Next Generation Frequency Disturbance Recorder Design and Timing Analysis

Lei Wang

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

Doctor of Philosophy

in

Electrical Engineering

Committee Members:
Yilu Liu (Co-Chair)
Virgilio Centeno (Co-Chair)
Richard W. Conners
Patrick Schaumont
Shu-Ming Sun

May 21st, 2010

Blacksburg, Virginia

Keywords: synchronized sampling, phasor measurements, time synchronization, indoor GPS, Oscillator, RTAI

Next Generation Frequency Disturbance Recorder Design and Timing Analysis

Lei Wang

Abstract

In recent years, the subject of wide-area synchronized measurements has gained a significant amount of attention from the power system researchers. All of this started with the introduction of the Phasor Measurement Unit (PMU), which added a new perspective in the field of wide-area measurement systems (WAMS). With the ever evolving technologies over the years and the need for a more cost effective solution for synchronized frequency measurements, the Frequency Monitoring Network (FNET) was developed and introduced by the Power IT laboratory at Virginia Tech. The FNET is comprised of many Frequency Disturbance Recorders (FDR) geographically distributed throughout the United States. The FDR is a dedicated data acquisition device deployed at the distribution level, which allows for a lower cost and easily deployable WAMS solution. With Internet connectivity and GPS timing synchronization, the FDR provides high accuracy frequency, voltage magnitude and voltage angle data to the remote servers.

Although the current FDR design is up to the standard in terms of the measurement accuracy and portability, it is of interest to further the research into alternative architectures and leverage the ever advancing technologies in high speed computing. One of the purposes of this dissertation is to present novel design options for a new generation of FDR hardware design. These design options will allow for more flexibility and to lower reliance on some vendor specific components. More importantly, the designs seek to allow for more computation processing capabilities so that more accurate frequency and angle measurements may be obtained.

Besides the fact that the accuracy of frequency and angle measurement is highly dependent on the hardware and the algorithm, much can be said about the role of timing synchronization and its effects on accurate measurements. Most importantly, the accuracy

of the frequency and angle estimation is highly dependent on the sampling time of local voltage angles. The challenges to accurate synchronized sampling are two folds. One challenge has to do with the inherent fallbacks of the GPS receiver, which is relatively high cost and limited in availability when the satellite signal is degraded. The other challenge is related to the timing inaccuracies of the sampling pulses, which is attributed to the remainder that results from the imperfect division of the processor counter. This dissertation addresses these issues by introducing the implementation of the high sensitivity (indoor) GPS and network timing synchronization, which aims to increase the availability of frequency measurements in locations that would not have been possible before. Furthermore, a high accuracy timing measurement system is introduced to characterize the accuracy and stability of the conventional crystal oscillator. To this end, a new method is introduced in close association with some prior work in generating accurate sampling time for FDR. Finally, a new method is introduced for modeling the FDR based on the sampling time measurements and some results are presented in order to motivate for more research in this area.

Acknowledgement

First and foremost I would like to express my sincere appreciations to my primary advisor, Dr. Yilu Liu, for her invaluable guidance and encouragements throughout my studies in the graduate school. In addition, she did not only allow me the flexibility to pursue my research interests but also gave me the opportunity to work in the industries where I have obtained valuable experiences in many different fields. Much can be said about her support throughout my studies and I will be forever in her debt for her advice, time and most of all friendship.

My sincere appreciation goes out to Dr. Richard Conners for his continuous support and invaluable guidance in all aspects of my research studies. Even upon retirement, he selflessly offered a tremendous amount of time to meet with me over the weekends to accommodate for my full-time work schedule. I would also like to send my sincere appreciation to Dr. Virgilio Centeno for serving as the co-advisor of my committee and provided the support throughout the years of my graduate studies. Finally, I would like to send my sincere appreciation to Dr. Patrick Schaumont and Dr. Shu-Ming Sun for serving as my committee members. The research work would not have been possible without their guidance and assistance.

Special thanks goes out to all of the past and present FNET team members, especially Javier Fernandez who initiated the testing of network time protocol and Bruce Billian for paving the way for the next generation FDR design. In addition, I would like to send special thanks to Dr. Kevin Zhong, Dr. Ryan Zuo, Dr. Emily Xu, Dr. Henry Zhang, Vivian Liang, Jon Burgett, Dr. Matt Gardner, Dr. Josh Wang, Dr. Will Kook, Dr. Mark Baldwin, Kevin Khan, Dr. Jason Bank, Dr. Jingyuan Dong, Dr. Tao Xia, Dr. Il-Yop Chung, Dr. Alan Yuan, Yingcheng Zhang, Lang Chen, Joanna Wu, Kelly Ye and Penn Markham. For it has been a great pleasure and an honor to have known and worked with these talented and dedicated colleagues.

This is dedicated to my parents, for whom education always comes first.

Table of Contents

Table of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation and Background

The power transmission network is a highly complex and vast system, yet it is one of the most important networks in the world. However, experiences such as the August 14, 2003 blackout has shown a dire need of an improved power transmission network with high reliability and efficiency. In fact, billions of dollars were bid in the recent years to improve the reliability of the creaking system in the United States alone. To this end, power system researchers around the world are looking for methods to ensure stability in a much heavily loaded power transmission network. One method that has been increasingly attracting attention in the recent years is the wide area measurement system (WAMS). Such monitoring tool can be used to accurately capture real-time events and system dynamics. The need for monitoring and recording of power system data has been recognized for a long time and one of the most well known methods is the synchronized phasor measurements or also known as synchrophasors. Synchronized phasor measurements provide a standard of referring the phasor representation of a power system voltage or current to an absolute time reference. The absolute reference is provided in the form of common high accuracy clocks synchronized to coordinated universal time (UTC). The first high accuracy phasor measurement was made possible by the invention of phasor measurement units (PMU).

The invention of the PMU goes back to 1988, when the first PMU prototypes were developed at Virginia Tech and was later commercialized in the early 1990's. In order to monitor a complex wide area system like the power transmission network, there is a need for a synchronized monitoring solution with highly dynamic phasor measurement technique. To meet these needs, a Global Positioning System (GPS) receiver is used in the PMU in conjunction with the frequency estimation algorithm invented under the leadership of Dr. Arun Phadke. However, the high installation cost of PMU's limits its deployment capabilities. Early studies clearly point to the need for much wide system measurement coverage, coverage that can be quickly and economically obtained. To meet

these goals, an Internet based Frequency Monitoring Network (FNET) was proposed in 2001 by the PowerIT group at Virginia Tech, and has since been implemented [1].

The FNET is composed of mainly the Frequency Disturbance Recorder (FDR) and the Information Management System (IMS) as it is shown in Figure 1.1. Somewhat similar to the PMU, the FDR acts as the sensor that performs local frequency data measurements and transmits the data to remote servers on the Internet. The FDR's are synchronized to the GPS and installed at 110V or 220V distribution voltage level of a typical office outlet. The IMS works as a central server, which provides data collection, storage, web service, post disturbance analysis and other information management functions. Working together, the FDR's and IMS provides continuous, real-time, wide area gathering of GPS time stamped frequency data for power system monitoring [1]. Based on these valuable data, the power system researchers can investigate a variety of protection and control applications, which can be used to improve the performance of the power transmission network.

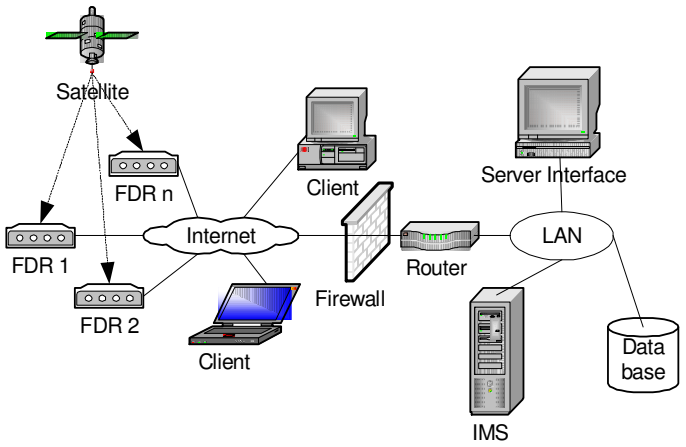

Figure 1.1 Frequency Monitoring Network (FNET) Architecture

1.1.1 Frequency Monitoring Network (FNET) and Frequency Disturbance Recorder (FDR)

The FNET developed at Virginia Tech consists of several FDRs distributed across the United States and Canada as well as a central data collection server. The advantages of the FNET implementation are twofold. Firstly, each FDR interfaces with the electric grid

at low voltage levels with easily accessible wall sockets. Secondly, the cost of FDR hardware is on the order of $1000 whereas the PMU ranges in several times of $1000. However, there are tradeoffs to be made for such a low cost installation and design. On the low voltage level, the FDR will not be able to measure the three-phase quantities and increases the vulnerability to noises of various forms, including high frequency switching loads and electromagnetic interference (EMI). Overall, with the FNET's capability for mass coverage, its shortcomings are overwhelmingly overshadowed.

1.1.2 FDR Algorithm Review

The phasor measurement algorithm has its roots from the days of Charles Proteus Steinmetz, who presented a paper on simplified mathematical description of the waveforms of alternating electricity. Since then, the word 'phasor' was invented and was eventually evolved into the calculation of real time phasor measurements.

The first generation of the FDR device was developed in 2002-2003 and there have been more than 40 units deployed in the United States power grid and a few 50Hz units in Europe and Africa. The first generation FDRs have virtually no algorithm error at 60Hz and their calculated frequency accuracy is better than ±0.0005Hz [1].

With the ever increasing advances in microprocessor technologies, more and more digital algorithms have been applied to calculating power system frequency, such as Modified Zero Crossing [8], Level Crossing [7], Least Squares Error [10], Newton Method [11], Kalman Filtering [12], Discrete Fourier Transform [6] and Smart Discrete Fourier Transform [9]. Each algorithm has its unique advantages and disadvantages. Nevertheless, phasor angle analysis provides fast and accurate frequency estimation over a wide range of frequency and the computation requirement is kept at a minimum for real-time implementation. From practical point of view, phasor angle analysis is recognized as the most appropriate for FDR measurement applications.

The phasor computation algorithm is based on the relationship that small frequency perturbation can be approximated by measuring the rate of the change of its phasor angle [6]. In mathematical terms this can be represented by the equation:

$$\frac{(d\phi)}{(dt)} \approx 2\pi(\Delta f)$$

Equation 1-1

Note that the phasor angle to frequency relationship in equation 1-1 is based on the assumption that frequency deviation from nominal is relatively small. To illustrate this, let's assume the input signal is a sinusoidal waveform written as:

$$x(t) = \sqrt{2}X\sin(2\pi(f_0 + \Delta f)t + \phi)$$

Equation 1-2

Where f_0 is the nominal system frequency, Δf is the deviation from nominal frequency and φ is phasor angle.

Using the recursive discrete fourier transform, the new *rth* phasor can be expressed as [4]:

$$\vec{X}^{(r)}_{f_0+\Delta f} = \vec{X}^{(r)}_{f_0} e^{j(N-1)\pi\frac{\Delta f}{Nf_0}} e^{j\frac{2\pi\Delta f}{Nf_0}r} \frac{\sin\pi\frac{\Delta f}{f_0}}{N\sin\pi\frac{\Delta f}{Nf_0}} - \vec{X}^{*(r)}_{f_0} e^{-j\frac{2\pi\Delta f}{Nf_0}r} e^{-j(N-1)\left(\frac{2\pi}{N}+\frac{\pi\Delta f}{Nf_0}\right)} \frac{\sin\frac{\pi\Delta f}{f_0}}{N\sin\left(\frac{2\pi}{N}+\pi\frac{\Delta f}{Nf_0}\right)}$$

Equation 1-3

Where N is the number of samples taken per cycle. Now assuming that Δf is relatively small, the following relationships are introduced [4]:

$$\frac{\sin\frac{\pi\Delta f}{60}}{N\sin\frac{\pi\Delta f}{60N}} \approx 1$$

Equation 1-4

$$\frac{\sin\frac{2\pi\Delta f}{60N}}{N\sin\left(\frac{2\pi}{N}+\pi\frac{\Delta f}{60N}\right)} \approx 0$$

Equation 1-5

As a result, the second term of Equation 1-3 can be eliminated and the simplified expression can be written as [4]:

$$X^{(r)}_{f_0+\Delta f} = X^{(r)}_{f_0} e^{j(N-1)\pi\frac{\Delta f}{Nf_0}} e^{\frac{j2\pi\Delta f r}{Nf_0}}$$

Equation 1-6

Since φ_r and φ_{r+1} are defined as the phasor angle of the *rth* and *(r+1)th* phasor respectively, the change in phasor angle can be calculated as [3]:

$$\frac{d\phi}{dt} = \lim_{t \to 0} \frac{\phi_r - \phi_{r-1}}{t}$$

Equation 1-7

The denominator *t* can be represented by the time between consecutive samples and can be approximated by [3]:

$$t \approx \frac{1}{N f_0}$$

Equation 1-8

Therefore as *N* is increased, t will approach zero and establish the relationship represented in Equation 1-1. Finally, the relationship between the frequency deviation and phasor angles can be obtained [3]:

$$f = f_0 + \Delta f = f_0 + \frac{1}{2\pi}\frac{d\phi}{dt}$$

Equation 1-9

At this point, it is important to note that as the system frequency approaches nominal system frequency, the phasor angle algorithm reaches its highest accuracy and conversely, the accuracy degrades as system frequency deviates from nominal system frequency. However, it will be seen later that a multiple resampling method can be used to minimize this error [3].

Given the relationship between phasor angle and frequency deviation, the implementation of the algorithm is discussed in [3] and being introduced here to provide as a reference. To illustrate phasor representation, one can write a sinusoidal input signal of frequency *w* in the form:

$$x(t) = \sqrt{2}x \sin(wt + \phi)$$

Equation 1-10

and its equivalent representation in phasor form:

$$Xe^{(j\phi)} = X\cos\phi + jX\sin\phi$$

Equation 1-11

Assuming that the signal *x(t)* is sampled N times per cycle of the nominal voltage waveform to produce the sample set shown below:

$$x_k = \sqrt{2}x \sin\left(\frac{(2\pi)}{N}k + \phi\right)$$

Equation 1-12

The discrete fourier transform (DFT) of equation 1-10 can be written as:

$$X = \frac{1}{(\sqrt{2})}(X_c + jX_s)$$ **Equation 1-13**

Where

$$X_c = \frac{2}{N}\sum_{k=0}^{(N-1)} X_k \cos\left(\frac{(2\pi)}{N}k\right)$$ **Equation 1-14**

$$X_s = -\frac{2}{N}\sum_{k=0}^{(N-1)} X_k \sin\left(\frac{(2\pi)}{N}k\right)$$ **Equation 1-15**

To obtain more phasor results the recursive phasor computation technique is used to obtain the successive phasor:

$$(X_c)^{(k+1)} = (X_c)^{(k)} + \frac{2}{N}\left(x_{(k+1)} - x_{(k+1-N)}\right)\cos\left(\frac{(2\pi)}{N}k\right)$$ **Equation 1-16**

$$(X_s)^{(k+1)} = (X_s)^{(k)} - \frac{2}{N}\left(x_{(k+1)} - x_{(k+1-N)}\right)\sin\left(\frac{(2\pi)}{N}k\right)$$ **Equation 1-17**

Where $k = Nf_0t$

Then the angle of the kth phasor can be calculated by:

$$\phi(k) = \tan^{-1}\left(-\frac{X_s{}^k}{X_c{}^k}\right)$$ **Equation 1-18**

Assuming that voltage phasor angles vary as a quadratic function with respect to the sample number,

$$\phi(k) = a_0 + a_1 k + a_2 k^2$$ **Equation 1-19**

Using a computation window of M phasor angles, the relationship between the phasor angles and the sample number can be put into matrix form:

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ . \\ . \\ . \\ \phi_M \end{bmatrix} = \begin{bmatrix} & & \\ 1 & 1 & 1 \\ 1 & 2 & 2^2 \\ . & . & . \\ . & . & . \\ . & . & . \\ 1 & M & M^2 \\ & & \end{bmatrix} \begin{bmatrix} \\ a_0 \\ a_1 \\ a_2 \\ \\ \end{bmatrix}$$ **Equation 1-20**

Equation 1-18 can also be simply written as:

$$\phi = Xa$$ **Equation 1-21**

The unknown matrix *a* can be solved using the least error square solution:

$$a = \left[X^T X\right]^{-1} X^T \phi$$ **Equation 1-22**

Where the pseudo-inverse matrix $\left[X^T X\right]^{-1} X^T$ is known as the gain matrix and can be computed offline.

Once the values of a_1 and a_2 are calculated, the frequency and rate of change of frequency can be calculated. Taking the derivative of Equation 1-17 with respect to *k*:

$$\frac{d\phi}{dk} = a_1 + 2a_2 k$$ **Equation 1-23**

Then taking the derivative of Equation 1-21 with respect to time *t*:

$$\frac{dk}{dt} = N f_0$$ **Equation 1-24**

The derivative of phasor angle with respect to time *t* can be written as:

$$\frac{d\phi}{dt} = \frac{d\phi}{dk}\frac{dk}{dt} = N f_0 \frac{d\phi}{dk} = N f_0 (a_1 + 2a_2 N f_0 t)$$ **Equation 1-25**

Deviation in frequency, Δf can be obtained by:

$$\Delta f \approx \frac{1}{2\pi} N f_0 (a_1 + 2a_2 N f_0 t)$$ **Equation 1-26**

$$\frac{df}{dt} \approx \frac{1}{2\pi} 2(N f_0)^2 a_2$$ **Equation 1-27**

Where *t* determines which instant inside the computation window the computed frequency corresponds to.

As mentioned before, the simplification of Equation 1-4 is made to obtain the relationship between phasor angle and deviated frequency. However, this approximation will introduce some error in estimation. The frequency estimation will be more accurate when the actual frequency approaches the frequency established for the sampling rate. Therefore, resampling the waveform with the estimated frequency and using the new phasor to perform corrections to the final estimation is rather an attractive solution [3]. In detail, assume the nominal frequency is 60Hz and the number of samples per cycle N is 24, resulting in 1440 samples taken per second. When the frequency has changed to 55 Hz, each cycle will now have about 26.18 samples instead of the 24 samples as indicated. To resolve this issue, re-normalization through resampling can be implemented to

interpolate the points so that there will always be 24 samples per cycle regardless of waveform frequency. The resampling algorithm can be summarized as the following:

$$Z_1 = Z_m \sin(\phi)$$ **Equation 1-28**

$$Z_2 = Z_m \sin(\phi + \alpha) = Z_m \sin\phi \cos\alpha + Z_m \cos\phi \sin\alpha$$ **Equation 1-29**

Where

Z_m is the amplitude of the waveform

φ is a sample instant and is an arbitrary known value

α is the interval between two samples at the new frequency and is equal to $2\pi f_{new}/(Nf_0)$

Combining Equations 1-26 and 1-27 results in:

$$Z_m \cos\phi = \frac{(z_2 - z_1 \cos\alpha)}{\sin\alpha}$$ **Equation 1-30**

Let x be the fractional distance between z_1 and z_2, the resampled point z' is then given by:

$$z' = Z_m \sin(\phi + x\alpha)$$ **Equation 1-31**

$$z' = Z_m \sin\phi \cos x\alpha + Z_m \cos\phi \sin x\alpha$$ **Equation 1-32**

$$z' = Z_1 \cos x\alpha + (z_2 - z_1 \cos\alpha)\frac{\sin x\alpha}{\sin\alpha}$$ **Equation 1-33**

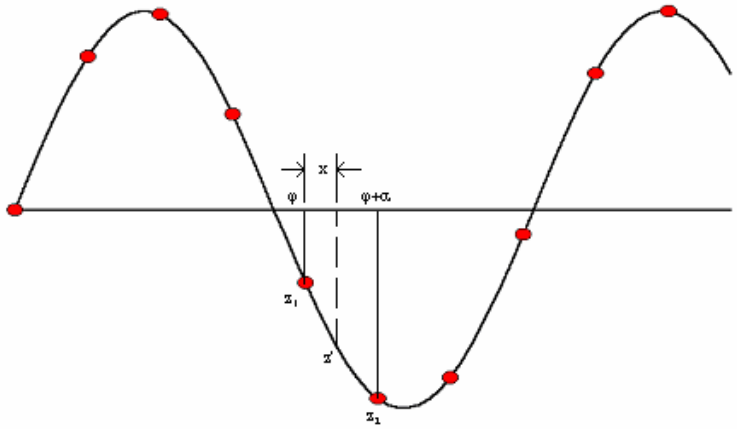

Figure 1.2 Illustration of resampling

After the resampling points are found, the phasor angles of the new resampled data are computed and another estimation is made using Equation 1-20, Equation 1-24 and Equation 1-25 to obtain the correction frequency, $\Delta f'$, and the final rate of change of frequency. As a result, the final frequency estimation is computed by:

$$f_{final} = f_0 + \Delta f + \Delta f'$$ **Equation 1-34**

1.1.3 FDR Algorithm Implementation

The FDR phasor algorithm is composed of three stages [4]:

1) Compute rough frequency estimation using raw voltage from sampled data.
2) Resample the waveform using the resampling frequency
3) Compute the correction frequency using the new resampled voltage and apply the correction to the rough estimation in step 1.

In the first two generation FDRs, embedded processors were used for phasor computation to allow for a compact and small size design. However, since a large amount of voltage is being collected and computation time is limited, considerations need to be given to maximizing computation speed and measurement accuracy. Hence, the actual implementation of the algorithm only uses certain selected phasors. This speeds up the computation time tremendously without significant losses in accuracy.

In the first two generation FDRs, the number of cycles to compute a frequency estimate was decided to be 6 nominal cycles. The number of samples per cycle used to compute a phasor is 24. So after initialization, a new phasor will be obtained every 1/1440 = 0.6944 millisecond. The whole estimation proceeds recursively with a sliding window length of 8 cycles and 6 cycles for estimating frequency. The reason to use an 8 cycle sliding window is because it takes one cycle of sampled data to compute the first phasor value. In addition, when the actual frequency is below nominal 60Hz (i.e. 55Hz), more data is needed to fulfill the 6 cycles criterion for resampling.

The implementation of the resampling is rather tricky. Since the resampling and subsequent frequency deviation computations are based on the first approximate frequency estimation, it is required to locate the exact instant in the waveform where the resampling frequency is used. This is not so significant when the waveform frequency is constant but is crucial when the frequency undergoes major frequency swings. Under such special circumstances it is practical to designate t in Equation 1-25 to be an appropriate value for the application. Since the measured frequency will exhibit an

oscillatory behavior due to its phasor angles, it is necessary to minimize this error by using the average of the estimated frequency as the resampling frequency. The average of the estimated frequency can be computed as:

$$f_r = \frac{\sum_{k=1}^{24} f_k}{24}$$ **Equation 1-35**

Where f_k is the first frequency estimate for k = 1 to 24 and f_r as the resampling frequency. At this point, it is necessary to make an assumption that the resampling frequency f_r is the frequency at the end of two and a half cycle in the first computation window. Such an assumption is valid for power system phasor measurement since the power system frequency deviation within the window is so small that the contribution to frequency estimation error is insignificant. Therefore, the second frequency estimation can be computed in the same manner as the first frequency estimation:

$$\Delta f' \approx \frac{1}{2\pi} N f_r (a_1 + 2a_2 C)$$ **Equation 1-36**

Where C is the sample instant of the estimation and is computed by:

$$C = \left(\left(N\left(\frac{6}{2} - \frac{1}{2}\right) - \left(6N - 6N \cdot \frac{f_0}{f_r}\right) \right) \frac{f_r}{f_0} + \frac{N}{2} \right)$$ **Equation 1-37**

To illustrate the phasor measurement algorithm, Figure 1.3 and Figure 1.4 show a flowchart of the algorithm in [4].

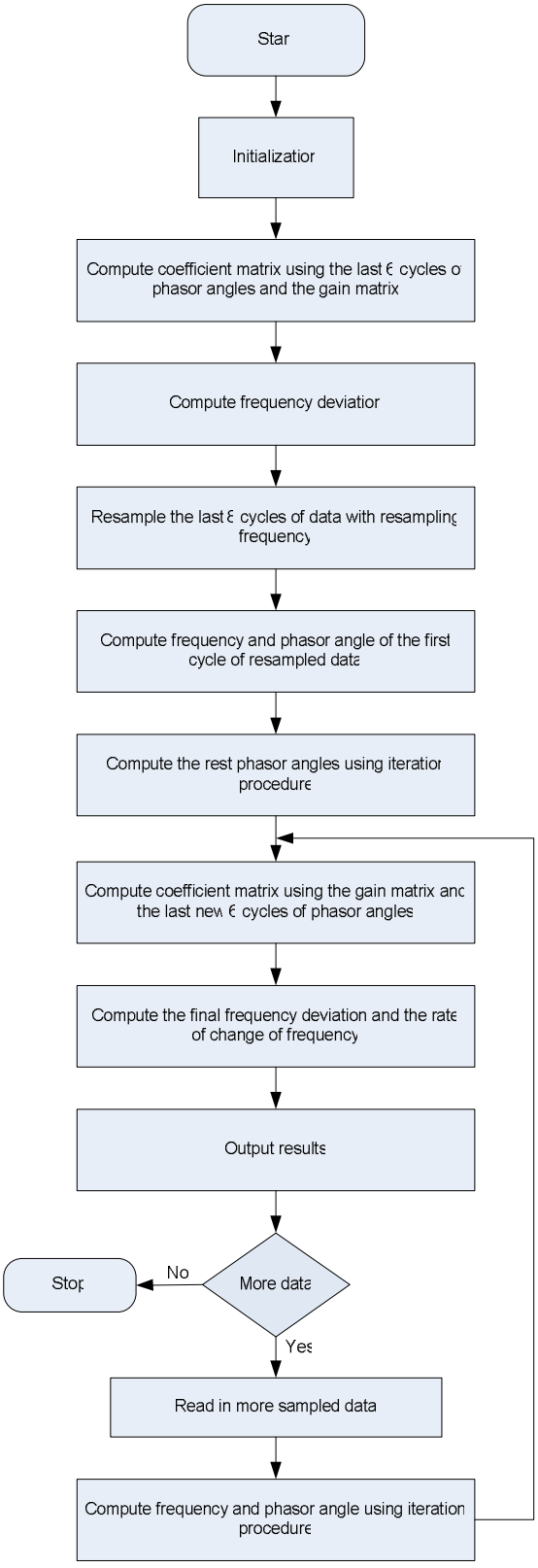

Figure 1.3 Flowchart of phasor estimation algorithm

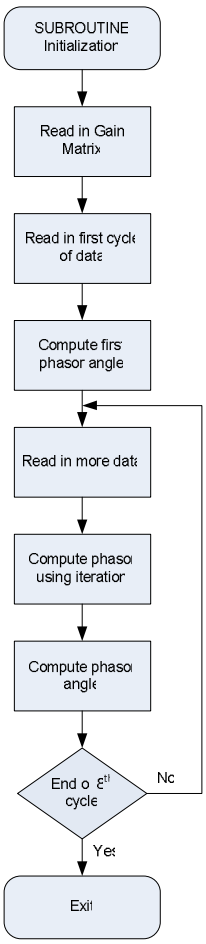

Figure 1.4 Flowchart of initialization for phasor estimation algorithm

1.1.4 FDR Software Architecture

The FDR software architecture is composed of three states of operation including acquisition, initialization and collection. The state machine diagram of FDR is shown in Figure 1.5 and each state is summarized below.

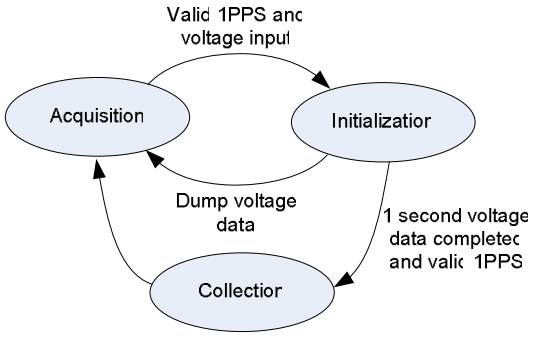

Figure 1.5 Top level state machine of FDR

Acquisition

The acquisition state is obtained by meeting the requirement of GPS satellite acquisition. Since the voltage measurements need to be aligned to the UTC time, the FDR should not begin phase measurement until a valid GPS 1PPS is obtained. An internal timer is used to verify the accuracy of the 1PPS by measuring the length between one PPS rising edge to the next. If the length is within the bound of a pre-determined acceptable error range the measurement continues for the next PPS until four consecutive PPS meets the requirement, at which point the system begins phasor measurement.

Initialization

Once the FDR establishes an accurate 1PPS, the system begins the sampling process to feed the phasor measurement algorithm. The initialization process is allocated one second so that the synchronized phasor measurement can begin the next second. If GPS signal is still valid as verified in the acquisition state, the FDR will switch to collection state. If the GPS 1PPS is not valid, the FDR will reset to acquisition state and discard all measurements. This ensures the FDR is making phasor measurement only when accurate time synchronization is established.

Collection

When the initialization process ends, the FDR is capable of making phasor measurements continuously until either a loss of GPS or input voltage signal. If either errors occurs, the FDR will dump the current phasor measurements and go back to acquisition state.

1.2 Objectives

This introduction has covered the theoretical basis for frequency and phasor angle estimation, as well as the implementation of the algorithm along with the FDR software architecture. The goal of this dissertation is divided into several parts but the focus is on the design of the next generation FDR with emphasis on improving the accuracy and availability of synchronized sampling. In order to approach the subject of FDR design, it is essential to first understand the requirements at the system level. Then the top-down

design approach can be used. In addition, it is equally as important to address the drawbacks of the current FDR designs so that those drawbacks can be mitigated in the next generation design. Also, some FDR architectures are being proposed based on the design requirements and an evaluation is conducted to seek out the advantages and disadvantages of each.

Given the fact that the crystal oscillator is inaccurate in long term timekeeping and phasor angle measurement requires microseconds synchronization accuracy with respect to Universal Coordinated Time (UTC), the conventional GPS is used as a precision timing reference for FDR synchronized sampling. However, the conventional GPS is very much limited with respect to availability and cost. The signal attenuation effects are more pronounced for FDR applications since some of the newer office buildings comes with energy efficient glass windows, which prohibits the conventional GPS from acquiring GPS satellite signals. It is of interest to seek out alternate timing references that provides higher availability with similar level of accuracy.

What is also important to the accuracy of frequency and phasor angle measurement is the subject of oscillator characteristics since it is used to generate sampling pulses for synchronized measurements. To this end, it is important to provide some insights into its accuracy and stability. Thus far, there has not been any study conducted in characterizing the oscillator to produce accurate FDR sampling time. Ultimately, the characterization of the conventional oscillator should provide valuable insights to improve the FDR sampling time accuracy. Specifically, a high resolution timing measurement system can be used to measure accuracy and stability of the conventional oscillator. Furthermore, it is intended that such a measurement system will be able to not only measure the accuracy and stability of the crystal oscillator, but also provide some insights into the accuracy of different precision time synchronization sources as well as the FDR sampling time. To this end, models can be developed based on the sampling time and simulations can be performed to observe the effect of the sampling time on frequency and angle measurements. Ultimately, any errors associated with timing reference and sampling time will be directly reflected in the frequency and angle measurements.

1.3 Organization

This dissertation is organized into 8 chapters starting with the introduction in the first chapter. The second chapter provides the background in FDR design requirements and specifications and addresses the limitations in the current FDR design. The third chapter goes over the evaluation of different FDR architectures with respect to its advantages and disadvantages. The fourth chapter addresses the issue with the obsolete GPS receiver and introduces the high sensitivity GPS receiver for high availability frequency and angle measurements. The fifth chapter summarizes the development of a high resolution and low cost timing measurement system. At the same time, measurements are taken from the processor clock counter to estimate the crystal oscillator accuracy and stability. The sixth chapter goes over the methodology to measure the FDR sampling time and takes the initial effort in modeling the FDR with the measurement data. The seventh chapter presents the network timing synchronization and proposes the use of network time protocol (NTP) for frequency estimation. Finally, the eighth chapter concludes the dissertation with a summary of the dissertation and future works.

Chapter 2 Frequency Disturbance Recorder Design Requirements

2.1 Background

One of the most popular system design method for real-time and embedded systems is the top-down design approach. The top-down method is a natural way to approach a complex design task, mainly because it relies on multiple levels of abstraction to limit the number of independent concepts at each level of the design. Such design approach is very much required for the next generation FDR design as all of the requirements and specifications are being refined over many years of experience with the first and second generation FDRs. Major levels of abstraction in the design process can be visualized in Figure 2.1.

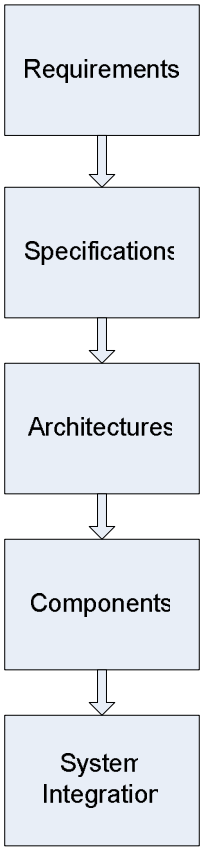

Figure 2.1 Top-down design approach

Starting at the very top, the requirements block gives an informal description of the FDR functionalities. The type of requirements should include both functional and non-

functional requirements. Functional requirements are solely the I/O (input/output) relationships and the non-functional requirements include timing, performance, cost, power consumption and physical size and weight. In the second block from the top, specifications should accurately define the application requirements. The specification should be simple and easily understood so that one can verify that it meets the system and overall expectation of the application requirements.

This chapter is focused on an overview of the requirements and specifications of FDR design. Also, some of the drawbacks of the present FDR designs will be presented so that the next generation design will be able to address these fallbacks and make the necessary improvements.

2.2 FDR System Level Requirements

Similar to the PMU design but with some different design goals mainly focused on portability and ease of deployment, the FDR is composed of four major subsystems as it is shown in Figure 2.2, these include the processor, analog input subsystem, timing subsystem and network communication subsystem. The analog input subsystem is composed of the analog to digital converter, the transformer and the signal conditioning system. The timing subsystem is composed of an accurate frequency and time reference that is synchronized with the UTC time. Additionally, the timing subsystem should be capable of generating variable frequency pulses synchronized to UTC time for triggering the ADC for conversion. Finally, the network communication subsystem provides the means to transmit the measurement results to the central server at Virginia Tech.

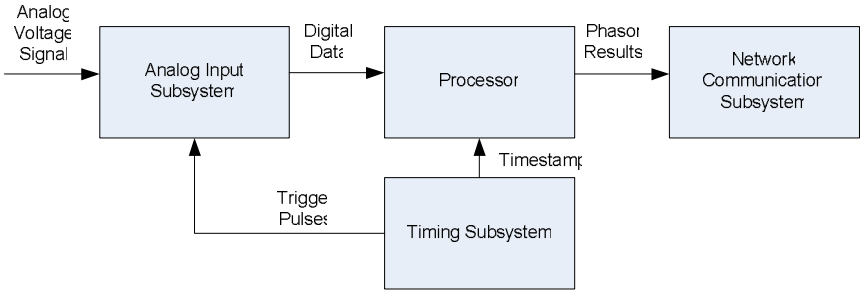

Figure 2.2 System block diagram of FDR

Since portability and ease of deployment are the some of the main goals of the FDR design philosophy, an embedded system is the most obvious choice for such a design and it has been the norm for the first and second generation FDR design. As a result, should the next generation FDR continue to follow the embedded trend, it is worthwhile to highlight the requirements for the system.

At the minimum, the system should be able to perform floating point arithmetic. Furthermore, a counter with input capture is required to count the number of clock cycles in between each 1 PPS output from the GPS. In the case such that the counter register overflows, an interrupt will need to be generated to count the number of overflows in between pulses. The resultant number of clock cycles and overflows as measured by the timing subsystem is used to validate the accuracy of 1PPS, which is also a GPS status indicator. Additionally, a pulse width modulator subsystem with output compare is needed to trigger the external ADC for conversion. It is crucial for the counter, PWM and input capture and output compare to have at least 16 bits register for high resolution timing. Depending on the resolution of the ADC to be used, one external interrupt pin is needed to indicate the external ADC has latched data into its output register and a general purpose input/output (GPIO) subsystem is needed to receive data from external ADC. To allow for transmission of data to the main server and receiving timestamps from the GPS, there needs to be two serial communication interfaces (SCI). To allow for efficient I/O processing the system should also have an interrupt controller that is capable of assigning priorities to the interrupts. One such example would be a hierarchical interrupt system. For the external interfaces, there should be a liquid crystal display (LCD) interface circuit and its physical port for ribbon cable connection, a port for mounting the external ADC, and some breadboard area for input filter circuitry.

2.2.1 Analog Input Subsystem

The analog to digital converter is at the heart of the analog input subsystem. There is a variety of ADC's available today. However, some of the most widely used ADC architectures are the integrating ADC (Dual slope) [109], Flash ADC (Parallel ADC) [105], Pipelined ADC [108], Sigma Delta ADC [107], and SAR (Successive

Approximation Register) ADC [106]. The five architectures differ in conversion algorithm and encoding method, but what's more important to the design of FDR are characteristics of each ADC such as speed of conversion, resolution, size and price. Tradeoffs need to be made to select the device meeting all of the criteria. However, the relationship between such factors and the phasor estimation accuracy is rather complicated. So far there is no absolute requirement established for neither the resolution nor the synchronized data samples in the IEEE synchrophasor standards [2]. However, what is required by these standards is the phasor measurement be within 1% of TVE (total vector error) at the reporting times specified. TVE can be calculated by [2]:

$$TVE_{(k)} = 100\% \cdot \frac{\left| X_{MEAS(k)} - X_{IDEAL} \right|}{\left| X_{IDEAL} \right|}$$

Equation 2-1

Furthermore, the resolution of the ADC is not the only determining factor of phasor measurement accuracy, characteristics such as linearity, noise and gain stability provides the fundamental limitations on performance. Upon past experiences with the first and second generation FDR, 16 bits of SAR ADC resolution is more than adequate with 14 bits as the minimum resolution required for accurate FDR frequency measurement. Finally, a throughput rate of at least 100kSPS and minimized conversion time, power consumption and size is needed to meet portable real-time requirements.

In general, the SAR ADC is widely used for nearly all multiplexed data acquisition systems as well as in instrumentation applications. Due to its ease of interfacing and integration, it is being used in both the first generation and second generation FDRs. It has no pipeline delay and is available with resolutions up to 18 bits and sampling rates up to 5Msps [106].

For a wide variety of industrial measurement applications, the Sigma Delta and Integrating ADCs are the ideal candidates. With respect to the other ADC architectures, the integrating ADC is slow in speed with typical conversion speed of 20 milliseconds and low input bandwidths but their capability to reject high frequency noise and fixed low frequencies makes them attractive for certain industrial applications [107][109]. In

addition its popularity in the industrial applications, the Sigma Delta converter dominates in audio and voiceband markets. The inherent oversampling capability in these converters lowers the requirements on the ADC anti-aliasing filter. Nevertheless, the Sigma Delta converter has several drawbacks. Firstly, the filter does not provide attenuation at integer multiples of the modulator sampling frequency. Also, the speed of the conversion and filtering results in long latency between the start of the sampling cycle and the first digital output [107].

Flash ADCs and pipelined ADCs are known to be high speed converters. In particular, flash ADCs are more accustomed to converting signals with large bandwidths at high speeds. However, the Flash ADCs has relatively low resolution, consumes more power and can be comparatively expensive [105]. This drawback alone limits the Flash ADC market to high frequency applications that typically cannot be addressed any other way. On the other hand, the pipelined ADCs are used in applications requiring sampling rates ranging from approximately 5 Msps to greater than 100 Msps and have lower power consumption than that of the Flash ADC. The pipeline ADC has a good balance of size, speed, resolution and power dissipation, and has become increasingly popular to major data converter manufacturers [105]. Finally Table 2-1 summarizes some of the characteristics of the commonly used ADC architectures in [105][106][107][108][109].

Table 2-1 ADC architectures and attributes

ADC types	Flash	SAR	Integrating	Pipelined	Sigma Delta
Method	Cascaded comparators	Binary search	Integration and comparator	Parallel comparators	Modulator and filter
Encoding	Thermometer code	Successive approximation	Analog integration	Digital Correction	Oversampling, decimation filter
Conversion time	Constant with increase in resolution	Increase linearly with resolution	Constant with increase in resolution	Increase linearly with resolution	Tradeoff between data output rate and resolution
Resolution	Limited to 8	8 to 18	10 to 18	8 to 16	12 to 24
Size	Increases exponentially with resolution	Increase linearly with resolution	Constant with increase in resolution	Increase linearly with resolution	Constant with increase in resolution

In addition to the ADC circuitry, the analog input subsystem has a signal conditioning component that is used to filter the transformer output, before it is being input to the ADC. Depending on the ADC and the transformer that is being implemented, the filter design could vary. However, the basic requirement remains the same, and that is to take a signal from the transformer and attenuate it to a lower voltage level acceptable to the ADC. Generally a simple voltage divider circuit can take care of this. In addition, to get rid of the high frequency noise that's associated with outlet voltages, a low pass filter is needed. In general, the low pass filter is essentially a second order anti-aliasing filter. An anti-aliasing filter could be passive or active but the smaller package size of active filter is rather attractive for FDR design. As a matter of fact, the first generation FDR used a passive filter design and the second generation FDR improvised by implementing an active filter design. Nevertheless, the design requirements remains the same, with a specified DC gain of unity, the filter needs to have a cutoff frequency of 720Hz [4].

2.2.2 Central Processing Unit

The Central Processing Unit (CPU) in the FDR design can vary by a great deal and with the drastic improvements in digital processor technologies, many different solutions are available. Nevertheless, there are several criteria that need to be addressed in the processor selection process.

The very first criterion is deciding whether a floating point or fixed point processor is needed. The fundamental difference between the two is their respective numeric representation of data. While fixed point hardware performs strictly integer arithmetic, floating point hardware supports either integer or real arithmetic. Therefore, fixed point hardware has less bit width and it is more hardware efficient to implement than its floating point counterpart. Furthermore, a fixed point processor requires less hardware resources resulting in a smaller package and some cost reduction comparing with that of floating point. As a result, fixed point processors are more favored for high volume applications like digitized voice and telecom. However, the fixed point processor requires fixed point implementation in software, which requires careful consideration in precision

losses and overflow as a result of fixed point arithmetic. Although recent advances in more sophisticated C compilers has merged the gap between the software complexity of floating point and fixed point processor [18]. To this end, the fixed point processor still maintains an edge in cost savings compared to the floating point processor, with the trade-off of less precision.

Another important factor to consider in the selection of processors is the number of bits for floating point arithmetic. The IEEE Standard for Binary Floating Point Arithmetic (IEEE 754-1985) is the most widely used standard for floating point computation with two commonly used formats, single precision and double precision. Almost all of today's microcontrollers and DSPs come with either single precision or double precision floating point unit (FPU). At this point it is necessary to take a look at the accuracy of phasor algorithm using both single precision and double precision arithmetic. Figure 2.3 shows the frequency and angle measurements conducted by two different FDRs with the same input. Also, Figure 2.4 shows the frequency and angle difference between the two FDRs. In comparison, Figure 2.5 and Figure 2.6 show the measurement results using single and double precision arithmetic, and the resulting differences between the two units respectively. If the measurements were perfect, the results would match each other between the two units. Nevertheless, the measurement differences between two FDRs with double precision arithmetic are similar in magnitudes compared to the measurement differences between two FDRs with single precision and double precision. Specifically, the average frequency measurement for the two units is within 0.1 mHz of each other, which is negligible for any FNET application. It is clear that single precision arithmetic is adequate for FDR algorithm implementation.

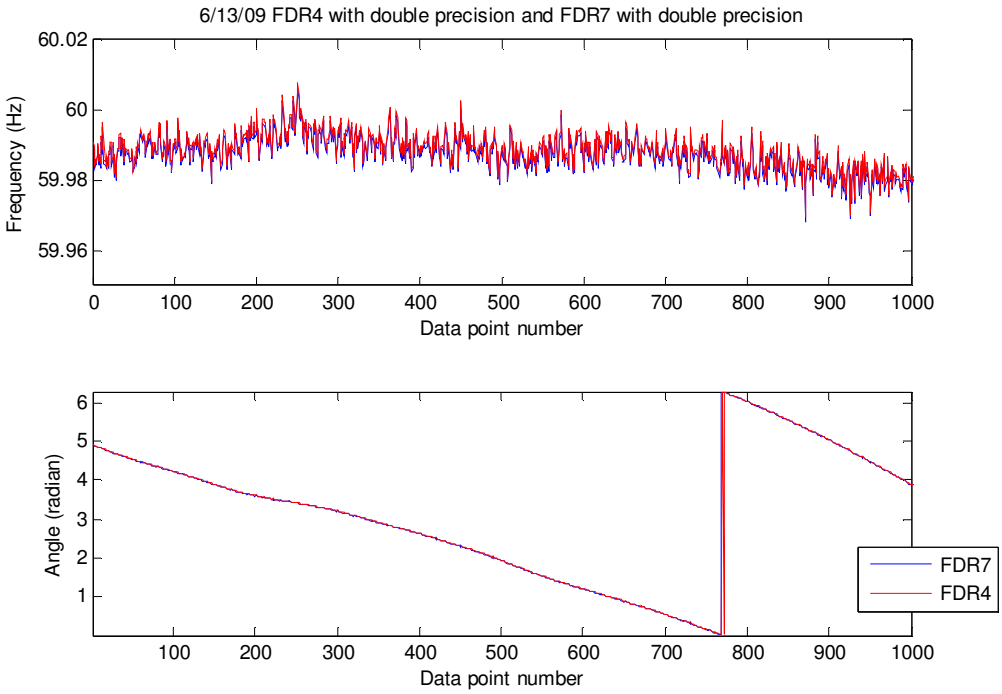

Figure 2.3 Frequency and angle measurements using double precision arithmetic

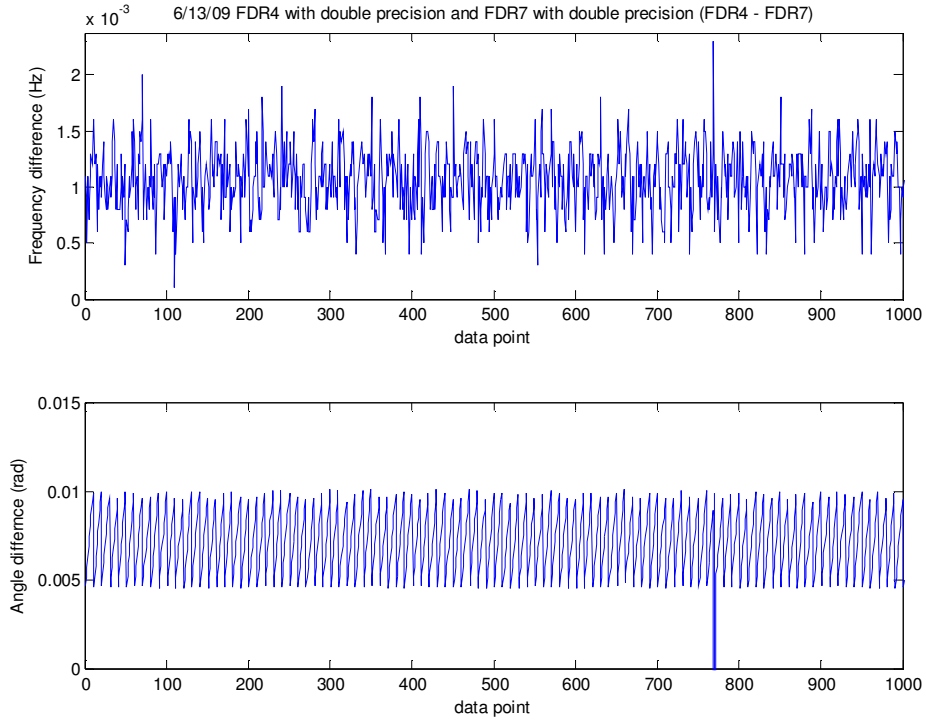

Figure 2.4 Frequency and angle measurement differences between two FDR units using double precision arithmetic

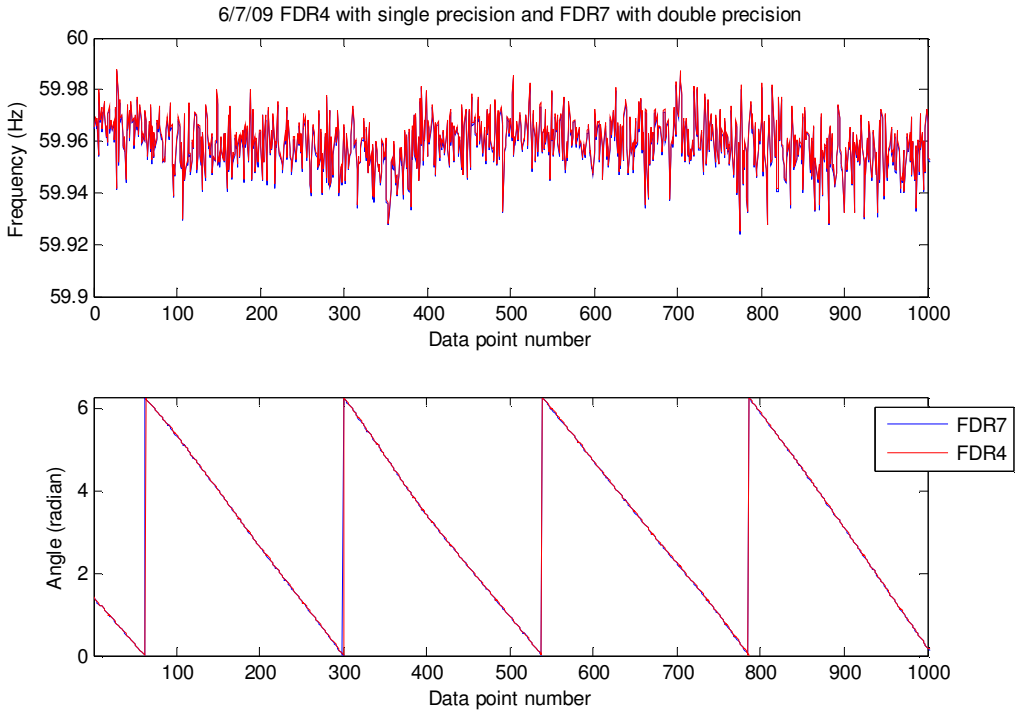

Figure 2.5 Frequency and angle measurements using single and double precision arithmetic

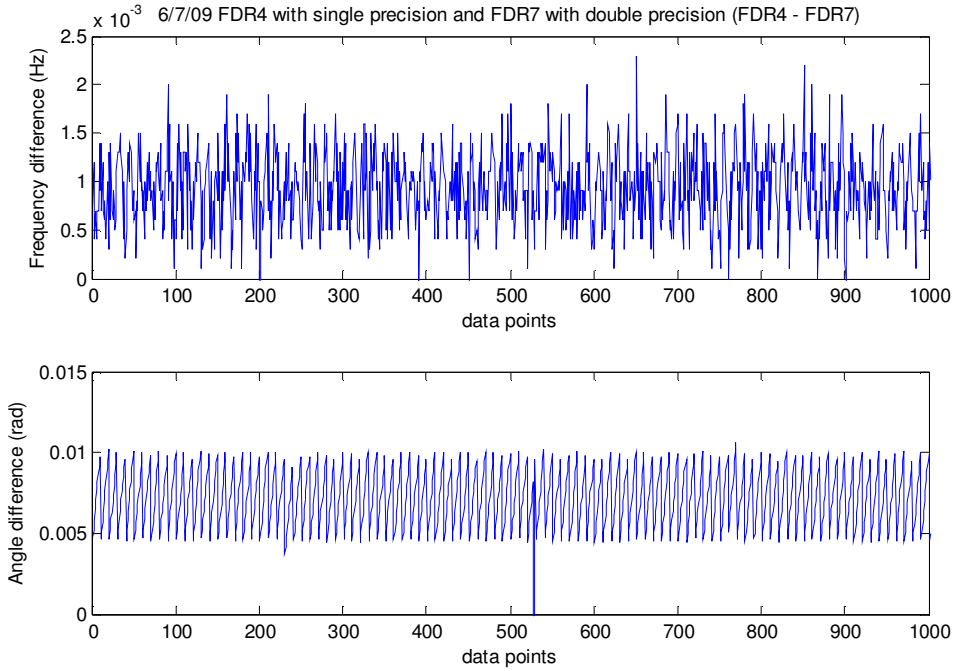

Figure 2.6 Frequency and angle measurement differences with single and double precision arithmetic

Table 2-2 Statistics of the differences in frequency measurements between FDR with double precision arithmetic and FDR with single precision arithmetic

	6/13/09 FDR4 – FDR7 freq.	6/7/09 FDR4 – FDR7 freq.
Minimum	0.0001	-0.0005
Maximum	0.0023	0.0024
Mean	0.00108	0.0009527
Median	0.0011	0.001
Standard deviation	0.0002826	0.0003576
Range	0.002	0.0029

Generally processor crystal frequency is one of the characteristics that have being emphasized in determining the processor speed. However, there is often a misconception that the processor crystal frequency is the main contributor in processor performance. The fact of the matter is that with the ever improving technologies in multi-processor architecture and the addition of peripherals and micros running directly off the crystal oscillator circuitry, the crystal frequency value becomes more and more meaningless.

