System development for characterisation of high precision frequency standards performance for radio astronomy and industrial applications

AnKuo Lin (0791949)

A dissertation submitted to
Auckland University of Technology
in partial fulfilment of the requirements for the degree of
Master of Computer and Information Sciences

2009

School of Computing and Mathematical Sciences

Table of Contents

List of Abbreviation and acronyms

AVAR	Allan Variance
ADC	Analogue-to-Digital Converter
CSA	Chip-Scale Atomic Clock
CIPM	Comité International des Poids et Mesures
DAVAR	Dynamic Allan Variance
FFT	Fast Fourier Transform
FDD	Feature-Driven Development
GNSS	Global Navigation Satellite Systems
GLONASS	GLObal NAvigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HVAR	Hadamard Variance
HFS	Hyperfine Structure
ISO	International Organization for Standardization
IP	Internet Protocol
IRASR	Institute for Radio Astronomy physics and Space Research
LPTF	Laboratoire Primaire du Temps et des Frequencies
MHVAR	Modified Hadamard Variance
MAC	Miniature Atomic Clock
MVAR	Modified Allan Variance
NASA	National Aeronautics and Space Administration
NBS	National Bureau of Standards
NIST	National Institute of Standards and Technology
NPL	British National Physical Laboratory
OAVAR	Overlapping Allan variance
PPS	Pulse Per Second
PSD	Power Spectral Density
PTB	Physikalisch Technische Bundesanstalt
PM	Phase Modulation
RAD	Rapid Application Development
RAM	Random Access Memory
SI	Système International d'Unités
TVAR	Total Variance
XP	eXtreme Programming

List of figures

List of table

Attestation of authorship

I hereby declare that this dissertation is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person in fulfillment, partial or otherwise of , any other degree or diploma of a university or other institution of higher learning, except where due acknowledgment is made in the acknowledgments.

Signature: _____

AnKuo Lin

Acknowledgement

For the completion of this dissertation, I would like to take the opportunity to express my honest gratitude to those people who have assisted me whatever directly or indirectly.

First and foremost, I am indeed indebted to my supervisor Tim Natusch. I could not have finished this dissertation without his decent guidance and enthusiastic support. His penetrating advice and comment have led me to a clear direction in the entire process of this research.

Furthermore, my thanks will go to the supports from Labjack support team, especially for Paul. His kind correspondence has provided me the straight-forward solution of dealing with the hardware issue on Labjack during the research.

Thirdly, I would like to thank all my friends whose support has been accompanied throughout my studies, particular for Xi Wang who has been patiently correcting my writing.

In the end, I will like to express my deepest thankfulness to my parents and younger brother. In spite of they are far away from me, their endless caring and encouragement with love are the motivations that make me progressive on the most arduous moment of doing this research.

Abstract

The IRASR has many projects associated with the operation of the new AUT radio telescope station. High precision atomic clocks, oscillators regulated by the atomic properties of matter play a vitally important role in the systems operation. Like any similar device, there are errors and a limit to the precision of the behaviour of these oscillators. The Allan variance is a well-known means to characterise frequency oscillators and clocks, especially for those measurement instruments of ultra-high precision. By taking advantage of the rapid application development approach, a system has been implemented within the Matlab® development environment for the measurement of Allan variance. Three major atomic frequency standards, a Rubidium atomic clock, GPS disciplined clock, and Hydrogen Maser, were utilised as the source of references in experiments that tested the developed system. Although the system developed finally meet all project requirements, the AVAR plotting results for different atomic frequency standards show that there is uncertainty on the level for specific tau. Thus, some limitations and recommendations have been identified to assist any future experimentation in this field.

Chapter 1 Introduction

The science of time keeping is one of the most important sciences for the conduct of modern human life. From the very beginning, the activities of human beings have been regulated by and require knowledge of time. Examples range from knowing the time to plant specific crops, for conducting work activities, for navigation and for regulating the performance of high speed computer networks. The precision with which we require knowledge of time varies enormously; to determine the correct season for planting crops we may only need to determine time to the nearest week, for advanced scientific techniques we may need to be able to reliably and accurately distinguish time at the picosecond or finer level. Time standards, clocks we can rely on to tell the time with a level of accuracy and precision appropriate for a particular application, play an important role in many scientific developments. We must understand the characteristics of these standards if we are to properly deploy them. In particular we must acknowledge that no measuring equipment is either absolutely accurate or precise, there are always intrinsic errors that occur because of variation of environment, mechanical mistakes in construction or a multitude of additional external factors. (Tomatis, Orgiazzi & Mulassano, 2008) Therefore, a means of characterizing the behaviour of varied frequency standards is essential for the proper use and calibration of high precision standards. This was the motivation behind the invention of the Allan deviation or variance. This measure of stability has been widely deployed by organizations all around the world. One of the most significant applications of atomic clocks is the Global Navigation Satellite System, which is commonly known as Global Positioning System by United States, GLONASS by Russia, and Galileo system by European Union. In addition, atomic clocks also play an important role in high precision astronomical applications. Back to the ancient time, human had made calendars by observing the orbit of stars in the sky. Since the invention of optical telescopes in the modern age, tracking stars with optical telescope has become very commonplace. With the advance of astronomy, radio telescopes now provide opportunities that allow humans to not only see further, but also over a greater range of the electromagnetic spectrum. However, the signals intercepted from a remote celestial object will be very weak when they are received by a radio dish at any observation point on the earth. Increasing the size of the telescope improves signal to noise ratio and also to filter the valid signals, but the costs are found to be relatively much more expensive for the infrastructure construction. So, a more economic and effective way through the collaboration of multiple observation points and Interferometry has been developed. With such a technique, it is essential that all observation points track the same target at the same time. Time synchronization by atomic clocks is the vital ingredient that makes this possible.

In this research, a simple and economic system for the calculation of Allan variance (a means to determine the stability of standards) will be implemented. It will follow the rapid software development methodology because of limited budget and development time. Matlab® is chosen as the programming language as it has built-in solutions for mathematical algorithms and provides the GUI for design. In the end of this research, a comparison of the result of Allan variance for various atomic time standards will be presented as well as suggestions for future developments that address some uncertainty about the technical details of computation of Allan Variance (AVAR) as invoked in this research. Some limitations of this research will be noted to help avoid problems should further development be undertaken in the future. In summary, the operation of the system developed from this research has given certain levels of satisfaction against the system requirements. It is assumed that further experiments could be implemented with the participation of a frequency counter to clarify the confusion faced in this research.

1.1 Research objective

The primary objective of this research is the development of an Allan Variance measurement system that can characterize the performance of high precision atomic time and frequency standards. More specifically we can state the goal of the work as;

“To develop a software-based system to characterize the performance of high precision frequency standards for radio astronomy and industrial applications.”

In order to achieve the goal of this research, a literature review aimed at gaining an understanding of the operation of different atomic clocks, the calculation of Allan Variance and the common methods of software development has been conducted. Following the flow path of design science research methodology, a program of rapid software development that includes practical experimentation on varied atomic frequency standards has been conducted.

1.2 Research contribution

This research could be implemented with the outcome of contribution in two fields.

- **Academic contribution**

The outcome of this research makes a contribution to the studies of signal engineering and software development. It provides the foundational knowledge for constructing a software-based application for signal computation, especially for the utilization of Matlab® to this field of high precision timing. Moreover, the literature review provides an overview of the characteristics of different types of atomic time and frequency standards, and methods for calculation of Allan variance.

- **Practical contribution**

This research will produce a solution for the characterization of a series of precise time and frequency standards. By utilising this system, a suitable atomic time reference can be selected to satisfy the requirement of time stability for a specific project. This research will also provide a fundamental programming platform for the further development and integration of high precision timing systems for the new AUT radio telescope operated by the Institute for Radio Astronomy and Space Research (IRASR). Furthermore, some possible difficulties and issues of atomic time reference characterization will be identified in this research and recommendations made that will help to avoid problems in future projects.

1.3 Research structure

The outcome of this research has been organized into 6 chapters. To begin, an introduction will describe the motivation and background for this research. Then, the literature reviews in chapter 2 present the base of relevant knowledge; the development and operational principles of atomic clocks, the calculation of Allan deviation, and the methods of software development. In chapter 3, the methodology used for this research will be cited and the reasons for choosing this methodology will be explained. In following chapter, the system development is discussed. Detailed specifications of the equipment used in the research are presented. Experiments implemented for the purpose of initial design are discussed and then followed by a description of the rapid software development process that was the backbone for system development. After that, in chapter 5, comparisons of the results of Allan Variance for various atomic clock references will be presented and discussed. Some limitations for the system will be revealed and their impact on further development considered. We also provide some recommendations to guide any further improvement of the system. Finally, in chapter 6, each chapter is briefly reviewed and final conclusions of this research are made. The entire research structure is represented in the following diagram.

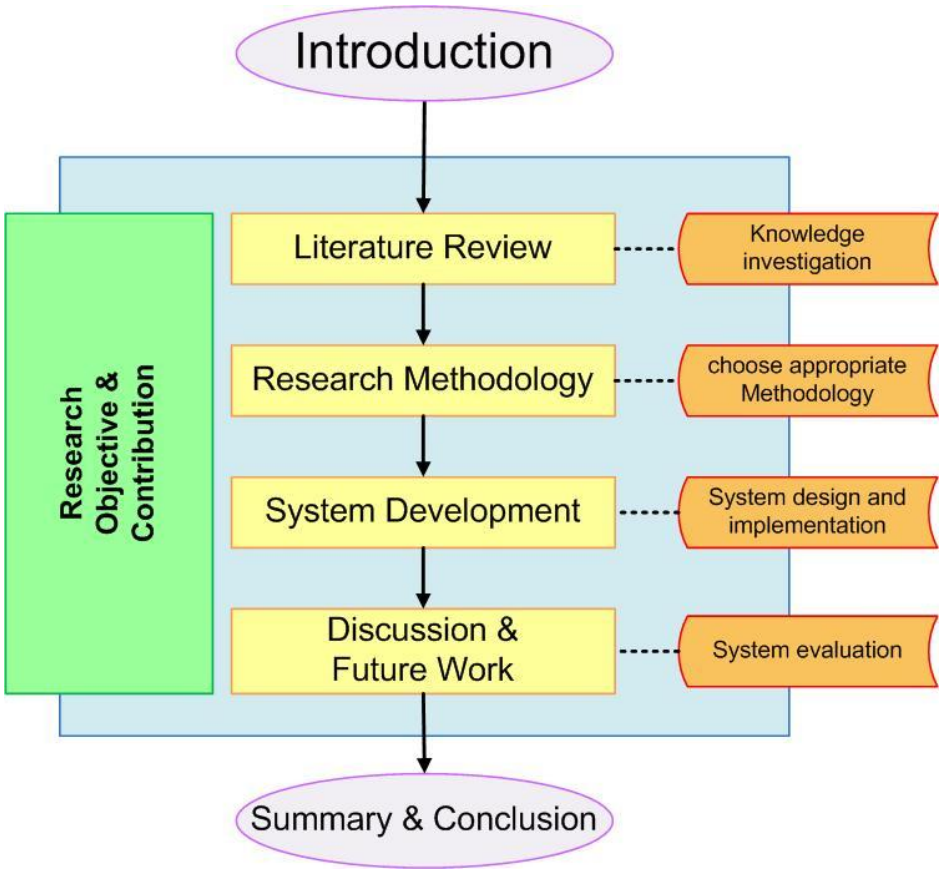

Figure 1.1: Research structure

Chapter 2 Literature Review

2.1 Atomic clock

In this section, a brief history of the development of a variety of atomic clocks will be firstly described. Applications of atomic clocks will be examined to highlight the importance of atomic clocks in our living context.

