Design of Data Acquisition System and Fault Current Limiter for an Ultra Fast Protection System

by

Arvind Thirumalai

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved March 2011 by the
Graduate Supervisory Committee:

George Karady, Chair
Vijay Vittal
Kory Hedman

ARIZONA STATE UNIVERSITY
May 2011

ABSTRACT

This research work describes the design of a fault current limiter (FCL) using digital logic and a microcontroller based data acquisition system for an ultra fast pilot protection system. These systems have been designed according to the requirements of the Future Renewable Electric Energy Delivery and Management (FREEDM) system (or loop), a 1 MW green energy hub. The FREEDM loop merges advanced power electronics technology with information technology to form an efficient power grid that can be integrated with the existing power system.

With the addition of loads to the FREEDM system, the level of fault current rises because of increased energy flow to supply the loads, and this requires the design of a limiter which can limit this current to a level which the existing switchgear can interrupt. The FCL limits the fault current to around three times the rated current. Fast switching Insulated-gate bipolar transistor (IGBT) with its gate control logic implements a switching strategy which enables this operation. A complete simulation of the system was built on Simulink and it was verified that the FCL limits the fault current to 1000 A compared to more than 3000 A fault current in the non-existence of a FCL. This setting is made user-defined.

In FREEDM system, there is a need to interrupt a fault faster or make intelligent decisions relating to fault events, to ensure maximum availability of power to the loads connected to the system. This necessitates fast acquisition of data which is performed by the designed data acquisition system. The microcontroller acquires the data from a current transformer (CT). Measurements are made at different points in the FREEDM system and merged together, to input it to the intelligent protection algorithm that has been developed by another student on the project. The algorithm will generate a tripping signal in the event of a fault. The developed hardware and the programmed software to accomplish data acquisition and transmission are presented here.

The designed FCL ensures that the existing switchgear equipments need not be replaced thus aiding future power system expansion. The developed data acquisition system enables fast fault sensing in protection schemes improving its reliability.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor and chair Dr. George Karady for his advice and continued support throughout my work. I am deeply indebted to the FREEDM system research centre for providing me this opportunity to work on this very interesting project. I greatly acknowledge the technical support offered by Microchip support and Stuart Lee Borden from the Microchip forum in helping me through various stages of hardware implementation. I would like to extend a special thanks to Xing Liu for the constant technical discussions about the project. I would also like to thank Dr. Vijay Vittal and Dr. Kory Hedman for their valuable suggestions and for being in the supervisory committee. I am obliged to the faculty of the power systems group at Arizona State University for their guidance throughout my tenure here.

A special thanks to the National Science foundation for funding my research. I would like to thank my parents and sister for their extended support during difficult times without which this would not have been possible. I would also like to extend a special thanks to Shweta Natarajan and all my friends from Arizona State University for always supporting me.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

NOMENCLATURE

A	Ampere
A/D	Analog to Digital Converter
AC	Alternating Current
ARP	Address Resolution Protocol
CT	Current Transformer
$\frac{dv}{dt}$	Voltage gradient with respect to time
D/A	Digital to Analog Converter
DAQ	Data Acquisition System
DC	Direct Current
DESD	Distributed Energy Storage Device
DGI	Distributed Grid Intelligence
DHCP	Dynamic Host Control Protocol
ESD	Energy Storage Device
FCL	Fault Current Limiter
FID	Fault Isolation Device
FREEDM	Future Renewable Electric Energy Delivery and Management
GPS	Global Positioning Satellite
GTO	Gate turn-off Thyristor
HEX2DEC	Hexadecimal to Decimal Converter
HTS	High Temperature Superconductor
ICMP	Internet Control Message Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IEM	Intelligent Energy Management

IFM	Intelligent Fault Management
IGBT	Insulated-gate Bipolar Transistor
IP	Internet Protocol
IRIG	Inter-range Instrumentation Group
J_c	Critical Current density of Superconductor
k	Kilo ($1x10^3$)
kA	Kilo Ampere
kbyte	Kilo Byte
kOhm	Kilo Ohm
kV	Kilo Volt
kVA	Kilo Volt Ampere
K	Kelvin
KEPRI	Korea Electric Power Research Institute
m	Milli ($1x10^{-3}$)
mH	Milli Henry
ms	Milli Second
mV	Milli Volt
M	Mega ($1x10^6$)
Mbps	Mega bits per second
MHz	Mega Hertz
MS	Microsoft
MVA	Mega Volt Ampere
MW	Mega Watt
n	Nano ($1x10^{-9}$)
ns	Nano Second

NSF	National Science Foundation
OP AMP	Operational Amplifier
p.u.	Per unit
rms	Root mean square
RC	Resistor-Capacitor
RSC	Reliable and Secured Communication
SFCL	Superconducting Fault Current Limiter
SSFCL	Solid State Fault Current Limiter
SNTP	Simple Network Time Protocol
TCP	Transmission Control Protocol
s	Seconds
SEL	Schweitzer Engineering Laboratories
SST	Solid State Transformer
u	Micro ($1x10^{-6}$)
uF	Micro Farad
uH	Micro Henry
US	United States of America
V	Volt
W	Watt

1 INTRODUCTION

1.1 Overview

With an ever-increasing load demand major changes to the electrical power generation and energy distribution patterns are being developed. One strategy from the US Department of Energy is to encourage and support widely distributed renewable electric generation, distributed control of energy management, active customer involvement, energy storage [1]. The Energy Information Administration predicts the total energy consumption in the US will increase at an average annual rate of 1.0 percent from 2008 to 2035. In order to meet this requirement, renewable energy generation must make up a larger share of the generation mix. While the renewable generation currently contributes to less than 7 % of the total electricity use in the US, the number will need to increase several folds in the next two decades. The FREEDM loop, a National Science Foundation (NSF) initiative, is an efficient power grid which provides solution to these problems. It eliminates the need for upgrading the existing power system equipments at distribution level and also addresses the renewable energy resourcing requirement for power generation. The addition of many new loads will place an additional burden on the electrical infrastructure. The FREEDM system's strategy to employ distributed generation and energy management through advanced power electronics and information technology is demonstrated in this work.

1.2 FREEDM system

The FREEDM system was developed as a smart grid infrastructure that will aid the large scale integration of renewable energy generation into the power grid. The system provides breakthrough technologies in energy storage, distributed control and power semiconductor devices. To demonstrate the various developments, a 1 MW green energy hub supplied by renewable energy resources is under development. The FREEDM system is aimed at enabling the customers to plug-and-generate, plug-and-store energy devices at homes and factories, as well as has the ability to manage their energy consumption through the load management system [1]. A single line diagram of the proposed FREEDM system is shown in Figure 1.1.

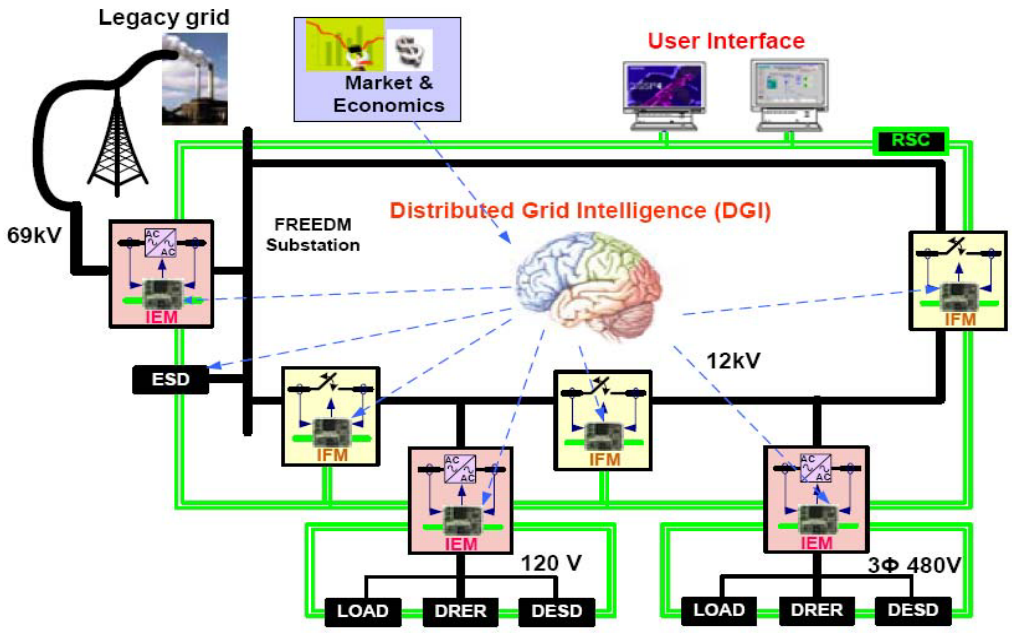

Figure 1.1 Proposed FREEDM system
(Taken directly from [1])

In this system, low voltage smart loads are connected to a regulated 120VAC. Low voltage Distributed Renewable Energy Resource (DRER), Distributed Energy Storage Device (DESD) and loads are connected to the 12kV bus through a highly efficient, power electronics based Intelligent Energy Management (IEM) system. Alternatively they can be connected to the DC bus, made available through the Solid State Transformer (SST). A SST is a three port system (one AC in, one DC out, one AC out) which performs bi-directional energy flow control and many other power management control functions. For industry loads at 480 V three phase, a higher rating IEM is designed. This IEM will generate regulated 480 V AC output for the loop and also generate high voltage DC, at voltage level higher than that at 120 V AC output-ends IEM, to connect to various DRER, loads and DESD at that higher DC voltage.

The Intelligent Fault Management (IFM) is utilized to improve the reliability of the FREEDM system by isolating potential faults in the 12kV bus. The IFMs and IEMs communicate with each other through a Reliable and Secured Communication (RSC) network. A Fault Isolation Device (FID) will perform the function of isolating the fault, being a part of the IFM. The system

one line diagram of the FREEDM loop indicating the location of FIDs and SSTs is shown in Figure 1.2.

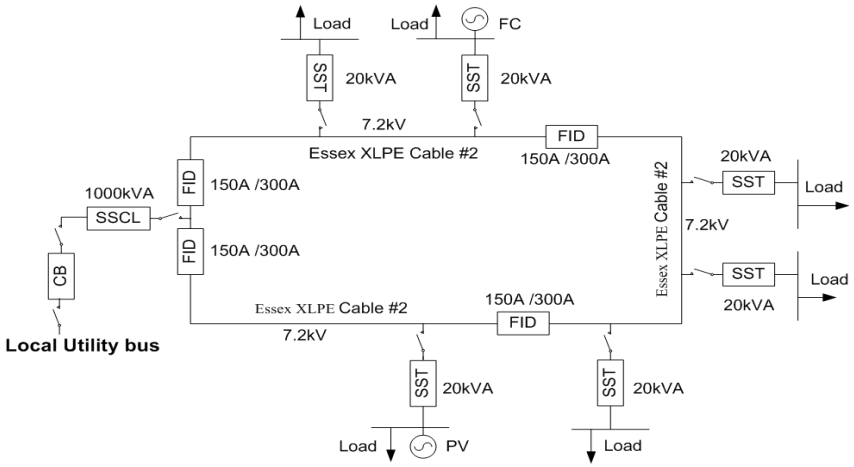

Figure 1.2 Single line diagram of FREEDM system
(Taken directly from [2])

The Fault Current Limiter (FCL) (not shown in Figure 1.1 and 1.2) is located at various points in the loop such that the fault current observed by the IFM will be a reduced current than the actual current. It can be seen from Figure 1.1 and 1.2 that the FREEDM system will still be connected to the traditional power grid through a higher rated IEM. This improves the system reliability in the event of failure of any DRER and it also reinstates the fact that, the FREEDM loop must be connected to the existing power grid. In the event of the fault in the power grid, the FREEDM system can operate independently as a smart grid. Coordinated control of the FREEDM system is achieved through the Distributed Grid Intelligence (DGI) software programmed into each IEM and IFM. The DGI could be considered the brain of the whole FREEDM loop since it performs total energy management and power balancing functions inside the loop.

The work performed as part of this thesis relates only to the FCL and the IFM. More on these topics will be reviewed in later chapters.

1.3 Objectives of the research

The basic objective of designing the FREEDM loop is to develop new technology which can handle the delivery and management of electrical energy generated using renewable energy

sources. The effective way of implementing this technology in the existing power grids should be with modifications to the grid which do not alter the system reliability. With growing load demand, the problem of efficient load management using the traditional technology could be difficult. The renewable energy penetration will enable new advances in the power electronics technology, thus necessitating new devices capable of handling these advancements. Faster switching strategies in power electronics technology cannot occur at higher magnitude of current, especially during a fault. This research describes a methodology to mitigate the problem of huge fault current using FCL. This device is designed to limit the fault current to a pre-set value, which is user-definable. A distribution system at 12.47kV has fault current of the magnitude 2500-3000 A in the case considered. The FCL limits this current to 500-1000 A based on its design configuration. A software simulation of the FCL based on solid state technology was performed using Matlab/Simulink and the results suggest that this device is capable of limiting the fault current while exercising lesser burden on the power grid during this process.

Another objective of the research is the design of a data acquisition system (DAQ) for the IFM system. The IFM is the protection system of the FREEDM loop which facilitates ultra fast protection. The function of the IFM is to sense faults and send a trip signal to the FID, which has to quickly restore system voltage following a fault, for continuous operation [3]. The time taken by the FID to interrupt a fault is estimated to be of the order of microseconds. Commercially available protection relays will take almost 2 cycles to sense a fault and send a trip signal and the result is presented in chapter 4. Hence there is a need to develop a custom-made data acquisition and fault management system to ensure fast performance of the IFM. The fault management algorithm has been developed by another student on the project using real-time algorithm to make fault decision faster. To implement the faster protection system, an ultra fast data acquisition device is the need. Current data needs to be acquired from various locations in the system, processed and merged into a single unit at the IFM terminal. The IFM will analyze this data for fault and send out signals accordingly. This research work presents the designed architecture for the data acquisition device, the hardware implementation and its related software program. The work presented

demonstrates the software simulation of the DAQ, the factors influencing the choice of components and a hardware prototype of the proposed DAQ. In the experimental setup for the prototype, the current data are obtained from various measurement locations, merged together at the IFM and fed into the protection algorithm. The protection scheme developed improves the reliability and availability of the FREEDM loop, by reducing the fault liability time, to supply the loads.

1.4 Organization of the thesis

Chapter 2 describes the literature review addressing the existing technologies in the FCL and explains the differential protection scheme for ultra fast protection.

Chapter 3 presents the simulation of the FCL performed using Matlab/Simulink and its results. A detailed description of the controlling strategy and a brief explanation of each component inside the controller complete this chapter.

Chapter 4 explains the FREEDM loop protection system in detail. This includes the protection algorithm employed and the interconnection between the data acquisition device and the protection algorithm. Moreover, various factors influencing the choice of hardware components like the signal processing speed, communication protocol, and latency are discussed.

Chapter 5 demonstrates the various steps in the choice of hardware and the integration of data acquisition device components. A simulation of the data acquisition device is presented. A voltage biasing circuit designed for this device on Matlab/Simulink using Operational Amplifier (OP AMP) is displayed. The complete hardware prototype developed, with three measurement locations, is finally presented.

Chapter 6 shows the merging of the data from three data acquisition device and the resultant tripping signal generated in the event of the fault is examined. Inferences are drawn on the observed output and the implementation challenges are discussed.

Chapter 7 presents the conclusion and the future work.

Appendix A presents digital values from simulation, B, C provide the microcontroller code developed using C language on the MPLAB 8.53 Integrated Development Environment (IDE), D provides the settings made to the GPS using the AcSELerator Quickset software.

2 LITERATURE REVIEW

2.1 Fault Current Limiter

A fault current limiter (FCL) is a device whose primary function is to limit the excessive current when a fault occurs. Added functions such as current interruption can also be incorporated into a fault current limiter to make its implementation more cost effective. Various kinds of limiters are being developed and the most common types are the superconducting fault current limiter and solid state fault current limiter. Other versions such as inductive type and variable resistor type are also under development. The proof of concept is shown in Figure 2.1. It shows the operation of a resistor type fault current limiter. The system is operating at 140 V rms and 0.22 ohm shunt resistor [4]. The FCL inserts a resistance when the current magnitude reaches 250 A and an intended 910 A peak short circuit current is limited to 400 A peak in the first cycle and 340 A peak in the fourth cycle (not shown in Figure 2.1). This describes the basic theory of operation of a FCL.

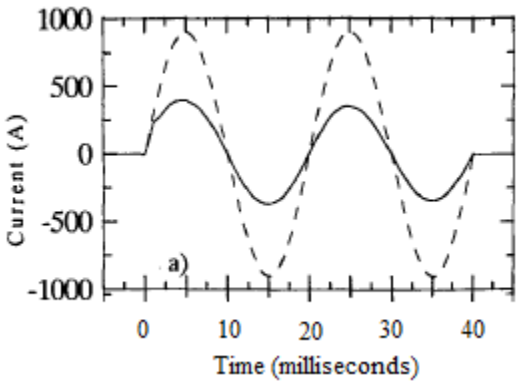

Figure 2.1 Concept of FCL
(Taken directly from [4])

2.1.1 Necessity of fault current limiter

Increasing demand for power results in greater power transfer along the transmission and distribution lines. This leads to line overloading and a heavier fault current during a fault, which the existing lines are not designed to handle, requiring establishment of newer lines. To interrupt the high fault current, circuit breakers of higher capacity would be needed [5]. An upgrade to the

equipments in the line would demand huge investment and it does not seem practical, considering the ever-growing demand for power. Other solutions such as construction of new substations, bus splitting, current limiting reactor, high impedance transformers, and sequential breaker tripping are found to be either a compromise on the cost or the reliability of the system. The necessity for a device which is economical, offers current limiting capabilities and which does not affect the reliability of the system by restoring the system voltage faster is the reason for the development of fault current limiters [3].

2.1.2 Types of fault current limiter

This section describes the different kinds of fault current limiters that are currently studied for research purpose and future installations.

2.1.2.1 Superconducting fault current limiter (SFCL)

A common type of a SFCL is the resistive type in which the superconductor is directly connected in series with the line to be protected [6]. Low temperature superconductors which lose their superconductivity a few degree K above absolute zero require liquid helium cooling. This coolant is really expensive. The development of high temperature superconductors (HTS), which require the less expensive liquid nitrogen cooling, is a major advantage in the practical application of a SFCL. The cross-section of a superconductor is determined such that at nominal current it offers ideally zero resistance. This is possible only for a DC circuit but in the case of an AC circuit, the alternating magnetic field produced by the current produces AC losses. In the event of a fault, the increase in current density J_c drives the superconductor from a superconducting state to a flux-flow state and the rapidly increasing resistance limits the current to a value below the prospective fault current. Once the limiting operation is complete, the superconductor needs to be cooled using liquid helium or liquid nitrogen to put it back in the circuit in superconducting state.

A transformer type SFCL consists of a series transformer and a superconducting current limiting device and is shown in Figure 2.2 [7]. The suffixes 1 and 2 represent the primary and secondary side values respectively. R_2 is the resistance of the superconducting current limiting. When the line current I_1 is small, the fault current limiter shows very low impedance, which is the

normal mode. In the event of a fault, the SFCL shows high impedance due to the resistive nature of the superconducting limiting device.

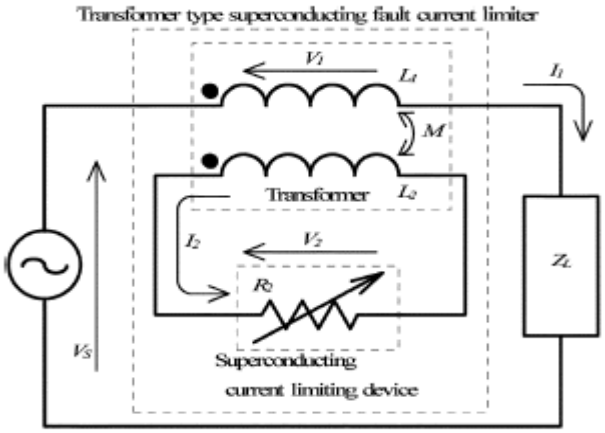

Figure 2.2 Circuit of a transformer type SFCL
(Taken directly from [7])

The ratings of the fault current limiter tested for a line fault is shown in Table 2-1.

Table 2-1 Ratings of fault current limiter for a line fault test [7]

Series Transformer	
Primary	100 V,48 A (4.8kVA)
Secondary	5.181 V, 886.9 A (4.595 KVA)
Frequency	50Hz
L1	62.90 mH
L2	0.1786 mH
M	3.318 mH
Superconducting current limiting device	
Structure	Non-inductive air-core superconducting winding (wire length 5.5 m)
Material	NbTi/CuNi=0.2/0.8
DC Critical current	1140 A (95X12 strands)
Residual inductance	2.0×10^{-6} H
Normal resistance	1.78 Ohm (at 10 K)

The fault current in this circuit is estimated to be around 1.7-2 p.u [7]. The output waveforms obtained in the event of successful operation of the transformer type SFCL are shown in Figure 2.3. In the figure, a fault is initiated at time 0 s. Figure 2.3(a) shows the supply voltage, 2.3(b) shows line current, 2.3(c) shows primary voltage of transformer and 2.3(d) shows the secondary voltage of the transformer. Figure 2.3 (b) shows that the fault current has been limited to less than primary current. Since the SFCL enters current limiting mode after fault occurrence, hence the voltage at the transformer terminals increases as observed in Figure 2.3 (c) and 2.3 (d).

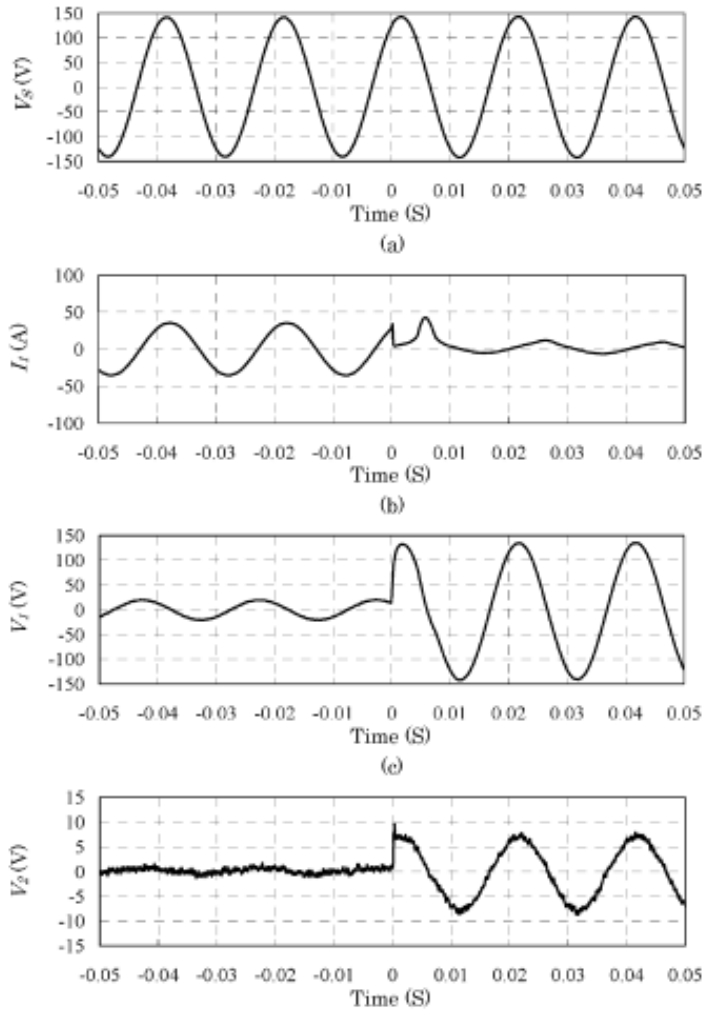

Figure 2.3 Output waveforms of a line fault test in transformer type SFCL
(Taken directly from [7])

Quench time is defined as the period between the initiation of the fault and the time when the device begins to limit the current [8]. As observed from waveforms in Figure 2.3, the quench time of this device is of the order of ms. There are a few issues in the practical application of these FCL [9]. This is shown in Table 2-2.

Table 2-2 Practical application issues of SFCL [9]

Issues	Fault current limiters
Coordination	• Protective coordination with relays • Reclosing scheme satisfaction • Fast recovery under loads
Life and maintenance	• Life expectancy as good as conventional ones • Minimized maintenance costs and process
Performance	• Consideration of fault current limiting time • Time delay according to installation site
Cost	• Total amount of superconductors • Cryostat capacity
Size	• Compact size comparable to switchgear

2.1.2.2 Hybrid fault current limiters

Due to the concerns in the practical application of SFCL, hybrid fault current limiters were introduced. They are combined devices including superconductors, semiconductors, conventional fuse or circuit breaker and fast switches [9]. LS industrial systems and KEPRI developed new hybrid fault current limiter using superconducting module, fast switch and current limiting sensors as shown in Figure 2.4.

These limiters have a rating of 24kV/630 A 3 phase. The basic requirement and solutions provided by this hybrid FCL is given in Table.2-3.

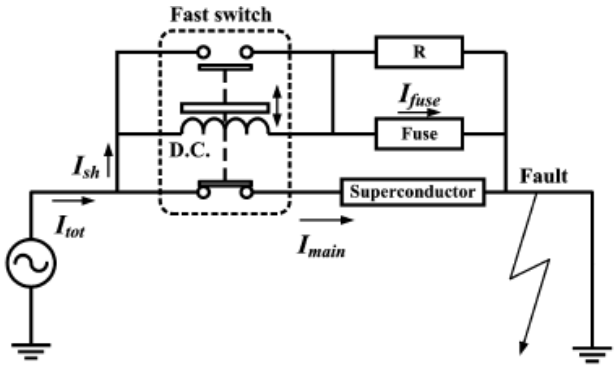

Figure 2.4 Hybrid SFCL
(Taken directly from [9])

Table 2-3 Requirements and solutions of hybrid FCL [9]

Requirements	Solutions
Minimization of superconductor usage and cryogenic burdens	Superconductors were used for only fault current sensing and commutation of fault currents
High voltage applications	A Fast switch could endure high voltage but not the superconductors
Fast switch operation and fault current commutation	Electromagnetic repulsion force was used, and its current source is fault current itself commutated by superconductors
Coordination and reclosing scheme	Coordination and reclosing scheme was considered by commutation method
Cost	1/10 reduction compared to superconducting fault current limiters

These types of fault current limiters have low cost, high performance and enable easier coordination with conventional systems compared to SFCL. Hence they are a promising technology for practical and commercial solutions in the near future.

2.1.2.3 Passive fault current limiter in parallel biasing mode

Passive fault current limiter is a combination of passive elements such as permanent magnet and a saturable core [10]. The structure of FCL in this mode is shown in Figure 2.5.

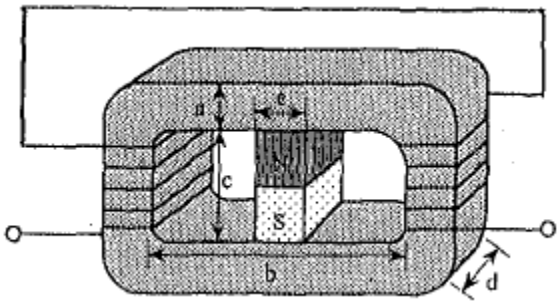

Figure 2.5 FCL parallel biasing mode
(Taken directly from [10])

The magnet is in the middle of the limbs and it biases the core. Each limb is used for one half cycle. The use of two coils in two limbs in magnetically counter direction makes it to function as a bipolar current limiter and it can be used with larger fault current. In the reference [10], a simulation was performed for this fault current limiter. For simulation, a variable resistance was connected as a load across the FCL, along with a source voltage. A fault was simulated by varying the load. It was seen that fluctuation of the flux was not so large which confirms that demagnetization of the permanent magnet was not severe. Based on the waveforms obtained it was concluded that, the performance was more successful in parallel biasing mode than in series biasing mode and also the longevity of the model is increased. The cost of this system is very high at higher voltages thus restricted the applications at higher voltages.

2.1.2.4 Solid state fault current limiter

The shortcomings of previously described fault current limiters are the cost and the limited fault current varies with the supply system condition and fault location. Moreover, the operation time is limited due to power dissipation in the impedance, which is usually resistive. A fault current limiter developed using solid state devices provides an advantage of producing repeated switching patterns such that on occurrence of a fault, the fault current is always limited to a predetermined level.

A solid state fault current limiter (SSFCL) can limit a fault current passing through it within one half cycle [11]. Many configurations of a SSFCL have been developed. One such configuration is a FCL made of two parallel connected branches with two GTO thyristors in one branch and a solid state switch using a thyristor switch in series with a reactor in another branch [12]. The GTO switch serves the purpose of clearing the faults. This switch is kept closed under normal condition until the load current reaches a preset value. It opens and interrupts the current flow. The thyristor switch on the other branch is normally open and it is chosen such that it maintains 15kA fault current for 15 cycles which is the required criteria in a distribution network. This works out as an economical solution for fault current limiting just using GTO for current limiting purpose. A model of the described FCL is shown in Figure.2.6.

The function of the ZNO arrester is to provide clamping voltage level and to protect the system from the surge current. The rms current level in the circuit is constantly compared with the predetermined reference value. The output of the comparator is used to generate signals for the GTO and the thyristor. If the magnitude of the measured current-detected is small for a period of time governed by the circuit parameters, a fault clearance is assumed and automatic resetting is provided.

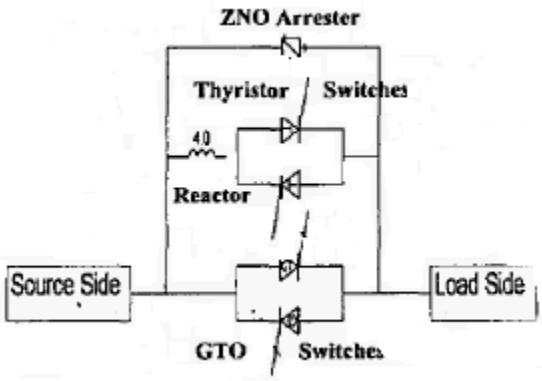

Figure 2.6 Configuration of FCL using GTO
(Taken directly from [12])

For a typical system of 12.5 kV system voltage, load of 1.2 MVA and 0.90 power factor a fault current of 8.5 kA is reduced to 3.7 kA. If the load is an induction motor, a reduced fault cur-

rent of 5.0 kA is observed. The SSFCL current limiting reactor protects busbars against voltage sags occurring due to faults. Hence a reactor is used in series with thyristor.

2.1.3 Conclusion

Based on the advantages and the advanced features provided by solid state fault current limiter, this research is dedicated to designing a solid state fault current limiter with a different controlling strategy which caters to the requirement of the FREEDM system.

2.2 Pilot Protection scheme for ultra fast protection

2.2.1 Overview

The basic working principle of working of a pilot protection scheme is similar to a differential protection scheme, except that in the case of pilot protection scheme the measurements are relayed to a central station through pilot wires for fault judgments [13]. The pilot wires could be either the line itself, or a communication cable running parallel to the line. Differential protection schemes are the fastest schemes for protecting lines, loads, equipments, and buses which connect multiple lines. They are also used to protect generators and transformers. When employed for the protection of lines or buses the common name used is pilot protection. Various advantages which make differential protection more preferred are:

- It provides primary zone protection without backup
- Coordination with protection in adjacent zones is eliminated, which permits high speed tripping
- Precise relay settings are not necessary
- Calculation of fault currents and voltages are not unnecessary.