In order to help designers to better evaluate processors, the Embedded Microprocessor Benchmark Consortium (EEMBC) has established many common benchmarks to easily quantify microprocessor performance with respect to each other. Nevertheless, considering the speed requirement of the FDR with 4 phasor angles computed per cycle, each new frequency has to be computed approximately every 0.00417 seconds. This means that the first phasor angle computation, the first Least Error Square fit, the resampling process, the second phasor angle computation and the second Least Error Square fit all have to be completed in approximately 0.00417 seconds. The initialization process can be ignored for the moment since it only has to be done once. A rough estimate of number of floating point operations in the algorithm is about 15000 operations per frequency estimation.

With 4 phasor angles computed per cycle, the processor needs to be able to perform at least 3.6 MFLOPS (million floating point operations per second) [3]. With today's fast processors, this requirement can be easily met. However, this requirement is rather misleading due to architecture differences of the processor. A study was conducted in [80] to evaluate the MFLOPS measurement in computer systems and the study indicates

that MFLOPS was a consistent measure of performance in computer systems up till 1984. For systems produced after 1985, MFLOPS became more dependent on architecture, configuration, and application so that a simplified model can be used to evaluate MFLOPS measure. Generally, MFLOPS measurement can be used more consistently for processors of the same architecture but the comparison becomes more complex across different architectures. Due to this reason alone, it is important to limit the architecture selection to a certain type and only then will the comparison in MFLOPS be valid.

There are two classification groups for the instruction set of processors. These include the CISC (complex instruction set computer) and the RISC (reduced instruction set computer). For the embedded version of FDR design, justifications can be made to use the RISC based processors. This is mainly due to considerations in simplicity, speed and short development cycle. The simplified instruction set has much emphasis on software whereas the complex instruction set puts its emphasis on hardware. As a result, the RISC based processor tends to spend more transistors on memory registers whereas the CISC based processor spends more transistors on storing complex instructions. As the price of memory chips decrease over the years, the RISC architecture has been gaining more attention in the embedded market. Additionally, the standard feature of pipelining in RISC processors allows for higher efficiency and faster processing speeds. Ultimately, with the exception of embedded processors, the boundary between RISC and CISC is becoming blurred in the recent years due to the fact that both architectures are evolving for the common goal of high performance computing.

Most of the processors today have some form of built in ROM (Read Only Memory) and RAM (Random Access Memory). An approximation of the amount of data memory needed can be approximated by the data size of the variables. With single precision representation, the amount of RAM is approximated to be 3120 bytes with the program memory estimated to be around 64000 bytes. However, additional memories are needed for either a high precision trigonometric look up table (LUT) or a math library for implementing the trigonometric functions. Specifically there needs to be at least functions for sine, cosine and arc tangent and the precision needs to be around 0.00001 radians [3].

To store a table with all of these values require a large amount of ROM. The alternative is to tradeoff speed against memory by approximate the trigonometric functions using methods such as polynomials approximation or Goertzels algorithm.

2.2.3 Timing Subsystem

Phase angle measurement accuracy is highly dependent on the sampling of local voltage angles. To provide high accuracy absolute time for synchronized sampling and timestamps for frequency and angle measurements, the FDR uses a GPS timing receiver in the first and second generation design. In order to synchronize the sampling of voltages to GPS timing, the processor uses the pulse width modulation (PWM) subsystem to generate the trigger for conversion signal for the ADC upon the rising edge of 1PPS. Such sampling scheme is used in both of the first two generations FDR.

The challenges to the design of the timing subsystem are two folds, one requirement being the need for a common and accurate timing reference and the other is related to the accuracy of the trigger for conversion signal. The IEEE synchrophasor standard states that the timing signal shall be accurate enough to keep the Total Vector Error (TVE) within the limits defined by the user-required compliance level. In connection to this it is important to note that an uncertainty of 1 microsecond on the synchronization signal leads to a phase angle error of 0.022/0.018 degrees for a 60/50Hz system [2]. In order to meet the 1% TVE requirement, assuming that the input voltage is free of noise, a maximum synchronization uncertainty of 26 microseconds is required for a 60Hz system and 31 microseconds for a 50Hz system (phase error of 0.57 degree causes 1% TVE) [2].

In order to obtain a common timing reference for the FDR acquisition process, it is essential to achieve an accurate synchronization of the sampling pulses. This requires the deployment of a timing source that may be internal or external to the FDR. In addition, the timing signal should provide enough information to provide the second-of-century in agreement with UTC. It must be available with minimum interruption at all measurement locations throughout the interconnected grid [17].

An obvious solution to synchronized sampling would be using a GPS disciplined oscillator with the nominal frequency in integer multiples of the sampling frequency. However, such a design is not practical as the oscillator will need to be changed when the sampling frequency is changed to a number that is not wholly divisible by the oscillator clock. In addition, restrictions are set to the oscillator frequency based on the sampling frequency. Past experience has shown that while using the internal processor oscillator clock to generate trigger for conversion signals is effective, it still has some intrinsic drawbacks in accuracy and flexibility.

At last, in assessing the performance of synchronization sources for FDR, it is important to consider factors such as accuracy, availability, continuity, reliability, integrity and the coverage [17].

2.2.4 Network Communication Subsystem

Internet connectivity is one of the basic requirements of FDR. Upon the first two generations of the FDR, the network communication system is based on the serial to Ethernet converter. Such a design is leveraged based on the popular serial communication interface, which is available on almost all of the processor architectures. Furthermore, the serial to Ethernet module allows for ease of integration and fast deployment. The module interfaces to the processor serial port via CTS/RTS handshaking and converts the data from the processor into a stream of TCP (Transmission Control Protocol) data that is transmitted over the Internet to the FNET server. For the transmission, TCP was selected as the transport mechanism because of its fault tolerant yet reliable transmission capabilities.

2.3 Limitations of the First and Second Generation FDR

While some enhancements were developed over the years since the first generation FDR, there are still a number of limitations to the second generation FDR design. Most importantly, the problem related to the timing subsystem and the processing capacity of the system is driving the need for an improved design. The following discussion will

highlight some of the improvements of the second generation FDR over the first generation, as well as the remaining limitations in the second generation FDR that will need to be resolved in the next generation design.

2.3.1 Timing Subsystem Limitations

The timing subsystems of the first and second generation FDR is one of the fundamental limitations to the accuracy of the phasor measurements. The issue is related to the method in which each second is divided into 1440 separate time periods. Since the 1440 trigger for conversion signals are driven by the processor PWM module, the period of the conversion signal is limited to the processor clock resolution and the accuracy of the clock division.

In the first generation FDR, the period of a sample is determined by dividing the number of clock cycles between the last two 1PPS input signals from the GPS by the desired sampling rate, 1440Hz. The integer result of this division is then used as the period of the PWM signal. The drawback to this method has to do with the remainder that is ignored after the integer division operation. The remainder, which may be as high as 1439 clock cycles, can introduce a great deal of error in the period of the last PWM pulse of the second. In addition, due to unavoidable variation in the system clock, a deviation of up to 100 clock cycles per 1 MHz of operation speed may have to be expected. This translates to 100PPM (parts per million) as it is commonly specified by oscillator manufacturers. As an example, with a sampling rate of 1440 Hz and an operating frequency of 20MHz in the first generation FDR, the division of 20MHz into 1440 yields 13888.88 clock cycles. However, since the PWM period register can only take integer values, only 13888 or 13889 can be used to approximate the number of clock cycles for each period.

In the actual software implementation the integer part of the theoretical number of clock cycles is used, or the value 13888. Such an approximation results in a residue of extra 1280 clock cycles in the last pulse period of the second. Translating to the time domain, 1280 clock cycles is 64 microseconds on the 20MHz clock. Such a figure may

seem insignificant but the residue accumulates each second leading to a 'snowball effect'. If left unresolved, the residue from each second would accumulate to the point where one sampling point could be lost or added. Figure 2.7 shows this phenomenon where the first sampling point of each second moves forward. After a certain amount of time, the 'snowball effect' drives the supposedly first sampling point after the 1PPS to move to the previous second. Since the algorithm only takes 1440 sampling points per second for phasor estimation, the extra sampling point is consequently discarded [14]. In effect, by unintentionally moving one sampling point, the whole sampling sequence is moved from one second to another. Hence the estimated angle data would have a periodic jump.

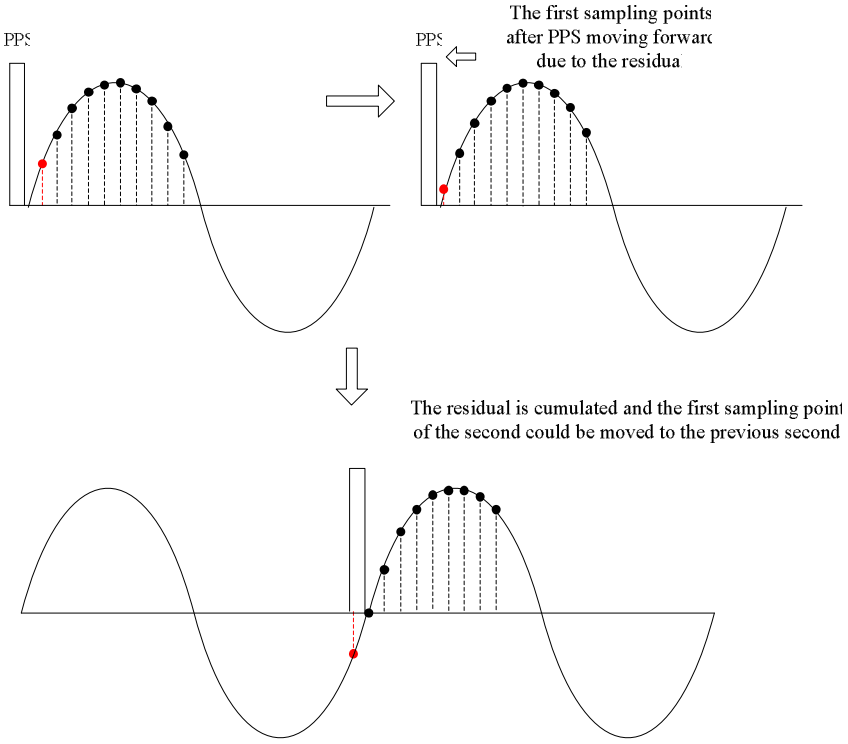

Figure 2.7 Effect of sampling point residues

To resolve the issue with the residue in the first generation FDR, a residue compensation approach was implemented for every second. The main aim of the implementation is to prevent the accumulation effect of the residue. To synchronize the sampling clock to the PPS and correct the residue every second, a fixed waiting time is defined at the beginning of every second as referenced by the rising edge of 1PPS [4]. The waiting time is defined as:

$$t_w = n_w t_{sys}$$ **Equation 2-2**

$$t_{sys} = \frac{1}{f_{sys}}$$ **Equation 2-3**

Where n_w is the number of system clock cycles to wait, t_{sys} is the period of the system clock in seconds. The waiting time t_w should be larger than the residue accumulated over one second but as small as possible to make the sampling clock start as close as possible to the rising edge of 1PPS. As a result, the residue will be confined to the second and will not accumulate over an interval of time. The drawback to this implementation is that the sampling interval between the last pulse of previous second and the first pulse of the current second is different from other sampling intervals. As shown in Figure 2.8, the sampling intervals of t_0 and t_1 are different. Since the algorithm uses computation window that encompass several fundamental frequency cycles, the sampling windows will introduce both t_0 and t_1 intervals. To counter this effect, the window size of the first phasor estimation was reduced from 8 cycles to 6 cycles to avoid the use of sampling interval t_1, which would otherwise introduce spikes in estimated frequency. The method

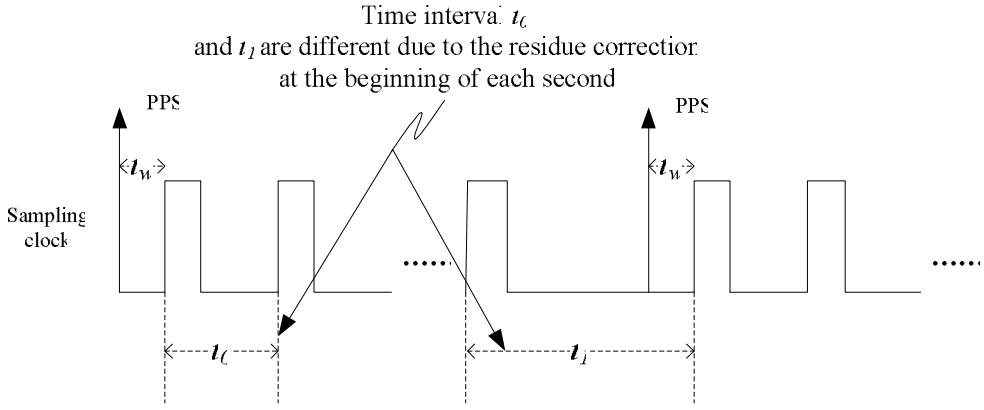

Figure 2.8 Effect of waiting a constant time for each sampling second

is effective in reducing the periodic phase jumps but the residue is not totally eliminated. Such phenomenon can be observed in the phase plot of measurement data as shown in Figure 2.9.

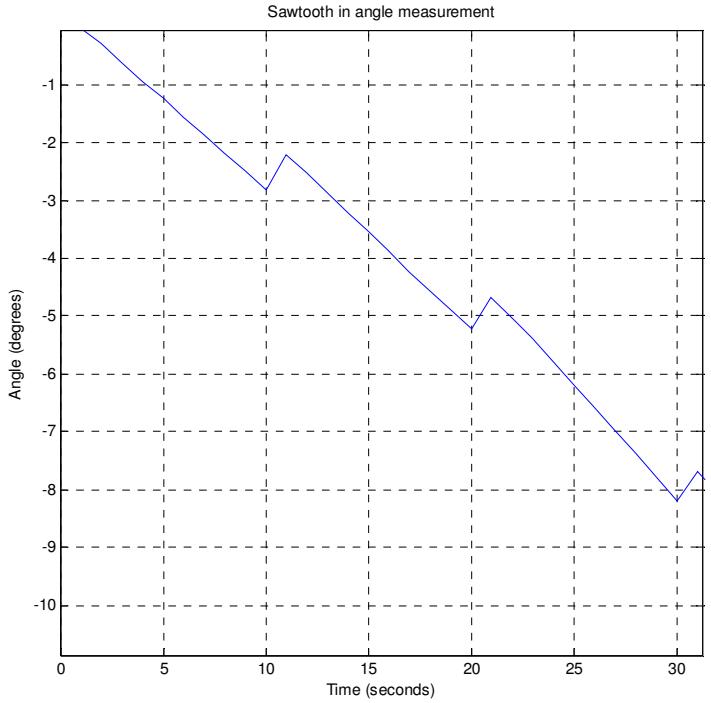

Figure 2.9 Phasor angle measurement from first generation FDR

In the second generation FDR, several enhancements were introduced to the timing subsystem. For one, the system clock frequency was increased from 20 MHz to 30 MHz thereby improving the timing resolution from 1/20MHz = 50 ns to 1/30MHz = 33.3ns. This improvement alone results in finer granularity of the timing division. However, a more significant improvement was introduced by matching the accuracy of the FDR timing subsystem to that of the PMU [14].

In order to achieve the sampling accuracy provided by the PMU, the FDR processor was setup to produce PWM with four different period lengths for the trigger for conversion signal. With a 30 MHz clock, the PWM period length is calculated to be 30MHz/1440 = 20833.3. With this mind, it is intuitive to use period lengths that are close to this value, such as 20832, 20833, 20834 and 20835. Figure 2.10 shows the phase error of the FDR with respect to that of PMU. Once the phase error is obtained with respect to the sampling

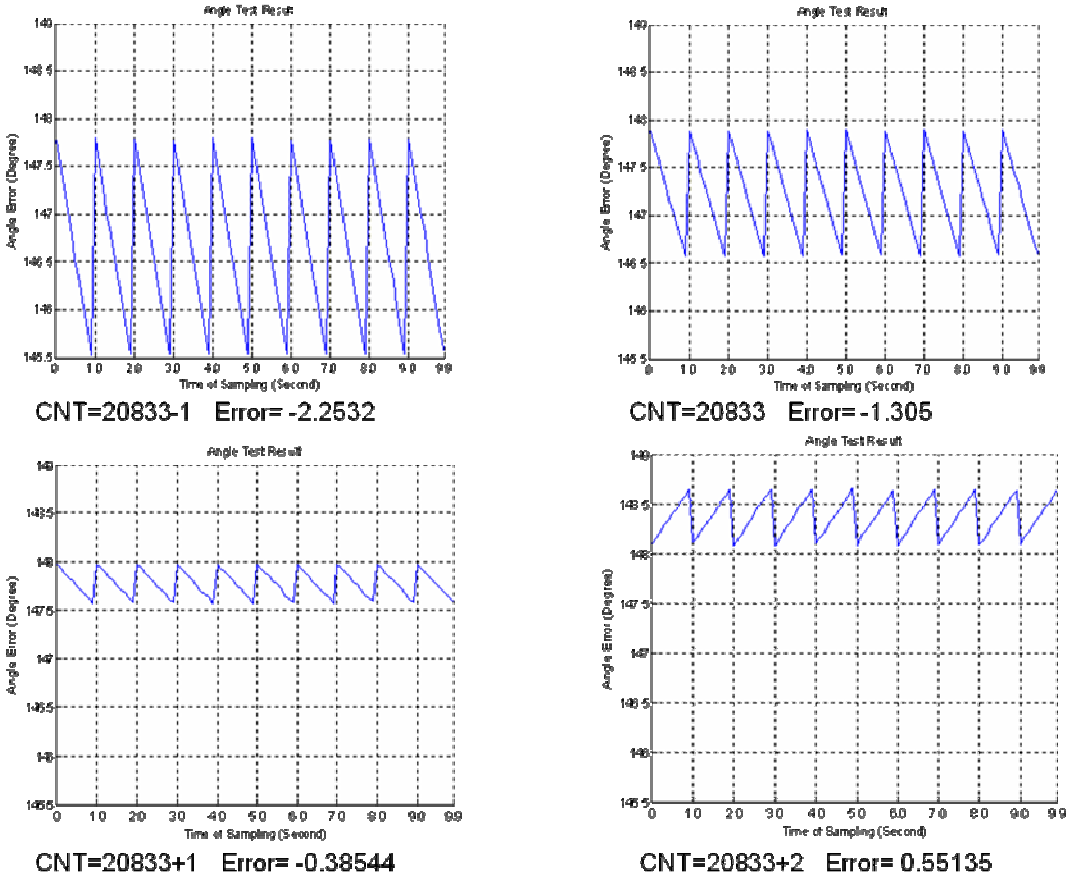

Figure 2.10 Effect of sampling period on phasor angle accuracy [14]

period length, it is easy to see that there is almost a linear trend of phase error with respect to the sampling period length. Given the four data points, linear regression can be applied to obtain an approximate linear model of the two parameters. Then the linear model can be used to estimate the period length which would minimize the phase error [14]. Specifically, the linear model is defined as:

$$N_{clock} - 10833 = AngleDriftRate \cdot p_1 + p_2$$ **Equation 2-4**

By using linear fit model, the four data points can be used to estimate p_1 and p_2, which are estimated to be 1.071 and 1.409 respectively. The AngleDriftRate is the actual phase error that results from the timing residue each second. Setting the AngleDriftRate to be 0, the parameter N_{clock} can be calculated to be 20834.409. As a result, when N_{clock} is set to be 20834.409, the angle drift rate will be approximately 0. To implement the fractional clock cycle of 0.409, the sampling period would add or subtract clock cycles so that the average sampling period is approximately 20834.409.

The second generation FDR timing subsystem has been dramatically improved since that of the first generation. The FDR frequency and angle measurement accuracy is enhanced by calibrating the DSP PWM period to match the phasor angle measurement with the PMU. However, the actual division of 30MHz by 1440 yields 20833.3 but the linear regression model estimated the length of the sampling interval should be 20834.4 to minimize the phase error that results from the timing residue. The inconsistency can be attributed to the hardware interrupt latency of the DSP and the frequency errors of the DSP oscillator.

Since the FDR timing subsystem is being calibrated against a PMU, the PMU is assumed to have higher accuracy with a superior timing subsystem. However, there are many different PMU manufacturers and models today and they may differ somewhat in the timing subsystem implementation. As an example, one PMU manufacturer may use a certain GPS receiver as the frequency reference but another manufacturer may use a different frequency reference. Therefore, the calibration of FDR timing subsystem to that of PMU may lead to inconsistencies across different FDR units.

A bigger drawback to such an implementation is the lack of cross platform compatibility and upgradeability. The next generation FDR may have a different processor or even a different architecture than the current one and such a modification would require a re-calibration of the timing subsystem. In addition, even if the hardware remains the same, a different sampling rate would also lead to re-calibration. By simply changing the sampling rate to 1200Hz to accommodate the power system with 50Hz nominal frequency, the timing subsystem would have to be re-calibrated. With the change in either the architecture or sampling frequency, the linear regression coefficients for estimating sampling period will be quite different. Consequently, the same calibration procedure will need to be applied against a PMU.

Another drawback of the second generation FDR timing subsystem has to do with the sampling period calculation. To this end, the original calculation of the period length is

based on a 30MHz clock. However, as it will be shown in Chapter 5, oscillators experience variations in frequency due to aging and temperature effects. In other words, the 30MHz frequency specified by the manufacturer is only the nominal frequency and it does not imply that the oscillator will always be 30MHz. Such deviation from the nominal frequency cannot be controlled and is part of the oscillator characteristic. As a result, the calculation of the PWM period is based on the assumption that the DSP oscillator will always be 30MHz, which leads to sampling time errors due to frequency variations of the oscillator. In addition, one could argue that the calibrated PWM period to be different when a different FDR is being calibrated against the PMU.

2.3.2 Computation Limitations

The first generation FDR was limited in computational capacity with almost all of its computational resources occupied in processing 1440 data points. The second generation FDR design improved in the system architecture but does not offer a significant increase in computation power. Furthermore, the second generation FDR split up the computation and I/O processing into two separate processors whereas the first generation used one processor for all of the I/O processing and computation. Such implementation offloads the I/O processing burden from one processor to another, which allows for more computational resources. Improvement can be made to the accuracy of frequency and angle measurement by including a scaling scheme where the sampling frequency can be increased to allow for oversampling. However, the requirement of oversampling will increase the computational requirements of the overall system by a factor of 2 to 10 times. The second generation FDR was not able to meet this goal due to emphasis in cost and portability. In the end, the oversampling will allow for the preprocessing of the digital sample data in order to produce more accurate input to the phasor algorithm.

2.3.3 Voltage Level and Communication Limitations

The next major limitation is related to the international capabilities of the FDR system. When the first generation FDR was developed, the criteria set out for the system was that it was to work on the United States power systems, therefore the system expects

a 60 Hz, 120 V analog input signal. As the interest in FNET applications has grown, a number of international groups have become interested in the research.

Power systems around the world run at a mix of 50 Hz and 60 Hz with nominal voltages that range from 120V to 240V. The first generation FDR design only addressed the need for a nominal frequency of 60 Hz for the sampling frequency and an input voltage of 120V. This limits the first generation FDR to be used in the US power system only. To accommodate the need for flexibility, the second generation FDR design incorporates a new transformer and filter design that's capable of accommodating both nominal voltage levels and frequencies. To switch between the two, a jumper is used in addition to overwriting the existing algorithm firmware to accommodate either 1440 or 1200 samples per second phasor estimation. This is an improvement to the first generation FDR where there are two separate transformer and filter design so that a 50Hz unit will not be able to switch to 60Hz and vice versa unless the hardware is changed.

The network interface of the first generation FDR is very much similar to that of the second generation. The only difference is that the second generation FDR uses an embedded version of the serial to Ethernet converter so that it is mounted onto the main PCB instead of operating as a separate unit. Such design is improved in terms of compactness and portability but does not gain any performance. There are times when the FDR communication interface is unreliable and a hard reset is required to put the unit back in operation. Problems have presented themselves related to the lack of reliability of the communications subsystems, and the lack of bidirectional data between the FNET server and the deployed FDRs. The serial to Ethernet converter is not robust enough to operate 24/7 without interruptions and that is an issue that needs to be dealt with in the next generation design.

Another factor that is limited by the communication subsystem of the FDRs is remote diagnostics. In both the first and second generation design, the deployed FDRs do not send any status information and have extremely limited remote diagnostic and repair capabilities. When problems occur, only elementary diagnostics can occur by phone or

email with the host at the site where the unit is deployed. The end result is that often users must send the physical FDR back to the FNET laboratory for evaluation. As the number of units deployed becomes greater, the number of problems will inevitably increase also. In addition, as more groups begin to rely on the information that FNET system provide them, the more they will desire faster recoveries from system faults and problems.

Chapter 3 Evaluation of Frequency Disturbance Recorder Architectures

3.1 Background

Over the years, the FDR has undergone numerous hardware changes in component upgrades. Ultimately, the overall architecture has remained the same, they are all consist of a transformer, a low pass filter, an ADC, a GPS, one or more embedded processors, interfacing board and a serial to Ethernet converter. In continuation of the FDR design requirements and limitations in Chapter 2, this chapter seeks to address the details of the existing FDR architectures as well as potential new FDR designs that deviate from the traditional architecture. Furthermore, the new design seeks to leverage state of the art technologies in field programmable gate arrays (FPGA) and common off the shelf (COTS) general purpose computers (PC) [5]. At the same time it is also necessary to explore upgrade possibilities for the existing embedded processor to allow for ease of upgradeability and minimize development time for deployment. Ultimately, the design requirements of the FDR should be satisfied and shortcomings of the existing FDR should be addressed and improvised in the new design.

Basically many options exist today for the FDR processor and the overall architectures can be divided into two large groups, the embedded design and general purpose computer (PC) based design. In an embedded system based design, there are many options available for the processor such as general purpose microcontrollers and digital signal processors (DSP). Nevertheless, the ever-growing market in system on a chip (SoC) technology is also being considered as a good candidate for FDR design. Features such as low cost, high reliability and flexibility make SoC very attractive for meeting the various requirements of FDR design. Due to its high flexibility and popularity, only programmable logic chips such as FPGAs are considered for integration into the FDR. Several architectures are presented based on the requirements and specifications indicated in Chapter 2, as well as the advantages and disadvantages each has to offer.

3.2 Microcontroller Based Design

The first generation FDR unit is based on a 32 bit microcontroller, the Freescale MPC555. The MPC555 was intended to be used most extensively in the automotive applications, such as engine control, transmission control, suspension and stability. In the past several years, the MPC555 is rapidly being integrated into applications such as avionics, controls, analysis equipment, robotics and power management. During the initial stages of the FDR design, much emphasis was put into the reliability, accuracy and portability of the device. Design optimization was not a crucial factor in the first generation hardware development. Since a proof of concept design was the main goal, an MPC555 evaluation board was used for connectivity and debugging capabilities. Figure 3.1 shows a block diagram of the first generation FDR and Figure 3.2 shows a photo of the first generation FDR.

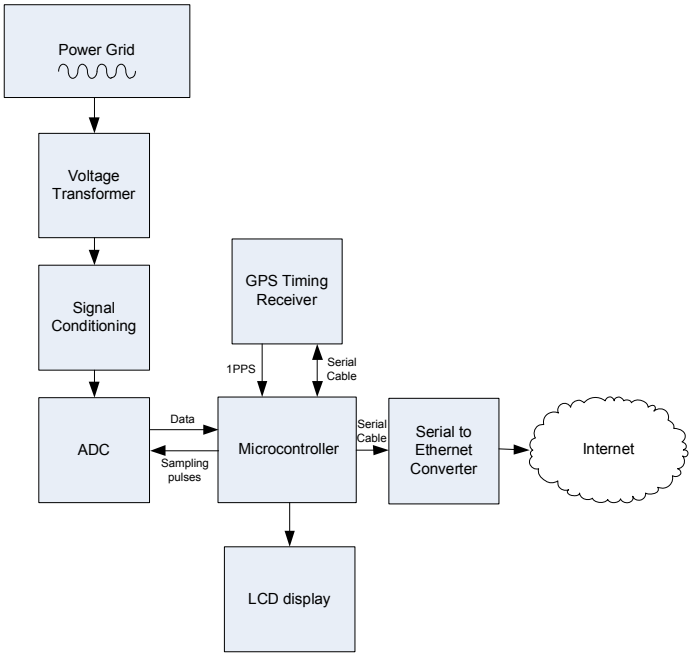

Figure 3.1 First generation FDR architecture

Figure 3.2 Photo of first generation FDR

As shown in Figure 3.1, the main computation unit, the Freescale MPC555 is located on the off-the-shelf evaluation board CME-555 from Axiom Manufacturing. The CME555 provides easy access to a number of peripherals on the main processor. In addition, the board has two memory sockets for memory expansion, two built-in RS232 connectors on board, two TouCAN ports with transceivers, a LCD connector with contrast, a 16 key keypad area and a JTAG development connector. However, most of the board modules are not being used for the FDR application and resulting in excessive cost, space and system complexity. Hence such design is not appropriate for high volume production and deployment.

Should the next generation design be based on the single microcontroller architecture, it is necessary to move from the evaluation board to a custom designed board. A custom designed board will not only be a more economical solution but also lowering the overall system complexity. By isolating the processor subsystems and the system peripherals that are needed for FDR, a custom board can be designed for the FDR [15].

3.2.1 Analog Subsystem

The signal conditioning system of the first generation FDR is composed of a two stage RC filter. Furthermore, the AD976A from Analog Device is being used for analog to digital conversion. The AD976A is a 16 bit SAR ADC with 200kSPS throughput and has a high speed parallel interface. Since the input range of the ADC is from -10V to +10VDC, a transformer is used to step down the 110VAC input from outlets. However, if the input voltage differs, a different transformer will need to be used, such is the case for 220VAC input. Improvements to the system can be made by re-design the transformer

and the filter to accommodate a wide range of voltage levels. Furthermore, the transformer, ADC and the filter should be integrated into one PCB.

3.2.2 Microcontroller

In the first generation FDR, all of the I/O processing and computation are done within the MPC555. The MPC555 is a high-speed 32 bit embedded microcontroller with up to 40MHz of clock speed and 448 Kbytes of embedded FLASH memory and 26 Kbytes of Static RAM. Moreover, the MPC555 contains a double precision floating point unit designed to meet the needs of high speed scientific computing. The MPC555 supports a wide range of on-board peripherals, these include the dual Time Processor Unit (TPU), two TouCAN Controller Area Network (CAN 2.0B) modules, and dual queued analog to digital converter (QADC). Of all these subsystems, only the TPU subsystem is being used for the FDR. The TPU is a microcontroller all on its own. Operating simultaneously with the CPU, the two TPU subsystems are capable of processing micro-instructions, schedules and real-time hardware events. In the FDR, one of the TPU channels performs the Request to Send (RTS)/Clear to Send (CTS) hardware handshaking to transmit measurement data to the serial to Ethernet converter. In addition, the other TPU channel provides the high accuracy timing for PWM generation, which triggers the external ADC for conversion. To allow for interfacing with external devices, the MPC555 has a plenty of I/O capabilities. The FDR design makes use of some of these I/O subsystems for interfacing with the GPS, the serial to Ethernet converter and the external ADC.

While the MPC555 meets all of the processing requirements of the FDR, the processor is heavily burdened in such a way that no oversampling or extended resampling can be implemented. Hence a trivial upgrade to the existing system is to develop a custom PCB with a more powerful processor of similar architecture [15]. Recent survey of processors similar to that of MPC555 is shown in Table 3-1 along with some of the important characteristics for FDR design. Since all of the processor instruction set is RISC based with some minimum differences in core architecture, comparison can be made directly in terms of Dhrystone MIPS (mega instructions per second). For example, the MPC5554 has about four times the processing power of MPC555. The ideal upgrade

for MPC555 is to make minimum change to the existing FDR architecture with the highest gain in processing power. With that concept in mind, it's easy to see that MPC5554 is the best choice for direct MPC555 upgrade. However, even with the high speed clock, there isn't any significant improvement in the speed of floating point operations. As a result, a direct upgrade of the MPC555 will not yield any significant gain in computation speed but only higher I/O processing speed.

Table 3-1 Options for direct upgrade of Freescale MPC555

Processor	MPC555	MPC566	MPC5554	MPC5200B
Core architecture	PowerPC	PowerPC	PowerPC e200z	PowerPC 603e
Maximum clock	40 MHz	56 MHz	132 MHz	400 MHz
Dhrystone MIPS	56 DMIPS	89 DMIPS	200 DMIPS	760 DMIPS
Instruction set	RISC	RISC	RISC	RISC
FPU	Single/Double Precision	Single/Double Precision	Single Precision	Single/Double Precision
UART	Yes	Yes	Yes	Yes
General purpose I/O	Yes	Yes	Yes	Yes
Interrupt controller	Yes (Hierarchal)	Yes (Hierarchal)	Yes (Hierarchal)	Yes (Hierarchal)
Timing module	Yes	Yes	Yes	Yes
Non-volatile memory	448 Kbytes EEPROM	1 Mbyte Flash	2 Mbyte Flash	None
RAM	26 Kbytes SRAM	36 Kbytes SRAM	64 Kbytes SRAM	16 Kbytes SRAM

In conclusion, the single microcontroller based FDR architecture has reached its limit in computation capabilities and a direct upgrade of the microcontroller will only provide faster I/O processing but not much gain in computation speed. Ultimately, it is not feasible to seek standalone general purpose microcontroller with high computation capabilities. For it is not optimized for any intensive signal processing capabilities. In

order to increase the computation capabilities to accommodate more sampling points, a new FDR architecture is needed.

3.2.3 Network Subsystem

The network subsystem of the first generation FDR is provided by the Moxa NPort 5110 serial to Ethernet converter. The module is simple to configure and easily integrated with the FDR. However, the module is not in an embedded form and is external to the FDR. Such a form factor makes relocation and deployment rather difficult. Also, the reliability of the device is not consistent as a system fault would stop data transmission.

3.3 Digital Signal Processor Based Design

The second generation of FDR design was realized more than four years ago and it has been showing some promising results. Instead of using a microprocessor as the central computation unit, the new generation implements a digital signal processor (DSP)-based computation unit. Most importantly, all of the hardware is integrated into a few PCBs, and the design has a switch that is able to switch between 110V and 220V input. Figure 3.3 shows a block diagram of the second generation FDR. As shown in Figure 3.4, this new generation of FDR has proven to be more portable and compact in design. In addition, the filter design is able to accommodate 60Hz as well as 50Hz input signals [16]. As a result, the second generation FDRs can be deployed on a global scale more conveniently.

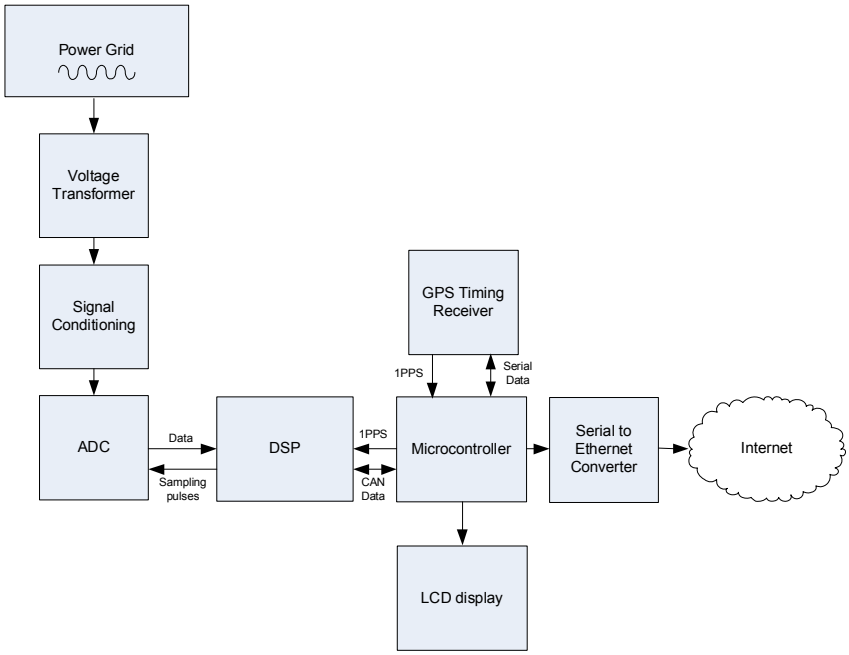

Figure 3.3 Second generation FDR architecture

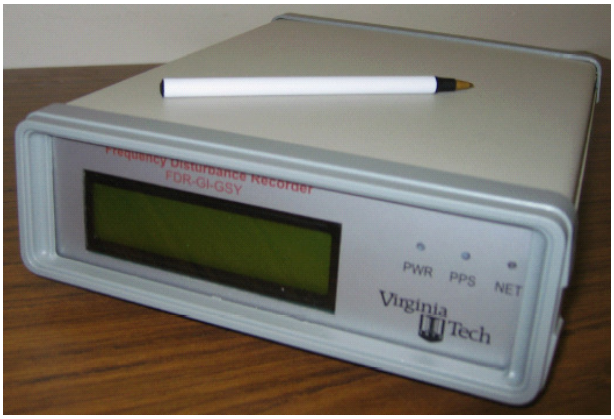

Figure 3.4 Photo of the second generation FDR

One of the most distinctive differences between the second generation architecture and that of the first generation is the use of a separate DSP to offload the algorithm burden from the microcontroller. In this case, the microcontroller is dedicated to the I/O processing functionalities of the FDR, thereby lowering the processing power requirement of the microcontroller. A less demanding microcontroller can be used in this case since all of the algorithm implementation is moved to the DSP. In addition, the use of a fixed point DSP has further lowered the overall cost of the FDR. Finally, the custom designed PCB for the second generation FDR eliminates the need for evaluation board from third party vendors. The lower cost implementation has drastically helped in terms

of high volume FDR production and deployment. Nevertheless, the computation power is still very much limited in the second generation design so that no improvement could be made in either sampling rate or multiple resampling. In order to accommodate higher computation capabilities, it is important to focus on the DSP since that is where all of the computation occurs.

3.3.1 Analog Subsystem

The second generation FDR has integrated the filter, transformer and the ADC into one PCB. Also, the transformer and filter are re-designed to accommodate both 120VAC and 22VAC inputs. The AD7865 from Analog Devices is used for analog to digital conversion. The AD7865 is a 14 bit SAR ADC with a fast conversion speed of 2.4 microseconds. Thus, the resolution of the ADC is lower compare to that of the first generation FDR at the expense of faster conversion speed and lower price. Nevertheless, the higher conversion speed is not useful unless the sampling speed is increased. Overall, the design of the new analog subsystem has resolved the issues associated with that of the first generation design.

3.3.2 Microcontroller and DSP Co-processor

The second generation FDR uses the C8051F020 from Silicon Laboratories as its auxiliary co-processor to perform several functions. The C8051F020 processor offers a fully integrated SoC design with subsystems including an ADC, two DACs, two UART serial interfaces implemented in hardware, five general purpose 16 bit timers and a programmable counter with five capture/compare modules. Furthermore, there are 64 kbytes of internal FLASH and 4352 bytes of on chip RAM memory to minimize external memory requirements.

The C8051 processor measures the 1PPS from GPS to determine whether it is valid and parses the GPS messages from the serial interface. It is also responsible for receiving the final phase measurement data from the DSP and combining it with the appropriate timestamp information into serial streams for the serial to Ethernet converter. Due to the

fact that CAN interface is not supported on the chip itself, an external CAN controller is needed to interface the C8051 with the DSP.

The main computation unit of the second generation FDR is the TI (Texas Instruments) TMS320LF2407. The TMS320LF2407 is part of the TMS320C2000 platform of fixed point DSPs. Operating at 30MHz with 30 MIPS performance, the TMS320LF2407 is mostly used in motor control and motion control applications. Moreover, the TMS320LF2407 offers an internal ADC, SCI (serial communication interface) module, SPI (serial peripheral interface) module, CAN communication module, two general purpose timers and eight PWM channels. The TI ANSI C compiler simplifies the software development by providing a run-time-support library that contains a custom coded set of floating point math functions. With the aid of the math library, the FDR algorithm implemented in C can be directly applied in the fixed point DSP with minimum modifications.

In addition to the algorithm implementation, the TMS320LF2407 is responsible for generating the trigger for conversion signal to the ADC. The event manager module receives the 1PPS signal from the GPS and generates a PWM signal to the ADC based on its internal 30MHz clock. In return, the ADC sends back the converted digital data upon the rising edge of the trigger for conversion signal. All of the I/O processing are interrupt driven to minimize latency.

The microcontroller and DSP co-processor architecture has a well balanced trade-off between cost and computation power. Nevertheless, if such architecture were to be used in the next generation FDR design, it is necessary to focus on the DSP hardware upgrade to obtain higher computation power. Considering that faster floating point computation is the major requisite for processing more data points, the obvious migration path would be switching over to the floating point DSP with up to 60MFLOPS and 80MFLOPS performance. Such transition would certainly increase the cost and complexity of the hardware. Nevertheless, the choice of a fixed or floating point DSP is no longer solely dependent on the complexity of the software development but rather the precision

requirement of the application. Since the current fixed point DSP is performing satisfactorily in frequency and angle measurement accuracy, the selection for upgrade should encompass both floating point and fixed point DSP. Ultimately, the fixed point DSP will always have an edge over the floating point DSP in FDR design.

3.3.3 Network Communication Subsystem

Similar to the first generation FDR, the network communication subsystem is provided by a serial to Ethernet converter. However, in this case, an embedded version of the device is being used. The Moxa NE4100T has a small form factor and is easily mounted to the processor board of the FDR. Since this is a new generation of the Moxa products, the performance of the device is better than that of the first generation FDR. Nevertheless, the performance is not consistent and the drop out in communication is one of the major causes of missing data. It is fair to say that in the next generation design, the network communication subsystem needs to be reconsidered with a more reliable replacement.

3.4 Commodity Personal Computer Based Design with FPGA

To address the need for flexibility in sampling speeds and high reliability, a PC based design was proposed in [5]. Overall, the architecture of a PC based design is a little different from that of the traditional embedded system architecture. Similar to the first and second generation FDR, the system will use a voltage transformer and low pass filter combination to convert the input voltage from the standard wall outlet power signal, and to condition the input voltage respectively. The same ADC board from the previous generation can be used. For the timing subsystem and the computation engine in the system there are a number of changes. In order to increase the timing accuracy and flexibility of the system, the GPS 1PPS signal will be input to a FPGA. This FPGA will monitor the incoming 1PPS signal and will send out accurate trigger for conversion pulses to ADC. The ADC will then assert an input to a smaller 16 bit microprocessor in order to indicate end of conversion. The FPGA will also indicate to the microprocessor

the start of each new second. The microprocessor will then gather the binary data from the ADC and GPS serial data stream and transmit to an embedded PC for computation and network communications. A block diagram of the FPGA based FDR is shown in Figure 3.5.

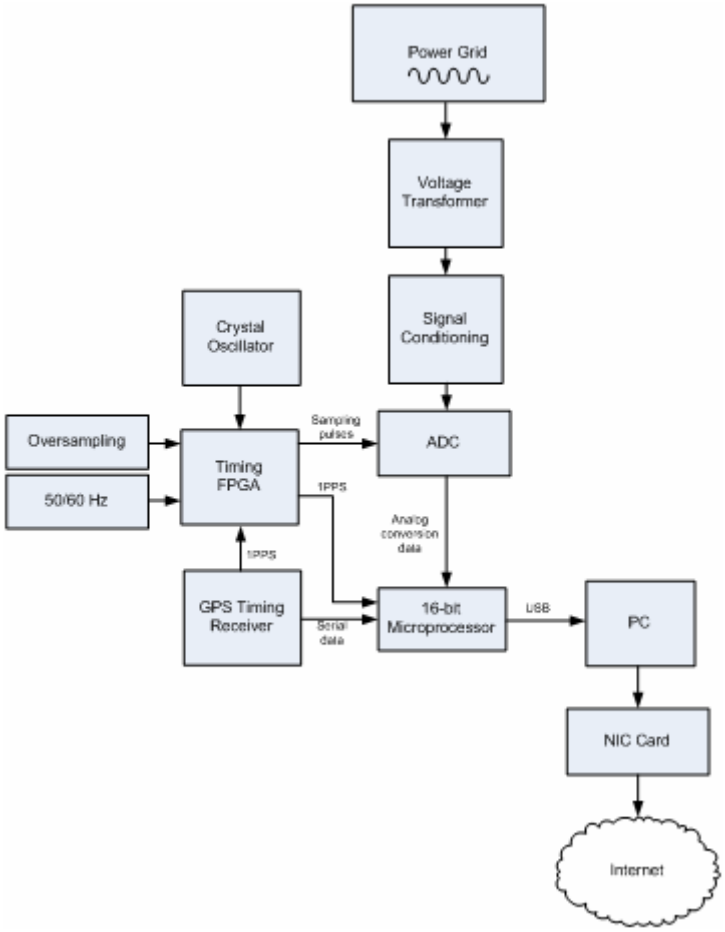

Figure 3.5 FDR architecture based on FPGA and PC

3.4.1 Microcontroller

The 16 bit embedded processor is responsible for all of the interfacing tasks in the system. These tasks include collecting all of the sampling data from the ADC, keep track of the sample number, as well as parsing the GPS serial data stream from the GPS timing receiver. Most importantly, the microcontroller is used to collect the ADC data and correlate that data with timestamps. In order to accomplish this, the microcontroller first receives the 1PPS signal from the GPS timing receiver and uses this pulse to reset an internal counter that tracks the sample number of ADC conversion. After receiving the

data from the ADC the microcontroller stores that sample along with the sample number in a buffer waiting to be sent to the computation system.

In addition to receiving data from the ADC, the microcontroller is also responsible for receiving and parsing the serial stream from the GPS timing receiver. The microcontroller uses the parsed time information to determine the actual second of the day that corresponds with the sample number that is maintained by the internal counter on the microcontroller. In order to transmit the data to the computation system, the microcontroller periodically packages the data that has been stored in the transmit buffer. The data are then sent to the computation system from a universal serial bus (USB) connection. There are a number of advantages in using USB interface. For one, nearly every PC today comes with USB standard interface. Also, the bandwidth of USB link far outperforms that of the RS232 serial bus, which was used in both earlier versions of the FDR. The throughput of the USB 2.0 is 480 megabits per second (Mbps), which surpasses that of the serial connection of merely 230 kilobits per second (kbps).

3.4.2 FPGA

In order to the address the need for a flexible timing subsystem that is capable of accommodating variable sampling rates, an FPGA is used as the main controller in the timing subsystem [5]. The advantage of using an FPGA is mainly due to its high performance in hard-real time, deterministic behavior. Through the reconfigurable nature of the FPGA, the 1PPS from the GPS can be accurately divided into sampling periods according to the sampling rate specified. The sampling rate can be selected via two separate on-board configuration jumper. Through the use of the jumpers the system can be configured for both input nominal frequencies of 50Hz and 60Hz. In addition, the jumpers allows for the selection of oversampling which increases the sampling rate to 10 times that of the original sampling rate. Ultimately, the final system design is to have an FPGA that took the 1PPS output from the GPS as an input and also reads the desired input signal frequency select line as well as an oversampling select line. The output will then produce a pulse train based on those input parameters.

Much of the software implementation in FPGA can reuse the logic from previous generation FDRs. The only difference is that the logic will need to be implemented in Verilog hardware description language (HDL) instead of C. Nevertheless, the FPGA hardware does not need to be highly sophisticated with high density logic cells and system gates. A relatively low-end FPGA such as those from the Xilinx Spartan II family of chips is adequate for accommodating the timing logic.

To address the need for remote diagnostic and upgrade, the microcontroller is used to configure the attached FPGA upon the basic initialization routines. Specifically, when the microcontroller starts up, there will be a basic initialization sequence that will get the operating mode of the system. The operating mode includes the input from switches related to the selection between 50 and 60Hz input frequency and 120V or 220V input voltage. After the initialization routines, the system will configure the attached FPGA. Using a stored image of the FPGA configuration file, the microcontroller will load the configuration on to the FPGA. This is an important feature as it will allow the FPGA portion of the system to be updated remotely. Furthermore, the microcontroller firmware can be changed by the use of a USB bootloader, which allows for the internal program memory to be reprogrammed directly using the USB connection. This feature will enable the remote updating of both the microcontroller and FPGA code [5].

3.4.3 Commodity PC

In order to step up a notch on the computation capabilities of the FDR, a PC based computation unit is necessary to meet the requirement of processing above 10 times the current sampling rate. Considering the low cost and abundance of hardware systems that are based on the Intel x86 architecture, the new system will be designed to take advantage of this hardware. It is the responsibility of the 16 bit microcontroller to control the data acquisition process and transmit the digital data to the x86 based PC. Considering the fact that commodity PCs currently have multi-gigahertz processors onboard with capabilities from a few to many hundreds MFLOPS, with possible giga FLOPS peaks achievable

with vector computation units (Single Instruction Multiple Data (SIMD)) that come for free with the recent processors, the computation needs of the system can be easily met.

In addition the high computation capabilities, the PC offers many other advantages to the FDR system. The abundant peripheral interfaces of the PC are very valuable to the FDR. Most of the PCs today come standard with interfaces such as USB and Ethernet and with custom serial and parallel ports. Current implementation makes full use of the USB and Ethernet ports but future applications may call for the use of either serial or parallel ports as well to support some legacy products.

In general, most of the commodity PCs today are bundled with an operating system and some security software. For the architecture described in Figure 3.6, the operating system could be either Microsoft Windows line of products or a standard Linux distribution. Both of these options come with standard firewalls and are provided with regular security updates.

One of the most important features of the PC based FDR is the high level configurability and upgradeability. What's more is that a remote based administration scheme can be used. Components such as computation algorithm software can be configured on the fly and remotely. Also, through the use of standard PC hardware, the upgrade cycle for more complex phasor measurement algorithm is inexpensive and very simple. Standard software development packages can be used to develop and deploy the algorithms [5].

3.5 Standalone Commodity Personal Computer Based Design

The idea of a PC based FDR design expands the possibilities in incorporating a large base of readily available hardware and software. For instance, most of the desktop PCs today come with open PCI slots for peripheral expansion and there a wide variety of PCI (peripheral component interconnect) based DAQ (data acquisition) cards available. With these considerations, it is logical to consider the possibility of a standalone commodity

PC based FDR. Figure 3.6 shows the architecture of a standalone PC based FDR. The main components of interest here are the PCI based DAQ card and the Internet based timing. Most of the DAQ cards today have 16 bit ADC capability and a minimum sampling rate of 200 kHz so that the FDR data conversion requirement can be easily met. In addition, timing synchronization over the Internet is a fairly matured technology and has been improving in accuracy and reliability over the recent years. Depending on the accuracy needed, there is a wide variety of software that seeks to synchronize the PC clock to that of an external time reference. Nevertheless, the topic of PC time synchronization is resorted to Chapter 7. Given the capability to synchronize to the UTC time with the acceptable accuracy, software triggers can be used to drive the DAQ for conversion.