2.1.1 History

Broadly speaking, the inspiration for atomic clock development can be traced back to 1873; "*A more universal unit of time might be found by taking the periodic time of vibration of the particular kind of light whose wave length is the unit of length.*" James Clerk Maxwell wrote in his book "*Treatise on Electricity and Magnetism*". (Ramsey, 2005) In 1938, Rabi successfully experimented with molecular beams by using a magnetic resonance method based on the studies of Dunoyer in 1911. In 1945, one year after Rabi was awarded the Nobel Prize, his acceptance lecture was written up by William Lawrence of the New York Times as the first publication on atomic clocks. (Ramsey, 1991) The first operational atomic clock was soon established in 1948 by Lyons at the National Bureau of Standards (NBS) in United States, the predecessor of National Institute of Standards and Technology (NIST). (Forman, 1985) In this first atomic clock, the ammonia molecule resonance at 24 GHz was used to discipline a quartz oscillator. At this stage in the 1940s, the accuracy was still not able to compete with the best pendulum or crystal time standards available. In 1955, the first operational Caesium (Cs) beam atomic clock was introduced to the world by Essen and Parry at British National Physical Laboratory NPL (Audoin & Guinot, 2001, p. 53) This was followed in 1959, by development of the Hydrogen Maser clock that relied on detection of stimulated emission at 21 cm of the neutral Hydrogen atom. The first commercial Hydrogen Maser atomic clock was manufactured by Varian Associates. Around that time, Dehmelt investigated the possibility of microwave resonance by observation of the reduction in intensity of light passed through a cell where atoms in the form of a buffer gas exist. This finding eventually materialized into the Rubidium atomic clock. (Ramsey, 2005)

In 1967, after repeated observations of the hyperfine structure (HFS) oscillations of the Caesium atom the International Committee on Weights and Measures defined the second as being equal to 9,192,631,770 periods of the radiation corresponding to the transition between

these two levels of the ground state of the Caesium 133 atom. (Peik, 2006, p. 4)

By the 1980s, optical pumping techniques using semiconductor lasers started to be employed for development of yet further types of atomic clocks.

It is worth noting that the studies and development of atomic clocks has rewarded a number of scientists with the Noble prize; I. Rabi in 1944, A. Prokhorov, C. Townes and N. Basov in 1964, A. Kastler in 1966, N. Ramsey, H. Dehmelt, W. Paul in 1989, C. Cohen-Tannoudji, S. Chu and W. Philips in 1997.

The measurement of time has been advanced dramatically since the first invention of the atomic clock. There have been many improvements that have resulted in the development of more stable and accurate atomic time references. These developments have included longer beams, better excitation methods, reduced size and weight, reversed beams, optical pumping, laser state selection, and improved cavities. (Ramsey, 2005) Higher accuracy and better stability have been achieved every few years for all the different classes of atomic clocks. Accuracy is common expression used in metrology of frequency standards that describes how close a measured frequency to the theoretical ideal frequency of an unperturbed atom. (Audoin & Guinot, 2001, p. 55) According to recent publications from NIST, the Caesium atomic clock could soon arrive at an accuracy of better than 1×10^{-16} from around 1×10^{-10} in 1950s. Moreover, as development of optical frequency standards progresses, they could conceivably achieve accuracies of less than 1×10^{-17}. (Lombardi, Heavner & Jefferts, 2007)

2.1.2 Different Types of atomic clock

2.1.2.1 Caesium beam clocks

The first Caesium beam atomic frequency standard was operated in 1955 by Essen and Parry at the National Physical Laboratory in United Kingdom. From that time, laboratories all around the world have been engaged in research and have contributed to various extents further developments of the Caesium atomic clock. All these laboratories have one identical goal, that is, to produce the best practical realization of the second. Devices developed by these leading laboratories are now used as the so called primary time and frequency standard due to the Caesium beam clocks characteristic of high stability over the long-term period of time, normally many years. For instance, the Caesium atomic standard at Physikalisch Technische Bundesanstalt (PTB) in Braunschweig in Germany has exhibited only 10^{-14} inaccuracy over periods of several years so far.

All Caesium atomic clocks were still using the magnetic deflection technique similar to the initial design by Essen and Parry until 1990 when PTB firstly launched the optical pumped Caesium atomic clock by using semiconducting lasers. Ten years later, the optically pumped Caesium beam clocks in Laboratoire primaire du temps et des frequencies (LPTF) in Paris and NIST both joined the primary standards. (Audoin & Guinot, 2001, p. 129) The use of laser radiation to cool atoms and ions has resulted in the optical frequency standard being able to beat the best microware clocks. (Sterr, 2007) Thus, due to the maturing of the technique of using laser cooling in the past decade, LPTF successfully produced the most precise standard ever by slowing down the moving speed of Caesium atoms in a fountain. This standard was able to achieve 10^{-15} as its stabilty. The stability and accuracy of modern Caesium atomic clocks are still advancing due to the efforts made by scientists around the world.

2.1.2.2 Hydrogen Maser

The invention of the Hydrogen Maser reference was based on the studies done by Ramsey's team at Harvard, USA. (Audoin & Guinot, 2001, p.184) In their experiment, the hyperfine resonance of a Caesium beam was observed successfully, however, the shift on frequency of resonance was found as well. In the attempting to solve this issue, hydrogen atoms were used due to the less perturbation of collisions. However, the hydrogen atoms' ionization energy is too high to be determined by the hot wire method. So, a simulated emission method was then used instead of the method commonly implemented for Caesium atoms. The Hydrogen Maser clock is very stable. However, it is apparent that the accuracy of Hydrogen Maser clock could be kept for only few days. That is because the hydrogen atoms will be collided with the walls of cavity intensely in long duration and this activity will influence the result of resonance resulted in the accuracy of Hydrogen Maser clock. (Rigden, 2002, p. 184-190) Therefore, the Hydrogen Maser atomic clock is very suitable for some application with need of short-term stability. Ramsey (2005) pointed out the advantages of Hydrogen Maser clock are high stability, lower noise, narrow bandwidth, and relatively higher output power. Normally, for Allan variance $(\sigma_y(\tau))$, the Hydrogen Maser could achieve 5×10^{-16} when the time interval between comparisons tau (τ) is at 1000 seconds.

2.1.2.3 Rubidium cell clock

The optical pumping method could be also applied on the Rubidium 87 atom resulted from the appropriate wavelength of semiconductor lasers. Studies of constructing laser-cooled Rubidium fountain reference started from interests of the experimental confirmation that the collisional frequency shift of Rubidium atom is supposed to be smaller than Caesium 133 atom based on the same densities. This finding could not only cut the frequency shift, but could also

increase the frequency stability by higher atom density. However, more comparisons should be done in the future to see the advantage of using various atoms in the fountain application with laser-cooling method, especially for Caesium and Rubidium atoms. As the result, the Rubidium atomic frequency standard could reach the measurement of uncertainty by 1.3×10^{-14}. (Audoin & Guinot, 2001, p. 212-217)

2.1.2.4 Stored ion clocks

The idea of trapping ions for hyperfine resonance was come out in 1960s when the $^{199}Hg^{+}$ ions were used for the observation of hyperfine resonance in successful experiment in NASA. Traditionally, there are two methods of confining ions in vacuum, such as Paul trap and Penning trap, using static electric and magnetic field, and radiofrequency electric field respectively. The frequency instability of below $10^{-13}\tau^{-1/2}$ was reached when the instrument used for ions confinement was made smaller enough. Lately, as similar as the techniques used for other atomic clocks, a laser-cooled mercury ion has achieved the instability of $3.3\times 10^{-13}\tau^{-1/2}$ and the inaccuracy of 3×10^{-15}. In early year of 2009, NIST announced that the Hg^{+} ion could have the maximum systematical uncertainty of 4×10^{-18} where the ion is nearly cooled to Doppler cooling limit and the trap is well balanced. In addition, the systematic fractional frequency uncertainty is achieved as 1.9×10^{-17} for Hg^{+} and 2.3×10^{-17} for Al^{+} ion. (Lorini et al., 2008)

2.1.2.5 Other atomic clocks

Most modern communication and navigation systems require the accurate synchronization, and atomic clocks have been widely adapted as the role as the precise reference for these applications. However, large size of equipments and non-economic power consumption of technologies are the main obstacles to make atomic clocks portable. Whereas the high demands of highly precise timing reference for portable devices, the prototype of Chip-Scale Atomic clock (CSAC) called Miniature Atomic Clock (MAC) come to the reality in 2005. The MAC could be potentially installed in commonly varied portable device, for example, GPS device or mobile phone, with only 10 cm^3 size and <200 mW power consumption. Although it could not reach good long-term stability, it could reach acceptable short-term stability of $\sigma_y(\tau)$<$4\times 10^{-10}\tau^{-1/2}$ where τ =1~100 seconds. (Lutwak et al., 2005) Furthermore, it still has the potentials to achieve the satisfying long-term stability as long as some limitations might be overcome. For instances, the MAC uses some components not found in normal atomic clock, such as the microfabricated vapor cell, the low-power local oscillator, and the laser. And it will be crucial for MAC to control the laser resulted in long-term performance in stability. Jau & Happer (2008) also stated that the complexity and power consumption are still the most

challengeable obstacles of developing MAC, and the laser-atomic oscillator could benefit in manufacturing MAC. Besides, environmental variances are also the constraints for MAC. Using the method of digital temperature compensation which is currently implemented in commercial Rubidium atomic clock could add the frequency correction in corresponds with real-time temperature. (Knappe et al., 2007)

2.1.3 Application

The high precision of time keeping is extremely related to the context of mankind. The application of atomic timekeeping could be seen everywhere everyday in the modern society, for example, Global Navigation Satellite Systems (GNSS), electric power grid, electronic computers, satellites, radio astronomy, communicational device, cellular telephones, and so on. (Zhang et al., 2005 & Lombardi, Heavner & Jefferts, 2007) These applications are highly dependent on the accuracy of atomic clocks. Furthermore, the atomic clocks will play a significant role of the remote signal transmission for the application of deep-space communication in future decades. Those signals transmitted over deep-space communication links will require the syntonization and synchronization by the calibrated atomic time. (Calhoun, Shouhua & Tjoelker, 2007) From the earlier survey of application in space for Rubidium and Caesium atomic clocks (Jeanmaire, Rochat & Boudy, 1997), to the application in space with Hg+ ion atomic clock (Prestage et al., 2003), those researches all determinate the thoughtful meaning of atomic clocks on the application in the space.

The most significant application of atomic clocks can be seen in Global Navigation Satellite Systems (GNSS). The GNSS operates with several satellites that long-term stability atomic clocks installed. There are lots of benefits of using GNSS. Theiss, Yen & Ku (2005) addressed eleven advantages of applying the GNSS, such as timing, surveying, logistics, traffic management, security, marketing, fishing, farming, banking, rental car industry, and weather forecasting. Meanwhile, the author also explained that deploying GPS in the industry will make the work more effectively and efficiently by reduce the unnecessary waste of resource in business. For example, the delivery companies no longer waste their time for finding the destination, and the clients could also trace the location of their goods on the system without the complicated communication with people.

There is also a trend to mobilize the atomic clock, miniaturizing it to make it portable. (Zhang et al., 2005) A miniaturised atomic clock could provide better positioning accuracy for hand-held devices. Firstly, if the GPS signal is poor or dead, the hand-hold device can work independently for a period of time without obvious deviation. Secondly, by integrating the GPS time reference and miniaturised atomic clock reference, it is assumed that the positioning will

be more accurate than only relying on one GPS time reference along. That is because the Global navigation system requires highly precise time reference where the more involvement of precise time references will improve the accuracy in positioning. (Bruggemann, Greer & Walker, 2006)

2.1.4 Summary

In summary, the emergence of atomic clocks has announced that the precise atomic age has come for time keeping. It is also the success for the hard efforts by scientists all around the world within decades. Basically, there are four types of highly precise atomic clock developments, such as Caesium beam clocks, Hydrogen Maser, Rubidium cell clock and stored ion clocks. Each of those atomic clocks is in charged with different application because of the variety of characters. For example, the Caesium beam atomic clocks are widely used as the time reference in general as its high stability over long-term time. The Hydrogen Maser clocks are deployed as the reference for applications over short-term time, because it could be run with only tiny inaccuracy within 1 second to 1 day. The Rubidium cell atomic clock has the advantage for size and cost. It is largely employed in industries due to its satisfied performance and affordable price. Similarly, the Caesium atomic clocks have been commercially used since it was invented. However, the Caesium atomic clock is still more expensive than the Rubidium atomic clock. Recent improvements in optics, are leading to development of ion clocks that are on track to be the next generation of atomic clock for extremely accurate reference (stability less than 10^{-16}). Current GPS satellites were mostly launched before 2000. (McDonald & Hegarty, 2000) The new state-of-the-art atomic clocks have better accuracy than the atomic clocks carried on the satellite. By taking the advantage of these highly precise time reference, in the future, we may assume that the inaccuracy of GPS could be greatly depressed to only 1 meter or less. Not only are people pursuing the performance of timekeeping, but are also paying attention to the increased mobility and portability of further precise timekeeping. That is the motivation of implementation for the miniature atomic clock. In future days, most hand-held devices may be equipped with those MACs for applications, such as quicker global positioning system, synchronization on any class of communication network, or any applications require precision of timing. (Kitching et al, 2005). Currently, we are in a stage where the optical frequency standard is prepared to replace the traditional microware clocks as the future definition of SI unit second in next generation. This encouraged the CIPM to suggest that four optical frequency standards (neutral Sr , Hg^{+} , Sr^{+}, Yb^{+}) could be seen as next representations of the second and could be joint into the list of "*Recommended frequent standard values for applications including the practical realization of the meter and secondary representation of second*". (Sterr, 2007)

2.2 Allan Variance

2.2.1 Theory

Allan variance (AVAR) is a means to demonstrate and characterize the performance of clocks, oscillators, time, and frequency standards. The behaviour of “real world” oscillators or clocks includes non-stationary errors and random noises that cause the average frequency of oscillation to vary with time. Due to this non stationary behaviour, traditional measures of stability such as the average and standard deviation of frequency are not suitable. However, by using Allan variance, common flicker noise and random walk noise (the two main contributors to non stationarity) can be shown to converge to a finite value. (Allan, 2004)

The basic idea of AVAR is to divide the long sequence of data into clusters with specific averaging period of time, τ. For example, imagine we have a stream of continuous frequency data as shown as figure 2.1. Divide the data into clusters of duration in time τ. The example in figure 2.1 shows two such time intervals; if tau is equal to 100 seconds, all bins will be grouped by red line. If tau is equal to 200 seconds, all bins will be denoted by a green line.