These advantages ensure this protection scheme offers fast tripping mechanisms which exactly address the requirements of the FREEDM system.

Current different protection is based on Kirchhoff's current law which is, sum of currents into a node (or a section) should match the sum of currents flowing out of it. With great progress in digital communication technique, it is playing a major part in power system protection. The function of communicating the input currents to a central station is performed by the digital com-

munication modules. This technique is immune to power swings, mutual coupling, and series impedance unbalances [14]. Limitations such as CT saturation, power system impedance nonhomogeneity, sampling asynchronization, and communication channel delay affect the performance of a current differential protection. The last two limitations produce a phase shift between local current and received current but the magnitude remains unchanged. Hence a relay should be programmed to withstand a phase error of up to 40° to ensure its reliability [14]. CT saturation affects the magnitude of currents and the relay should be programmed accordingly to accommodate the magnitude error in the measurement. A differential relay employing a digital communication technique should account for both the phase error and the magnitude error.

Protection devices, in the last century, have migrated from electro-mechanical to semiconductor, to integrated circuit and to microprocessor technologies. Microprocessor based digital and numerical relays are gradually replacing conventional relays in power system protection [15]. Existing pilot wire relays have disadvantages such as, they require high quality pilot wires for sending protection signals because of the accuracy-requirement of the signal and the noise while others do not justify the investment incurred into the numerical technology. Hence numerical relays based on digital communication technique are the norm for the future.

2.2.2 Conventional differential protection

The basic working of a conventional differential scheme has been explained next and the same concept applies to a modern numerical current differential relay [15]. Under normal conditions the currents at both ends of the protected zone are equal in magnitude and opposite in phase. It causes the secondary current to circulate around the pilot wire circuit without producing any differential current in the relay R as shown in Figure 2.7.

The currents shown in Figure 2.8 represent the through-fault/external fault condition. There is no current flowing in the relay with the increased primary and secondary currents at the protected zone boundaries.

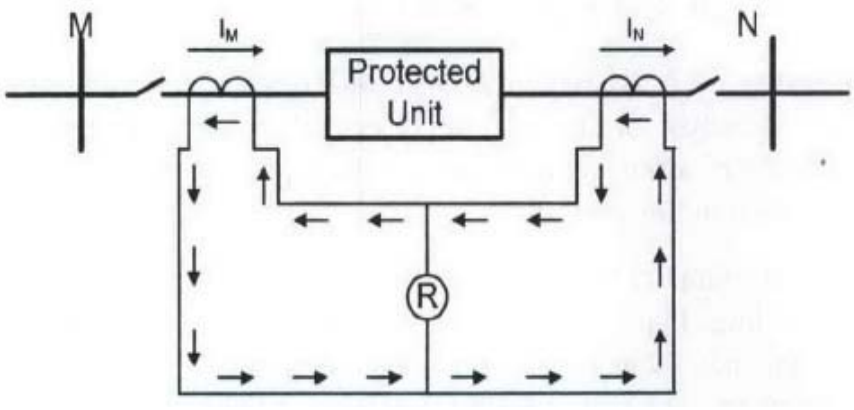

Figure 2.7 Normal condition of a differential protection scheme
(Taken directly from [15])

A fault within the protected zone will cause the CT secondary current to imbalance, and the difference between these currents will flow in the relay as shown in Figure 2.9.

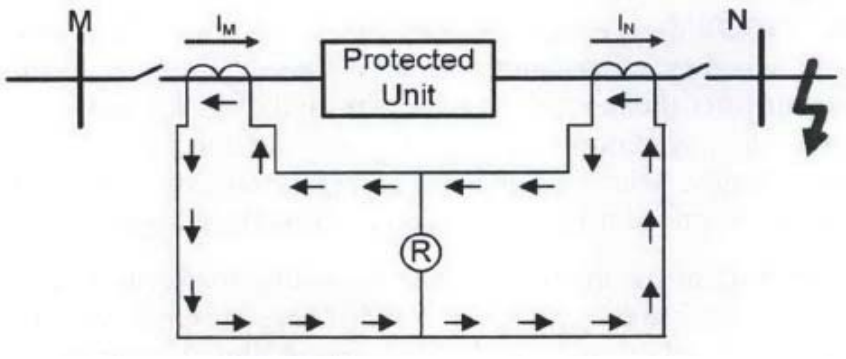

Figure 2.8 External fault condition of a differential protection scheme
(Taken directly from [15])

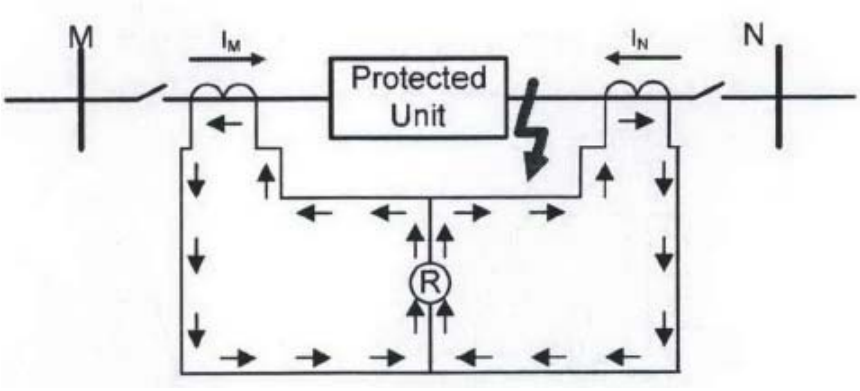

Figure 2.9 Internal fault condition of a differential protection scheme
(Taken directly from [15])

The conventional differential relay has a few drawbacks such as the relay setting is via plug bridges and potentiometers, which means that remote setting and flexible settings are not possible. Secondly, the relay cannot record disturbances, make power system measurements, or communicate with other devices in the substation. To achieve these, numerical relays are applied in modern power systems [15].

2.2.3 Modern numerical differential protection

A radical change from conventional to numerical relays, means the signal transmitted on pilot wires should be digital. This requires high quality pilot wire and concerns the investment incurred in this transition. The ideal way to do this would be, to let the elements connected to the CT circuit and pilot wires as per conventional relay and make the logical decision unit and human machine interface to be improved by numerical technology [15]. In this case, the relay measures and digitizes the secondary current of the CTs and the decision about the fault in the protected zone is made by the software programmed into the numerical relay [26]. A block diagram model of a numerical relay is shown in Figure 2.10.

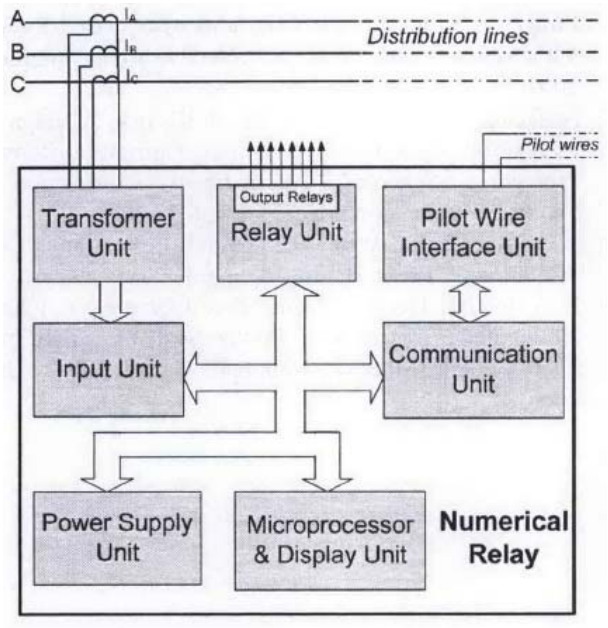

Figure 2.10 Block diagram of a numerical relay
(Taken directly from [15])

The transformer unit acquires data from each phase of the line. The input unit processes the data into the correct data format before transmitting it to the relay unit, which makes fault decisions about the protected zone. The pilot wire interface unit acquires data from other measurements along the distribution line and it also sends out to other connected numerical relays. A block diagram showing the inner modules of this unit is shown in Figure 2.11. The communication unit handles the communication between various inner blocks of a numerical relay. The display unit performs the function of a human machine interface.

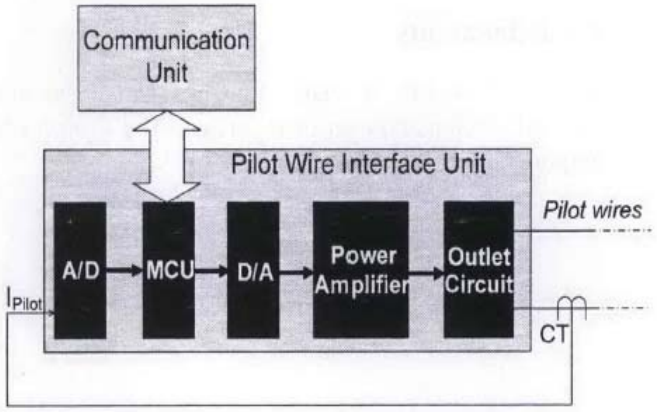

Figure 2.11 Block diagram of a Pilot Wire Interface Unit
(Taken directly from [15])

A microcontroller in the pilot wire interface unit receives the summated current data from the communication unit and sends it to the D/A (digital-analog converter). Then, analog current is output from the power amplifier and flows into the pilot wire via an outlet circuit [15]. The outlet circuit is designed to apply a circulating current system and limit the maximum voltage on the pilot wire, and it can be arranged referring to conventional relay designs. At the same time, a CT delivers the pilot wire current to A/D (analog to digital converter), and the micro-controller sends relevant data back to the communication unit.

2.2.4 TCP communication- Overview

In a typical digital relay, the data acquisition is performed using microcontroller, which also performs analog to digital conversion of data. This data is transferred to the central station

through existing communication system like serial or Ethernet [16]. A standard communication protocol using the Ethernet connection is Transmission Control Protocol/ Internet Protocol (TCP/IP). The different layers in the TCP/IP protocol are:

- Physical Layer - This layer incorporates the physical properties of a transport media, which includes the mechanical and electrical characteristics. Also included are parameters such as voltage levels, connectors and protectors.
- Network Access Layer - This layer deals with logical addresses and provides routing through the network nodes. This layer uses the IP and ICMP (Internet Control Message Protocol)
- Internet Layer – This layer performs the function of sending packets across one or more networks.
- Transport Layer - This layer is responsible for the reliability of the link and maintains the TCP under its control. One of the major applications of this layer occurs when the first layers are in control of the carrier provider.
- Application Layer - This layer is the highest or final layer and is where the applications are executed.

The important function of a TCP/IP protocol can be explained as follows. The application layer initiates a request for a connection to a remote computer through the transport layer. The transport layer delivers a message over the network access layer, informing the network to route and gain access to the remote computer. Presently existing fast Ethernet communication speeds are 10 Mega bits per second (Mbps), 100 Mbps and 1 Giga bits per second (Gbps) and speeds such as 10 Gbps and 100 Gbps are being developed as the next generation Ethernet [17].

2.2.5 Microcontroller-based modern numerical differential protection

The data acquisition device using a microcontroller performs the function of acquiring signals, processing them and transmitting them to the monitoring station [18]. The measured data is first conditioned using precision electronic circuits and then interfaced to the computer at the monitoring station through a microcontroller card [19]. Commercially available data acquisition

cards have higher cost, decreases the flexibility in the changes that can be programmed onto it and also narrows its usability to a fewer applications. Hence there is a need to develop a custom-made data acquisition system to satisfy the requirements of lesser cost and higher flexibility.

In general, protection systems are affected by the presence of harmonics pollution on voltage and current waveforms resulting in mis-operation and therefore diverse problems such as economic losses, discontinuity of service or even the presence of a permanent fault due to the failure of the protection system [20]. This is true in the case of old electromechanical protection systems and solid state based protection systems but not in the case of modern digital protection systems because settings can be programmed to disable trip signal generation depending on the magnitude of the harmonic pollution. The increased use of non linear loads has resulted in high harmonic distortion in voltage and current waveforms thus reducing the reliability of electromechanical and solid state based protection systems. Data acquisition and signal filtering are essential parts of a digital protection system. The filtering process has to accomplish:

- Bandpass response close to the system frequency
- DC and ramp rejection
- Harmonic attenuation or rejection
- Fast response.
- Good transient behavior.

Microcontroller based digital protection systems ensure high accuracy of filtering, reliable data conversion and processing. This explains the reason to transform from electromechanical and solid state based protection systems to modern microcontroller based protection systems.

2.2.6 Conclusion

From the literature review, it was concluded that the FREEDM system requires a fast protection scheme, which should be faster than the conventional protection systems since this ensures ultra fast tripping using the FID. To design a protection scheme which is fast, data needs to be acquired faster for processing. This is done using a microcontroller which acquires signals from current transformers connected in series with the FREEDM loop. The data acquired needs to be

transported to a monitoring station in a fast and secure manner and the TCP communication protocol performs this function expertly. Considering the flexibility that is required in designing such modern systems, rather than using commercially available data acquisition systems a modern custom-made data acquisition system is designed as part of this research work.

3 FAULT CURRENT LIMITER

3.1 Introduction

The FREEDM system is operating at 12.47kV in three phase. This converts to 7.2kV in single phase. In the event of a fault in this single phase system, fault current reaches approximately 3000 A in assumed circuit conditions in the simulation. A fault current limiter has been developed to limit this fault current to 1000 A (shown later). This reduced fault current situation enables development of advanced semiconductor based devices which can interrupt a fault faster compared to the conventional devices. The FID and the SST are being developed as part of the FREEDM project under the basic assumption that the whole FREEDM loop is operating under limited fault currents [3]. Hence the function of fault current limiter is of utmost importance to the proper working of next- generation devices in the FREEDM project.

3.2 The design

The fault current limiter designed is solid-state based. It consists of a bidirectional semiconductor switch which can be turned on and off depending on the fault current magnitude. A voltage clamping element which is the surge arrester is connected in parallel with this switch helps in controlling the surge current that might flow through the switch, and cause switch breakdown, if there was no such surge arrester. A RC (resistor-capacitor in series) snubber circuit is also connected in parallel with the semiconductor switch to protect against high $\frac{dv}{dt}$, voltage gradient with respect to time, which occurs at the turn off instant of the switch.

The basic design of the circuit is inspired from reference [21] and is shown in Figure 3.1. The semiconductor switch used for the design is an IGBT, capable of handling short circuit current of 3000 A [22]. The system designed is operating at 7.2kV single phase voltage. The surge arrester has a clamping voltage of 12kV, since it should be greater than the peak supply voltage. The RC snubber is set as 60k ohm resistance and 1uF capacitance. The current fall time, which is the fall from 100 % to 10 %, is set at 100 us (microsecond) and the current tail time, which the fall of current from 10 % to 0 % is set at 1us. These settings ensure fast operation of the switch in the event of a fault or clearing of a fault.

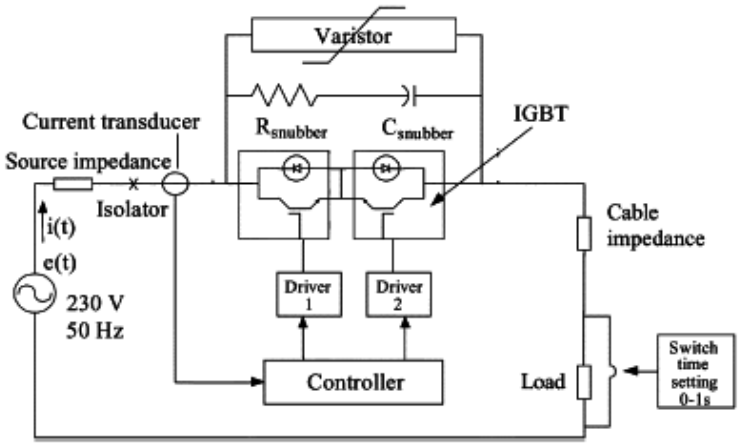

Figure 3.1 Schematic of FCL [21]

The circuit model developed in Matlab/Simulink software is shown in Figure 3.2

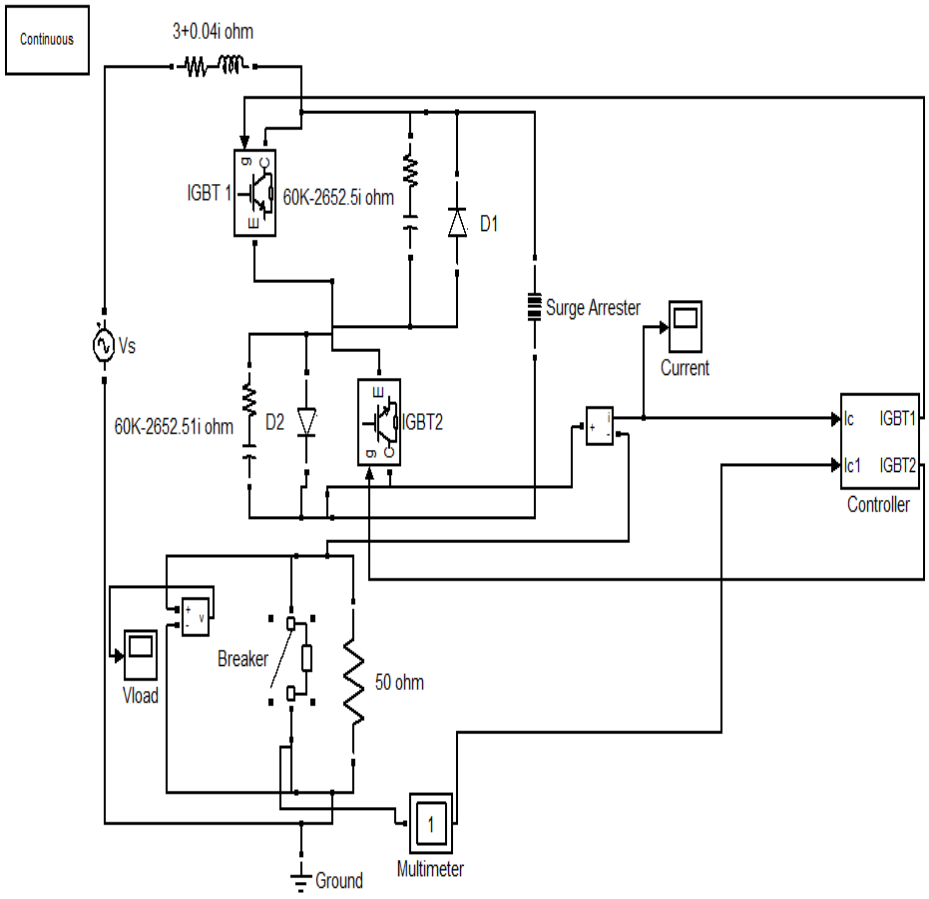

Figure 3.2 Matlab/Simulink model of the FCL

The distribution line parameters are set as 3 ohm resistance and 100uH (micro henry) inductance. A resistive load of 50 ohm is connected in series with this system. To simulate the event of a fault, a breaker switch is connected in parallel with the load. The on-state resistance of this switch is 0.001 ohm. Anti-parallel diodes connected across each IGBT ensure successful operation in both positive and negative half cycle of current.

3.3 FCL controller

The function of the controller is to provide gating signals to both IGBTs depending on the magnitude of the current. It is shown in Figure 3.3. During normal condition the IGBTs are always gated on.

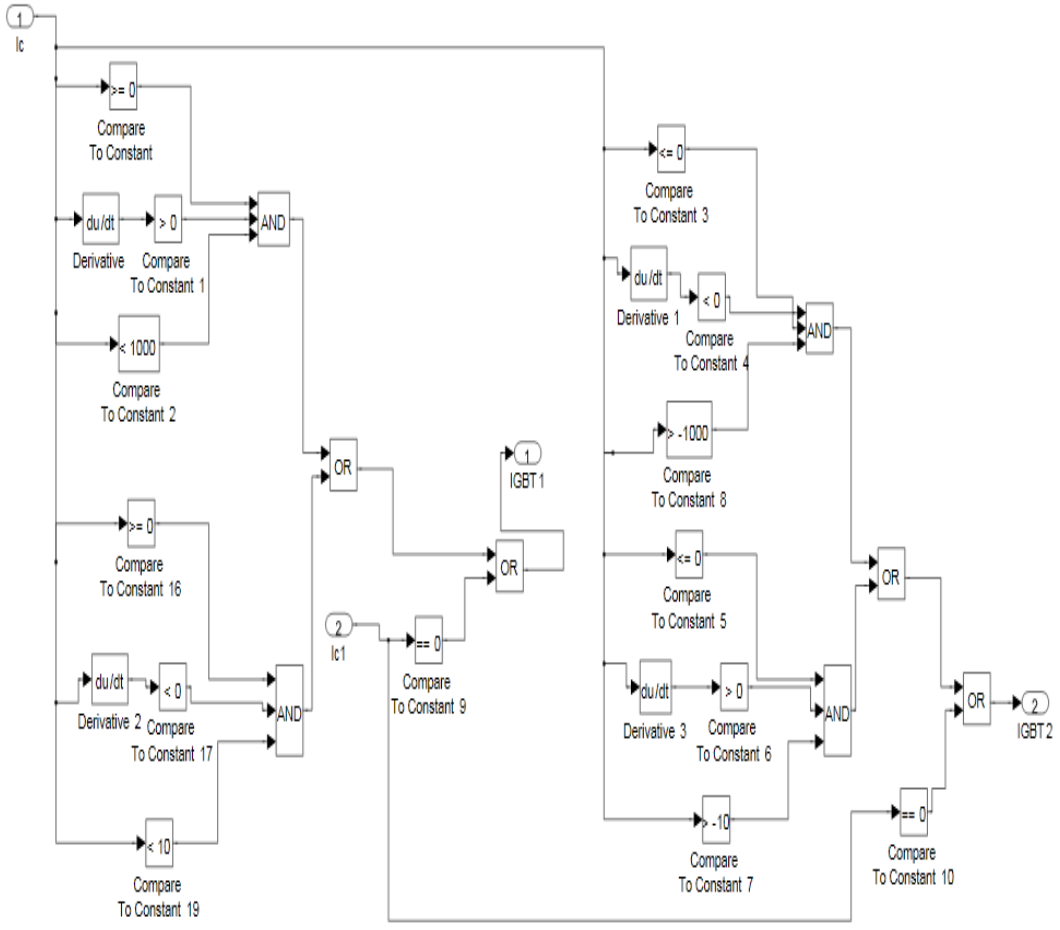

Figure 3.3 Gating signal controller of the FCL

When a fault occurs, each IGBT conducts the current till it reaches a maximum value, I_{max} and at that instant it is turned OFF. The maximum value is a pre-set value. The current is then diverted to the snubber and the varistor. The snubber damps high $\frac{dv}{dt}$ and in this process the capacitor gets charged and becomes inactive [23]. The surge arrester current builds up rapidly and so the current in the main circuit starts to decrease when circuit voltage reaches close to the clamping voltage of the surge arrester. When the current approaches a preset low value I_{min} the IGBT is turned on again to re-establish the current in the main circuit. This switching logic continues until the fault is cleared by a circuit breaker.

The working of the controller is explained as follows. The load current is measured as I_c and the current through the switch is measured as I_{c1}.The measurement I_{c1} is used to identify if there is any short circuit in the system. If its value yields anything other than a zero, it indicates a fault. The value I_c is first compared to zero, to check if the fault current is in the positive or negative cycle of the current waveform. Next, the slope of the current waveform is checked to verify if the current value is increasing. This ensures that the current is allowed to increase till it reaches a preset maximum value. At the instant when the current reaches value I_{max}, this setting is user-definable, a zero is sent to the gating signal of that IGBT thus guaranteeing that the IGBT is turned off at that instant. This will redirect the current through the snubber and surge arrester. The load current reduces in magnitude because of the surge arrester and so there is change in the slope of the current waveform when it reduces. This is verified by comparing the slope to zero. The switching strategy is maintained in such a way that this current decrease does not reach a value lower than I_{min}. At the instant current waveform reaches this minimum value, the gating signal is turned on and the IGBT is conducting current again. This switching pattern is repeated for both positive and negative half cycle until the fault is cleared, which is indicated by I_{c1}. The advantage of this switching is that, it gives ample time for coordination of this system with its protection schemes. It

also ensures that the presently existing switchgear components and protection schematics still work effectively in power systems that are expanding their power delivery capacity to meet the load demand [21].

3.4 Simulation results

The values of I_{max} and I_{min} for both positive and negative half cycle are shown in Table 3.1.

Table 3-1 Constraint values of FCL controller

Cycle	I_{max}(A)	I_{min}(A)
Positive half cycle	1000	10
Negative half cycle	-1000	-10

The supply voltage, V_s used for the simulation was 7.2kV rms. The waveform is shown in Figure 3.4. This means the peak value is 10.18kV. Hence the surge arrester clamping voltage is set at 12kV.

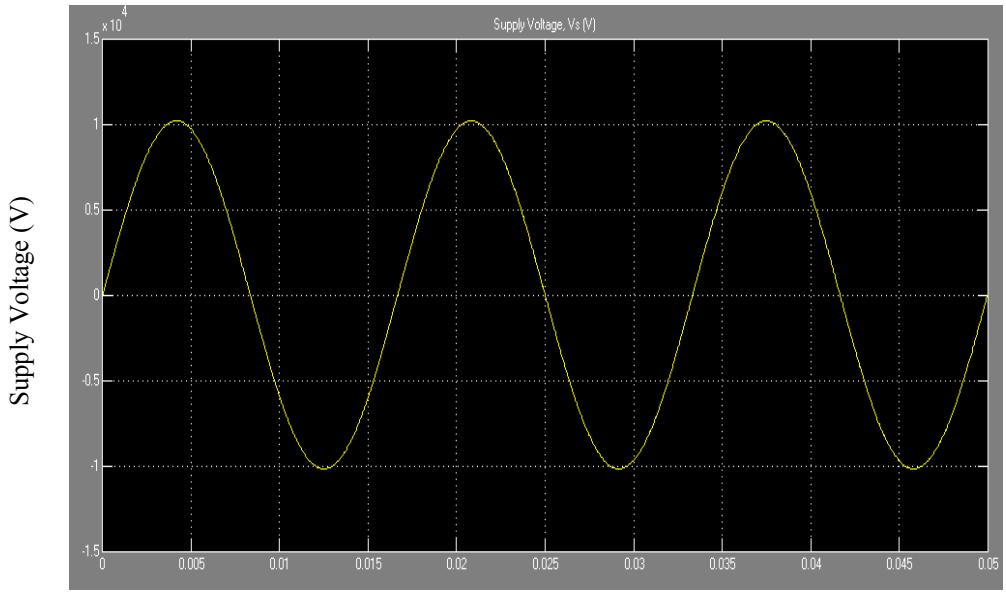

Figure 3.4 Supply voltage of the FCL

The fault creating mechanism and its timing is described in Table 3.2

Table 3-2: Fault event timing strategy

Event	Timing(s)
Circuit initiation	0.00
Line Fault created	0.02
Fault cleared	0.05

The event that a fault is created is confirmed by observing the load voltage waveform shown in Figure 3.5. At 0.02 s, the load voltage drops to a zero indicating that there is no current flowing through the load. Once the fault is cleared at 0.05 s, the load voltage resumes to normal and this is shown later in the chapter.

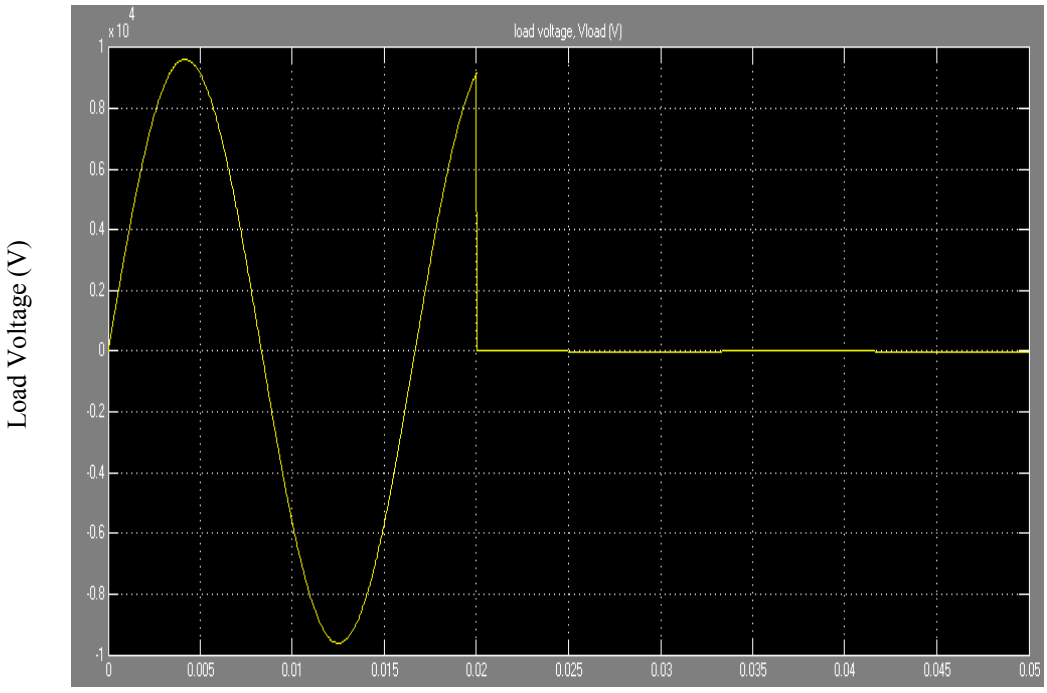

Figure 3.5 Load voltage waveform

The switching strategy employed on the load current is shown in Figure 3.6. This figure displays the switching of current waveform between I_{max} and I_{min}. The peak of the current in the event of a fault was clearly observed to be 1000 A and a confirmation of this shown later.