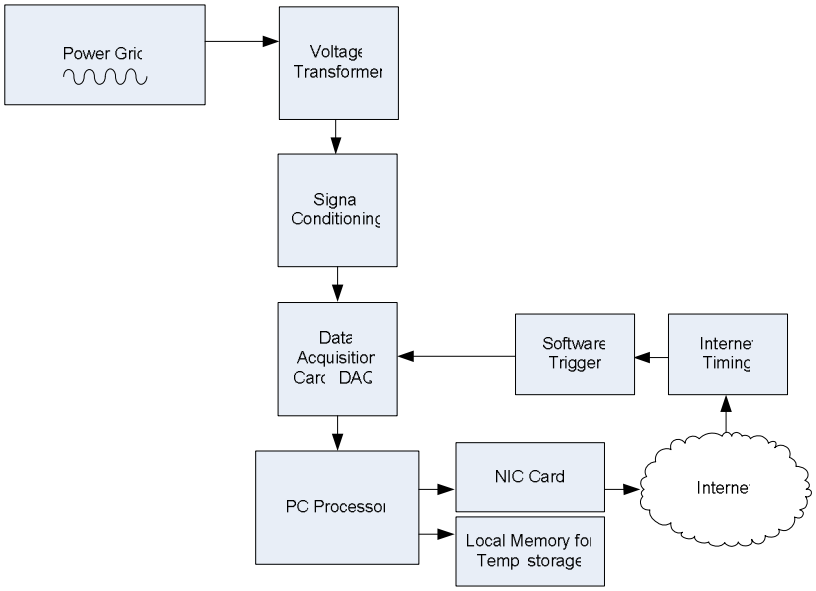

Figure 3.6 PC based FDR design block diagram

It is important to note that given the high flexibility in the PC based FDR design, there are several factors that can affect the accuracy of the frequency and angle measurements. Specifically, the performance of the system is highly dependent on the hardware and operating system used. Since the data acquisition module is being integrated into the PC, it is likely that a real-time operating system will be needed for better deterministic behavior versus that of the general purpose operating systems. Although Chapter 7 will present a more detailed description of the network synchronization technique and its performance, it is known that such synchronization

method will not provide as high accuracy timing as the GPS. With that taken into consideration, there are decisions to be made in regard to the applications that are suitable for the lower accuracy FDR. This is especially significant for applications that involve phasor angle measurements where there is very small leeway in the timing error. Nevertheless, the introduction of the PC based FDR raises the bar for large scale deployments and providing high density coverage in large geographical regions.

Chapter 4 Implementation of Global Positioning System as a High Availability, High Accuracy Timing Reference for Frequency Disturbance Recorder

4.1 Background

Global Positioning System (GPS) plays an important role in many different applications. They are best known to provide navigation and positioning solutions but more and more applications are using GPS as a time and frequency reference for synchronization. GPS provides very precise time as a by-product of their navigation metrology. For measurement applications GPS can provide accurate time and positioning for a single device. In addition, it can also be used for synchronization of multiple distributed measurement systems.

Specifically designed for high accuracy timing applications, there are some receiver manufacturers that optimize their products for timing applications. Such receivers include the Resolution T GPS timing module from Trimble, the Jupiter-T timing GPS receiver from Navman and the receiver implemented in the FDR, the M12+ GPS Timing Oncore module from Motorola. These receivers differs in that they have an over determined clock mode using position hold mode, which enables the coordinates of the antenna to be fixed in the solution leaving the clock offset only to solve for. Such a receiver will only need one satellite to provide a timing solution once the position solution has been established.

Timing Receiver Autonomous Integrity Monitoring (TRAIM) is supported on all of the timing GPS receivers. TRAIM is an algorithm that detects the integrity of 1PPS signal using the redundancy in measurements. The algorithm relies highly on the number of satellites being tracked and any abnormalities in the measurement can be detected and isolated. Specifically, there are two modes of operation where one mode uses the redundant measurements to alert the user if the solution does not meet specified accuracy

requirements [19]. The other mode is commonly known as the isolation mode where the degraded solution is improved by isolation and removal of faulty satellites. TRAIM is mostly used in the avionics industry where the subject of integrity is crucial to the industry's safety requirements [19].

This Chapter first explores how GPS obtains a timing solution based on time of arrival (TOA) concept and present some limitations of the technology. Alternative satellite navigation systems will also be presented. Furthermore, emphasis will be placed on the upgrade of the first generation FDR GPS receiver as well as a close examination of high sensitivity GPS implementation for high accuracy frequency and angle measurements. The high sensitivity GPS was introduced to address the issues associated with signal attenuation in certain environments. Ultimately, the high sensitivity GPS will not only provide high availability for frequency and phasor angle measurements but also maintain the same level of timing accuracy compared with the conventional GPS.

4.1.1 Global Positioning System

The NAVSTAR Global Positioning System (GPS) is made up of a constellation of more than 24 active satellites orbiting approximately 20,200 kilometers above the surface of the Earth. The constellation is oriented in such a way that at least four satellites are visible 24 hours a day all over the world [26]. Originally deployed by the United States Department of Defense (DoD), these satellites all have four onboard atomic clocks that synchronize to within 3 ns of the official atomic clock located at the Unite States Naval Observatory (USNO). Another section of GPS is situated on Earth to control the satellites. Within this section of the GPS there is a master control station (MCS) located at Schriever Air Force Base, Colorado Springs, CO. USA and five monitor stations distributed over the world. Data recorded by the monitor stations are processed at the master control station for calculation of satellite ephemerides and modeling of the satellite clocks. After doing the corresponding calculations, correction messages are sent to the satellites [26]. The purpose of these corrections is to maintain high accuracy of orbital location and synchronization of the satellite atomic clocks. The unique fact that the signals from these satellites are synchronized with the atomic clock allows end users

of the system to receive synchronized frequency information. Henceforth GPS is not only a navigation system, it is also a time-transfer system. As a time-transfer system it provides stability very close to one nanosecond per one day [27]. GPS is a versatile and global tool which can be used to both distribute time to unlimited number of users and synchronize clocks over large distances with high degree of precision and accuracy. This feature alone makes GPS a highly competitive timing synchronization source for wide area monitoring applications.

To obtain a navigation solution, the processor within the GPS receiver can calculate the difference between its on-board clock and either GPS time or Coordinated Universal Time (UTC) as determined at the U.S. Naval Observatory (USNO). In most cases the local GPS clock is a quartz crystal clock but there are some cases where an external clock such as a rubidium frequency standard or a cesium beam frequency standard acts as the local reference for the GPS receiver. The GPS receiver can be programmed to output UTC (USNO) or GPS time as calculated by the difference between the local receiver clock and GPS time. The UTC (USNO) is usually kept to within approximately 10 ns of actual UTC [27]. A GPS receiver tracking at least four GPS satellites can solve for the receiver's position and time at almost any location on the globe with high precision. A timing receiver operating from a given fixed location can derive time from GPS using just one satellite.

The exact nature of how GPS works can be obtained from a variety of literature such as [56]. Since the main aim of this study is focused on obtaining precise timing for frequency and angle measurements it is not necessary to delve into the algorithms of GPS location estimation. However, it is important to give an overview of the basic principles as well as how the timing solution is solved. GPS operates on five basic principles: (1) triangulation and Newton Raphson approximation methods are used to solve the position solution from four satellites; (2) GPS measures distance between orbiting satellites to a position on Earth by using the travel time of radio waves; (3) GPS requires very accurate clocks to measure radio wave travel time; (4) satellite position at any point in time is being kept track by DoD and the satellites relay an almanac with this information to the

receiver; (5) The radio wave experiences delays as it passes through the atmosphere with the most significant delays occurring in the ionosphere and troposphere.

In order to correct for errors on the GPS receiver oscillator clock, each satellite transmits a signal in the format of direct sequence spread spectrum (DSSS) on each of the two radio frequencies (L1 operating on 1575.42MHz and L2 operating on 1227.60MHz). In its most basic form, spread spectrum signaling consists of taking a given signal and spreading its bandwidth through multiplication by a high bandwidth spreading waveform. The effect of this waveform is to distribute the spectral power of the GPS signal over a range of frequencies. Such a distribution enables the 'jamming' of the signal more difficult as well as reduces unwanted interference of GPS signals with other terrestrial systems. Two distinct encodings are used: the coarse/acquisition (C/A) code (also known as Gold code or pseudorandom noise code) and the precise (P) code. The C/A code is open to public and used by civilian GPS receivers while the P code can only be used for military application.

As the signal reaches the GPS receiver, the receiver replicates the C/A code by generating a sequence of 1023 bits with a period of one millisecond and compares it with the received signal using trial and error. The offset between the replicated code and the received signal is the time of transmission and it will be multiplied by the speed of light yielding what is known as the pseudorange, or also known as the measured range. It is important to recognize that the range measurement just described only works properly if the satellite and receiver clocks are synchronized, hence the name pseudorange. To illustrate these concepts in terms of receiver clock offset (dt_{RX}), or the offset between the receiver clock and one satellite clock the following relationship can be obtained:

$$c \cdot dt_{RX} = p - \rho - d\rho - c \cdot dt_{SV} - d_{iono} - d_{tropo} - \varepsilon_p \qquad \textbf{Equation 4-1}$$

where c is the speed of light, p is the measured pseudorange, ρ is the true geometric range to the satellite, $d\rho$ is the satellite orbit error, dt_{SV} is the satellite clock error, d_{iono} is the ionospheric delay, d_{tropo} is the tropospheric delay and ε_p denotes multipath and noise. To correct for the receiver clock offset (dt_{RX}), a fourth range measurement is provided by the

fourth satellite which will not intersect at a single point. At which point the receiver will add or subtract time until it arrives at a solution that allows ranges from all satellites to go through one point. Finally, it determines the time offset and makes the necessary adjustments. The method that is being is described is really an overview of how the timing solution is obtained. However, the acquisition and tracking of satellite signals involves some more in-depth discussion on the GPS signal.

In order for a GPS receiver to acquire and track satellite signals, it must perform a two dimensional signal replication process of the received satellite signals. The type of GPS spread spectrum signal employed by the GPS is known as binary phase shift keying (BPSK DSSS). BPSK is a form of phase modulation where the carrier signal is instantaneously phase shifted by 180° at the time of bit change. The GPS receiver synchronizes or locks to the incoming satellite signal by using cross correlation with its locally generated code. When the two codes align they cancel one another, which allows for the acquisition of the satellite signal. However, in the case when more than one pseudorandom code is received, it is necessary distinguish them from each other. Gold codes are generated from the modulo-2 addition of two specifically chosen pseudorandom codes. Before the addition of the two codes, the second code is delayed a certain number of bits from the first code. To decode the incoming signal in the receiver, the correlator compares the Gold signal for each satellite with the receiver generate replica. Generally, the receiver searches all possible leads and lags between the signal and the replica to achieve the highest correlation. When this occurs, the receiver has acquire the signal can track it through control loops.

4.1.2 Limitations of the GPS Accuracy

Some error sources of GPS have been reduced throughout the years of ever improving technology. However there are some remaining factors that need to be taken into consideration while evaluating GPS accuracy. The GPS performance can be degraded by a number of factors and even inoperable under certain conditions.

The most well known error source of GPS was the Selective Availability (SA). This error was intentionally introduced by the DoD in order to make the GPS positioning inaccurate for civil use while preserving the higher accuracy for military use. S/A was put into effect on March 25th 1990 and was discontinued on May 1st 2000. While S/A was introduced in the GPS solution, it was the greatest source of error with up to 100 meters of inaccuracy (note that GPS distance error is directly proportional to timing error by a factor of speed of light). Such inaccuracies were introduced due to the intentional addition of dithering signal to the satellite clock. In addition, the satellite orbital parameters are incorrect so that accurate determination of satellite position in real-time was not possible.

Ephemeris errors are often introduced while the satellite paths are affected by forces like solar winds and earth gravitational pull. Although ephemeris messages are transmitted every 30 seconds, the messages itself may get delayed in the transmission process and providing a false update. What's more important is that the satellite path will be deviated away from the predicted, which will propagate into the receiver position solution. In addition, the satellite's atomic clocks also experience noise and drift errors. While these errors tend to be small, they can still add up to a few meters of inaccuracies.

Propagation errors are provoked by the fact that a GPS signal has to propagate through different layers of Earth's atmosphere. As an example, the ionosphere adds an inaccuracy of about 20 meters to the pseudoranges. Although not as significant of an impact, the troposphere also contributes some error to the GPS solution. These effects are minimized when the satellite is directly overhead but becomes greater for satellites near the horizon since the path through the atmosphere is longer. Furthermore, these effects can often be mitigated by modeling, although not perfectly. There will generally be some residual atmospheric error due to incorrect model or atmospheric disturbances. Using a dual frequency receiver or picking up pseudorange corrections from other reference stations are two ways to lessen the residual error.

A further important aspect affecting the precision of obtained GPS position is the geometry of the satellites used for pseudorange measurements. A parameter that is often used to quantify this effect is the Dilution of Precision (DOP) factor. Naturally the DOP value varies as the satellites are moving along their orbital planes, which means that the DOP values vary over time. DOP value can be calculated based on satellite orbital path and does not need any measurements. The accuracy of a GPS solution increases as the DOP value decreases with a threshold value of 5, indicating a value below 5 as the most accurate solution.

Although very well studied, multipath continues to contribute some errors in the GPS position solution. Multipath is the effect of wave signal reflecting off of different objects and arriving at the antenna, introducing noise in the received signal. Thus the receiver makes a pseudorange measurement based on an incorrect signal. However, the multipath errors are being kept to the minimum due to the high autocorrelation nature of the GPS signal, which tends to lessen the effect of signal reflections.

4.1.3 Alternative Global Navigation Satellite Systems

There are two globally used positioning systems, namely the Galileo and GLONASS, which are conceptually similar to the GPS. However both systems are yet to be fully operational due to various reasons.

Galileo is a joint initiative between the European Commission and the European Space Agency (ESA) [43]. It is a satellite navigation system similar to GPS developed in order to create a European satellite based navigation system independent of the GPS. Started in 2001, Galileo is still currently undergoing development phase. The final system will contain 30 Medium Earth Orbit (MEO) satellites in a height of 23,636 km above the Earth. The general system is composed of three main component groups, the global components, regional components and the local components. The global components are the satellites, regional and local components being the EGNOS (European Geostationary Navigation Overlay Service) [43]. The EGNOS allows for enhanced positioning needs by processing GPS or GLONASS signals resulting in an increase of accuracy of about five

meters. EGNOS consists of three geostationary satellites and a network of ground stations. Like the GPS, use of basic Galileo services will be free and open to everyone but the high accuracy capabilities will be resorted to military use and paying commercial users. Even though the Galileo system development has had its share of controversies, the EU transportation ministers have reached an agreement that it should be operational by 2013 [43].

GLONASS can be considered to be the Russian counterpart to the GPS. GLONASS was developed rather quickly resulting in operational capability with 24 satellites in 1995 [44]. However due to shortage in government funding the satellite constellation was degraded until 1998. Ever since then Russia strived to rebuild the satellite infrastructure and as of January 12th, 2010 the GLONASS system consists of 22 satellites of which 16 are operational. The system requires 18 satellites for continuous navigation services covering the Russian Federation and 24 satellites to provide services worldwide. The Russian government has estimated 2011 as the completion date of all 24 satellites in operation [44].

4.2 GPS Time Synchronization for FDR

4.2.1 First Generation FDR GPS Receiver

In the first generation FDR, the Motorola M12+ GPS timing receiver was used as the frequency and timing reference. The Motorola M12+ timing receiver is specifically designed for high precision timing applications with the highest claimed accuracy of less than 2 nanoseconds offset from UTC at the one sigma level. The receiver uses a parallel architecture with 12 channel L1 C/A code. Two communication protocols are supported, NMEA (National Marine Electronics Association) and the Motorola binary protocol that includes commands for output PPS control, elevation angle selection and ASCII message output. The evaluation board for the M12+ allows for simplified interface with the CME555 board. There are two serial ports used separately for GPS and AGPS (Assisted GPS) message transmission, a SMA (SubMiniature version A) connector allows for high precision 1PPS output. The MPC555 processor sends initialization commands through

one of the serial ports to initialize the M12+ receiver during system start up. The initialization includes configuring the 1PPS output to be enabled when tracking at least one satellite and a satellite mask angle of 3 degrees. Given that only one satellite is needed for precise timing once the position solution is established, there will always be timing correction provided by at least one satellite to the receiver. Furthermore, the configuration of satellite mask angle eliminates the satellites near the horizon from the GPS solution since the signals from those satellites are more susceptible to noise as they travel through the troposphere and ionosphere. Ultimately, such configuration is intended to increase the reliability and integrity of the 1PPS signal.

4.2.2 Second Generation FDR GPS Receiver

In the end of the year 2005 Motorola discontinued the M12+ receiver product line. There was a need for a new GPS receiver with equivalent performance and functionalities. Several GPS receivers were evaluated with the i-Lotus M12M topping the list in backward compatibility with the M12+ [45]. Specifically, the M12M receiver is very much similar to the M12+ receiver with the following exceptions [47] [49]:

- The M12M receiver PCB is made from slightly thicker (0.020") material so that the resultant thickness is approximately 0.063".
- The upper power supply voltage limit has been raised to +3.3V for M12M to simplify interface with +3.3V systems.
- The acceptable external gain range from the antenna system is much wider with the M12M than it was with the M12+. Whereas the M12+ required external gain limited to a range of +18dB to 36dB, the M12M front end can handle input gains from +10dB to +50dB, making antenna system design less demanding.
- Typical Cold-Start time for the timing receiver has been reduced from 200 seconds to 150 seconds.
- The M12M GPS receiver has a temperature controlled oscillator (TCXO) operating at 16.367MHz instead of the crystal oscillator (XO) used in the M12+.
- Timing accuracy is similar to that of M12+ with less than 2 ns within 1 sigma interval and less than 6 ns within 6 sigma interval.

The block diagram in Figure 4.1 shows the hardware interface for M12M GPS receiver, the MPC555 processor and the ADC. The GPS messages are transmitted via DB9 crossover cable and are driven by null modem interface where the transmit line and receive line are crossed and request to send (RTS) line and clear to send (CTS) line are crossed.

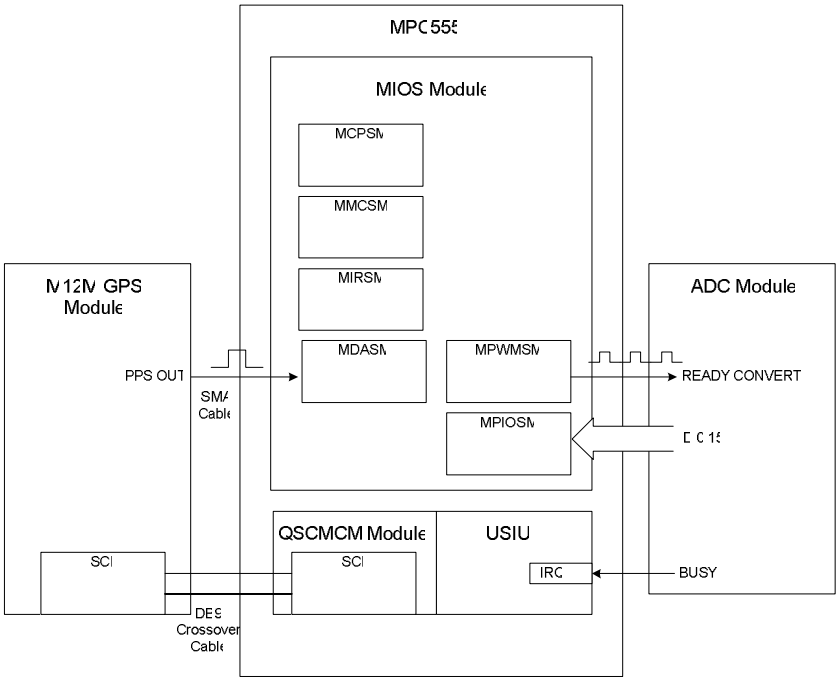

Figure 4.1 M12M GPS module interface to the MPC555 and ADC

To validate the timing accuracy, testing was conducted using M12M GPS receiver as the timing source for FDR. Figure 4.2 shows the test setup to compare the frequency and angle data collected by FDRs using different GPS. FDR4 is synchronized against the M12+ GPS receiver and FDR24 is synchronized against the M12M GPS receiver. Figure 4.3 shows that the frequency and angle data obtained from FDR24 is comparable to that of the FDR4. In addition, the same initialization sequence used in the M12+ can also be applied to the M12M to maintain the integrity and reliability of the 1PPS signal.

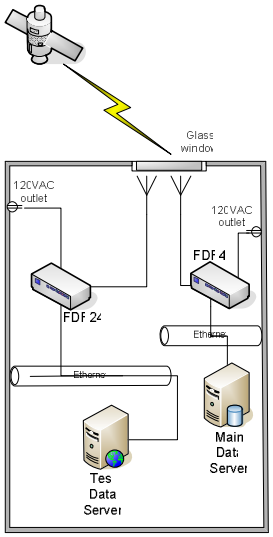

Figure 4.2 Test setup for comparing M12+ and M12M

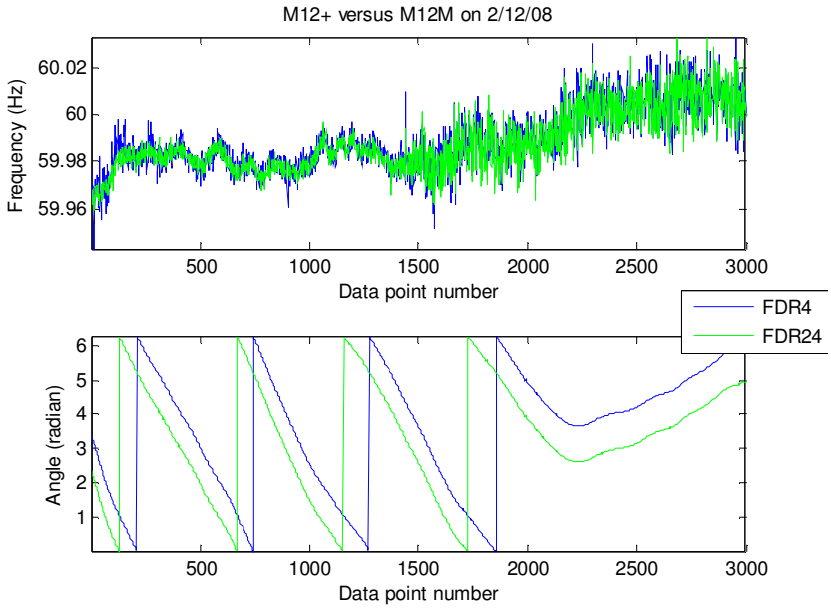

Figure 4.3 FDR4 with M12+ versus FDR24 with M12M

4.3 Introduction to High Sensitivity GPS

To allow for higher availability of timing source and more possibilities in FDR placements, the option of replacing the conventional GPS with a high sensitivity or indoor GPS receiver was proposed. The main drawback with the use of conventional GPS is the lack of availability in signal degradation environments. As an example, some of the newer office buildings have low-emittance (low-e) or also known as energy saving

windows. These windows uses a very thin metallic oxide coating to block the electromagnetic radiation, thus rejecting the heat outside of a building during the summer and keeping the heat inside during the winter. From a radio wave point of view, the electromagnetic radiation in the microwave region is blocked and the effect becomes more significant as the frequency increases. For a commercial low-e window, the metallic oxide coating provides 20dB to 35dB of signal attenuation in the frequency range 1-2 GHz. Consequently, FDRs placed in these locations are prohibited from acquiring any GPS satellite signals. To this end, there is a need to increase the availability of the conventional GPS and maintain the same level of accuracy for FDR time synchronization.

Several high sensitivity GPS receivers were evaluated for implementation in the FDR and only very few were considered due to its inherent high prices. The implementation of high sensitivity or simply indoor GPS presents a major challenge due to the significant signal degradation in indoor environments. The critical parameter in determining signal acquisition and tracking performance is not the absolute signal strength but the signal to noise ratio (SNR) and more specifically the SNR in a 1 Hz bandwidth, or the carrier to noise power density ratio (C/N_o) [31]. However, since the noise power is largely dependent on the equipment being used and supposedly remains constant for any given equipment, the C/N_o measurement is a rather accurate indication of the actual signal strength. In other words, the C/N_o figure is a measure of the signal quality present at the input to a GPS receiver [31]. According to the latest version of IS-GPS-200 specification [54], the GPS C/A code signal at L1 is designed to arrive on the ground at not less than -158.5dBW power level (refer to Appendix A for more information on GPS signal strength calculations). Conservative models suggest that the attenuation in buildings can reach levels of 2.9 dB per meter of structure [32]. Experiments indicate that the attenuation of the GPS signal through the buildings is typically higher than 1 dB per meter of structure [32]. Therefore, to track the GPS signals indoors inside high buildings the GPS receiver needs to be able to track signals with power levels ranging from approximately -160dBW to -200dBW [32]. To convert the figure into C/N_o values, signal strength of -160dBW corresponds to a C/N_o of about 41 to 44 dBHz.

In 1996, the Federal Communications Committee (FCC) mandated that wireless carriers provide Public Safety Answering Points (PSAPs) with precise location for all emergency calls from mobile phones. This system is known as Enhanced 911 (E911) system [32]. Such a system sparked the need for a GPS solution that will meet the requirements for indoor positioning, fast time to first fix and very low power consumption. Furthermore, the market demands have pushed the development of low-cost GPS receivers which should be able to operate in virtually any environment where also a cell phone works. The key to enhancing the sensitivity of GPS receiver is to extend the signal integration time significantly beyond the typical 2 to 5 milliseconds [22]. This effort requires several modifications to the receiver architecture. The high sensitivity receiver architecture differs from that of the typical receiver by having significantly more number of correlators and the ability to search all possible code delays in parallel [22]. To illustrate, a typical GPS receiver will often have hundreds or thousands of correlators whereas a high sensitivity GPS receiver would have tens of thousands of correlators. Ultimately, such implementation allows for longer integration time to acquire very weak signals and decreases the acquisition time when the signal is strong. High sensitivity GPS receivers are now commercially available at low cost from several manufacturers. Such receivers are claimed to be able to track and acquire signals with C/N_o values as low as 12 dBHz and thus be able to operate beneath heavy forests, in urban canyons and indoors. [30]

Studies conducted in [33] and [30] indicate that a high sensitivity receiver can indeed acquire significantly more satellites than the conventional GPS and is able to track satellites behind obstruction. Also, positioning availability is nearly 100% for the high sensitivity receiver. However, studies have also shown that there are drawbacks to the use of high sensitivity GPS. Specifically weak signals are usually not only attenuated but also delayed. Effects such as reflections and diffraction of the signal upon objects increase the traveling time of the signal, thus introducing errors in the range calculations [30]. Furthermore, the errors are mostly dependent on the angle of arrival of the signal and the properties of the material. Material thickness, reflectivity, index of refraction,

conductivity and absorption properties all affect the attenuation of the line of sight signal. Therefore, the signals additionally provided by a high sensitivity receiver are usually less accurate than those which can also be tracked with a conventional receiver [30]. The difference in quality needs to be taken into consideration when computing position and time solution. Several literatures go into details on some of the algorithms that are being used to estimate the positioning errors including [40], [41], and [30]. Nevertheless, the purpose of this study is not to delve into these algorithms but introducing the technology to synchronized frequency and angle measurements, especially for the FDR which aims for mass deployment in some environments where a direct line-of-sight to the satellites is not available. The following study takes a first step in evaluating the use of high sensitivity GPS for synchronized frequency and angle measurements.

4.4 Implementation of a High Sensitivity GPS for FDR

Initial evaluation focused on cost effective indoor timing GPS receiver with similar cost to that of M12+ GPS receiver. Table 4-1 lists the indoor GPS receivers with performance data as indicated in its respective datasheets [49][50][51][52][53]. The M12+ GPS receiver is also listed for comparison.

Table 4-1 Initialization characteristics of indoor GPS receivers and FDR GPS receiver

	Motorola M12+	QinetiQ Q20	NavSync CW12-TIM	SigNav TM3-02	ublox LEA-4T
Cold start	45s	45s	45s	60s	34s
Warm start	38s	38s	38s	48s	33s
Hot start	1s	1s	5s	6s	3.5s
Timing	15ns	15ns	30ns	10ns	15ns

A closer look at Table 4-1 indicates that the QinetiQ Q20 has the most competitive figures in terms of fast signal acquisition and accuracy in timing synchronization. However, the Q20 GPS receiver was still in the development stages at QinetiQ when it was first tested for FDR timing synchronization. Early test results have shown that the Q20 is not stable and would stop sending messages and goes idle after start up. Technical support at QinetiQ has indicated the problem may be associated with flaws in power

supply design which enables high current draw and puts the receiver at its thermal upper limit. The SigNav TM3-02 and ublox LEA-4T are good candidates for FDR implementation but their cost is significantly higher than that of the M12+. The NavSync CW12-TIM indoor GPS was then selected largely due to similar characteristics and cost compared with the legacy GPS receiver.

Similar to the Motorola M12+, the NavSync CW12 indoor GPS receiver module is specifically designed for use in synchronization and timing applications. Furthermore, the CW12 was designed to meet the form and functionality of the M12+ as closely as possible. Identical features include 12 channels L1 C/A code, antenna current sense circuitry, voltage input, optional on-board battery, time receiver autonomous integrity monitoring (T-RAIM), optional radio technical commission for maritime (RTCM) format input and data output format in NMEA 0183 or Motorola binary [49]. In addition to its high sensitivity feature of acquiring signals at -185dBW and tracking at -186dBW, the CW12 offers variable frequency output between 10Hz to 10MHz from its on board programmable NCO (numerically controlled oscillator). At the heart of the CW12 is the CW25-TIM module which can massively increase the number of correlators applied to each receiver channel with a maximum number of 12288 correlators. This enables the CW25 to allocate large number of correlators to each receiver channel to allow the receiver to search time/frequency bins in parallel rather than sequentially as in the traditional GPS receivers [49]. As a result, the receiver is specified to work in applications requiring indoor environments.

The migration from the M12+ or M12M to the CW12 is simple and low-cost. As it is shown in Table 4-2 [48][49][50] the physical and electrical characteristics of the three receivers are very similar thereby making the hardware integration fairly straightforward.

Table 4-2 Characteristics of GPS receivers - M12+, M12M and CW12

Characteristics	Motorola M12+	i-Lotus M12M	NavSync CW12
Dimensions	1.57”x2.36”x0.39”	1.57”x2.36”x0.53”	1.57”x2.36”x0.39”
Output Message on Serial Connector	Motorola binary/NMEA	Motorola binary/NMEA	Motorola binary/NMEA/ Proprietary ASCII/binary
TTFF- Hot/Warm/Cold Start	15s/40s/60s	<15s/40s/150s	5s/38s/45s
Power Requirements	2.85 to 3.15VDC, 50 mV max ripple	2.8 to 3.3VDC, 50 mV max ripple	3.3VDC
Connectors	10 pin header with 0.05” space	10 pin header with 0.05” space	10 pin header with 0.05” space
Timing Accuracy	<2 ns, 1 sigma <6 ns, 6 sigma	<2 ns, 1 sigma <6 ns, 6 sigma	<30 ns, 1 sigma
Weight	25 g	12.5 g	Not specified
Track signal below -180dBW	No	No	Yes

However modification was made to the FDR firmware in the parsing of timestamp messages. Even thorough it is not stated in the manufacturer specification, the CW12 outputs variable length messages each second whereas the M12+ and M12M outputs a constant 8 bytes of message each second. To illustrate this, the variable length messages can be shown below:

"@@Eq,03,23,08,20,22,05,37,13.8999,N,080,13.8999,W,+00567.7,121.9,177.2,0,0,0,03,0000,00,034\r\n”

"@@Eq,03,21,08,18,43,51,37,13.8838,N,080,13.8838,W,+00600.9,003.7,177.9,0,2,27.3,07,0000,00,008\r\n”

"@@Eq,03,23,08,20,22,31,37,13.9039,N,080,13.9039,W,+00567.1,038.3,001.6,0,1,999.9,04,0000,00,060\r\n”

"@@Eq,03,23,08,16,23,17,37,13.8827,N,080,13.8827,W,+00612.9,04246.5,160.2,0,2,23.1,07,0000,00,003\r\n"

Where the first message has a length of 8 bytes and the rest are longer in length. To accommodate the longer length messages, the queue buffer to store the message was increased in length from 96 to 98 characters. Also, the parsing function was modified so that it will be able to accept messages of different sizes and extract the timing information according to the comma separation instead of the character positions.

4.5 Availability and Accuracy Analysis

Since availability, signal quality and accuracy are of the utmost concern for GPS performance evaluation, it is necessary to examine how indoor GPS compares with the conventional GPS in this regard. As it was mentioned earlier, the C/N_o figure can be a relatively good indicator of the signal strength. Since both the M12M and the CW12 provide the C/N_o figure for all 12 channels as part of its output messages each second, these messages can be recorded and through post-processing, one can extract the C/N_o information. In addition, the DOP value and the number of acquired satellites will be examined to compare the accuracy and availability of the M12M and the CW12. Finally, the position solution in latitude and longitude is presented in Appendix A along with the necessary derivations for transforming from WGS 84 (World Geodetic System 1984) to ECEF (Earth-Centered Earth-Fixed) coordinate system. Generally, the timing accuracy of the GPS receiver is directly related to the positioning solution given that the position of the receiver is constant. Studies conducted in [55] indicate that the errors in the receiver position propagate to timing accuracy at the rate of 3 nanoseconds per meter of position errors. As a result, the performance of the CW12 receiver can be visualized by plotting the latitude and longitude solution and quantified by converting from positioning errors to timing errors.

Initially, some data were collected from both the CW12 and the M12M in the FNET laboratory. The antenna of both GPS receiver was placed next to the window and close to each other to simulate the effect of a common view of the satellites. Figure 4.4 shows the results from the CW12 and Figure 4.5 shows the results from the M12M. Clearly, the CW12 receiver was able to acquire a broad range of signal strength, specifically from around 10dBHz to almost 50dBHz. On the other hand, the M12M receiver was able to acquire satellite signal strength of around 19dBHz to a little above 50dBHz. It is interesting to note that the vast majority of the signals acquired by the M12M have C/N_o figures above 30 dBHz. Also, the M12M was able to acquire higher signal strength of around 55dBHz whereas the maximum acquired signal strength for the CW12 was less than 50dBHz. Finally, the availability of the satellite signals for the CW12 appears to supersede that of M12M. Figure 4.5 shows that the M12M receiver lost acquisition of all

satellites during a small period of time. As a result, the CW12 receiver provides higher availability of satellite signals than that of M12M. In comparison of the accuracy of the two GPS receivers, Figure 4.4 shows that the DOP value of the CW12 ranges from 0 to 100 whereas Figure 4.5 shows that the DOP value of the M12M ranges from 0 to 12. Such observations indicate that although the CW12 is capable of acquiring more satellites, the M12M can acquire satellites with better geometry. Nevertheless, Appendix A shows the corresponding position solutions given by the two GPS receivers to compare their performances with respect to each other. It can be seen that under nominal conditions where the antenna is placed next to the window, the CW12 is capable of providing comparable position solution to the M12M.

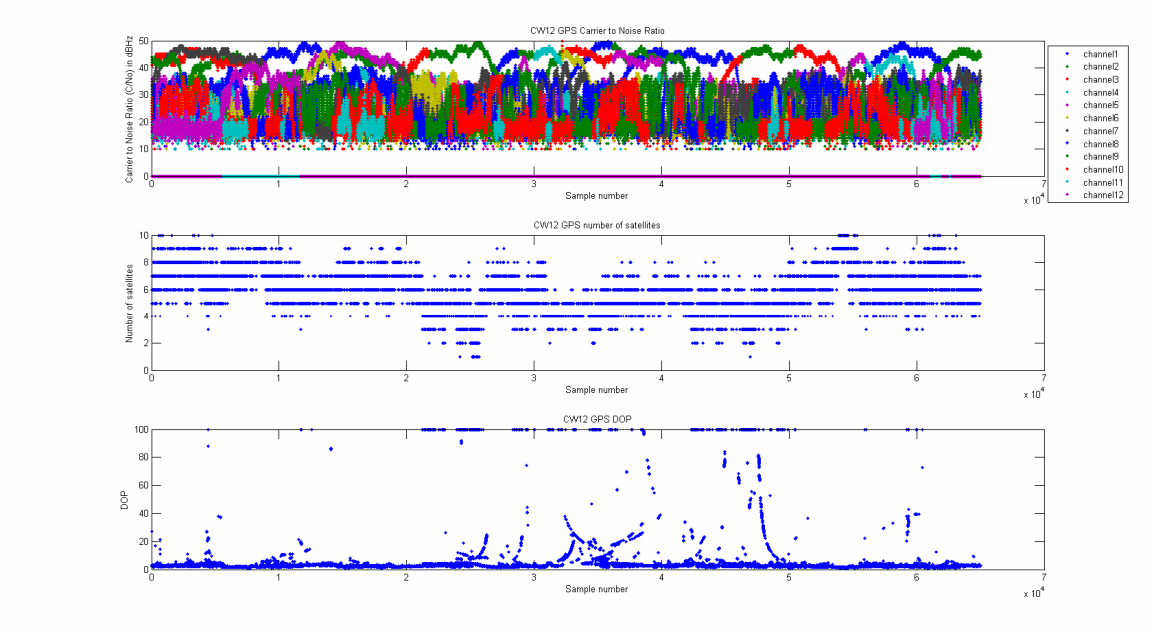

Figure 4.4 Indoor GPS receiver data with antenna placed next to window

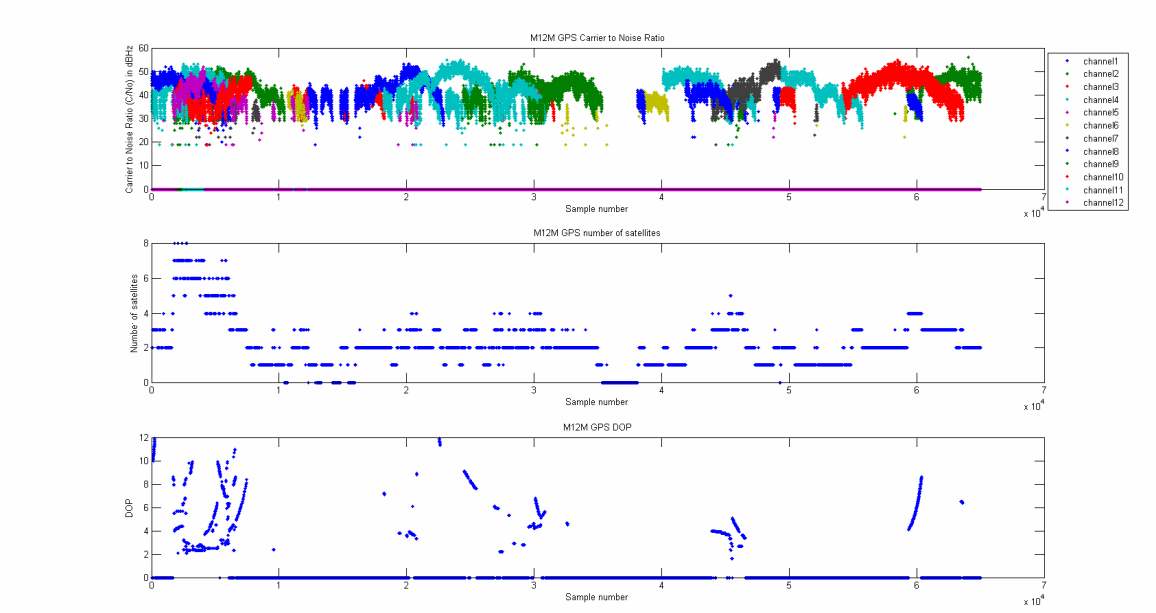

Figure 4.5 Conventional GPS receiver data with antenna placed next to the window

To examine the effect of signal degradation on the CW12, another set of data was collected from the CW12 with its antenna hidden away in a desk drawer. The collected data is shown in Figure 4.6. As it clearly indicates, the CW12 receiver was able to acquire satellite signals ranging from 10dBHz to around 35dBHz. In comparison with what is shown in Figure 4.4, there is a maximum of about 10dBHz attenuation when the antenna was inside the desk drawer. In addition, the corresponding number of acquired satellites is lower comparing with the data that was collected when the antenna was placed next to the window. The corresponding DOP value is also affected as it reaches 100 after some time and poor satellite geometry becomes significant. Finally, the position solution in latitude and longitude is shown in Appendix A. The results indicate that the accuracy of the position solution is degraded by more than 1000 meters when the antenna was placed in the desk drawer.

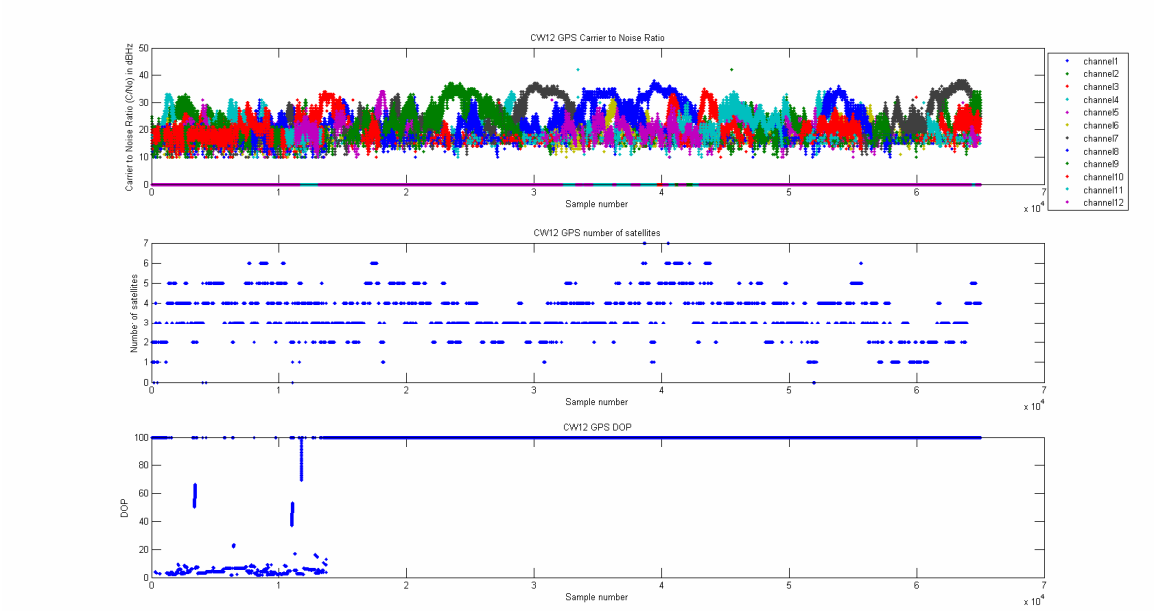

Figure 4.6 Indoor GPS receiver data with antenna placed in a desk drawer

To further illustrate the signal degradation effects on the CW12, Figure 4.7 shows the data collected while the CW12 antenna was moved into different locations. Initially, when the antenna was placed next to the window, the C/N_o figure is as high as 45dBHz and above. Then when the antenna was moved to inside the desk drawer next to the window, the signal degradation is clearly shown with more than 10dBHz of attenuation. At last, the antenna was moved to inside of a desk drawer that is far away from the window and again the C/N_o figure was reduced by as high as 20dBHz. Nevertheless, the receiver was still tracking at least one satellite with signal strength around 15dBHz to 25dBHz. The corresponding number of acquired satellites decreases when the signal degradation effect is increased. Similarly, the DOP value reaches beyond 5 after about 600 seconds, indicating that the GPS receiver will no longer capable of updating the position with reliable data.

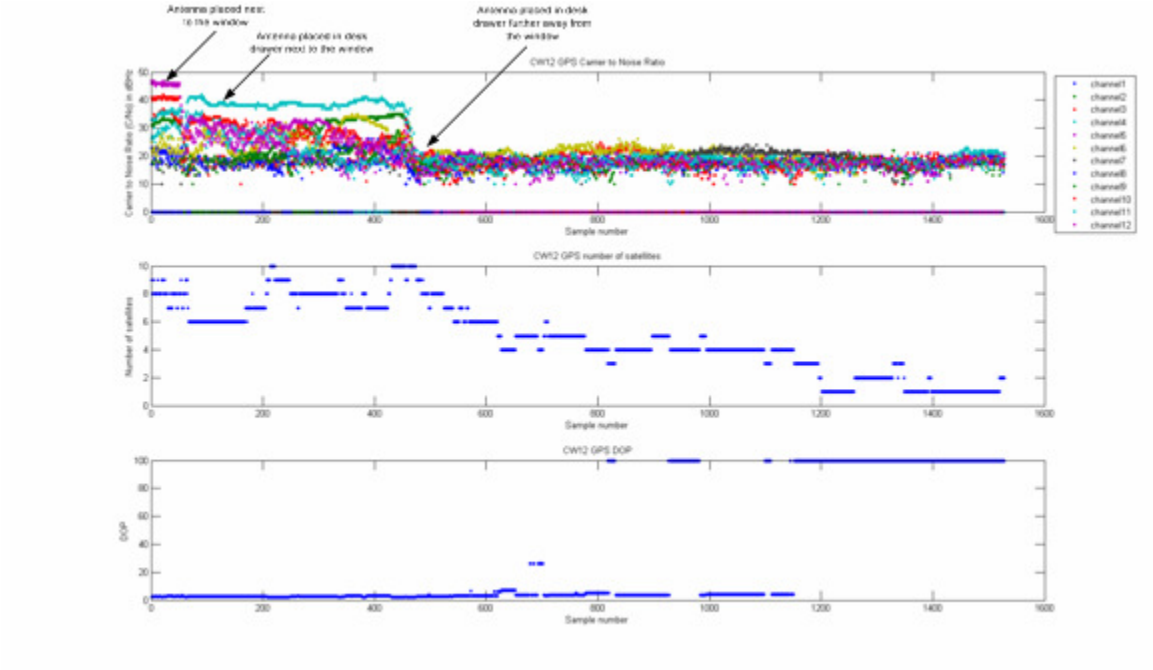

Figure 4.7 Indoor GPS receiver data with signal degradation

Upon close examination of the C/N_o for both the conventional GPS receiver and the high sensitivity GPS receiver, one can conclude that the high sensitivity GPS receiver is capable of acquiring signal strength with significant attenuation. In the case of the CW12 receiver, signal strength as low as 10dBHz can be acquired but consistent tracking of the satellites does not occur until the signal strength reaches above 15dBHz. On the other hand, conventional GPS receiver such as the M12M is only able to acquire signals as low as around 20dBHz and consistent tracking occurs at above 30dBHz. The results indicate that the CW12 receiver is behaving as expected in acquisition of signals with significant attenuation. In addition, it is capable of acquiring signals with much lower C/N_o figure than that of the M12M. However, it was also illustrated that when operating under significant signal attenuation, the position solution given by the CW12 is degraded. Such decrease in accuracy can be attributed to high noise level and multipath, leading to measurement faults ranging in magnitudes beyond 1 kilometer.

4.6 Frequency and Angle Measurements with Indoor GPS

To evaluate the performance of the FDR with the CW12 GPS receiver, initial measurement setup involved the use of an AC source for generating variable frequency and voltage waveforms. For this test the AC source was configured to output constant 60Hz, 120VAC waveform. Two FDRs were connected to the same AC source and connected via Ethernet to send data to a test server. The goal of this test was to see the accuracy of the FDR with indoor GPS under static frequency input. As a comparison, FDR unit 7 used the CW12 GPS receiver and FDR unit 24 used the M12M GPS receiver. To examine the performance of the indoor GPS in an environment with some signal degradation, the CW12 GPS antenna was placed next to the window initially to obtain a position solution and then placed inside a closed office desk drawer in the FNET laboratory. On the other hand, the M12M GPS antenna was placed next to a window in the FNET laboratory. A diagram of the test setup is shown in Figure 4.8. Some data were collected using the test setup, as it is shown in Figure 4.9. The frequency measurements from the two units match each other. This initial test has confirmed the 'indoor' capability of the CW12 but more long term measurement data are needed to understand its limitations in environments with significant signal degradation.

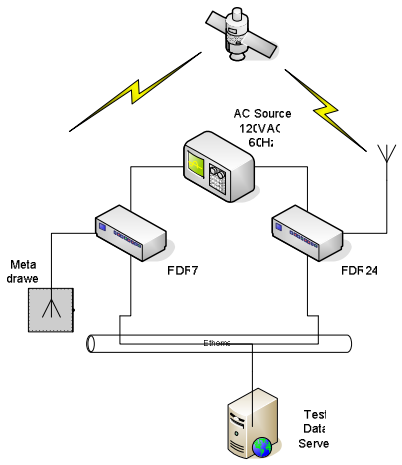

Figure 4.8 Indoor GPS test setup using the AC source

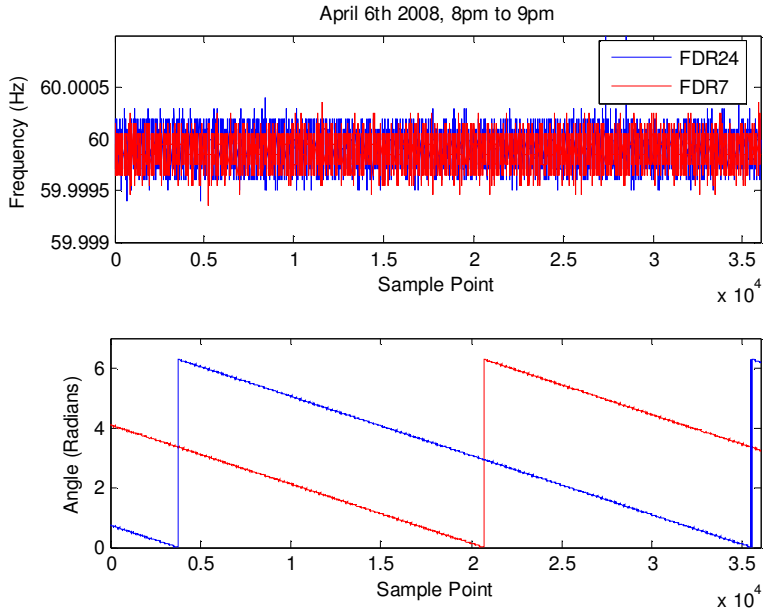

Figure 4.9 Frequency and phasor angle measurements of the AC source signal using indoor GPS

To push the envelope further on the CW12 indoor capabilities, another measurement trial was conducted for several days with the input signal from the wall outlet. The same FDR7 with the CW12 used in the initial test was moved to an office with no window and behind several walls to the nearest window. FDR24 with the M12M was used again for comparison at the same location and setup as the initial test. An additional unit, FDR4 with the M12M was added to this test setup as a backup to FDR24 for data comparison. Figure 4.7 shows the setup with the three FDRs. The measurement was initiated on April 29th and the CW12 did not acquire any satellites for the first few days. It wasn't until May 1st when FDR7 started acquiring signals from multiple satellites. Figure 4.11 to Figure 4.17 shows plots of one hour's worth of data (1AM to 2AM UTC time) collected from May 1st to May 7th. For each hour of frequency and angle data plot, there is a plot of corresponding number of acquired satellites along with a magnified plot of the first 3000 samples.