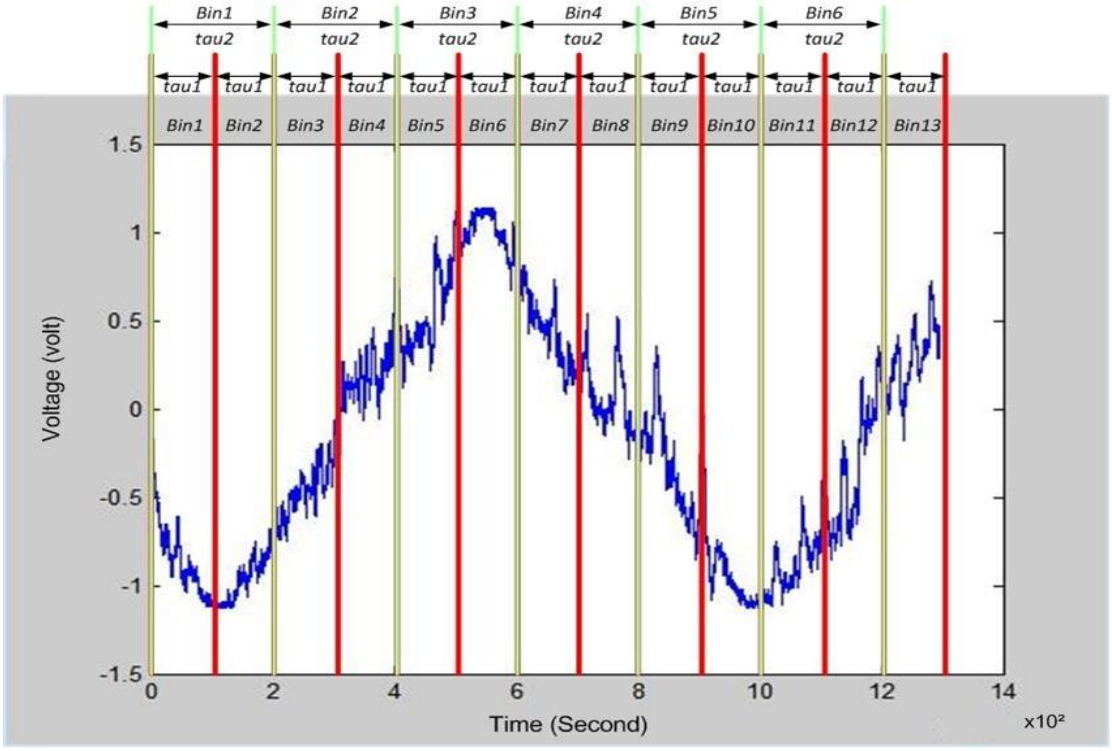

Figure 2.1: Raw data divided into bins by tau 1 and tau 2 for AVAR calculation

After that, the standard deviation of each cluster Δy is calculated. Then Δy is squared, accumulated, and then divided by a rescaling factor to derive the quantitative measurement $\sigma_y^2(\tau)$. The result, AVAR is the square root of $\sigma_y^2(\tau)$.

The equations below show the calculations for Allan variance. (Allan, Ashby & Hodge, 1997)
$\mathrm{AVAR} = \sigma_y^2(\tau) = \frac{1}{2}\langle(\Delta y)^2\rangle$, $\langle\ \ \rangle$ donates the expectation value.
Where $\Delta y = y(\tau)_{i+1} - y(\tau)_i$ and $y(\tau)_i$ is the average value of the frequency variation in the 'ith' time interval / data cluster.

The expression for AVAR can be also expanded as follows.

$$\mathrm{AVAR} = \sigma_y^2(\tau) = \frac{1}{2 \times (n-1)} \sum_i (y(\tau)_{i+1} - y(\tau)_i)^2$$

$$\mathrm{ADEV} = \sigma_y(\tau) = \sqrt[2]{\frac{1}{2 \times (n-1)} \sum_i (y(\tau)_{i+1} - y(\tau)_i)}$$

The signal generated by an oscillator can be mathematically modelled as:
$u(t) = (U_0 + \varepsilon(t)\sin(2\pi v_0 t + \phi(t)))$
Because $\varepsilon(t)$ is a random variation of signal amplitude and does not affect signal frequency it can be neglected, and the value of signal in voltage could be re-written as:

$$v(t) = v_0 + \frac{1}{2\pi}\frac{\Delta\phi}{\Delta t}e$$

Therefore, the frequency offset could be defined as:

$$y(t) = \frac{v(t) - v_0}{v_0}$$

On combining the two equations above, we obtain the following;

$$y(t) = \frac{\frac{1}{2\pi}\frac{\Delta\phi}{\Delta t}}{v_0}$$

As both v_0 and $\frac{1}{2\pi}$ could be seen as constant, $y(t)$ will be represented as:

$$y(t) = \frac{dx}{dt}$$

Where *dx* denotes the fluctuation of phase, and $d\tau$ is the difference of time internal. (Galleani & Tavella, 2003)

After calculating Allan deviations for all sensible values of time interval τ, it is then normal to plot AVAR versus τ on a log-log graph.

According to Greenhall (1992), the classical Allan variance ($\sigma_y^2(\tau)$) is useful for the study of frequency and distributed systems in two ways. First of all, it provides a way to measure the

performance of frequency source by indentifying the frequency stability over a specified averaging time. Secondly, the log tau (τ) versus log sigma $\sigma_y(\tau)$ plotting is also used for the characterization, especially for the operation of residual time $x(t)$ or residual phase $\emptyset(t)$.The author also stated there are two known weaknesses of the classical Allan variance. In the standard Allan variance plots it may not be possible to distinguish white phase noise from flicker phase noise.

Similarly, Howe & Tasset (2004) explain that the major advantage of Allan variance is the sensitive frequency response for a fixed period of time (τ) which (τ) could be a cluster of period of time. One very important point to note is that AVAR cannot measure the frequency stability properly when an interval τ is larger than one-half the length of the data run. For example, if using AVAR to characterize the behaviour of two clocks or oscillators for a month, the AVAR would not be able to represent the frequency stability where the interval is greater than half a month or two weeks.

Because of the drawbacks of AVAR described above several variants have been proposed that address these limitations. For example, overlap Allan variance (overlap-AVAR), Modified Allan variance (MVAR), Total variance (TVAR) and recently the Dynamic Allan variance (DAVAR). (Galleani & Tavella, 2003)

The overlap Allan variance is very similar to the traditional AVAR. The only difference between AVAR and overlap AVAR is the way in which data bins / data clusters are calculated. For the AVAR, it requires the average value for the differences over all bins. In the overlap AVAR however bins are arranged into two sequences with the second sequence starting at an offset to the first (see figure 2.2). All possible offset values (that retain an overlap) of those two sequences are then used derive values that will be squared and summed as below. The summation is further divided by $2m^2(N-2m+1)$, where the current time interval $\tau = m \times \tau_0$, N is the number of initial data. (Riley, 2007)
In term of phase data, the equation is as following,

$$\mathrm{OAVAR} = \sigma_u^2(\tau) = \frac{1}{2(N-2m)\tau^2} \sum_{k=1}^{N-2m} [X_{k+2m} - 2X_{k+m} + X_k]^2$$

As explained by Riley (2007) this can be also defined for the measurement of frequency data as,

$$\sigma_y^2(\tau) = \frac{1}{2m^2(N-2m+1)} \sum_{j=1}^{N-2m+1} \left\{ \sum_{i=j}^{j+m-1} [y_{i+m} - y_i] \right\}^2$$

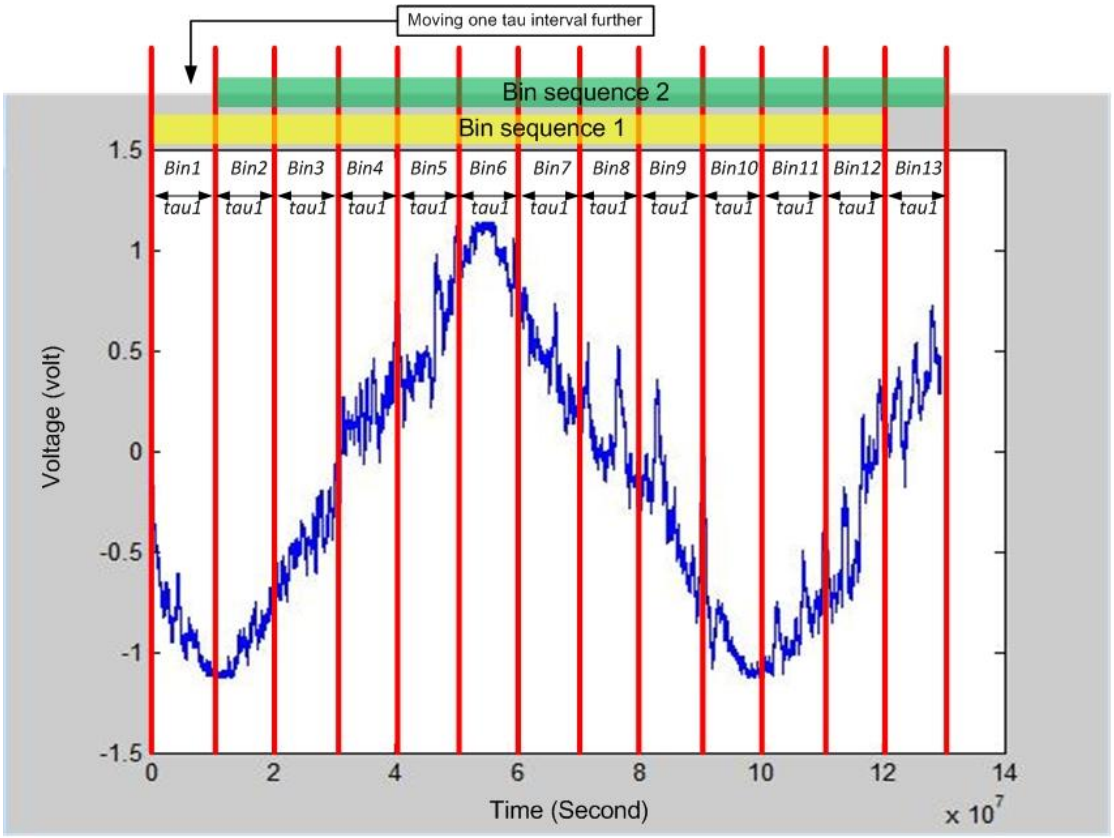

Figure 2.2: Raw data divided into bins by tau 1 to make bin sequence for overlap AVAR calculation

MVAR is another common measurement for frequency stability in the time domain. According to Riley (2007), the Modified Allan variance for the measurement of phase data is defined as;

$$\mathrm{Mod.}\,\sigma_y^2(\tau) = \frac{1}{2\tau^2}\langle(\Delta^2\bar{x})^2\rangle = \frac{1}{2m^2\tau^2(N-3m+1)}\sum_{j=1}^{N-3m+1}\left\{\sum_{i=j}^{j+m-1}(X_{i+2m}-2X_{i+m}+X_i)\right\}^2$$

Where $\{X_k\}$ is the infinite sequence consisting of signal samples $x(\tau)$, which are spaced with sampling interval τ_0. τ is the averaging time, where $\tau = m\tau_0$.