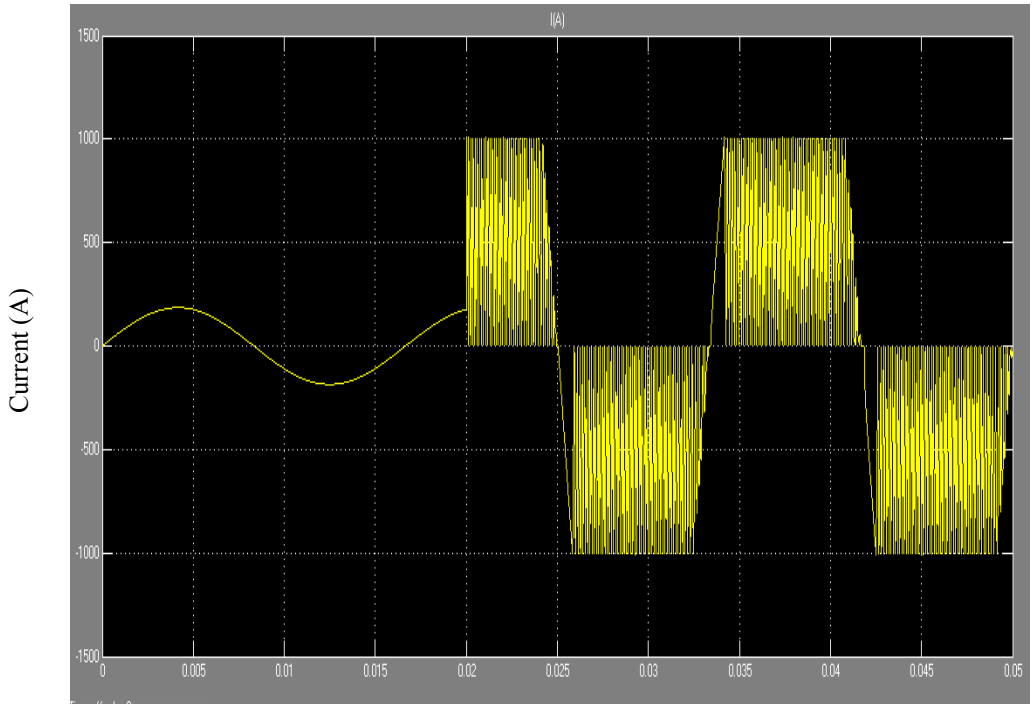

Figure 3.6 Circuit current Waveform

The effectiveness of the controller and the FCL design is demonstrated in Figure 3.7 and 3.8. The ability to control the I_{max} and the capability of the design to make this setting user-definable stand out as big advantages of this design.

It was observed that the value of I_{max}, in the positive half cycle, was always close to 1000 A and the maximum it reached at few instants was 1005 A. The I_{min} value observed was close to 10 A and the minimum it reached was 9.5 A. Similar results were obtained for the negative half cycle as well. This explains the robustness of the design and the controller, to various circuit parameter-variations like impedances, switching frequency.

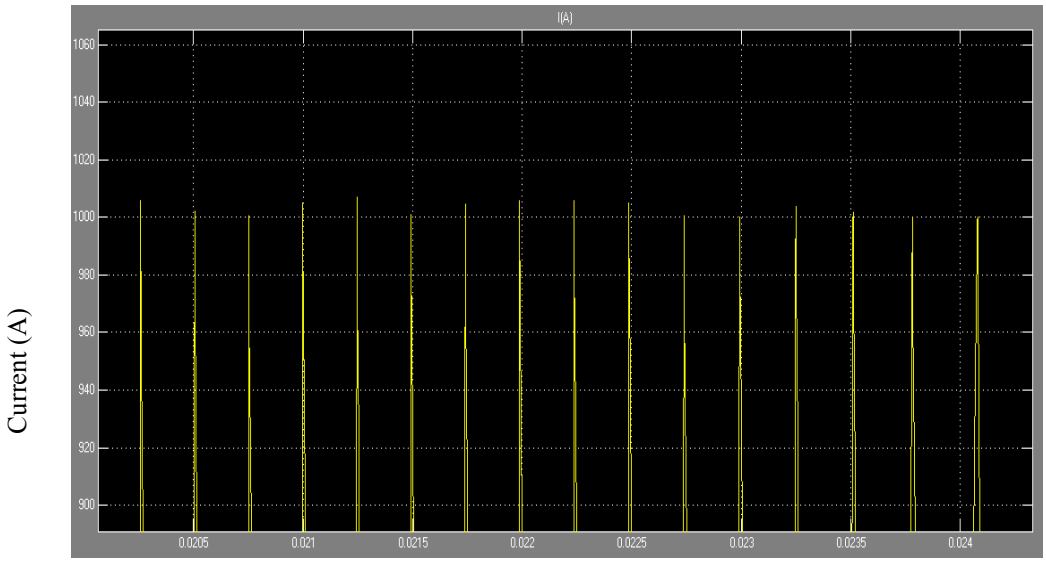

Figure 3.7 Waveform showing I_{max}

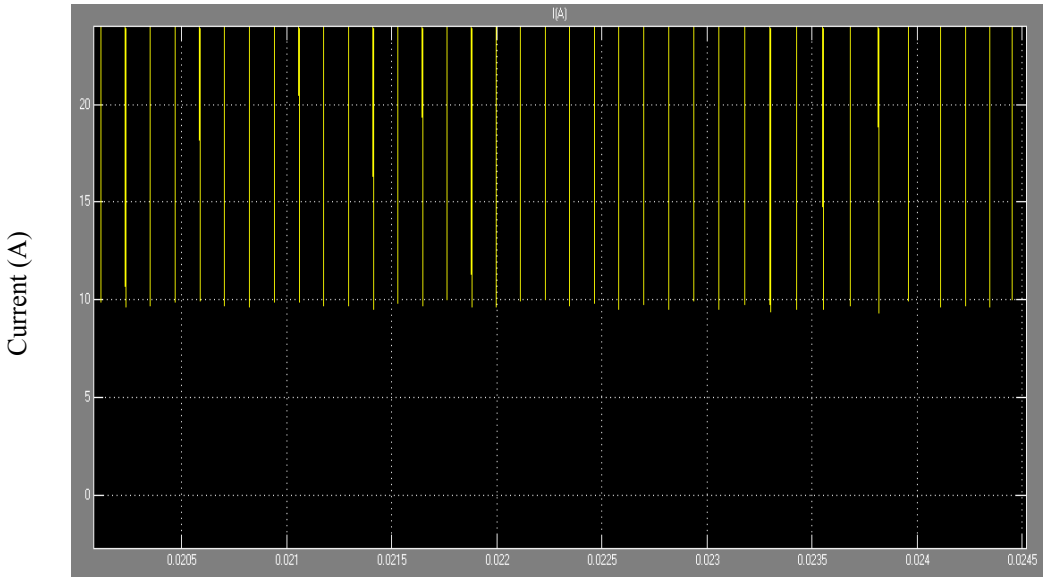

Figure 3.8 Waveform showing I_{min}

In the event of no FCL in the circuit, the fault current reached more than 3000 A in magnitude. This is shown in Figure 3.9. This figure proves that the FCL designed is capable of limiting the current from more than 3000 A to 1000 A, or any other user defined setting.

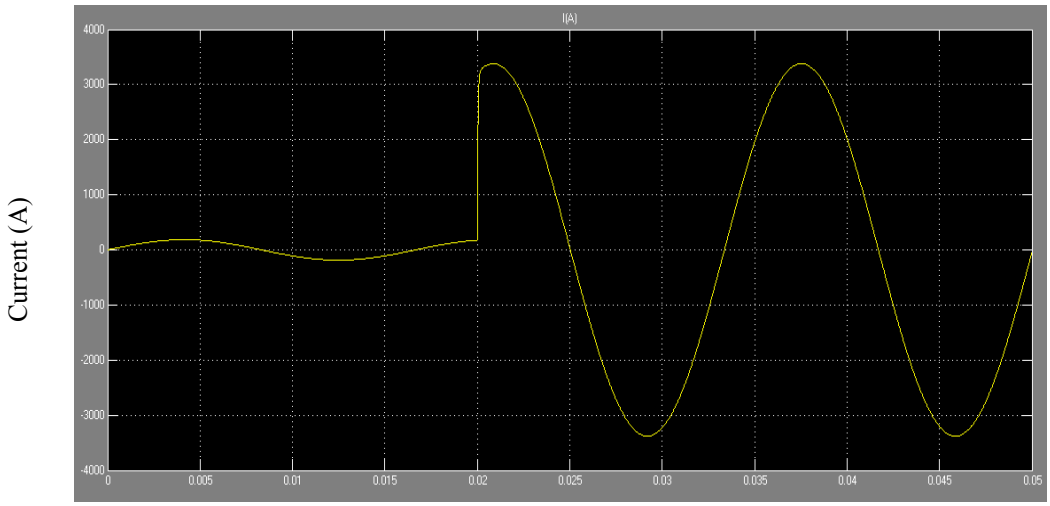

Figure 3.9 Fault current magnitude without FCL

The response of the design when the fault is cleared after 0.05 s is shown in Figure 3.10. This validates the faster response time of the controller.

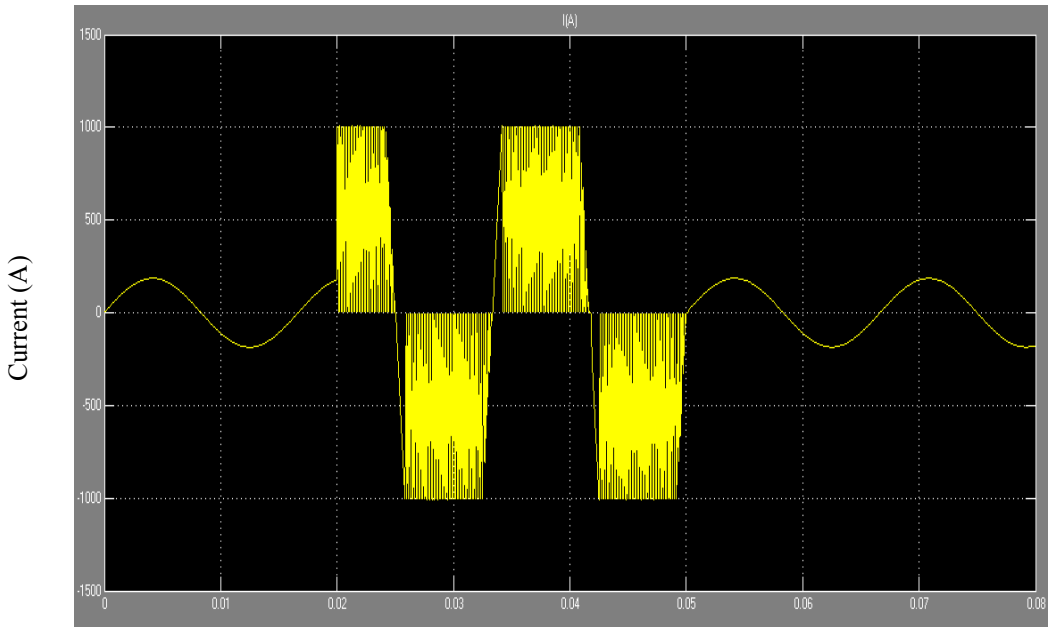

Figure 3.10 FCL response to fault clearance

The load voltage waveform after the fault is cleared is shown in Figure 3.11. This waveform shows that, the load voltage reaches a zero in ideal conditions and gets back to normal when the fault is cleared. Since Matlab/Simulink assumes that devices are ideal hence ideal responses and waveforms have been displayed assuming ideal operation of each device [24].

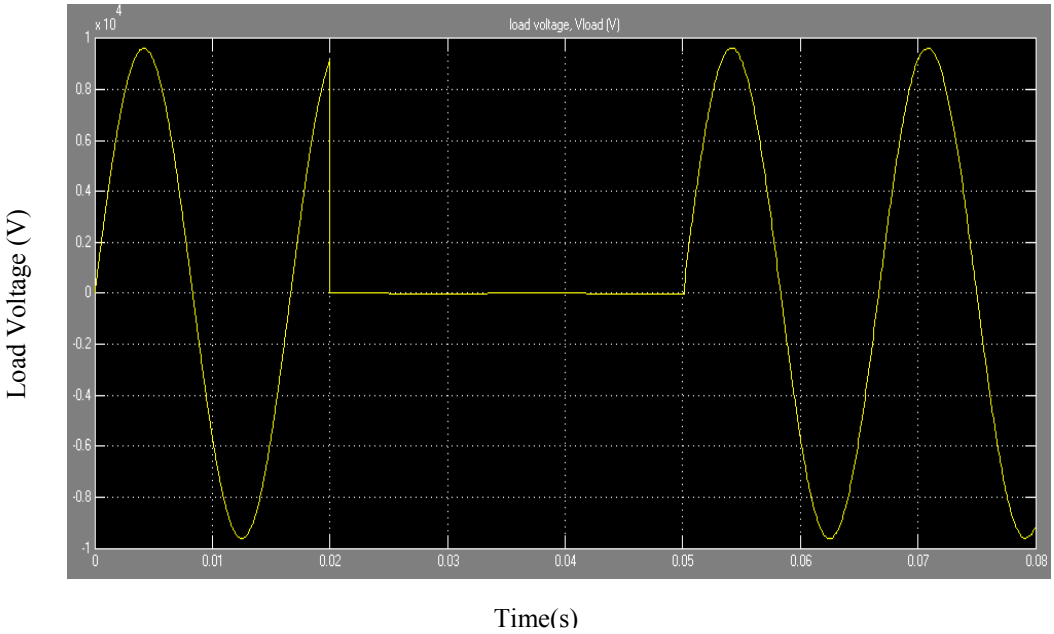

Figure 3.11 Load voltage response to fault clearance

3.5 ***Summary***

The fault current limiter (FCL) developed for the FREEDM system was described in this section. This design caters to the requirement of limiting the fault current to a user defined value and sustaining that for a period of time to allow proper coordination of the protection. The way this setting can be made user-defined is by connecting the "Compare to constant 1000" block in Figure 3.3 to the human-machine interface (HMI) which would be developed as part of the hardware setup of the FCL [25]. This way the user can enter the desired value and the controller will perform its function. This design can be implemented using currently available components thus solving the need to wait or develop customized components for the hardware setup.

4 FREEDM PROTECTION SYSTEM

4.1 Overview of the protection system

The FREEDM loop is divided into sections (or zones) for the purpose of protection. Each zone is terminated by a FID at its terminal ends. In the event of a fault inside the zone, the section FIDs interrupts the fault and isolates the zone [2]. This arrangement is shown in Figure 4.1.

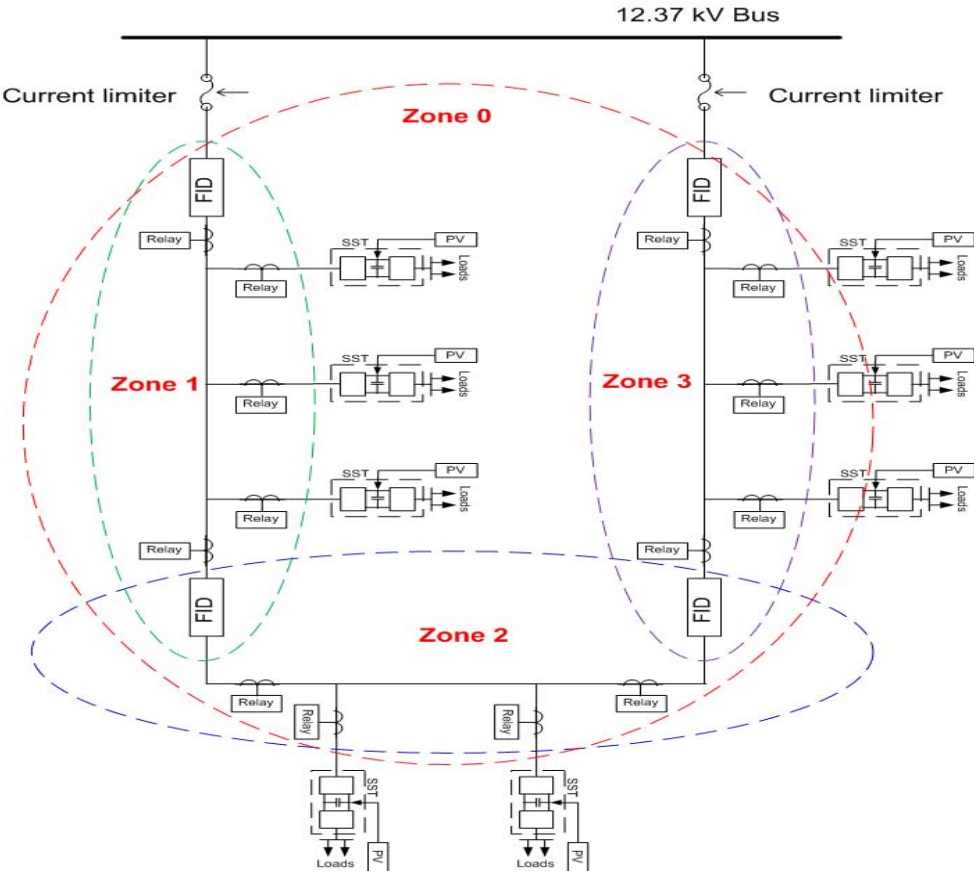

Figure 4.1 FREEDM protection zones
(Taken directly from [2])

Zone 0 acts as the backup for zones 1, 2, and 3. In the event of a fault inside zone 1, 2, or 3 and the non-operation of the corresponding section FID, zone 0 operates with a time delay and trips the main circuit breaker, as was shown in Figure 1.2. This operation isolates the loop from the local utility bus thus protecting the grid [2].

The idea to utilize existing protective relays was explored and a hardware test bed similar to one zone of the FREEDM system was built for this purpose (shown in chapter 6). The system details are presented later in chapter 6. A Schweitzer Engineering Laboratories (SEL) 387E current differential relay was used and it was set for instantaneous trip in the event of sensing a fault [26]. The trip signal generated was observed on an oscilloscope and it is shown in Figure 4.2.

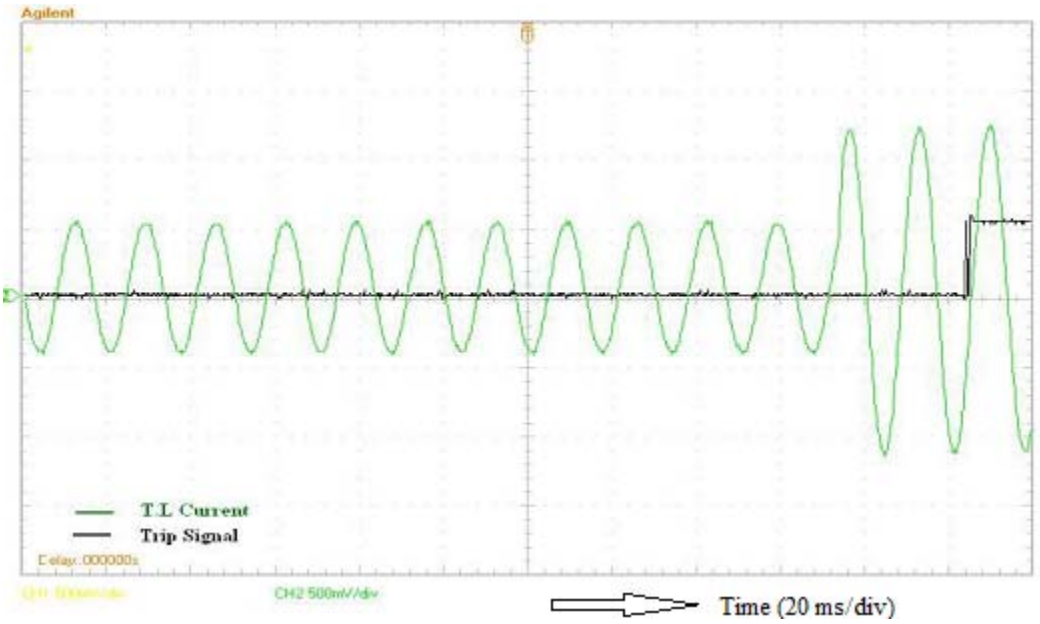

Figure 4.2 Commercial relay trip response

It was observed that the response time of this relay to a fault is approximately two cycles after a fault. Such delayed response times are not acceptable according to FREEDM project requirements. The reason for this delayed response is the calculation of rms values of the waveform by the relay and this eventually results in a response time of more than a cycle [26]. Hence it was concluded that commercially available protective relays could not satisfy the requirements of the FREEDM system and hence a custom-made protection algorithm based on IFM and a programmed data acquisition method using microcontroller has been the objective.

The IFM in each section performs the function of sensing a fault and sending a trip signal to the FID. The schematic of one zone with the IFM connected is shown in Figure 4.3.

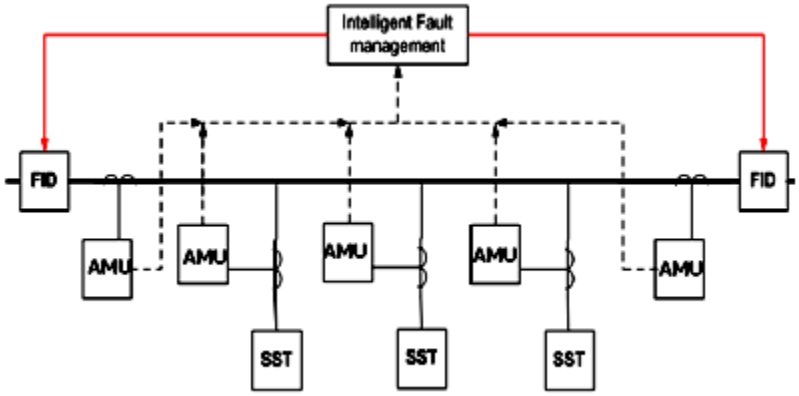

Figure 4.3 IFM and FID connection in a zone
(Taken directly from [2])

Measurements are made at various points in a zone and these measurement data are transmitted to the IFM in the path followed by incoming arrows to the IFM, which is a computer that executes intelligent protection algorithms. The primary protection algorithm executed by the IFM is differential protection and the backup protection algorithm is overcurrent protection. If the IFM concludes the existence of a fault condition it sends a trip signal to the FID through the path followed by outgoing arrows from the IFM [2].

The accuracy of the protection algorithm is improved by adding GPS time synchronization to the measured values to synchronize the data acquisition process before they are transmitted to the IFM. This ensures safe and reliable protection [27].

4.2 Protection algorithm

The algorithm programmed into the IFM executes both differential protection and over current protection method in Labview environment. After the data is acquired from each measurement point the algorithm verifies if each measurement from every data point was measured at the same time instant. A flowchart explaining the protection algorithm is shown in Figure 4.4.

The complete protection algorithm, the data acquisition system and the interconnection between these two have been developed assuming only three measuring points in a zone. The

proof of concept, verified using three measurements can be extended to nearly 12-15 measuring points after considering the total time allowed for the protection algorithm to respond to faults.

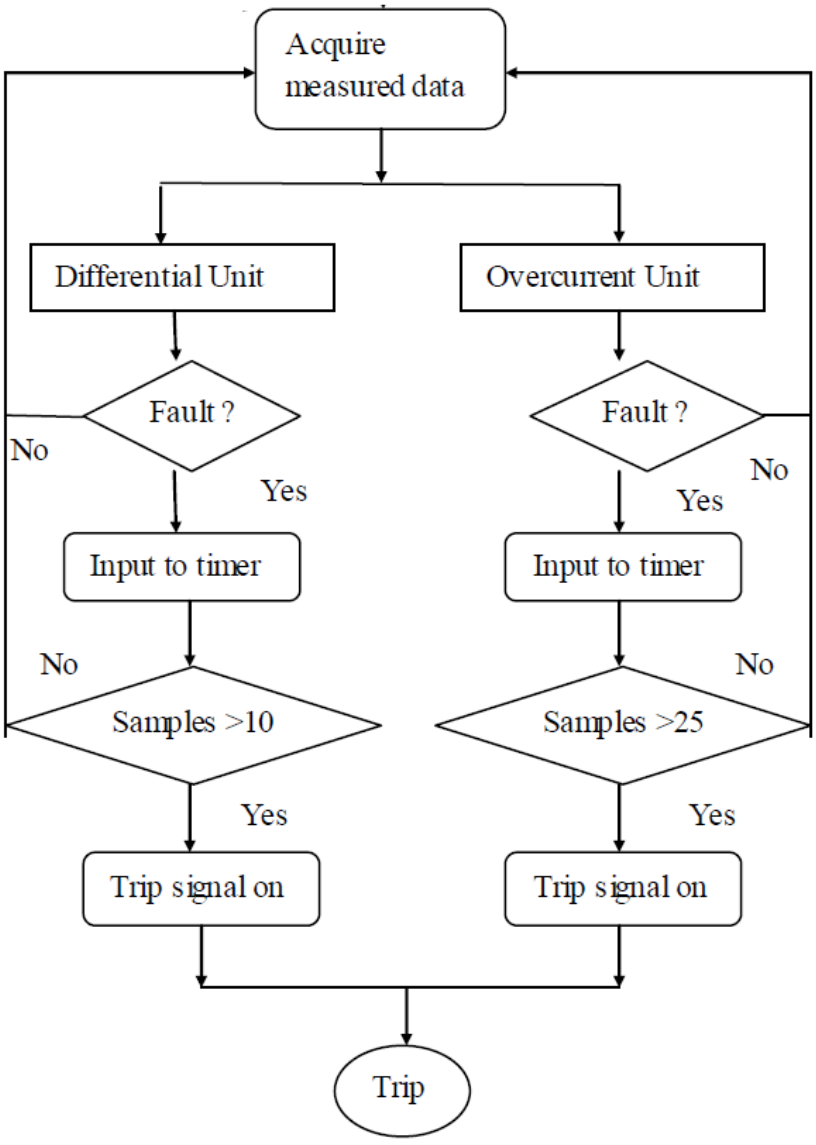

Figure 4.4 Protection algorithm of the IFM

Once the data is acquired from the DAQ device to the computer, it is input directly to the Labview model developed to execute the protection algorithm. This acquisition is done by requesting for data through the Internet Protocol (IP) address of each microcontroller data acquisition board and also the port of each board, through which the data is sent it out [16]. The request

sent is automatically done using the TCP/IP protocol and hence no manual setting is required for this purpose. The system could be considered analogous to delivering letters to the mailbox by any carrier. The IP address is similar to the address of the recipient to whom the letter is meant for and the port number is similar to the gate through which the letter needs to be delivered. Any communication with the microcontroller from the computer has to be executed through the port and be addressed to the particular IP address of each board.

The microcontroller transmits the data on receiving a request from the computer and data is finally acquired by the computer. This data in the form of digital samples is input to both the differential and the overcurrent unit algorithm programmed on Labview. Since currently only three measurements are considered in a zone, the system under study becomes two source currents feeding one load. Hence the load current should be the sum of the two source currents under normal circumstances. The function of the differential unit is to add these two source currents acquired from their corresponding microcontroller boards and compare it to the load current data acquired from its corresponding microcontroller board. The comparison equation also permits a minimal restraint value below which the differential unit does not conclude a fault. When the calculated value is greater than this restraint, a fault condition is confirmed and the internal counter/timer on Labview which was initialized to 0 will be incremented to 1. If the calculation result from the next sampling point yields a conclusion as a fault then the timer/counter is incremented from 1 to 2 and this continues until a fault condition is concluded by the differential unit. If the timer value reaches 10 then a trip signal is issued to the FID. In case if any one calculation of the differential unit finds the non-existence of a fault, even though the timer has a value of 9, a reset signal is issued to the timer and it returns back to 0.

The reason for monitoring 10 continuous sample points is to ensure the accuracy and the reliability of the protection algorithm. Since the data acquired is instantaneous sample points and not rms as in conventional protection systems, there is a possibility that instantaneous values of the current and voltage waveforms can have momentary flicker and this should not be considered

a fault situation [28]. Hence 10 continuous sampling points are required to confirm a fault. Since 10 continuous sampling points need to be monitored, to achieve a faster response of the protective algorithm, data needs to be acquired faster such that the decision on the primary protection could be made in less than one quarter cycle, which is the FREEDM system requirement [1].

The overcurrent unit also acquires the data similar to the differential unit. In this case, it does not add the sample points, rather it individually compares each sample point data to a preset value. This preset value is set depending on the circuit parameters. If any sample point exceeds this preset value, it indicates a fault situation and the internal timer corresponding to the overcurrent unit is incremented from 0 to 1. Again, if the next sample point also exceeds the pre-set value, this value is incremented from 1 to 2 and it continues. If 25 such continuous samples are all above the pre-set value, then a trip signal is issued to the FID by the IFM. If even one sample point does not exceed this pre-set value, then the timer is reset back to 0 and it is concluded that the fault has been cleared by the differential fault or it was a temporary fault. Since 25 samples are monitored in this case, compared to 10 samples in the case of differential unit, this method ensures that the overcurrent unit operates as back-up to the differential unit. In the event of a fault, this situation is equally assessed by both the units, but since the differential unit acting as the primary protection operates first and sends a trip signal not all the 25 sampling points of the overcurrent can exceed the pre-set value and hence perfect synchronization is achieved between both the protection units.

Only the instantaneous values are considered by the protection algorithm as opposed to the technique of commercial relays, and so no time is consumed on rms calculations which justify the assumption of ultra fast decision-making process of this IFM algorithm. Only care needs to be taken in setting the restraint limit for the differential unit and the pre-set value for the overcurrent unit, which can always be fine-tuned based on system parameter calculations and the system performance. The IFM algorithm needs to send a trip signal in less than one quarter cycle through its primary protection algorithm and so the role of the data acquisition system assumes utmost impor-

tance to facilitate this function. This thesis is focused upon the data acquisition system for the FREEDM IFM system.

4.3 Data Acquisition System

Current is measured from the FREEDM loop through current transformers (CT). A resistor is connected at the output terminals of the CT and this current signal is converted to a voltage signal measured across the resistor. This voltage signal is hardwired into the data acquisition system that has been developed using microcontroller [2]. The architecture is shown in Figure 4.5.

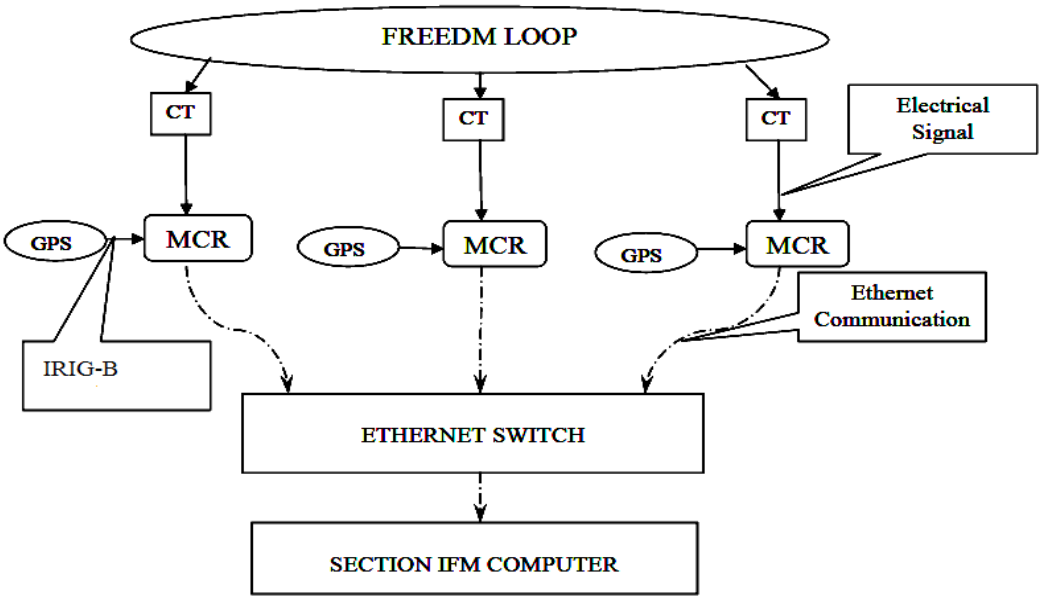

Figure 4.5 Schematic of the data acquisition system

The analog voltage signals are input to the analog to digital (A/D) channel port of the microcontroller, represented as MCR in the figure [29]. Simultaneously time synchronizing pulses are added by the GPS (Global Positioning Satellite) receiver to one other input port of the microcontroller. The pulse frequency is fixed depending on the required operation speed of the microcontroller. The basic format used for this time communication between the GPS and the microcontroller is the Inter-Range Instrumentation Group (IRIG-B) time code [30]. Satellite synchronized pulses are received from the GPS in this time format and input to the microcontroller as hardwired cable. The function of the microcontroller is to perform synchronized A/D conversion. Synchro-

nized A/D conversion time is achieved by choosing three microcontrollers of the same configuration and performance capabilities. Synchronized start time of each A/D conversion is achieved through the GPS pulses, which will trigger each A/D conversion. The GPS receiver is synchronized to the satellite, this ensures a highly synchronized data acquisition process by each microcontroller.