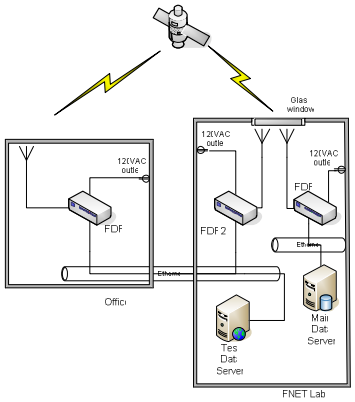

Figure 4.10 Indoor GPS test setup for completely isolated environment

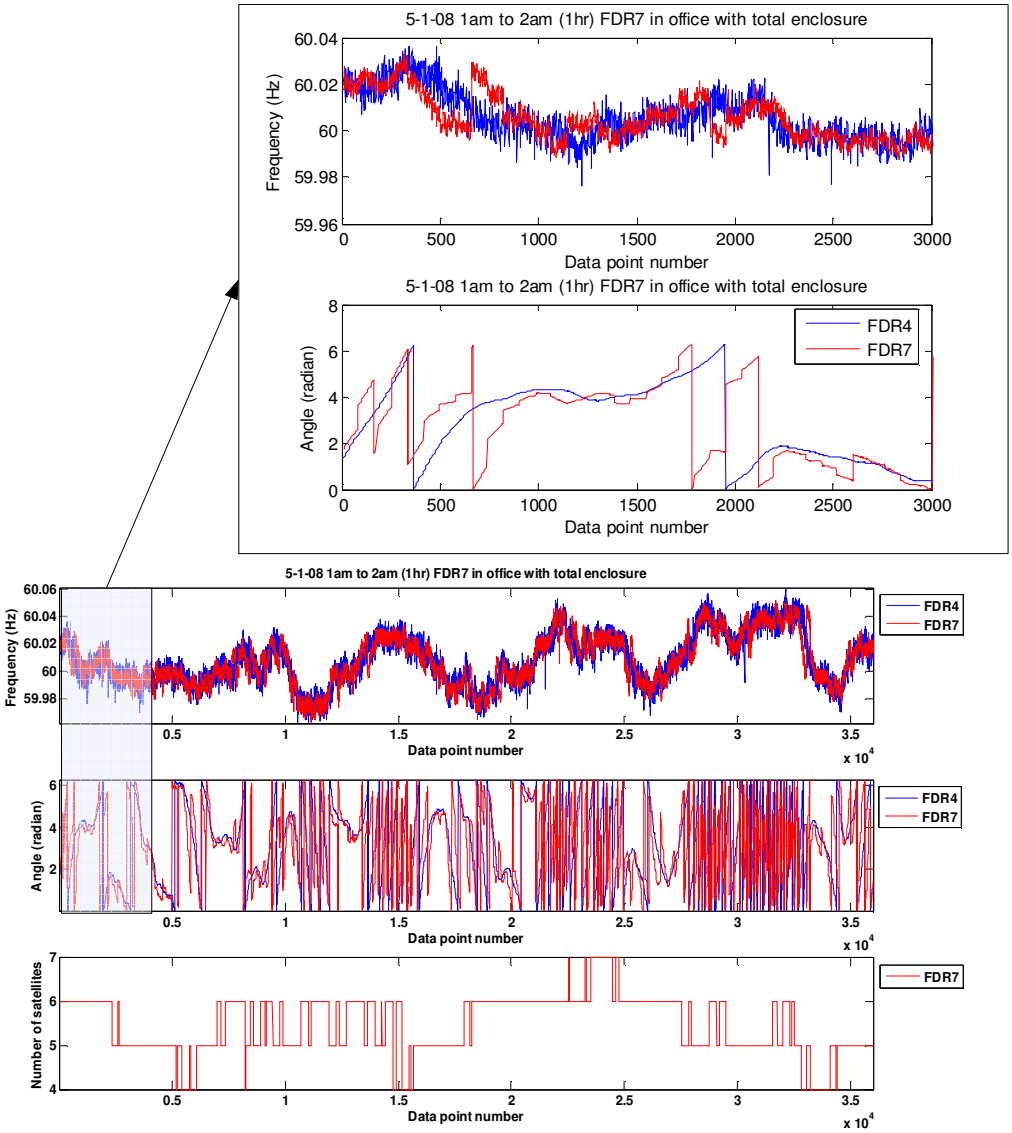

Figure 4.11 Frequency and angle measurements with the number of acquired satellites on May 1st from 1AM to 2AM

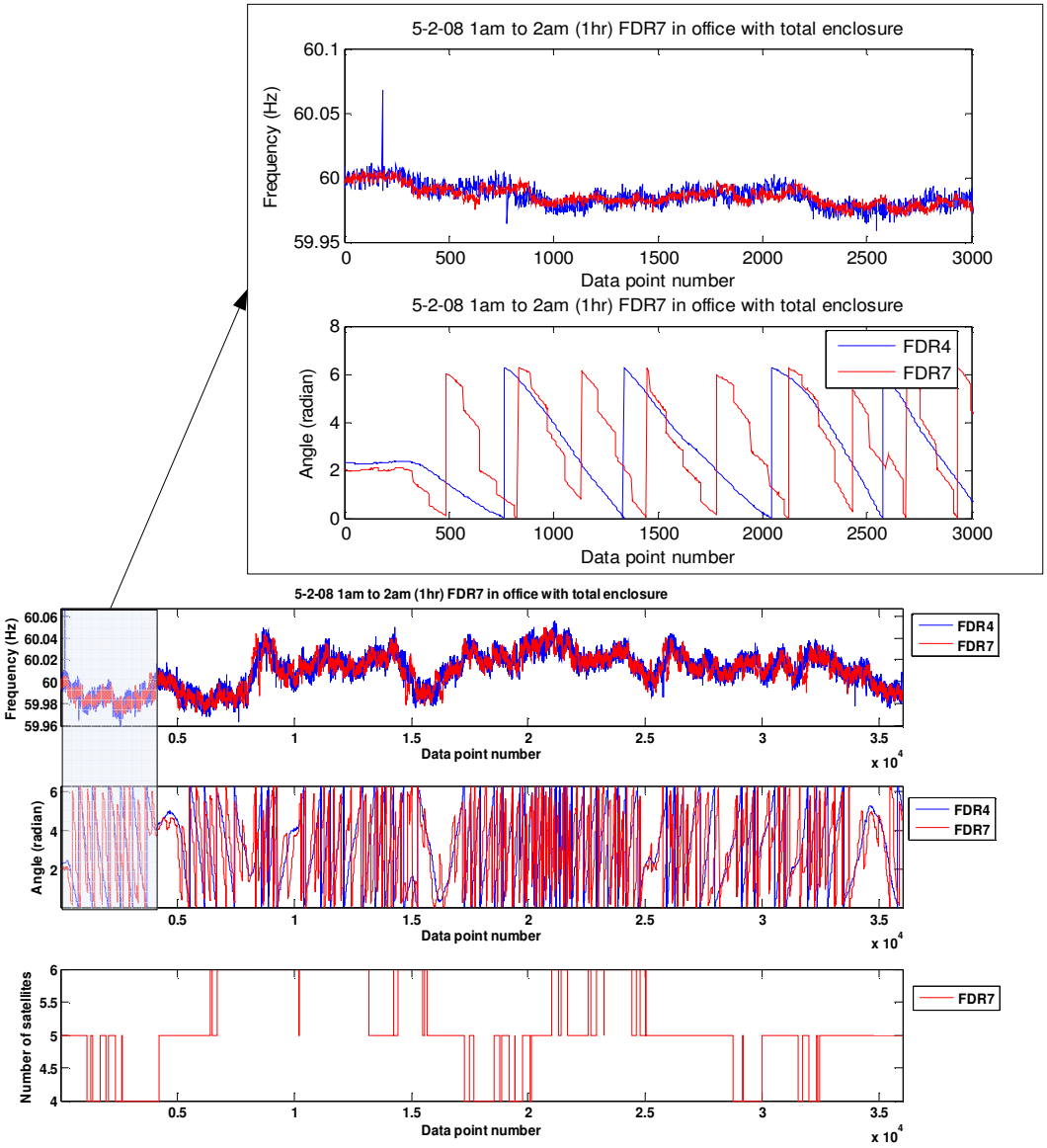

Figure 4.12 Frequency and angle measurements with the number of acquired satellites on May 2nd from 1AM to 2AM

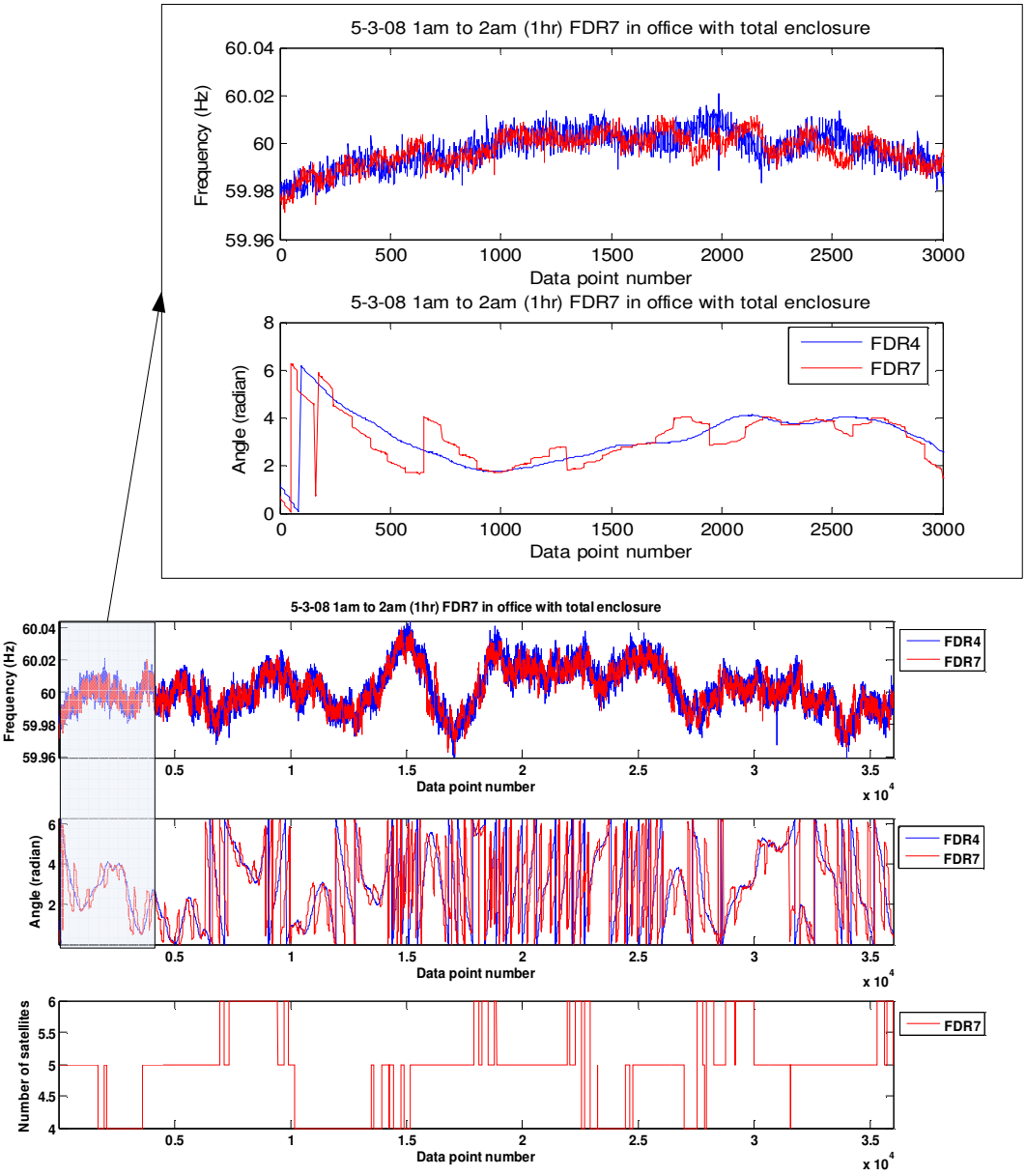

Figure 4.13 Frequency and angle measurements with the number of acquired satellites on May 3rd from 1AM to 2AM

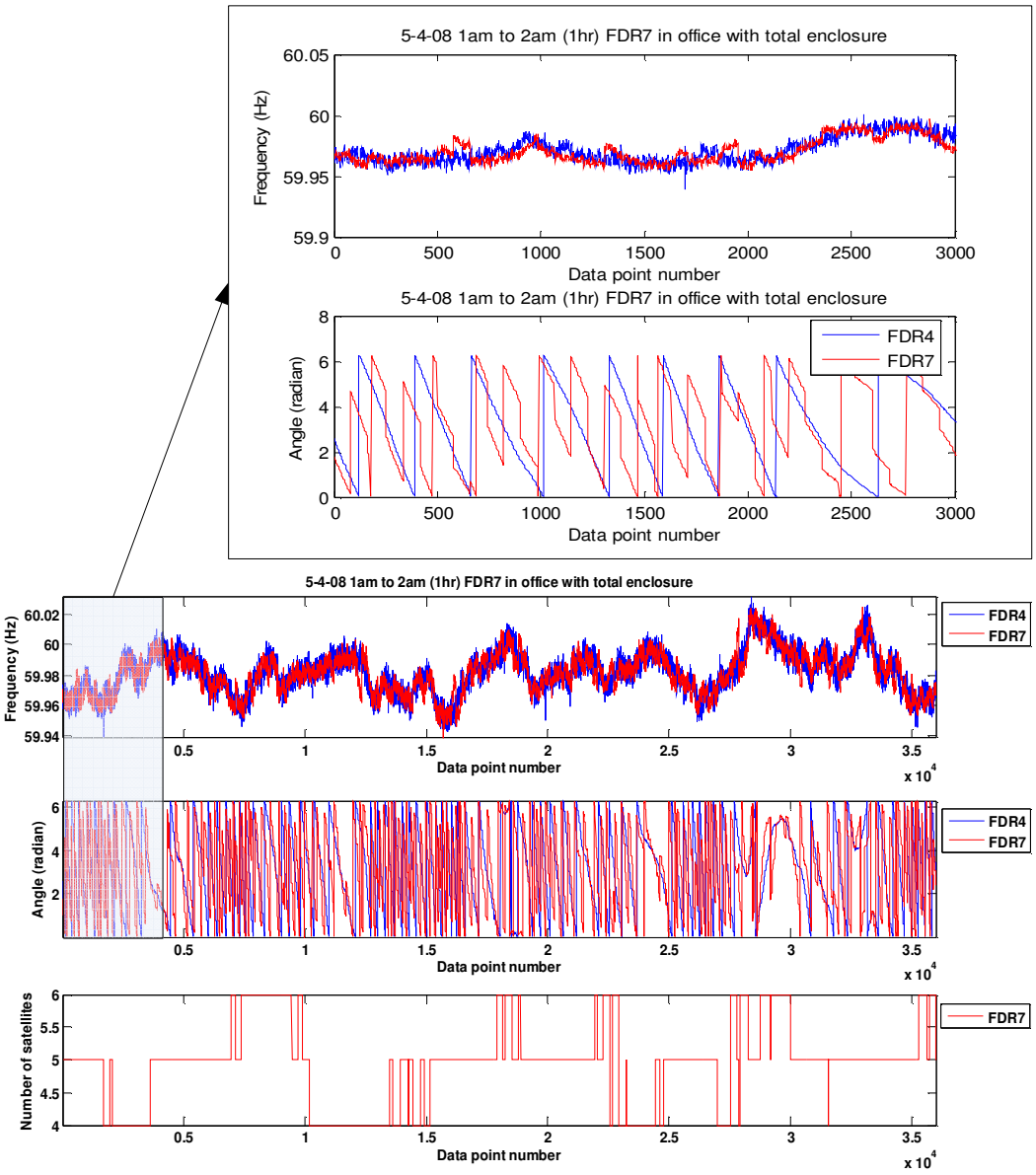

Figure 4.14 Frequency and angle measurements with the number of acquired satellites on May 4th from 1AM to 2AM

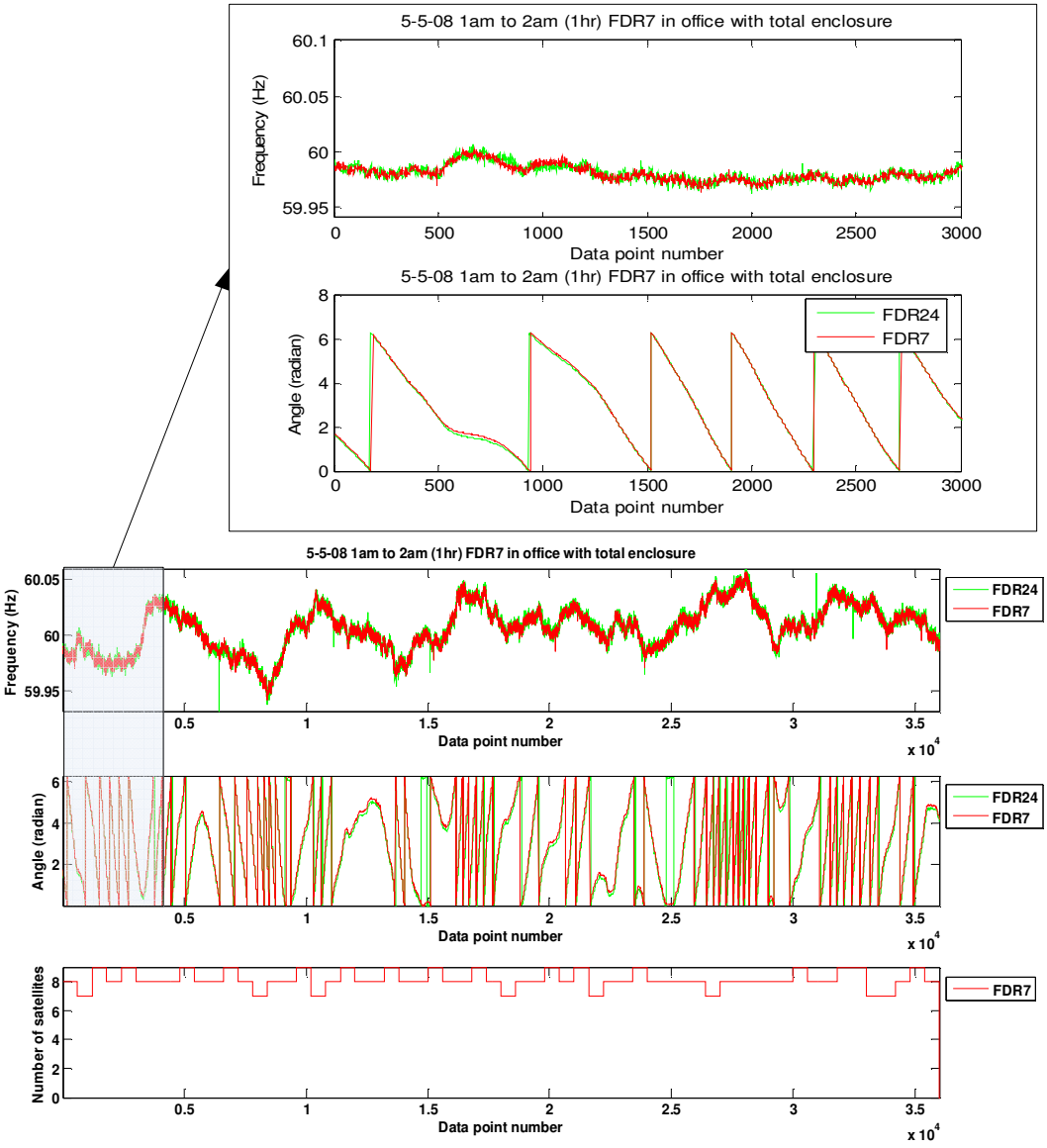

Figure 4.15 Frequency and angle measurements with the number of acquired satellites on May 5th from 1AM to 2AM

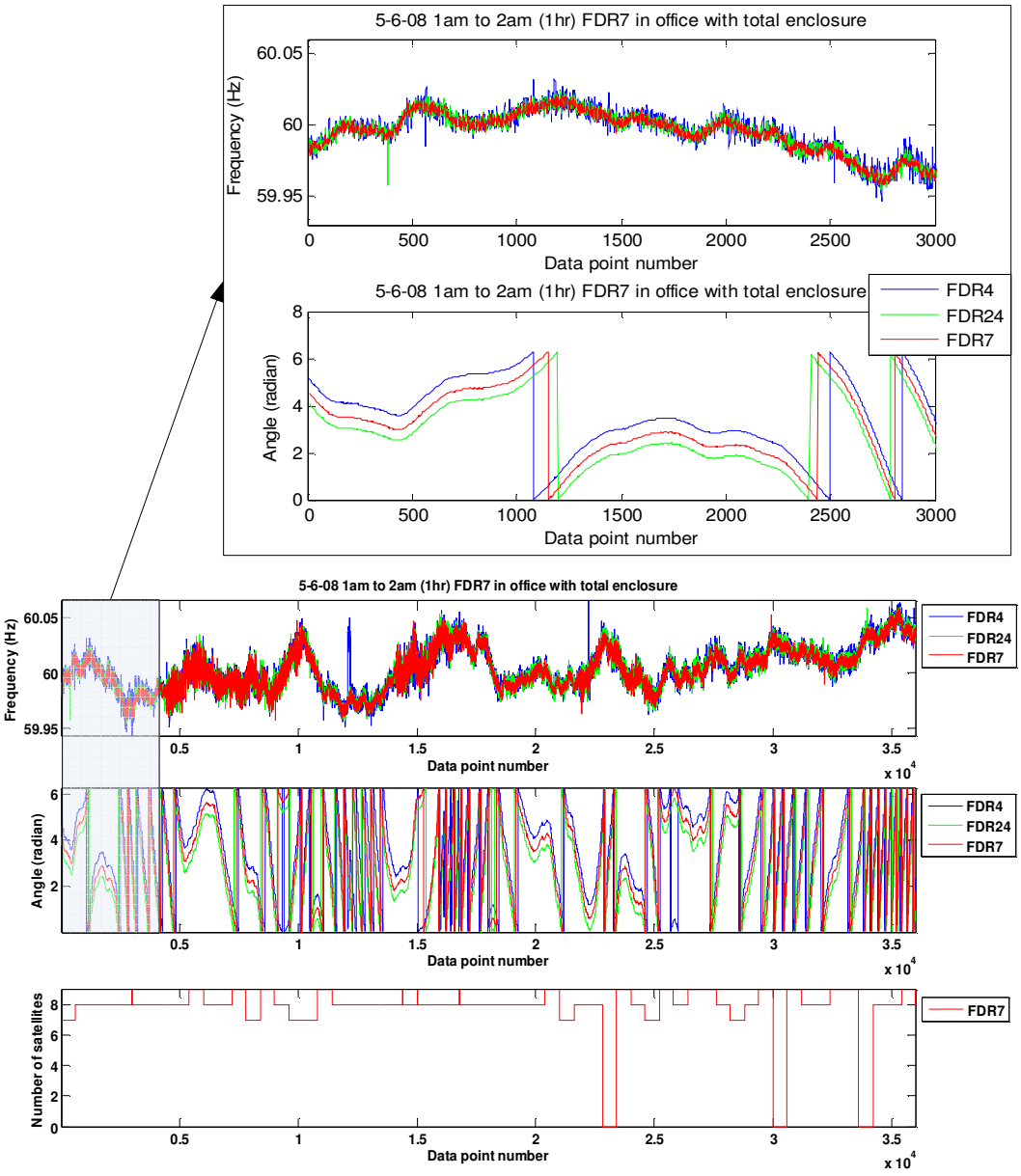

Figure 4.16 Frequency and angle measurements with the number of acquired satellites on May 6st from 1AM to 2AM

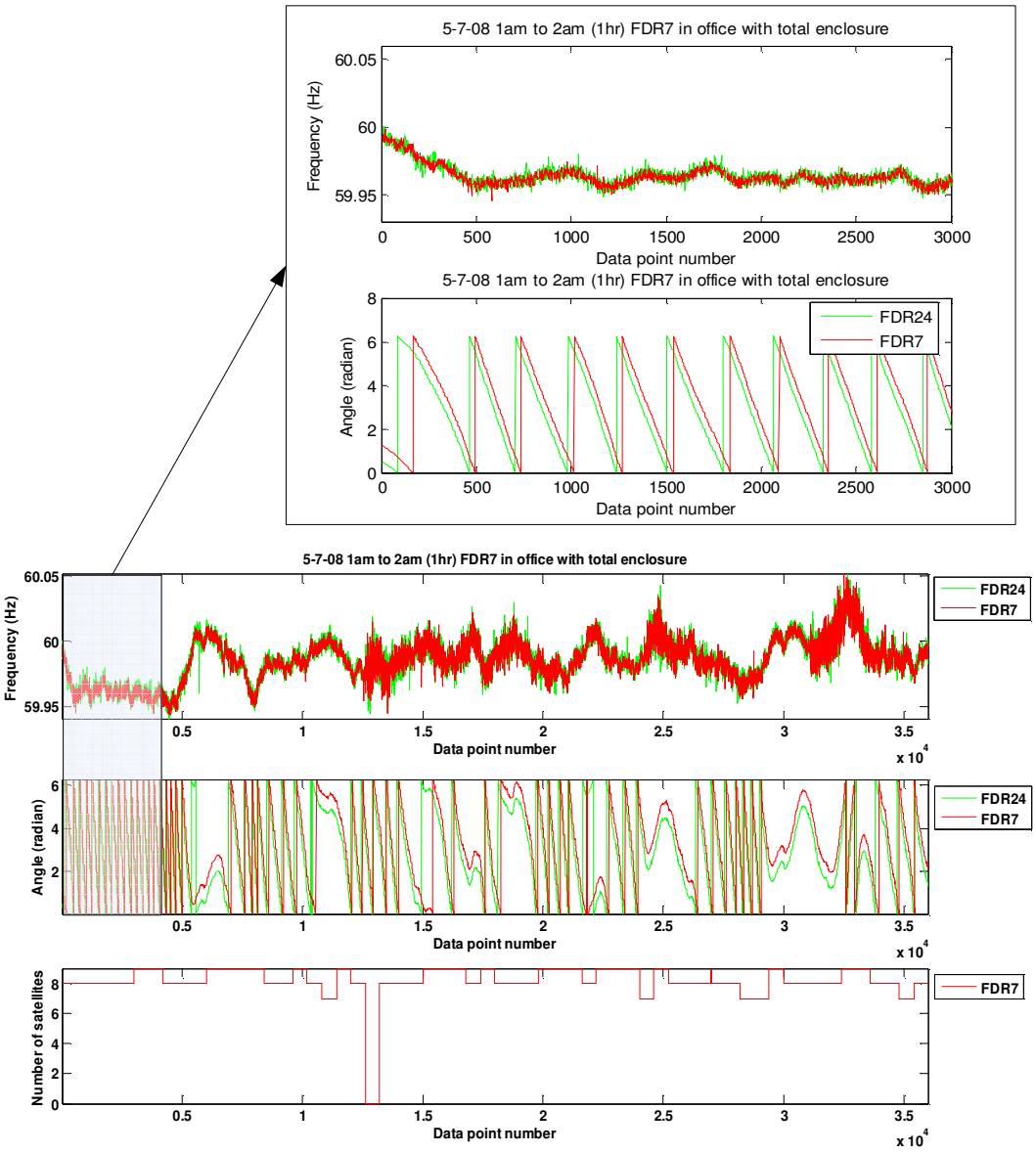

Figure 4.17 Frequency and angle measurements with the number of acquired satellites on May 7st from 1AM to 2AM

Under the assumption that there is a direct relationship between the signal strength and the quality of the position solution, it was shown in the previous section that under significant signal degradation the indoor GPS is susceptible to high DOP figures. In this section, the frequency and angle measurements are presented over the course of a week to illustrate the quality of indoor GPS timing as well as its capability in acquiring satellite signals in an indoor environment. The measurement data has shown that the CW12 is capable of operating under stringent conditions with significant signal degradation. Not only was the indoor GPS receiver able to acquire satellite signals without a line of sight

to the satellites, the frequency and angle measurements are comparable to the FDR with the conventional GPS receiver. Another measurement trial was conducted in an apartment bedroom where the antenna was initially located at the window and was slowly moved away from the window at a daily basis to observe its effect on frequency and angle measurements. The following observations can be gathered from the measurement results.

- The measurement data verifies that as the acquired number of satellites increases, the frequency and angle measurements become more accurate. Starting from May 1st to May 4th, the zoomed-in angle plot for FDR7 shows some discontinuities in comparison with FDR4. This phenomenon can be attributed to the low quality timing solution provided by the indoor GPS receiver. Since the frequency estimation is a direct result of angle measurements, the corresponding frequency data from the two units does not match each other very well. Then starting from May 5th, the sample frequency measurement data from different units matches each other and the angle measurement data from FDR7 is relatively smooth without the discontinuities observed before. It is also shown that after May 5th the number of satellites increased to the range of 7 to 9. However, there are instances where the number of acquired satellites drops to 0 for about 10 minutes and rebound back to a large number of acquired satellites. Such phenomenon does not seem to affect the frequency and angle measurement accuracy. It is most likely that the CW12 was steering the 1PPS with its local oscillator without corrections from the satellite signals during the 10 minutes of outage.
- The test was initiated on April 29th but the FDR with indoor GPS was not able to perform accurate frequency and angle measurements until May 5th. So it took about 7 days for the FDR with indoor GPS to acquire a large number of satellite signals and also provide acceptable frequency and angle measurements.
- During the course of the measurements, the indoor GPS acquired up to 9 satellites whereas the conventional GPS only acquired up to 5 satellites at any instance in time. Furthermore, as the number of acquired satellites increases, the indoor GPS is shown to provide similar levels of accuracy in timing as that of the conventional GPS operating under nominal conditions. This improvement in

accuracy is a direct consequence of the indoor GPS obtaining more accurate position fix over time as the number of acquired satellites increases.

- The plots show that after May 5th, FDR7 (with indoor GPS) has less noise in its measurements comparing with that of FDR4 and FDR24 (with conventional GPS). It is most likely that the extra measurement noise is a direct consequence of the laboratory environment.

4.7 Recommendations

Based on these observations, in an extremely high signal degradation environment such as an internal office room without any window, the CW12 indoor GPS behaves differently from that of the conventional GPS operating under nominal conditions. It can be observed that even with 4 to 6 satellites acquired, the quality of the timing solution was poor and not acceptable for accurate phasor angle measurements. This may be attributed to the lowered signal strength and poor geometry of the acquired satellite signals. However, when the number of satellites has reached the range of 7 to 9, the frequency and angle measurements are comparable to that of the FDR using conventional GPS. The accuracy improvement can be attributed to the redundant measurements as they are averaged to the overall position solution. Furthermore, with the aid of the assistance data including time, approximate position and satellite ephemerides, the indoor GPS should reveal better performances. In the case of FDR, approximate location data is often given before deployment, hence providing the possibility of incorporating the position information into the FDR firmware. With the position information given, the indoor GPS should have a shorter time to first fix. To improve the accuracy of the GPS solution, the satellite mask angle can be increased to 5° so that the receiver only tracks satellites for which the elevation angle is greater than 5°. Ultimately, the measurement solution becomes more accurate as the satellites near the horizon are removed from the solution. Given that the CW12 is able to provide accurate timing under significant signal degradation, new possibilities in FDR installation location can be proposed. Furthermore, given the verification that the CW12 is capable of providing accurate timing in an indoor

environment, it can be inferred that the CW12 will also be able to operate with the similar performances when it is located in office building with metallic coated window.

Since the measurement data suggest that the CW12 behaves differently from the M12+ and M12M, a new control strategy needs to be developed to maintain the credibility of the timing solution provided by the CW12. For the existing control strategy in support of the M12+ and the M12M is mostly based on the number of acquired satellites and the 1PPS output is enabled when the receiver is tracking at least one satellite. Clearly this control method is ineffective for the CW12 as it was shown in an indoor environment, the frequency and angle measurements were inaccurate even when the CW12 was tracking 4 to 6 satellites. It is important to note that the fundamental difference between the indoor GPS and the conventional GPS is the enhanced capabilities to acquire and track weak GPS satellite signals to provide high availability, at the expense of incorporating erroneous measurements into the computation of position and timing solution. In the case of the indoor GPS, the accuracy varies as a function of the environment and the receiver provides measurements almost regardless of the environment. The fact that the indoor GPS accuracy varies makes the validation of the GPS solution difficult. The trivial method is to incorporate the use of the TRAIM feature into the verification of GPS timing solution. As an example, an alarm limit of 1 microsecond can be set so that redundant measurements can detect timing errors greater than 1 microsecond. The drawback to this method is that a large number of acquired satellites with high-quality geometry are needed to provide for the redundant measurements. Hence the TRAIM feature will not always be available and is subject to outage when the number of acquired satellites is low. Another possibility for estimating the accuracy of the indoor GPS solution is by interpreting the transmitted GPS messages. Like most of the commercial GPS receiver, the raw GPS measurements are not readily available in the message stream provided by the CW12. Although multiple studies have suggested the estimation of the pseudorange errors using least squares and Kalman filtering method to quantify indoor GPS accuracy [40].

To estimate the accuracy of the indoor GPS when it is operating in an indoor environment, it is possible to monitor the signal strength, DOP and the acquired number of satellites. To this end validations can be performed for each receiver channel with respect to carrier to noise ratio. Specifically a C/N_o threshold can be set for 35 dBHz and when all of the acquired satellite signals reach below 35 dBHz and the number of acquired satellites is low the FDR should stop making measurements. However, in the case when the acquired number of satellites reaches above 7 it would no longer be necessary to validate the signal strength. Furthermore, to handle the occasional short term lose of all satellite signals, the FDR should continue to perform measurement for 10 minutes without interruption. When the 10 minutes has elapsed and the acquired number of satellites does not rebound, the FDR should stop making measurements and switch from collection state to acquisition state. Nevertheless, one single method cannot perform the challenging task of showing accuracy estimates reliably. Thus, choosing the right combination of fault detection methods for indoor GPS timing as it is applied in accurate frequency and phasor angle measurements will be a subject of future research.

Chapter 5 Timing Measurement Based on a High Stability and High Resolution PC Counter

5.1 Background

As it was described in Chapters 1 and 2, the phase voltage angle calculated by the phasor algorithm is highly dependent on the timing accuracy of the FDR timing subsystem. The different aspects of timing, such as keeping accurate time, frequency, and the ability to time stamp events when they occur are all crucial aspects of a phasor measurement device. The focus of this chapter is on the development of a high resolution yet inexpensive timing measurement infrastructure and an analysis of inherent timing and frequency stability of PC oscillator.

The motivation for a PC based timing subsystem was initiated by the concept of the PC based FDR design. Similar to the embedded systems solution, a PC based FDR will also need an accurate timing subsystem to synchronize the sampling voltage as well as providing accurate timestamps. However, what is different about a PC based implementation is that there are many readily available accurate timekeeping and synchronization techniques for PC's. To this end it is necessary to explore how timekeeping is conducted on a PC as well as exposing any limitations that it may have and more importantly, how to go about improving the accuracy of the PC timekeeping. In fact, most of the precise timekeeping on the PC is regulated by an external GPS or some other forms of higher accuracy timing source, based on the assumption or general acceptance that the PC clock is not a precise timing source.

It is fair to say at this day and age that almost all of the clocks are made of two essential components, an oscillator and a counting mechanism with the oscillator operating as the main driving force behind the timekeeping. As it will be shown, there are many factors that influences the accuracy of oscillator timekeeping and a variety of compensation methods have been developed to further enhance the capabilities of crystal oscillators. However, it is just as important to gain some insights into why the standalone PC clock cannot be a precise timing source and quantify its accuracy and stability.

Consequently, an in-depth analysis is conducted on the inherent accuracy and stability of the PC clock and how it can be applied in the implementation of a high resolution and high stability timing measurement system, which can ultimately be used to evaluate the accuracy of different precision timing sources and possibly the FDR sampling time.

5.1.1 Oscillator Characteristics and its Accuracy

Timekeeping accuracy has improved about 10^6 fold in the first sixty years after the introduction of quartz crystal oscillators (about 1920) [59]. However, in the last two decades the progress in certain areas such as long term aging has not shown as significant progress.

Two of the most fundamental characteristics of any clock are its rate and offset. Both rate and offset can be measured. To measure the rate of the clock, a time interval measurement method can be implemented, which determine the elapsed time between two events. Similarly, the offset of a clock can be measured using time synchronization measurements, which determine the time offset between the test signal and the UTC second [58]. As a result, with a high accuracy 1 pulse per second signal synchronized with UTC, one can easily obtain a very accurate time of day clock. Nevertheless, almost all of the oscillators today are in the Megahertz if not Gigahertz range, thereby providing resolutions measured in microseconds or nanoseconds.

In the modern world, a quartz crystal oscillator is in the heart of nearly all frequency control devices. Furthermore, the quartz crystal oscillator is low cost and provides a reasonably precise and stable frequency source. Although some other materials such as ceramic resonators have been developed but their accuracy cannot surpass that of quartz crystals [60]. In addition, there is a wide variety of quartz oscillators that can be made in accordance to the requirements in cost, accuracy and stability. While the design of quartz crystal oscillator is outside of the scope of this research, it is important to get a basic understanding of how quartz oscillates in order to learn its associated uncertainties. As shown in Figure 5.1, quartz crystal is a piezo-electric substance and it can generate voltage when mechanical stress is applied. Conversely, when a voltage is applied across

the crystal will generate mechanical movement within the crystal. In the event that an AC voltage is applied, the crystal will begin vibrating. The rate that the crystal will vibrate with the most accuracy is its resonant frequency, which is determined by the cut, size, and shape of the crystal, as well as any fixed mechanical stress applied to the crystal [60].

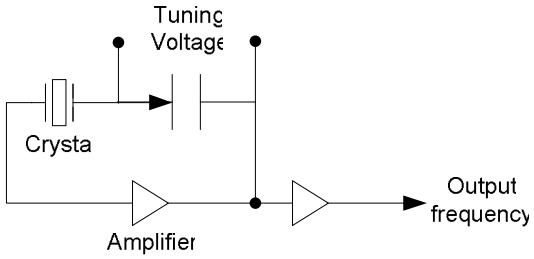

Figure 5.1 Crystal Oscillator Block Diagram

Before delving into the details of crystal oscillator performance, it is necessary to take note of the difference between accuracy and stability. Generally, the accuracy can be characterized by the capability of a crystal oscillator to generate a frequency where the systematic uncertainties such as frequency offset relative to the ideal are known. An accuracy statement usually involves the upper and lower limit for deviations from that of the ideal. On the other hand, the frequency stability can be characterized by the oscillator's ability to stay within specific frequency limits for some sampling time, τ. Nevertheless, in the standards laboratory setting, it is intuitive to argue that the best approach to long term stability is to improve accuracy. The basis for this claim is based on the idea that long term variations in output are caused by variations in the systematic offsets. By reducing and controlling offsets, both accuracy and long-term stability should improve [59]. On the other hand, in practical situations stability is often the key consideration. Consider the example of synchronized phasor measurement, more emphasis are placed on the stability of the frequency reference to allow for accurate synchronized sampling. To this end, it is important to recognize the importance of both accuracy and stability but the requirement for stability is far stricter.

With any quartz crystal oscillators, there are some innate fall backs that can usually be compensated in some way. Factors such as temperature, crystal aging and retrace

establish the frequency accuracy of the oscillator whereas the tuning port noise, power supply noise, and vibration establish the stability of the oscillator [59]. In order to compensate for some of these anomalies that would otherwise affect the oscillator frequency, the quartz crystal oscillator designers came up with different ways of minimizing their influences. Since temperature is a main contributor to frequency inaccuracies, most of the sophisticated quartz crystal oscillators are temperature compensated. These include the temperature controlled crystal oscillator (TCXO), microprocessor compensation crystal oscillator (MCXO) and oven controlled crystal oscillator (OCXO) with the OCXO being the most accurate [58]. The TCXO measures the ambient temperature and adjust the oscillator to a calibrated compensation curve. This compensation makes TCXO's more stable with temperature variations in comparison with crystal oscillators without temperature control, but it does not make them immune. TCXO temperature stability is typically $+/-5x10^{-7}$ over the range of 0 to 70°C [58]. The MCXO is very similar to TCXO except that it uses microprocessor to store the frequency versus temperature characteristics of the crystal thereby making more accurate compensation. As a result the frequency stability of MCXO increases by a factor of about 10 to 100 times that of TCXO. In the case of OCXO, the crystal resides in an oven that holds the temperature constant, independent of the ambient temperature. OCXO typically exhibit stability of $+/-5x10^{-10}$ over 0 to 70°C [58].

For applications that may require more stringent timing accuracy, there are the atomic clocks such as the Cesium atomic clocks, Rubidium gas cell atomic clocks and the Hydrogen maser frequency standard. These are more precise and more stable than the crystal oscillators but they come with a higher price, larger size and more power consumption, which makes them unattractive for many common applications requiring a frequency source. Table 5-1 [61] tabulates the characteristics of different crystal oscillators with compensation as well as the atomic clocks.

Table 5-1 Oscillator characteristics

	TCXO	MCXO	OCXO	Rubidium	Cesium	Hydrogen maser
Accuracy per year	$2x10^{-6}$	$5x10^{-8}$	$1x10^{-8}$	$5x10^{-10}$	$7x10^{-10}$	$2x10^{-11}$
Aging per year	$5x10^{-7}$	$2x10^{-8}$	$5x10^{-9}$	$2x10^{-10}$	$2x10^{-10}$	0
Temperature stability (range ℃)	$5x10^{-7}$ (-55 to +85)	$3x10^{-8}$ (-55 to +85)	$1x10^{-9}$ (-55 to +85)	$3x10^{-10}$ (-55 to +68)	$5x10^{-10}$ (-55 to +85)	$2x10^{-11}$ (-28 to +65)
Size (cm3)	10	30	20-200	200-800	1,000	6,000
Price (~$)	10-100	<1,000	200-2,000	2,000-8,000	<10,000	50,000

The simplest of all quartz crystal oscillators is the simple packaged crystal oscillator (SPXO) which is often used in electrical devices such as computers using it as a clock signal source. Although SPXO is more susceptible to temperature variations than the TCXO and OCXO but it is in high availability and low cost, which makes it an attractive solution for most of the laboratory measurement applications where the temperature is usually well regulated. Nevertheless there are many factors other than temperature that may influence the frequency of a crystal oscillator.

5.1.2 Factors Affecting Crystal Oscillator Frequency Accuracy

Since PC timekeeping is the focus of this research work, more emphasis will be placed on the factors that may affect the PC oscillators. Listed below are some of the factors that make some contribution to the frequency instabilities in the PC oscillators. It will be clear that out of all the factors that affect crystal oscillator frequency, temperature and aging are the biggest violators of frequency stability.

Temperature

As mentioned in the previous section, temperature is a significant factor which affects the frequency of resonators. Different crystal cuts have a different frequency and

temperature characteristics. The temperature and frequency relationship is somewhat complicated. First, the change of frequency with temperature may not be a linear function. In addition, even with the same kind of crystal, each crystal has a very different frequency-temperature curve. With some intentional temperature adjustments such as those described in [86], it is possible to have a general estimate on the temperature effects on PC oscillator clock. An Intel Pentium II 350MHz processor temperature was adjusted from room temperature to the maximum rated temperature and clock frequency readings were taken every second. The results indicate that the CPU frequency decreases at a rate of about 150Hz (4.29×10^{-7}) per degree Celsius of increase [86]. Nevertheless, the effects of temperature change can be reduced by providing a more constant ambient (room temperature control). To this end, most of the laboratory environments are temperature controlled rooms with good ventilation.

Aging (long term)

A gradual change in frequency over days or months is known as aging. The main causes of aging are mass transfer due to contamination and stress relief in the crystal's mounting and sustaining circuitry aging [58]. Aging usually occurs at a relatively constant rate per decade for each crystal. However, there are instances where the aging rate can reverse sign over time. Computer simulated aging from [60] shows a positive aging and a negative aging characteristic governed by the equations:

$$A(t) = 5\ln(0.5t + 1) \qquad \textbf{Equation 5-1}$$

$$B(t) = -35\ln(0.006 + 1) \qquad \textbf{Equation 5-2}$$

Simply combine the two characteristics to obtain the behavior when both aging mechanism are occurring simultaneously, which in effect reverses the aging direction. Some important concepts to be pointed out here are the fact that the aging rate of an oscillator is highest when it is first turned on and decreases as time goes on. The higher aging rate can reach any where from 1×10^{-7} per month to 1×10^{-8} per month [59]. For the later period, when the crystal has been operational for more than 2 months, the aging rate can reduce to anywhere from $(1\sim2) \times 10^{-9}$ per month to $(1\sim2) \times 10^{-10}$ per month [59]. To

compensate for the aging behavior, it is possible to periodically adjust the frequency of the oscillator. This change is usually done by an adjustable capacitor. However, in the case of the PC oscillator, it may not be practical to periodically adjust the frequency of oscillator to compensate for aging. Due to this the PC oscillator will always be susceptible to long term aging and such parameter can not be determined independent of time.

Retrace

As explained earlier, the frequency output of the crystal oscillator changes over time due to aging. When an oscillator is turned off and then back on, it will not necessarily start at the same frequency as it has been operating. Eventually the oscillator will begin to age according to its previous rate but will most likely be offset slightly from its original frequency. This effect is known as retrace and typically exhibits an offset in the order of $1x10^{-8}$ [58]. The causes of retrace are believed to be the same as those responsible for aging and temperature excursion.

Noise (short term stability)

Short term stability of the crystal oscillator is often portrayed as small variations in frequency for a specific averaging time. Specifications for short term stability have been often defined as the root mean square (RMS) uncertainty in the time base, averaged over one second. The reason that this value is a result of averaging is due to the fact that the short term stability is not known at any instant in time. It is primarily caused by noise in the active circuitry in the oscillator. Short terms stability can be eliminated by averaging over time. It should only be taken into consideration when measurements are significantly less than a second. In today's newer oscillator circuitry, the short term stability uncertainty has been reduced to $1x10^{-10}$ RMS over one second [58].

Others

Some other factors that affect the oscillator frequency instabilities include electromagnetic, power supply noise, gravity, shock, vibration, acceleration sensitivity and ionizing radiation [59]. Although in the case of the PC clock oscillator these are not as significant of a threat to the frequency stability as temperature and aging effects. For example, mechanical shock and vibration affects the physical stress on the quartz crystal but it can usually be treated as transient effects and be ignored. Typical sensitivity for quartz crystal time base is $1x10^{-9}$/g, where g is a force equal to one times the force of gravity [59]. Quartz crystal's inherent magnetic field sensitivity is smaller than 10^{-11}/T for fields smaller than 10^{-4}T [59]. Some instability in the oscillator can be traced to instabilities in the power supply. The frequency changes occur because changes in various voltages change the capacitance of active and passive components. The end result is a slight phase shift which directly influences the frequency. Crystal oscillator variations will typically be less than $1x10^{-7}$ for a 10% line voltage change [59]. However, most of the COTS PC has decent voltage regulator design and there isn't any known limit to oscillator frequency stability due to this effect. At last, the back ground radiation due to radioactive trace elements in the soil and building materials, cosmic rays and such will produce drift that is difficult to distinguish from aging. Nevertheless, this drift is not considered aging but a radiation effect. The radiation effect is at the minimum in the case of PC clock oscillator. Cosmic ray effects does not become a factor unless it's located in a significantly higher altitude and background radiation (on the surface of earth) contributes to aging rates of approximately 10^{-13} per day [59].

5.2 Timekeeping for COTS PC

In the old days of timekeeping almost all of the clocks are driven by the 110 or 220 volt power line and cause interrupt on every voltage cycle at 50-60Hz. Such clocks are outdated and are replaced with more modern clocks such as those used on the PC platform, where a 1.19318 MHz clock provides the base frequency to the PC timer [72]. 1.19318 MHz is one third of the National Television System Committee (NTSC) color subcarrier frequency to allow for TV output in the legacy systems. All i386 family boards have 14.318 MHz quartz crystal oscillator from which both the main clock frequency

(1.19318 MHz) and the CPU clock are derived [72]. The PC clock hardware is served by the clock driver software to maintain the time of the day.

5.2.1 Hardware Clock

In the earlier PC architectures starting with the introduction of the IBM-AT PC in 1984, there were two hardware devices that were used to update the operating system internal time, namely the battery-backed real time clock (RTC) and the programmable interrupt timer (PIT). The RTC is used to maintain the time even when the computer is off and in the IBM PC compatible computers, the RTC circuit is the Motorola 146818 with a resolution of approximately one second and significant drift. The PIT is used to measure elapsed time and trigger operations in a PC. In the Intel x86 architecture, either the Intel 8253 or the 8254 PIT is used to provide the base frequency for timer operation. Both the 8253 and the 8254 are capable of generating interrupts at specified timing intervals as designate by the programmer.

The majority of the new microprocessors today, starting with the Pentium in the i386 architecture, have a built-in CPU clock cycle counter. This cycle counter is used differently based on the operating system but it has a much higher resolution timing compared with the PIT and the RTC. In Intel Pentium processors, this register is named Time Stamp Counter (TSC) and is 64 bits long. In addition the new x86 PIT include a counter through the Advanced Configuration and Power Interface (ACPI), named the Local Advanced Programmable Interrupt Controller (APIC), which is mostly found on Symmetric Multi-Processor (SMP) computer systems. The APIC timer was designed to allow for per-processor timing in a multi-processor architecture.

5.2.2 Software Clock

The PC software clock (also known as the system clock) is generated by the Intel 8254 timer or a functionally equivalent device. Most of the operating systems have relied on the interrupts generated from the timer to keep track of time where the length in between interrupts is known as a tick. Upon every timer interrupt, there is a software

clock variable that is being updated by the fixed number of microseconds or nanoseconds in the tick interval. The timer generates an interrupt either every 10 or 15 milliseconds in the Windows operating system to maintain the absolute time. On the newer distributions of the Linux based operating systems, the division of the 8254 timer can be adjusted but it is often set at a default value of an interrupt being generated every 4 milliseconds, which enhances the timekeeping capabilities of Linux to surpass that of Windows operating system. Furthermore, the PC Basic Input Output System (BIOS) contains a software routine that counts the interrupt requests and generates a time of day clock that can be read or set by other software programs. As an example, the operating system may get the time of day information from the software clock to timestamp files.

The software clock by itself is not a good timekeeper. Among several things, its timing accuracy is limited by the performance of interrupt requests. Any changes in the interrupt request timing will be reflected in the software clock inaccuracies. Standalone software clock can be off up to a minute or more in a day. In addition, the software clock also has a limited resolution in the order of milliseconds. In other words, the software clock can only represent time of day in even multiples of the time interval between interrupts. Nevertheless, software clock accuracy is reduced even further by the nature of cycling the PC power. When the PC is first powered on, the software clock sets itself to within one second of RTC, which is susceptible to significant drift. After this initial synchronization, the RTC and the software clock will be running at different rates while the PC is running. To compensate for these inaccuracies, there are many networks based timing synchronization techniques being implemented. The topic of timing synchronization will be revisited in more details in Chapter 7.

A common enhancement to the Unix based software clock is to make use of the time stamp counter (TSC) register to interpolate time between PIT interrupts, thus increasing the resolution from milliseconds to microseconds. The resulting higher resolution is maximized at 1 microsecond, the smallest time unit available on the standard data structure for Unix. The updating of this register is done by hardware and reading it and storing its value takes only a few instructions. [93] indicate a median access latency of

420 clock cycles for the TSC counter. However, a conservative estimate would put the figure to be around 500 clock cycles. With today's gigahertz range PCs, the latency involved in retrieving the TSC counts is well below 1 microsecond. In the Pentium-class processors, the TSC counter is 64 bits long and can be accessed directly. Furthermore the TSC counter has a resolution that's directly related to the CPU core frequency. For example, a 1 GHz processor would have 1 nanosecond resolution in TSC counts. Provided that the duration of a clock cycle is accurately estimated, the TSC counts can be converted to an accurate relative time. At the same time it is important to note that there are a few underlying assumptions regarding the hardware architecture. Features such as power management, frequency stepping, and unsynchronized multi-processors affect the stability and consistency of the TSC. In order to avoid these pitfalls it is necessary to disable these features while when using the TSC as a timekeeping and timestamping source.

Recent literature has shown that the TSC counter exhibit high frequency stability and can be used in high precision measurement applications [82][83][84]. Results are presented by Veitch and his students, showing the implementation of an accurate clock based on TSC counter and its applications in Internet probing and measurements. The TSCclock is one instance of the implementation of Robust Absolute and Difference clock (RADclock) algorithm. With today's processor frequency, a rollover on the 64 bit TSC counter would take almost 200 years. Taking this into consideration, the stable long term clock rate estimates can be obtained using a feedforward approach. Results in [82] indicate such algorithm can achieve 10 microseconds accuracy in a LAN network. At the same time [93] shows that the hardware counters in the PC exhibit stability below 0.1 PPM in a temperature controlled room. Ultimately, TSCclock exploits the high stability of the TSC counter over relatively large averaging intervals to estimate the period of one clock cycle in the processor. With an accurate estimate of the period and constant calibration over time using network time synchronization, it is possible to use the TSC counter for accurate timekeeping.

In another recent study, [86] has also mentioned the use of TSC counter to measure the internal clock drift of computer clusters. Several Intel and AMD processor frequency was measured in variable temperature and variable load adjustments. Results indicate that PC processor frequency diminishes at the rate of approximately 150Hz per degree Celsius with a linear relationship between frequency and temperature. In addition, load variation has little effect on either the frequency or the temperature of the processor. Finally the frequencies of several processors were monitored from several hours to several days and the results shows stability of less than 1PPM.

To this end, the TSC counter has the potential to be used in many applications that requires high resolution and high stability timing. What's more important is the fact that the TSC is easily accessible in almost all of today's COTS PCs, thus eliminating the need to purchase a rather expensive oscilloscope data logger.

5.3 The Measurement Infrastructure

The measurement system based on the TSC counter was developed for several purposes. For one, it is important to examine the inherent stability of the TSC counter which directly reflects the internal PC clock oscillator stability. Such study will be useful in determining whether the inherent PC processor clock can be used in FDR timing and investigate its limitations. Another reason for building such a measurement system is to exploit its high resolution timing to measure the accuracy of different timing mechanisms and the possibility of measuring the FDR trigger for conversion signal in the ADC. Finally, the implementation of a measurement system with COTS PC and readily available open source software is attractive in comparison with some of the high cost measurement equipments.