However, when working directly with frequency data the average value of frequency over time interval τ is $y(\tau) = x'(\tau)$, where y_k is the measurement at instant τ_k.

Hence, $y_k(\tau)$ could be defined as,

$$y_k(\tau) = \frac{1}{\tau}\int_{\tau_k}^{\tau_k+\tau} y(\tau)d\tau = \frac{X_{k+n}-X_k}{n\tau_0}$$

Thus, the equation of modified AVAR for fractional frequency sequence $\{y_k\}$ can be expressed as;

$$\mathrm{Mod}\sigma_y^2(\tau) = \frac{1}{2m^4(N-3m+2)}\sum_{j=1}^{M-3m+2}\left\{\sum_{i=j}^{j+m-1}\left(\sum_{k=i}^{i+m-1}(y_{k+m}-y_k)\right)\right\}^2$$

Normally, the modified Allan deviation $\mathrm{Mod}\sigma_y(\tau)$, the square root of result above is used. Like the standard AVAR this is also most commonly plotted using a log-scale diagram with corresponding τ. Bregni & Primerano (2004) and Allan & Barnes (1981) explained that the advantage of MAVAR is its superior ability to distinguish between white and flicker phase modulation (PM) noises.

2.2.2 Application

Allan variance has been the predominant measure for frequency metrology of high stability standards performance for many decades. Cosart, Peregrino & Tambe (1997) explained that Allan variance is commonly used to characterise the behaviour of oscillators, token ring networks, computer clocks, chirp radar and in communication systems. Moreover, Han et al. (2006) and Fletcher & Witt (2008) noted that the Allan variance is a potent tool which could effectively identify different types of random noise within oscillators by their statistical character. AVAR is also widely and internationally adapted for astronomical and satellite applications, especially for the supervision of the highly precise clocks in GNSS (a system that is becoming more and more important day by day) (Sesia & Tavella, 2008) For some satellite and navigation systems that use inertial sensors errors resulting from noise could result in a disastrous decline of performance and clearly the characterisation of noise by using methods such as the, Allan variance and its variants is vital (Grantham & Bailey, 2006).

In conclusion, Allan variance can be applied to any application which requires the characterisation of oscillator behaviour associated with the presence of random noises.

2.3 Software Development

There has been a large amount of discussion on methods for developing software. The main objectives of using an appropriate software development approach are focused on the development of effective, efficient and low cost products. In addition, a further driving force behind these methodologies is to ensure that the software development procedure is on track (time wise) and also that the software will meet the requirements as initially planned. Some software development approaches have been widely adopted for decades with remarkable success, for example, waterfall model, incremental model, rapid application development (RAD), spiral, top-down programming, object oriented model, unified process, and so on.
In the following section, some common software development methodologies will be reviewed as the basis for knowledge required to choose an appropriate way to develop the system in this specific research.

2.3.1 Common software development models

2.3.1.1 Waterfall

The waterfall model adopts a sequential software approach. It follows an activity-centred principle and the process never turns back once the activity is done. (see figure 2.3) It is acknowledged that the Waterfall approach is the reference for other software development cycles. (Guimarães & Vilela, 2005)

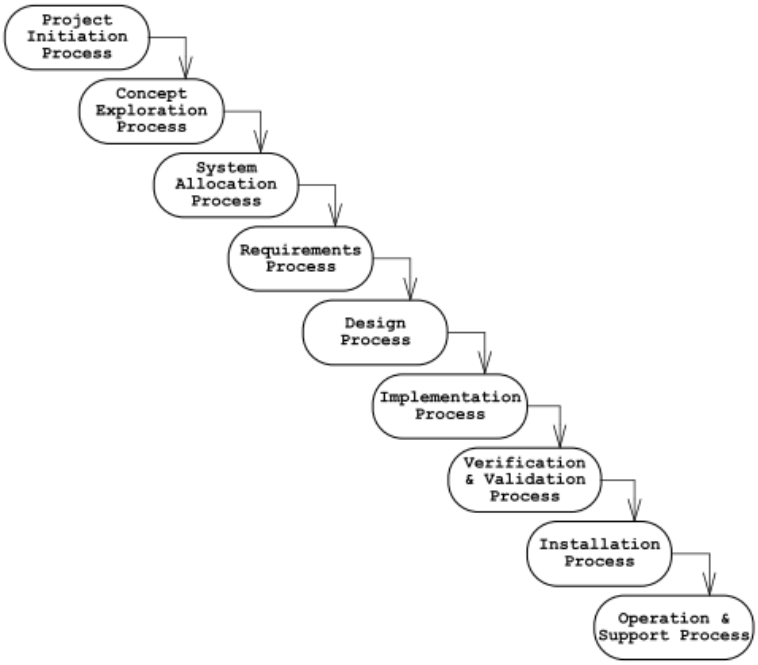

Figure 2.3: Waterfall software development model (Bruegge & Dutoit, 2003, p. 500)

The waterfall model is also a simple software development cycle which implements five basic stages, such as analysis, design, construction, evaluation and maintenance. The main character of this model is the continuous verification activity that guarantees redundant requirements are not propagated forwards nor essential requirements cancelled at any stage of the development process. (Bruegge & Dutoit, 2003, p. 500) Another feature of the waterfall model is that the output of any current process becomes the input of the following process. Therefore, the following process is highly affected by the quality of outcome in the previous process. A major consequence of this is that one small mistake in a process could result in a big problem for any of the later processes.

Whereas there are some drawbacks to the waterfall development cycle, some improvement has been done as time goes by. Nowadays, it is still being used for varied modern projects. Although there is a certain fraction of the market still using the waterfall software development cycle, a survey in New Zealand conducted by Groves et al. (2000) shows that participants from different companies felt the ISO 9001 (basically waterfall) model was not efficient and is costly for maintenance.

2.3.1.2 Feature-Driven Development

According to Highsmith(2002) feature-driven development contains 5 processes that focus on the essential features or components to be developed. FDD has a focus on the simplicity of development and also requires the achievement of primary features. Ramsin & Paige (2008) explained these 5 processes as below.

- Develop an overall model

This process is to define the model for the entire project. The main object of the project must be clarified in this phase. There is no need to point out the detailed features in this phase, but it may be vital to shape the project, to identify what rough outcome this project may produce. It is also the phase to organize every role in the project team and to make some notes for building the feature list in following step.

- Build a feature list

A comprehensive feature list will be made in this phase. The features on the list are the important features which must be completed in the project. The features defined in this phase will be narrowed to the mission specific tasks instead of the technology-centric features that are usually seen as common functionalities. For example, a feature could be to select the top 10 value from specific data sheet, not to access the data from specific database which may be used in other feature.

- Plan by feature

In this phase, weighting and prioritization of features on the list is important. Those features will be planned and prioritised by factors, such as risk, complexity, dependencies, work-load, checkpoints and required milestones. The features should be also scheduled for development.

- Design by feature

Each feature will be designed by an independent method. By different features, there could be many design packages which divide out the workload, allocation of roles, technical development techniques and sequence diagrams for each feature.

- Build by feature

Each feature will be built separately according to individual design. Each feature also requires implementation, inspection, and testing.

Overall, FDD has some advantages, such as ease of adding other features in the future, light

up-front modelling, and easy-achieved processes. (Highsmith, 2002) However, Hunt (2005, p. 175-178) explained that it is critical to start with a good design of solid system architecture which is indispensable for the success of FDD. As FDD is developed by features, it is easy to evolve into a disorganised and unwieldy system. A well-designed architecture will be flexible enough to deal with future evolvement.

2.3.1.3 Extreme Programming (XP)

Extreme Programming is designed for system development in a small, co-located and skilled team. (Highsmith, 2002 & Abrahamsson, 2003 & Succi et al., 2001) Succi et al. (2001), Hightsmith (2002) and Macias et al. (2003) classified the practices of XP into 12 principles, such as the planning game, small releases, metaphor, simple design, refactoring, testing, pair programming, collective ownership, continuous interaction, 40-hour week, on-site customer, and coding standards. The values brought by XP are the communication, simplicity, feedback, courage and quality of work. Communication will concentrate on the face to face communication among people, for example, the developers, clients and the management level, and shun using technology. That will reflect in that the inspiration will be exchanged easier around the team members and some unnecessary misunderstanding can be avoided. Simplicity means that the process of XP should be kept as simple as possible at all stages from design, construction, to maintenance. If necessary a new small cycle development is rather to be added in the project instead of extending the original cycle. Keeping the development simple is expected to bring the benefit of lower cost of change in the future as well. Feedback is very important for XP development. Feedback can be both internal and external, from the developers or clients. Whereas the XP developing environment is changeable, a good understanding of feedbacks will help the developer to solve the problems effectively and efficiently. Courage means that all members in the team should have self-confidence to detect the problems. In other words, if there is a challenging program happening during a project, it will be acceptable to try new solutions. Also, if some tasks done formerly go wrong, it will be imperative to just abandon the task and start a new task instead of fixing it repeatedly. Following the principles of the XP development method, the quality of work is ensured as the people involved in XP development should desire high-quality work will be done by the end of project. In practice, pair-programming will make certain that the uncovering of programming defects can be inspected in a cost-effective way. The participants will have the ability to identify the problem and have bravery to try a fix or to innovate. The quality of work is controlled and reviewed by all participants during every single part of the development cycle.
XP is an iterative and agile software development method which innovatively has an on-site

customer representative as a member of the development team. It also allows the requirements to be changed during the process of development. The emphases of XP are focused on productivity, flexibility, teamwork, informality, creativity and innovation.(Macias, Holcombe & Gheorghe, 2003). The development of XP is also based on diverse short-cycle developments by completing a subset of requirements under a more complicated requirement set. Abrahamsson (2003) mentioned that the process of XP could be grouped into short development cycles, incremental planning, evolutionary design, and sensitive response to the changeable requirement from business. The author also conducted a report showing that the performance of product had improved about 26% and productivity had increased nearly twice from the first release of product to second release.

2.3.1.4 Rapid Application development

As Howard (2002) stated, the aim of RAD is to trim down the time between the requirement specification and implementation. In other words, there will be less likelihood to change the application at the end of development as all key requirements have been reviewed and implemented iteratively. However, some people claim that the RAD may not meet the satisfied level of quality caused by a rough and dirty development cycle. Howard (2002) described that it is frequently acknowledged that the speed of development is incompatible with the quality of development. The developer can likely only perform one or the other best (speed or quality), but not both at the same time. Nevertheless, from the point of RAD supporters, modern quality disciplines can be readily implanted inside RAD and improve the suitability of purpose, avoid wastage, ensure things are done right at the beginning, personal responsibility for quality is understood, and that the correct requirements of the customer must be understood. Besides, according to Jim Highsmith (2002), one significant difference between RAD and traditional methodologies is that the functionalities of application are allowed to change when new things are learnt over the period of the software development life cycle. In contrast, the traditional software development prefers the functionality stay fixed once the requirement has been confirmed in the early development stage.

Agarwal, Prasad, Tanniru & Lynch (2000) explained that there is no clear definition for RAD. However, RAD could be treated as the simplified methodology which advocates the iterative developing phases for software development. Also, RAD could be seen as a set of usage of tools which has features, such as object development, reusable code for client and server application, and graphical user interface. Similar to the points noted formerly, these authors also identified that RAD is often argued against because the speedy implementation

sometimes cannot deliver satisfactory quality in the software development cycle. Apart from this opposition, it is undoubted that RAD is a timely and effective solution for software development within the modern software environment.

2.3.1.5 Spiral Model

Another software development model revised from the Waterfall model is the spiral model devised by Boehm in 1987. The spiral model improved the fault that existed in the Waterfall model which allows the occasional alteration of completed activities. One important contribution is that the prototyping idea is associated with every activity in the spiral model. Basically, each development round in spiral model consists of four phases, such as determine objectives, find alternatives, and constraints in first phase (top-left quadrant), alternatives evaluation, risk identifications and solutions in the second phase (right-top quadrant), development and test in the third phase (right-bottom quadrant), and next round planning in the fourth phase (left-bottom quadrant). The last phase could be seen as a review from the points of all participants in the project, such as developer, users, and clients. Boehm grouped all rounds in the spiral model into several milestone rounds, such as *concept of operation, software requirements, software product design, detailed design, code, unit test, Integration and test, acceptance test, and implementation*. (Boehm, 1988) All these rounds are associated with the activities within four quadrants.