The output from the microcontroller is transmitted to the computer through the Ethernet channel on the TCP communication protocol. Currently available Ethernet speeds of 10 Mbps or 100 Mbps are used for this purpose depending on the data latency that is observed [31]. Latency is the time lag between the data observed and its actual source. For example, if at the start of a cycle of an analog voltage signal, a spike occurs in the actual system due to some switching and this spike is observed 2 cycles later at the monitoring station, the latency of the whole data acquisition system is 2 cycles. Since the FREEDM system requires the fault decision to be made in less than one quarter cycle, a system with latency less than this would be ideal. This latency-permitted determines Ethernet network speed. A microcontroller which has an Ethernet controller that can support the required network speed is the automatic choice.

The digital sample data from each microcontroller are transmitted to a section IFM computer through an Ethernet switch. The Ethernet switch again should support the required network speed. Moreover the buffer in the switch, a ring buffer, should be large enough such that with fast transmission of data the buffer should not get overfilled [31]. If such an event happens, then the data transmission speed from the microcontroller to the switch is slowed down which will increase the system latency. Since the TCP protocol is very reliable, until a data packet is delivered to its destination IP address, it is preprogrammed to keep resending the packet [17]. This might cause heavy traffic in the ring buffer of the Ethernet switch especially with three microcontrollers simultaneously transmitting data packets. In some cases, it might also lead to packet loss, meaning loss of useful information. This loss of information is very critical to this data acquisition system designed for protection purpose since it might significantly affect the trip decisions and proving to be

a disaster to the FREEDM loop and its components. Hence a switch with a higher speed than the network speed and a greater buffer size than the expected network traffic is chosen.

Once the data is transmitted from the switch, it reaches the computer, which is calling for data from the respective IP address and port of each microcontroller. The Ethernet output from the switch is connected directly to the Ethernet port of the computer which is internally connected (by itself) to the Ethernet controller on the motherboard of the computer [32]. This Ethernet controller should be able to match the speed and buffer rate of the Ethernet switch. This will reduce the latency and avoid any speed or buffer mismatch between the switch and the computer's Ethernet controller. The IFM algorithm is running on a section IFM computer in the Labview environment which acquires all the data from the switch output. The basic data structure of the TCP protocol enables the microcontroller to send its IP address as the source address and the computer's IP address as the destination address [16]. So, the Labview algorithm requests data individually from the three microcontroller boards and hence three data streams are achieved. The experimental results for this data acquisition system is shown in Chapter 6.These time synchronized data streams are input to the protection algorithm that executes its functions discussed in chapter 4.2.

4.4 Summary

This chapter described the FREEDM loop protection system design. The division of FREEDM loop into sections enables a greater flexibility in design of the protection system since trip operations can be executed for a zone separately to isolate the zone and moreover the data acquisition latency would be more if data needs to be acquired at every measurement point in the FREEDM loop. The protection algorithm is novel as it is based upon instantaneous values on the current waveform and is reliable too since many sample points are considered before making a decision. The data acquisition system has many constraints as discussed in chapter 4.3 and the choice of components for this and the integration to develop a working system is a major challenge. The components chosen to perform each function and the reason behind each choice are described in the chapter 5.

5 DATA ACQUISITION SYSTEM DESIGN

5.1 Requirements of the DAQ Design

Based on the description of the data acquisition system in Chapter 4, the requirements or the criterion based on which the components of the DAQ have been chosen, are given below:

- Capable of converting analog voltage to digital sample output (A/D).
- At least 8 bit A/D converter to ensure good output voltage precision.
- Ability to sample at least 1000 samples per second.
- Capable of transmitting the data through Ethernet.
- Ability to support transport layer of the TCP protocol at least a speed of 10 Mbps.
- Device frequency at least 10 times greater than the required sampling speed of the A/D.
- Ethernet buffer greater than 1 Kilobyte (Kbyte) to prevent TCP packet choking in the microcontroller.

The microcontroller should be able to acquire hard-wired data, process it through its A/D converter and send the data out through the Ethernet.

The logic behind expecting a capability to sample at least 1000 samples per second and 1 Kbyte Ethernet buffer is that, when there are more samples than 1000 in a second, the Ethernet buffer at a size of 1Kbyte will get filled with the first 1000 and the rest other samples will choke the TCP transmission process or they get lost [33]. Hence the A/D sampling speed and the Ethernet buffer size need to be matched, else the Ethernet buffer size should be greater than the A/D sampling speed.

The device frequency needs to be at least 10 times more than A/D sampling speed and this is because, the microcontroller device performs parallel-processing executing both the A/D and the TCP functions in parallel along with its own device functions which are not known to the user [34]. Moreover, the TCP protocol structure is huge and requires many functions like setting a

socket, opening a port, transfer the data from buffer to the socket through the port, to be completed to successfully execute a packet transmission.

The selected device should be able to support a speed of 10 Mbps because more than half the speed is utilized for the internal process of the TCP architecture and the speed does not represent the transmission speed, it actually represents the speed of the Ethernet controller on the microcontroller board. Higher this speed, faster is the TCP process inside the microcontroller [33].

5.2 Simulation of the DAQ system

A Matlab/Simulink simulation was performed for this DAQ system. The model is shown in Figure 5.1

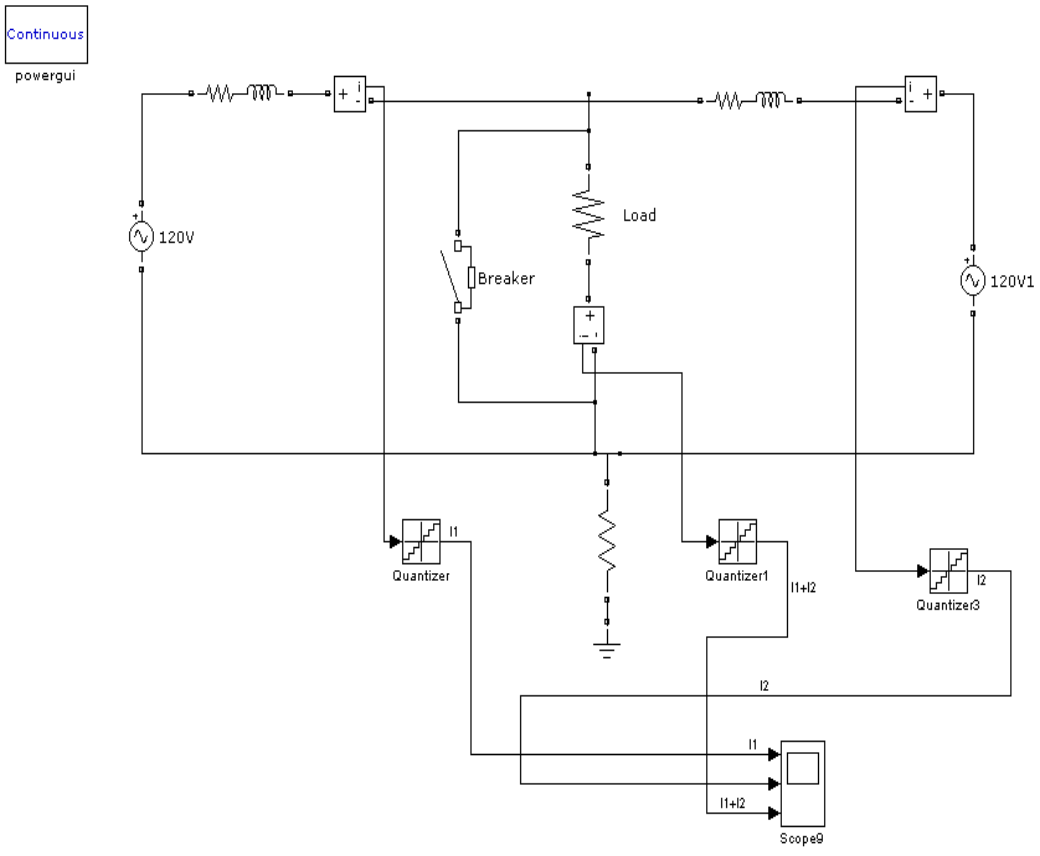

Figure 5.1 Simulation circuit of the DAQ

The 120 V sources on either side of the load indicate that the load is fed by two sources from either side. The nominal current in each branch is 5.3 A under non-fault condition. The current through the load is 10.6 A. The fault condition is simulated by a breaker switch with negligible resistance, which is connected in parallel with the resistive load. The current in each branch is measured using current transformers and these are input to the digitizer blocks. The digitizer block is analogous to the microcontroller block (MCR) of data acquisition system schematic from Figure 4.5. This digitizer converts the analog current signal to its equivalent digital samples. The communication using Ethernet could not be simulated because the main concern is the latency caused by the TCP module and this latency is a variable in the actual system and the variation could be from 2 ms to even 300 ms [33]. Simulating this latency using a delay block is possible, but not useful since there are numerous possibilities and hence this part is expected to be studied using the actual DAQ hardware developed.

When a fault is initiated, a fault current of magnitude 10.2 A flows through each of the branches feeding the load. Since the breaker is closed, this high magnitude of 20.4 A fault current flows through that and nothing through the resistive load. The analog current waveform from one of the sources is shown in Figure 5.2.

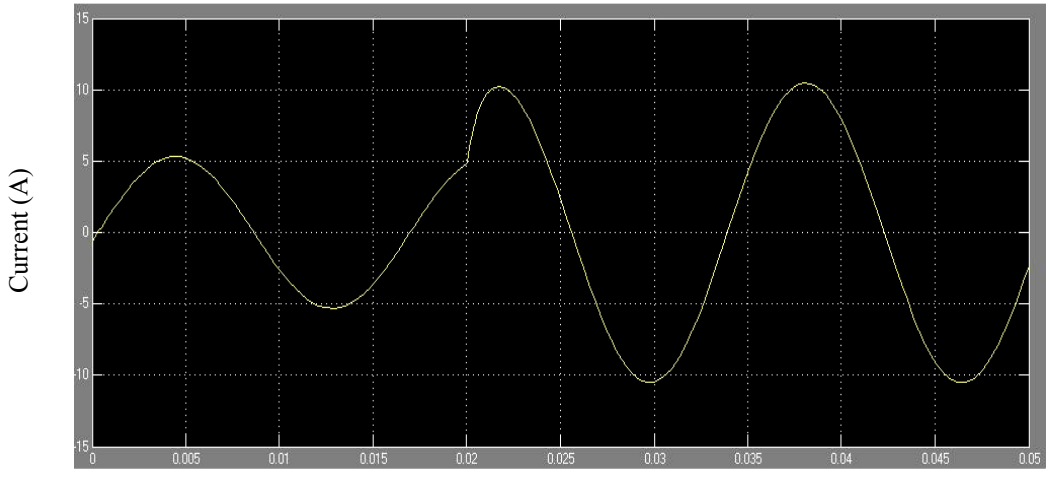

Figure 5.2 Analog current waveform of source current

The digitizer produces digitized samples of these analog current values and the waveform from the three measurement locations is shown in Figure 5.3. I1 and I2 indicate the two source currents and I1+I2 indicate the load current. As can be observed from the waveform these are

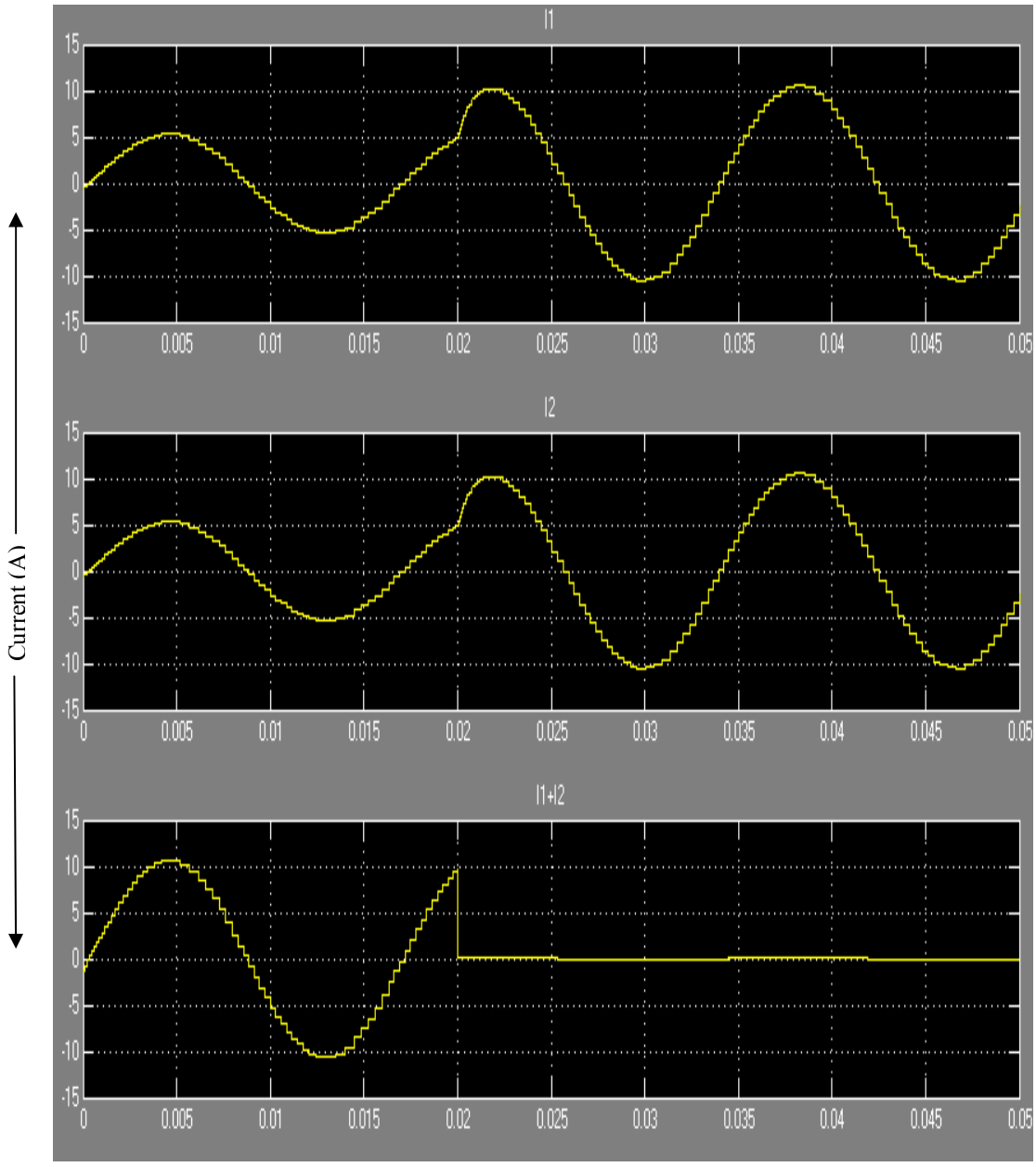

Figure 5.3 Sampled digital current waveform from simulation

digitized sample points and a fault condition could be clearly identified by the absence of any current flowing through the load. The purpose of using digital values instead of analog signals is that, these digital samples can be utilized to provide information about the instantaneous value of the waveform [19]. While in the analog signal, to obtain the waveform characteristic, an acquisition of one full cycle needs to be completed to analyze the rms value of the waveform. This rms value is utilized in performing fault identification process and hence conventional protection systems take more than one cycle to isolate a fault, as observed in Figure 4.2.

The sampled digital values were transported to an excel sheet along with their time stamps. The time stamp was generated using the internal timer of Matlab/Simulink. The digital values are shown in Appendix A. The time represents the time stamp since the start of the simulation. Only the first 10 sample points are shown. Similar results were obtained for all the digital values observed.

It is observed from these values that, I1 and I2 when individually added match exactly with I1+I2. In the event of a fault, I1+I2 would be very small, almost zero and I1 and I2 would individually carry higher digital values, thus indicating a fault. The availability of time stamps of each measurement ensures, only those digital values with the same time stamp would be compared to make a trip decision. This confirms the successful operation of the data acquisition system to enable a differential protection.

5.3 PIC18F97J60 microcontroller

Based on the design criterion the PIC18F97J60, an 8 bit microcontroller from Microchip technology has been chosen [35]. The main features of this microcontroller that satisfy the design requirements are:

- 10 Mbps integrated Ethernet controller.
- Compatible with 10/100/1000 Mbps networks.
- 41.667 MHz maximum device oscillator frequency.
- 10 bit A/D converter.

- Operating voltage range of 2.35 to 3.6 V.
- Five timer modules, enabling flexibility in programming logic
- Configurable transmit/receive buffer size.
- In-built TCP/IP stack structure.

If the required sampling speed is 1000 samples per second, then the sampling frequency of the A/D converter is 1 kHz which is very small compared to the device frequency of the micro-controller. Moreover, the A/D converter has a 10 bit precision and its permissible input voltage peak is at 3.6 V. This results in a voltage precision of 3.515 mV, which is high precision output for an A/D converter. The 3.6 V input voltage corresponding to the microcontroller's core voltage, VDD, is obtained when the on-chip regulator is enabled as shown in Figure 5.4 from the device data sheet. This regulator is always enabled unless turned off manually through the program. The regulator is always ON for this application [35].

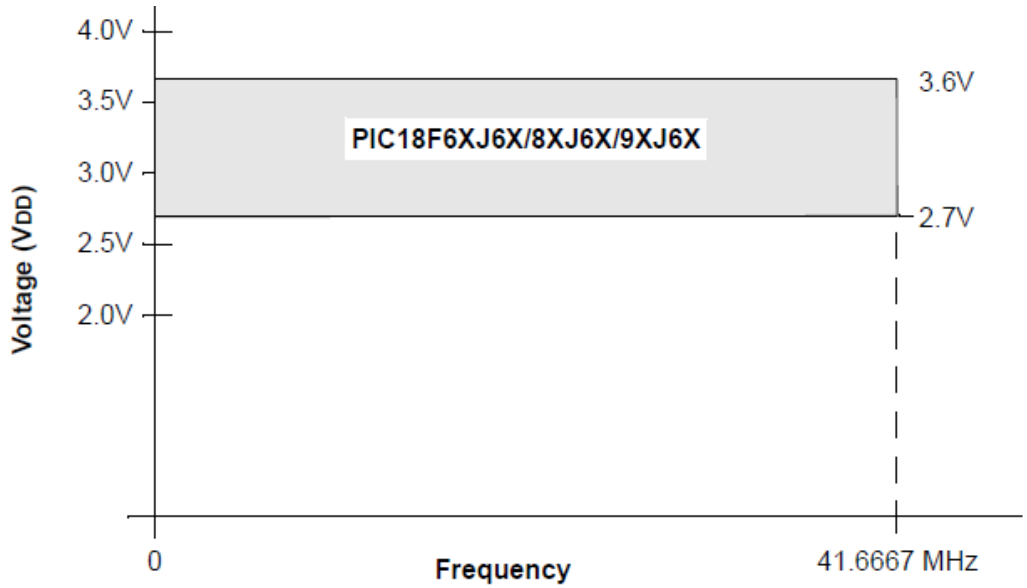

Figure 5.4 Device voltage corresponding to frequency
(Taken directly from [35])

The device has a 10 Mbps Ethernet controller that is compatible with even a 1000 Mbps network which will improve the speed of the data transmission from the microcontroller to the

remote client, which is waiting for the data through the high speed network [31][33]. Overall, that enhances the speed performance of the microcontroller.

Availability of timer modules, which are required for timed data processing and also parallel processing the microcontroller's internal functions like stack task, LCD update, stack applications enables optimized performance of the microcontroller and also increases the flexibility of the code developed. The in-built TCP/IP stack structure is the best advantage of this microcontroller since programming a new stack is a big project and it would be time consuming [36]. Hence the whole program to execute the functions such as A/D conversion and the TCP communication between the computer and the board is written onto this stack structure. Moreover, a configurable transmit/receive buffer implies that the Ethernet buffer can be used either for transmit or receive purpose or for both the purposes. The other option is to allocate a part of the buffer to transmit and the rest to receive. But for this current application, it involves sending bytes of data to the computer and receiving only acknowledgement and handshake signals from the computer, so the transmit buffer is allocated to be bigger than the receive buffer.

5.3.1 Oscillator

The oscillator system in PIC18F97J60 is different from that in the other PIC18F's. The reason is the requirement for a stable 25 MHz clock for the Ethernet module [35]. Hence the oscillator system should be capable of generating this frequency and many other frequencies. Accordingly the oscillator frequency ranges from 2.7778 MHz to 41.6667 MHz. The internal oscillator is 31 kHz and is slower compared to the requirements in Chapter 5.1. The internal oscillator is not used in the project because of its slow speed. The oscillator frequency is set using OSCTUNE register which is an 8 bit register. Once this register value is set, an external oscillator corresponding to the setting should be connected to the OSC1 and OSC2 pins of the microcontroller, as shown in Figure 5.5. The oscillator frequency is the heartbeat of all the microcontroller operations. Higher this speed, faster the operations and better the performance of the microcontroller. The A/D sampling speed is also controlled by this oscillator speed and a proper coordination of the A/D sam-

pling and the Ethernet transmit speed is effected by varying the oscillator speed. Moreover other functions such as constant calls to TCP stack and its applications and functions such as checking for TCP socket, sensing interrupt are directly linked to the oscillator speed. Higher the oscillator speed, faster these operations.

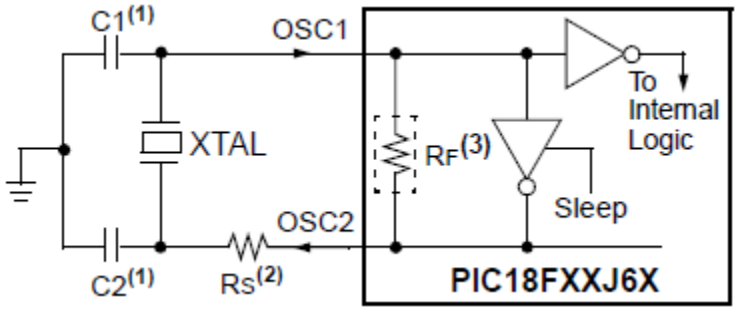

Figure 5.5 External oscillator connection to the PIC
(Taken directly from [35])

There are commercially available PIC boards with an on-board PIC18F97J60 and these boards will have the oscillator connected to a high speed crystal, which has the capability to generate 41.6667 MHz because they will have the Ethernet module on-board as well. Hence the choice of the right board solves the issue of building an oscillator circuit. More information on the choice of board is explained in Chapter 5.4.

5.3.2 Interrupts

There are thirteen interrupt registers available on the microcontroller but for the project, the only interrupt register of interest is the INTCON3 [35]. Every register has the similar organization of bits, except that each bit corresponds to the interrupt generated by a different source. These interrupt sources have three bits in general, to control their operation.

- Flag bit - To indicate that an interrupt event has occurred.
- Enable bit- Allows the program execution to branch to the interrupt vector address based on its priority level, when the flag bit is set.
- Priority bit- Assigns high priority or low priority to an interrupt source. The high priority sources are serviced first and then the low priority interrupt.

The IPEN register is used to globally enable all low priority and high priority interrupts. On device power-on, or device reset every interrupt register assumes a default priority level for each interrupt source and this setting could be accepted by globally enabling all low and high priority interrupts. The other way to work is to individually enable each interrupt by setting its enable bit as 1. Another way of working this problem out is to enable globally every high and low priority interrupt but only place those interrupts to be serviced, which are required for the application, inside the interrupt service routine (ISR). This way any device related interrupt that needs to be serviced does not get hindered and also the required interrupt source for the application gets served [37]. The third approach is utilized in the program of the microcontroller.

The INTCON3 register has the enable and flag bit of the INT1-interrupt set which is utilized in the program when a GPS is connected to the microcontroller. More about the connection is explained in Chapter 5.5. From the programming point of view, whenever a GPS pulse is received, an interrupt flag is created and the program in the ISR for this interrupt is serviced by the microcontroller. The most important idea is that, whenever an interrupt flag is generated the microcontroller's processor stops its current execution, stores that current execution command's address in a stack and goes immediately to the ISR. When the commands corresponding to the interrupt are executed completely, the processor returns back to the main program, pops out the address from the stack and executes functions corresponding to that command [37].

5.3.3 I/O ports

The microcontroller has 9 Input/output ports, port A, B-H, J [35]. Of these only ports A and B are used in this application. Port A inputs hardwired analog voltage signal from the current transformer secondary and the port B accepts constant pulses from the GPS. Bit-addressing of these ports is possible, and hence checking for interrupts or enabling only a pin as an input pin and the rest other pins as output is also possible. In this application, only one pin in port A is used for acquiring voltage signal, similarly only one pin in port B is used to acquire GPS pulses. The reason to choose port A, and the specific pin RA3 is that this pin is one of the available A/D input

channels available on the demo board used for this application [35]. The choice of port B and pin RB2 was influenced by the external interrupt function embedded into that pin and the port of the microcontroller. More on these ports is discussed in Chapter 5.4.

5.3.4 Ethernet Module

The Ethernet module requires 3.3 V on VDD while the transmission is in progress, since the microcontroller has 3.6 V on its VDD at 41.6667 MHz (from Figure 5.4), it enables successful operation of the Ethernet module. It has 8 Kbyte RAM buffer for storing packets that have been received and those packets that are to be transmitted. The buffer allocation to both receive and transmit buffer is programmable and for this application, the buffer is more a transmit buffer. Before the packets reach the transmit buffer and also before they are accepted to the microcontroller from the receive buffer they have to follow the same path and reach the Ethernet data (EDATA) register of the microcontroller [38]. This is shown in Figure 5.6. The EDATA register acts as a window between the microcontroller bus and the Ethernet buffer, thus enabling reading and writing to the Ethernet buffer from the microcontroller special function register (SFR).

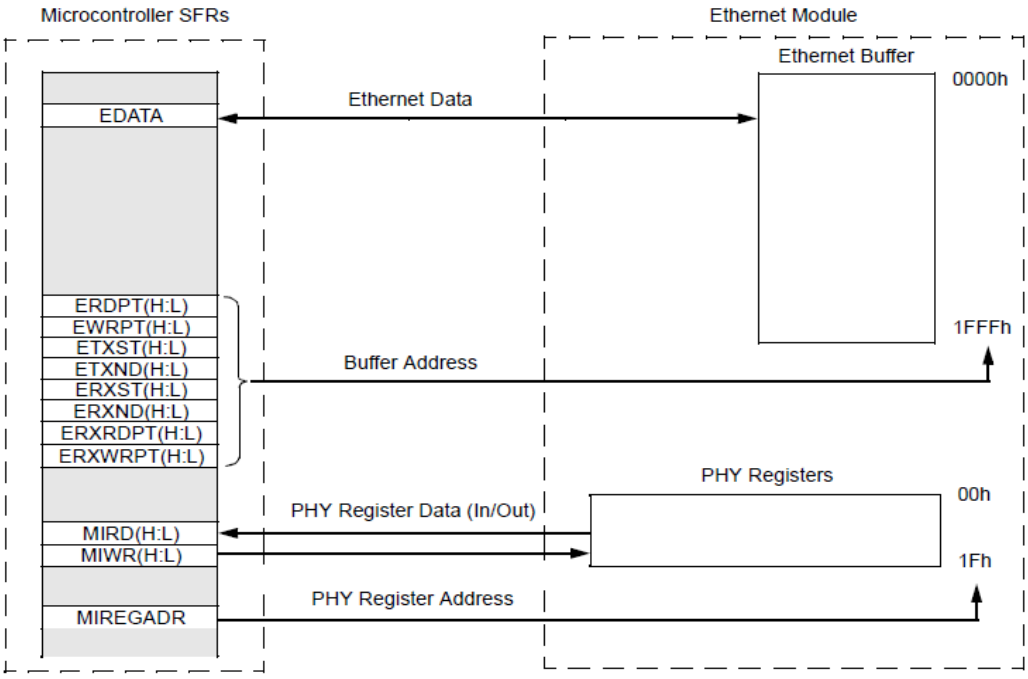

Figure 5.6 EDATA register in the microcontroller
(Taken directly from [35])

The Ethernet buffer is a circular buffer such that data enters at one point in the buffer and leaves at a different point forming the structure of a circle [38]. This is shown in Figure 5.7.

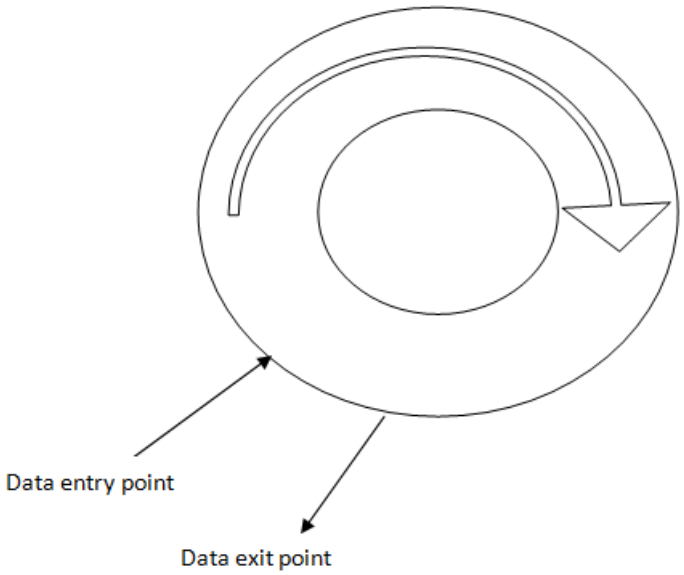

Figure 5.7 Circular ethernet buffer

The straight line arrows indicate the data transaction points into and out of the buffer and the solid curved arrow indicate the flow of data inside the buffer. This structure prevents choking of data at the entry and exit points. If there was only one point for both entry and exit, in this situation both entry and exit operations cannot work at the same time and hence might lead to choking of data either inside the EDATA buffer of the microcontroller or the Ethernet buffer. Hence having two separate points solves the problem of choking.[38] Moreover a circular buffer enables that continuous stream of data can be added to the buffer and also continuously transmitted out of the buffer without requiring any change to the pointers. The pointers are required to point to the exact location of the data to access that data. But with circular buffer, the pointer concept is no longer required thus reducing computation time of data receive and transmit operations and improving the Ethernet controller performance [38]. For this application, the same circular buffer architecture is used thus elimination data choking and ensuring faster data processing from the buffer.