Some of the basic requirements for the measurement system are fairly obvious. A PC with the TSC counter is needed along with a general purpose operating system for ease of user interaction. In the earlier stages of developing the timing measurement system, it was decided that the Linux based operating system Ubuntu to be used. A Unix based operating system offers flexibility, stability, open source software, and most of all

improved timekeeping in comparison with its counterpart Microsoft Windows platform. The main reason Ubuntu was chosen instead of another Unix based operating system is that it offers ease of installation, usability and it is part of open source Linux distribution. In fact Ubuntu installation makes up for a large percentage of all the Linux distribution for the desktop.

In addition to the Linux based platform, the integration of real-time operating system is crucial in a precision timing measurement system. In comparison with a general-purpose operating system, a real-time operating system will always be able to meet the timing requirements of the processes under its control. In the case of high resolution timing measurement with strict timing requirements, it is important to have a deterministic environment with low response time. Clearly, in order to achieve hard real-time, guaranties must be made that no deadline is missed. However Linux alone is not preemptible, which is another characteristic of the real-time operating system. Preemptibility allows for a higher priority process to run over an already running lower priority process in the kernel. As a matter of fact, this measurement system differs from that of [86] largely due to the introduction of a real-time operating system, which minimizes operating system latency and interrupts latency variability.

There are many real-time operating systems available but only few are open source. Open source software allows for flexibility in the modification of the source code as well as zero cost. With such criteria in mind, Linux provides some readily available open source code that modifies the Linux environment to be real-time. There are two different approaches in providing real-time in Linux. In the first one the standard Linux kernel is improved either by making the kernel preemptible or by adding preemption points to the code [77]. However, in order to make Linux kernel fully preemptible, a more drastic approach is needed. There are two on-going projects that enable full preemptibility in the kernel by adding a hardware abstraction layer (HAL) in 'between' the system hardware and Linux. In addition, a new separate real-time scheduler is used which runs Linux as its lowest priority thread. The hardware abstraction layer is only allowed to take over the control of system interrupts when no real-time task is running. When Linux tries to

disable interrupts it only sets a flag in the abstraction layer and cannot really turn off the interrupts. As a result the real-time scheduler has full control of the system and Linux runs without any significant modifications. Only a small real-time kernel is added to the system.

The real-time tasks are written as kernel modules and executed in the kernel space. With the aid of a special real-time Application Programming Interface (API) the user is able to fully control the real-time tasks. The two on-going projects are real-time Linux (RTLinux) and real-time application interface (RTAI). RTLinux is the oldest project of the two and RTAI is based on the ideas behind RTLinux. Since the implementation of RTLinux and RTAI is similar, not much performance gain is obtained in using one over another. However, RTAI has always been an open source initiative and is being actively developed. On the other hand, RTLinux started as open source but was later commercialized and all of the developments are applied to the commercial version only. Therefore RTAI was chosen as the platform to enable real-time in the measurement system. Finally the characteristics of the measurement system are shown in Table 5-2.

Table 5-2 Measurement PC specifications

PC specification	
Model	Dell Optiplex GX240
CPU frequency (nominal)	~ 1595291000
TSC resolution (based on nominal frequency)	~ 0.6268 ns
Operating System	Ubuntu
Linux distribution	Version 2.6.28.7
RTAI distribution	Version 3.7

5.3.1 RTAI (Real Time Application Interface) for Linux and Timers

RTAI was developed by Paolo Mantegazza and the team at Dipartimento di Ingegneria Aerospaziale - Politecnico di Milano (DIAPM). The development started right after the release of RTLinux. The people at DIAPM were not satisfied with the performance offered by the first version of RTLinux and added some new features. DIAPM modified all of the real-time timing, such as introducing periodic timing and greatly improved the one-shot timing by using the CPU TSC instead of the timer circuit.

RTAI originally supported only the x86 architecture in the first release but now supports a wide variety of architectures including PowerPC, ARM and MIPS. A description of the RTAI features and capabilities are presented in Appendix C.

RTAI real-time processes can either run in the kernel space or user space. Real-time applications running in kernel space are implemented as normal Linux kernel modules. As the very first action it is necessary to setup timers. Based on the application RTAI is capable of supporting both periodic and oneshot timers for scheduling. If the periodic mode is selected, the 8254 PIT will be in mode 2 and it is used to generate interrupts periodically. In this mode all periodic tasks must have a common time base for its period. On the other hand, in the oneshot mode, the timer will be programmed in mode 0 and is re-programmed on each timer interrupt. This leads to more overhead but allows tasks to be scheduled with any period time without regard to a common time base. More significantly, in the oneshot mode the time is measured using the TSC counter. The 8254 is only used to generate interrupts.

5.3.2 Measurement Software and RTAI Latency Mitigation

To measure the PC clock accuracy and stability, it is necessary to provide a frequency reference. Given a higher accuracy reference clock, it is possible to evaluate the accuracy of the PC clock with respect to the reference clock. Since the i-Lotus M12M GPS receiver is currently being used in the FDR units and it is known to provide high stability 1PPS over long intervals, the receiver will be used as a frequency reference in the measurement system.

In order to interface with the 1PPS signal from the M12M receiver, there are two options. The receiver offers a 1PPS output on its RCA connector and on the DCD (Data Carrier Detect) pin of the serial port for GPS message transmission. Both signals can be connected to the PC via either the serial port or the parallel port. However, since the signal is readily available on the DCD pin of the serial port and no additional wiring is needed to make the connection, the serial port was chosen as the 1PPS interface.

Since the use of interrupts is the most effective way to deal with external events, the serial port interrupt system is used to capture the 1PPS with minimized latency. For PC hardware, interrupt signals are received by the programmable interrupt controller (PIC) and forwarded to call the routine which is responsible for handling the interrupting device. For the processor to know which device or devices need to be serviced, the interrupt controller maps each of its interrupt input lines to an interrupt request number. The processor gathers this number or also known as the vector from the controller and executes the particular handler. [76] Depending on the PIC that is being implemented in the PC, there are a total of either 16 or 24 interrupt requests (IRQ) connected to the input of the PIC. In the x86 systems, the first serial port is being assigned to IRQ 4 with an IO address of 0x3F8. With all of this information in mind, the C code was developed to read the TSC counter or the system clock upon every interrupt generated on the DCD pin of the serial port. Furthermore, the TSC counts are being recorded into the kernel log by using the RTAI instruction for writing to kernel log in real-time. Appendix C shows the C code listing. Basically there are three modules in the code, the interrupt handler, initialization of the real-time module and resource clean up upon termination of the program.

The most vital characteristic of a real-time operating system is how responsive the operating system is in servicing internal and external events. These events include external hardware interrupts, internal software signals, and internal timer interrupts. The measure of the responsiveness can be attributed to latency and jitter noise. Literature often defines latency as the time between the occurrence of an event and the execution of the first instruction in the interrupt code whereas the jitter is the variations in the period of the events. Although RTAI is known to provide hard real-time capabilities, there are some factors that affect the performance of the system.

As today's PC technology improves, more advanced hardware is integrated into the system introducing latencies in real-time environment. Some common suspects that results in system latency are SMI (system maintenance interrupts), power management systems such as APM (advanced power management) and ACPI (advanced configuration

and power interface), X Window system and video mode emulation, CPU frequency scaling and USB interface. The latency sources cause unpredictable timing results and they are incompatible with the concept of real-time. In general, most of the new motherboards with Intel chipset are likely to have SMI. SMI introduces anywhere of about 100 microseconds to up to a few milliseconds of latency in a real-time environment. RTAI provides a kernel module that disables most of the SMI as a workaround to minimize latency that results from SMI. Other latency sources including APM, ACPI, and CPU frequency scaling can be disabled when configuring the Linux kernel. The X Window system in Ubuntu is GNOME (GNU Network Object Model Environment) based and can be disabled so that the system would boot into terminal mode upon default. However, out of all of the latency sources, the USB controller is known to contribute the most latency, ranging greater than 100 milliseconds.

Initial measurements of the PC TSC counter were not successful due to latency. As it was mentioned earlier, the performance of RTAI can be significantly degraded with some PC peripheral hardware. Experience has shown that the biggest latency contributor for RTAI is the legacy USB controller. The TSC counter readings were acquired from using the code in Appendix C for reading the TSC counter upon every 1PPS and store to kernel log. Since there is no possibility of the TSC counter ever rolling over in the lifetime of the experimentation, a post-processing program was developed in C# to find the difference between each TSC counter reading. This can be illustrated in Equations 5-3 shown below:

$$f_{CPU} = (TSC_{n+1} - TSC_n) \qquad \textbf{Equation 5-3}$$

Where f_{CPU} represent the frequency of the PC clock oscillator as measured with reference to 1PPS.

Figure 5.2 shows f_{CPU} as measured with the legacy USB controller on the default setting. It is easy to see that the supposed deterministic behavior of RTAI is being disrupted. When magnified, Figure 5.2 shows there is an approximately 1 millisecond latency that occurs periodically at about every 16 seconds. In comparison, Figure 5.3 shows that when the legacy USB controller is disabled the periodic latency is eliminated.

Since the USB controller is disabled, USB devices such as keyboard and mouse can not be used. Nevertheless, the motherboard offers two PS/2 interfaces for PS/2 compatible keyboard and mouse so that user interface is still possible during runtime. Once the measurements are taken and stored into the kernel log, the USB controller can be enabled to allow for transfer of the files via USB storage for post-processing on a separate machine.

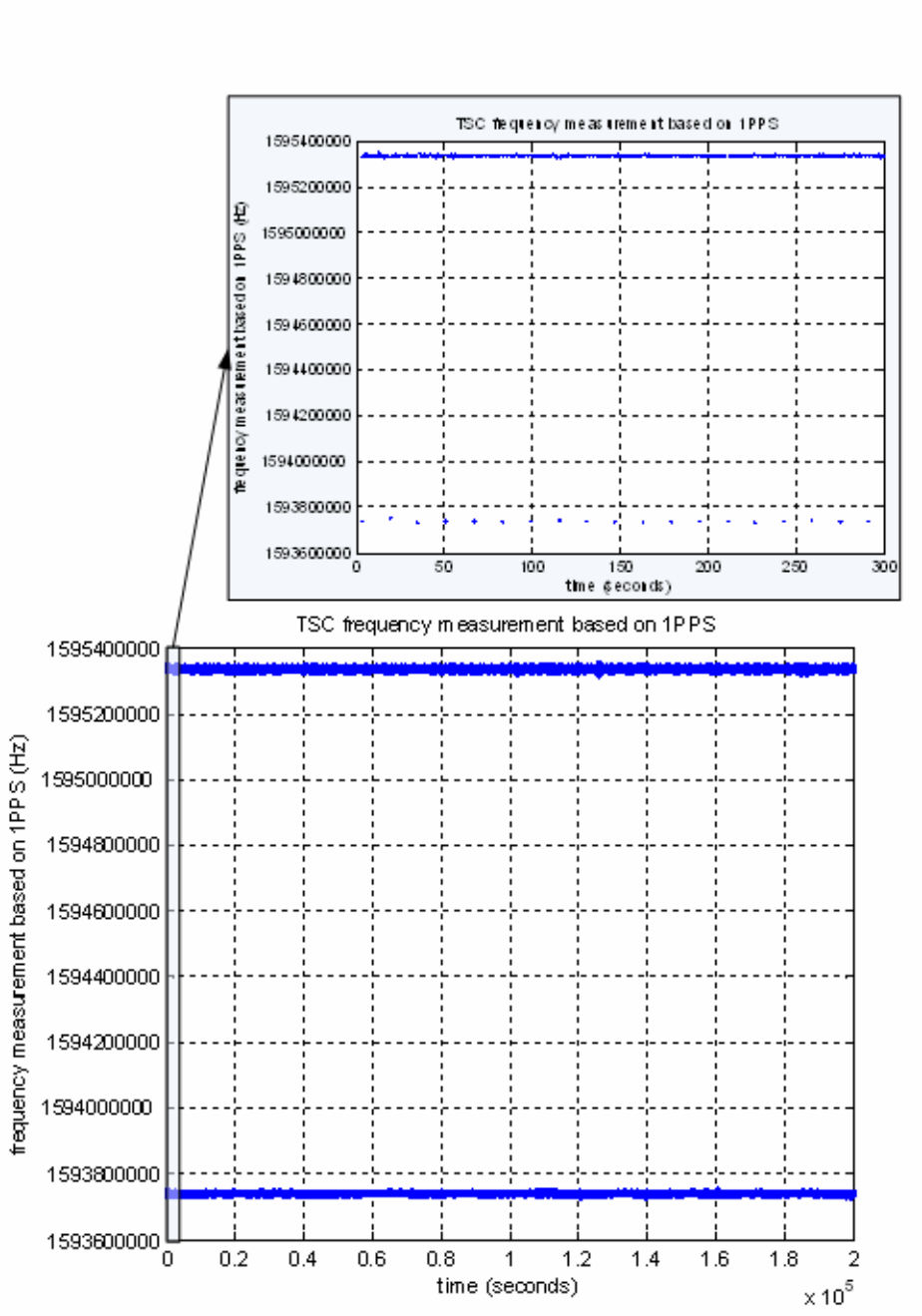

Figure 5.2 Measured TSC frequency with USB controller enabled

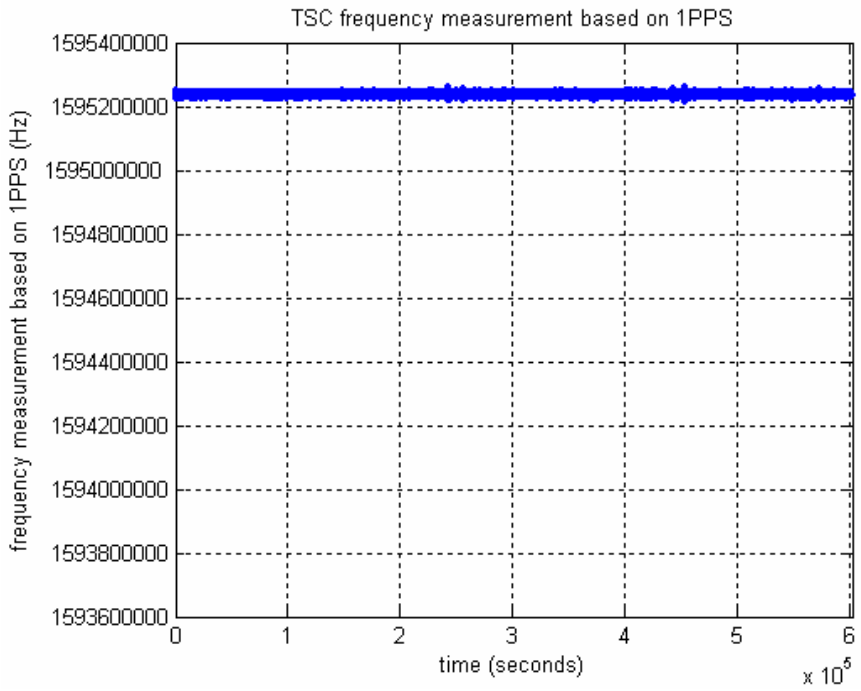

Figure 5.3 Measured TSC frequency with USB controller disabled

5.4 Measurement Results and Time Domain Analysis

With the measurement system described previously, it is possible to measure the accuracy and stability of different timing sources. However, before the analysis is conducted it is important to review some definitions and terminologies for quantifying the accuracy and stability of clocks.

The basis for time domain stability analysis starts with an array of equally spaced phase (time error), or fractional frequency deviation data arrays, x_i and y_i, respectively, where the index i refers to data points in time. These data are equivalent and conversions between the two are possible. The x values have units of time in seconds and the y values are dimensionless fractional frequency. The x(t) time variations are related to phase variations by:

$$\phi(t) = x(t) \cdot 2\pi\nu_o$$ **Equation 5-4**

Where ν_o is the carrier frequency of the frequency source. However, since infinite bandwidth measurement equipment is not available in this context, the instantaneous phase Φ(t) is not measurable. Since the primary concern here is frequency stability, the

amplitude fluctuations of the frequency source can be neglected. As a result, the sine wave output voltage of a frequency source can be approximated by the equation:

$$V(t) = V_o \sin(2\pi v_o t + \phi(t))$$ **Equation 5-5**

Where V_o is the nominal peak output voltage, v_o being the nominal frequency and $\Phi(t)$ represents the phase deviation. For the analysis of frequency stability, the $\Phi(t)$ term is the primary subject of interest. The instantaneous frequency is the derivative of the total phase:

$$V(t) = V_o + \frac{1}{2\pi}\left(\frac{d\phi}{dt}\right)$$ **Equation 5-6**

For precision frequency sources such as the crystal oscillator, the second term on the right hand is quite small so it is useful to define the fractional frequency y(t) in terms of the instantaneous frequency:

$$y(t) = \frac{\Delta f}{f} = \frac{v(t) - v_o(t)}{v_o}$$ **Equation 5-7**

y(t) is dimensionless and it is the fractional or normalized frequency deviation of v(t) from its nominal value. In relationship to phase and time, y(t) can also be defined as:

$$y(t) = \frac{1}{2\pi v_o}\frac{d\phi}{dt} = \frac{dx}{dt}$$ **Equation 5-8**

Where:

$$x(t) = \frac{\phi(t)}{2\pi v_o}$$ **Equation 5-9**

The data sampling or measurement interval is often being represented by the symbol τ_o, which has units of seconds. The averaging time used in the analysis, τ can be a multiple of τ_o to increase the averaging time.

The classical method of characterizing oscillator stability is a plot of Allan variance, which is defined using a series of time differences measured between a computer clock and some external standard. The fractional frequency in this case can be defined as:

$$y_k = \frac{x_{k+1} - x_k}{\tau}$$ **Equation 5-10**

Where x_k is the kth measurement and τ is the interval between measurements. The Allan variance, σ_y^2, is based on a "two-sample" variance measurement of the data. Instead of measuring the difference between each data point and the mean, it measures the difference between each data point, y_n and the next one y_{n+1}:

$$\sigma_y^2(\tau) = \frac{1}{2(M-1)} \sum_{i=1}^{M-1} (y_{i+1} - y_i)^2$$

Equation 5-11

Where m is the number of y data points in the calculation and the quantity τ being the sample time of the frequency measurement. The Allan variance is better than the classical variance in characterizing oscillator stability due to the fact that each frequency measurement is strongly correlated with the points near it in the sequence. As a result, when more data points are added to the set, the classical standard deviation value of the data set varies and is unbounded. On the other hand, the Allan variance has the advantage of being convergent for most types of clock noise. The Allan variance is the fundamental quantity, but Allan deviation, σ_y, is more frequently used and can be obtained by taking the square root of the Allan variance. Finally, Allan deviation results are presented as a plot of deviation versus sample time, or given as a value at a certain τ (Allan deviation of 1Hz at 100 seconds). Essentially, longer sample times are calculated by grouping the data into bins of length τ. The average value of the data points in each bin is used for the Allan deviation calculation.

To address the treatment of outliers in the measurement data, the median absolute deviation (MAD) is recommended in [103] for outlier recognition. In this context, the outliers will most likely be a result of the GPS losing acquisition of satellites at times and stops outputting the 1PPS. The MAD is a robust statistic based on the median of the data. It is the median of the deviations of the data points from their median value. Specifically, it is defined as:

$$MAD = Median\left\{\frac{|y(n) - m|}{0.6745}\right\}$$

Equation 5-12

Where m is the median of the data set and the factor 0.6745 makes the MAD equal to the standard deviation for normally distributed data. Therefore, based on their deviation from the median of the data, a deviation limit can be set in terms of MAD and a 5 sigma limit is most commonly used [103]. These median statistics are more robust because they are insensitive to the size of the outliers. This outlier rejection method is used in the following analysis.

5.4.1 PC Oscillator Accuracy and Stability Analysis

The subject of PC clock oscillator accuracy and stability is well known but there are limited literatures which seek to precisely measure the system. One survey based on Internet time synchronization has indicated that the median frequency error of 20,000 PCs is about 78PPM, with some PCs showing errors over 500PPM [87]. However, more optimistic results are shown for the oscillator stability in [84] and [86] with less than 1PPM error when averaged over a relatively long time.

The measurement trial was conducted in a one week period using the measurement setup where the GPS 1PPS is used to trigger the reading of the TSC counter every second. Then using the post-processing C# program, the interval length between each 1PPS in TSC counts can be extracted. Figure 5.4 shows a histogram of the TSC counter offset from the nominal processor frequency, calculated using Equation 5-13 shown below:

$$f_{offset} = f_{CPU} - f_{nominal} \qquad \textbf{Equation 5-13}$$

Where f_{offset} is the number of clock cycles offset from the processor nominal frequency and $f_{nominal}$ is the estimated processor nominal frequency. Given that oscillators are prone to fluctuations in frequencies, $f_{nominal}$ is only an estimate and subject to change as the measurement time is extended. Hence the sole purpose of defining $f_{nominal}$ here is to get an approximation of the timing error in seconds. In addition, Figure 5.4 shows the histogram of TSC counter offset scaled to seconds by dividing the processor nominal frequency. It is important to note that the measurement is approximate due to the deviations of the clock frequency each second and the latency associated with system. Nevertheless, as the averaging period is prolonged, these effects will be minimized. An estimate of the PC clock deviation can be made according to Figure 5.4 at about -31 to -37 microseconds, or 31 to 37 PPM of deviation error, which is close to what is often quoted in the oscillator specification of 50PPM deviation. In addition, the box plot of Figure 5.5 takes another perspective and shows the daily variations of the TSC counter with respect to the nominal frequency. Although the box plot is known to be an effective statistical tool in observing the median, lower quartile (Q1), upper quartile (Q3), sample minimum and sample

maximum, it is used here to observe the dispersion and skewness of the data. In this case, it can be observed that the daily variations in oscillator frequency is relative large. From these observations one can conclude that this particular PC oscillator may have undergone aging and experienced some temperature variations during the measurement which led to the disagreement between the measured and nominal frequency.

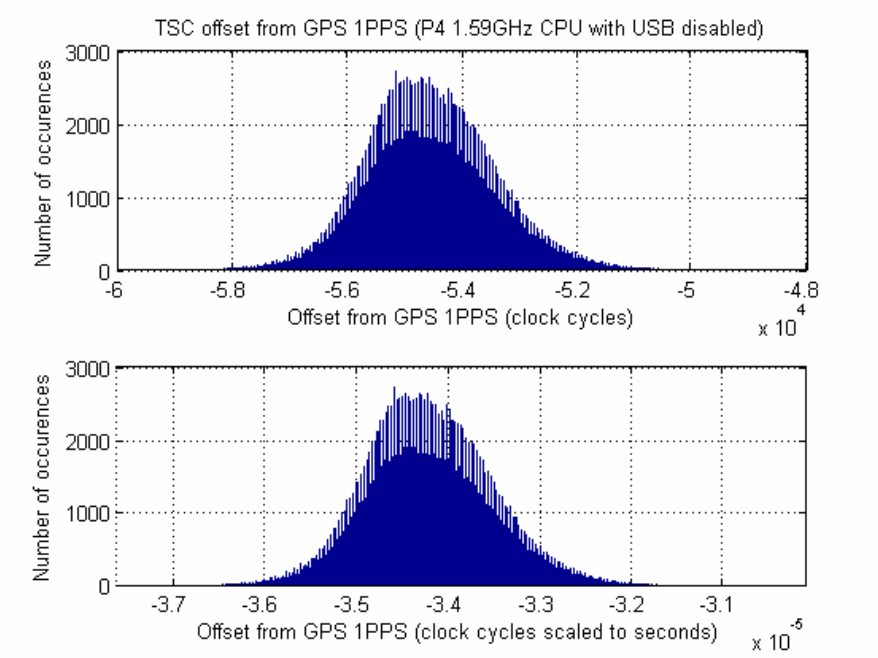

Figure 5.4 Histogram of TSC frequency measurements – offset from nominal CPU frequency

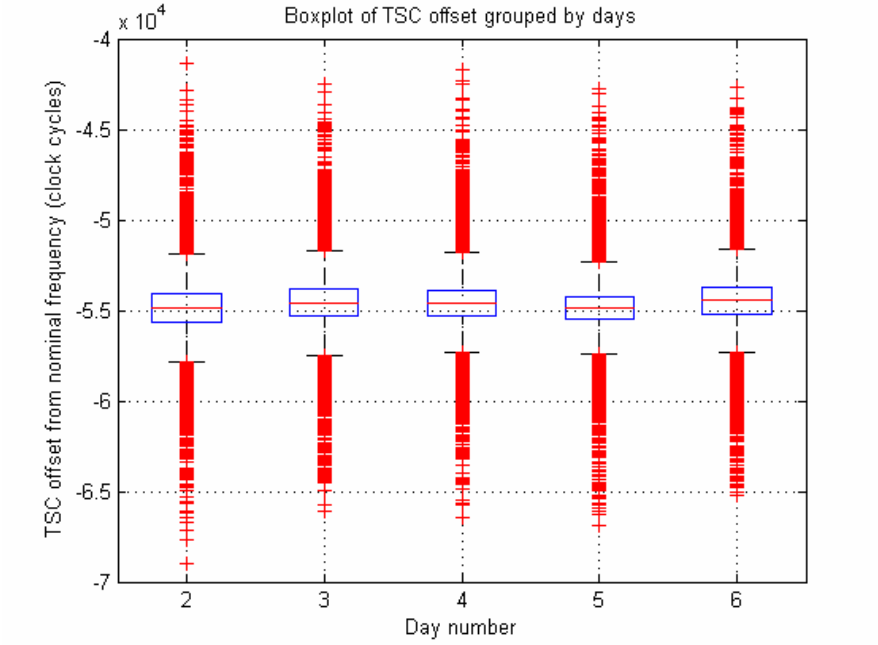

Figure 5.5 Boxplot of TSC frequency measurements – offset from nominal CPU frequency grouped by days

The most important aspect of this study is in looking at the stability of the TSC counter. Figures 5.6 shows the second to second difference of the frequency measurement in clock cycles and clock cycles scaled to seconds respectively. The second to second difference of the frequency measurement is simply obtained by y_{n+1} - $y_{n.}$, then the frequency measurement can be scaled to seconds by dividing by the nominal frequency. The plot demonstrate that the TSC counter has a maximum deviation of approximately 5 microseconds or 5PPM variation going from one second to the next. Furthermore, the histogram for the second to second differences is bell-shaped and similar to that of a normal distribution.

To examine whether the data follows the normal distribution, one can use the quantile-quantile (Q-Q) plot as a graphical analysis tool to interpret how well the data follows the distribution. Figure 5.7 shows the Q-Q plot of the data collected from second to second difference of TSC counter. It is shown that the data points close to zero (within 2 standard normal quantiles) closely follow the normal distribution whereas the data points at the tails deviate away. The reason for this phenomenon is closely tied to the random jitter noise associated with the serial port interrupt. Random jitter is often characterized by a normal probability distribution and theoretically it should be unbounded. Such characteristic is the result of the accumulation of various random processes including thermal noise and flicker noise. With the sum of many independent random functions, the resultant distribution tends to converge to a normal distribution by the central limit theorem. Also shown in Figure 5.8 is a box plot of the second to second difference in clock cycles. Once again it has confirmed the high stability of the TSC counter as the variations in frequency is very small.

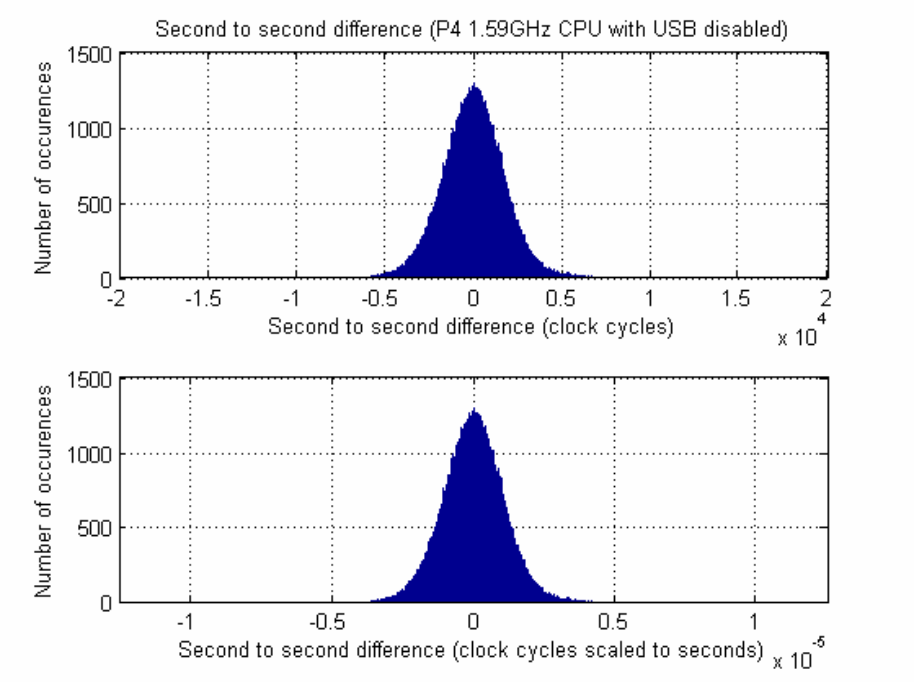

Figure 5.6 Histogram of TSC frequency measurements – second to second difference

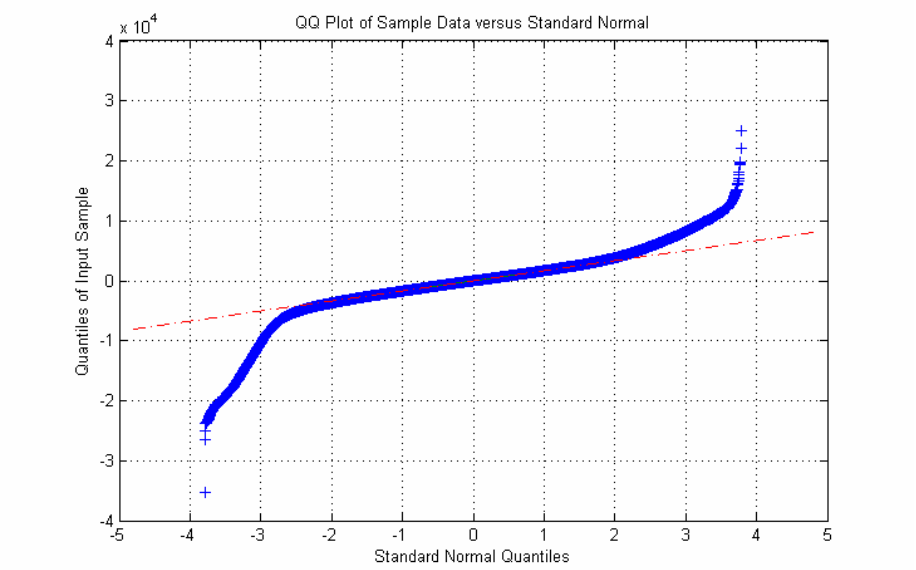

Figure 5.7 Q-Q Plot of TSC frequency measurements – second to second difference

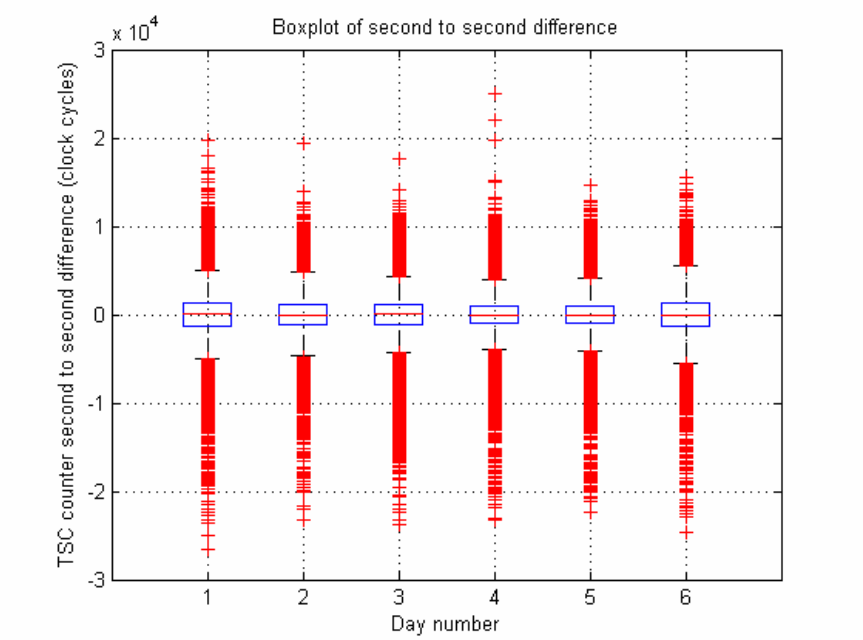

Figure 5.8 Boxplot of TSC frequency measurements – second to second difference grouped by days

Taking a step further in the stability analysis, Figure 5.9 shows the Allan deviation plot of the same set of data with the averaging period, τ_0 being one second. Figure 5.9 is consistent to the results obtained in [93], illustrating a nearly straight line starting from the left with slope near -1, reflecting the characteristic of white phase noise [89] and is largely attributed to the random jitter noise in the serial port. In this region, increasing τ increases the frequency stability in direct proportion. However, at the end of the line there are some sudden jumps which may have been caused by temperature variation in the room. Ultimately, the TSC counter is shown to reach well below 0.1PPM when the averaging period τ reaches above 120 seconds. At the point where τ reaches above 10000 seconds, the Allan deviation curve slowly deviates from the constant slope of decreasing PPM until it flattens in the end, indicating less correlation between one averaging interval and the next. At this point the effects of the serial port jitter noise will be dominated by the oscillator drift. Nevertheless, if the clock frequencies drifted in a measurable way, the values would vary over time and Allan deviation would tend to increase with time. The drift behavior does not seem to be significant in the averaging interval shown in the plot. A significantly longer averaging interval would most likely reveal such characteristic.

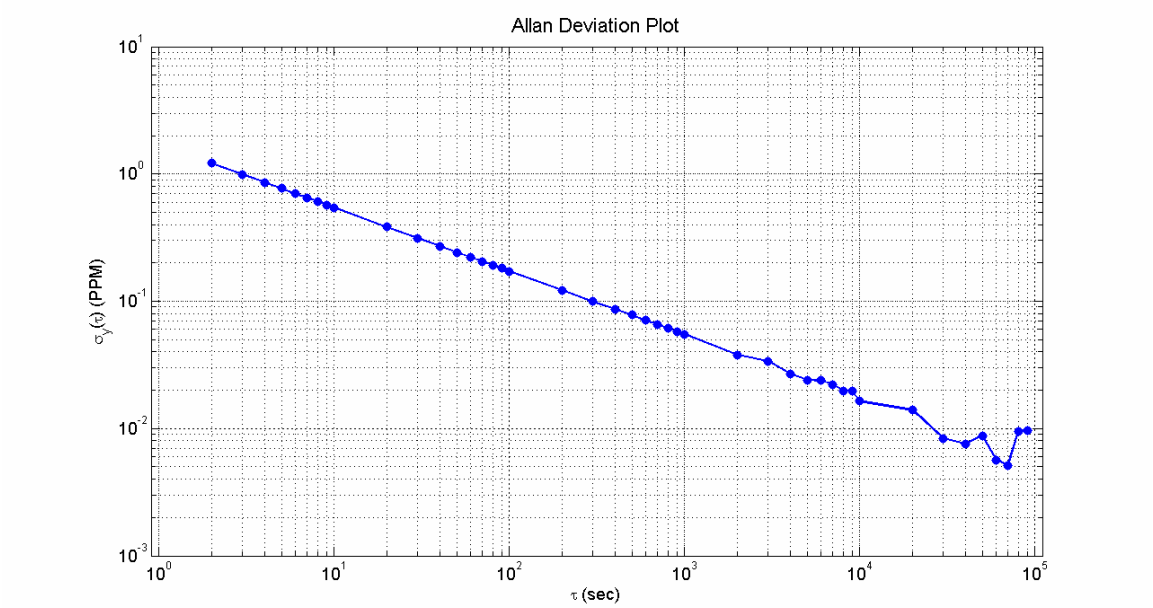

Figure 5.9 Allan deviation plot of TSC frequency measurements with $\tau_0 = 1$

To summarize the one week's worth of data, some classical statistic parameters can be calculated and is tabulated in Table 5-3. Even thorough the results indicate good frequency stability, such small variations in frequency does accumulate over time and propagate to the actual timekeeping accuracy. Nevertheless, the measurements indicate that the TSC counter exhibit good frequency stability characteristics when averaged over long intervals.

Table 5-3 Statistics of the TSC frequency measurements

	Offset from nominal frequency	Second to second difference
Number of data points	599384	599294
Mean	-54541	0.016
Median	-54592	0
Standard Deviation	1240.7	1937.7
Variance	1539442	3754702
Minimum	-71976	-35184
Maximum	-36792	25072
Range	35184	60256

In the practical sense, it is worthwhile to note the delay of the PPS reaching the computer. The delay outside of the computer is in the traversing of the electric signal in copper at about 5 nanoseconds per meter is fairly constant, thus has no effect on the drift measurement. Even thorough a real-time operating system is used here, there is still some

small variable internal delays introduced in the system. The difference between the signal arrival time and the execution of the counter read time is the interrupt latency and is largely dependent on the hardware being used. Such latency does contribute to the accuracy of the measurement. However, for the purpose of accuracy and stability analysis the relative effect of interrupt latency is minimized as the measurement time is extended.

5.5 Summary

The results presented here shows that the crystal oscillators of the Pentium class processor cores exhibit inaccurate but relatively stable frequencies. The same can be implied for other conventional oscillators. Hence, it is possible to leverage its high stability and high resolution counter for either timekeeping or timing measurements. There are several reasons for introducing the measurement system in this context. Firstly, the measurement system provides the theoretical basis for understanding the inaccuracies in the crystal oscillator as well as the high stability it offers. It was shown that smallest variations in the oscillator frequency occur when the measurement is averaged over longer time intervals. In effect, by averaging the measurement result over time, the influence of the white phase noise and flicker noise is minimized.

The concept introduced in this chapter can be applied to the FDR timing subsystem, where the processor oscillator frequency can be measured each second based on a high precision timing source. Furthermore, this study indicates that by averaging the measurement result over time, a more accurate estimate of the oscillator frequency can be obtained. The overall effect of the averaging is equivalent to filtering the short term instabilities of the 1PPS and as the averaging interval increases the filtering effect is pronounced. Ultimately, the Allan deviation analysis can be interpreted as a tool that calculates how accurately one can predict the occurrence of the next 1PPS and the results presented in this chapter can be incorporated into the design of the next generation FDR. Given its high computation capabilities and large memory size, the PC based FDR can easily record the frequency of the oscillator as measured by the 1PPS or some other form of frequency reference over time and the measurements can be averaged in appropriate

intervals to predict the arrival of the next 1PPS. With an accurate estimate of the oscillator frequency, the FDR sampling time can be more accurately divided within each second and hence resulting in more accurate sampling time.

In addition to the study of oscillator characteristics, it is intended that such measurement system to be used for the analysis of timing accuracies achieved by different timing mechanisms. Since there are a wide variety of precision timing sources available today, it's important to be able to accurately quantify the accuracy that is achievable for each timing source. What is more important is the ability to characterize the accuracy of frequency and angle measurement based on the sampling time. To this end the next chapter takes a closer examination at the FDR sampling time and addresses the sampling time error resulting from the oscillator imperfections.

Chapter 6 Analysis of Frequency and Phasor Angle Measurements Based on Timing of Conversion

6.1 Background

As it was shown in Chapter 5, the crystal oscillators tend to have good frequency stabilities in the short term but they are prone to drift in the long term. Furthermore, the frequency offset of the oscillators differs due to manufacturing defects and the slight variations in the cut of the crystal. Nevertheless, with the aid of a high stability frequency reference such as the GPS 1PPS, the frequency drift that's associated with any particular oscillator can be removed through either software or hardware implementation. In the case of the FDR, an internal processor counter is used to reset upon every rising edge of the 1PPS to synchronize with the UTC time for synchronized sampling. Throughout the years of FDR design refinements, the implementation of synchronized sampling has changed several times to improve the accuracy of the frequency and phasor angle measurements.

The method to which synchronized sampling is conducted has evolved over the years with the highest accuracy obtained in the second generation FDR [14]. Although there hasn't been any significant change in the timing subsystem hardware, consisting of a GPS receiver and some form of processor timer. The sampling pulses are generated by the pulse width modulation (PWM) subsystem of the processor and the PWM timing is based on the internal timer. Since the processor timer frequency varies according to the oscillator that is being used, it is practically difficult to obtain an oscillator with a frequency that is wholly divisible by the sampling rate. In the practical world, the division of the timer by the sampling rate would always end up with a remainder which would accumulate at the end of each second and causing sudden jumps in phasor angle measurements. Different methods were implemented in both the first and second generation FDRs to mitigate the effect of the remainder accumulation. As it was described in Chapter 2, the timing subsystem of the second generation FDR was calibrated against a PMU using phasor angle measurement results [14]. Assuming that the PMU has a more accurate timing subsystem and given the condition that both the

FDR and PMU are measuring the same voltage source, the trigger for conversion pulses of the FDR can be adjusted so that the magnitude of the phasor angle measurement matches that of the PMU. A linear fit method was used to find the length of the trigger for conversion pulse period in clock cycles which would minimize the saw-tooth error in the angle measurements [14].

Although laboratory measurements have confirmed the frequency and angle measurement accuracy of this implementation [14], there are still some drawbacks to the design. The main flaws being the lack of cross platform compatibility, consistency among different FDR units and flexibility. If the FDR DSP hardware were to be upgraded or changed to another processor, the FDR will need to be recalibrated against the PMU with the same linear fit method. Furthermore, if the sampling rate were to be changed the FDR will also need to be recalibrated. At last, inconsistencies may be introduced across different FDR units due to differences in oscillators caused by manufacturing imprecision, aging and temperature effects. Given the frequency differences from different oscillators, it is most likely that the calibrated PWM period would differ across different FDR units. However, the sampling clock in the second generation FDR uses a constant value for PWM period. Since the quality of the phasor angle measurements are directly related to the timing of conversion, the variability in oscillator frequencies can lead to discrepancy in frequency and angle measurements across different FDR units. This chapter addresses these issues by introducing a new clock division algorithm which does not only adapt to different hardware architectures but guarantees to produce the most accurate sampling time based on the underlying hardware.

Since there are numerous possibilities for the implementation of FDR precision timing source and the sampling time accuracy varies to a great extent, it is highly plausible to develop a model of FDR which can be used to evaluate the performance of frequency and angle measurement using different timing mechanism, whether it is based on network time synchronization, GPS receiver or any other form of precision timing source. Given a universal clock division algorithm which is platform independent, different models can be developed based on the timing of conversion from different

hardware and precision timing reference. This chapter proposes the methodology to develop such model based on the measurement of the absolute second as referenced by UTC time and the sampling time.

6.2 Clock Division Algorithm

In order to address the need for a common method of PWM generation that does not only yield the maximum frequency and phasor angle measurement accuracy for any clock speeds but also minimizes the remainder accumulation effect, it is necessary to develop an algorithm that varies the period width of the PWM based on the amount of remainder for each pulse period. The most significant advantage of developing such an algorithm is the high flexibility it provides. The algorithm implementation is hardware independent provided that there is a timer and some means of producing variable length pulses are available. In addition, the algorithm can be implemented for any clock speeds and theoretically provide the most accurate sampling time solely based on the timing for conversion.

The core of the algorithm for the clock division is shown in Figure 6.1 in a flowchart. The basic concept of the design is similar to what is being proposed in [5] and the basic objective is to vary the period of the trigger for conversion pulses over the course of the second to distribute the error associated with the imperfect division. Upon the acquisition phase of the FDR state machine, the length of the 1PPS will need to be measured by the internal timer of the processor to get an accurate estimate of the oscillator frequency. The averaging of the measurements over time should yield a more accurate estimate of the arrival of the next 1PPS. Depending on the hardware memory that is available, the averaging interval could vary and the longer averaging time should provide for more accurate estimate. Once an accurate estimate has been obtained, the FDR should enter the initialization state where the algorithm in Figure 6.1 can be used to produce the PWM timing signal for conversion.

As shown in Figure 6.1, the flowchart illustrates the logic to generate the PWM timing pulses upon the rising edge of 1PPS. Upon system start up, the variable storing the previous remainder value, CURR_REMAINDER is reset to zero and upon the rising edge of the 1PPS, the system will add the base remainder, REMAINDER to the current remainder that had just been reset. The base remainder is obtained by dividing the length of the previous 1PPS as measured by the internal timer or PPS_Count by the sampling rate. If the current remainder is equal to zero then the division is actually an exact integer multiple division. This situation would result in the current remainder being set back to zero and the VAR_PERIOD_LEN variable being set to the quotient of the original 1PPS period division. On the other hand, if the base remainder is not equal to zero, then the system would either set VAR_PERIOD_LEN to the quotient of the original 1PPS division, or simply add one to the quotient and set it to be VAR_PERIOD_LEN. The resultant PWM pulses will have a period length of either the default quotient obtained from the 1PPS division or one more clock cycle added to the default quotient. The condition to determine whether or not to add one to the quotient is dependent on the accumulation of the remainder. When the accumulation of remainder is less than half of the sampling rate, the period of the PWM will be set as default quotient as obtained from the division of 1PPS. On the other hand, when the accumulation of remainder is greater than the half of the sampling rate, the period of the PWM will be set as the default quotient plus one.

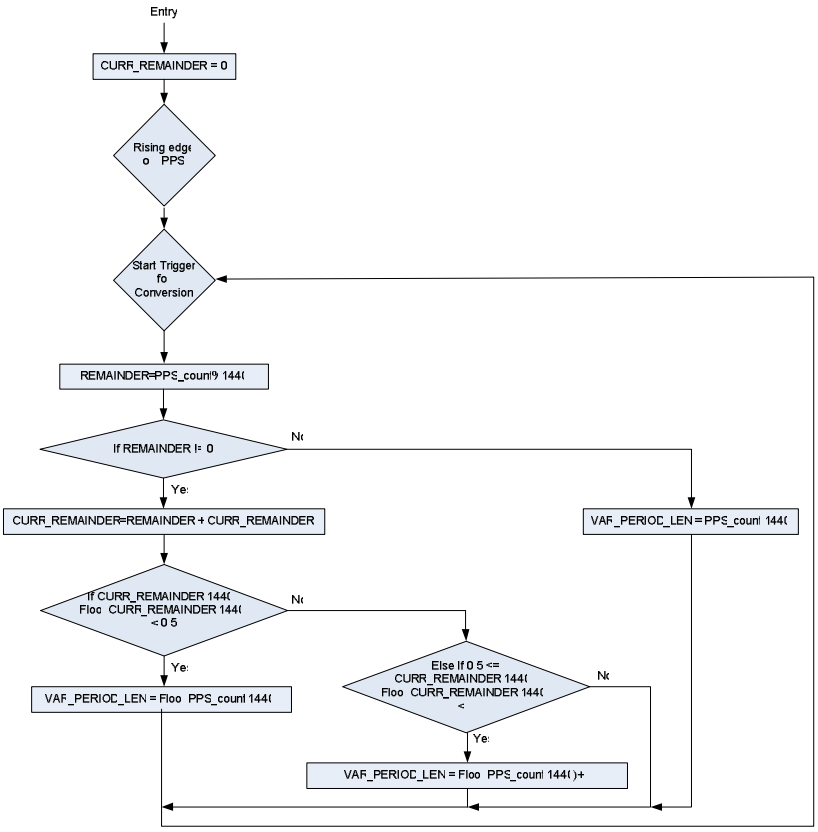

Figure 6.1 Flowchart of clock divider algorithm

The end result of this process is to not accumulate the error caused by the remainder but instead distribute it among the pulses. So when the remainders add up to equal or greater than the one pulse period, the current remainder is decreased by the value of the original remainder. By implementing such algorithm in the FDR, the accuracy of the frequency and phasor angle measurements will be heavily dependent on the clock speed that is used to generate the PWM, or equivalently the resolution of the clock. Lower clock speeds tend to introduce jitter that amounts to some significant effects in the frequency and phasor angle measurements. Since the smallest possible increment that can be made to the output pulse is 1 clock cycle, the minimum jitter would correspond to the length of the 1 clock cycle. For example, if the algorithm were to be implemented in the second generation FDR with the 30MHz DSP clock, the minimum jitter would be +/- 33 nanoseconds. This could be decreased by improving the system to operate from a higher speed oscillator. Another advantage of this algorithm is the incorporation of the oscillator frequency measurement each second. Since the existing clock division scheme does not measure the 1PPS each second and uses the nominal oscillator frequency to calculate the

timing for conversion, errors associated with the aging and long term drift of the oscillator are not being taken into consideration.

To better illustrate the performance of the clock division algorithm with higher speed clock, some numerical simulations can be conducted. A 60 Hz sinusoidal signal with 0 phase offset was sampled at 1440 Hz according to the sampling time calculated from the clock division algorithm and the resultant samples were input to the phasor analysis algorithm. Nominal clock speeds (f_{CPU}) ranging from 1MHz to 100MHz were used as the input to the clock division algorithm to sample the 60Hz signal. The nominal clock speed was incremented at a step size of 1MHz and for every clock speed there were 10 frequency and phasor angle points being calculated (10 frequency/angle output per second). Figure 6.2 and Figure 6.3 illustrates the simulation process for two different method of generating the sampling clock. Two sets of results are shown in the figures for comparison using the conventional clock divider where each sampling period is constant and rounded to the nearest whole number and the new clock divider algorithm where the sampling period is varied according to the accumulated remainder. With respect to the conventional method for clock division, the new clock division algorithm is shown to be effective in lowering the error caused by the timing residuals at then end of each second.

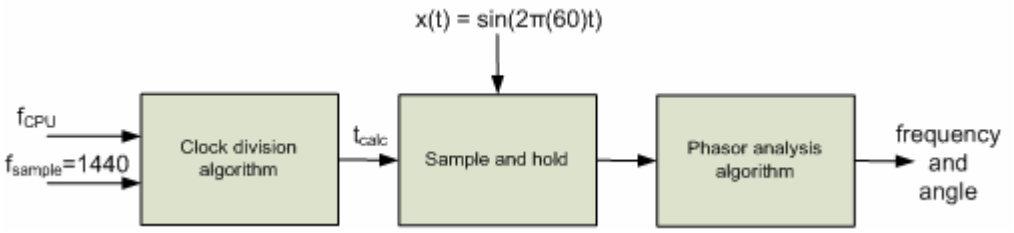

Figure 6.2 Simulation of clock division algorithm for frequency and phasor angle measurements

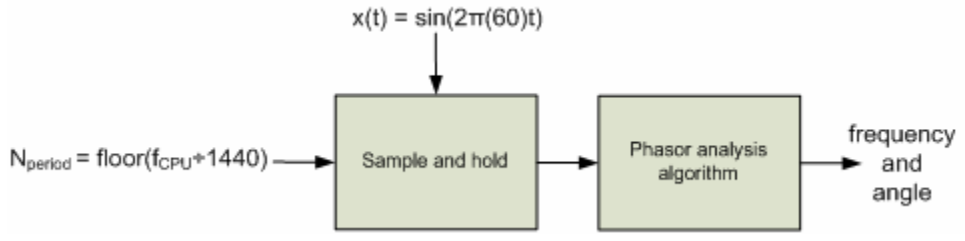

Figure 6.3 Simulation of conventional PWM method for frequency and phasor angle measurements

Since frequency is the derivative of phasor angle measurement, the root of the inaccuracies in frequency can be attributed to the angle measurements. Due to the timing remainders at the end of each second, the last phasor angle measurement of each second is erroneous and tends to have a large magnitude when the clock speed is low. In the case of the new clock divider algorithm, this is largely due to the jitter that is created as a result of adding a clock cycle to certain sampling periods, and low clock speeds tend to enlarge the jitter. The consequence of this error is a corresponding spike created in the last frequency measurement of each second, as well as an erroneous constant offset for the rest of the frequency measurements. As expected and illustrated in Figure 6.5, the magnitude of the supposed saw-tooth error in the angle data at the end of each second decreases as the clock speed increases. Such accuracy improvement is propagated to the frequency estimation as it is shown in figure 6.4 where the spike at the end of each second also decreases as the clock speed increases.