One of the main focal point of the spiral model is the ability to incrementally address the risks by its precedence. (Fairley & Willshire, 2005 and Bruegge & Dutoit, 2003, p. 502-505) As the spiral model attempts to identify the risks and to provide alternative solutions as detailed as possible, the development time can be longer than other development models, therefore, it will suit for the projects demanding high degree of accuracy, especially for large and expensive projects with high priority of risk avoidance. However, Boehm (1988) explained that the project manager for spiral model use should be fully-skilled and experienced for dealing with its complex characteristic. Moreover, regarding the quality control in spiral mode, Hashmi & Baik (2007) concluded that a major similarity between the spiral model and XP is the focus on the assurance of the quality for products. Iteration is a fundamental component of both spiral and XP models. In both cases, development does not proceed to the next step before passing inspection of the current development step.

On the other hand, even if the spiral model covers comprehensive details for every development stage, loose deadlines could incur the indistinct termination condition for ongoing

cycle. Indeed, that could be most risky disaster to projects, running out of budget and time. (United States Department of Health & Human Services, 2008)

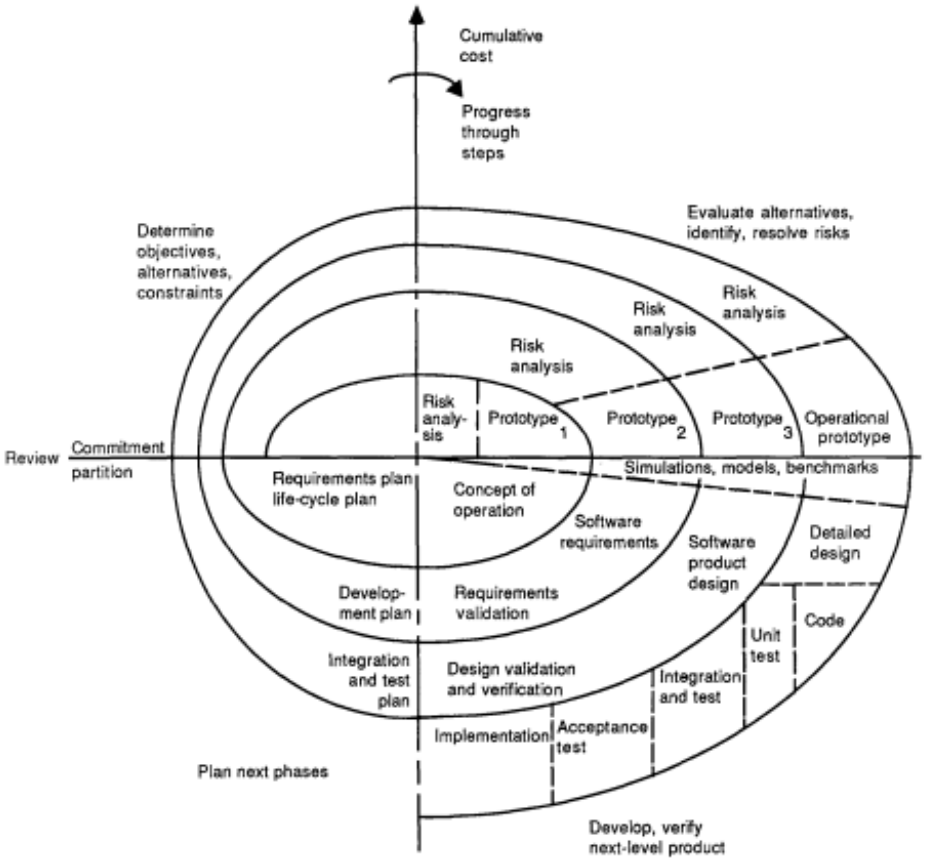

Figure 2.4: Spiral software development model (Boehm, 1988)

2.3.1.6 V-Model

V-model is also a variant that improves on the traditional Waterfall model. It explicates how the development activities relate to the verifying activities. The basic structure of V-model is represented in figure 2.5.

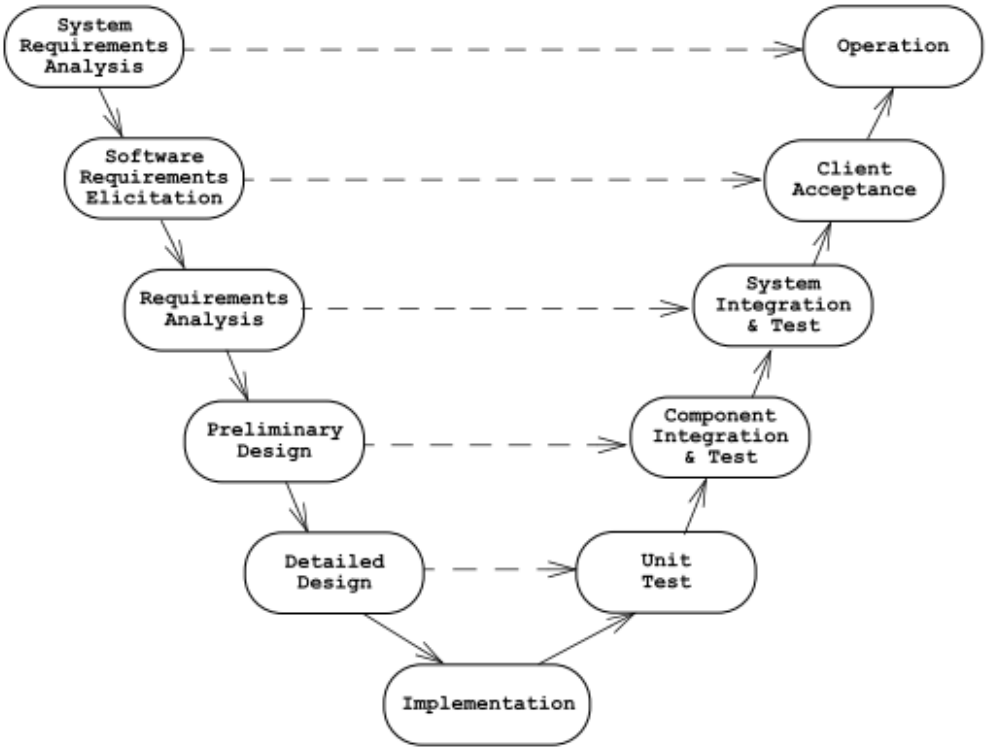

Figure 2.5: V-model software development (Bruegge & Dutoit, 2003, p. 503)

The main diversity between V-model and Waterfall model is that the V-model makes a clear distinction of the intrinsic character of each level of development. Activities from requirement analysis to implementation have been split into more detailed phases, which correspond to the more detailed activities from implementation to system validation. In terms of the V-model operation, it provides a high-level abstraction for participants to satisfy the system requirements. The V-model also focuses on the dependence between the left side activities, the design stage, and the right side activities, the testing stage, to solve any problems in the entire software architecture. In the middle-bottom of V-model there is a concentration on implementation and assembling the coding. (Bruegge & Dutoit, 2003, p. 502)

As V-model is a variant it inherits the benefit of simplicity of the software development process from the Waterfall model. However, it also takes over the common defect from Waterfall model that the procedure will not go backward when previous activity has been reviewed and finalized to be the baseline of whole development. That means this software development will only have no conflicts when there is assurance that the initial requirements will not be changed. Unfortunately, most projects in reality are not idealized. More often than not, alterations happening in the activity being processed call for the change in prior activities. (Pyhäjärvi, Rautiainen & Itkonen, 2003)

The main benefit of V-model will be reflected in clear discrimination over each level in testing phase and also the relative link between development and testing phases. However, all phases are often interconnected during the development. It is difficult to rigorously make a distinction for every phase in reality. (Yao et al., 2006)

2.3.2 Summary

A software development methodology is the principle employed to design, plan, implement, and manage the steps of development for an information system. From the traditionally linear software development, Waterfall, to varied emerging software development methods, XP, RAD, FDD, V-model, Spiral model and so on, the objective is the same, to make the development easier and to avoid some possible risks that could bring failure to the project. There is no such perfect software development method that will suit all kinds of project. In other words, there are plenty of variables that influence the decision of choice of an ideal software development approach. Each methodology has relative advantages and disadvantages for the development in a particular situation (Dalcher, Benediktsson & Thorbergsson, 2005). The survey conducted by Dalcher, Benediktsson & Thorbergsson (2005) for the comparison of Incremental, Evolutionary, XP, and V-model methodologies also showed that only participants involved in a development using the V-model have strong preference for using other methodologies. Software development is a complex process that depends on individual judgements, and it should not be restricted by corporeal laws. (Guimarães & Vilela, 2005) However, it is clear that the main discipline of software engineering should be to embrace the cost-effective perspective. This may also explain that the software methodology with agility often gain the favour of the project manager due to the intention that modern projects may try to diminish the cost and shorten development timeframes. (Sommerville, 2004, p. 436-438)

Chapter 3 Research Methodology

The purpose of this research is to develop a software-based tool for determining statistical properties of highly stable atomic clocks and to examine ways in which this could be applied to improve the performance of an ensemble of various atomic clocks. For instance, in an assumed scenario, there are five highly precise atomic clocks available for use. Each clock will have individual characteristics. Some may have desirable short-term stability and uncertainty and some will be very reliable for long-term stability. Suppose now a project requires very stable time reference for long-term usage. Therefore, we could use this analysis tool to examine these five atomic clocks interactively and establish a scoreboard for their performance. From the rank, top three suitable atomic clocks will be selected out of five. The technique called 'three cornered hat' (Vernotte et al., 2004) could then allow the creation of a unique time reference which is more precise than any of the three selected atomic clocks.

The background that prompted this current research is that there is a need for an analysis tool to characterize the varied atomic clocks of the IRASR. The design science approach will be used as the backbone of this research. Design science approach is an assembly of activities that intends to solve a problem in an inventive or creative way. (Venable, 2006, p. 2) Adikari, McDonald and Collings (2006) described that the natural science focuses on the exploration of existing phenomena whereas the design science is concentrated on how to alter existing things to attain an artificial goal. Adam and Courtney (2004) stated that the research in information system should be bi-directional. It is not only beneficial for the academic contribution, but it should be also linked to the needs and practises of industrial communities. Using the design science approach emphasises the value of facilitating the achievement of demands or needs from business communities. According to Vaishnavi & Kuechler (2008, p. 19), the design science research can generally be classified into five steps; awareness of problem, suggestion, development, evaluation and conclusion. In the first phase, the problem of instability and uncertainty of highly precise atomic clock time reference will be recognized. Realizations of the operational nature and the statistical quantification of performance for diverse atomic clocks will be carried out in this stage. Secondly, the Allan variance will be studied as the suggested technique for measuring the stability of different atomic clocks. Also, two main requirements of the analysis tool will be identified, such as the functionality of plotting Allan variance and specific timeframe of data recording. In the development phase, the emphasis will be to create a system for acquiring raw data and producing Allan variance plots according to the discussion in evaluation phase. Due to limited time, the rapid software developing approach is to be utilized.

The rapid application development model with iterative and incremental features will be driven to implement the system in this research. It will ensure the development timeframe can be minimised, and the performance and satisfaction of system be maximised to meet the initial system requirements. The basis of rapid application development structure is represented as in figure 3.1.

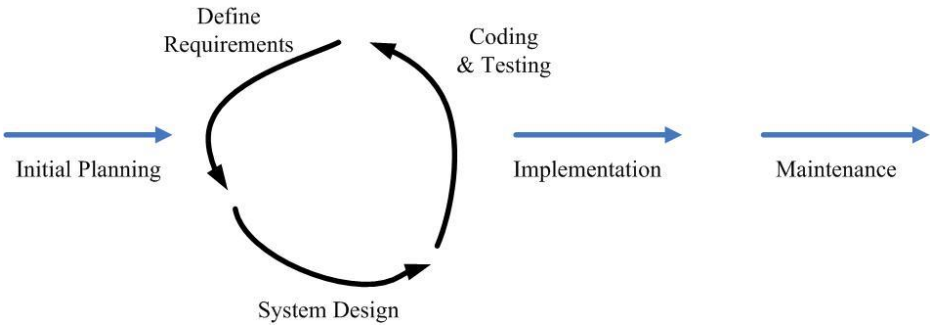

Figure 3.1: Iterative and incremental rapid application development workflow

Next, performance of system and result of plotting will be evaluated for efficiency and effectiveness for the purpose of future improvement, optimization and extension. Eventually, conclusions will be made and presented as the outcome of this dissertation. Overall, the research work flow combined with design science approach is as in the diagram below.