The above process explained how the data exchange transaction works between the Ethernet buffer and the microcontroller, and also inside the Ethernet buffer. The process through which the data is transported from the Ethernet buffer to a remote computer is explained next. This is handled by the coding and the TCP/IP stack that has been pre-programmed into the microcontroller [39] . Once the routine check of data size and available size in one Ethernet packet are calculated, a release command is issued specifying the number of bytes of data to be released from the Ethernet buffer to the TCP packet. As explained earlier in Chapter 2.2.4, the TCP packet is sent to the TCP layer of the TCP/IP model and the transport layer carries the information across to the computer requesting the data over the physical layer, which is the Ethernet cable connecting the computer and the microcontroller. The Microchip TCP/IP stack model is analogous to the TCP/IP model and this is shown in Figure 5.8. The Host-to-Network corresponds to the combination of Physical layer and Network layer of the model described in Chapter 2.2.4. For this application, there is no connection to the internet also there is no requirement of any other TCP services in the application layer like Simple Network Time Protocol (SNTP), Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Dynamic Host Configuration Protocol (DHCP). Hence this application is only concerning the transport layer and the Host-to-Network layer [39].

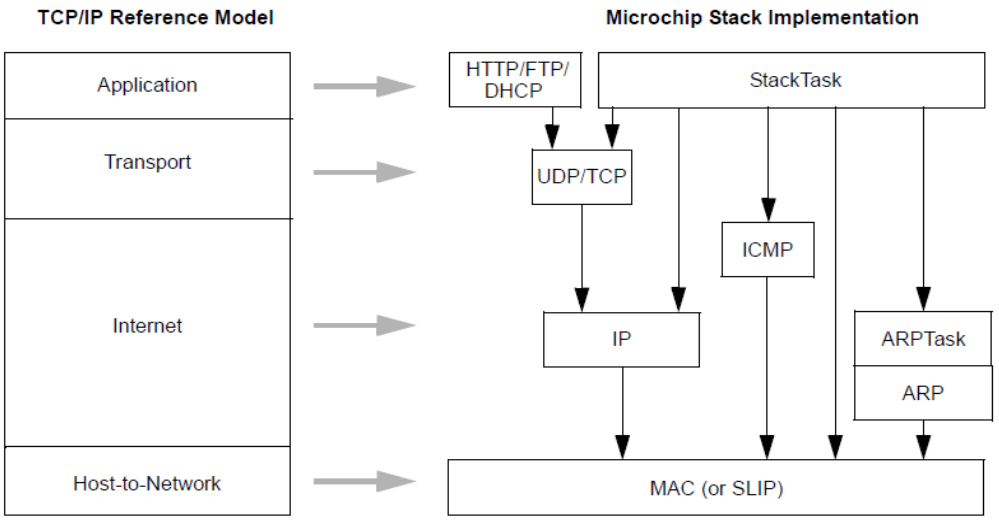

Figure 5.8 Microchip TCP/IP stack and TCP/IP reference model
(Taken directly from [39])

The other layers required to successfully complete a TCP communication from the microcontroller to the computer are auto-generated by the TCP/IP stack which has been pre-programmed to the microcontroller board [38][39]. The code written for this application only pertains to converting analog to digital, transporting the data to the circular Ethernet buffer and finally placing the data packets in the TCP layer of the stack.

5.3.5 Analog to digital converter (A/D)

This module performs the function of conversion of an analog signal to a 10 bit digital number. The module has five registers:

- A/D result register High byte (ADRESH)
- A/D result register Low byte (ADRESL)
- A/D control register 0 (ADCON0)
- A/D control register 1 (ADCON1)
- A/D control register 2 (ADCON2)

ADCON0 controls the operation of the A/D module like calibration, choosing an input channel, turning the A/D module ON and starting an A/D conversion. ADCON1 configures the function of the port pins as either analog or digital and also selects the reference voltage for the A/D conversion. ADCON2 register configures the A/D clock source, programmed acquisition time and result justification [35].

The analog reference voltage is software selectable to either the positive and negative supply voltage of the device or the voltage level on input pins. Since one of those input pin is also an A/D input channel and it has been used as the channel for data acquisition in this project, microcontroller's internal supply voltage is the reference for A/D conversion. Before the analog signal is converted to digital, an acquisition time must elapse after the A/D conversion is set to GO. The steps in performing an A/D conversion are [35]:

1. Configure the A/D module
    a. Select A/D input channel using ADCON0

b. Select A/D acquisition time using ADCON2

c. Selecting A/D conversion clock using ADCON2

d. Turn on the A/D module using ADCON0

2. Configure A/D interrupt

a. Clear ADIF bit, which is the A/D interrupt flag bit.

b. Set ADIE bit to enable the A/D interrupt.

c. Set GIE bit to enable all global interrupts.

3. Wait the required acquisition time

4. Start the A/D conversion by setting the GO/DONE bit in ADCON0

5. Wait for A/D conversion to complete, by either:

a. Polling for the GO/DONE bit to be cleared, or

b. Waiting for the A/D interrupt

6. Read A/D result registers (ADRESH:ADRESL), clear the bit ADIF if ADIE is enabled

7. The next conversion begins from step 2.

The A/D conversion time per bit is called TAD and a minimum of 2TAD is required before the next acquisition starts after completing an A/D conversion. For an A/D acquisition to meet its specified speed and accuracy, the charging capacitor in its internal circuit should be fully charged to the input channel voltage level [35][40]. The source impedance is the main factor which will affect the charging characteristics of this capacitor. So the recommended value for this source impedance is a maximum of 2.5 kohm. In this application, this is kept well inside this limit, at 0.3 ohm. Hence a faster acquisition time is ensured. A typical acquisition time for half LSB error of A/D conversion, which the maximum error to obtain high resolution conversion, at 3 V is 2.4 us [35][40].

The most important choice is the A/D conversion clock. Based on this, the wait time between acquisitions changes, also the total time taken to convert from analog to digital samples is

improved. The A/D conversion takes 11 TAD for a 10 bit conversion. The possible options for TAD and their corresponding maximum device frequency are shown in Table 5.2.

Table 5-1 TAD versus device operating frequencies [35]

AD Clock Source	Maximum Device Frequency (MHz)
2 TOSC	2.68
4TOSC	5.71
8TOSC	11.43
16TOSC	22.86
32TOSC	41.67
64TOSC	41.67
RC	1

For this application, the device frequency is set as 41.67 MHz and hence either 32 TOSC or 64TOSC should the value of TAD. The minimum TAD permitted for the microcontroller is 0.7 us, and so it cannot be set lower than this. Moreover, smaller the value of TAD, faster the A/D process [35][40]. At 41.67 MHz, the value of TOSC, which is the inverse of 41.67MHz, is 23.99 ns. At 32 TOSC, the TAD would be 0.77 us and at 64 TOSC, TAD would be 1.54 us. Since a smaller TAD is required, 32TOSC is made the required choice. For an A/D conversion, taking 11 TAD as its conversion time, for a 32 TOSC choice with 41.67 MHz oscillator, this will be 8.45 us [40]. A switching time from convert mode back to sample mode is on the next instruction cycle, which is 96 ns, for the 41.67 MHz clock. This time is very small and usually neglected in time calculations. The discharge time of the capacitor to get it ready for the next acquisition is 0.2 us. Hence for the total A/D process, an acquisition time of 2.4 us, a wait time of 2TAD equal to 1.54us, conversion time of 8.45 us and add to it the discharge time of the capacitor as 0.2 us are elapsed, which is 12.59 us [40].

Inside one second this will yield approximately 79,430 digital samples. This will certainly not be possible, since the microcontroller is not just performing A/D conversion, but also sends out the data through the Ethernet [40]. Every time a data packet is sent to the computer, there is a sequence of steps, as discussed in Chapter 5.3.4. The TCP stack is heavier in structure and has a much higher latency than this A/D conversion time [39]. Latency is defined as time difference between the time when the data is available for transmission and the time when the data has actually reached the remote computer. A higher latency would require a higher buffer size and hence the 8Kbyte Ethernet buffer is set mostly for transmission[34]. In the case of A/D conversion, 79.430 samples is its maximum capability but it is not number of generated samples. The actual generated samples will be much lesser than this, because every time analog voltage comes in, the microcontroller's processor needs to break its service from stack operations and go serve the A/D converter. Moreover, the addition of GPS pulses and its corresponding interrupts also reduce the number of conversions, since the processor of PIC18F's cannot handle so many functions in multitasking in a faster way, similar to PIC32's [41]. The reason to choose a PIC18F is that this is a basic board and it would be comparatively easier to showcase a proof of concept of the data acquisition system than in a PIC32 microcontroller. Other modes of communication such as serial communication were not considered since their speed at 192Kbps is much slower compared to Ethernet speed of 10/100 Mbps [42].

5.4 PICDEM.net2 development board

This development board enables the understanding of Ethernet and internet related solutions of the PIC18F microcontrollers. Using the preprogrammed TCP/IP stack on its on-board microcontroller and the availability of many demo applications, various generic solutions can be explored. The board contains the PIC18F97J60 microcontroller, a separate ENC28J60 stand-alone Ethernet controller to provide Ethernet connectivity to microcontroller-based applications using a standard Serial Peripheral Interface (SPI) [43]. Also the board contains a serial port RS 232 DB9 connector, and a RJ 45 10 Mbps Ethernet connector. The I/O ports of the microcontroller of ports

A,B, C, D, and E are available on-board. These ports can also be used to connect to any extended I/O board, which has more I/O ports. The same ports can be connected to a Microchip PICtail daughter board series.

The main purpose of using this board is to utilize the functionalities offered by the PIC18F97J60. Since the PICDEM.net2 board has an on-board microcontroller along with the Ethernet controller and the Ethernet connectivity option, this board is the best choice. If it was attempted to develop a custom-board for this purpose, it would be time consuming and due to complexity of the TCP stack and adding to it the multiple functionalities offered by PIC18F97J60 it would become a separate project of its own and this entire project would take more time than the deliverable deadline [44].

The board needs to be setup to start communicating with the computer through the on-board Ethernet connectivity. This requires that the IP address of the board, gateway address and the subnet mask match with the computer's configuration [17]. The board is assigned an IP through HyperTerminal, a telnet program [43]. The microcontroller is connected to the computer through the serial cable on one terminal and also its Ethernet terminal is connected to the PC through a crossover cable. The board is preprogrammed to acquire an IP address from the computer, when it connects to the computer. When the serial configuration menu is enabled the HyperTerminal screen looks as shown in Figure 5.9.

From this menu, the IP address, gateway address and subnet mask can be set to the required value. Moreover, DHCP is disabled. This is because, if the DHCP is enabled, every time the microcontroller connects to the computer, or if the microcontroller is rest or if the computer is restarted it will acquire a new IP address [45]. This will cause confusion and in some cases loss of Ethernet data packets proving to be disastrous to the application. Once this configuration is done, the TCP/IP stack is updated to the latest one from v3.75 to v5.20 from the Microchip technology website. Now the board is ready to be utilized for this application as is shown in Figure 5.10.

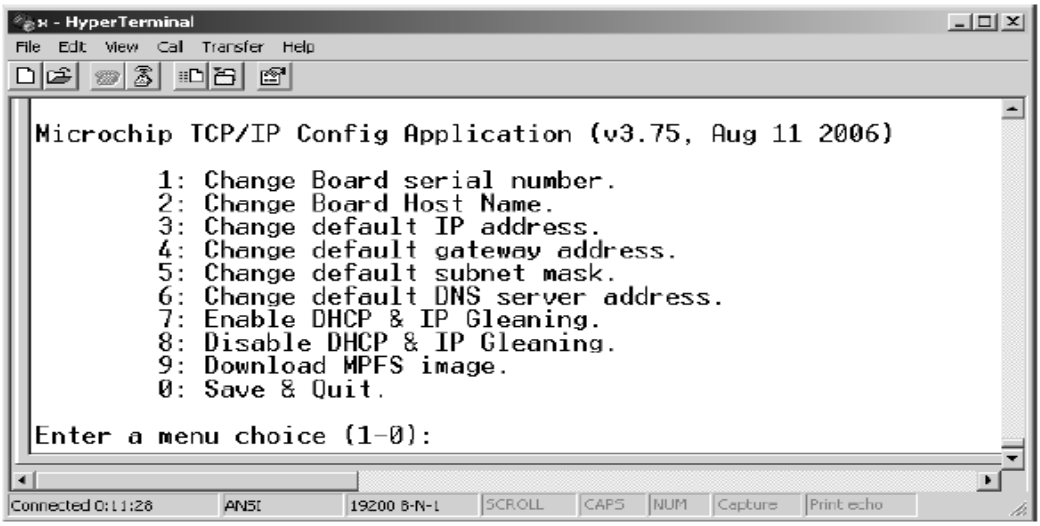

Figure 5.9 Serial configuration Menu
(Taken directly from [26])

Microchip Lab Integrated Development Environment (MPLAB IDE) is the development software on which the code for this microcontroller has been written. [46] One of the many demo applications as part of the Microchip library, available on Microchip website is the TCPIP DEMO APP. This file was used for this project and all required additional code and the modifications to the existing ones were made into this file using the MPLAB IDE.

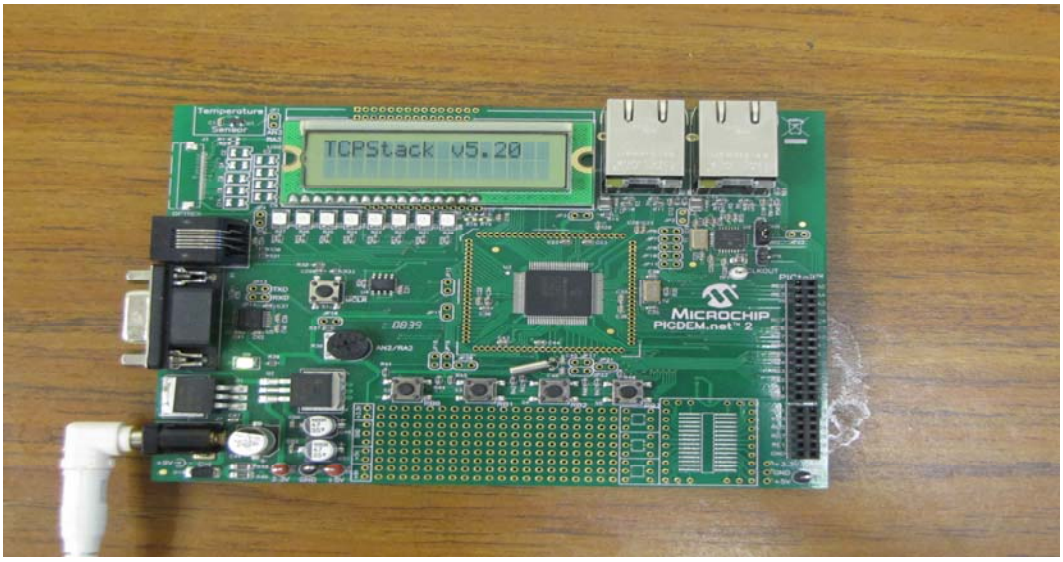

Figure 5.10 Initial setup of PICDEM.net2 board

The changes and the additions to the code have been discussed in Chapter 5.6. Based on the program, the obtained sample speed was very slow compared to the requirement of 1000 samples per second. Since at 1000 samples per second it is one sample every ms, the on-board Ethernet controller is not able to support this high speed, due to the stack applications and its related interrupts. Hence a PICtail daughter board operating at its highest speed of 100 Mbps is connected to the PICDEM.net2 board through its J5 PICtail connector [47]. The connection is shown in Figure 5.11.

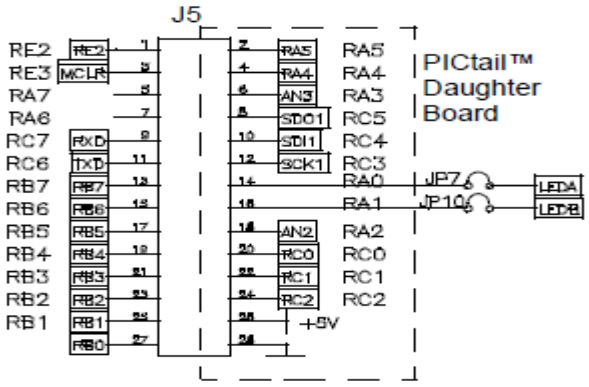

Figure 5.11 Connecting PICtail daughter board to PICDEM.net2 board
(Taken directly from [43])

It can be observed that the pins of PICtail daughter board occupy the pin position RA3/AN3 which is the A/D analog input channel. Hence the pin occupying that position on PICtail daughter board has been removed. Similarly the RB1 pin of the PICtail daughter board is removed to accommodate the incoming pulses from the GPS [43][47]. More on this GPS pulse is discussed in Chapter 5.5.

The main attraction of the PICtail daughter board is the presence of a 100 Mbps controller compared to the 10 Mbps Ethernet controller on-board of the PICDEM.net2 development board [47]. Based on the discussion about the Ethernet module in Chapter 5.3.4, the data from the EDATA register of the microcontroller is transferred to the same Ethernet buffer. Till here the process is same for both the PICDEM.net2 board and the PICtail daughter board. In the next step, the data is routed out through the PICtail daughter board, using its on-board 100 Mbps controller.

This enables faster transmission of data from the microcontroller to the computer. . Since it is connected to the PICDEM.net2 board, it operates on the oscillator of the PICDEM.net2 board. The reason why only a PICtail daughter board cannot be used is because it is only an extension board. This means, there is no memory, no buffer space, no A/D converter, no other peripheral connectivity like serial port. The only objective of the PICtail board is to improve the performance of its mother PIC board, and hence it is name the daughter board [47].

5.5 SEL 2407- Satellite Synchronized Clock

The shortcoming of conventional pilot protection is the comparison of two waveforms that are not synchronized in time. This may happen due to the delay in the communication modules or it can also happen at the data acquisition modules [15]. For example, in a fiber optic communication, the transmitter and receiver of each acquisition module cannot be exactly same due to small deviations in their crystal frequency [48]. Minor deviations will result in a minor delay in data transmission and reception. When such inputs are compared to evaluate a trip decision, mistripping is a definite possibility and this is the reason data acquisition systems are architecture-dependent. To solve this problem, in this data acquisition system, comprising of three such PICDEM.net 2 boards each with one PICtail board, a Global Positioning Satellite (GPS) receiver, (SEL) 2407, is connected to the PICDEM.net2 board. This GPS receiver acquires satellite time from at least 3 satellites, synchronizes continuously while maintaining a time accuracy of 500 ns peak based IRIG output [49]. The GPS is also capable of providing IRIG-B based output at 1 pulse per second (PPS) or 1000 pulses per second (1KPPS). A picture of the GPS receiver is shown in Figure 5.12

The function of this clock in this application is to provide synchronized pulses at 1KPPS to each microcontroller so that the A/D conversion on each microcontroller can be synchronized. This will result in the digital output value from each PICDE.net2 board to be synchronized in time.

Figure 5.12 SEL 2407 satellite synchronized clock
(Taken directly from [49])

From the figure, ENABLED indicates the clock is ON and if its LED is green it means all tests are passing and the clock is running in continuous mode. If the LED is yellow, it indicates that all tests are passing but the clock is in Force-Time-Quality mode. In this mode, the user can set the time quality, compared to the usual case of clock acquiring time quality based on the satellite signal [49]. This mode is useful when there is no or less satellite signal, and so the time has to be manually set for the GPS receiver. In this case, the time quality is uncontrollable, since the receiver is not synchronizing to sufficient satellites. Hence if a time quality is enforced, it ensures good accuracy for a longer period of time. The experiment using the data acquisition system was conducted in Power Electronics –I laboratory. This lab has no GPS satellite signals, hence a time was manually set using the TIM command and the time quality was forced using the FTQ command [49]. The SATELLITE LOCK LED is green when the clock is tracking three or more satellites else it is blank. The HOLDOVER LED is an indication of the time quality, since for this application the time quality is manually set, it blinks yellow.

Due to the lack of availability of three receiver clocks, only one clock has been used in this project. This should not create a difference to the final results. A roof-top test was conducted at the roof of the Engineering Research Center building with one GPS receiver clock and satellite synchronizing signals were obtained, thus confirming the assumption of GPS signals availability at the FREEDM installation site. Since the boards and their corresponding GPS will be placed outside near the FREEDM distribution line, there would be sufficient satellite signal to synchronize all three GPS receiver clocks. Hence the pulse output from the three clocks would be syn-

chronized. In the current setup, one GPS receiver clock provides synchronizing pulses to every PICDEM.net2 board. The GPS configuration settings were made using the AcSELerator Quickset, an IDE from SEL to configure all SEL devices [50]. The settings changed and the commands issued are shown in Appendix D. The pulse output from the GPS clock is transmitted through the C962 cable from SEL. This is a RG-58 coaxial cable with BNC connectors on one side and tinned wires on the other side. The tinned wire's center cable is connected to the RB1 interrupt pin of the PCIDEM.net2 board [51]. This way, the pulses are input as interrupts to the microcontroller. The shield of the tinned wire is connected to the ground terminal of PICDEM.net2 board. Based on this connection, and the assumption that the interrupt on RB1is enabled, an interrupt is generated for every GPS pulse, which means an interrupt every 1 ms. Using this interrupt, the TCPIP code can be programmed to start A/D conversion in a synchronized manner. The details of the coding are described in Chapter 5.6.

5.6 TCPIP code details for the project

The exact code programmed into PICDEM.net2 board is shown in the Appendix B, C. With the addition of the PICtail daughter board, the base application code from the Microchip TCPIP library is the TCPIP ENCX24J600 Demo APP. The main files used for the project are the MainDemo.c, TCPIPConfig.h, TCP.c, and TCPPerformanceTest.c. Various functions inside these files are linked to other sub files which are related to the Stack application and stack tasks that run continuously in a cooperative multitasking to execute TCP related functions, like checking for sockets, checking for packet requests, constantly reading the buffer [refer to Appendix B, C].

In the file TCPIPConfig.h the IP address, the gateway address and the subnet mask to which the PICDEM.net2 should communicate are set. Moreover functions like DHCP, SNTP, Address Resolution Protocol (ARP) and other function which are executed in the application layer of the TCP model are disabled, thus restricting the application to only Transport layer [39]. Moreover, inside the transport layer, User Datagram Protocol (UDP) another transport layer protocol like the TCP is also disabled [52]. Since this application is only related to the TCP layer, disabling

other layers renders sockets due to other protocols ineffective, thus increasing the speed at which a new TCP socket is made available to the microcontroller [39]. Only the Internet Control Message Protocol (ICMP) protocol is enabled both as the server and the client. This is to ensure the computer is able to ping the board to check for sockets and the board will send messages to the computer saying no socket is available, in case no socket could be formed between the computer and the microcontroller. This protocol structure is only responsible for sending query messages and no data can be sent using ICMP [53]. The Announce function in the file is used to send dummy messages to the computer in the event of loss of sockets and still the connection needs to be maintained between the PICDEM.net2 and the computer.

The TCP.C file is used to perform general stack functions such as initializing the TCP module, creates a socket, allocates socket memory based on the socket type as receiving or transmitting type socket, checks for successful TCP connection, estimates the number of bytes of data sent and received [39]. All these functions and their structures are defined in TCP.c and called in MainDemo.c file.

TCPPerformanceTest.c is the function which performs the important TCP functions of this application such as requesting a socket, framing the Ethernet packet, and sending the data out through the Ethernet. The functions corresponding to receiving data packets are disabled since this project is only aimed at sending data packets out and not receiving anything. So, the TCP receive buffer is made smaller in TCP.c and the receive functions are disabled in TCPPerformanceTest.c. This file assigns a port to the transmit function, and also tests the performance of the TCP transmit module. An increase in the TCP transmit-buffer size in TCP.c enables faster processing of the data inside the microcontroller [38]. Moreover, the round trip time of the data packets or the latency of the TCP communication could be measured using this file. The Ethernet controller is capable of transmitting data packets faster than the internet in the event of a table top implementation [54]. Currently the experimental setup is a table top and hence the latency measured will not give the right picture of the actual latency that will be observed at the field. Hence only the size of the

transmit-buffer is increased and the TCP module is tested. Parts of this test program are also used in the MainDemo.c file to execute TCP related functions.

MainDemo.c file is the most important file in the whole project. All primary coding for this project is done inside this file. This file is equivalent to the brain of the whole project. The calls to different TCP related tasks, routing the operation to TCP.c file and to change the configuration of the stack, the control is routed to the TCPIPConfig.h file. Every time these functions perform the required operation, the processor returns back to the MainDemo.c file to continue performing the next function.

Inside the MainDemo.c the variables that are required for all purposes for the entire program are declared as global variables. Functions such as initializing the board, configuring the application related to stack and general purpose input/ output functions are called initially before the actual process starts. These processes are executed first to get the board ready for the application [39]. A separate function for the TCP process pertinent to only transmitting the A/D value through the Ethernet output is defined. This takes care of the TCP packet transmission part. Functions and commands inside the ISR service the interrupt that is generated due to the GPS pulses. In this application, to synchronize the A/D conversion, every GPS pulse starts an A/D conversion, implying every GPS pulse results in one A/D converted data. The program in the ISR starts the A/D conversion after receiving every GPS pulse, and finally converts the data into hexadecimal format.. This converted data is stored in the buffer, which was declared globally at the beginning of the MainDemo.c file. Since this buffer is global, it can be accessed by the TCP packet handling function described earlier [38]. Thus the A/D converted data is ready to be transmitted to the computer. When this TCP function checks for a socket and confirms the available buffer space in the microcontroller through the functions in TCP.c, this hexadecimal data is transmitted to the computer.

Next is the main function in the MainDemo.c is where the processor actually starts. This is where many startup function such as LCD initialize, TCP initialize from TCP.c, Stack initialize

also from TCP.c are performed initially before the recurring processes start running. For the purpose of fast processing and reducing the delay from the stack operations, functions StackTask and StackApplications are called once outside the cooperative multitasking "while" loop. These functions are defined in StackTsk.c file. They execute primary functions to eliminate functions such as DHCP, ARP which were disabled in TCPIPConfig.h and also start the search for any received socket [55]. They do not send any request for socket though. The execution of StackTask and StackApplications functions is mandatory to perform any TCP related function. Hence these functions are also called inside an indefinite while loop, in the event a TCP action is required. A TCP action is only required when there is data in the buffer so that control can transfer TCP packet processing function. Hence these two functions are called only if the A/D output buffer is having some data.

It was observed that some time gets wasted in waiting to transfer the contents from the buffer to the TCP process module. If that time is saved, more number of samples can be obtained in a cycle. When there was one A/D buffer, because of the extra time taken to process one data packet through the A/D and then onto its buffer and then to the TCP buffer, a GPS pulse was missed intermediately and hence not every GPS pulse could generate an A/D value which is a sensitive problem to the protection algorithm which needs every sample point. Hence, two buffers were created such that when one buffer gets filled, and that data is getting transferred to the Ethernet buffer, the other buffer can start filling up with the A/D output value and these two buffers will get filled alternately and sending data out to the same Ethernet buffer. Since the buffer size of the Ethernet was made larger for transmit purpose, the Ethernet buffer could handle the high speed of data transaction. The concept is shown in Figure 5.13 and 5.14. In 5.13 buffer 1 accepts data from A/D result register and buffer 2 transfers its data to the TCP buffer In 5.14 the roles reverse.

The main logic of the data processing and transmission is shown in Figure 5.15.It explain the concept starting from the reception of the GPS pulse till the data is ready to be transmitted.

The only decision made is about the temporary buffer space in the PIC18F97J60 memory. If it is not empty, that indicates there is data ready to be transmitted.

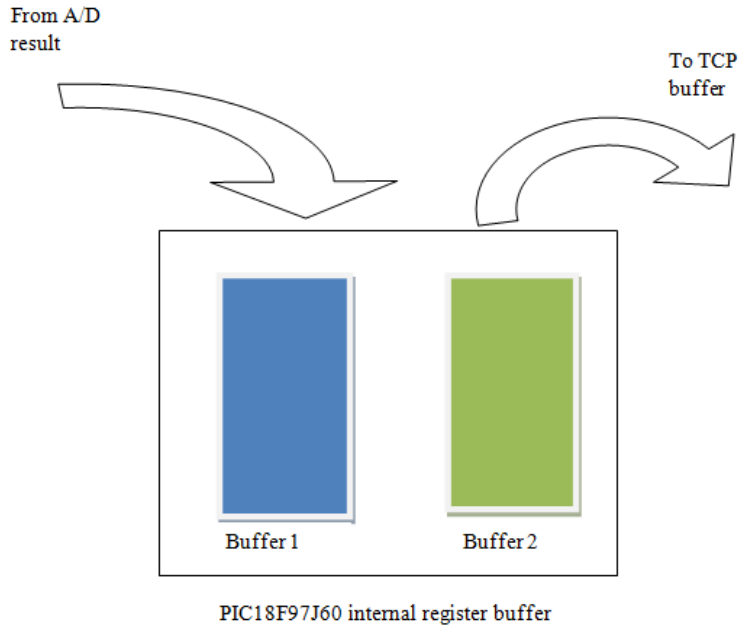

Figure 5.13 A/D output data to buffer 1 and TCP buffer data from buffer 2

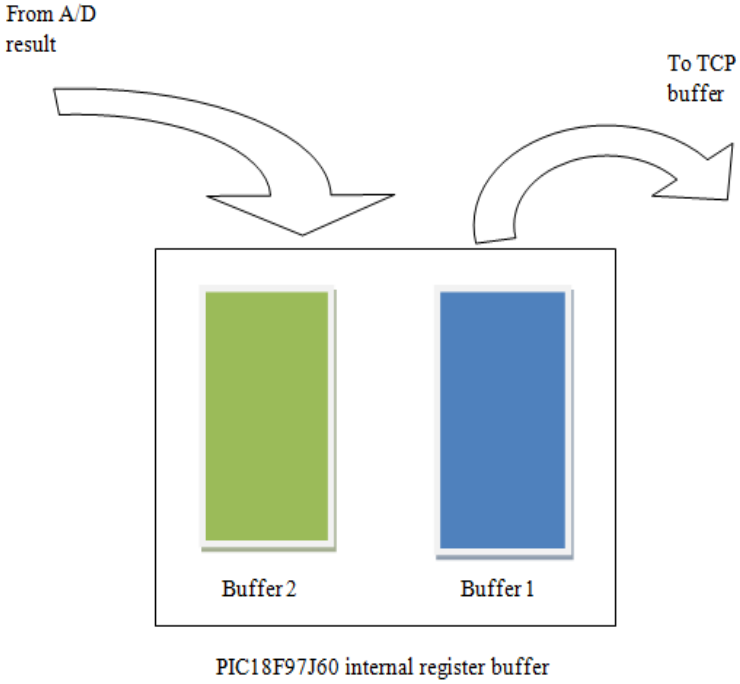

Figure 5.14 A/D output data to buffer 2 and TCP buffer data from buffer 1

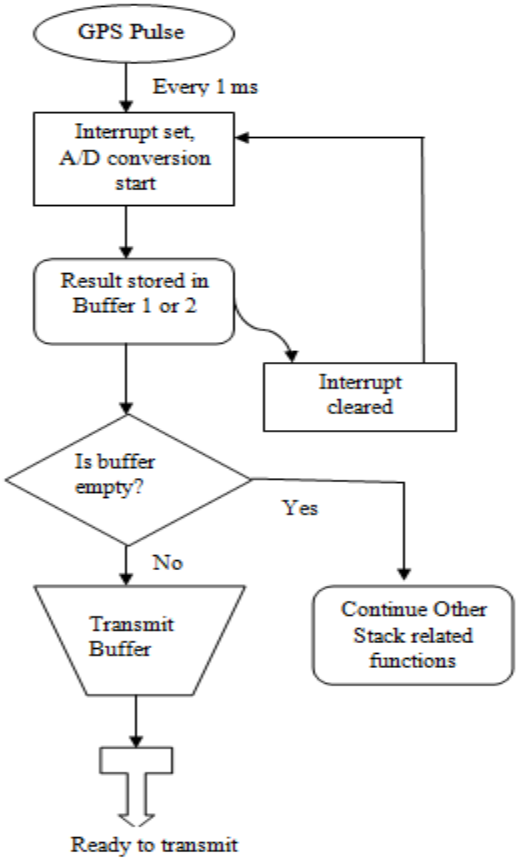

Figure 5.15 Programming logic to acquire synchronized A/D output on the computer

5.7 Biasing circuit

For the PICDEM.net2 board, the minimum input voltage is 0 V and so the negative voltages will not be converted to digital and rather will be neglected [35]. Hence it was required to create a biasing circuit to bias the input voltage to 0-3.3V range. The component ratings for the design were obtained using the RESISTOR-CALC program available on Texas Instruments website [56]. The analog voltage at the secondary of one of the current transformers, across a 0.3 ohm resistor is found as 1.25 V peak. So a biasing circuit was developed to transform from -1.25V to +1.25V to a 0-3.3 V peak and is shown in Figure 5.16. The addition of a DC battery and the resistor ratios in the circuit enables the effective functioning of this biasing circuit.