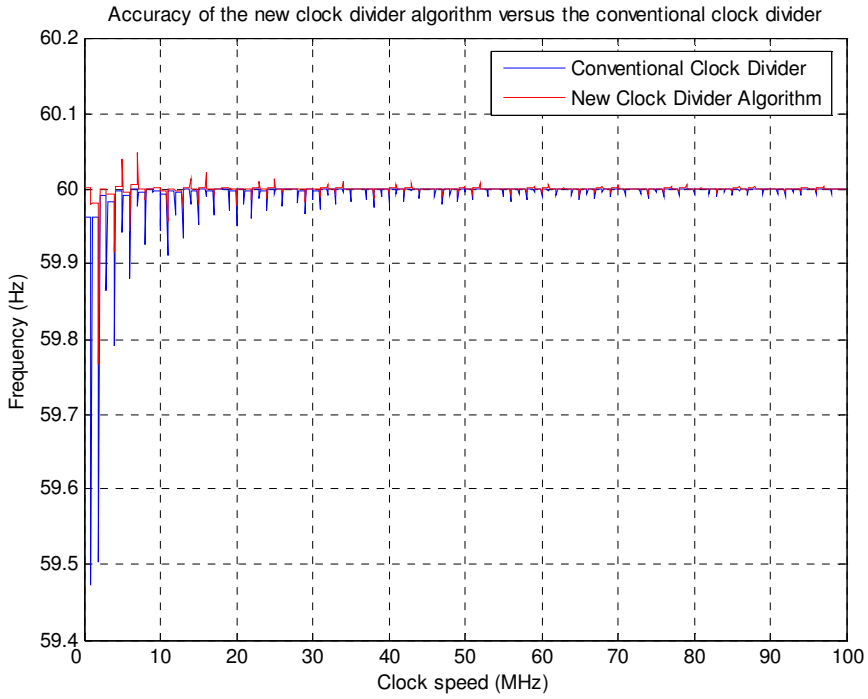

Figure 6.4 Effect of sampling clock speed on frequency estimation – clock division algorithm versus conventional PWM method

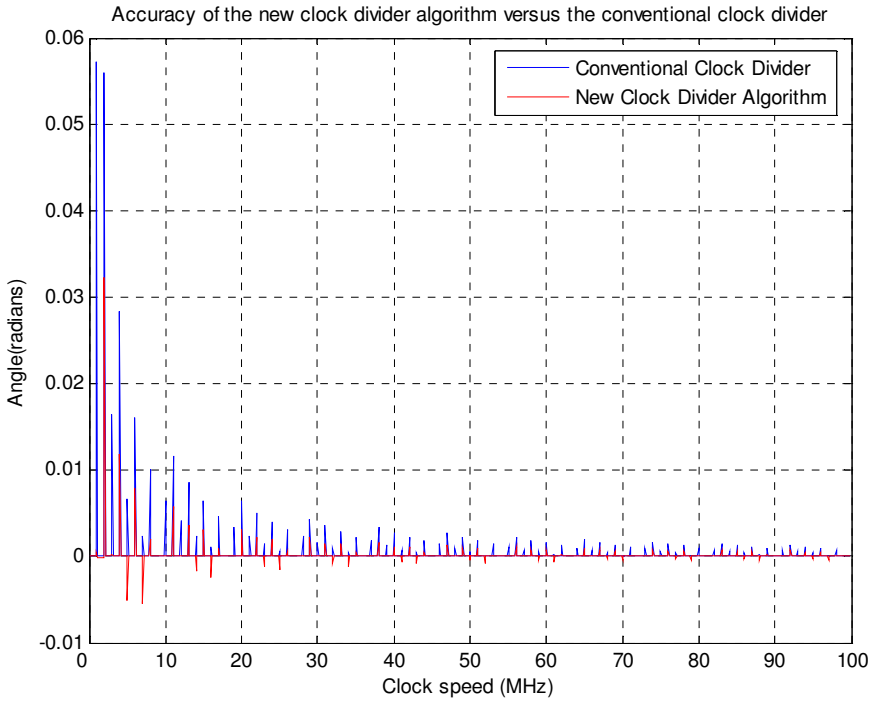

Figure 6.5 Effect of sampling clock speed on phasor angle estimation – clock division algorithm versus conventional PWM method

To illustrate the effectiveness of the algorithm for actual processor clock speeds, Table 6-1 lists two different clock speeds with two being the actual FDR processor speeds of 20MHz (MPC555) and 30MHz (TMS320LF270A). Also listed are the frequency and phasor angle results obtained by simulating the phasor algorithm with a 60Hz sinusoidal input with no phase shift. The algorithm provides accurate frequency and phasor angle measurement results for 30MHz clock speed, but appears to be affected by the relatively large jitter produced by the 20MHz clock as well as the remainder that is at the end of the second. The last phasor angle result deviates by 0.0031 radians, leading to erroneous frequency results.

Table 6-1 Frequency and phasor angle measurements using the new clock division algorithm with actual processor clock (input 60Hz with no phase shift)

20MHz (MPC555)		30MHz (TMS320LF270A)	
Frequency (Hz)	Angle (rad)	Frequency (Hz)	Angle (rad)
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9980	0.0000	60.0000	0.0000
59.9760	0.0031	60.0000	0.0000

Hence it is shown that the clock divider algorithm is theoretically sound in the cure of the accumulating sampling period remainders and it is flexible in the sense that it can be re-used for any clock speeds and provides the most accurate clock division for any particular clock speed. However, the development of such algorithm was not intended to replace the existing calibration method that was developed for the second generation FDR. Rather it can be used for the next generation FDR design whether it is microcontroller, DSP or PC based, and also serve as a basis for comparison of the maximum accuracy that could be achieved in different platforms.

6.3 Development of FDR Model

In the view of the fact that the clock division algorithm presented in the last section is accurate to below 1 clock cycle given any clock speed, it would be of interest to develop a simulation method based on a model of timing for conversion signal. With such a model, one can explore the accuracy of the frequency and phasor angle measurements under different scenarios where the timing for conversion is dependent on the underlying hardware clock and the precision timing source. Such model is useful in determining the accuracy of frequency and phasor angle measurement using different timing mechanisms. In addition, due to the inherent random jitter noise that is part of all digital electronics and the deterministic jitter noise that results from intentionally varying the period of

sampling pulse, it would be interesting to examine how this would affect the accuracy of sampling time and ultimately the resulting measurements.

6.3.1 Conceptual Design of FDR Model

The approach to developing the model starts with the high resolution measurement PC from Chapter 5. As it was shown, the measurement PC takes advantage of the high resolution and high stability TSC counter, which has an inherent timing resolution of below 1 nanosecond and high stability over long averaging periods. With such a high resolution timing measurement system combined with deterministic timing behavior provided by the real-time operating system, one may wonder the possibility of measuring the timing of the trigger for conversion signal in the FDR. Only the second generation FDR is considered here as it is the most accurate in frequency and phasor angle measurements. With its 30MHz clock, the DSP has a timing resolution of about 33 nanoseconds, which is more than ten times the magnitude of 1 clock cycle on the measurement system. Fundamentally, the high resolution counter combined with a real-time operating system such as RTAI enables the COTS PC to become a high performance oscilloscope data logger. The concept is highly plausible in many measurement application as the x86 based PC has a wide spread usage and the Linux/RTAI software packages are open source and royalty free.

Assuming that the voltage transformer and the input filter of FDR is perfect in the sense that they do not introduce any noise, the timing of conversion and the quantization effects of the ADC would be the largest contributor to the sampled signal noise. The quantization effects are the errors predominately produced by rounding and truncation of quantization levels and the timing of conversion is associated with deterministic jitters produced from the intentional adding of clock cycles to sampling periods as well as random jitters from the imperfection of the electronics. Based on the assumptions, one can model the FDR using the ADC timing for conversion and quantization levels. Much of the modeling is from a statistical technique, where each parameter can be modeled by a random variable.

Since timing for conversion is the parameter of interest in this model, the quantization noise can be assumed to be uniformly distributed between -1/2 LSB (least significant bit) and +1/2 LSB. Such assumption is equivalent to as if the quantization noise is uniformly distributed which may not be valid at all times and is dependent on the amplitude the signal. Although a more accurate quantization model can be obtained by static testing method of injecting several different DC voltage levels to the input of the ADC and collect the digital data at the output to form noise histograms. With the data collected, a statistical model can be developed based on the histogram. Nevertheless, since the timing of conversion is the main parameter of interest, either a uniform quantizer is assumed or the quantization effect of the ADC can be totally neglected to observe the effect of sampling time error.

To model the timing of conversion, the measurement PC can be used to timestamp the trigger for conversion signal using its high resolution processor counter. Since the measurement PC has a much higher resolution clock comparing to the embedded DSP, the timestamps provided by the PC processor clock should be very accurate and reflect any inaccuracies in the lower resolution DSP clock. Ultimately, the model is based on any errors of the DSP clock with respect to the measurement PC clock. Therefore, the clock division algorithm will be used here to provide a common basis for the DSP and the measurement PC. Since the DSP operates in a much lower speed, it is expected that the clock division algorithm would provide more accurate timing on the measurement PC. Therefore, both platforms will use the same clock division algorithm to generate the timing for conversion and assessment can be made based on the differences between the two. Furthermore, since a reference time is still needed to measure the precise length of 1 second, the GPS 1PPS will also need to be time-stamped as well. Once the length of the 1PPS or $1PPS_{length}$ is given in processor clock cycles, it is used in the clock division algorithm to mitigate the errors caused by imperfections in the oscillator. Hence, the timing results generated on the PC is considered to be much more accurate than that of the FDR and is considered to be the theoretical timing for conversion. Given the theoretical sampling time, t_{calc}, a histogram can be generated by subtracting the measured

sampling time, $t_{measured}$, from the theoretical sampling time. In summary, the process for developing the sampling time histogram is illustrated in Figure 6.6.

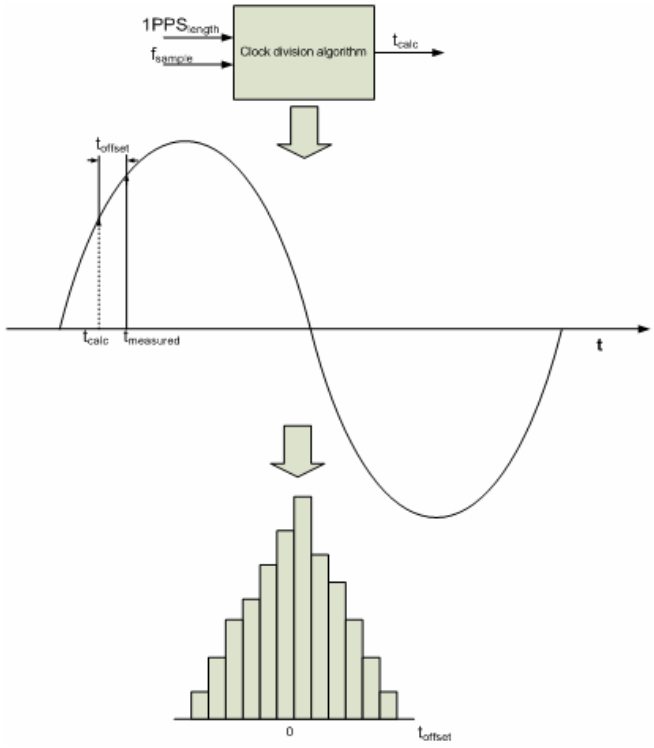

Figure 6.6 Illustration of developing sampling time histogram

Since the measurement is conducted using a higher speed clock, the raw measurements will need to be quantized or scaled to its equivalent for the lower speed clock. In this case, the scaling factor can be obtained by dividing the nominal frequency of PC clock by the nominal frequency of the DSP clock. Hence, the raw measurement data based on the PC clock can be scaled by dividing the scaling factor. The scaled results should form another histogram based on the clock cycles of the DSP clock. Finally, the histogram based on timing conversion will need to be normalized where the y-axis is the probability of occurrences and the x-axis is the error with respect to the theoretical value in number of clock cycles. The conversion involves taking each value in the histogram and divide by the total number of samples, which results with values in between 0 and 1, and the sum of all of the values in the normalized histogram will be equal to 1. Hence, the probability of obtaining any value is easily obtainable using the

normalized histogram. Then it is necessary to sum the probabilities to create the cumulative histogram. Figure 6.7 summarizes the process for developing the FDR model.

Figure 6.7 Procedure for developing FDR model based on sampling time measurement

In order to sample randomly from the histogram, a random number generator is needed. In this case, the Matlab uniformly distributed pseudorandom number generator can be used to generate numbers in the range of 0 to 1 to represent the probabilities. Given the probabilities generated from the pseudorandom number generator, the corresponding value in clock cycles can be obtained based on the cumulative histogram. Since a discrete random variable is considered here, it is necessary to apply the condition illustrated in Figure 6.8 where k will be used if M(k-1) < randn() ≤ M(k), where randn() is the function that provides the uniformly distributed pseudorandom numbers.

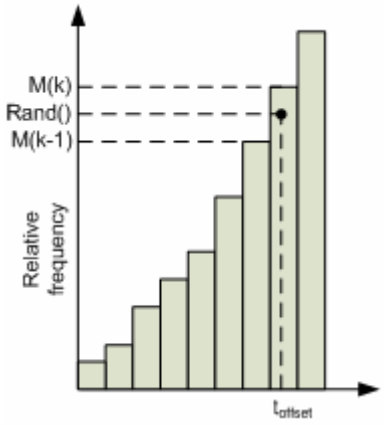

Figure 6.8 Illustration of random sampling

As a result of the random sampling process, t_{offset} can be obtained and applied as an offset to t_{calc} in the actual simulation as it is shown in Figure 6.9. The resulting $t_{simulate}$ is used to sample any arbitrary waveform and generate discrete values to input into the phasor angle analysis. Finally, the frequency and phasor angle results forms another histogram based on the difference between the theoretical and simulated values. Ultimately the final histogram should provide much information on the accuracy of the

frequency and phasor angle estimation based on the sampling time. The random sampling process should be performed many times to approach the point where all of the data in the histogram has been sampled. Such methodology is very similar to the classical Monte Carlo method, which shares the similar concept of modeling systems with uncertainties in inputs.

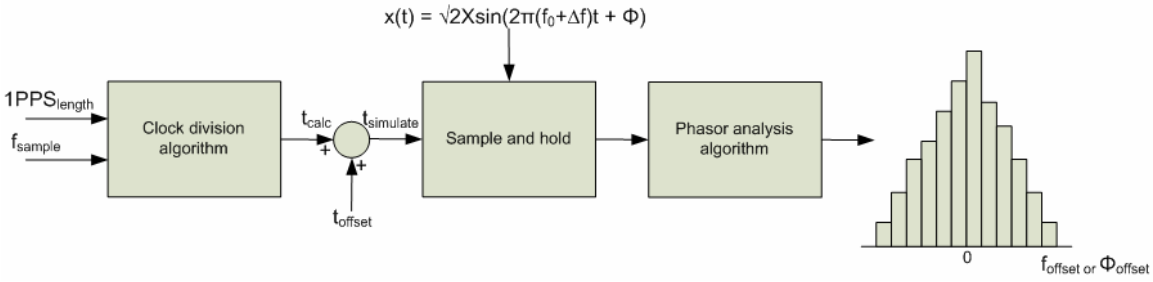

Figure 6.9 Procedure for simulating FDR model based on sampling time

6.3.2 Measurement of FDR Timing for Conversion

Since the measurement is conducted on a device external to the measurement PC, there needs to be some way of interfacing to the external signal. The trigger for conversion signal that is generated from the DSP is connected to the CONVST' input of the ADC, or the convert start input pin. A low to high transition on this input starts the conversion process and it is TTL (transistor transistor logic). Considering that the parallel port of the PC is also TTL and given its versatility in interfacing with external peripherals, it is sometimes being used for data collection, testing and control systems. Although the original intent of the parallel port was to interface with printers but it has evolved to perform sophisticated PC control applications, such as multiple stepper motors control. [A parallel port interface circuit for computer control applications involving] Given these justifications and the simplicity in implementing parallel port interrupts, the parallel port is used to capture the CONVST' signal. In the x86 systems, the parallel port interrupt is assigned to IRQ 7 and is triggered by rising edges in the ACK (acknowledge) pin (refer to Appendix C for schematic). Therefore, upon every low to high transition of the CONVST' signal, the system enters the ISR where the TSC counter is being read and logged into the kernel log. Furthermore, since the reference timing for conversion is

based on the 1PPS signal from the GPS, the same interfacing software that was implemented in Chapter 5 can be re-used here to provide the start of the sampling reference. Figure 6.10 shows the measurement setup for FDR trigger for conversion signal.

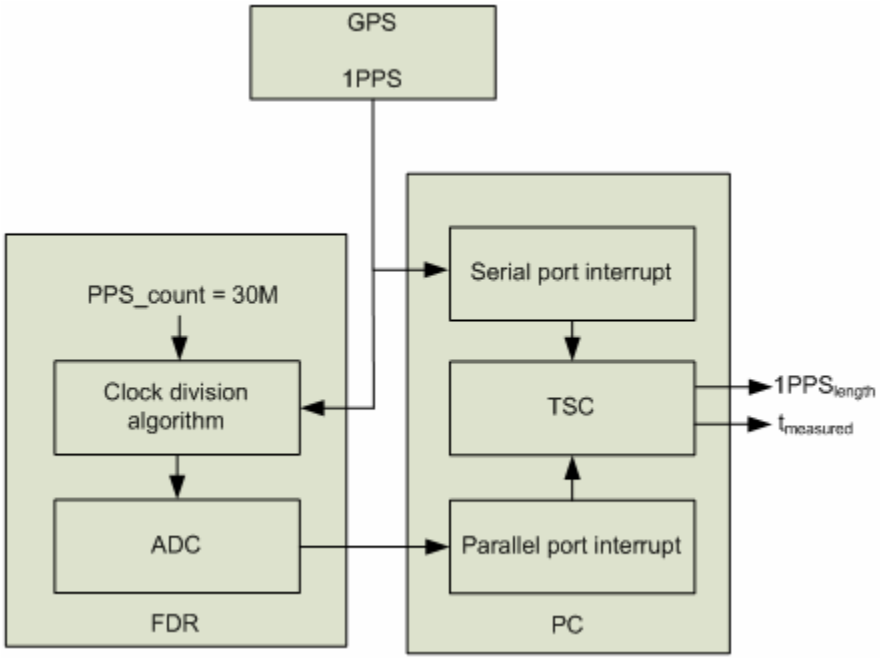

Figure 6.10 Timing measurement setup for FDR trigger for conversion signal

Due to the fact that the TSC counter is started when the machine boots up and will not rollover in the time span of the measurements, a post processing tool was developed in C# to extract the data from the kernel log and convert the TSC data point to counts with respect to the start of the second as indicated by the 1PPS. Also, given the length of the 1PPS as measured by the TSC, the C# program will calculate the timing of the trigger for conversion with respect to the start of the second using the clock division algorithm. By subtracting the actual timing for conversion as measured by the PC from the 'perfect' timing derived from the C# program, the error associated with the DSP clock is obtained.

Figure 6.11 shows the histogram representing the difference between when the actual trigger for conversion occurred and the calculated timing, in clock cycles and its equivalent in seconds. With a sample size of 1000000 data points, it appears that most of

the data points are concentrated within +/- 20000 clock cycles, or about 12 microseconds. Considering that the 30MHz DSP clock has a resolution of about 33ns per clock cycle, it is clear that the measurement result is diluted with much larger errors. Much speculation was placed on the performance of RTAI and whether it was lacking the real-time behavior that is expected of all real-time operating systems.

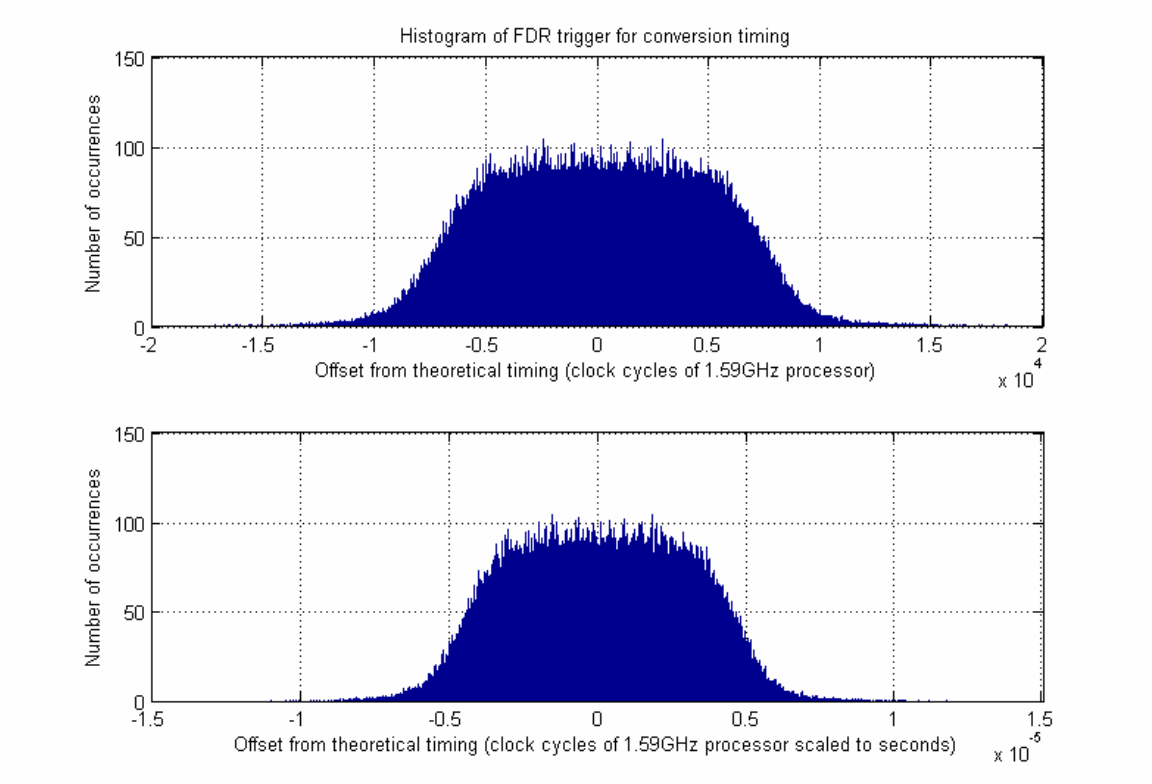

Figure 6.11 Histogram of timing measurements for FDR trigger for conversion – offset from theoretical timing

Taking another perspective in examining the measurement results, if the measurement were ideal, the period difference between each of the trigger for conversion pulses should always be less than 1 clock cycle on the 30MHz DSP clock, or equivalently 53 clock cycles on the 1.59GHz PC clock. Figure 6.12 shows the period differences in the trigger for conversion pulses in clock cycles and seconds. Once again, the results indicate otherwise as the differences between pulses periods exceed +/- 10000 clock cycles on the 1.54GHz PC clock, or equivalent to about +/- 8 microseconds. Since the histogram is approximately bell shaped and resembles a normal distribution, one may attribute the

error to system noise. To confirm, direct comparisons can be made between the pulse to pulse period differences and the second to second frequency difference from Chapter 5. Specifically, the measurements taken here is very much similar to that of Chapter 5 where the length of the 1PPS is being measured. The only exception is the differences in the frequency of the PC oscillator and the FDR sampling rate. Nevertheless, in comparing Figure 6.12 with Figure 5.6 from Chapter 5, there is no discernable differences between the two measurement results in terms of the distribution of the data.

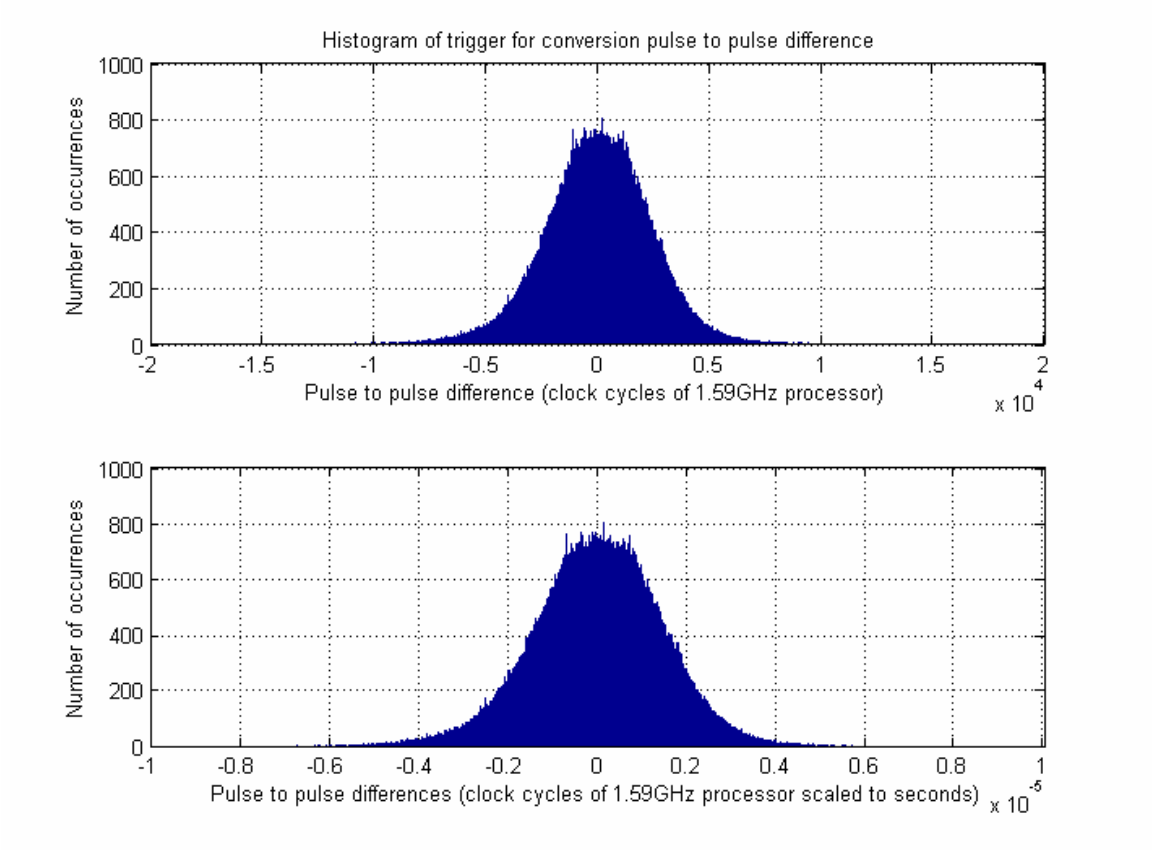

Figure 6.12 Histogram of timing measurements for FDR trigger for conversion - pulse to pulse timing differences

A better representation of the comparison between the two is shown in Figure 6.13, where the Q-Q plot for both trigger for conversion pulse to pulse period difference and 1PPS second to second frequency difference is shown. Both data sets can be approximated to the normal distribution within +/-2 standard normal quantiles. Then the tails on the two sides deviates away from the normal distribution at about the same position. The result indicates that the jitter noise associated with the 1PPS measurement

is very similar to that of the trigger for conversion pulse measurement. Hence, the performance of the measurement PC is consistent across different measurements and the frequency difference between the two signals that are measured do not seem to make any significant differences in the jitter noise. Nevertheless, it is important to recognize that the accuracy of the FDR model is dependent upon the accuracy of the sampling time measurements. The measurement from Chapter 5 is different in this aspect because statistical tools such as the Allan deviation and averaging can be used to identify and mitigate the effect of jitter noise.

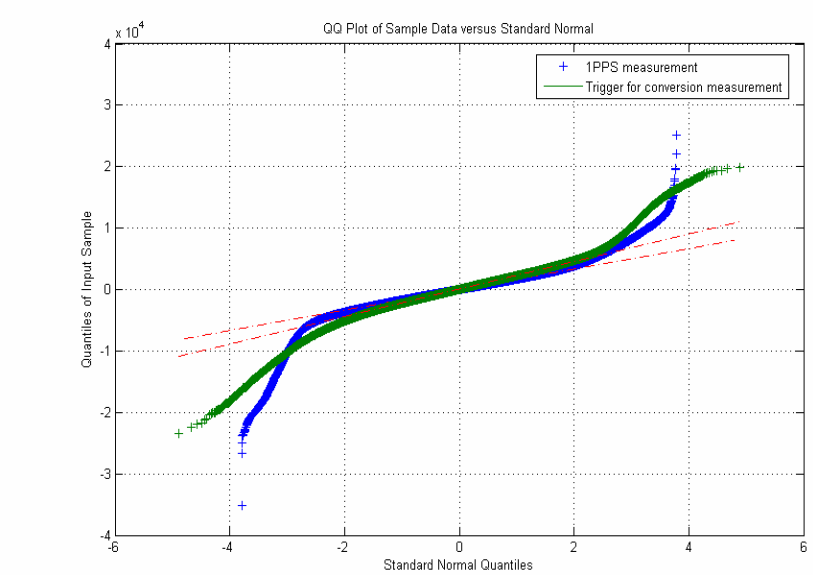

Figure 6.13 Q-Q Plot of 1PPS measurement versus Q-Q Plot of trigger for conversion measurement – pulse to pulse timing difference

Although the jitter noise in the microseconds range is a relatively large contributor to the inaccuracies of the measurement PC, the interrupt latency is yet another concern for accurate measurements. The first trigger for conversion pulse with respect to the 1PPS can be examined to approximate the effect of interrupt latency. At every rising edge of the 1PPS, the DSP loads the period of the trigger for conversion into the PWM period register and the first rising edge of the trigger for conversion should occur within one or a few clock cycles in the worst case latency. Since both the 1PPS and trigger for conversion timing information is available, the interrupt latency effect can be approximated. Figure 6.14 shows the measurement of the delay of the first trigger for conversion pulse with respect to 1PPS, in clock cycles and scaled seconds. The average

delay in the arrival of the first trigger for conversion pulse with respect to 1PPS is about 2500 clock cycles in the 1.59GHz clock, or about 16 microseconds. That's more than 400 times the length of 1 clock cycle in the 30MHz clock, signifying a relatively large latency in comparison with what was predicted.

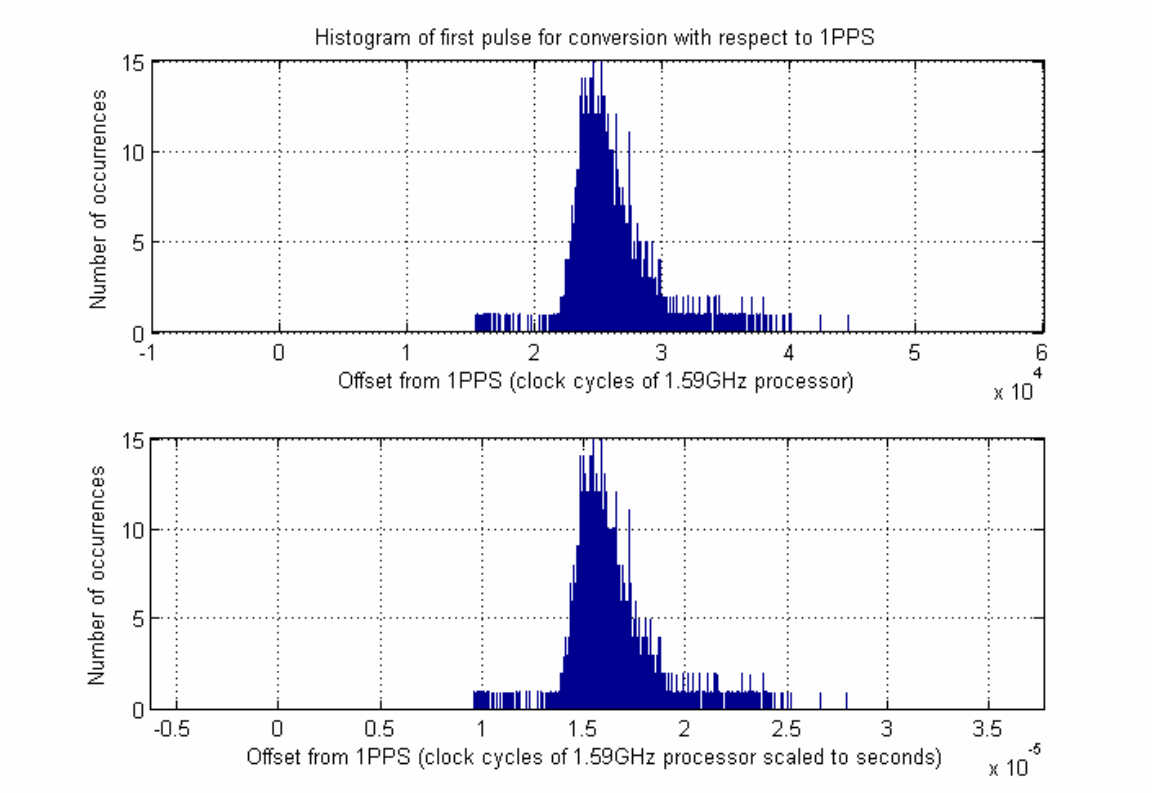

Figure 6.14 First trigger for conversion latency with respect to 1PPS

The original intention of this effort was to accurately measure when the trigger for conversion occurs in each second and develop the model based on the difference between the measured value and calculated value. The concept of the higher timing accuracy can be obtained when the resolution and the precision of the measurement system timing is higher than that of the DSP. However, the results show that given the approximate timing behavior of the DSP PWM, the measurement PC is not capable of accurately depicting when the trigger for conversion actually occurred. On the other side of the spectrum, it is known that dedicated processors such as the DSP has the capability of guaranteed interrupt latency in the range of a single or in the worst case a few execution cycles. Nevertheless, it was shown that the jitter noise is consistent between the measurement of trigger for conversion signal and the 1PPS from Chapter 5. Hence, it is clear that the error

is not associated with the way that the ISR was implemented, but rather it is indicative of the imperfections in the real-time operating system or the underlying hardware. Finally, Table 6-2 shows the statistics of the measurement in both clock cycles and microseconds.

Table 6-2 Statistics for the measurement of the trigger for conversion signal

	Offset from calculated values	First pulse delay wrt. GPS 1PPS	Pulse to pulse period difference
Number of data points	1000000	7504	1000000
Mean	114 (~71.45 ns)	2556 (~16 us)	-0.00754 (~ 0 us)
Median	116 (~72.71 ns)	2521 (~15.8 us)	40 (~0 us)
Standard Deviation	4453 (~2.79 us)	2290 (~14.3 us)	2466 (~1.54 us)
Minimum	-18404 (~-2.13 us)	15360 (~9.63 us)	-2340 (~-1.46 us)
Maximum	19132 (~11.99 us)	44540 (~27.9 us)	1989 (~1.24 us)
Range	37536 (~23.5 us)	29180 (~18.29 us)	4329 (~2.71 us)

6.4 Discussion of Results

Although it was verified that the measurement results indicate a relatively large magnitude jitter noise and latency which is consistent throughout all of the measurements, it is important to characterize the underlying limitations of measurement system and the sources of inaccuracies. The original motivation for this work was the observation that the use of general-purpose microprocessors, such as the Intel Pentium, is increasing for real-time applications [78]. The reasons for this increase are the readily available TSC counter and the convenience of complete PC compatible platforms that provide abundant peripherals at low cost. However, the results indicate that there are limitations to such implementation rooted in either the real-time operating system or the underlying hardware. At this point it is worthwhile to revisit the topic of real-time operating system performance evaluation and the possibility of improvising the existing system.

Literature such as [78] indicates that despite the best efforts to make RTOS deterministic, the underlying hardware in general-purpose computers introduces timing uncertainties due to microprocessor and bus effects. While there is a large amount of literatures on the research of RTOS and real-time programming, real-time software by

itself does not guarantee a deterministic system. Microprocessors such as the Pentium contain some optimization features which interfere with the deterministic capabilities of RTOS. These optimization features include instruction and data caches, instruction pipeline and speculative execution. These features are often incorporated into the processor architecture to significantly speed up the average execution times, but occasionally introduce large delays.

In the case of the data cache, copies of data from external memory are kept in the internal processor cache, reducing the time for subsequent access. However, if the cached data is replaced and is referenced again, it must be fetched again. [78] indicate that in an environment where real-time and non-real-time tasks share the processor, it is practically impossible to prevent real-time task data from being replaced occasionally, even if the non-real-time tasks run at the lowest priority. Furthermore, an increase in processor speed leads to more profound effect in mixing the real-time and non-real-time tasks. The faster the processor, the more non-real-time task code can run during the idle period and interfere with the determinism of the real-time task. Nevertheless, the common misconception is that disabling a microprocessor's cache would reduce timing uncertainty. But in reality the absence of a cache magnifies the uncertainties in instruction pipelining and speculative execution. Even if the timing uncertainty was eliminated by disabling the cache, the performance in speed would be penalized significantly.

It is also worthwhile to address the issue with the interrupt latency as seen in the measurement results. Specifically, interrupt latency measure the ability of a system to respond to an asynchronous event and the system's ability to change the processor's state. In general, [73] indicate that the minimum and maximum latency measurements have virtually no relationship to the processor's clock speed or the processor model (Pentium versus Pentium 2). The minimum latency is often attributed to the I/O and memory bus performance where as the maximum latency is often attributed to the state of the machine at the moment the interrupt occurred. These are considered extreme cases and are dependent on a machine's bus speed, memory, peripheral devices and associated drives.

Overall, increasing a processor's clock speed increases the average performance level of a system but does not significantly affect the speed at which a processor will respond to an interrupt.

In contrast to general purpose processors, DSPs bypasses unpredictable features like caches and pipelines, opting instead for simple instruction sets that optimize commonly used instructions for speed. Such implementation allows the DSP to have guaranteed interrupt latency in the range of a single or few execution cycles. As a result, the measurement of the FDR trigger for conversion signal reflects a mixture of latency and jitter noise that are associated with the uncertainties of the Pentium processor, rather than the DSP itself. Many literatures present the evaluation of real-time operating system performances and provide actual figures in latency and jitter. However, since the machine under test varies in certain aspects such as configuration and hardware, the range of latency and jitter figure can vary by some extent. Nevertheless, RTAI is known to exhibit comparable latency and jitter to commercial real-time operating systems but is still susceptible to latency and jitter in the orders of anywhere from 1 to 10 microseconds for processors running at a clock rate of 100s of MHz [75]. Although [78] shows that the worst case jitter values can reach the orders of ten microseconds. The timing uncertainties in jitter are mostly related to the execution environment where other running tasks can provide interference and also the scheduling algorithm. Although it was indicated in the measurement of the FDR trigger for conversion that the estimated maximum latency exceeds 10 microseconds, but the average estimated latency indicates that the system is performing close to the optimal conditions. The same can be said about the estimated jitter which has a range of 2.71 microseconds, well within the bound of what is indicated in the literature. Nevertheless, even with the highest performance real-time operating systems, the accumulation of latency and jitter noise can easily reach within the microseconds range [74]. Ultimately, in order to achieve high deterministic timing that is needed to measure the DSP clock, it is necessary to acquire the use of a FPGA based counter [102][104].

6.5 Summary

In this chapter a new clock division algorithm is proposed for generating accurate sampling pulses using PWM. The existing method of producing sampling pulses is based on the calibration of the period of the FDR sampling pulses against the PMU. Although the calibration method is effective in increasing the accuracy of the frequency and phasor angle measurement, it is hardware dependent and susceptible to timing errors in the sampling pulses due to the inaccuracies of the oscillator. To address these issues, the proposed clock division algorithm keeps track of the accumulated timing error caused by the imperfect division and adds or subtracts one clock cycle to the next sampling period so that the sampling time error is always within one clock cycle and does not accumulate. This method is hardware independent and produces the most accurate sampling time for any oscillator frequency. This clock division algorithm can be integrated into the next generation FDR regardless of its hardware architecture. By the same token, it is just as important to emphasize that the measurement of the 1PPS is critical for accurate calculation of sampling time.

Given the clock division algorithm, one can measure the timing of the sampling pulses generated from oscillators of different frequencies and different precision timing reference. Examples of the precision timing reference include the indoor GPS and the network time synchronization. Given that these timing references provides different levels of timing accuracy and are difficult to characterize in terms of its influence on the frequency and phasor angle measurements, it is logical to develop a method to model the FDR based on the timing characteristic. The methodology is based on the measurement of the FDR sampling time and the absolute second given by the frequency reference. Although the timing measurement is carried out using the measurement system from Chapter 5, it is shown that the measured sampling time does not accurately depict the FDR. This is due to the optimization features of the Pentium processor which adds comparatively large jitter noise and latency to the measurements. Hence the measurement system is not suitable for this application and more deterministic hardware is needed to develop a more accurate FDR model. The measurement of the FDR sampling time has proven to push the measurement system to its limitations but these results does not

conflict with that of Chapter 5. In the measurement of the 1PPS, statistical tools such as averaging and Allan deviation are used to mitigate and distinguish the effect of jitter noise and latency. In the case of measuring the FDR sampling time, a large sample size is needed to model the FDR and carry out the random sampling. Ultimately, it would not be practical to apply data filtering in such application. More accurate model can be developed by means of more deterministic timing measurement system such as a dedicated hardware solution using FPGA.

Chapter 7 PC Time Synchronization

7.1 Background

Ever since the introduction of PC based FDR design, there is much optimism in using an Internet based timing synchronization technique to provide for frequency reference and timing reference for the FDR. Although GPS timing receivers has proven to be an effective frequency and timing reference, there are still many factors that would affect the availability of the GPS as outlined in Chapter 4. An indoor GPS solution is superior to that of the conventional GPS in terms of operating under environments which suffers from significant signal degradation. However, it's also shown that under significant signal attenuation environments the indoor GPS exhibit lower accuracy. The most significant advantage that the network time synchronization has to offer is the elimination of the extra cost and hardware that is associated with GPS receiver.

Part of Chapter 5 investigated the stability of the PC oscillator clock, or the TSC counter. It is shown that the TSC counter is relatively stable in frequency for a period of a week. Therefore a PC based FDR implementation would be able to make use of such a counter synchronized to a reference clock with UTC time to generate the trigger for conversion signal. Even though the PC software clock provides absolute time information and it can be retrieved by simple software instructions, its timekeeping accuracy suffers from instability in interrupt requests, limited resolution and inconsistencies due to power cycles of the machine. To improve the accuracy of the PC software clock and compensate for the inherent drawbacks in its timekeeping mechanism, it is necessary to investigate clock synchronization strategies and perform the appropriate analysis to quantify the achievable accuracy and the required trade-offs.

Clock synchronization can be implemented in either hardware or software. The synchronization technique to be used is heavily dependent on the required precision and the geographic spread of the distributed system. Dedicated hardware synchronization

provides the highest accuracy ranging in several nanoseconds. Although such implementation requires dedicated synchronization network and the appropriate phase locked loop (PLL) hardware with distributed devices that are located within a few meters of each other. Nevertheless, the hardware synchronization approach requires high density installation with high cost and its accuracy far surpasses that of what is needed for synchronized phasor measurements. Furthermore, many approaches to clock synchronization today are based on a hybrid solution combining software algorithm with moderate hardware support, which are capable of reaching microseconds accuracy in local area network (LAN). Finally, solely software driven clock synchronization algorithms use standard communication networks and send synchronization messages to get the clocks synchronized. They are not as accurate in comparison to the hardware and the hybrid solution but are more used due to the fact that many applications do not need as strict of timing requirements.

The major challenge to the WAMS time synchronization is the need for high accuracy synchronization in a wide area network (WAN). Such requirement is rarely addressed in many clock synchronization techniques as most of the distributed system applications are either LAN based or simply does not need as high of accuracy. Henceforth, this chapter seeks to address these challenges and characterize the accuracy of the one of the most popular network synchronization methods. Ultimately, recommendation can be made on whether network synchronization can be used for accurate frequency and phasor angle measurement applications.

7.1.1 Network Synchronizations

Generally, in network synchronizations there are at least two clocks with one being the local clock and the other being the reference clock. When compared to the reference clock A, the errors of a crystal oscillator based PC clock B, or in this case the local clock can be characterized by offset, skew and drift. Hence, a simple model of the local clock can be illustrated as:

$$C_B(t) = a_B t + b_B \qquad \textbf{Equation 7-1}$$

Where $a_B(t)$ is the clock drift, and $b_B(t)$ is the offset of clock B. Drift is often defined as the rate of the clock and offset is the difference in value from real time t. Using Equation(), a comparison can be made between clock A and clock B as:

$$C_A(t) = a_{AB} \cdot C_B(t) + b_{AB}$$ **Equation 7-2**

Where a_{AB} is the relative drift, and b_{AB} is the relative offset between the clock A and clock B. If the two clocks are closely synchronized, then their relative drift should approach 1, indicating both clocks are approaching the same rate and relative offset is approaching 0. Assuming a perfect clock synchronization scenario, the relative drift would be 1 and relative offset would be 0 indicating the same time for both clocks at that instant. In some literature, the phrase 'clock skew' is used in place of drift to indicate the difference between clock rates. Likewise, the phrase 'phase offset' is equivalent to offset and often used interchangeably.

The synchronization of multiple devices is the same as equalizing the computer clocks of different devices. Only by correcting offset of clocks is not enough for synchronization as clocks tend to have different rates so that drift between clocks occurs over time. As a result, the method of synchronization should correct for both the clock rates and offset to equalize the values for the clocks. The major challenge to precise network clock synchronization is non-determinism. Inaccuracies in latency estimates present major problems to asymmetric round-trip message delivery delays. Purely software implemented methods are often designed to run in asynchronous environments that do not have timing guarantees. One extreme example is the Internet where the performance of network synchronization is dependent on many random variables. Furthermore, software often runs on general purpose operating systems where determinism is almost non-existent.

Network synchronization can often be divided into two large groups, one being the peer to peer synchronization and the other being the client and server architecture where the reference time is obtained from one or more number of sources. Furthermore, most

systems are equipped with a pure oscillator with counter based clock where synchronization of the local hardware clock is not possible. Instead, logical clocks are introduced. Adjustments can be made to the logical clock by adding an adjustment term to the local hardware clock. The adjustment could be a discrete value obtained from each re-synchronization or a linear function of time. The discrete clock adjustment method may cause a logical clock to instantaneously leap forward or be set back and then continue to run at the speed of the underlying hardware clock. On the other hand, a linear function of time for clock adjustment prevents the sudden changes in the progression of the logical clock and is more accustomed to network synchronization with distributed systems.

The subject of time synchronization in sensor networks has only gained attention in the recent few years. Synchronization techniques include Reference Broadcast Synchronization (RBS) [94], Timing-Sync Protocol for Sensor Networks (TPSN) [95], Tiny-Sync and Mini-Sync [96], Lightweight Tree based synchronization [97], 802.11 synchronization [98], Precision Time Protocol (PTP) [101] and Network Time Protocol (NTP) [89]. Nevertheless, out of all of these synchronization methods, only NTP meets the basic requirement for wide area monitoring applications where the capability to synchronize with UTC time for a large geographical area is essential.

7.2 An Overview of Network Time Protocol (NTP)

For the past thirty years, Dave Mills' Network Time Protocol (NTP) has been the 'de facto' standard protocol for network time synchronization. It is the oldest continuously operating protocol in the Internet has undergone 5 different version releases (numbered 0 through 4). Almost all of the server machines in the world use it to synchronize to UTC. The basic concept of NTP is to synchronize the software clocks or system clocks of computers using messages transmitted over the Internet thereby increasing the accuracy of PC timekeeping [89]. It not only corrects the current time, it can keep track of consistent time variations and automatically adjust for system time offset on the client. Nevertheless, the actual variability in clock frequency is not being compensated. In the NTP clock synchronization, the clock consists of an absolute offset, called an epoch, and

a frequency ratio scale. The clock is defined as the epoch plus the hardware elapsed time multiplied by the frequency ratio. Finally, offset is the difference in epochs, skew is the difference in frequency and dispersion is the known error in the local clock.

NTP is built on the Internet Protocol (IP) and User Datagram Protocol (UDP) but it can also adapt to other protocol packages [89]. The main principle behind the NTP protocol is a client/server engine that stamps messages when sent and received. A message transmitted from a host returns with three timestamps in its packet body including originate, receive and transmit timestamp, and a fourth stamp separate from it. Received messages are automatically replied and sent messages are periodically initiated according to local state information. As a result, both the server and the peer can independently calculate delay and offset using a single message stream. This method is advantageous in the sense that the transmission times and received message orders are not important and it does not require reliable delivery. Rather, the accuracies are dependent upon the statistical properties of the outbound and inbound data paths [89]. Figure 7.1 shows the NTP message format.

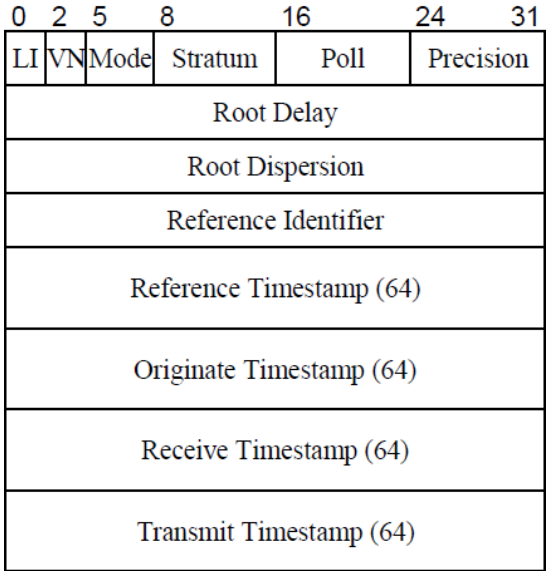

Figure 7.1 NTP message format

In general, NTP reads a remote clock by sending a NTP message to the remote node and waits for a reply in a later time. Upon receiving the reply message, the client/server

engine combines the incoming packet with the current local time and processes it. The message is stamped at each transmit and receive point as shown in Figure 7.2.

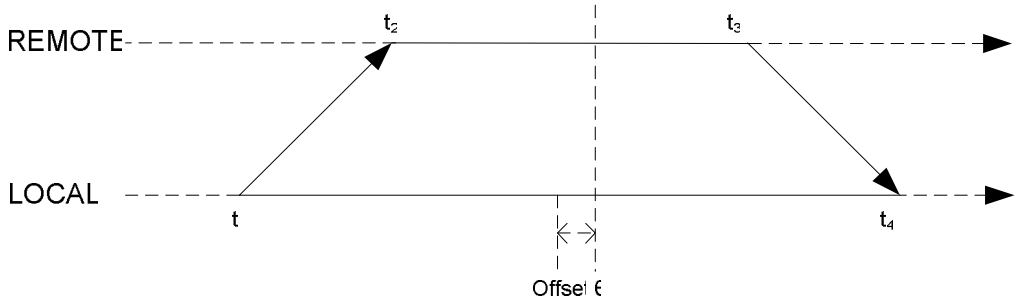

Figure 7.2 NTP message exchange

The four timestamps are used to compute round trip time and offset measurements. Assuming that the timestamp is accurate, the round trip time, δ can be computed from Equation 7-3. The clock time is defined as the average over an interval. Such is the case for both local time and remote time estimation as shown in Equation 7-4 and Equation 7-5. Furthermore, offset θ is defined as the difference between the remote clock time and the local clock time, where positive offset indicate the remote clock is ahead of the local clock and vice versa for a negative offset. Equation 7-7 illustrates the calculation for offset [87].

$$\delta = (t_4 - t_1) - (t_3 - t_2)$$ **Equation 7-3**

$$t_{local} = \frac{(t_4 + t_1)}{2}$$ **Equation 7-4**

$$t_{remote} = \frac{(t_3 + t_2)}{2}$$ **Equation 7-5**

$$\theta = t_{remote} - t_{local}$$ **Equation 7-6**

$$\theta = \frac{(t_2 - t_1)}{2} + \frac{(t_3 - t_4)}{2}$$ **Equation 7-7**

At this point, it is important to note that NTP is making several underlying assumptions as the offset is being calculated. Firstly, the clock time varies linearly during the exchange which may not always be the case due to factors such as oscillator instabilities and error in reading the clock. Second, the path from transmitting and receiving the message takes an equal amount of time. Such assumption can be made invalid purely due to differences in network delays. Ultimately, the clock value is a linear interpolation of the send and receives times of the message [89].

The NTP network is composed of primary and secondary time servers, clients and interconnecting transmission paths. A primary server is directly connected to a high accurate reference source, such as a timecode receiver or atomic clock. A secondary server provides time synchronization over path networks which may be shared with other services. Each time server is given by a number called the stratum, which reflects its distance from the reference clock.

Figure 7.3 shows a typical NTP network topology with atomic clocks and GPS timing receivers as the most accurate timing source. Any NTP server having as a time reference of stratum 1 server is categorized as a stratum 2. Any NTP server having as a time reference of stratum 2 server is categorized as a stratum 3, and so on. The stratum 1 servers are connected to the high accuracy atomic clock or GPS via an RS-232 cable or an IRIG-B (Inter-range instrumentation group) time code and they are considered to be primary servers. Strata 2-255 are considered secondary servers and their distance to the primary server are defined by their respective stratum number. Clients never communicate directly with a stratum 0 server and they always go through a stratum 1 server synchronized to a stratum 0 server. By arranging the network into stratum and allowing inaccurate higher stratum number servers to synchronize against a lower stratum number server, the demand on the NTP server and the network is minimized [89].