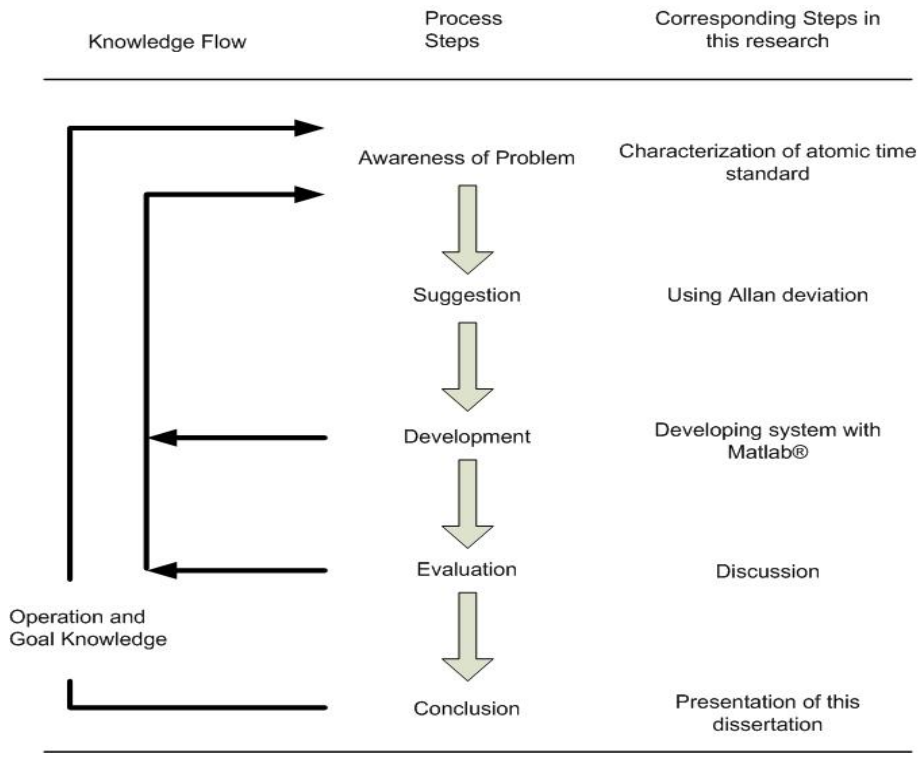

Figure 3.2: Application of design science approach by Vaishnavi & Kuechler (2008) in this research

Chapter 4 Software Development and Experiments

4.1 Experiment equipment

4.1.1 Rubidium atomic clock

The Rubidium atomic reference used in the experiments is a commercial type Rubidium atomic clock made by Stanford Research Systems®, model is FS725. Basically, it will be used as one of the oscillators that provide 10 MHz reference in the following experiments. This Rubidium atomic clock can be adjusted from an external 1 pps GPS disciplined signal for the sake of calibration.

The basic specification of the SRS FS725 Rubidium atomic clock is shown as table 4.1.

Figure 4.1: Stanford Research Systems® FS725 Rubidium atomic clock

Figure 4.2: Rear panel of Stanford Research Systems® FS725 Rubidium atomic clock

Physical Dimensions	215 mm (width) × 89mm (height) × 330 mm (length) Weight: 4 Kg
Output frequencies	10 MHz sine, 5 MHz sine, 10 µs wide 1 pps pulse
Amplitude	0.5 Vrms, ±10 %, (both 5 MHz and 10 MHz outputs)
1 pps pulse amplitude	2.5 V into 50 Ω, 5 V into high-impedance loads
Phase noise (SSB)	<-130 dBc/Hz (10 Hz) <-140 dBc/Hz (100 Hz) <-150 dBc/Hz (1 kHz) <-155 dBc/Hz (10 kHz)
Spurious	<-100 dBc (100 kHz BW)
Harmonics	<-60 dBc
Accuracy at shipment	$\pm 5 \times 10^{-11}$
Aging (after 30 days)	$<5 \times 10^{-11}$ (monthly) $<5 \times 10^{-10}$ (yearly) 5×10^{-9} (20 years, typ.)
Short term stability (Allan variance)	$<2 \times 10^{-11}$ (1 s) $<1 \times 10^{-11}$ (10 s) $<2 \times 10^{-12}$ (100 s)
Frequency retrace	$\pm 5 \times 10^{-11}$ (72 hrs. off, then 72 hrs. on)
Stability	$<5 \times 10^{-12}$
Trim range	$\pm 2 \times 10^{-9}$ (0 to 5 VDC) ±0.5 ppm (via RS-232)
Warm-up time	<6 minutes (time to lock) <7 minutes (time to 1×10^{-9})
Operating temperature	+10 °C to +40 °C
Temperature stability	$\Delta f/f < \pm 1 \times 10^{-10}$ (+10 °C to +40 °C)
Storage temperature	-55 °C to +85 °C
Magnetic field	$\Delta f/f < 2 \times 10^{-10}$ for 1 Gauss field reversal
Relative humidity	95 % (non-condensing)

Table 4.1: Specification of Research Systems® FS725 Rubidium atomic clock

4.1.2 Hydrogen maser atomic clock

The active Hydrogen Maser MHM 2010™ manufactured by Symmetricom® will be the most accurate time reference for short-term time we will use in the experiments. The stand-alone

cavity switching auto tuning technique of the MHM 2010™ allows it to deliver the time reference with levels of long-term stability, which is rare for a Hydrogen Maser and commonly seen in Caesium atomic clocks. The elementary specification of MHM 2010™ is presented in table 4.2.

Figure 4.3: Symmetricom® MHM 2010™ Hydrogen Maser

Physical Dimensions	1066 mm (height) X 457 mm (width) X 760 mm (depth) Weight: 215 Kg (without batteries)	
Temperature sensitivity	<1.0E-14/°C	
Magnetic sensitivity	<3.0E-14/Gauss	
Power source sensitivity	<1.0E-14	
Synthesized frequency resolution	7.0E-17	
Frequency control range	7.0E-10	
Available inputs and outputs	5 MHz, 10 MHz, 100MHz outputs with frequency amplitude 13 dBm, also 1pps input and 1 pps output	
PHASE NOISE	5 MHz Output	10 MHz Output
1 Hz	≤-112dBc	≤-106dBc

10 Hz	≤-130dBc	≤-124dBc
100 Hz	≤-148dBc	≤-142dBc
1 KHz	≤-155dBc	≤-149dBc
10 KHz	≤-155dBc	≤-149dBc
100 KHz	≤-155dBc	≤-149dBc
Standby battery	8 hours operation	

Table 4.2: Specification of Symmetricom® MHM 2010™ Hydrogen Maser

4.1.3 Global Position System disciplined clock

In this research, we also compare the local atomic time reference against the GPS disciplined time reference. The GPS disciplined clock used is manufactured by Trimble® Navigatiion Ltd and called the ThunderBolt™. The GPS system operates by sending signals including the time from the Caesium atomic clocks aboard orbiting satellites. The ThunderBolt™ operates on the GPS L1 frequency, and supports up to 12 continuously tracked signals from satellites. It provides two outputs, a 10 MHz signal reference signal and 1 pps signal output. The specification of ThunderBolt™ is summarised in table 4.3

Figure 4.4: Trimble® ThunderBolt™ GPS disciplined clock

Physical Dimensions	127 mm (length) x 102 mm (width) x 40 mm(height) Weight: 285 Kg
Update rate	1 Hz
PPS accuracy	UTC 15 nanoseconds (one sigma)
10 MHz accuracy	1.16×10^{-12} (one day average)
Harmonic level	–40 dBc/Hz max

Spurious	–70 dBc/Hz max
Phase noise	
10 Hz	–120 dBc/Hz
100 Hz	–135 dBc/Hz
1 kHz	–135 dBc/Hz
10 kHz	–145 dBc/Hz
100 kHz	–145 dBc/Hz
Operating temperature	0 °C to +60 °C
Storage temperature	–40 °C to +85 °C
Operating humidity	95% (non-condensing)

Table 4.3: Specification of Trimble® ThunderBolt™ GPS disciplined clock

4.1.4 Signal generator

A signal generator is a device designed for generating electrical signals. The signal generator in charge in our experiments is made by Rohde & Schwarz®, Inc. It can produce 10MHz frequency out from both internal and external references with the ability of amplification. In addition, it also provides a function with low frequency output which allows sampling the high input frequency into lower frequency. The basic specification is presented in table 4.4.

Figure 4.5: Rohde & Schwarz® SML01 signal generator

Physical Dimensions	426.7 mm (Width) X 87.6 mm(Height)X450 mm(Length) Weight: 8 kg
Minimum Frequency	9 kHz
Maximum Frequency	1.1 GHz
Frequency Resolution	0.1 Hz
External Frequency Reference	10 MHz

Minimum Output Power	-140 dBm
Maximum Output Power	13 dBm
Power Resolution	0.1 dB
Output Level Switching Time	10 ms
Maximum Single-Side-Band Noise	-122 dBc/Hz
Harmonics (noise)	-30 dBc
Modulation	AM, FM, Phase
Sweep Modes	External, Log, Manual
Safety Issue	meets DIN EN 61010-1, IEC 1010-1,UL 3111-1, CSA 22.2 No. 1010-1
Input Power	100 V to 120 V (AC), 50 Hz to 60 Hz,200 V to 240 V (AC), 50 Hz to 60 Hz, autoranging, max. 150 VA

Table 4.4: specification of Rohde & Schwarz® SML01 signal generator

4.1.5 Mixer

A mixer is a circuit device which accepts two diverse frequencies as its input and produces an output signal that mostly consists of the mixture of these two frequencies, principally the sum of the two input reference frequencies, and the offset (difference) frequency of the two input references. For example, with two input signals running 10 MHz and 11 MHz independently, the output of mixer will be frequencies which are the sum of two input signals, 21 MHz, and the difference between the two input signals, 1 MHz. The mixer used in this research is model ZP-5H+ manufactured by Mini-Circuits®. The circuit of ZP-5H+ mixer is designed as figure 4.7 and the detailed specification of this mixer is described in table 4.5.

Physical Dimensions	58 mm (Width) X 30 mm (Length) X 15 mm(Height) Weight: 75 g
Operating Temperature	-55° C to 100° C
Storage Temperature	-55° C to 100° C
RF power	200mW
IF current	40mA
Wideband	20MHz to 1500MHz
LO-RF isolation	50dB
LO-IF isolation	29dB
Conversion loss	7.5dB

Table 4.5: specification of Mini-Circuits® ZP-5H+ frequency mixer

Figure 4.6: Mini-Circuits® ZP-5H+ frequency mixer

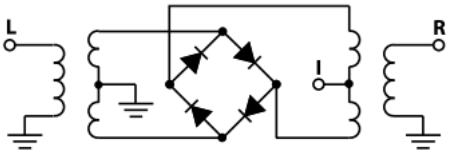

Figure 4.7: Electronic circuit of Mini-Circuits® ZP-5H+ frequency mixer

4.1.6 HP Oscilloscope

An oscilloscope is the instrument used to examine the electronic signals by means of a graphic display. The Oscilloscope 1740A manufactured by Hewlett-Packard® will be employed in this research to view the investigated signals. Also, it will be used for the general detection of drift over two references. The HP 1740A may be not a state of the art instrument; however, it could readily cope with our signal detection needs in this project. The overall specification of HP 1740A oscilloscope is listed in table 4.6.

Figure 4.8: Hewlett-Packard® 1740A oscilloscope

Physical dimensions	335 mm (width) X 197 mm (height) X 597 mm (length) Weight: 13 Kg
Operating Temperature	0℃ - 55℃
Relative humidity	95% at +40℃
Vertical display modes	Channel A, channel B, channel A and channel B displayed alternately on successive sweeps, channel A and channel B displayed by switching between channels at an approx. 250 kHz rate with blanking during switching, channel A plus channel B (algebraic addition), and trigger view
Horizontal display modes	Main, main intensified, mixed, delayed, mag X10, and input A vs. input B
Main time bases	50ns/div to 2s/div (24 ranges) in 1, 2,5 sequence
Delayed time bases	50ns/div to 20ms/div (18 ranges) in 1, 2, 5 sequence

Table 4.6: Specification of Hewlett-Packard® 1740A oscilloscope

4.1.7 Analogue-to-Digital Converter (ADC)

The function of an analogue-to-digital converter is to convert a sustained analogue input signal (normally a voltage or current) to discrete binary digital numbers according to the proportion of magnitude of input reading. The ADC used in this research is a high voltage reading model manufactured by Labjack® called U3HV. It will play the role of converting the analogue frequency signal in voltage to the digital numbers that can be utilised by the software developed in this project. The main features of Labjack® U3HV are summarised as below.

