						D12ITxtField.setText("Clear");
					}
					else if(DITrip12==1)
					{
						D12ITxtField.setText("Trip");
					}
					if(DOTrip12==0)
					{
						D12OTxtField.setText("Clear");
					}
					else if(DOTrip12==1)
					{
						D12OTxtField.setText("Trip");
					}
				} catch (IOException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}
		socket12.close();
		} catch (SocketException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}
	}

	public void ReceiveD13()
	{
		DatagramSocket socket13;
		try {
			socket13 = new DatagramSocket(port13);
				DatagramPacket packet13 = new DatagramPacket(new byte[10],10);
				try {
					socket13.receive(packet13);
					byte[] data = packet13.getData();
					//Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
					//Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
					DITrip13=data[0]&0xFF;
					DOTrip13=data[1]&0xFF;;
					if(DITrip13==0)
					{
						D13ITxtField.setText("Clear");
					}
					else if(DITrip13==1)
					{
						D13ITxtField.setText("Trip");
					}
					if(DOTrip13==0)
```

```
                                        {
                                                D13OTxtField.setText("Clear");
                                        }
                                        else if(DOTrip13==1)
                                        {
                                                D13OTxtField.setText("Trip");
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        socket13.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }

                public void ReceiveD14()
                {
                        DatagramSocket socket14;
                        try {
                                socket14 = new DatagramSocket(port14);
                                        DatagramPacket packet14 = new DatagramPacket(new byte[10],10);
                                        try {
                                                socket14.receive(packet14);
                                                byte[] data = packet14.getData();
                                                //Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
                                                //Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
                                                DITrip14=data[0]&0xFF;
                                                DOTrip14=data[1]&0xFF;;
                                                if(DITrip14==0)
                                                {
                                                        D14ITxtField.setText("Clear");
                                                }
                                                else if(DITrip14==1)
                                                {
                                                        D14ITxtField.setText("Trip");
                                                }
                                                if(DOTrip14==0)
                                                {
                                                        D14OTxtField.setText("Clear");
                                                }
                                                else if(DOTrip14==1)
                                                {
                                                        D14OTxtField.setText("Trip");
                                                }
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                        socket14.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }

                public void ReceiveD23()
                {
                        DatagramSocket socket23;
                        try {
                                socket23 = new DatagramSocket(port23);
                                        DatagramPacket packet23 = new DatagramPacket(new byte[10],10);
```

```
                                        try {
                                                socket23.receive(packet23);
                                                byte[] data = packet23.getData();
                                                //Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
                                                //Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
                                                DITrip23=data[0]&0xFF;
                                                DOTrip23=data[1]&0xFF;;
                                                if(DITrip23==0)
                                                {
                                                        D23ITxtField.setText("Clear");
                                                }
                                                else if(DITrip23==1)
                                                {
                                                        D23ITxtField.setText("Trip");
                                                }
                                                if(DOTrip23==0)
                                                {
                                                        D23OTxtField.setText("Clear");
                                                }
                                                else if(DOTrip23==1)
                                                {
                                                        D23OTxtField.setText("Trip");
                                                }
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                        socket23.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }

                public void ReceiveD24()
                {
                        DatagramSocket socket24;
                        try {
                                socket24 = new DatagramSocket(port24);
                                        DatagramPacket packet24 = new DatagramPacket(new byte[10],10);
                                        try {
                                                socket24.receive(packet24);
                                                byte[] data = packet24.getData();
                                                //Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
                                                //Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
                                                DITrip24=data[0]&0xFF;
                                                DOTrip24=data[1]&0xFF;;
                                                if(DITrip24==0)
                                                {
                                                        D24ITxtField.setText("Clear");
                                                }
                                                else if(DITrip24==1)
                                                {
                                                        D24ITxtField.setText("Trip");
                                                }
                                                if(DOTrip24==0)
                                                {
                                                        D24OTxtField.setText("Clear");
                                                }
                                                else if(DOTrip24==1)
                                                {
                                                        D24OTxtField.setText("Trip");
```

```
                                        }
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                }
                        socket24.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }

                public void ReceiveD34()
                {
                        DatagramSocket socket34;
                        try {
                                socket34 = new DatagramSocket(port34);
                                        DatagramPacket packet34 = new DatagramPacket(new byte[10],10);
                                        try {
                                                socket34.receive(packet34);
                                                byte[] data = packet34.getData();
                                                //Mag12=((data[2]&0xFF)<<24) | ((data[3]&0xFF)<<16) | ((data[4]&0xFF)<<8) |
((data[5]&0xFF));
                                                //Ang12=((data[6]&0xFF)<<24) | ((data[7]&0xFF)<<16) | ((data[8]&0xFF)<<8) |
((data[9]&0xFF));
                                                DITrip34=data[0]&0xFF;
                                                DOTrip34=data[1]&0xFF;;
                                                if(DITrip34==0)
                                                {
                                                        D34ITxtField.setText("Clear");
                                                }
                                                else if(DITrip34==1)
                                                {
                                                        D34ITxtField.setText("Trip");
                                                }
                                                if(DOTrip34==0)
                                                {
                                                        D34OTxtField.setText("Clear");
                                                }
                                                else if(DOTrip34==1)
                                                {
                                                        D34OTxtField.setText("Trip");
                                                }
                                        } catch (IOException e) {
                                                // TODO Auto-generated catch block
                                                e.printStackTrace();
                                        }
                        socket34.close();
                        } catch (SocketException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }
                }

                public void setRunning(boolean bool)
                {
                        running=bool;
                }

                public boolean getRunning()
                {
                        return running;
                }
}
```