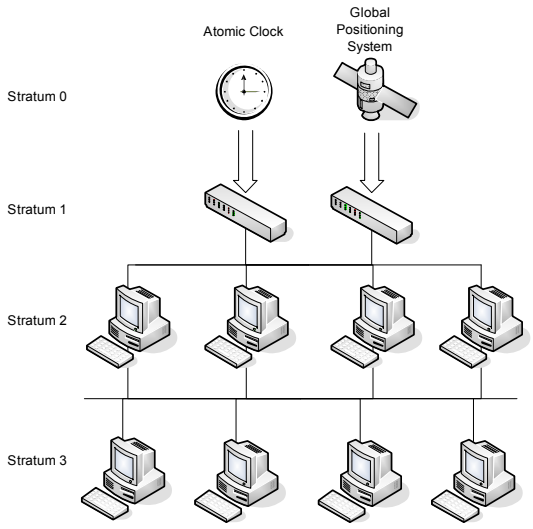

Figure 7.3 Typical NTP Network Topology

Due to network delays and traffics on the Internet, a stratum 2 time server will have anywhere from 10-100 milliseconds accuracy to UTC and each subsequent stratum time servers will add an additional 10-100 milliseconds of inaccuracy. Nevertheless, in practice it is rare to find clients with stratum numbers above 4 or 5 in most real-world configurations. As a matter of fact, according to a survey conducted by David Mills, more than half of the Internet connected NTP clients are in stratum 3, with almost all of the remainders in strata 2 and 4. In addition to the client and server model, the NTP servers operating on the same stratum can be associated with others in a peer to peer basis, so they may decide highest accuracy clock and then synchronize against that particular clock.

Ultimately, the goal of the NTP algorithms is to minimize both the time difference and frequency difference between UTC and the system clock. When these differences have been reduced to below a threshold of 128 milliseconds, the system clock is considered to be synchronized to UTC. Once the time offsets of the local clock is below 128 milliseconds, the local clock is being gently steered in small steps. For offsets larger than 128 milliseconds, the synchronization process may take a long time or never happens. Furthermore, if the clock offsets is greater than 1000 seconds, the algorithm would reset or the whole process reboots.

7.2.1 NTP version 4

Up to date, various versions of NTP exist, beginning with RFC (request for comments) 778 which included only data and packet formats and specification of server/client engine. Version 1 introduced the concept of hierarchical clock organization and a clock adjusting algorithm. Version 2 added authentication, control message option, and asymmetric modes of operation. It wasn't until version 3 that a significant progress was made in the clock synchronization algorithm. The most recent release of NTP is version 4, which have not been documented into RFC [88].

Version 4 of NTP addresses some significant shortcomings of the version 3 design. Most importantly the new implementation uses double precision data types throughout with the exception of the calculation of first order timestamp differences required to determine offset and delay. The time resolution is better than one nanosecond and the frequency resolution is better than one nanosecond per second. Additionally, improvements are made to the clock discipline algorithm where a true hybrid of frequency locked loop (FLL) and phase locked loop (PLL) is being implemented. The advantage to such a hybrid feedback control system is its capability to mitigate both network jitter and oscillator wander. Specifically, the FLL is more effective in minimizing oscillator wander effects whereas the PLL is more effective when network latency and jitter dominates [87].

In the previous version of NTP, the selection of using either PLL or FLL is based on the update interval τ with PLL being used when τ is less than 1024 seconds and FLL being used when τ is greater than 1024 seconds. Such scheme was implemented based on the Allan intercept illustrated by the Allan deviation plot from Chapter 5, where the oscillator stability is optimum, just before oscillator drift occurs. Version 4 of NTP combines both PLL and FLL in such a way that the FLL prediction is weighted more heavily when network jitter dominates and PLL prediction is weighted more heavily under conditions of oscillator drift. Ultimately, the new implementation allows for τ to be increased without losing significant accuracy and at the same time compensating for a wide range of network jitter and oscillator drift [88].

7.3 Evaluation of NTP accuracy

The literature on software based clock synchronization over networks is relatively limited. The original literature [89] contains much information on implementation but there is no formal analysis and no benchmarking study on the performance of the newly released NTP version 4.0. [20] indicate that the version 4 of the NTP can usually maintain time to within 10 milliseconds over the public Internet and can achieve accuracies of 200 microseconds or better in local area networks under ideal conditions.

Such figures are arbitrary since there are many different factors that would affect the accuracy of NTP. In general, reference clocks are assumed to be accurate and the accuracy of the synchronized clocks are judged according to how 'close' a clock is to the reference clock, the network latency to the reference clock, and the claimed accuracy of the clock. However, the network latency is practically uncontrollable and the NTP clock discipline seeks to minimize the errors as a result of this as well as the inherent instabilities of the crystal oscillator.

There are many publicly available NTP servers exist today and the most common route of obtaining accurate time is through those servers. Many organizations setup public NTP servers as a public service. However, setting up a large number of clients to use the public external servers is inefficient and a poor use of computing resources. Therefore, it is more reasonable to have a few servers receive accurate time readings from the Internet and use each of these stratum 2 or stratum 3 servers in localized areas. All the clients at the site can then receive updates from the stratum 2 or stratum 3 servers at the site. The alternative to the use of public NTP server is to setup reliable reference server dedicated for specific applications. Such method may be applicable to FNET since the FDRs are usually distributed in separate LANs and are dispersed throughout in the WAN.

In understanding the selection process for reference clocks, it is important to note that by synchronizing to a stratum 1 server does not necessarily dictate more accuracy comparing with that of synchronizing to a stratum 2 server. Selecting NTP source requires careful consideration of accuracy and reliability. In addition, it is always best to synchronize with multiple servers to mitigate the effect of incorrect or inoperational server. To ensure the performance of the reference clock, it is important to find a server that is peered with several other servers to provide robustness. The NTP protocol is designed as a hierarchy to prevent large numbers of clients from accessing the same primary time sources. Therefore it is always a good practice to synchronize to a stratum 2 server when there are a small number of clients in the network and only synchronize to stratum 1 server when the number of clients becomes large.

Although there is very limited literature on the performance of the version 4 NTP release, some factors that may affect its accuracy can be foreseen. It is most likely that the achievable accuracy is heavily dependent on the operating system being used due to the differences in the timer resolution of Windows and Linux based operating systems. Furthermore, it is known that the NTP accuracy is maximized when synchronization occurs in a LAN with a small number of routers and switches but rather difficult to characterize in the WAN.

Early measurements of the time offset were conducted on the Windows XP platform using the W32Time tool [17]. The W32Time Service is a fully compliant implementation of the Simple Network Time Protocol (SNTP) as detailed in IETF RFC 1769. The results indicate that the polling interval plays an important role in the accuracy of NTP time synchronization. Furthermore, it was shown that NTP version 4 is several orders of magnitude more accurate than that of Windows Time Service. The polling interval was configured for both automatic and manual polling intervals where manual polling mode allows for a fixed polling interval, and automatic polling mode is based on an algorithm where the polling interval is determined based on a balance between accuracy and network overhead. It was shown that with a fixed polling interval of 1024 seconds, or polling at about every 17 minutes yields the best results in clock offsets. Windows Time Service operating at a fixed polling interval of 1024 seconds yields a maximum clock offset of about 60 milliseconds. Nevertheless, another measurement conducted on NTP Daemon (NTPD) for Windows has yielded much better results with maximum offset of about 15 milliseconds as shown in Figure 7.4. Ultimately, short polling intervals update the parameters frequently and are more susceptible to jitter whereas long polling intervals may require larger corrections with some significant errors between the updates.

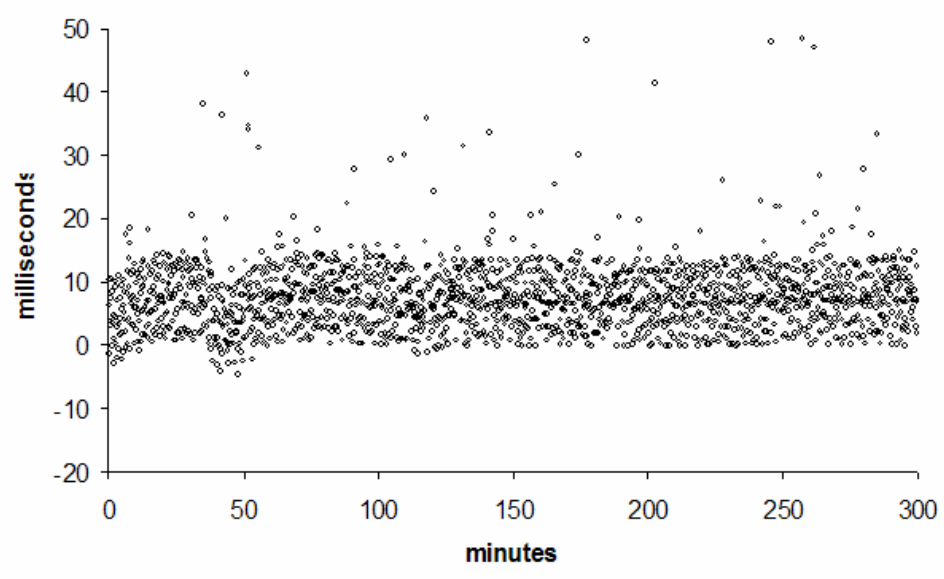

Figure 7.4 Local clock synchronized by NTPD – offset given by W32Time

The results shown in Figure 7.4 are rather biased as it is very much dependent on the resolution of W32Time tool. Since most of the Windows NT based platforms has an inherent timer resolution of 10 to 15 milliseconds and the W32Time tool is completely Windows based, it is very much limited to the Windows timer resolution. As a result, offset values lower than 15 milliseconds will not be able to be taken accounted for in the measurement and hence the measured offsets are very much bounded to 15 milliseconds.

In order to get a more precise measurement of the NTP version 4 time offset and be able to compare the performance between different operating system, it is necessary to turn to a more precise measurement tool other than what is provided in the Windows operating system. One of the most common methods to debug NTP is through the interpretation of the statistic log files generated. Specifically used for precision timing requirements, the loopstats (loopfilter statistics) file provide information on date, time, offset, drift compensation, estimated error, polling interval and timing stability. Parameters such as offset and drift are the measurement values as predicted by the NTP loop filter. The information can be extracted from the log file for post-processing and performance analysis. Also, since there is a readily available timing measurement system detailed in Chapter 5, it is worthwhile to exploit its high determinism and high resolution clock to be applied in timing measurement with NTP synchronization.

7.3.1 Characterization of NTP Time Synchronization on Different Operating Systems

To compare the performance of NTP in Windows and Linux, it is necessary to use a common measurement system with similar levels of precision in both operating systems. The loopstat file is available on all recently released versions of NTP and can be activated through the NTP configuration file. Since minimizing the offset between the local and reference clock offset is the main objective of NTP time synchronization, the scope of analysis will be limited to the offset of the local clock with respect to the reference clock. In addition, the reference servers that are being used were selected at random with the exception of the Virginia Tech servers, which are known to have the shortest network distance to the host PC and considered to be stratum 2 servers. The polling interval is set at a maximum of 1024 seconds to allow for a fair trade-off in fast synchronization and low network overhead.

Figure 7.5 shows the time offset of the local PC clock in a period of a week with respect to the reference servers (see Appendix B for the NTP servers), which include 4 NTP servers located at Virginia Tech. The plot illustrate that the clock offset is the largest at the beginning when NTP service was first started and over the course of about 3 days the offset is minimized due to synchronization.

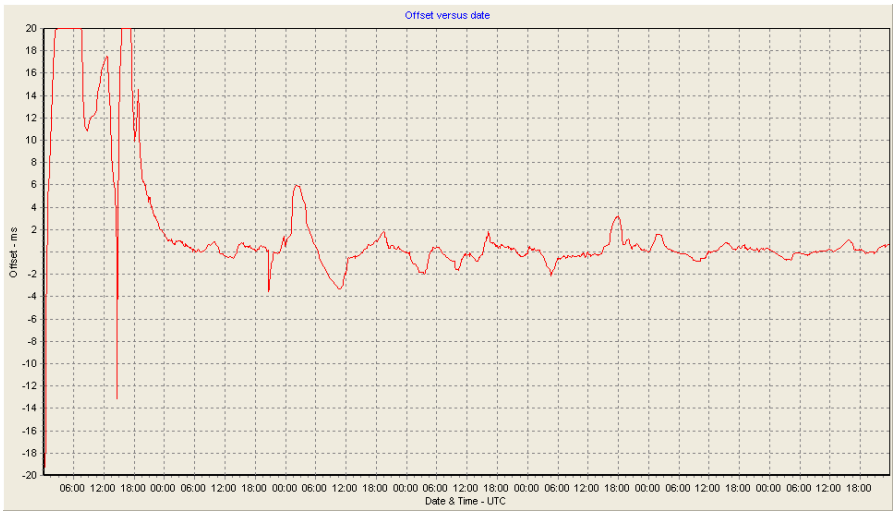

Figure 7.5 NTP local clock offset from server clock for Windows XP Pro. – first week of synchronization

The largest offset is around 46 milliseconds, which may be attributed to system reboot while the average offset over the week is around merely 1.6 milliseconds. Such accuracy is relatively high considering the low resolution of the Windows timer. Nevertheless, the newest version of NTP (NTPD) attempts to use the TSC counter to interpolate between the timer interrupts to obtain higher resolution timing and thereby increasing the accuracy of the offset. Therefore the results obtained here is far more accurate than the 15 milliseconds offset obtained from W32Time measurements.

Given the fact that the Virginia Tech NTP servers are close in network distance to the host PC, one may wonder the synchronization performance when such servers are removed. Therefore another set of data was collected with reference servers that are relatively far away in network distance (see Appendix B for the list of NTP servers). Figure 7.6 shows the time offset plot of the local PC with respect to the server clock, which does not include the Virginia Tech servers.

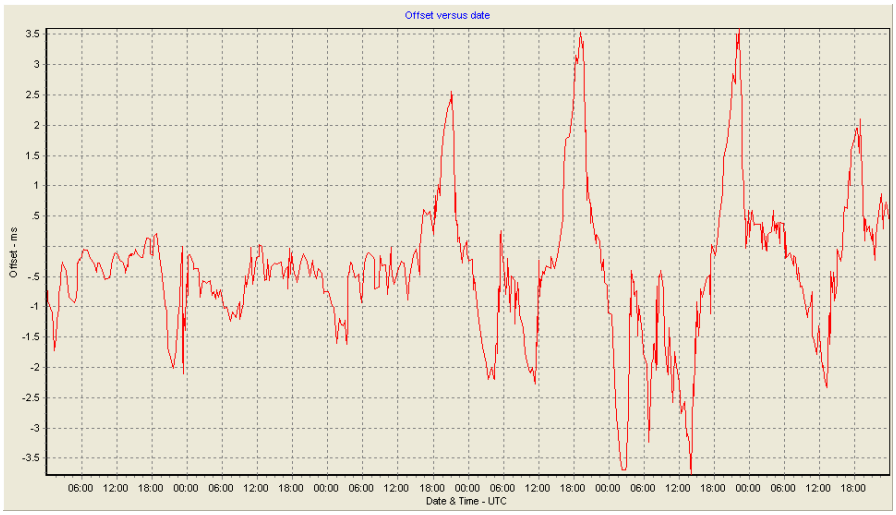

Figure 7.6 NTP local clock offset from server clock for Windows XP Pro. – second week of synchronization

Since the time offset was already in steady state from the previous week of synchronization, the beginning part of Figure 7.6 shows small time offset of less than 2 milliseconds but then the offset increased on the third day with two peaks of more than 3.5 milliseconds offset. This phenomenon may be attributed to synchronizing servers that are further away in network distance hence introducing longer latencies in the network.

Hence, the network distances of the NTP servers in the WAN may affect the accuracy of the time synchronization but the effect does not seem to be significant.

It is also worthwhile to note that the number of times for the host PC to poll the server is significantly different for the two sets of data. Such observed behavior is due to the nature of the NTP algorithm, which allows for more frequent polling when the time offset is large and less polling when the time offset is relatively small. Since the second set of time offset data without VT NTP servers was collected right after the synchronization with the servers including the VT NTP servers, the offset of local clock is already small so there were less frequent polling of the servers. Considering that the second set of data was collected in 6 days and the first set was collected in 7 days, the average number of times that the servers are being polled each day was approximately 102 for the 7 days and 154 for the 6 days.

To examine the performance of Linux based operating system with NTP time synchronization, Ubuntu was used to synchronize with the same set of NTP servers (Appendix B including Virginia Tech servers) as Windows XP for comparison. In addition, the configuration for polling is the same as it was used in Windows XP with 1024 seconds maximum polling period. Figure 7.7 shows the time offset of the Ubuntu local clock offset with respect to the server for a period of 7 days. Similar to Figure 7.4, the largest clock offsets occurs at the beginning of synchronization, indicating the local clock is relatively far off from UTC time upon boot up. However, the beginning of the synchronization also exhibits a semi-transient response for a period of over one day and enters the steady state in the middle of the second day. In addition, the maximum offset is about -60 milliseconds and occurs at the very beginning of synchronization. Although the initial clock offset is relatively large compare with that of the Windows XP, Ubuntu still manages to obtain synchronization with the remote server on the second day as opposed to taking up to three days for the Windows XP. What's more important in this set of results is the verification that Ubuntu was able to obtain below 1 millisecond offset with respect to the server clock. Statistics for the one week's worth of data is tabulated in Table 7-1 along with the Windows XP results for comparison. It is interesting to note that

Ubuntu polled the servers only about half as much as that of Windows XP and obtained an average clock offset value of about a half of its Windows XP counterpart.

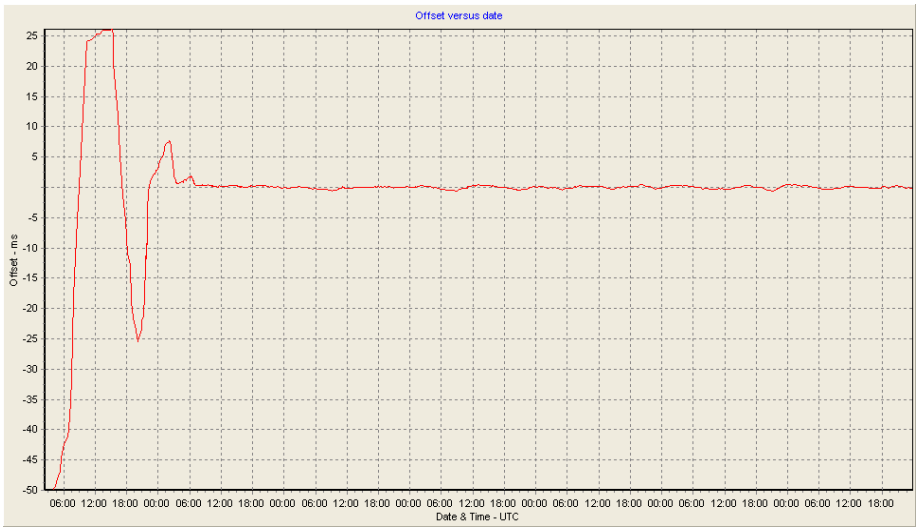

Figure 7.7 NTP local clock offset from server clock for Ubuntu – first week of synchronization

Table 7-1 NTP local clock offset statistics for the first week – Windows XP Pro. versus Ubuntu

	Time offset from remote server for Ubuntu	Time offset from remote server for Windows XP
Number of data points (number of polls)	546	1084
Mean	-0.8042 ms	1.689 ms
Median	0.0895 ms	0.2634 ms
Standard Deviation	9.82 ms	6.259 ms
Range	86.27 ms	72.66 ms
Minimum	-60.16 ms	-25.88 ms
Maximum	26.11 ms	46.77 ms

Since the time offset data illustrated in Figure 7.7 shows a relatively accurate clock with offsets below 1 millisecond at the end of the second day and throughout the rest of the week, it was decided to collect more data for another week to observe if NTP can maintain the synchronization. Figure 7.8 shows the time offset data of the second week in succession of what is shown in Figure 7.7. Also, statistics of second week's data are tabulated in Table 7-2. Again, lower number of polls is shown for the second week due to the smaller offset of local clock. Maximum deviation occurs at the first day of the week with about 0.63 millisecond offset, and then stays within the bound of about 0.5 milliseconds offset throughout the rest of the week.

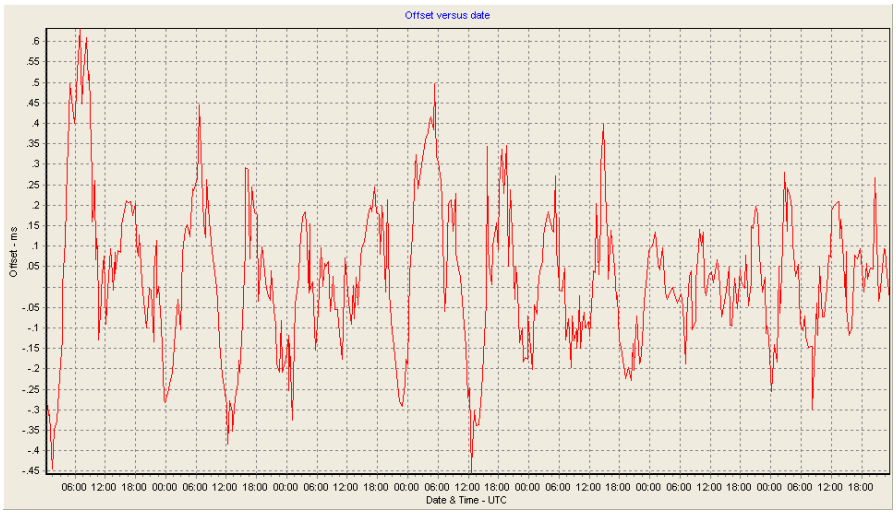

Figure 7.8 NTP local clock offset with respect to server clock for Ubuntu – second week of synchronization

Table 7-2 NTP local clock offset statistics for the first week – Ubuntu

	Offset from remote server for Ubuntu
Number of data points (number of polls)	425
Mean	32.26 us
Median	26 us
Standard Deviation	0.1854 ms
Range	1.09 ms
Minimum	-0.458 ms
Maximum	0.632 ms

Clearly, the measurement conducted using NTP version 4 is more accurate than the standard SNTP implementation in the Windows XP operating system. Furthermore, it was shown that the Linux based operating system Ubuntu exhibits better accuracy in NTP synchronization and synchronizes to the remote clock at a faster speed added with less server polling than its Windows XP counterpart. This is mainly due to the differences in timekeeping implementation of the two operating systems. The newer Linux distributions (version 2.4 and beyond) has a significant change in the timekeeping implementation. Specifically the frequency of the timer interrupt is increased from 100Hz to 1000Hz whereas the Windows XP timer interrupt is kept at 100Hz. In addition, a new abstraction layer called ‘clocksource’ was introduced. In this subsystem the operating system selects the hardware counter it considers the most reliable at boot time and

provides an interface to access it. Throughout the measurements conducted in this study, the TSC was selected by the operating system and deems to be the most reliable source.

Although not sufficient to address all needs, especially for high accuracy synchronized phasor angle measurements where a specialist solution such as the use of GPS receivers is required, NTP synchronization is a low-cost alternative that can be used for applications where the timing requirement is not as stringent. With less than 1 millisecond offset from the UTC time, it's possible to use NTP for frequency measurement where timing requirements are not as high as the angle measurement. As an example, the detection of sudden frequency excursions does not need accuracies in the microseconds range and NTP is more than adequate for such applications. Nevertheless, the main goal of NTP is to maintain bounded offset between the local system clock with the server clock, it does not put any emphasis in the correction of system clock skew or clock drift. The next section takes a closer look at the inherent TSC clock skew by measuring the TSC clock skew with respect to the GPS 1PPS.

7.3.2 Measurement of TSC clock skew

It was observed in the previous section that the NTP synchronization can keep the local clock offset to within 1 millisecond with respect to the reference clock. In this section, measurement is performed on the skew of the local processor clock with respect to the GPS 1PPS. Since a high precision measurement system was developed and characterized in Chapter 5, the same measurement system can be used to measure the TSC clock skew, where TSC clock represents the relative time obtained by scaling the TSC counts. The reason that the TSC clock skew is being measured here instead of the system time is due to the fact that real-time operating systems such as RTAI can not access the system time without breaking the real-time performance. However, one method of getting around this incapability is to have a soft time based scheduler running in Linux that would periodically synchronize the TSC counts to the Linux system time. By retrieving the Linux system time and the TSC counts in the soft task, one can easily scale the system time to TSC counts and then subtract the TSC counts that was retrieved to obtain the system boot time. The drawback to such implementation is that the soft task

is at the mercy of being preempted by the RTAI task, thereby losing synchronization at times with the system time. Also, with periodic synchronization to the Linux system time, it is expected that the rate of the clock will be fluctuating at relatively large magnitudes.

Given that the TSC counter has inaccurate but fairly stable frequencies and that it is used to interpolate between the timer interrupts in the NTP algorithm, it would be interesting to examine how the TSC clock compares with the 1PPS. The concept of the TSC clock is to simply scale the TSC counts to nanoseconds and the scaling process takes on the following form:

$$TSC_{ns} = \frac{TSC_{counts} \cdot 1000000000}{f_{CPU}}$$

Equation 7-8

Where f_{CPU} is the estimated processor frequency determined during system boot. Specifically, an attempt is made during every system boot to calibrate the processor frequency by comparing it with the PIT. The calibration result is in the format of TSC counts per microsecond. This 50 milliseconds calibration is coarse mainly due to I/O delays in accessing the PIT. As a result, the accuracy of timekeeping with TSC selected as the 'clocksource' is highly dependent on the accuracy of TSC frequency calibration. When the TSC is used for interpolation in between system timer interrupts, NTP either skews or set the system time to synchronize with the server time. However, since the initial calibration is susceptible to error, some initialization time needs to be resorted for NTP to estimate the skew value. Such phenomenon was observed in the previous section where there is a startup transient in the time offset when NTP was first started.

To measure the TSC clock skew, the same setup that was used in Chapter 5 is used here. The GPS 1PPS triggers the serial port interrupt and the system enters the ISR where the TSC count is being read and converted to nanoseconds based on Equation 7-9. Post-processing calculates the interval of 1PPS using Equation 7-10 and the clock skew is calculated based on Equation 7-11. It should be noted that these equations are similar to Equation 5-3 and Equation 5-12 from Chapter 5 since the clock skew is the time domain representation of the frequency offset. However, in this context the processor frequency used for scaling is a result of the calibration with the PIT upon system boot, thereby reflecting the inherent timekeeping accuracy of the TSC. The TSC clock skew is plotted

in Figure 7.9 along with the corresponding histogram representing one week's worth of data. Note that the phrase 'offset from 1PPS' is used here due to the fact that the skew is a relative offset and can be approximated by the difference between the clock's rate and the reference rate of 1, or in this case the 1PPS.

$$TSC_{1s} = TSC_{(ns)_{n+1}} - TSC_{(ns)_n}$$ **Equation 7-9**

$$TSC_{skew} = TSC_{1s} - 1000000000$$ **Equation 7-10**

From observing Figure 7.9, it's clear that the TSC clock exhibit a 'jump' in the beginning of synchronization going from one rate to another. This change in clock skew was intentionally introduced when the system was rebooted. When data collection was initiated, the average offset from the ideal rate of 1PPS is about 35 microseconds behind, or 35PPM of error per second. Then after the reboot, the average offset changed to about 10 microseconds, or 10PPM of error per second. Furthermore, with the exception of the random jitter noise the clock rate stayed relatively constant for the rest of the week averaging about 10 microseconds offset from the 1PPS. The change in the rate of the clock is caused by the initial calibration of the TSC upon system boot. Since the calibration results in a different value upon each system boot and it is propagated to the processor frequency parameter given by the operating system, the TSC clock skew varies depending on the accuracy of the initial calibration. As a result, a relatively large TSC clock skew would slow down the process for NTP to synchronize with the remote server clock and conversely a relatively small TSC clock skew would most likely speed up the process.

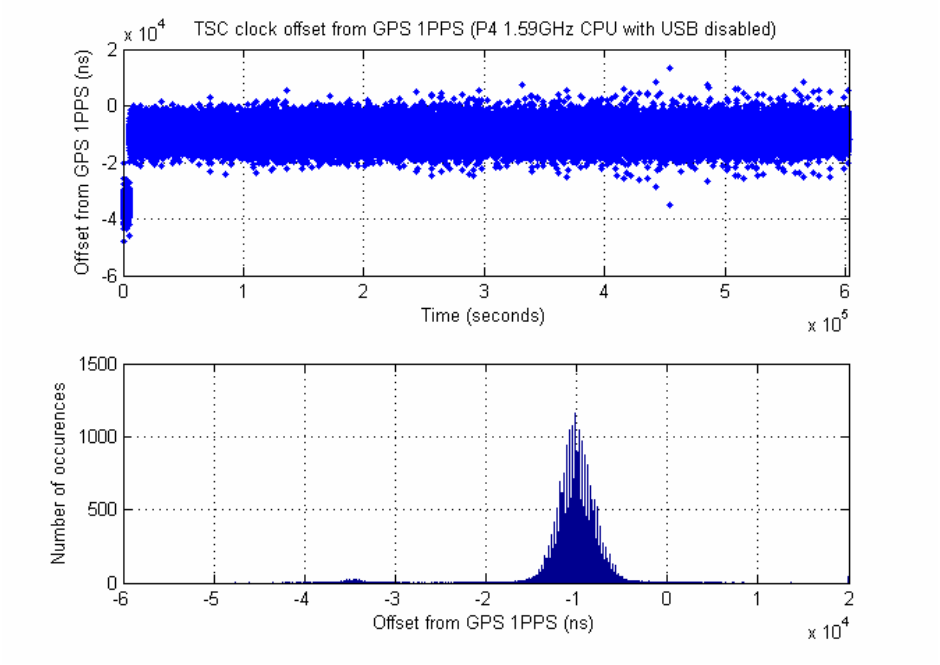

Figure 7.9 TSC clock skew (with reboot)

To further the investigation, two more measurement trials were conducted over the course of a week each to examine the variability of the PIT calibration results. The measurement result is plotted in Figure 7.10 and Figure 7.11 along with its respective histograms. Interestingly, the average rate of the clock in Figure 7.10 is about 40 microseconds behind with respect to the 1PPS, which is similar to the measurement result observed in Figure 7.9 before the reboot. In the second trial measurement, the clock is about 10 microseconds behind with respect to the 1PPS as it is shown in Figure 7.11. Hence, the calibration error is in the range of 10PPM to 40PPM.

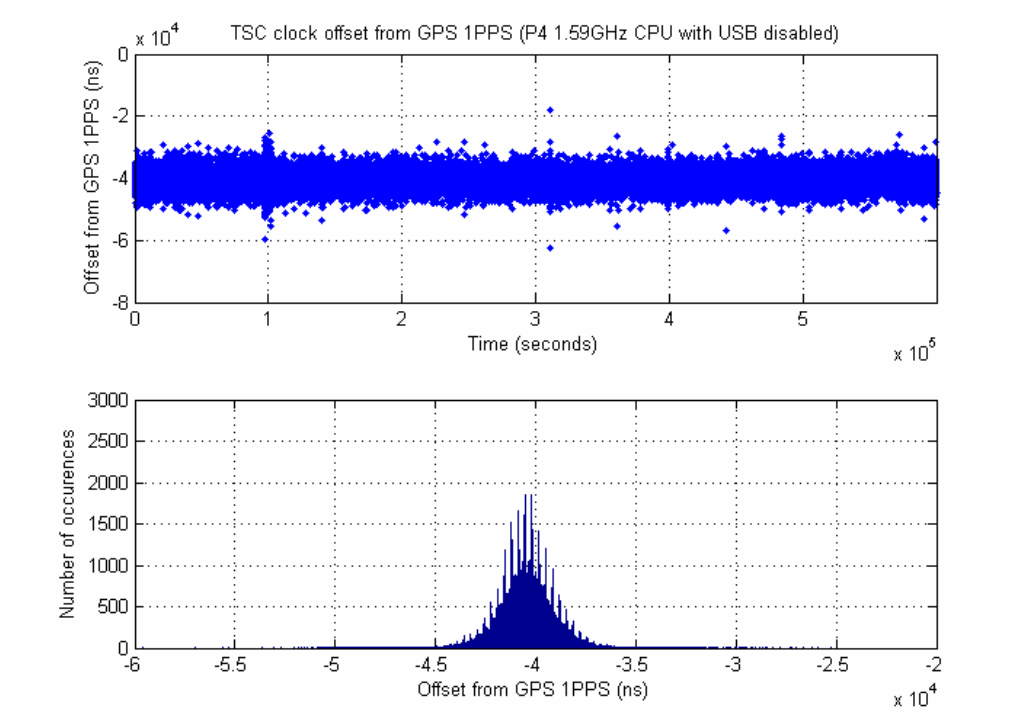

Figure 7.10 TSC clock skew measurement case 1

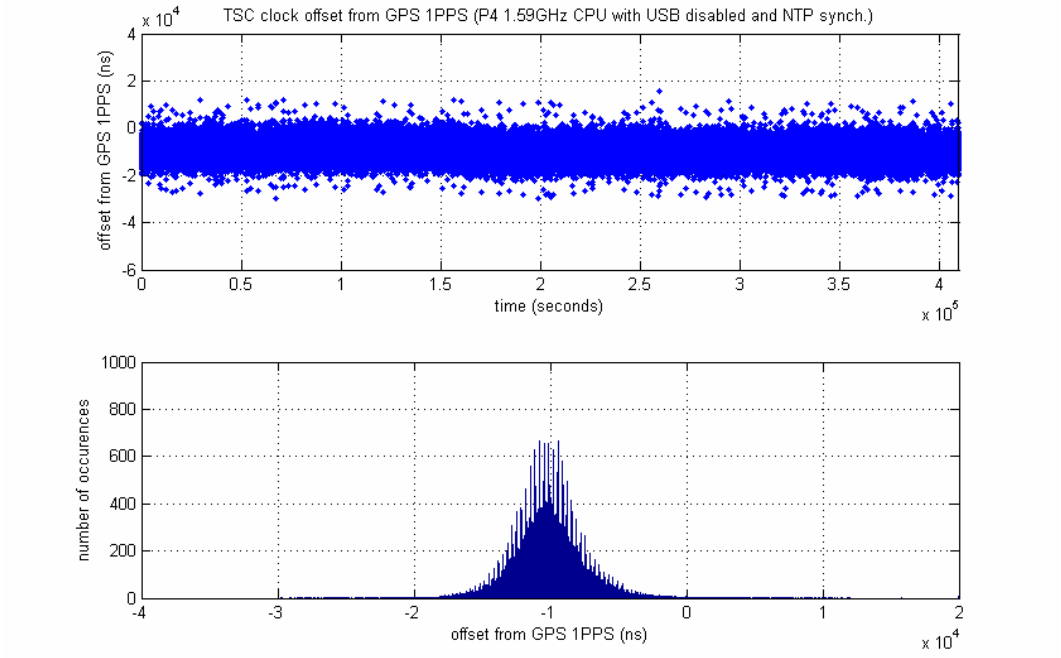

Figure 7.11 TSC clock skew measurement case 2

7.4 Summary

In this chapter, the network time synchronization is introduced as an alternative to the GPS for FDR time synchronization. Since NTP provides coverage in the wide area network (WAN) with time synchronization referenced by the UTC, it is the most appropriate for FDR time synchronization. Although it is well known that NTP is incapable of matching the accuracy obtainable by the GPS, there is very limited literature on the achievable synchronization accuracy on the WAN, let alone any data to illustrate the accuracy.

Preliminary evaluation of NTP accuracy was conducted using the Windows XP operating system due to its wide acceptance. With minimal configuration the results indicate that the operating system timer API is limited to about 16 ms of resolution. Hence it was decided to use the time offset data provided by the NTP loopfilter, which is a more accurate depiction of the local clock time offset with respect to the server clock. In addition, the time offset data can be used to compare the accuracies achieved by different operating systems. It is shown that the Linux based operating system can achieve less than 1 millisecond time offset with respect to the server clock when given the appropriate configurations and hardware. Furthermore, it is also shown that when the TSC is selected as the clocksource, the local clock offset undergoes transient upon reboot. When given enough time for NTP to correctly estimate the TSC frequency, the local clock time offset reaches well below 1 millisecond.

In relationship to FNET applications, NTP time synchronization does not provide the accuracy that is needed for phasor angle measurement but it can be used for frequency measurements where the timing requirement is not as strict. The timing accuracy of NTP is limited by the network traffic as well as the underlying hardware. Chapter 6 has shown the limitations of a real-time operating system characterized by microsecond latencies and jitter noise. Hence it is trivial to imply that a general purpose operating system such as Linux has larger variations in its timing and lower determinism. If the next generation FDR were to be based on a standalone PC, the system time of the PC can be synchronized with one or several remote servers. Given the results from this study, it

would be the most appropriate to use a Linux based operating system installed on a uniprocessor desktop PC. In addition, if the TSC is selected as the clocksource, the power management and frequency stepping features should be disabled to optimize the synchronization accuracy. Once NTP has started, there needs to be enough time allocated for the local clock time to synchronize with the server clock. Once the local clock time offset has reached below a certain threshold, software polling can be used to detect the start of the second and begin the sampling process. Although the frequency estimation is not as accurate compared with a GPS synchronized FDR, larger geographical coverage is possible with a standalone PC based FDR. Furthermore, the measured frequency data can be used for generator trip detection, generation loss estimation and event localization.

Chapter 8 Conclusions and Future Work

8.1 Conclusions and Contribution

This dissertation presents several novel architectures for the next generation FDR design. Although there are tradeoffs that need to be considered for each one, the overall goal of higher computation capabilities and lower cost can be achieved through the integration of a PC in the overall design. By leveraging the high performance floating point processing capabilities of the x86 architecture and its large installation base, a PC based FDR would be able to achieve a significantly higher sampling rate and be deployed at a faster pace. All of this is made possible by a standalone PC, a FPGA, a GPS receiver and a microcontroller. At the same time, it is also important to recognize the availability of network time synchronization for general purpose operating systems, which eliminates the dependency of GPS hardware for time synchronization. Nevertheless, given the fact that network time synchronization has lower accuracy compared with the GPS and the phasor angle estimation requires accuracy in the lower microseconds range, only frequency measurement and limited applications can be considered for the standalone PC based design.

Since frequency and phasor angle measurement accuracy is directly related to the timing of the sampling pulses, this dissertation addresses the subject matter of timing from several different perspectives. Firstly, the fact that conventional GPS is limited to operate with line of sight to the satellites is prohibiting its uses in certain locations. This work has introduced the implementation of the high sensitivity or indoor GPS for synchronized frequency and phasor angle measurements at locations that are not possible with the conventional GPS. Due to effects such as reflection and diffraction, the signal strength of the acquired satellites is weaker and the accuracy of timing and position solution degrades. Nevertheless, results are presented in this dissertation with regard to the accuracy and availability achievable by the indoor GPS and most importantly, its applicability for FDR. It is shown that the indoor GPS can be used in FDR time synchronization and it is capable of providing similar level of timing accuracy with

respect to the conventional GPS under signal degradation. Given its higher availability and comparable performances with respect to the conventional GPS, the indoor GPS adds a new dimension to the next generation FDR and expands the opportunity for more FNET coverage without degrading the measurement accuracy.

Another aspect of the timing accuracy has to do with the inherent accuracy and stability of the common crystal oscillator. Study was conducted in the characterization of the crystal oscillator to provide some insights for why the conventional oscillator can not be a precision timing source. In addition, a timing measurement system is proposed based on the TSC counter of the Intel x86 processor and a real-time extension to Linux. The measurement system provides high resolution timing through the multi-gigahertz clock of the Pentium processor and hard real-time determinism provided by RTAI. Measurement results are presented in this dissertation which shows that the PC oscillator has relatively high stability when averaged over time but inaccurate frequencies. Nevertheless, with the aid of a high precision timing source such as the GPS, the frequency inaccuracies in the crystal oscillator can be measured and removed. Statistical analysis is conducted in this work to provide some insights into the most optimized method of synchronizing the local oscillator with GPS. The results presented here is significant in the sense that a more accurate estimate of the oscillator frequency can be obtained by averaging the measurement over time whereas the conventional method of synchronization is based on the rising edge of 1PPS alone without any compensation in oscillator frequency errors. In relationship to the next generation FDR design, the averaging of the measurement can be easily performed on the PC based architecture given its fast computation capabilities and large memory size.

To investigate the accuracy of frequency and phasor angle measurement using different precision timing mechanism, a novel methodology is proposed to model the FDR based on the sampling time of the ADC. The approach was motivated by the fact that the timing resolution of PC processor clock is significantly higher than the DSP and any latency associated with the system is minimized by using a real-time operating system RTAI. Starting with the establishment of a high accuracy clock division algorithm

that is self-regulating and is independent of the clock speed, direct comparisons can be made for different systems with regard to the timing of trigger for conversion. Given the length of the 1PPS as measured by the TSC counter, the clock division algorithm can be used to calculate the exact time that the conversion should occur. The errors associated with the DSP clock is simply the difference between when the conversion should occur and when it actually occurred, as measured by the PC. Then with the histograms generated based on this error, one can use random sampling to simulate the sampling process. However, due to some optimization features of the Pentium processor, the inherent jitter noise associated with the measurement system is significant with respect to the resolution of the DSP clock. Ultimately, the FDR can be modeled more accurately given a more deterministic system with delays and jitter noise much smaller than the resolution of the DSP clock.

The concept of network time synchronization is proposed for the standalone PC based FDR. Since NTP provides wide-area coverage and is synchronized to UTC time, it is introduced in this dissertation as an alternative to GPS timing synchronization. Although at this stage of development, the accuracy of NTP time synchronization is limited by the Internet traffic load and the inherent stability of the hardware clock. Hence its accuracy cannot match that of the GPS. Nevertheless, results presented in this dissertation indicate that given the proper operating system, configuration and hardware, NTP can maintain below 1 millisecond offset with respect to UTC time. Such accuracy is not suitable for phasor angle measurement but it is acceptable for the monitoring of frequency dynamics where the timing requirement is not as strict. In addition, it was shown in the measurement results that immediately after a machine reboot, NTP is susceptible to transients in the time offset with respect to the server clock. This behavior is closely linked to the inaccuracies of the TSC calibration procedure which tends to produce varied results for TSC frequency upon every system boot. Ultimately, network time synchronization can still be a viable choice for the next generation FDR. The standalone PC based architecture can bring about the largest geographical coverage at the lowest cost.

Regardless of the architecture that will be used in the next generation design, whether it is embedded system, PC based or even a combination of both, the timing analysis results presented in this dissertation can be applied to improve upon the current FDR design and bring about higher accuracy and availability for synchronized sampling. Furthermore, the hardware architecture is also a determinant factor for FDR measurement accuracy. The standalone PC based FDR is not only limited by the accuracy of time synchronization but it is also susceptible to variability in the sampling time caused by the underlying hardware and the operating system. However, much improved real-time determinism can be achieved by integrating an FPGA to the PC based FDR timing subsystem. As a result, the PC based FDR architecture combined with FPGA does not only provide for higher computation capabilities but also deterministic timing. Nevertheless, the biggest limitation of the PC based FDR architecture is related to the security issues. Since an operating system is required for all of the general purpose PC, it is more vulnerable to intruder attacks from the wide area network. This issue did not exist in the first and second generation FDR since the embedded system does not need an operating system. Although there are some ways to lower the risk of network intrusions, these include the preferential use of Unix based operating system over the Windows based operating system or installing sophisticated firewalls and anti-virus software. More future work is needed to further the investigation on this topic and bring about more secure FDR architecture.

8.2 Future Works

- Expand design architectures for FDR to use more off the shelf components and increase the modularity of the design.
- Increase accuracy of NTP by using better quality hardware clock (TCXO, OCXO) and using dedicated FNET NTP servers.
- Develop fault detection algorithm in FDR to mitigate the effect of erroneous timing solution caused by signal attenuation in indoor GPS.
- Measure the FDR trigger for conversion timing using a more deterministic timing solution such as the time interval counter to develop a more accurate FDR model.

The model can be used to evaluate the accuracy of frequency and phasor angle measurement using network time protocol or any other time synchronization sources.

References

[1] Z. Zhong, C.C. Xu, B.J. Billian, L. Zhang, S.-J.S. Tsai, R.W. Conners, V.A. Centeno, A.G. Phadke,; Y. Liu, "Power system frequency monitoring network (FNET) implementation", IEEE Transactions on Power Systems, vol. 20, no. 4, pp. 1914 – 1921, Nov. 2005.

[2] IEEE Standard for Synchrophasors for Power System, C37.118-2005, prepared by the IEEE Power System Relaying Committee of the Power Engineering Society.

[3] J. Chen, "Accurate frequency estimation with phasor angles," M.Sc. dissertation, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 1994.

[4] C. Xu, "High Accuracy Real-time GPS Synchronized Frequency Measurement Device for Wide-area Power Grid Monitoring" Ph.D. dissertation, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 2006.

[5] B. Billian, "Next generation design of a frequency data recorder using field programmable gate arrays", Master thesis, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 2006.

[6] A.G. Phadke, J.S. Thorp, et al, "A new measurement technique for tracking voltage phasors, local system frequency, and rate of change of frequency", IEEE Transaction on PAS., 1983.

[7] C. Nguyen and K. Srinivasan, "A new technique for rapid tracking of frequency deviation based on level crossings", IEEE Transaction on Power Apparatus and Systems, 1984.

[8] M. Begovic, P. Djuric, et al, "Frequency tracking in power networks in the presence of harmonic", IEEE Transaction on Power Delivery, 1993.

[9] J.Z. Yang, C.W. Liu, "A precise calculation of power system frequency and phasor", IEEE Transaction on Power Delivery, 2000.

[10] M. Giray, M. Sachdev, "Off-nominal frequency measurement in electric power systems", IEEE Transaction on Power Delivery, 1989.

[11] V. Terzija, M. Djuric, et al, "Voltage phasor and local system frequency estimation using Newton type algorithm", IEEE Transaction on Power Delivery, 1994.

[12] M. Sachdev, H.C. Wood, et al, "Kalman filtering applied to power system measurements for relaying", IEEE Transaction on Power Apparatus and Systems, 1985.

[13] Arbiter Systems, Inc., "Model 1133A Phasor measurement specifications", 2007.

[14] T. Xia, "Frequency Monitoring Network Algorithm Improvements and Application Development", Ph.D. dissertation, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 2009.

[15] L. Wang, "Printed circuit board design for frequency disturbance recorder", Master thesis, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 2005.

[16] L. Wang, J. Burgett, J. Zuo, C.C. Xu, B. Billian, R.W. Conners, Y. Liu, "Frequency disturbance recorder design and developments", IEEE Power Engineering Society General Meeting, 2007.

[17] L. Wang, J. Fernandez, J. Burgett, R.W. Conners, Y. Liu, "Network time protocol for clock synchronization in wide area measurements", IEEE Power Engineering Society General Meeting, 2008.
[18] G. Frantz, R. Simar, "Comparing fixed and floating point DSPs", White Paper, Texas Instruments, 2004.
[19] J. Carroll, K. Montgomery, "Global positioning system timing criticality assessment – preliminary performance results", 40th Annual Precise Time and Time Interval (PTTI) Meeting, 2008.
[20] V. Madani, D. Novosel, and et.al. PSTT, "Guidelines for Synchronization Techniques Accuracy and Availability", NASPI report, 2008.
[21] H. Hellwig, "Frequency Standards and Clocks: A Tutorial Introduction"
[22] F. Diggelen, "Indoor GPS Theory and implementation", IEEE Position, Location and Navigation Symposium, 2002.
[23] D. Allan, J. Barnes, "Optimal Time and Frequency Transfer using GPS Signals", 36th Annual Frequency Control symposium, 1982.
[24] T. Dao, G. Lachapelle, "GPS Performance for Various Applications", Proceedings of ACRS, 2005.
[25] J. Carroll, K. Montgomery, Global Positioning System Timing Criticality Assessment – Preliminary Performance Results", Precise Time and Time Interval (PTTI) Meeting.
[26] M. Braasch, GPS Receiver Architecture and Measurements" Proceedings of the IEEE, Vol. 87, No. 1, Jan. 1999.
[27] W. Lewandowsk, C. Thomas, "GPS time transfer", Proceedings of the IEEE, 1991.
[28] P. Misra, P. Enge, "Global Positioning System Signals, Measurements, and Performance", Ganga-Jamuna Press, Lincoln, Massachusetts, 2001.
[29] L. Vittorini, B. Robinson, "Optimizing Indoor GPS Performance" GPS World, Nov. 2003
[30] Wieser, "High-sensitivity GNSS: The Trade-off Between Availability and Accuracy", Proceedings of third Symposium Geodesy for Geotechnical and Structural Engineering.
[31] G. Lachapelle, H. Kuusniemi, D. Dao, G. MacGougan, M. Cannon, "HSGPS Signal Analysis and Performance under Various Indoor Conditions", Proceedings of GPS/GNSS 2003 Conference.
[32] G. Dedes, A. Dempster, "Indoor GPS Positioning Challenges and Opportunities",
[33] F. Diggelen, "Global locate indoor GPS chipset and services", Institute of Navigation (ION) Technical Meeting, 2001.
[34] T. Parker and D. Matsakis, "Time and Frequency Dissemination, Advances in GPS Transfer Techniques", GPS World, Nov. 2004.
[35] J. Rutman, "Characterization of Frequency Stability in Precision Frequency Sources", Proceedings of the IEEE, June, 1991.
[36] J. Barnes et al., "Characterization of Frequency Stability", IEEE Transactions on Instrumentation and Measurement, May, 1971.
[37] F. Diggelen, "Indoor GPS Theory and Implementation", IEEE Position, Location and Navigation Symposium, 2002.

[38] S. Bednarz, "Adaptive Modeling of GPS Receiver Clock for Integrity Monitoring during Precision Apporaches", MS. Thesis, M.I.T., 2004.
[39] J. Carroll, K. Montgomery, "GPS Timing Criticality Assessment – Preliminary Performance Results", 40th Annual Precise Time and Time Interval Meeting.
[40] J. Collin et al., "HSGPS under Heavy Signal Masking – Accuracy and Availability Analysis", Proceedings of 6th Nordic Radio Navigation Conference, 2003.
[41] H. El-Natour et al., "Impact of Multipath and Cross-Correlation on GPS acquisition in Indoor environments", Institute of Navigation (ION) Technical Meeting, 2005.
[42] R. Watson et al., "Investigating GPS Signals Indoors with Extreme High-Sensitivity Detection Techniques", Journal of the Institute of Navigation, 2005.
[43] G. Geier et al., "Prediction of the Time Accuracy and Integrity of GPS Timing", IEEE International Frequency Control Symposium, 1995.
[44] J. Benedicto, D. Ludwig, "Galileo defined: proposed architecture and services for the new European satellite positioning system", GPS World, Eugene, Oregon, 2001.
[45] Coordination Scientific Information Center, "GLONASS – Global navigation satellite system", http://www.glonass-ianc.rsa.ru/, 2009.
[46] K.J.H. Khan, "Wide area power system monitoring device design and data analysis", Master thesis, Bradley Dept. Elect. Comput. Eng., Virginia Polytechnic Inst. State Univ., Blacksburg, VA, 2006.
[47] M. Lombardi, L. Nelson, A. Novick, V. Zhang, "Time and Frequency Measurement Using the GPS", 2001.
[48] i-Lotus Corporation, "M12M GPS Receiver User's Guide", 2006.
[49] NavSync Corporation, "CW12 GPS User Manual", 2007.
[50] Motorola Corporation, "M12+ GPS Receiver User's Guide", 2004.
[51] SigNav Pty. Limited, "TM3 Timing Module User Guide", 2006.
[52] QinetiQ, "The QinetiQ Q20 high sensitivity GPS receiver integration board", datasheet, 2007.
[53] u-Blox AG, "LEA-4T ANTARIS 4 programmable GPS module with precision timing", datasheet, 2006.
[54] GPS Joint Program Office, IS-GPS-200, Interface Specification: Navstar GPS Space segment/Navigation User Interfaces, Rev. D, 2006.
[55] D. Helling, M. Hense, H. Auweraer, J. Leuridan, "Data stream synchronization of distributed measurements systems using GPS technology", IEEE Workshop on intelligent data acquisition and advanced computing systems: technology and applications, 2005.
[56] Kintner, Paul M., Global Positioning System Theory and Design, Class Notes, 1999.
[57] M. Lombardi, "Remote Frequency Calibrations: The NIST Frequency Measurement and Analysis Service", NIST Special Publication 250-29.
[58] J. Vig and F. Walls, "Fundamental Limits on the Frequency Instabilities of Quartz Crystal Oscillators", IEEE International Frequency Control Symposium, 1994.
[59] H. Zhou, C Nichools, T. Kunz, H. Schwartz, "Frequency Accuracy and Stability Dependencies of Crystal Oscillators", Technical Report SCE-08-12, Nov. 2008.