- USB interface
- First 4 inputs are high voltage inputs (±10 Volt or -10/+20 Volt Range), the rest 12 inputs are 12-bit flexible I/O (Digital Input, Digital Output, or analogue Input with volt range 0-2.4 V, 0-3.6 V, Single-ended or Differential)
- Supports 4 additional digital I/O
- Supports 2 Timers (timer clock frequency from 4MHz to 48MHz)
- Supports 2 counters (32-bits each)

- 2 Analogue Outputs (10-Bit, 0-5 volts)
- Supports SPI, I2C, and Asynchronous Serial Protocols

Figure 4.9: Labjack® U3HV Analogue to Digital Converter

4.1.8 Workstation

In addition to equipment used for hardware implementation in this research, a workstation is employed for the software implementation, especially for the software development part of system. In order to deal with massive data processing, an Intel ® Core™ 2 Duo based workstation is utilised in this research. Also, 4GB RAM was pre-installed on the workstation. The workstation are both running 32-bits Microsoft® Windows XP™ and 64-bits Microsoft® Windows Vista™ operating system for the need of system development. The basic specification for the workstation in use is presented as table 4.7.

Model	ASUS® U6SG
CPU	Intel® Core™ 2 Duo T9500 2.6GHz
Ram	4 GB
Hard disk	250 GB

Operating System	Microsoft® Windows XP™ 32 bit Microsoft® Windows Vista™ 64 bit

Table 4.7: Basic hardware specification of workstation in this research

4.2 System development process

4.2.1 Initialization

In the project planning phase, some key points must be pointed out for the following software development. In the process of developing the software outcomes will be examined iteratively by these key points. Basically, the key features of this software development can be classified into three areas; functionality, other requirements and operating environment.

Functionality

One major objective of this system is to calculate the Allan deviation between two references. So, it is expected that the software must have the ability to do Allan Variance / Deviation calculation. Another expectation after the Allan variance calculation is to plot the result out. Plotting Allan deviation is vital to the performance of this system; these plots are a very concise way of representing the accumulated data that readily allow the characteristics of a particular standard to be determined. Before starting planning the development for this software application, it is necessary to obtain an idea of what the system will look like. In order to gain some familiarity with the operation of the hardware in use and the meaning of signals that target frequency reference represents, we conducted a first experiment to using

DAQfactory software to acquire the data read by Labjack U3HV. DAQfactory is software included with the purchase of Labjack™ U3HV data acquisition device. It is developed by AzeoTech, one of the leading companies worldwide in supervisory control and data acquisition (SCADA) and human machine interface (HMI) software for industrial and scientific use. DAQfactory allows users to customise their data acquisition via a friendly graphical user interface (GUI), no advanced programming knowledge is required. For those who do already have a programming background, DAQfactory provides script functions for customisation. The initial experimental structure is described in figure 4.10.

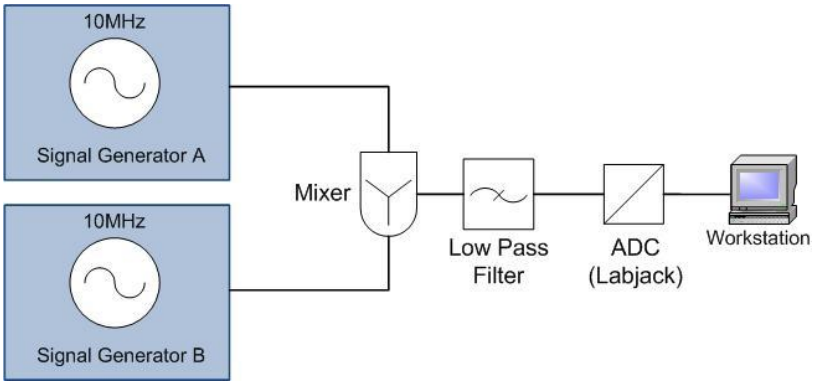

Figure 4.10: Experiment design for data acquirement over two references produced from signal generators internally

In this experiment, two signal generators will be used as statistically independent references. The two signal generators are both set to produce 10MHz at an output level of +5dBm, each references running independently from its own internal oscillator. In this case, the two references play the role of the atomic clocks that will be used in later experimentation. Even if the atomic clock reference standard was not used in this experiment, the setting of using two 10MHz internal reference from these signal generators is very similar to the later scenario with operating atomic clock standard references. That is because these atomic clock standard references also output 10MHz frequency. Before starting this experiment, we tried to see the frequency offset between the signals produced by the signal generators. The two 10MHz outputs from signal generators were connected to an oscilloscope as input A and input B. By dislaying input A versus input B, we could tell there was still drift between these two signals. Therefore, the adjustment on either A or B should be done. We adjusted the frequency of input A to 10000010 Hz to make the drift as small as we can tell by eye. After the adjustment, these two 10MHz outputs (*F1* and *F2*) from signal generators were connected to a mixer to produce signals with frequencies of (*F1* + *F2*) and (*F1* - *F2*). In order to only obtain the offset between these two references, a Low Pass filter is utilized to filter the value (*F1* + *F2*) out, and retain the value (*F1* - *F2*). In order to sample the low signal value from mixer, the sampling rate must be higher than the low value signal rate. In this case, the minimum sampling rate for the low signal value is 10 Hz (10000010Hz-10000000Hz). However, the faster sampling rate will lead to

better result. The signal output from the low pass filter is a voltage between +0.6V and -0.6V. After wiring the positive and negative cables into the Labjack U3HV input sockets, the reading from Labjack U3HV was plotted in the DAQfactory software.as shown in figure 4.11. The green line represents the historic reading of voltage.

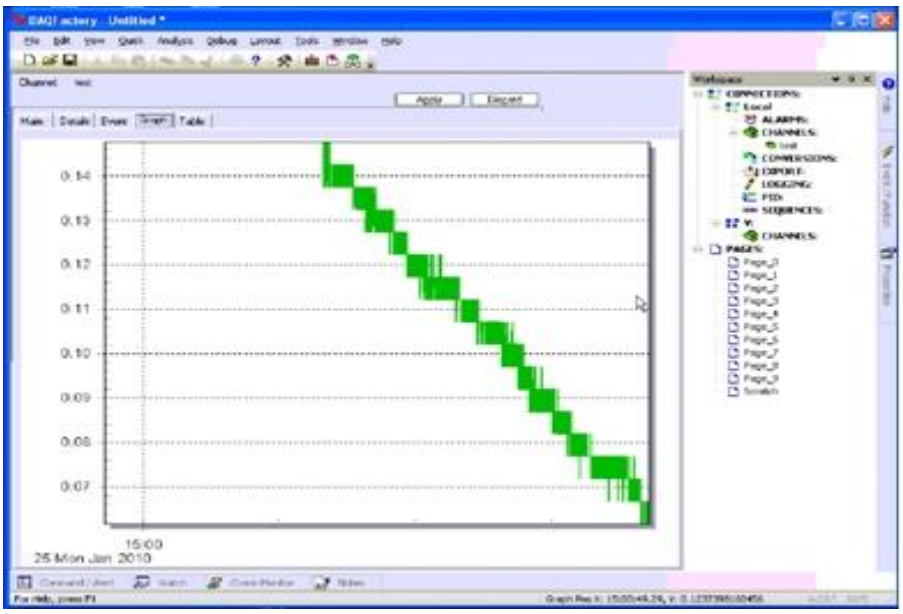

Figure 4.11: snapshot of DAQfactory software

However, there are some limitations of using DAQfactory to log the data converted from the analogue output signal of the mixer and filter combination. We attempted to record the data read by Labjack U3HV as fast as we could via DAQfactory. In our experiment, we found the DAQfactory software could reach 200Hz sampling rate without suffering any data loss but attempts to use higher sampling rates resulted in data loss. For instance, with 1 kHz sampling, we should gain 1000 samples per second. However, we only acquire around 166 samples per second as shown as table 4.8 below.

Time	Volt	
2009/9/10 23:19:39.996	0.435713	
2009/9/10 23:19:40.001	0.430658	
2009/9/10 23:19:40.009	0.430658	
2009/9/10 23:19:40.014	0.435713	
2009/9/10 23:19:40.018	0.435713	
2009/9/10 23:19:40.022	0.435713	
2009/9/10 23:19:40.037	0.435713	
2009/9/10 23:19:40.047	0.430658	
2009/9/10 23:19:40.052	0.435713	

166 Samples
/per second

2009/9/10 23:19:40.056	0.435713
2009/9/10 23:19:40.064	0.435713
2009/9/10 23:19:40.072	0.435713
2009/9/10 23:19:40.077	0.435713
… 135 samples abridged …	
2009/9/10 23:19:40.920	0.430658
2009/9/10 23:19:40.925	0.435713
2009/9/10 23:19:40.929	0.430658
2009/9/10 23:19:40.940	0.435713
2009/9/10 23:19:40.957	0.435713
2009/9/10 23:19:40.964	0.435713
2009/9/10 23:19:40.969	0.435713
2009/9/10 23:19:40.978	0.435713
2009/9/10 23:19:40.983	0.430658
2009/9/10 23:19:40.987	0.430658
2009/9/10 23:19:40.991	0.435713
2009/9/10 23:19:40.996	0.435713
2009/9/10 23:19:41.001	0.430658

Table 4.8: The data acquired from Labjack™ via DAQ factory software

It is believed that this slow operation could be caused by the delay in communication between the DAQFactory application and operating system. To prevent this problem the system we are going to develop will communicate with the Labjack directly via the Labjack driver since experiments have shown this has no such delay associated with it. The Labjack U3HV is capable of sampling rates up to 4kHz among all channels, we used 1kHz sampling rate as the basis for this particular test in order to potentially allow reliable exploration of tau below 1 second . Therefore, work to develop the ability to log streaming data at 1 KHz sampling rate was done and is reported on in section 4.2.2.2.

Up to current stage, we may conclude some functionality that the new system should provide as below.

- To calculate the AVAR
- To produce the AVAR plotting
- To acquire data at sampling rate 1000 Hz

Requirement

In the planning phase we must consider all requirements for this development. In addition to those stated immediately above there can also be requirements on the points of convenience, portability and ease of use. Firstly, we thought of that the system may need to be run on any computer at some situation. Therefore, the finalized production has to be an executable file or a self-installed package. Furthermore, it will be better if the application has an installer program for the targeted computer operating system. Next, the ease of use of the application is involved into our considerations. We may expect that outcome should have a graphic user interface (GUI) that will make for easier usage.

Operating environment

The operating environment means the background environment for the operation of the program. It is necessary to identify two factors, for example, the operating system which the program will be run upon and the programming language. In this case, due to the limited timeframe to finalize the project, Matlab was chosen to be the principle programming environment. Venkataraman (2009, p. 28) mentioned that Matlab® provides a friendly environment integrating computation, programming, and visualization into a mathematical expression of solving problems. Also, the Matlab® code is very compact and intuitive. Users could simply solve the problem by entering a few lines of code from the command line or making a Matlab® m file instead of complex and long coding from other programming languages. The applications of Matlab® can be easily seen in varied fields, for instance, math and computation, modeling, prototyping and simulation, scientific and engineering graphics, application and graphic user interface (GUI) development, data acquisition, analysis, exploration and visualisatiion, algorithm development, and so on.

Also, as the workstation in IRASR is already running windows operation system, we will expect our final program will be fully compatible with the Windows family of operating systems.

4.2.2 Developing

4.2.2.1 AVAR calculation

The first feature of this system we developed was the program to plot Allan variance. A system that can automatically gather the data and calculate the Allan variance after some parameters have been set was the goal. According to the AVAR formula (Allan, Ashby & Hodge, 1997), we began with establishing some Matlab® code for plotting Allan deviation as follows.

```
for i=1:MaxTau/tau0
    %Calculate the tau
    tau(i)=i*tau0;
%Calculate the value for each bin
    for j=0:n/i-1
        Bin(j+1)=(sum(y(i*j+1:(j+1)*i))/i);
    end
%Calculate the number of bins
m=length(Bin);
%Calculate every possible AVAR
AVAR(i)=(sqrt(0.5*(sum(diff(Bin).^2))/(m-1)))';
%Once one AVAR calculated, print one "@" mark
fprintf('@');
%Delete the Bin array
clear Bin;
end
```

(Note: tau0 is the sampling time interval in seconds; n is the number of samples)

Due to the absence of some equipment, a frequency counter for example, we experimentally attempted to input the voltage data generated from Labjack directly as the resource for the calculation of Allan deviation. We initially use the data produced from two signal generators (using their independent internal oscillators) with a 200Hz sampling rate. The Allan variance plot that resulted is as in figure 4.12.