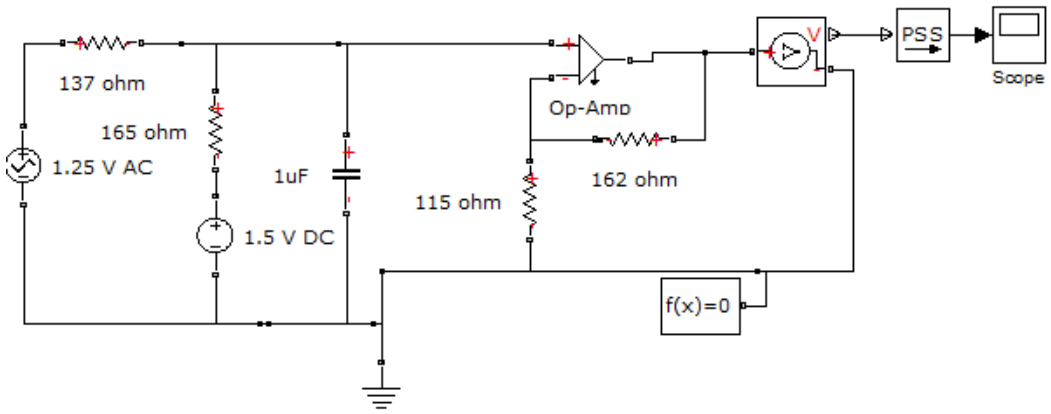

Figure 5.16 Biasing circuit converting to 0-3.3 V peak

The input voltage waveform is shown in the Figure 5.17 and the output biased voltage waveform is shown in Figure 5.18.

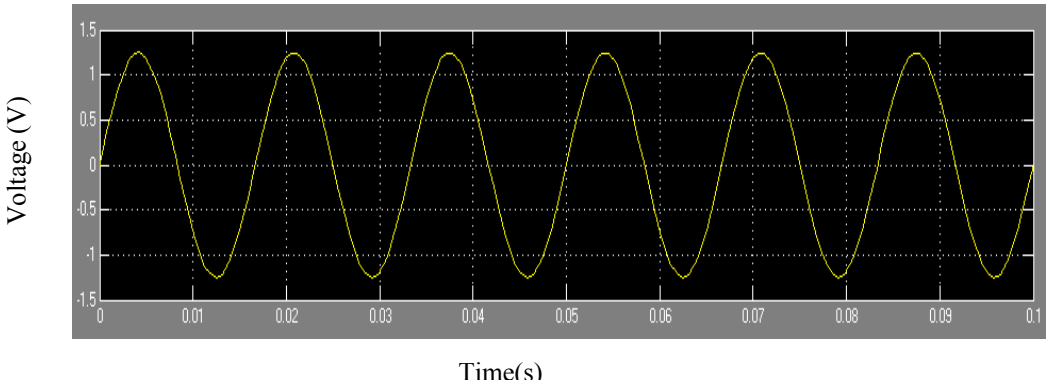

Figure 5.17 Input voltage at +1.25 peak

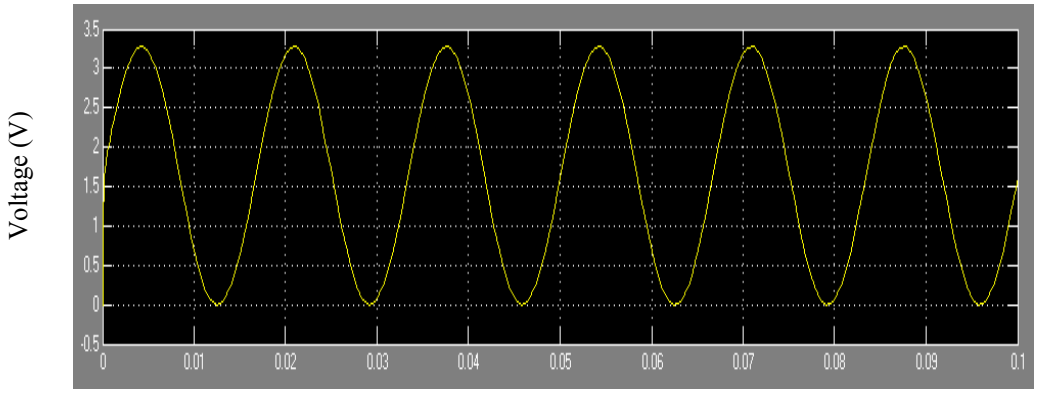

Figure 5.18 Output voltage at 0-3.3V peak

The current setup does not have this bias circuit, instead biasing batteries have been added. Though that is not the exact way of accomplishing the biasing, the battery provides good consistent biasing which is observed in waveforms shown in Chapter 6, when three identical batteries are chosen. The circuit realization of the biasing circuit has been considered as a future work because of the time constraint associated with the deliverable date of the project.

5.8 Hardware prototype of the testbed for protection implementation

This hardware prototype is a miniaturized version of a zone in the FREEDM loop [1]. Two sources feed the resistive load through a distribution line realized using resistor and inductor in series. Three current transformers measure the current through the two source branches and the current through the load. A 0.3 ohm resistor is connected across the secondary terminals of each current transformer and the voltage across the resistor is the input to each of the PICDEM.net2 board. One SEL 2407 satellite-synchronized clock provides continuous pulses to all three boards and this is input to pin RB1 on-board to generate an interrupt. Once the transmit buffer has data in it, this data is transmitted to the computer through the Ethernet output from PICtail daughter board, that is routed to the computer using an Ethernet switch. On the computer the protection algorithm captures this data and executes both differential and overcurrent protection algorithm. This system is shown in Figures 5.19, 5.20, 5.21. A switch connected in parallel with the load, creates a short circuit to ground, which is a fault.

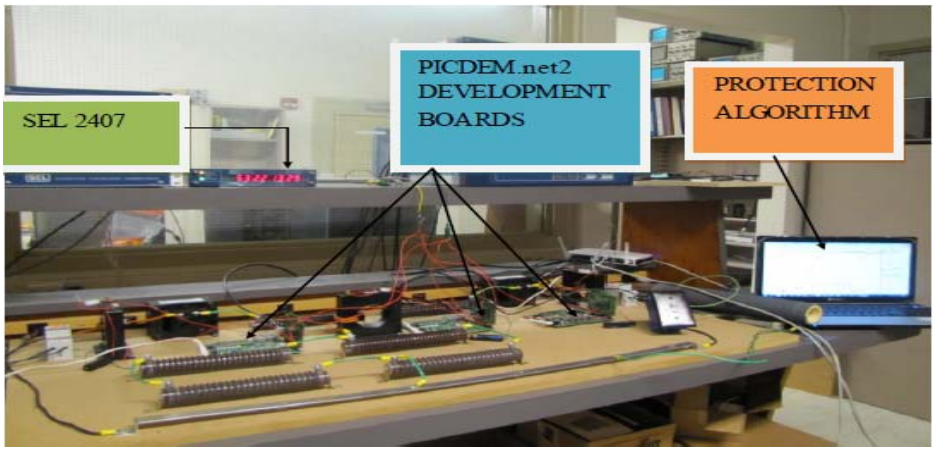

Figure 5.19 Hardware prototype of the data acquisition system

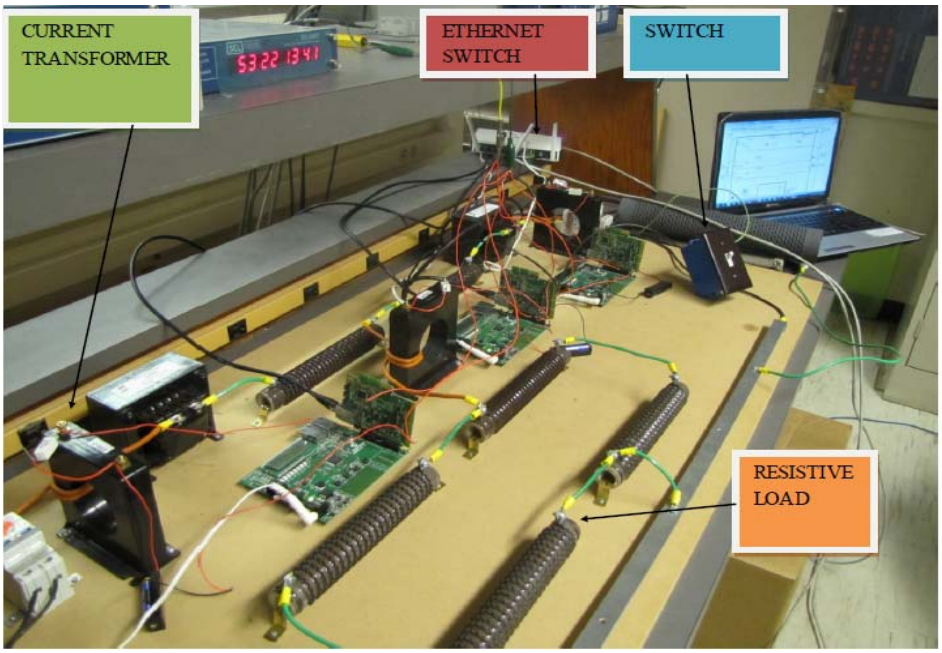

Figure 5.20 Three PICDEM.net2 boards with their connections

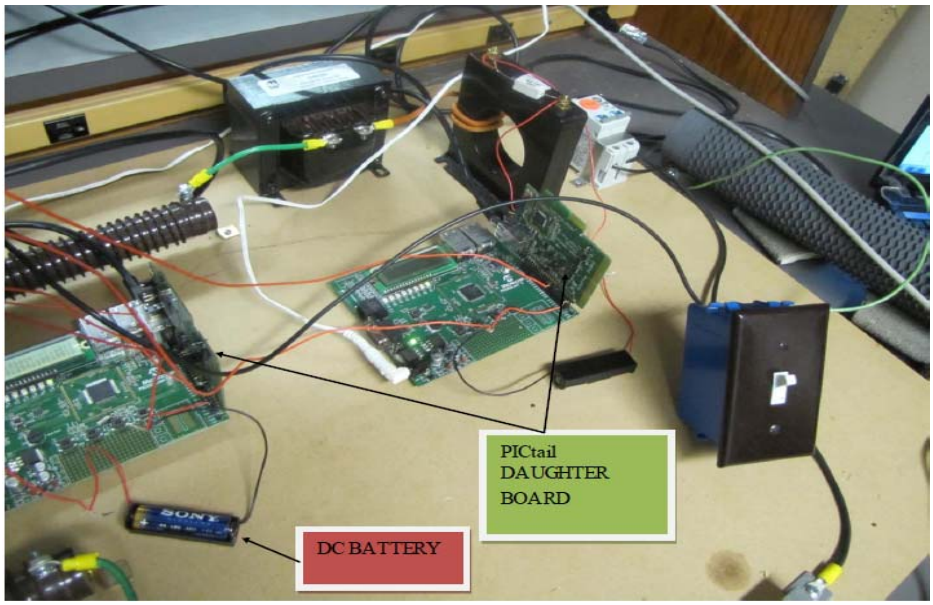

Figure 5.21 Prototype showing the PICtail daughter board

The hardware prototype developed and design methodology has been presented in this chapter. Tests were conducted over this prototype and the results are shown in Chapter 6.

6 RESULTS AND INFERENCES

6.1 Implementation Challenges

The challenges faced during the implementation of this designed data acquisition system and the steps undertaken to overcome those challenges are explained below.

1. The speed of data transmission from the PICDEM.net2 board was found to be slower because of various stack related functions described in Chapter 5. Tests concluded that the speed cannot be any faster and a new board PICtail daughter board was connected to this board and packet transmission was routed through the daughter board.
2. A separate buffer was created to store the result of A/D conversion. This buffer was called from the interrupt and also from the TCP module. The interrupt started the A/D conversion and the TCP module transmitted the values. Since only one buffer was used, it was alternately called by both the modules and the processor could not handle the speed, which led to the loss of data packets. Rather than 1000 samples per second, the speed was reduced to 400 samples per second. Hence this method was scrapped.
3. Since the A/D conversion was generating interrupts, sampling and converting at high speed resulted in very fast interrupts. The processor always stayed inside the ISR and no data transmission processes were executed. There could be no socket formed and so no data was transmitted to the computer. Hence the speed of the A/D conversion was reduced by changing the acquisition time, the TAD to match with the time taken by the processor in the Ethernet module.
4. A timer was programmed to start counting when the A/D starts. This way no time is wasted in setting and clearing the A/D interrupt flag bit. The overflow setting of the timer was fixed as the time taken by the A/D to complete a conversion. As calculated in Chapter 5.3.5, this setting was 12.59 us. So the timer was set to overflow every 12.59 us. Since the stack operations relating to transmitting the previous data packet are executed inter-

mittently during the A/D conversion, the time setting was not accurate and so interrupts were set after uneven A/D conversions which resulted in error values. Hence the timer strategy was not implemented.

5. Since a timer strategy was not accurate, an inbuilt function of the microcontroller, Enhanced Capture/ Compare/ PWM module (ECCP2), was used to trigger the A/D conversion. The ECCP2 function was linked to the timer internally, hence reducing the burden on the processor of the PIC18F97J60 [57]. When the timer's count reaches the value set in ECCP2 register, an A/D conversion is automatically started. This way, there is no external trigger to start an A/D converter, since external triggers such as internet or other source will have communication delay depending on the TCP network to further slow down the whole data acquisition process. Moreover, the commands to control the timer and reset it are not implemented. Since the interrupt routines are the most time-consuming function on a processor, this method saves that burden by not generating any interrupt [37]. If each board can be programmed to run the same program, the A/D conversion should take the same time in each of them, it results in synchronized A/D output. The shortcoming of this method was that the wait time between two conversions was too short and hence the A/D sampler could not produce pure sine waves as observed (only positive half cycle is shown) in Figure 6.1 using Labview 8.6 in which time represents the sample number The consequence is mis-representation of the AC sine waveform which will result in incorrect trip signal generation by the protection algorithm.

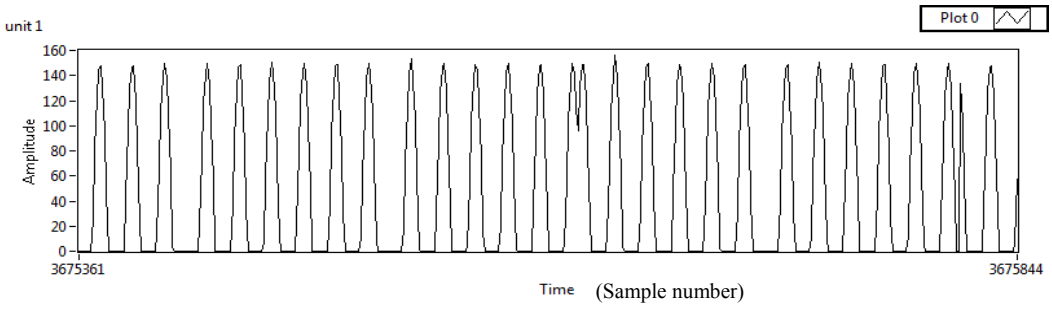

Figure 6.1 Incorrect A/D output using the ECCP2 trigger

6. It was required to confirm if the issue was because of the A/D sampler not being able to sample faster or the data communication method and this was tested using the serial RS-232 port available on-board PICDEM.net2. This was done using the commands in the file UART.c, which controls both the synchronous and asynchronous communication. A synchronous communication was implemented and the output was observed on Labview 8.6 as shown in Figure 6.2.

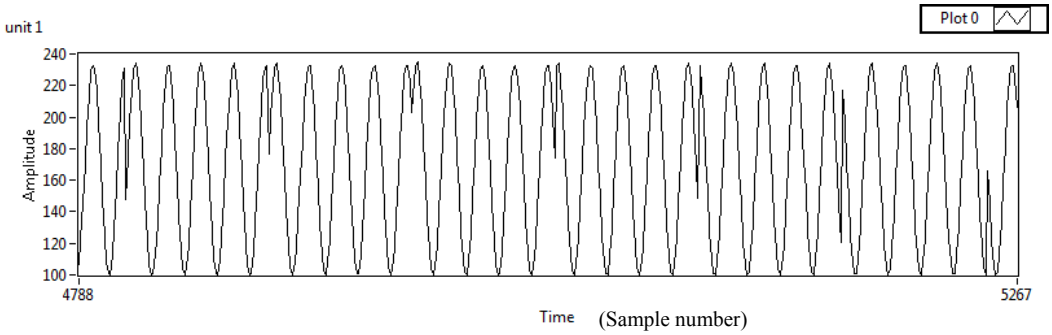

Figure 6.2 A/D output on Serial Port

Similar waveforms compared to those obtained through the Ethernet were observed over the serial port too. It was understood that the A/D sampler has issues or the processor was demanding samples more frequently than the sampler could provide it [58]. The same waveforms were obtained from all three boards, so the possibility of a defective board was not considered. The problem was narrowed down to the optimization of operations through programs.

7. When it was observed the high speed requirement was not met by the A/D sampler, the device frequency was reduced to 25 MHz to obtain proper sine waves. Though the sampling speed was reduced, this method slowed down the processor and all processes in the PIC18F97J60 were slowed. The A/D output got better but the total number of samples in a cycle reduced from 12 to 7. This resulted in poor waveform shape though lesser distortion compared to that in the case of a 41.67 MHz. This is shown in Figure 6.3 (only positive half cycle is shown). Since the processes inside the PIC were all slow, the throughput

of this board is reduced, which is not acceptable according to the design requirements slated in Chapter 5.1. Hence the device frequency was restored back to 41.67 MHz and the possible optimizations in stack operations were considered as the viable option.

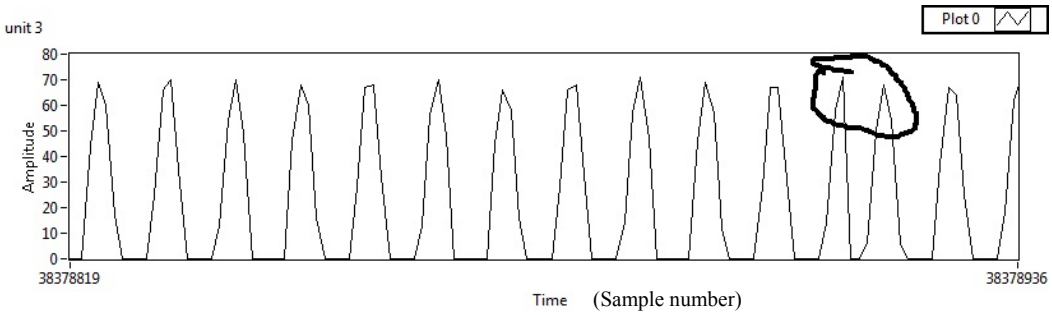

Figure 6.3 A/D output at 25 MHz clock frequency

8. The buffer size of the Ethernet buffer was reduced such that it gets filled faster and the TCP packet gets delivered faster to the computer, thus improving the speed of the whole system, though the waveform was not a perfect sine wave. In this case, since the buffer size was too small to match the speed of the A/D conversion, it leads to a choke at the buffer. When a choke happened, the TCP connection was lost because there was no data to transmit for a longer timer. This loss of communication is detrimental to the reliability of the data acquisition system and also the protection system. Hence the buffer was always maintained at its maximum.
9. Due to the increase in the number of global variables for various processes and the creation of a separate A/D buffer, the space available for storing global variables was not sufficient. Hence a new space was created in the microcontroller program memory using the "# pragma" command [58]. This is a directive to the compiler directing it to look at these commands before starting the program execution. This way the variables will remain global and be available throughout the program.
10. The TCP module was timed to understand the latency caused by each instruction inside it. A high state was latched to one pin when the processor enters the TCP module and it was latched low when the processor came out of the module. This pulse remained on for

more than the allowable time and every single instruction inside the module was timed using the same method and it was determined that the "uitoa" function, which converts the unsigned integer to ASCII strings was the main reason for the latency associated with the TCP module. Hence the PIC was reprogrammed to convert the data type to hexa-decimal which is faster compared to "uitoa" command. Moreover, the conversion program was changed too, which resulted in 14 samples per cycle.

11. When it was observed that no time synchronization could be obtained even with more samples in a cycle, a GPS was connected to provide pulses. As the three boards do not share the same crystal, so it is difficult for them to give out synchronized outputs, even though the program is the same. Synchronized inputs are the highest priority for the successful operation of the differential protection. Since there was no satellite signal inside the laboratory, the personal computer was synchronized to the internet time service "time.windows.com" and the exact time was obtained. This time was manually set to the GPS and also the time quality was forced to 1nanosecod (ns), so that the output from the GPS is accurate. If the time quality is not 1 ns, no pulse output will be generated from the GPS. This GPS provided 5 V pulses at 1KPPS, which created interrupts in the PIC18F97J60. The rest of the process has been explained in Chapter 5.6.

6.2 Labview Output

With the algorithm setup for the A/D converter, as explained in Chapter 5.6, the A/D output was streamlined and there were no error values or wrong sampling points obtained as in Figure 6.1, 6.2, 6.3. The waveforms have been corrected because of the change in the A/D conversion logic using the GPS. Since proper sine waves have been obtained, any issues with the A/D sampler have been ruled out. This confirms the suitability of this board to perform high quality and high speed A/D conversion. Now the number of samples in a cycle has been increased to 17. This is correctly verified by the GPS pulses logic. Since the pulses are received at 1000 samples per second, it implies a pulse is received every 1 ms. Hence in a 60 Hz cycle, there should be 17

pulses. This 17 pulses continues for many cycles and once becomes 16 in a cycle, which is understandable since the sine wave time period is 16.67 ms. So trying to accommodate 17 pulses in 16.67 ms cycle at 1 pulse every ms will yield 16 pulses occasionally in one cycle at a fixed frequency. The front panel of the Labview Virtual Instrument (VI) showing the observed values is shown in Figure 6.4.

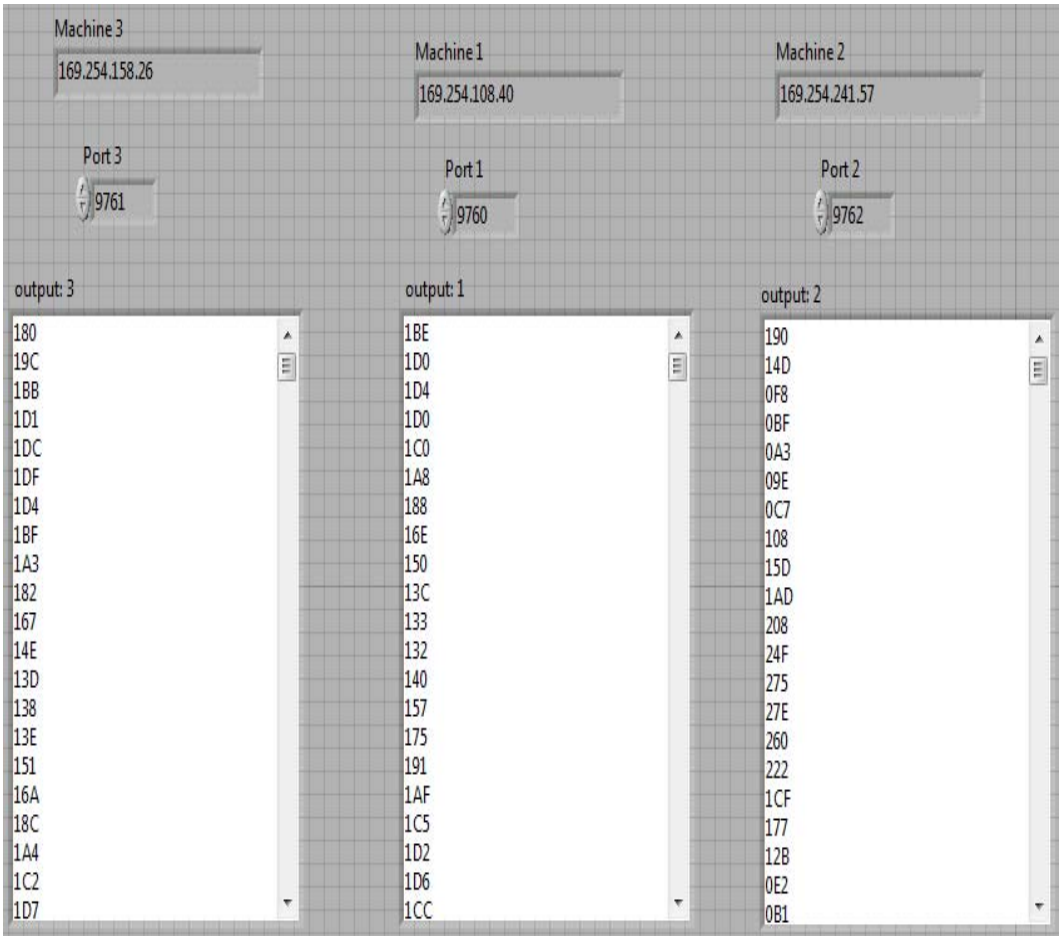

Figure 6.4 Snapshot of Labview vi front panel obtaining values from three PICs

It can be observed from the above Figure 6.4 that the values obtained are hexadecimal and hence the number of samples increased to 17 compared to the previous best of 14. Moreover, the A/D waveform obtained in this case is very smooth compared to every other waveform. The waveforms from each PICDEM.net2 corresponding to their output number are shown in Figures 6.5, 6.6, 6.7. It must be noted in Figure 6.4 that the values corresponding to “output 1” signify the

load current waveform, with their equivalent voltage waveforms. It is equivalent because there was only a resistor at the secondary terminals of the current transformer, the voltage across which is being represented here. “Output 2” and “Output 3” correspond to the source current waveforms.

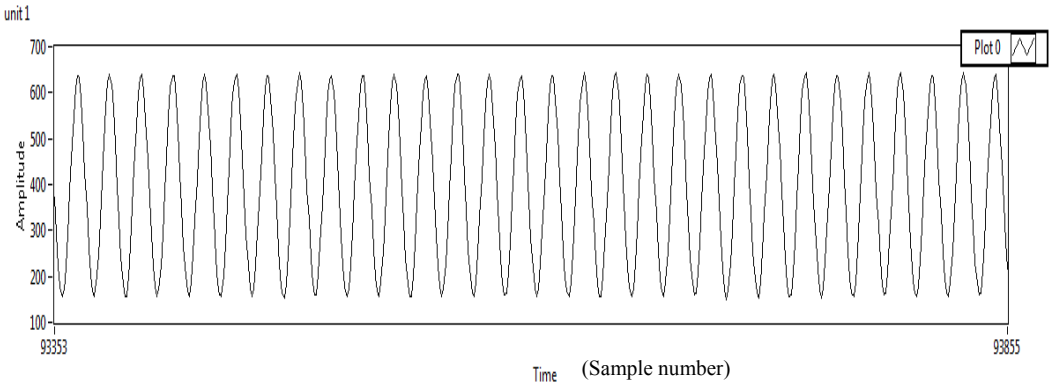

Figure 6.5 A/D output from unit 1(load)

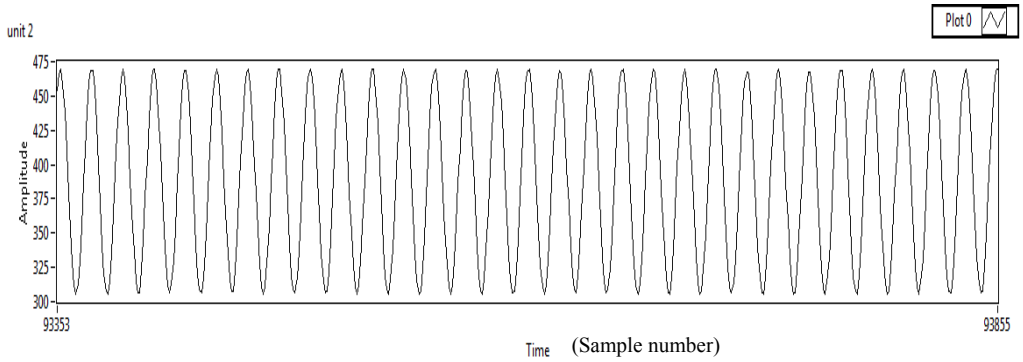

Figure 6.6: A/D output from unit 2 (source)

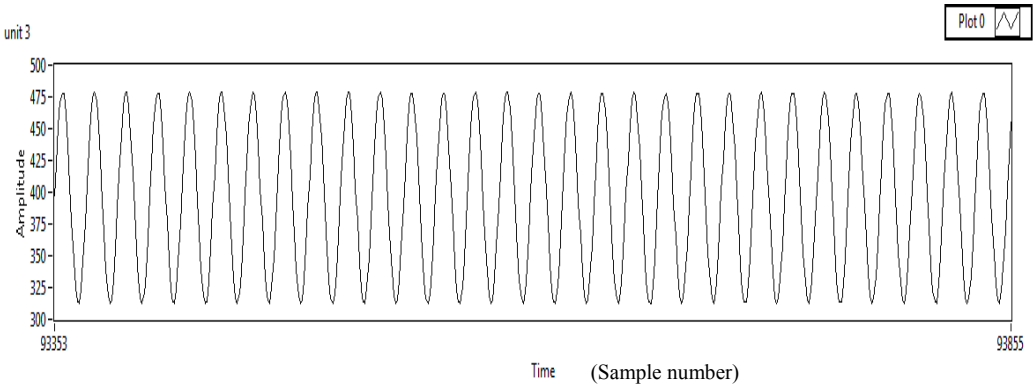

Figure 6.7: A/D output from unit 3 (source)

The plots in Figures 6.5, 6.6 and 6.7 start and stop at the same time instant respectively. The starting point is sample number 93353 and the end point is 93855. These numbers indicate the sample numbers. Hence it is inferred that the same sample points collected individually from each board are examined here. It is clearly seen that the waveforms are perfect sine waves without any distortions, or any error sample values. The only problem is the synchronization. The outputs from the three boards are not time synchronized. Outputs from units 2 and 3 differ from each by 1 ms, this difference is acceptable to the differential protection algorithm. But unit 1 varies by 8 ms from unit 3 and 6 ms from unit 2. This difference is not acceptable and the reasoning in provided in Chapter 6.4. These waveforms will lead to incorrect tripping signal generation by the differential protection algorithm. The differential unit waveform is shown in Figure 6.8.