[60] F. Walls and J. Vig, "Fundamental Limits on the Frequency Stabilities of Crystal Oscillators", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, July 1995.
[61] J. Vig, "Introduction to Quartz Frequency Standards", Technical Report SLCET-TR-92-1, Army Research Laboratory, 1992.
[62] R. Fillter, "Long-Term Aging of Oscillators", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, July 1993.
[63] D. Allan, "Time and frequency (time domain) characterization, estimation, and prediction of precision clocks and oscillators", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, Nov. 1987.
[64] D.B. Percival, "Use of robust statistical techniques in time scale formation", Preliminary Report, U.S. Naval Observatory Contract No. N70092-82-M-0579, 1982.
[65] P.H. Kamp, "Timecounters: efficient and precise timekeeping in SMP kernels", Proceedings of the BSDCon Europe 2002, Nov. 2002.
[66] Brannon, "Sampled Systems and the Effects of Clock Phase Noise and Jitter", Analog Devices Appliation Note AN-756, 2004.
[67] V. Rosati, R. Fillter, S. Schodowski, J. Vig, "State of the Art in Crystal Oscillators, Present and Future", US Army Electronics Technology and Devices Laboratory, 1983.
[68] R. Fillter, "The Acceleration Sensitivity of Quartz Crystal Oscillators: A Review", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, May 1988.
[69] J. Vig and T. Meeker, "The Aging of Bulk Acoustic Wave Resonators, Filters and Oscillators", 45th Annual Symposium on Frequency Control, 1991.
[70] Sullivan and J. Levine, "Time Generation and Distribution", Proceedings of the IEEE, July, 1991.
[71] R. Wilson, "Uses of Precise Time and Frequency in Power Systems", Proceedings of the IEEE, July, 1991.
[72] J. Roberson, "Accurate, high resolution absolute timing on the PC platform", Dedicated Systems Magazine, 2001, Q3.
[73] J. Zhang, R. Lumia, J. Wood, G. Starr, "Achieving Deterministic Hard Real-time Control On An IBM-Compatible PC: A General Configuration Guideline", IEEE International Conference on Systems, Man and Cybernetics, 2005.
[74] L. Dozio, P. Mantegazza, "Linux Real Time Application Interface (RTAI) in low cost high performance motion control", Proceedings ANIPLA., 2003.
[75] National Institute of Standards and Technology, "Introduction to Linux for real-time control", available at http://www.isd.mel.nist.gov/projects/rtlinux/, 2002.
[76] T. Wiedemann, "How fast can computer react?", Seminar paper, 2005.
[77] F. Proctor, W. Shackleford, "Timing studies of real-time Linux for control", Proceedings of the 2001 ASME computers in engineering conference, 2001.
[78] F. Proctor, W. Shackleford, "Real-time operating system timing jitter and its impact on motor control", Proceedings of the SPIE International symposium on Intelligent Systems and Advanced Manufacturing, 2001.

[79] P. Cloutier, P.Mantegazza, S. Papacharalambous, I. Soanes, S. Hughes, K. Yaghmour, "DIAPM-RTAI position paper", Real Time Operating System Workshop, 2000.
[80] R. Giladi, "Evaluating the MFLOPS measure", IEEE Micro, 1996.
[81] R. Wilson, "Methods and uses of precise time in power systems", IEEE Transactions on Power Delivery, Jan. 1992.
[82] J. Ridoux, D. Veitch, "Ten microseconds over LAN, for free", IEEE Transactions on Instrumentation and Measurement, June, 2009.
[83] Veitch, J. Ridoux, S. Korada, "Robust synchronization of absolute and difference clocks over networks", IEEE/ACM Transactions on Networking, Apr. 2009.
[84] Pasztor, D. Veitch, "PC based precision timing without GPS", Proceedings ACM Sigmetrics Confernce Meas. Model. Comput. Syst., 2002.
[85] K.Correll, N. Barendt, M. Branicky, "Design considerations for software only implementations of the IEEE-1588 precision time protocol", Proceedings ISPCS, 2005.
[86] H. Marouani, M. Dagenais, "Internal clock drift estimation in computer clusters", Journal of Computer Systems, Networks and Communications, 2008.
[87] D. Mills, "Adaptive hybrid clock discipline algorithm for the network time protocol", IEEE/ACM Transaction on Networking, Vol. 45, 1999.
[88] D. Mills, "Clock discipline algorithms for the network time protocol Version 4", Electrical Engineeering Report 97-3-3, University of Delaware, March 1997.
[89] D. Mills, "The network computer as precision timekeeper", Proceedings Precision Time and Time Interval Applications and Planning Meeting, Dec. 1996.
[90] D. Mills, "Unix kernel modifications for precision time synchronization", Electrical Engineering Report 94-10-1, University of Delaware, Oct. 1994.
[91] D. Mills, "Network time protocol version 4 reference and implementation guide", NTP working group technical report 06-6-1, Jun. 2006.
[92] D. Mills, "The nanokernel", Proceedings Precision Time and Time Interval Applications and Planning Meeting, Nov. 2000.
[93] T. Broomhead, J. Ridoux, D. Veitch, "Counter availability and characteristics for feed-forward based synchronization", International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication, Oct. 2009.
[94] J. Elson, L. Girod, D. Estrin, "Fine grained network time synchronization using reference broadcasts", Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, Dec. 2002.
[95] J. Greunen, J. Rabaey, "Lightweight time synchronization for sensor networks", Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, Sept. 2003.
[96] M. Sichitiu, C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks", IEEE Wireless Communications and Networking Conference, 2003.
[97] S. Ganeriwal, R. Kumar, M. Srivastava, "Timing sync protocol for sensor networks", ACM SenSys, Nov. 2003.

[98] J. Chiang, T. Chiueh, "Accurate clock synchronization for IEEE 802.11 based multi-hop wireless networks", IEEE Transactions on Parallel and Distributed Systems, 2007.
[99] W. Kester, "Mixed-signal and DSP design techniques", Analog Devices, 2000.
[100] F. Sivrikaya, B. Yener, "Time synchronization in sensor networks: a survey", IEEE Networks, 2004.
[101] J. Eidson, M. Fischer, and J. White, "IEEE-1588 Standard for a precision clock synchronization protocol for networked measurement and control systems", Proceedings of the 34th Precise Time and Time Interval (PTTI) Systems and Applications Meeting, 2000.
[102] Hewlett-Packard, "Hewlett-Packard Application Note: 200-3 The Fundamentals of Time interval measurement", Hewlett-Packard company, 1997.
[103] J. Riley, "Handbook of frequency stability analysis", Beaufort Hamilton Technical Services, 2007. Available on web: http://tf.nist.gov/general/pdf/2220.pdf
[104] W. Kester, "Which ADC architecture is right for your application?", Analog Dialogue, vol. 39-06, Jun. 2005.
[105] W. Kester, "MT-020: ADC Architectures I: The Flash Converter", Tutorial MT-020, Analog Devices Inc., Norwood, MA. Jan. 15, 2006.
[106] W. Kester, "MT-021: ADC Architectures II: Successive Approximation ADCs", Tutorial MT-021, Analog Devices Inc., Norwood, MA. Jan. 25, 2006.
[107] W. Kester, "MT-022: ADC Architectures III: Sigma-Delta ADC Basics", Tutorial MT-022, Analog Devices Inc., Norwood, MA. Jan. 25, 2006.
[108] W. Kester, "MT-024: ADC Architectures V: Pipelined Subranging ADCs", Tutorial MT-024, Analog Devices Inc., Norwood, MA. October, 2008.
[109] W. Kester, "MT-027: ADC Architectures VIII: Integrating ADCs", Tutorial MT-027, Analog Devices Inc., Norwood, MA. October, 2008.
[110] G. I. Kiani, A. Karlsson, L. Osslon, and K. P. Esselle, "Glass characterization for designing frequency selective surfaces to improve transmission through energy saving glass windows," Asia Pacific Microwave Confernce, Bangkok, December, 2007.
[111] M. Gustafsson, A. Karlsson, "Design of frequency selective windows for improved indoor outdoor communication", IEEE Transactions on Antennas and Propagation, Volume 54, 2006.

Appendix A

GPS Signals and Positioning Analysis

I Review of GPS Signal Power Levels

Power in Decibels

Power can be expressed in decibels by dividing the power by a reference power level. Typical reference levels are either one watt or one milliwatt. A power level in decibel-milliwatts can be computed as:

$$P_{\mathrm{dB}\,m} = 10 \cdot \log\left(\frac{P_{mw}}{1_{mw}}\right)$$

Also, it can be expressed in decibel-watts:

$$P_{\mathrm{dB}\,W} = 10 \cdot \log\left(\frac{P_w}{1_w}\right) = 10 \cdot \log\left(\frac{P_{mw}}{1000_{mw}}\right) = 10 \cdot \log\left(\frac{1}{1000} + \frac{P_{mw}}{1_{mw}}\right) = -30 + 10 \cdot \log\left(\frac{P_{mw}}{1_{mw}}\right) = P_{\mathrm{dB}m} - 30$$

Signal-to-Noise Ratio (SNR)

Signal to noise ratio (SNR) is defined as the ratio between signal power and noise power, expressed in decibels. For a signal power (S) and a noise power (N) defined in common units of power, the SNR is:

$$SNR = 10 \cdot \log\left(\frac{S}{N}\right)$$

Similarly:

$$SNR = S_{\mathrm{dB}\,m} - N_{\mathrm{dB}m} = S_{\mathrm{dB}\,W} - N_{\mathrm{dB}\,W}$$

Power Spectral Density and Thermal Noise

Thermal noise has a constant power spectral density and the power of thermal noise generated is dependent on the temperature and the noise bandwidth. Hence, the noise power spectral density is the product of Boltzmann's constant, k and the absolute temperature, T. Ambient thermal noise is typically calculated to be 280 Kelvin. This is generally used as the effective noise temperature of the earth:

$$N_T = kT = 1.38 \cdot 10^{-23}\,\frac{J}{K} \cdot 290K = 4 \cdot 10^{-21} J$$

Since a watt is one joule per second, the power spectral density can alos be expressed as watts per hertz. As a result, ambient thermal noise power spectral density is:

$$N_T = 4 \cdot 10^{-21}\,\frac{W}{Hz} = -204\mathrm{dB}\,\frac{W}{Hz} = -174\mathrm{dB}\,\frac{m}{Hz}$$

Carrier to Noise Ratio

The carrier to noise ratio is defined as the carrier power divide by the noise power spectral density. To calculate the carrier to noise ratio for a GPS receiver operating at thermal noise floor, it is necessary to obtain carrier power. The C/A code GPS signal is specified to arrive at the surface of the Earth at a power level of -160dBW or above. Hence, carrier to noise ratio of this power level is calculated as:

$$\frac{C}{N_o} = \frac{-160\text{dB}\,W}{-204\,\frac{\text{dB}\,W}{Hz}} = 44\text{dB}\cdot Hz$$

II Relationship between Carrier to Noise Ratio and Power Levels

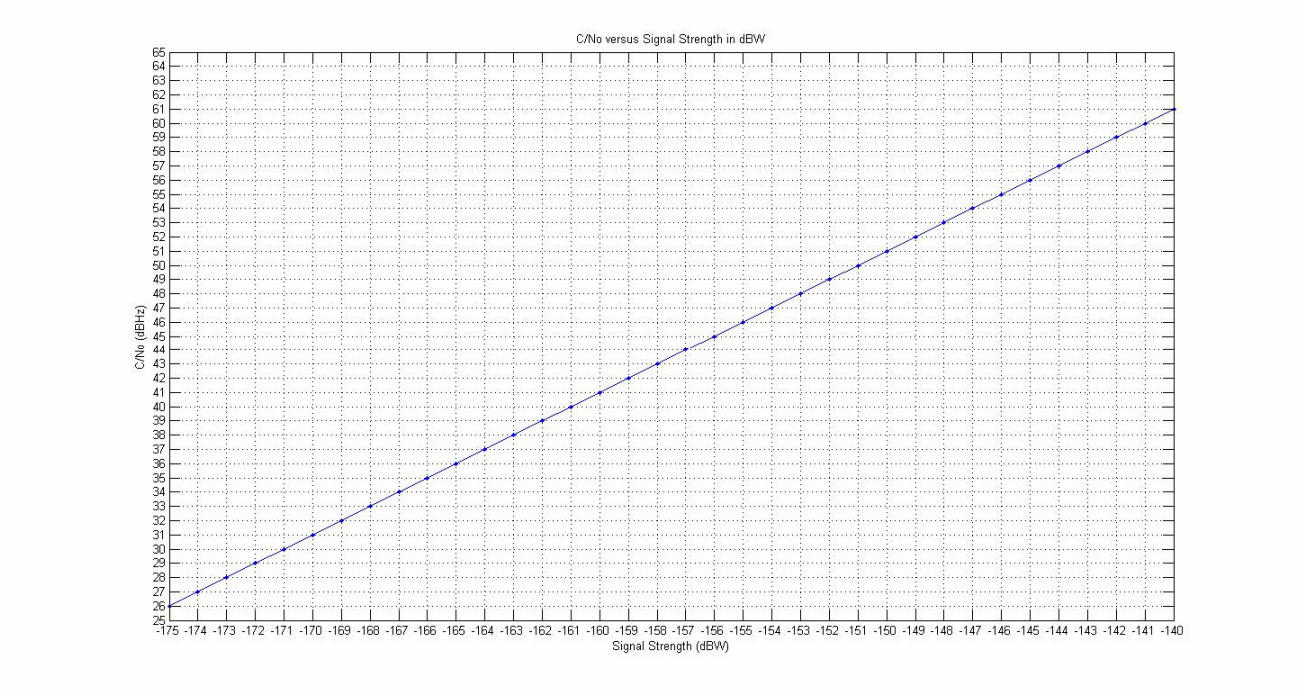

Figure A.1 Relationship between carrier to noise ratio and signal strength

III Indoor GPS positioning

The unit of measurement for latitude and longitude is in milliseconds, where 1 degree in latitude or longitude is equivalent to 3600000 milliseconds. The longitude (λ) defines west-east position with respect to the prime (Greenwich) meridian, while the latitude (φ) indicates north-south position with respect to the equator. For GPS technology, the WGS 84 (World Geodetic System 1984) and the Earth-Centered Earth-Fixed (ECEF) frame of reference. The ECEF frame is an orthogonal Cartesian frame while the WGS-84 frame is an ellipsoidal approximation to the Earth's surface for determining latitude, longitude and altitude. By convention the constants for the WGS-84 coordinate system are:

The semi-major axis, a = 6378.137 km
The semi-minor axis, b = 6356.75231425 km
Since the CW12 and the M12M GPS receivers output the position solution in the WGS-84 datum while the GPS determines the user position in ECEF, a practical means is required to transform between the two coordinate systems. Given the user's position in the WGS-84 systems, the conversion to ECEF is given below:

$$x = (N + h)\cos(\phi)\cos(\lambda)$$
$$y = (N + h)\cos(\phi)\sin(\lambda)$$
$$z = \left(\frac{b^2}{a^2} \cdot N + h\right)\sin(\phi)$$
$$N = \frac{a^2}{(a^2 \cos^2(\phi) + b^2 \sin^2(\phi))^{\frac{1}{2}}}$$

Comparison of position fix between conventional GPS and indoor GPS

To compare the position fix obtained by the conventional GPS (M12M) and the indoor GPS (CW12), both the M12M and the CW12 were placed next to each other at the same location next to the window. Position solution in latitude and longitude with respect to time are shown in Figure A.2 and Figure A.3 respectively. Figure A.4 shows the overall position solution with the average position solution. Besides some relatively small deviations in the longitude solution given by the M12M and the noise that is associated with the latitude solution given by the CW12, the difference between the averaged positions provided by the two receivers in x, y and z directions are all below 50 meters. The result indicates that the positioning error is within the bound of the most commonly quoted GPS positioning tolerance of 50 meters.

CW12
X_{ECEF} = 845898.3 m
Y_{ECEF} = -5014126.4 m
Z_{ECEF} = 3838244.2 m
M12M
X_{ECEF} = 845888.5 m
Y_{ECEF} = -5014150.6 m
Z_{ECEF} = 3838241.3 m
Difference in position
Δx_{ECEF} = -9.8627 m
Δy_{ECEF} = -24.2310 m
Δz_{ECEF} = -2.8388 m

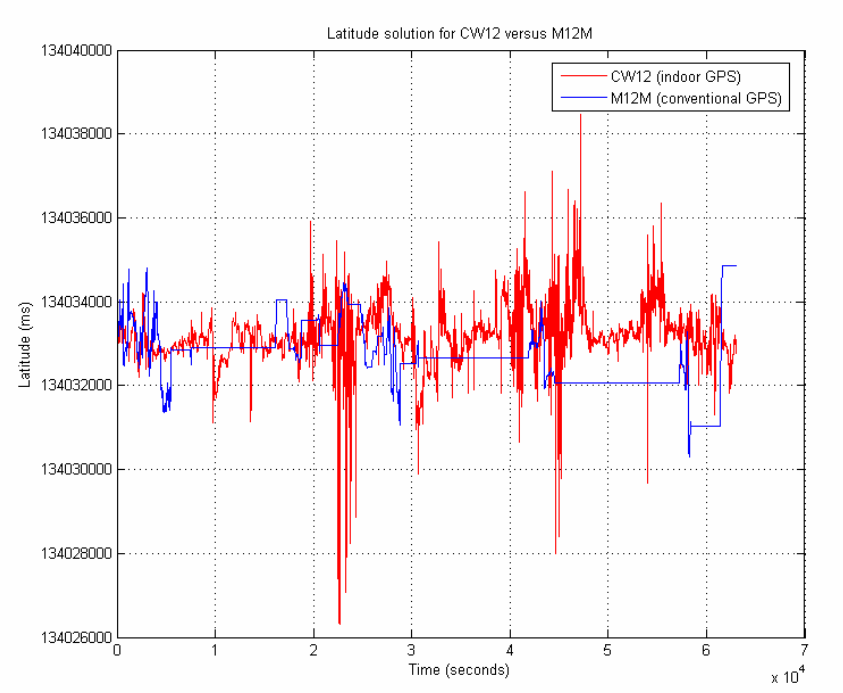

Figure A.2 Comparison of CW12 with M12M in latitude solution

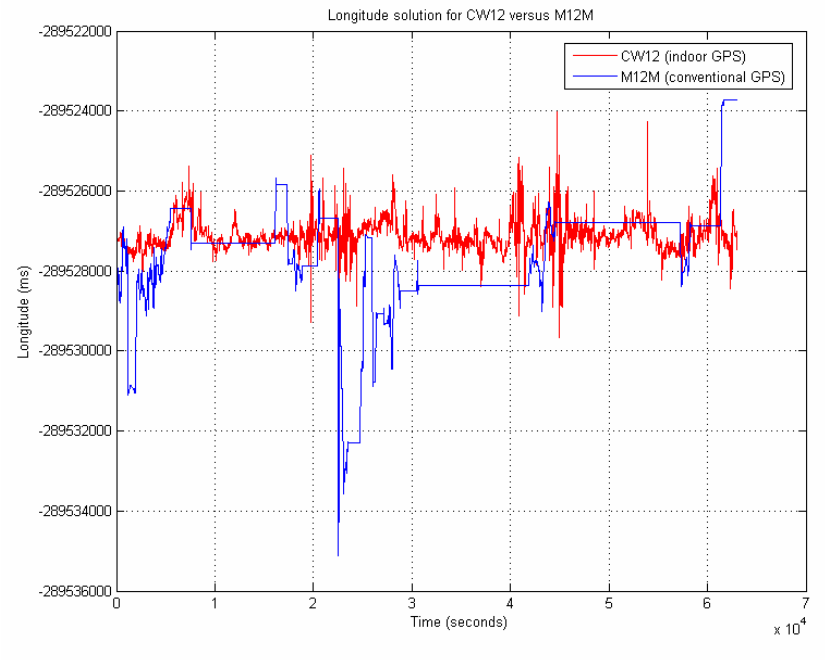

Figure A.3 Comparison of CW12 with M12M in longitude solution

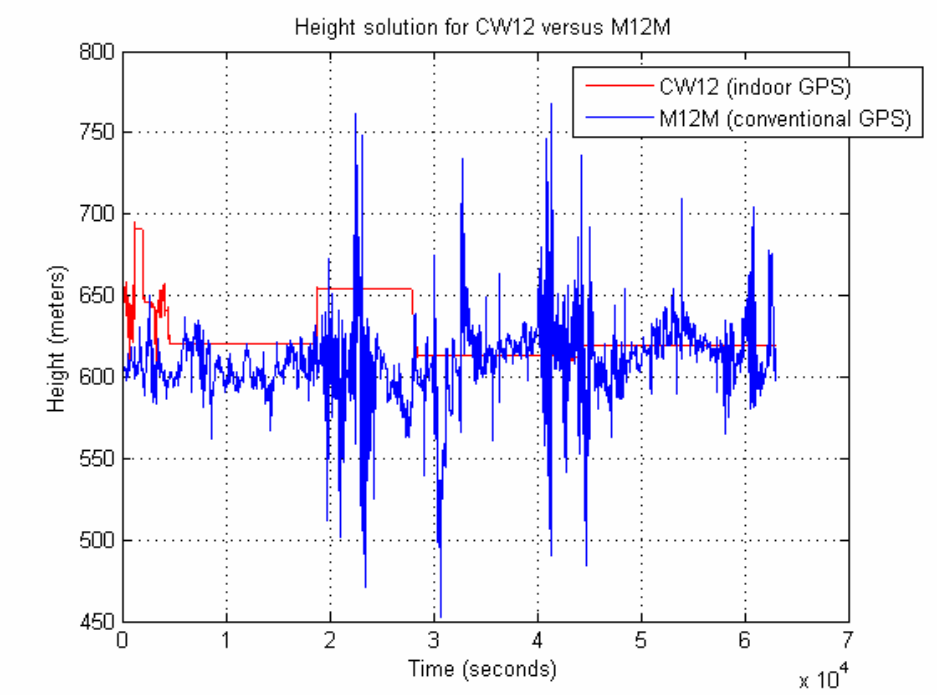

Figure A.4 Comparison of CW12 with M12M in altitude solution

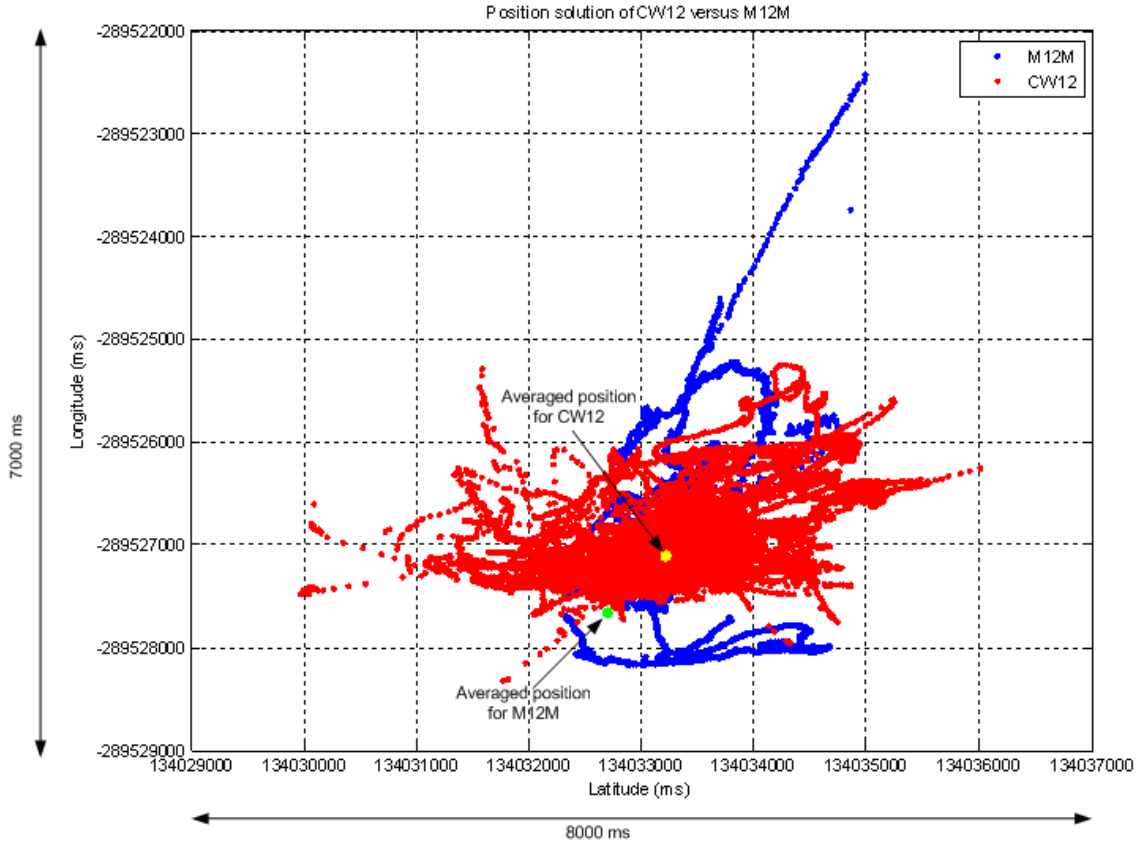

Figure A.5 Overall position solution of CW12 and M12M

Table A-1 Statistics of M12M and CW12 position solution (ms)

	CW12 latitude	M12M latitude	CW12 height	M12M height	CW12 longitude	M12M longitude
Mean	134033209	134032699	609.3	625.3	-289527096	-289527654
Minimum	134030000	134030307	452.9	600.4	-289529655	-289535127
Maximum	134038479	134034851	767.4	695.2	-289524011	-289523730
Range	12154	4544	314.5	94.81	5644	11397
Std. deviation	684	754	22.65	15.99	375	1398

Comparison of the position fix given by CW12 operating under nominal conditions and signal degradation

To examine the effect of the signal attenuation on the position fix of the indoor GPS, the CW12 antenna was placed both inside a drawer and next to a window. Position solution in latitude, longitude and altitude with respect to time are shown in Figure A.6, Figure A.7 and Figure A.8 respectively. Moreover, the actual position given by Google Map is plotted as a green line for comparison. Figure A.9 shows the overall position solution with the average position solution. In the case where the antenna is placed next to the window, the position solution is relatively stable. However, when the antenna is placed in the drawer, the position solution has large variations at first and it took more than 10,000 seconds for the position solution to stabilize. The difference between the averaged positions provided by the two receivers in the x and y directions are in the hundreds of meters whereas in the z direction there is more than 1000 meters in difference.

CW12 antenna at window
X_{ECEF} = 845906.5 m
Y_{ECEF} = -5014129.8 m
Z_{ECEF} = 3838260.4 m
CW12 antenna in drawer
X_{ECEF} = 845417.3 m
Y_{ECEF} = -5014473.8 m
Z_{ECEF} = 3836970.9 m
Difference in position
Δx_{ECEF} = 489 m
Δy_{ECEF} = 343.9 m
Δz_{ECEF} = 1289.5 m

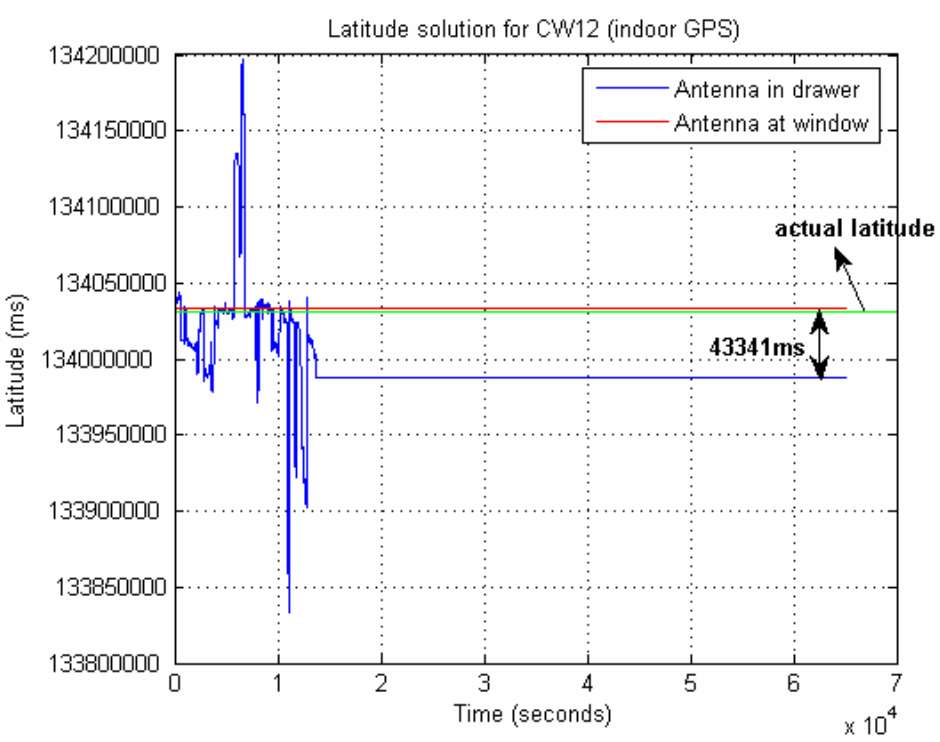

Figure A.6 CW12 latitude solution with signal degradation

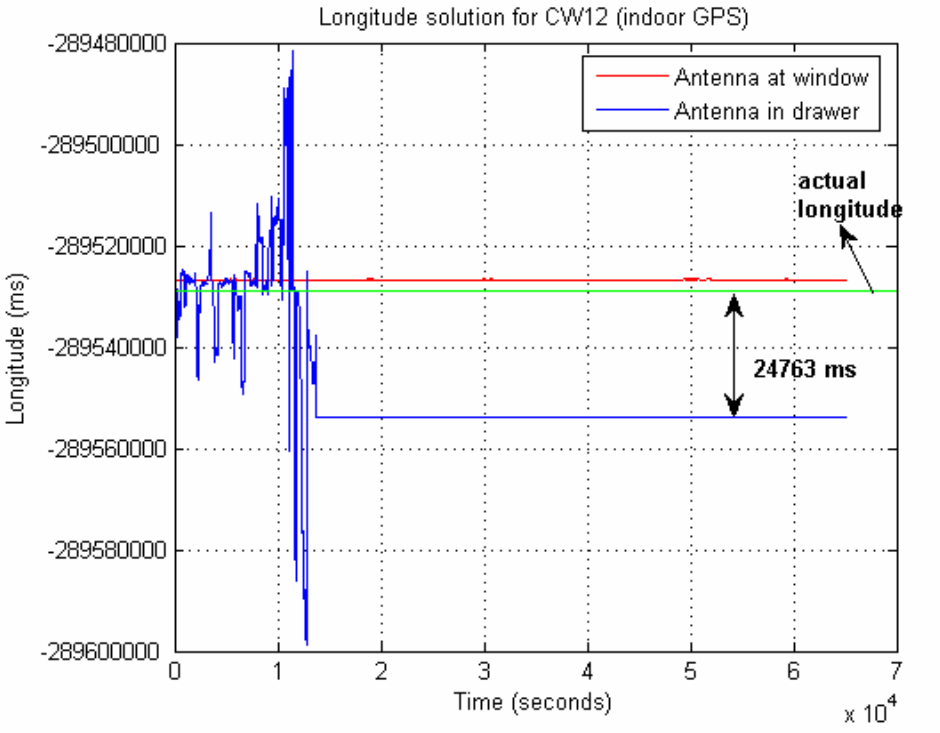

Figure A.7 CW12 longitude solution with signal degradation

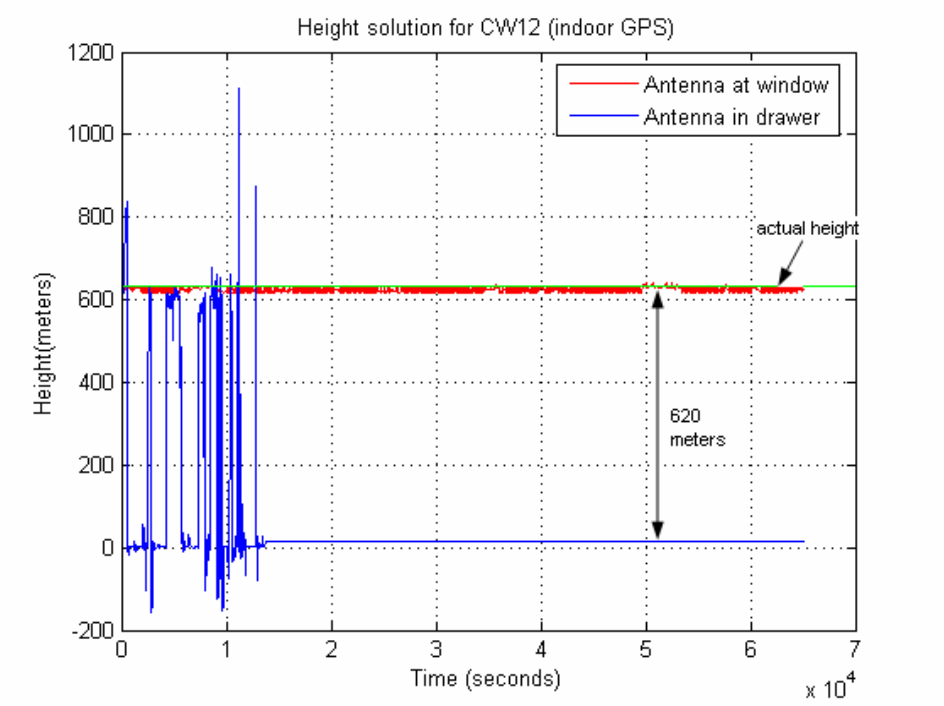

Figure A.8 CW12 altitude solution with signal degradation

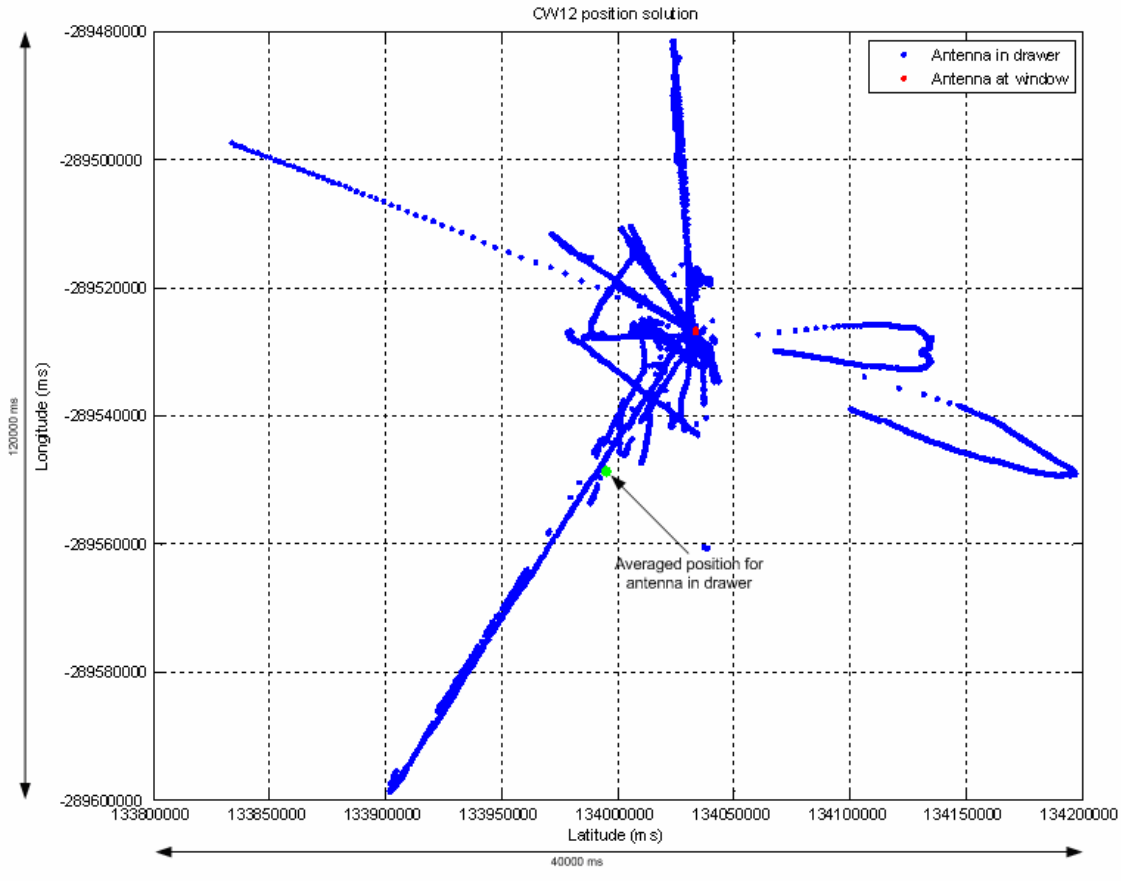

Figure A.9 Overall position solution of CW12 with signal degradation

Table A-2 Statistics of CW12 position solution when operating under signal degradation (ms)

	CW12 latitude (drawer)	CW12 latitude (window)	CW12 longitude (drawer)	CW12 longitude (window)	CW12 height (drawer)	CW12 height (window)
Mean	133995170	134033534	-289548678	-289526793	622.9	48.05
Minimum	133833830	134033430	-289598666	-289526900	616.9	-155.3
Maximum	134197142	134033688	-289481410	-289526575	640.9	1112
Range	363312	258	117256	59.3	23.99	1268
Std. deviation	25706.1	35.1	12942.1	325	4.738	144.1

Appendix B

NTP Servers for Time Synchronization

I NTP Servers

NTP servers including Virginia Tech servers
server ntp-1.vt.edu
server ntp-2.vt.edu
server ntp-3.vt.edu
server ntp-3.vt.edu
server time-a.nist.gov
server time-b.nist.gov
server time.nist.gov

NTP servers excluding Virginia Tech servers
server nist1-dc.witime.net
server tick.uh.edu
server time-a.nist.gov
server time-b.nist.gov
server ntp.myfloridacity.us
server tick.usno.navy.mil
server ntp0.broad.mit.edu
server nist.expertsmi.com
server time.keneli.org

Appendix C

Measurement System Wiring Diagram and RTAI

I Parallel Port Connection with FDR Trigger for Conversion Signal

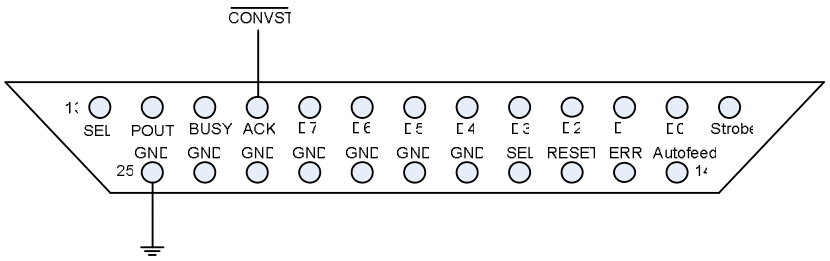

Figure C.1 PC interface with FDR trigger for conversion signal

II Brief Summary of RTAI

The design of RTAI is highly modularized and is composed of several components. In the first versions it consisted of an abstraction layer called Real-Time Hardware Abstraction Layer (RTHAL) and a small real-time kernel that runs Linux as its idle task. RTHAL is a structure installed in the Linux kernel which is used to intercept the hardware interrupts and process them. The purpose of RTHAL is to minimize the modifications needed to the kernel code. With RTHAL interrupt handlers are easily changed or modified without interfering Linux. Furthermore, the real-time kernel is not restricted to the supplied RTAI kernel and any real-time kernel can be used to interact with the HAL interface. Figure 5.2 shows a block diagram of RTAI. In terms of the implementation of RTAI, only a kernel patch is required to install the RTHAL. The layer acts as an interface between Linux and the hardware. In the newer versions of RTAI (beyond version 3.0), RTHAL is being replaced with another HAL known as Adaptive Domain Environment for Operating System (ADEOS). The reasons for replacing RTHAL with ADEOS are two folds. Most importantly ADEOS is not covered by the RTLinux patent. Also, ADEOS is much more generic and can do more tasks than just

providing hardware abstraction. ADEOS is a resource virtualization layer available as a Linux kernel patch. It allows for several domains to coexist on the same hardware. A domain could be an operating system like Linux but it could also be real-time tasks. The domains do not see each other but every domain sees ADEOS. Furthermore, each domain is attached to a central data structure called the event pipeline or I-Pipe, which offers the capability to notify the domains for external interrupts, system calls issued by Linux or other system events. Each domain has been assigned a static priority, which is used for controlling the order of events. Once an event such as an external interrupt occurs, it is first handled by the domain that has the highest priority. After processing the interrupt it is being sent down along all the other attached domains. Each domain relies on the Linux kernel to load the kernel modules in order to put it into operation.

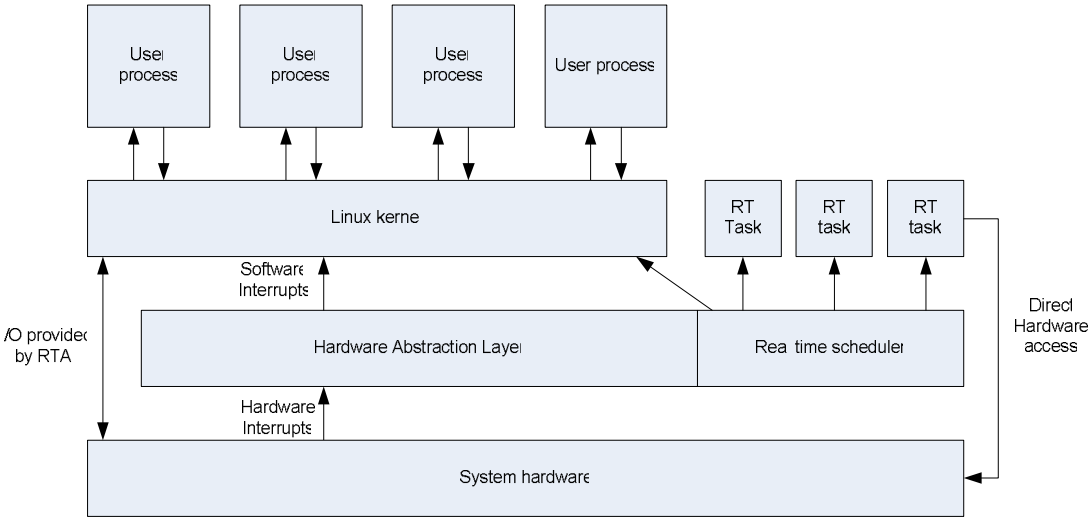

Figure C.2 RTAI Functional Block Diagram

The RTAI extension LXRT (Linux Realtime) is an API for RTAI which makes it possible to develop real-time applications entirely in user space without having to create kernel modules. The LXRT is useful because the use of kernel modules introduce risks in modifying memory locations unintentionally, which causes data corruption and malfunction of Linux kernel. In other words, kernel space memory is not protected from unintended access. The LXRT provides the developer with a safer environment for testing and debugging of application code. Once the application code is considered to be bug free the task can be converted into kernel space module as a hard real-time task.

Finally LXRT allows applications to dynamically switch between soft real-time and hard real-time by using a single function call in the user space. This is useful in a multi-tasking environment where the tasks can be prioritized according to their timing requirements.

The scheduling of RTAI tasks are prioritized with the Linux kernel running as a low priority task. When real time tasks are executed, the scheduler gives them priority over the Linux kernel. The scheduler itself is implemented as another kernel module which enables the implementation of alternative schedulers if required. There are three different types of schedulers depending on the machine type. Uniprocessor (UP) scheduler is intended to be used on uniprocessor platforms and can not be used with multiprocessor machines. Symmetric Multiprocessor (SMP) scheduler is designed for SMP (multiprocessor) machines. SMP provides an interface for the applications to select the processor on which a given task is run. Multi-uniprocessor (MUP) scheduler can be used with both multi and uniprocessor machines. However, unlike the SMP scheduler, the tasks must be bound to specific processor when MUP scheduler is used.

In order to make the application development flexible, RTAI developers have introduced several different mechanisms for inter-process communication (IPC) between real-time tasks and user space processes. Different IPC mechanisms are included as kernel modules which can be loaded in addition to the standard RTAI modules if several processes need to communicate with each other. The IPC and some of the other features of RTAI are outlined below.

FIFO (First In First Out)

The most basic communication method of RTAI are FIFOs. FIFO is an asynchronous and unblocking one-way communication between a Linux process and a real-time task having a size limit indicated by the user. The developer has the responsibility of managing the FIFO when it becomes full, in which case new data can not be written until the old data is consumed.

Semaphores

Semaphores are used for communication and synchronization among real time tasks. Semaphores are counters allocated and released by the tasks and processes. RTAI provides an API for using semaphores and each semaphore is technically associated to a FIFO. As a result each semaphore uses one entry from the global FIFO.

Shared Memory

Depending on the application, shared memory provides an alternative to FIFOs. Shared memory is a common block of memory which can be read or written by any processes and tasks in the system. Since different processes can operate on the shared memory asynchronously, it is important to ensure that data on the shared memory is not unintentionally overwritten. In this case semaphores can be used to guarantee the mutual exclusion of a memory block.

Mailbox

One of the most used IPC method is the mailbox. Any number of processes can send and receive messages to and from a mailbox. Similar to the FIFO, mailbox stores messages up to its size limit. There can be multiple mailboxes active simultaneously. The mailbox sending and receiving operations can be associated with a timer, which allows a time-out alarm to be sent to the user whenever a sending or receiving operation do not complete within the given time.

Memory management and Posix threads

RTAI features memory management and Posix threads in the real-time environment. These two features are only used in a few applications. The present version of RTAI include a memory management module which allows dynamic allocation of memory in the real-time tasks. This allows the developer to allocate memory sizes other than the default, which is preallocated by RTAI before real-time execution. The Posix thread can be implemented through RTAI according to the POSIX (Portable Operating System Interface for Unix) 1003.1c standard.

III C Code for Measurement PC

```
/*
serial_hard.c

Kernel module (RTAI hard real time) for capturing the serial port interrupt when modem
status lines like Carrier Detect (CD), Data Set Ready(DSR) and Clear To Send(CTS)
change their state. Since the GPS receiver provides the 1PPS signal on its CD line, this
program can be used to measure the length of the 1PPS using the TSC counter provided
by rdtsc() instruction.

Note: Since the serial interrupt is triggered by both rising and falling edge, two interrupts
will be generated each second.

*/

#include <linux/module.h>
#include <linux/interrupt.h>
#include <rtai.h>
#include <rtai_sched.h>

#define SERPORT 0x3F8                                   //location of serial port control
register

static RTIME counter1, counter2;                        //type long long
static void serial_handler(void)                        //define ISR
{
      //counter2 = rt_get_time;
      //rt_printk("interrupt generated, count number is %lld \n", counter2);
      counter2 = rtai_rdtsc();                          //read TSC upon interrupt
      rt_printk("rtai_rdtsc() returns %lld \n", counter2);      //write counts to kernel log
      tmp = inb_p( SERPORT + 5);
      tmp = inb_p( SERPORT + 6);
      rt_ack_irq(4);                                    //acknowledge interrupt
}

int xinit_module(void)
{
      int ret;

      outb_p(0, SERPORT + 3); // reset DLAB
      outb_p(0, SERPORT + 1);

      ret = rt_request_global_irq(4, (void *)serial_handler);       //interrupt init. routine
      if (ret)    { printk ("##### error requesting irq 4: returned %d\n", ret); }
      rt_enable_irq(4);

      outb_p(0, SERPORT + 3);
```

```
	outb_p(0xC7, SERPORT + 2);
	outb_p(0x0B, SERPORT + 4);

	// Bit 3 Enable Modem Status Interrupt
	//Bit 2 Enable Receiver Line Status Interrupt
	outb_p(0x0C, SERPORT + 1);

	rt_set_oneshot_mode();                                          //oneshot mode

	(void) start_rt_timer(1);
	//counter1 = rt_get_time();
	//rt_printk("one shot mode set, count number is %lld \n", counter1);
	counter1 = rtai_rdtsc();
	rt_printk("rtai_rdtsc() returns %lld \n", counter1);

	rt_printk("Interrupt generated. You should see the latency messages\n");
	return 0;
}

void xcleanup_module(void)
{
	rt_printk("Unloading serial port test\n");
	rt_disable_irq(4);
	outb_p(0, SERPORT + 3);  // disable DLAB
	outb_p(0, SERPORT + 1); // disable serial ints
	rt_free_global_irq(4);

}

module_init(xinit_module);
module_exit(xcleanup_module);
MODULE_LICENSE("GPL");


/*
parallel_hard.c
```

Two interrupt service routines (ISR) are implemented in this module. One ISR is for capturing the serial port interrupt. Since the GPS receiver provides the 1PPS signal on its CD line, the length of the 1PPS is measured using the TSC provided by rdtsc() instruction. Another ISR is used for capturing the parallel port interrupt. The parallel port interrupt is triggered by the FDR trigger for conversion signal and is measured by the TSC provided by rdtsc() instruction.

Note: Since the serial interrupt is triggered by both rising and falling edge, two interrupts will be generated each second.

```
*/

#include <linux/module.h>
#include <rtai.h>
#include <rtai_sched.h>

#define PARPORT 0x378
#define SERPORT 0x3F8

static int time;
static int time2;
static int timex;
static int timex2;
RTIME counter1, counter2;

static void handler(void)                                   //parallel handler
{


      counter1 = rtai_rdtsc();
      rt_printk ("%lld,", counter1);
      rt_printk ("%d \n", time);
      //rt_printk ("%lld \n", counter1);
      time++;

      rt_ack_irq(7);
}

static void handler2(void)                                  //serial handler
{
      int tmp;
      int timediff;

      time = 0;
      counter2 = rtai_rdtsc();
      rt_printk("*%lld \n", counter2);
      //counter2 = rt_get_cpu_time_ns();
      //rt_printk("*%lld ns\n", counter2);



      tmp = inb_p( SERPORT + 5);
      tmp = inb_p( SERPORT + 6);
      rt_ack_irq(4);
}
```

```
int xinit_module(void)
{
	int ret;
	//initialize serial port stuff
	outb_p(0, SERPORT + 3); // reset DLAB
	outb_p(0, SERPORT + 1);
	ret = rt_request_global_irq(4, (void *)handler2);
	if (ret)	{ printk ("##### error requesting irq 4: returned %d\n", ret); }
	rt_enable_irq(4);
	outb_p(0, SERPORT + 3);
	outb_p(0xC7, SERPORT + 2);
	outb_p(0x0B, SERPORT + 4);

	// Bit 3 Enable Modem Status Interrupt
	//	Bit 2 Enable Receiver Line Status Interrupt
	outb_p(0x0C, SERPORT + 1);

	counter1 = rtai_rdtsc();
	rt_printk("rtai_rdtsc() returns %lld \n", counter1);
	counter1 = rt_get_cpu_time_ns();
	rt_printk("serial done, rt_get_cpu_time_ns() returns %lld \n", counter1);


	//initialize parallel port stuff

	ret = rt_request_global_irq(7, (void *)handler);
	rt_enable_irq(7);
	outb_p(0x10, PARPORT + 2);		//set port to interrupt mode; pins are input

	counter1 = rtai_rdtsc();

	rt_printk("rtai_rdtsc() returns  %lld \n", counter1);
	counter2 = rt_get_cpu_time_ns();
	rt_printk ("parallel done, rt_get_cpu_time_ns() returns %lld \n", counter2);

	rt_printk("Ports done. You should see the latency message\n");
	return 0;
}

void xcleanup_module(void)
{
	rt_printk("Unloading  parallel port latency test\n");
	rt_disable_irq(7);
	rt_free_global_irq(7);
	rt_printk("Unloading serial port test\n");
```

```
        rt_disable_irq(4);
        outb_p(0, SERPORT + 3);  // disable DLAB
        outb_p(0, SERPORT + 1); // disable serial ints
        rt_free_global_irq(4);
}

module_init(xinit_module);
module_exit(xcleanup_module);
MODULE_LICENSE("GPL");
```