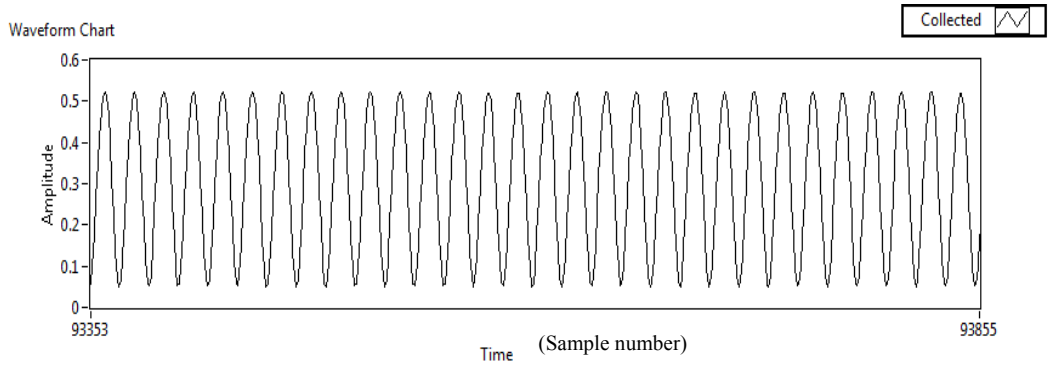

Figure 6.8: Differential unit output

The differential output obtained in Figure 6.8 is calculated using the difference ratio between the three outputs with a restraint value added, to prevent it from tripping for small differences. For any differential value greater than 0.3, this unit concludes the existence of a fault. In the event of no-fault, this differential unit calculation indicates a fault and this can be attributed to the time difference between the samples at the same instant. Hence this system cannot be used to provide data for this differential unit. However in systems where the differential units calculate the rms values of each signal to determine a fault or the peak-to-peak, instead of instantaneous values, this data acquisition system will work very effectively. Adding to it, the speed of data acquisition at 1 KHz, this system could generate a trip signal at the same speed as the commercial relay that

was tested initially in Chapter 4.1 for the FREEDM project. As was noted earlier, since the response time of that relay was slower than the requirement of the FREEDM project, that relay could not be used. Similarly, though this data acquisition system is capable of generating a trip signal at the same speed as the conventional relay, but it cannot be used for the FREEDM project because of the response time requirement of the project. The protection unit calculation using a different strategy is shown later.

The calculation of the overcurrent unit does not depend on the time synchronization issue. It only depends on the magnitude of the signal. Hence this protection algorithm works for the three PICDEM.net2 boards. The output of the overcurrent protection algorithm in the event of a fault generated using the switch and the DC trip signal generated is shown in Figure 6.9, 6.10, 6.11(the time scale indicates sample numbers).

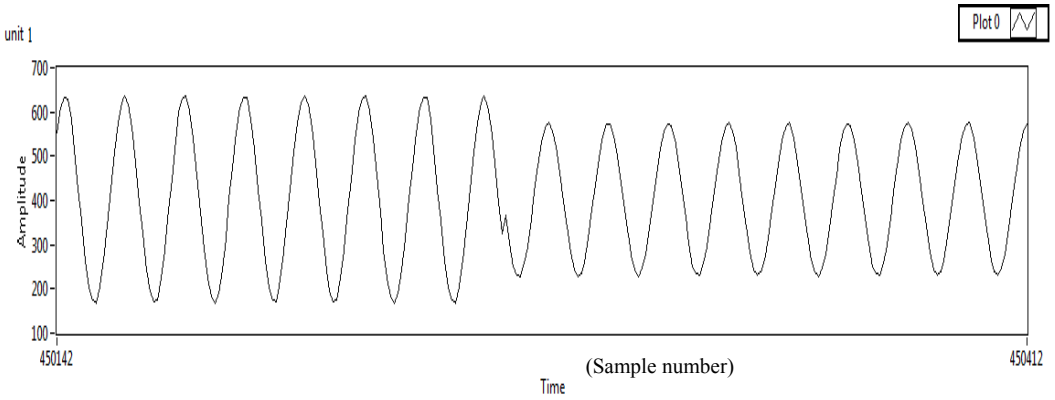

Figure 6.9 Overcurrent protection unit 1 output during a fault

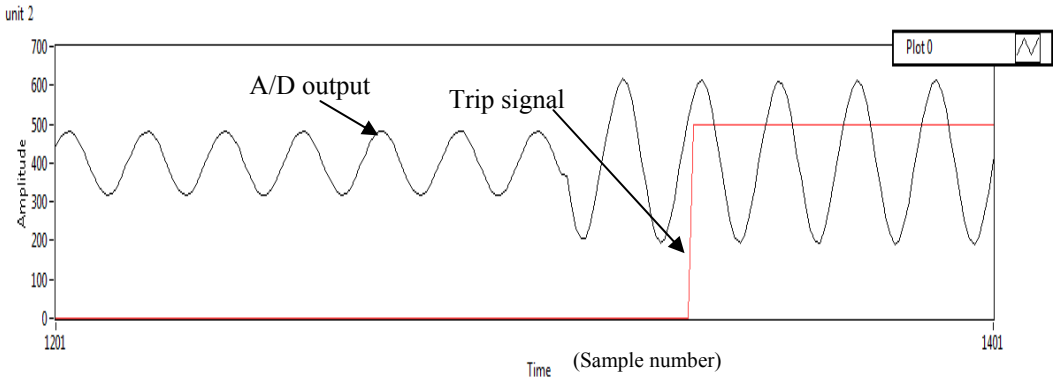

Figure 6.10: Overcurrent protection unit 2 output during a fault

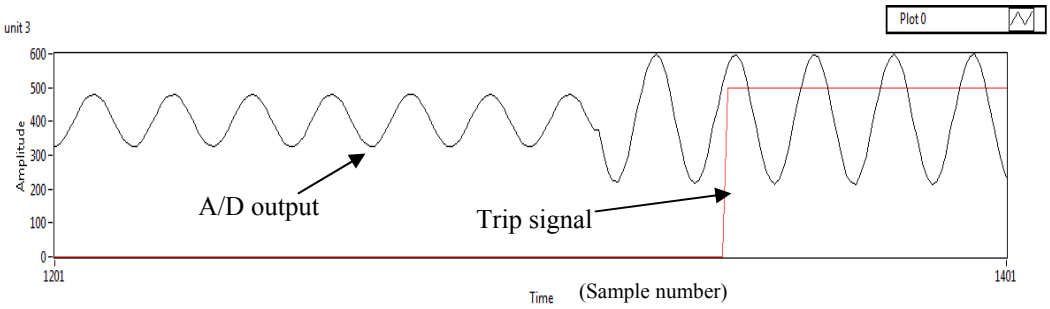

Figure 6.11: Overcurrent protection unit 3 output during a fault

It can be observed from Figure 6.10 and 6.11, a trip signal is generated just after 1.5 cycles after the fault initiation, which is still faster to the conventional relay tested in Chapter 4.1.

6.3 Confirmation Test

It has been understood that the A/D sampler is not having problems also there is a GPS pulse constantly at 1 KHz speed and hence the problems cannot be until the A/D conversion is completed. To check if it was the problem with Labview calculation, the two source current and the load current branches were tested again but the calculation was done using MS Excel. The input from GPS was cut off for a brief time and started again to synchronize the three boards. The output from each board was observed on a computer. The output is shown in Table 6-1

Table 6-1 A/D output and differential value in MS Excel

UNIT 2	UNIT 3	UNIT 1	Differential Value
327	413	633	**0.6993**
326	442	586	**0.5423**
338	464	516	**0.3333**
359	478	426	0.0449
386	486	354	**0.4797**
415	474	264	**0.6923**
442	455	202	**0.7468**
464	427	173	**0.7608**
477	395	167	**0.7349**
473	343	264	**0.4153**
456	324	345	0.1220
430	318	427	**0.3305**
400	320	511	**0.6564**

The values obtained from the three computers were hexadecimal values. These were transferred to MS Excel and converted to decimal format using the HEX2DEC function on Excel. The output from each unit and the differential value obtained by comparing the three values observed at the same sampling instant is shown for the first 13 values

The differential value is the value calculated by the differential protection algorithm. In this situation, no fault was initiated in the hardware and values were observed using Labview. The values marked in bold are those which are greater than 0.3. It is observed that, there are more values in bold compared to those in normal font in the differential value column. This indicates wrong data acquisition by the microcontroller board. The output waveform obtained from each board was verified to be a perfect sine wave in Figures 6.5, 6.6, 6.7. It implies that the A/D output from each board is not synchronized and hence due to comparison of data measured at different time instants, the differential unit concludes a fault. This mis-synchronization could prove fatal to the protection system since the protection algorithm developed requires synchronization upto 2 ms accuracy.

The output waveform obtained from the three units was plotted using MS Excel, utilizing the values obtained from Labview in this confirmation test. It is shown in Figures 6.12, 6.13, 6.14.

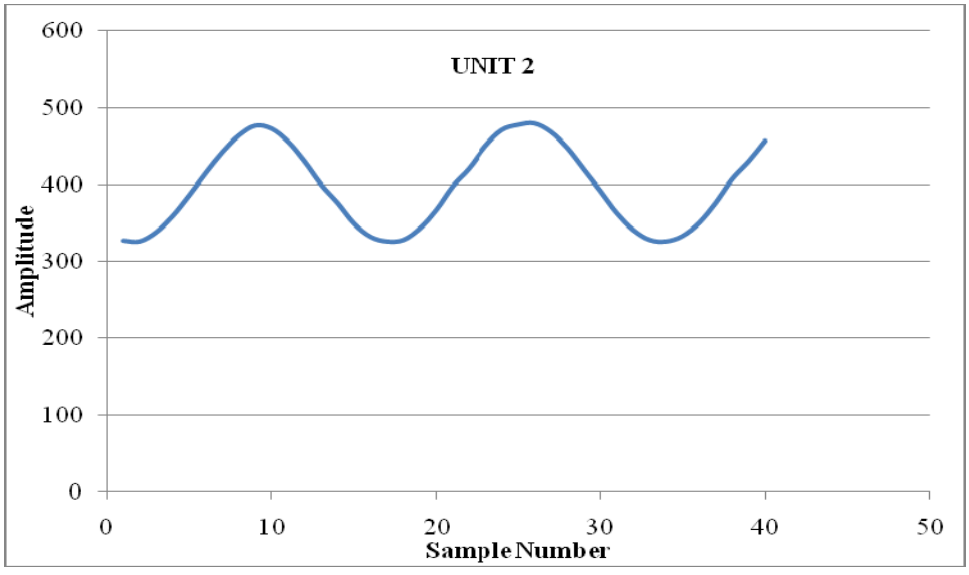

Figure 6.12 Unit 2 output from MS Excel

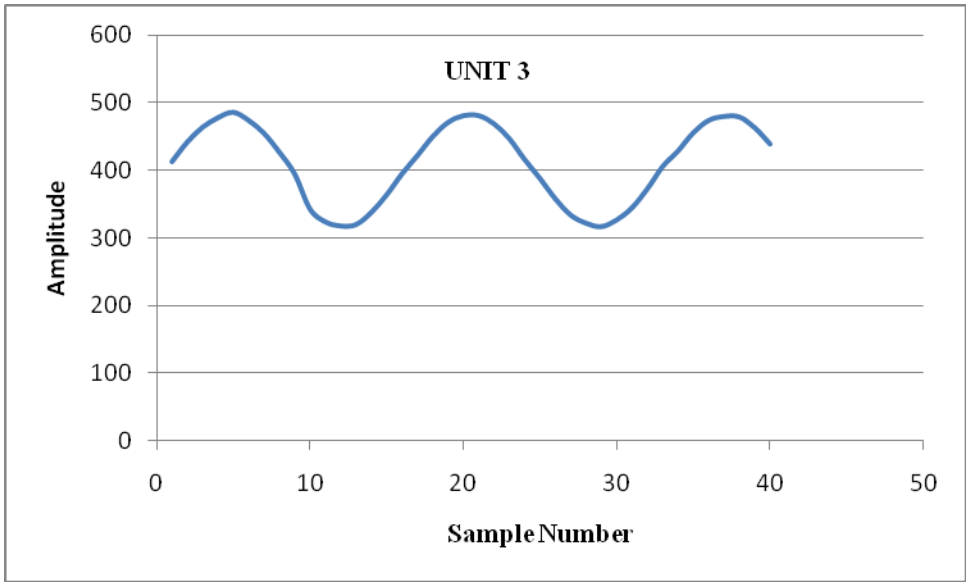

Figure 6.13 Unit 3 output from MS Excel

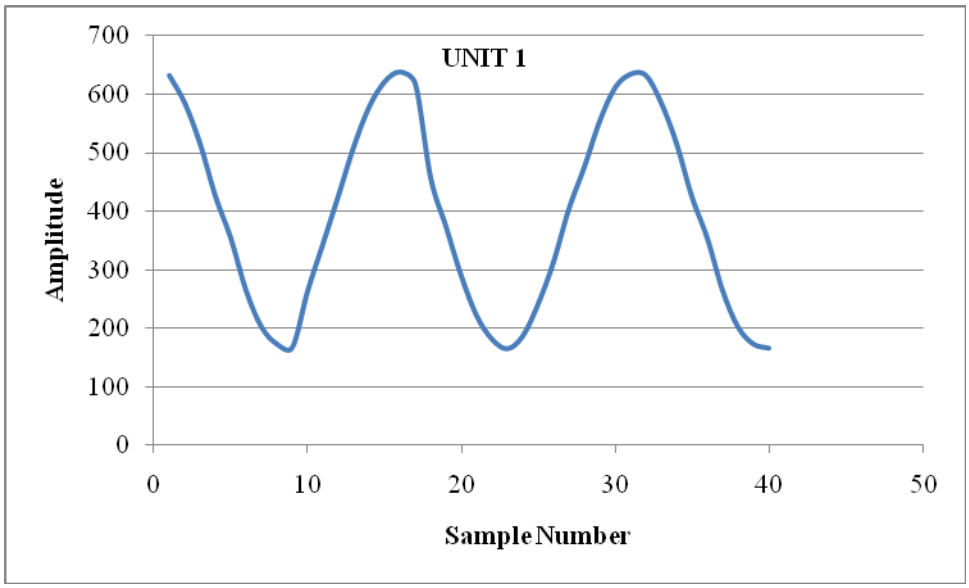

Figure 6.14 Unit 1 output from MS Excel

It is seen that all three output waveforms are sine wave and match with the characteristics plotted in Figures 6.5, 6.6, 6.7. Figure 6.14 looks skewed compared to Figure 6.12 and 6.13 because of the range of the plot on the Y axis and lesser number of samples used to represent each sine wave, 17 in this case. This further confirms the correct operation of the A/D converter. The

unit 1 waveform has a minimum amplitude lesser than 200, which is smaller than the output from either of unit 2 and unit 3. This has been handled inside the differential unit which adjusts the protection equation based on the maximum amplitude of the sample value. The above test confirms that the Labview calculation is not at fault. Further analysis was performed by shifting the sample values of the three boards in such a way that, values of the same instant with respect to the 60 Hz sine wave would be compared. This functionality could not be performed by the calculation unit in Labview, but could be manually done using MS Excel.

Table 6-2 Differential value comparing correct sampling points

UNIT 2	UNIT 3	UNIT 1	Differential Value
327	413	633	0.163865546
326	442	586	0.157464213
338	464	516	0.015873016
359	478	426	0.186046512
386	486	354	**0.435028249**
415	474	264	**0.437229437**
442	455	202	**0.335276968**
464	427	173	0.247706422
477	395	167	0.186858316
473	343	264	0.137420719
456	324	345	0.26779661
430	318	427	**0.555555556**
400	320	511	**0.612403101**
376	338	579	
349	364	622	
332	395	639	
326	422	619	
328	450	458	
343	471	374	
367	481	289	
398	481	221	
421	469	181	
450	447	165	

The values have been shifted considering unit 2 as the reference. It can be observed that the number of points marked in bold has reduced considerably. Presently 10 samples are being compared to make a fault decision. In the future, if 5 samples are compared to make the trip decision faster, the differential values in Table 6-1 will yield a trip decision even in the absence of a fault. The values in Table 6-2 will not generate a trip decision. Hence time synchronization is very important and the wrong trip decision is not because of the Labview calculation unit.

6.4 Inference

It is observed that the values are not time synchronized even before reaching the Labview calculation unit. The confirmation test rules out the problem due to Labview software. The other possibility is the Ethernet controller defect. But all three boards cannot have defective Ethernet controllers. Hence this problem of mis-synchronization has been attributed to the network issue or the TCP protocol. Since the number of data points is 1000 in a second from each board, the Ethernet switch receives 3000 sample points in one second and it has to transfer this data without any delay or a maximum delay of 1 ms. Since the TCP protocol has the issue of non sequential packets and also has the property of resending the packets that have not been received successfully by the remote computer, there is a definite possibility that the data points get delayed and hence the latency is increased [38][39]. The observation that there is a constant delay between the samples in all three boards, confirms the error due to network traffic is less likely to occur. A network traffic problem occurs randomly and there is no possibility of a constant and predictable latency in this case [33]. Hence the problem is more with the TCP protocol's capability to handle data points at a very high frequency. Since the protocol itself performs a lot of heavy operations, operations that take longer time to execute, and when this protocol is added to the PIC18F97J60 whose processor is not able to multi-task both the TCP operations and the A/D conversion at 1 KHz, there is a constant delay because of the constantly running TCP stack operations. Though the code written is the same, there is a constant delay between three identical boards to execute the same process. This difference is attributed to minor differences in the device frequency, which affects the TCP stack

call time, time to response to interrupt, complete an A/D conversion and other internal operations. Since these operations are constant for every sampling point, hence there is a fixed delay between two boards for the same sampling point. The final inference from the observations and the literature is that, TCP protocol is very heavy to handle, for this board when the data acquisition speed of 1 KHz is expected and any acquisition slower than that speed will not satisfy the design requirement from the FREEDM system.

6.5 Peak-to-peak test

To validate the output from the three microcontroller boards, the protection algorithm was modified in such a way that, the digital data points will be accumulated until every packet completes its data transmission to the computer. When this data transmission is completed, the algorithm monitors the complete packet and compares it to the peak value that is preset in the algorithm. The differential unit again compares the difference between the sum of the source currents and the load current, this time it monitors only the peak value in each data packet. Since the microcontroller has been programmed in such a way to send 17 samples in one packet, so each packet received from every microcontroller will be one full cycle of the sine wave. This implies there should be two peaks, both the positive and negative half cycle, for every packet. The differential unit compares the peaks from each data packet and calculates the difference. If the difference is more than 0.3, a trip signal is generated. The rms test was not executed because, a peak detector works faster than the rms calculator, because of the square root and mathematical complexities associated with the rms computation requiring a dedicated computer processor. The peak-to-peak signal for one unit and the trip signal generated is shown in Figure 6.15.

In figure 6.15, the bottom curve is the trip signal and the one on top of it is the peak-to-peak amplitude of one of the source units. It has to be noted that, the sample number in the horizontal axis starts at 3382386775 and ends at 3382386777. These are the packet numbers. This implies that when the packet arrives with 17 data points in it, this peak-to-peak algorithm sends a trip signal instantaneously. This is further confirmed by the rise in the peak-to-peak waveform. The in-

stant peak-to-peak waveform rises it indicates a fault and at that same instant the trip signal is generated. This confirms instantaneous trip signal generation by the protection algorithm.

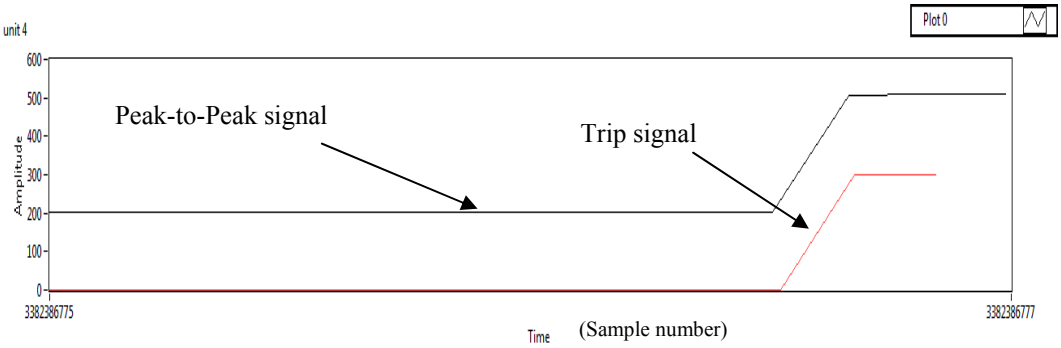

Figure 6.15 Peak-to-Peak signal and trip signal

The fact that this signal was initiated by the differential unit is shown in Figure 6.16.

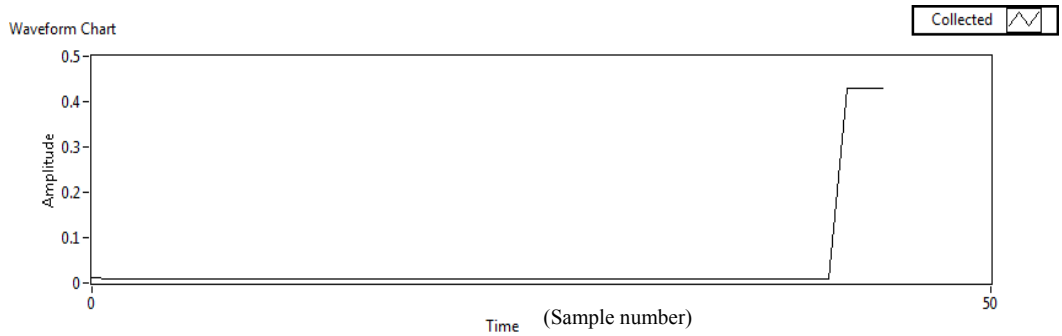

Figure 6.16 Differential unit output in new algorithm

This waveform can be compared to the one obtained in Figure 6.8, where constantly the differential unit calculated the difference to be greater than 0.3, even in the absence of fault. In Figure 6.16, in the absence of a fault, the differential unit yields a values almost 0, and when a fault occurs a value greater than 0.3 was calculated and a trip signal was generated as shown in Figure 6.15. This confirms the successful operation of the protection system. Though this system matches with the speed of conventional protection system, it does not trip inside a quarter cycle.

The trip signal generated is instantaneous, as observed in Figure 6.15, this was generated after obtaining 17 samples in a packet, which is one cycle time period. Hence the response time of the peak-to-peak protection system is one cycle.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusion

This research work was aimed at designing a fault current limiter and developing the data acquisition system to build an ultra fast and reliable protection system. The FCL designed is flexible in its settings in such a way that the user can set the maximum fault current to which the system is required to limit. Anything above that limit will be cut off by the FCL. The possibility of implementing the FCL using commercially available components is an added advantage.

The data acquisition system that has been developed ensures a very smooth depiction of circuit conditions. The sampling rate achieved is high enough to satisfy the requirements of the FREEDM system. Various conclusions drawn from this research work are:

- The FCL is designed for 7.2kV single phase and a fault current more than 3000 A. This design limits the fault current value to 1000 A, though this value is user-definable. This FCL has an important role in the design of SST and FID. These two devices are being developed considering the fault currents are limited. The capability of the FCL to limit the fault current and sustain that for a few cycles relaxes the design complexities of SST and FID.
- An IGBT based design of the FCL has been developed. This power electronics based FCL ensures fast switching and the capability to handle high magnitude currents such as 3000 A. The momentary high currents can be handled by the IGBT switches and the snubber circuit connected in parallel with the IGBT takes care of sudden high voltage that appears during the switching ON and OFF process of the IGBT.
- The FCL controller has been developed using logic gates which ensures easier realization into circuitry and also improves the flexibility of the controller to either modify or add additional functions such as fault current interrupting capability.
- The I_{max} and I_{min} values shown in the waveforms in Chapter 3.4 validate the robustness and the reliability of the FCL controller.

- The FREEDM protection system requires high speed tripping and also the capability to handle limited fault currents. Hence commercially available relays cannot be used for the design. This necessitates the design of a protection system which generates a trip signal in less than one quarter cycle. For the successful operation of a fast protection system, data needs to be acquired at a fast speed, which implies the design of a custom-made data acquisition system, to match the input characteristics required by the protection algorithm.
- The data acquisition system developed using PIC18F97J60 microcontroller has advanced features such as a 12 bit A/D converter, a 10 Mbps Ethernet controller and 8 Kbyte transmit buffer which makes it suitable choice based on the design requirement.
- The PICDEM.net2 development board with an on-board PIC18F97J60 and a PICtail daughter board as an addition to improve the system speed have been used to execute this DAQ system for the project.
- A satellite-synchronized clock provides synchronizing pulses to the A/D converter of each microcontroller. This ensures synchronized starting and completion of A/D conversion process in each PIC18F97J60.
- The strategy of creating two buffers, one to obtain A/D converted values and the other to send out these values to the Ethernet transmit buffer ensures fast processing speed and reduces the latency of the Ethernet communication.
- Outputs obtained from the three PIC's were observed using Labview software and it was inferred that the three outputs were not exactly time synchronized, but had a constant delay between any two waveforms. This implied that this data acquisition system cannot support the protection algorithm that compares the instantaneous values of the sample points. Since the output waveform obtained is a very smooth sine wave, an rms calculation could yield good results for the differential protection algorithm, though the response speed in this case will be slower than the requirement of the FREEDM system. The DAQ works successfully for an overcurrent protection algorithm, since that does not depend on time synchronization.

- The reason for the loss of synchronization of data packets has been attributed to TCP protocol, which performs many stack operations that are too heavy for the PIC18F97J60 processor to handle and to generate A/D output at 1 KHz.
- When the differential unit is modified to calculate the peak-to-peak amplitude and compares the outputs from the three microcontroller boards, a trip signal is generated successfully after 1 cycle. This 1 cycle time is the data acquisition and buffering time inside the protection algorithm.
- The final conclusion could be that the design is a working data acquisition system, but not meeting the timing requirements of the FREEDM system.

7.2 Future Work

- Implement the designed FCL using IGBT and analyze the impact of IGCTs replacing the IGBT.
- Explore the possibility of adding time synchronization using IEEE 1588 [59]
- Sourcing for another microcontroller which is 32 bit, which will be faster compared to this 8 bit PIC18F97J60, with an onboard 100 Mbps Ethernet controller may be executed. This improves the accuracy of the A/D converter also increases the possibility for more number of samples in a cycle, which may suit the input requirements of the differential protection algorithm.
- Develop the biasing circuit for the microcontroller analog input using OP AMP
- Understand the constraints posed by the TCP protocol and analyze the possibility of implementing a better protocol which is faster, such as the DNP3 or IEC 61850 [60][61].

REFERENCES

[1] FREEDM Systems Center First Year Annual Report, vol.1, Apr. 2009

[2] George G. Karady, Xing Liu, "Fault Management and Protection of FREEDM Systems," *IEEE PES General Meeting,* pp. 1-4, 2010

[3] M. Steurer, C. Edrington, O. Vodyakho, G. Karady, B. Chowdhury, S. Bhattacharya, "Fault Isolation Device for Power Electronics based Distribution Systems," *Proceedings of 2009 FREEDM Systems Center Annual Conference,* pp.75-78, May 2009

[4] H. Kubota, Y. K. Arai, M. Yamazaki, H. Yoshino, "A New Model of Fault Current Limiter using YBCO Thin Film," *IEEE Transactions on Applied Superconductivity,* vol. 9, No. 2, pp. 1365-1368, 1999

[5] L.Kovalsky, Xing Yuan, K.Tekletsadik, A.Keri, J.Bock, F.Breuer, "Applications of Superconducting Fault Current Limiters in Electric Power Transmission Systems," *IEEE Transactions on Applied Superconductivity,* vol. 15, No. 2, pp. 2130, 2005

[6] W.Paul, "Superconducting Machines: Energy Distribution Component," *Encyclopedia of Materials - Science and Technology,* vol. 1-11, pp.8938-8940, 2001

[7] H. Yamaguchi, K.Yoshikawa, M. Nakamura, T. Kataoka, K. Kaiho, "Current Limiting Characteristics of Transformer Type Superconducting Fault Current Limiter," *IEEE Transactions On Applied Superconductivity*, vol. 15, No. 2, pp.2106-2109, 2005

[8] Y.S.Cha, "Semi-Empirical Correlation for Quench Time of Inductively Coupled Fault Current Limiter," *IEEE Transactions On Applied Superconductivity*, vol. 15, No. 2, pp.1974-1977, 2005

[9] B. W. Lee, J. Sim, K. B. Park, I. S. Oh, " Practical Application Issues of Superconducting Fault Current Limiters for Electric Power Systems," *IEEE Transactions On Applied Superconductivity*, vol. 18, No. 2, pp.620-623, 2008

[10] M. Iwahara, S.C.Mukhopadhyay, S.Yamada, F.P Dawson,, "Development of Passive Fault Current Limiter in Parallel Biasing Mode," *IEEE Transactions On Magnetics*, vol. 35, No. 5, pp.3523-3525,1999

[11] M.M.R.Ahmed, " Comparison of the Performance of Two Solid State Fault Current Limiters in Distribution Network," *4th IET Conference on Power Electronics, Machines and Drives*, pp.772-776, Apr 2008

[12] M.A.Hannan, A.Mohamed, "Performance Evaluation of Solid State Fault Current Limiters in Electric Distribution System," *Proceedings of Student Conference on Research and Development*, pp.245-250, Aug 2003

[13] J.D.Glover, M.S.Sarma, T.Overbye "*Power System Analysis and Design*," 4th Ed. Thomson Learning, Inc. 2008

[14] M. Zhang, X.Dong, Z.Q.Bo, B.R.J. Caunce, A. Klimek, " A New Current Differential Protection Scheme for Two-Terminal Transmission Lines," *IEEE Power Engineering Society General Meeting,* Florida, pp.1-6, 2007

[15] N.Zhang, X.Z.Dong, Z.Q.Bo, S.Richards, A. Klimek, "Universal Pilot Wire Differential Protection for Distribution System," *IET 9th International Conference on Developments in Power System Protection*, pp.224-228, 2008

[16] R.E.Alonso, J.Leffew, S.Shreinivasan, W.Moreno, "Data Acquisition Device with Packet Based Communication Protocol for Engine Monitoring," *Proceedings of the Fourth IEEE International Caracas Conference on Circuits and Systems,* pp. I029 1-5, 2002

[17] M.Beck, "*Ethernet in the First Mile- The IEEE 802.3ah EFM Standard*," McGraw-Hill Professional Publishing, 2005

[18] A. Rafa, S. Mahmod, N. Mariun, W.Z. Wan Hassan, N.F. Mailah, "Protection of Power Transformer using Microcontroller-Based Relay," *Proceedings of Student Conference on Research and Development*, pp.224-227, Nov. 2002

[19] A.Katalin, "Acquisition System for Power Measurements and Harmonic Determination," *IEEE International Conference on Automation Quality and Testing Robotics*, pp.1-4, 2010

[20] A.Medina, M.Martinez-Cardenas, "Analysis of the Harmonic Distortion Impact on the Operation of Digital Protection Systems," *IEEE Power Engineering Society General Meeting*, vol.1, pp.741-745, 2005

[21] M.M.R.Ahmed, G.Putrus, L.Ran, R.Penlington, "Development of a Prototype Solid-State Fault-Current Limiting and Interrupting Device for Low-Voltage Distribution Networks," *IEEE Transactions on Power Delivery*, vol. 21, no.4, pp.1997-2005, 2006

[22] V.Khanna, "*Insulated Gate Bipolar Transistor IGBT Theory and Design*," 1st Ed. IEEE, pp.499-544, 2005

[23] W.McMurray, "Optimum Snubbers for Power Semiconductors," *IEEE transactions on Industry Applications,* vol.8, no.5, pp.593-600, 1972

[24] S.Kozala, *Designing for Reliability and Robustness*, Matlab Digest, The Mathworks, 2008

[25] M.A.Azam, K.Khan, "Design of the Ethernet based process data extraction algorithm and storage technique for industrial HMI systems," *The 2nd International Conference on Computer and Automation Engineering* , vol.1, pp.551-553, 2010

[26] Anonymous, *SEL 387E Relay-Current Differential and Voltage Protection Relay,* Schweitzer Engineering Laboratories, 2011

[27] F. Steinhauser,T. Matsushima,Y. Sukegawa,H. Watanabe, "Testing Communication aided Power System Protection Schemes using GPS Times Synchronization and Remote Control," *Transmission and Distribution Conference and Exhibition,* vol.1, pp 280-284, 2002

[28] N.Golovanov,G.C. Lazaroiu,M. Roscia, D.Zaninelli, "Harmonic summation in power systems with power electronic interfaced loads," *14th International Conference on Harmonics and Quality of Power,*pp.1-5, 2010

[29] A.Thirumalai, X.Liu, G.Karady, "Novel Digital Protection System for FREEDM loop", *Proceedings of FREEDM Systems Center Annual Conference, pp 22-26, 2010*.

[30] K. E. Martin, G. Benmouyal, M. G. Adamiak, M. Begovic, R.0. Bumett, Jr., K. R. Carr, A. Cobb, J. A. Kusters, S. H. Horowitz, G. R. Jensen, G. L. Michel, R. .I. Murphy, A. G. Phadke, M. S. Sachdev, J. S. Thorp, "IEEE Standard for Synchrophasors for Power Systems," *IEEE Transactions on Power Delivery*, vol.13, no. 1, pp.73-77, 1998

[31] A.Rindos,S Woolet, L.Nicholson,M. Vouk, "A performance evaluation of emerging ethernet technologies:A Switched/High-Speed/Full-Duplex Ethernet and Ethernet LAN emulation over ATM," *Proceedings of the Southeastcon, Bringing together Education, Science and Technology,* pp. 401-404, 1996

[32] H.Hashim, Z.A.Haron, "A study on industrial communication networking: Ethernet based implementation," *International Conference on Intelligent and Advanced Systems*, pp/1111-1114, 2007

[33] P.A.Farrell, Hong Ong; "Communication performance over a gigabit Ethernet network," *International Proceedings of the IEEE Performance, Computing and Communications Conference*, pp.181-189, 2000

[34] A.L.Hoi, R.J. Coomer, "Micro-controller for substation and network automation," *Third International Conference on Future Trends in Distribution switchgear*, pp.84-89, 1990

[35] Anonymous, *PIC18F97J60 Family Data Sheet*, Microchip Technology Inc., 2009

[36] B.Guha, B.Mukherjee, "Network Security via Reverse Engineering of TCP Code: Vulnerability Analysis and Proposed Solutions," *IEEE Network,* vol.11, no. 4, pp. 40-48, 1997

[37] J.L.Dolmeta, *MCF5272 Interrupt Service Routine for the Physical Layer Interface Controller,* Motorola Semiconductor Products Sector, 2001

[38] M.Simmons, *AN 1120 Ethernet Theory of Operation,* Microchip Technology Inc., 2008

[39] N.Rajbharati, *AN 833 The Microchip TCP/IP Stack*, Microchip Technology Inc.

[40] S.Bowling, *AN 693Understanding A/D Converter Performance Specifications,* Microchip Technology Inc., 2000

[41] Anonymous, *PIC32MX3XX/4XX Data Sheet,* Microchip Technology Inc., 2010

[42] Z. Hongfu, X. Xinyan, T. Yong; "Serial Communication Interface Design Based on Lab View and VC Mix Programming," 8th International Conference on Electronic Measurement and Instruments, pp.3-44 – 3-49, 2007

[43] Anonymous, *PICDEM.net2 Ethernet/Internet Development Board User's Guide*, Microchip Technology Inc., 2007

[44] H.A.Aziz, N.M.K.N.Yusoff, M.Z.B.M. Sapien, "MINI 11-Microcontroller Development Board for SCL Approach," *Proceedings of 2010 IEEE Student Conference on Research and Development* , pp.178-182, Dec 2010

[45] R. Droms, “Automated configuration of TCP/IP with DHCP,” *IEEE transactions on Internet Computing,* vol.3, no.4, pp.45-53, 1997.

[46] Anonymous, *MPLAB IDE User’s Guide with MPLAB Editor and MPLAB SIM Simulator,* Microchip Technology Inc., 2009

[47] Anonymous, *Ethernet PICtail Plus Daughter Board info sheet,* Microchip Technology Inc.

[48] O.Strobel,J. Lubkoll, “Fiber-Optic Communication,” *International Crimean Conference on Microwave and Telecommunication Technology,* pp.16-20, 2010

[49] Anonymous, *SEL-2407 Satellite-Synchronized Clock Instruction Manual,* Schweitzer Engineering Laboratories*,* 2010

[50] Anonymous, *SEL AcSELerator Software- Design and Set your System,* Schweitzer Engineering Laboratories

[51] Electrical Data Cables, available at http://www.selinc.com/tc/ElectricalDataCables/

[52] J.Postel, “RFC 768: User Datagram Protocol,” *Internet Engineering Task Force*. Available at http://tools.ietf.org/html/rfc768

[53] J.Postel, “RFC 792: Internet Control Message Protocol,” *Internet Engineering Task Force*. Available at http://tools.ietf.org/html/rfc792

[54] X.Guo,D.Lin, “Research on New Industrial Ethernet Network Management and Internet Remote Control System,” *International Conference on Computer Science and Information Technology,* pp.593-596, 2008

[55] D. C. Plummer ."RFC 826, An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware". *Internet Engineering Task Force*, Available at http://tools.ietf.org/html/rfc826

[56] Anonymous, Resistor Calculator for OP Amp Gain and Zero Shift, Texas Instruments.

[57] Anonymous, *Compiled Tips ‘N Tricks Guide,* Microchip Technology Inc., 2009.

[58] Anonymous, *MPLAB C18 C Compiler User’s Guide,* Microchip Technology Inc., 2002.

[59] L.Shuai,L.Yueming. J. Yuefeng, “An Enhanced IEEE 1588 Time Synchronization for Asymmetric Communication Link in Packet Transport Network,” *IEEE Communication Letters,* vol.14, no.8, pp.764-766

[60] IEEE Standard for Electric Power Systems Communications- Distributed Network Protocol (DNP3), 2010

[61] C.R. Ozansoy, A. Zayegh,A. Kalam, “Time synchronization in a IEC 61850 based substation automation system,” *Power Engineering Conference, Australasian Universities,* 2008

APPENDIX A

DIGITAL SAMPLE VALUES FROM THE DAQ SIMULATION

Sample Number	Time Stamps (s)	CT1 (A)	CT2 (A)	Load CT (A)
1	0	-0.6017	-0.6017	-1.2034
2	0.000122	-0.3588	-0.3588	-0.7175
3	0.000244	-0.115	-0.115	-0.2301
4	0.000365	0.1289	0.1289	0.2579
5	0.000487	0.3727	0.3727	0.7453
6	0.000609	0.6156	0.6156	1.2311
7	0.000731	0.8572	0.8572	1.7144
8	0.000852	1.097	1.097	2.194
9	0.001008	1.4009	1.4009	2.8018
10	0.001164	1.7	1.7	3.3999

APPENDIX B

PIC18F97J60 CODE TO GET A/D OUTPUT THROUGH THE ETHERNET

Main file is the “TCPIP ENCX24J600 Demo APP-C18”. Every file and function related to TCP function and also PIC18F related functions are available in this project file. This file is a part of the “Microchip Application Libraries v2009-11-18” available at www.Microchip.com.

TCPIPCONFIG.h

```
#ifndef __TCPIPCONFIG_H
#define __TCPIPCONFIG_H
#include "GenericTypeDefs.h"
#include "Compiler.h"
#define STACK_USE_ICMP_SERVER           // Ping query and response capability
#define STACK_USE_ICMP_CLIENT           // Ping transmission capability
#define STACK_USE_ANNOUNCE              // Microchip Embedded Ethernet Device
                                        Discoverer server/client
// Settings of the computer to connect from the PIC
#define MY_DEFAULT_HOST_NAME                    "MCHPBOARD"

#define MY_DEFAULT_MAC_BYTE1    (0x00)      // Use the default of
#define MY_DEFAULT_MAC_BYTE2    (0x04)      // 00-04-A3-00-00-00 if using
#define MY_DEFAULT_MAC_BYTE3    (0xA3)      // an ENCX24J600 or ZeroG
ZG2100
#define MY_DEFAULT_MAC_BYTE4    (0x00)      // and wish to use the internal
#define MY_DEFAULT_MAC_BYTE5    (0x00)      // factory programmed MAC
#define MY_DEFAULT_MAC_BYTE6    (0x00)      // address instead.

#define MY_DEFAULT_IP_ADDR_BYTE1    (169ul)
#define MY_DEFAULT_IP_ADDR_BYTE2    (254ul)
#define MY_DEFAULT_IP_ADDR_BYTE3    (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4    (1ul)

#define MY_DEFAULT_MASK_BYTE1       (255ul)
#define MY_DEFAULT_MASK_BYTE2       (255ul)
#define MY_DEFAULT_MASK_BYTE3       (0ul)
#define MY_DEFAULT_MASK_BYTE4       (0ul)

#define MY_DEFAULT_GATE_BYTE1       (169ul)
#define MY_DEFAULT_GATE_BYTE2       (254ul)
#define MY_DEFAULT_GATE_BYTE3       (1ul)
#define MY_DEFAULT_GATE_BYTE4       (1ul)

#define MY_DEFAULT_PRIMARY_DNS_BYTE1  (169ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE2  (254ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE3  (1ul)
#define MY_DEFAULT_PRIMARY_DNS_BYTE4  (1ul)

#define MY_DEFAULT_SECONDARY_DNS_BYTE1      (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE2      (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE3      (0ul)
#define MY_DEFAULT_SECONDARY_DNS_BYTE4      (0ul)
```

```
#define STACK_USE_TCP
// Allocate how much total RAM (in bytes) you want to allocate
// for use by your TCP TCBs, RX FIFOs, and TX FIFOs.
        #define TCP_ETH_RAM_SIZE                    (16384ul)
        #define TCP_PIC_RAM_SIZE                    (0ul)
        #define TCP_SPI_RAM_SIZE                    (0ul)
        #define TCP_SPI_RAM_BASE_ADDRESS            (0x00)
        #define TCP_CONFIGURATION
                ROM struct
                {
                        BYTE vSocketPurpose;
                        BYTE vMemoryMedium;
                        WORD wTXBufferSize;
                        WORD wRXBufferSize;
                } TCPSocketInitializer[] =
                {
                {TCP_PURPOSE_TCP_PERFORMANCE_TX, TCP_ETH_RAM, 2000, 1},
                };
                #define END_OF_TCP_CONFIGURATION
```

TCP.c

```
//Opens a TCP socket for listening or as a client.
TCP_SOCKET TCPOpen(DWORD dwRemoteHost, BYTE vRemoteHostType, WORD wPort,
BYTE vSocketPurpose)
{
        TCP_SOCKET hTCP;
        // Find an available socket that matches the specified socket type
        for(hTCP = 0; hTCP < TCP_SOCKET_COUNT; hTCP++)
        {
                SyncTCBStub(hTCP);

                // Sockets that are in use will be in a non-closed state
                if(MyTCBStub.smState != TCP_CLOSED)
                        continue;
                SyncTCB();
                // See if this socket matches the desired type
                if(MyTCB.vSocketPurpose != vSocketPurpose)
                        continue;
                // Start out assuming worst case Maximum Segment Size (changes when MSS
                // option is received from remote node)
                MyTCB.wRemoteMSS = 536;
                // See if this is a server socket
                if(vRemoteHostType == TCP_OPEN_SERVER)
                {
                        MyTCB.localPort.Val = wPort;
                        MyTCBStub.Flags.bServer = TRUE;
                        MyTCBStub.smState = TCP_LISTEN;
                        MyTCBStub.remoteHash.Val = wPort;
                        #if defined(STACK_USE_SSL_SERVER)
                        MyTCB.localSSLPort.Val = 0;
                        #endif
```

```
		}
		// Handle all the client mode socket types
		else
		{
			#if defined(STACK_CLIENT_MODE)
			{
// Each new socket that is opened by this node, gets the
// next sequential local port number.
if(NextPort < LOCAL_PORT_START_NUMBER || NextPort > LO-
CAL_PORT_END_NUMBER)
NextPort = LOCAL_PORT_START_NUMBER;
			// Set the non-zero TCB fields
				MyTCB.localPort.Val = NextPort++;
				MyTCB.remotePort.Val = wPort;
				// Flag to start the DNS, ARP, SYN processes
				MyTCBStub.eventTime = TickGet();
				MyTCBStub.Flags.bTimerEnabled = 1;
					switch(vRemoteHostType)
				{
					#if defined(STACK_USE_DNS)
					case TCP_OPEN_RAM_HOST:
					case TCP_OPEN_ROM_HOST:
MyTCB.remote.dwRemoteHost = dwRemoteHost;
MyTCB.flags.bRemoteHostIsROM = (vRemoteHostType == CP_OPEN_ROM_HOST);
MyTCBStub.smState = TCP_GET_DNS_MODULE;
						break;
					#endif

			case TCP_OPEN_IP_ADDRESS:
		// dwRemoteHost is a literal IP address.  This
		// doesn't need DNS and can skip directly to the
		// Gateway ARPing step.
MyTCBStub.remoteHash.Val = (((DWORD_VAL*)&dwRemoteHost)-
>w[1]+((DWORD_VAL*)&dwRemoteHost)->w[0] + wPort) ^ MyTCB.localPort.Val;

MyTCB.remote.niRemoteMACIP.IPAddr.Val = dwRemoteHost;
			MyTCB.retryCount = 0;
			MyTCB.retryInterval = (TICK_SECOND/4)/256;
			MyTCBStub.smState = TCP_GATEWAY_SEND_ARP;
					break;

					case TCP_OPEN_NODE_INFO:
MyTCBStub.remoteHash.Val = (((NODE_INFO*)(PTR_BASE)dwRemoteHost)-
>IPAddr.w[1]+((NODE_INFO*)(PTR_BASE)dwRemoteHost)->IPAddr.w[0] + wPort) ^
MyTCB.localPort.Val;
memcpy((void*)(BYTE*)&MyTCB.remote,
(void*)(BYTE*)(PTR_BASE)dwRemoteHost, sizeof(NODE_INFO));
MyTCBStub.smState = TCP_SYN_SENT;
SendTCP(SYN, SENDTCP_RESET_TIMERS);
		break;
				}
			}
```

```
                        #else
                        {
                                return INVALID_SOCKET;
                        }
                        #endif
                }

                return hTCP;
        }

        // If there is no socket available, return error.
        return INVALID_SOCKET;
}
```

Maindemo.c

```
/*
* This macro uniquely defines this file as the main entry point.
 * There should only be one such definition in the entire project,
 * and this file must define the AppConfig variable as described below.
 */
#define THIS_IS_STACK_APPLICATION

// Include all headers for any enabled TCPIP Stack functions
#include "TCPIP Stack/TCPIP.h"
#include "TCPIP Stack/TCP.h"

// Include functions specific to this stack application
#include "MainDemo.h"

#pragma udata Maindemo2             //to globally declare variables
#define SAMPLES_PER_SET  17u   // Transmit every 50 samples
#define SAMPLE_LENGTH    7u    // "0x" + 3 Hex digits + CR + LF
volatile char BUF_STR0[3+(SAMPLES_PER_SET*SAMPLE_LENGTH)];
volatile char BUF_STR1[3+(SAMPLES_PER_SET*SAMPLE_LENGTH)];

#pragma udata Maindemo3
volatile char *workstring = BUF_STR0;
volatile char *outstring = NULL;
volatile char buffno = 0, samps = 0;
  volatile unsigned short value;

// Declare AppConfig structure and some other supporting stack variables
APP_CONFIG AppConfig;

// Private helper functions.
static void InitAppConfig(void);
static void InitializeBoard(void);
static void ProcessIO(void);
```

//This function performs all TCP related functions such as creating a socket, listening to a port and sending data over TCP. The port numbers for the three boards are 9760, 9761, 9762. These values are chosen arbitrarily

```
void TCPTXPerformanceTask(void)
    {
     static TCP_SOCKET MySocket = INVALID_SOCKET;
     WORD w;


      // Start the TCP server, listening on PERFORMANCE_PORT
     if(MySocket == INVALID_SOCKET)
     {
          MySocket = TCPOpen(NULL, TCP_OPEN_SERVER, 9760,
          TCP_PURPOSE_TCP_PERFORMANCE_TX);
          // Abort operation if no TCP socket of type
          TCP_PURPOSE_TCP_PERFORMANCE_TEST is available
           // If this ever happens, you need to go add one to TCPIPConfig.h
        if(MySocket == INVALID_SOCKET)
          return;
     }


     w = TCPIsPutReady(MySocket);
if(w < (SAMPLES_PER_SET*SAMPLE_LENGTH) )
       return;

          TCPPutString(MySocket, (BYTE*) outstring );
           TCPFlush(MySocket);

          // Reset the "transmit" flag
          outstring=NULL;
   }
```

//The interrupt service routine (ISR) handles both high priority and low priority interrupts. Any interrupt when enabled will generate a flag and the processor will be automatically routed to this ISR.

```
#if defined(__18CXX)
          #if defined(HI_TECH_C)
          void interrupt low_priority LowISR(void)
          #else
          #pragma interruptlow LowISR
          void LowISR(void)
          #endif
          {
             TickUpdate();
          }

          #if defined(HI_TECH_C)
          void interrupt HighISR(void)
```

```
        #else
        #pragma interruptlow HighISR
        void HighISR(void)
        #endif
        {
        if(INTCON3bits.INT1IF) //Checking if the interrupt 1 flag is set
        {
  // Start a conversion
  ADCON0bits.GO = 1;

  // Wait until A/D conversion is done
  while(ADCON0bits.GO);
  //
  // Change last 4 bits
  if( ADRESH>9 )
    *workstring++ = 0x37+ADRESH;
  else
    *workstring++ = 0x30+ADRESH;
  //
  // Change next 4 bits
        value = (ADRESL>>4) & 0x0F;
  if( value>9 )
    *workstring++ = 0x37+value;
  else
    *workstring++ = 0x30+value;

   // Change first 4 bits
        value = ADRESL & 0x0F;
          if( value>9 )
    *workstring++ = 0x37+value;
  else
    *workstring++ = 0x30+value;

  // Send carriage return
  *workstring++ = 0x0D;
  //
  // Send Line Feed
  *workstring++ = 0x0A;
  //
  // Check for all points collected
  if( SAMPLES_PER_SET < ++samps  )
        {
          // Switch buffers
    *workstring = 0;  // Terminate the string
    samps = 0;        // New set
    if( buffno )
{
  // Using buffer 1
     workstring = BUF_STR0;   // Start filling buffer 0
     outstring = BUF_STR1;    // Submit buffer 1 for sending
                 buffno = 0;          // Switch to buffer 0
    }
```

```
            else
            {
        workstring = BUF_STR1;   // Start filling buffer 1
        outstring = BUF_STR0;    // Submit buffer 0 for sending
                        buffno = 1;          //Switch to buffer 1
      }
     }
     INTCON3bits.INT1IF=0; //clear interrupt flag
   }


        #if defined(STACK_USE_UART2TCP_BRIDGE)
                        UART2TCPBridgeISR();
                        #endif
                        #if defined(ZG_CS_TRIS)
                                zgEintISR();
                        #endif // ZG_CS_TRIS
            }
            #if !defined(HI_TECH_C)
            #pragma code lowVector=0x18
            void LowVector(void){_asm goto LowISR _endasm}
            #pragma code highVector=0x8
            void HighVector(void)
            {
            _asm goto HighISR _endasm
            }
            #pragma code // Return to default code section
            #endif


#if defined(__18CXX)
void main(void)
#else
int main(void)
#endif
{
            static DWORD t = 0;
            static DWORD dwLastIP = 0;

            // Initialize application specific hardware
            InitializeBoard();

            #if defined(USE_LCD)
            // Initialize and display the stack version on the LCD
            LCDInit();
            DelayMs(100);
            strcpypgm2ram((char*)LCDText, "TCPStack " VERSION "  "
                        "               ");
            LCDUpdate();
            #endif

            // Initialize stack-related hardware components that may be
```

```
        // required by the UART configuration routines
          TickInit();
          TCPInit();
        // Initialize Stack and application related NV variables into AppConfig.
        InitAppConfig();

           // Initiates board setup process if button is depressed
        // on startup
         if(BUTTON0_IO == 0u)
        {
                #if defined(EEPROM_CS_TRIS) || defined(SPIFLASH_CS_TRIS)
        // Invalidate the EEPROM contents if BUTTON0 is held down for more than 4 seconds
                DWORD StartTime = TickGet();
                LED_PUT(0x00);

                while(BUTTON0_IO == 0u)
                {
                        if(TickGet() - StartTime > 4*TICK_SECOND)
                        {
                        #if defined(EEPROM_CS_TRIS)
                           XEEBeginWrite(0x0000);
                           XEEWrite(0xFF);
                           XEEEndWrite();
                           #elif defined(SPIFLASH_CS_TRIS)
                           SPIFlashBeginWrite(0x0000);
                           SPIFlashWrite(0xFF);
                           #endif

                        #if defined(STACK_USE_UART)
putrsUART("\r\n\r\nBUTTON0 held for more than 4 seconds.  Default settings restored.\r\n\r\n");
                                #endif

                                LED_PUT(0x0F);
                while((LONG)(TickGet() - StartTime) <= (LONG)(9*TICK_SECOND/2));
                                LED_PUT(0x00);
                                while(BUTTON0_IO == 0u);
                                Reset();
                                break;
                        }
                }
                #endif

                #if defined(STACK_USE_UART)
    DoUARTConfig();
                #endif
  }
        // Initialize core stack layers (MAC, ARP, TCP, UDP) and
        // application modules (HTTP, SNMP, etc.)
  StackInit();
```

```
StackTask();        // This task performs normal stack task.  Call one time before main loop to
make sure all is running
   StackApplications(); // Invokes each of the core stack application tasks .  Call one time before
main loop

// Now that all items are initialized, begin the co-operative
// multitasking loop.  This infinite loop will continuously
// execute all stack-related tasks, as well as the
// application's functions.  Custom functions should be added
// at the end of this loop.
   // Note that this is a "co-operative mult-tasking" mechanism
   // where every task performs its tasks (whether all in one shot
   // or part of it) and returns so that other tasks can do their
   // job.
   // If a task needs very long time to do its job, it must be broken
   // down into smaller pieces so that other tasks can have CPU time.

   while(1)
   {
      // Blink LED0 (right most one) every second.
      if(TickGet() - t >= TICK_SECOND/2ul)
      {
         t = TickGet();
         LED0_IO ^= 1;
      }

          if( NULL!=outstring )                  //INDICATES data ready to transmit
   {       TCPTXPerformanceTask();

          StackTask();
          StackApplications();
   }

          #if defined(STACK_USE_ICMP_CLIENT)
                    PingDemo();
                    #endif

          // If the local IP address has changed (ex: due to DHCP lease change)
      // write the new IP address to the LCD display, UART, and Announce
      // service
                    if(dwLastIP != AppConfig.MyIPAddr.Val)
                    {
                              dwLastIP = AppConfig.MyIPAddr.Val;

                              #if defined(STACK_USE_UART)
                                        putrsUART((ROM char*)"\r\nNew IP Address: ");
                              #endif

                              DisplayIPValue(AppConfig.MyIPAddr);

          #if defined(STACK_USE_ANNOUNCE)
                              AnnounceIP();
```

```
                #endif
        }

// Writes an IP address to the LCD display and the UART as available
void DisplayIPValue(IP_ADDR IPVal)
{
//      printf("%u.%u.%u.%u", IPVal.v[0], IPVal.v[1], IPVal.v[2], IPVal.v[3]);
   BYTE IPDigit[4];
        BYTE i;
#ifdef USE_LCD
        BYTE j;
        BYTE LCDPos=16;
#endif

        for(i = 0; i < sizeof(IP_ADDR); i++)
        {
           uitoa((WORD)IPVal.v[i], IPDigit);

                #if defined(STACK_USE_UART)
                        putsUART(IPDigit);
                #endif

                #ifdef USE_LCD
                        for(j = 0; j < strlen((char*)IPDigit); j++)
                        {
                                LCDText[LCDPos++] = IPDigit[j];
                        }
                        if(i == sizeof(IP_ADDR)-1)
                                break;
                        LCDText[LCDPos++] = '.';
                #else
                        if(i == sizeof(IP_ADDR)-1)
                                break;
                #endif
        #if defined(STACK_USE_UART)
                        while(BusyUART());
                        WriteUART('.');
                #endif
        }

        #ifdef USE_LCD
                if(LCDPos < 32u)
                        LCDText[LCDPos] = 0;
                LCDUpdate();
        #endif
}


// 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.
```

```
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;
            LED_PUT(0x00);

#if defined(__18CXX)
  OSCTUNE = 0x40; //41.667Mhz

// Enable internal PORTB pull-ups
    INTCON2bits.RBPU = 1;

// Configure USART
    TXSTA = 0x20;
    RCSTA = 0x90;

//Setup the A/D converter
#if defined(PICDEMNET2)
       ADCON0 = 0x0D;               // ADON, Channel 3
       ADCON1 = 0x0A;               // Vdd/Vss is +/-REF, AN0, AN1, AN2, AN3, AN4 are analog
       ADCON2 = 0x8A;               //right justify 2 TAD, FOSC/32
#endif


// Do a calibration A/D conversion
#if defined(__18F97J60)

            ADCON0bits.ADCAL = 1;
             ADCON0bits.GO = 1;
            while(ADCON0bits.GO);
            ADCON0bits.ADCAL = 0;
#endif


// Enable Interrupts
   RCONbits.IPEN = 1;               // Enable interrupt priorities
    INTCONbits.GIEH = 1;//Enable high priority interrupts
    INTCONbits.GIEL = 1; //Enable low priority interrupts
    INTCON3bits.INT1IE=1;           //Enable interrupt on INT1

      }
```

APPENDIX C

PIC18F97J60 CODE TO GET A/D OUTPUT THROUGH THE RS-232

Every function declared in Maindemo.c in APPENDIX B applies here. Except one, TCPTXPerformanceTask. Modifications have been made into this function to enable serial communication. All the functions are the same because they only perform data processing and there is only change in the data communication method. The TCP.c is disabled since there is no TCP related operation. TCPIPConfig.h is modified accordingly. Only the changes to the program in APPENDIX B are shown here.

TCPIPConfig.h

```
#define STACK_USE_UART              //Performs UART related functions
// #define STACK_USE_ICMP_SERVER    // Ping query and response capability
// #define STACK_USE_ICMP_CLIENT    // Ping transmission capability
#define STACK_USE_ANNOUNCE          // Microchip Embedded Ethernet Device Discoverer
                                    server/client
```

UART.c

```
#define __UART_C

#include "TCPIPConfig.h"

#if defined(STACK_USE_UART)

#include "TCPIP Stack/TCPIP.h"

#if defined(__18CXX)      // PIC18

        char BusyUSART(void)
        {
                return !TXSTAbits.TRMT;
        }

        void CloseUSART(void)
        {
          RCSTA &= 0x4F;  // Disable the receiver
          TXSTAbits.TXEN = 0;   // and transmitter

          PIE1 &= 0xCF;   // Disable both interrupts
        }

        char DataRdyUSART(void)
        {
                if(RCSTAbits.OERR)
                {
                        RCSTAbits.CREN = 0;
                        RCSTAbits.CREN = 1;
                }
```

```
  return PIR1bits.RCIF;
}

char ReadUSART(void)
{
  return RCREG;                    // Return the received data
}

void WriteUSART(char data)
{
  TXREG = data;      // Write the data byte to the USART
}

void getsUSART(char *buffer, unsigned char len)
{
  char i;    // Length counter
  unsigned char data;

  for(i=0;i<len;i++)  // Only retrieve len characters
  {
    while(!DataRdyUSART());// Wait for data to be received

    data = getcUART();    // Get a character from the USART
                          // and save in the string
    *buffer = data;
    buffer++;             // Increment the string pointer
  }
}

void putsUSART( char *data)
{
  do
  {  // Transmit a byte
    while(BusyUSART());
    putcUART(*data);
  } while( *data++ );
}

void putrsUSART(const rom char *data)
{
  do
  {  // Transmit a byte
    while(BusyUSART());
    putcUART(*data);
  } while( *data++ );
}             } while( *data++ );
}
#endif


#endif   //STACK_USE_UART
```

Maindemo.c

```
#define THIS_IS_STACK_APPLICATION

// Include all headers for any enabled TCPIP Stack functions
#include "TCPIP Stack/TCPIP.h"
#include "TCPIP Stack/UART.h"
#include "TCPIP Stack/TCP.h"

void TCPTXPerformanceTask(void)
    {

        putsUSART(AN0String);  //send it through the USART

        // Reset the "transmit" flag
        outstring=NULL;
    }
```

The other functions ISR(), Main() and the InitializeBoard ()are the same as in APPENDIX B.

APPENDIX D

COMMANDS IN AcSELERATOR QUICKSET SOFTWARE TO SEL 2407

Level 1

=>2ac

Password: ? &&&&&&&&&&&&

Level 2

=>>FOR

Port	Function	Reference	Port	Function	Reference
+OUT1	IRIG-B000	Local Time	OUT2	IRIG-B000	Local Time
OUT3	IRIG-B000	Local Time	OUT4	IRIG-B002	Local Time
OUT5	IRIG-B002	Local Time	OUT6	IRIG-B002	UTC Time
SERIAL	IRIG-B002	Local Time	FIBER	IRIG-B002	Local Time
MOD	IRIG-B122	Local Time			

=>>FOR OUT1 KPPS

Current format of OUT1 is IRIG-B000 Local Time

Change format to KPPS Local Time (Y,N)? Y

OUT1 format changed.

=>>FTQ 1

Force Time Quality to 1 nanosecond (Y,N) ? Y

Force IRIG Time Quality Active.

=>>