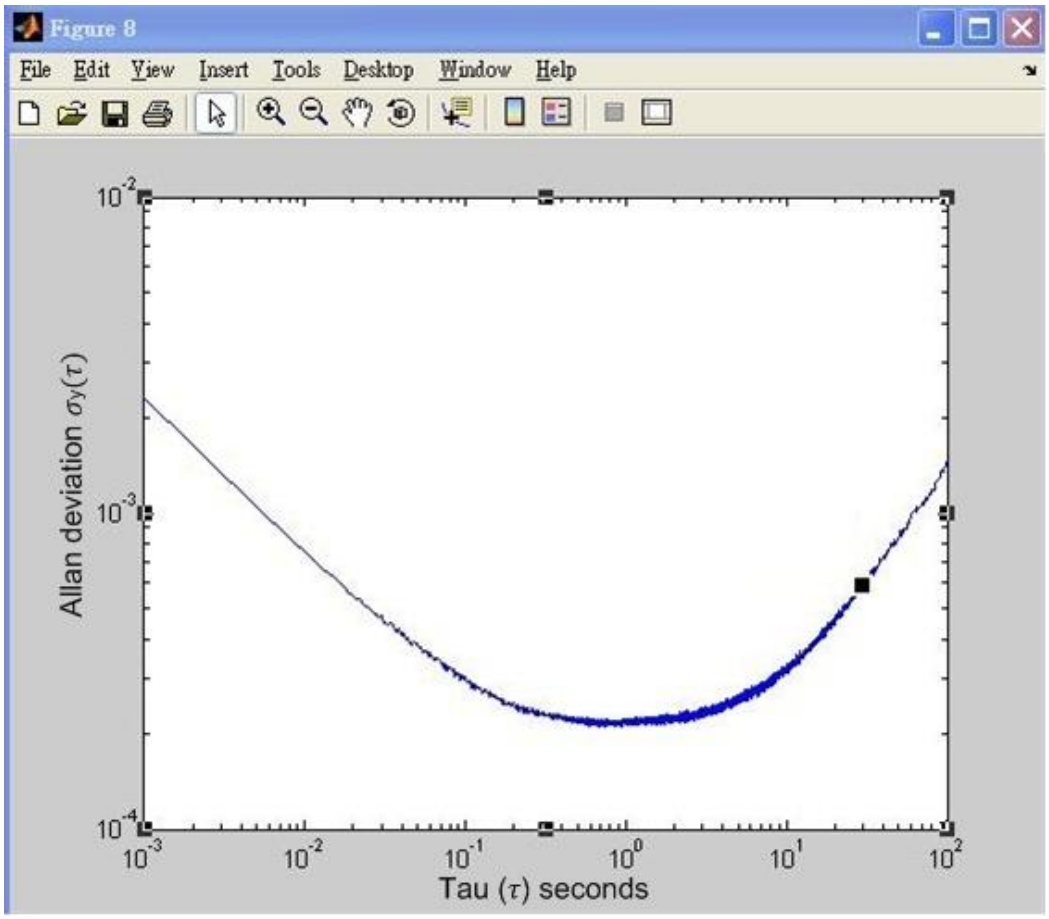

Figure 4.12: AVAR for data gathered from two 10 MHz references produced by signal generators at 200Hz sampling rate

On examination the AVAR plot appears somewhat odd in as much as that the y scale is not as low as the level we expected. Therefore, we reviewed the calculation in coding, and diagnosed that the average value of each bin should be normalized, that is divided by the frequency of reference, 10 MHz in line with the use of fractional frequency values in the calculation of AVAR . We replaced the appropriate line in the previous program with the following code.

```
Bin(j+1)=(sum(y((i*j)+1:(j+1)*i))/nt)/1e+7;
```

Due to previous strange result, we also suspected that it may be caused by an insufficiently fast enough sampling rate on the Labjack. As we could not complete the program with 1 KHz sampling rate at the moment, another experiment was designed to use both low frequency signals produced from signal generators running at 100Hz output for the Allan deviation calculation. The structure of this experiment is shown as figure 4.13.

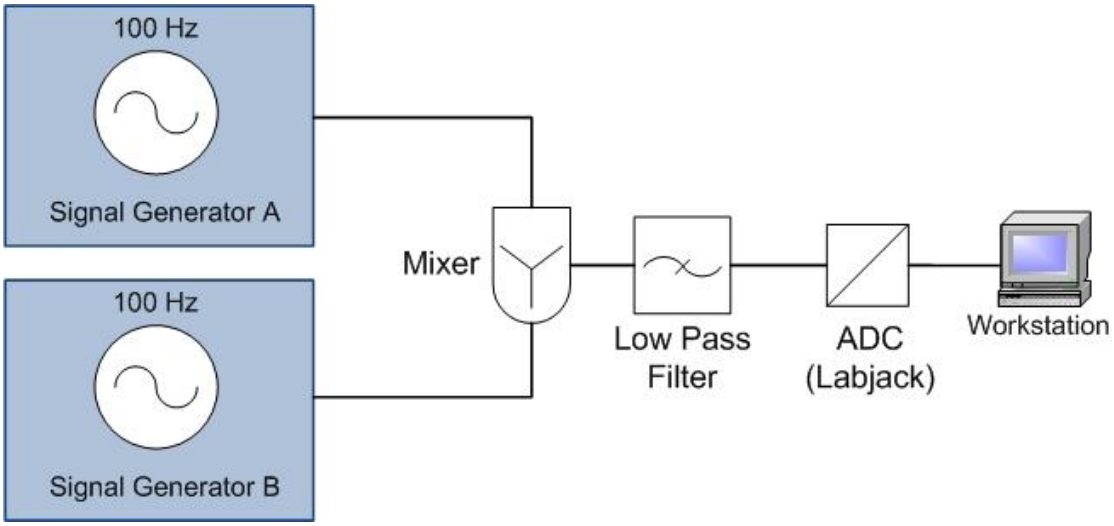

Figure 4.13: Experiment design for data acquisition from two 100 Hz low frequency references produced by signal generators

The two 100Hz signal outputs from both signal generators were connected to a mixer, and the Labjack was used to digitise the signal from the mixer at a 200Hz sampling rate. Unfortunately, this experiment failed because of the mixer could not work properly with 100 Hz inputs.

We then attempted to conduct another experiment to obtain the frequency offset in software by directly digitising the outputs of the two frequency references. We connected the two low frequency outputs directly into Labjack as shown on figure 4.14.

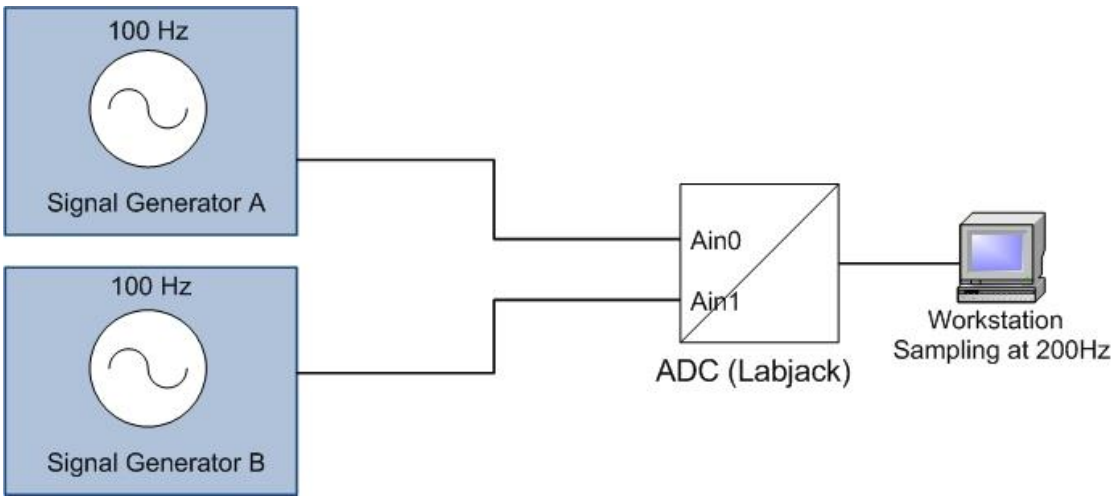

Figure 4.14: Experiment design for software-side data acquisition from two 100 Hz low frequency references produced by signal generators

Then, we calculated the difference between those two inputs read by Labjack and utilized the program to plot out the AVAR as figure 4.15.

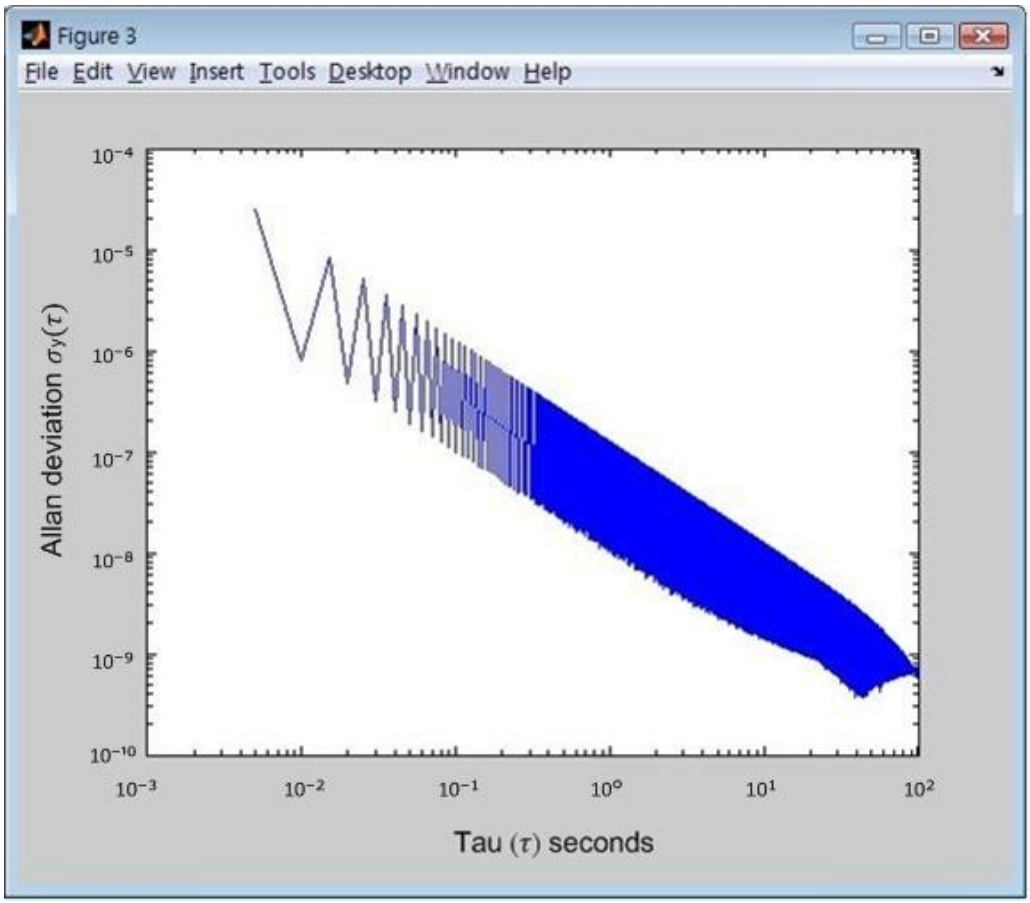

Figure 4.15: ADEV for data sampling on software side over two internal references produced by signal generators

As the result, we still could achieve not a satisfactory looking AVAR plot. Therefore, we decided to keep the original concept of using a physical mixer, and to accelerate the development on the 1KHZ streaming function.

In addition, it was realized that the data we used for AVAR calculation may be phase offsets instead of frequency offsets due to the way in which the mixer used operates. According to early experiment done by Tim Natusch at IRASR, the mixer ouputs a signal that is dependent on the phase difference of the two input signals. (See figure 4.16)

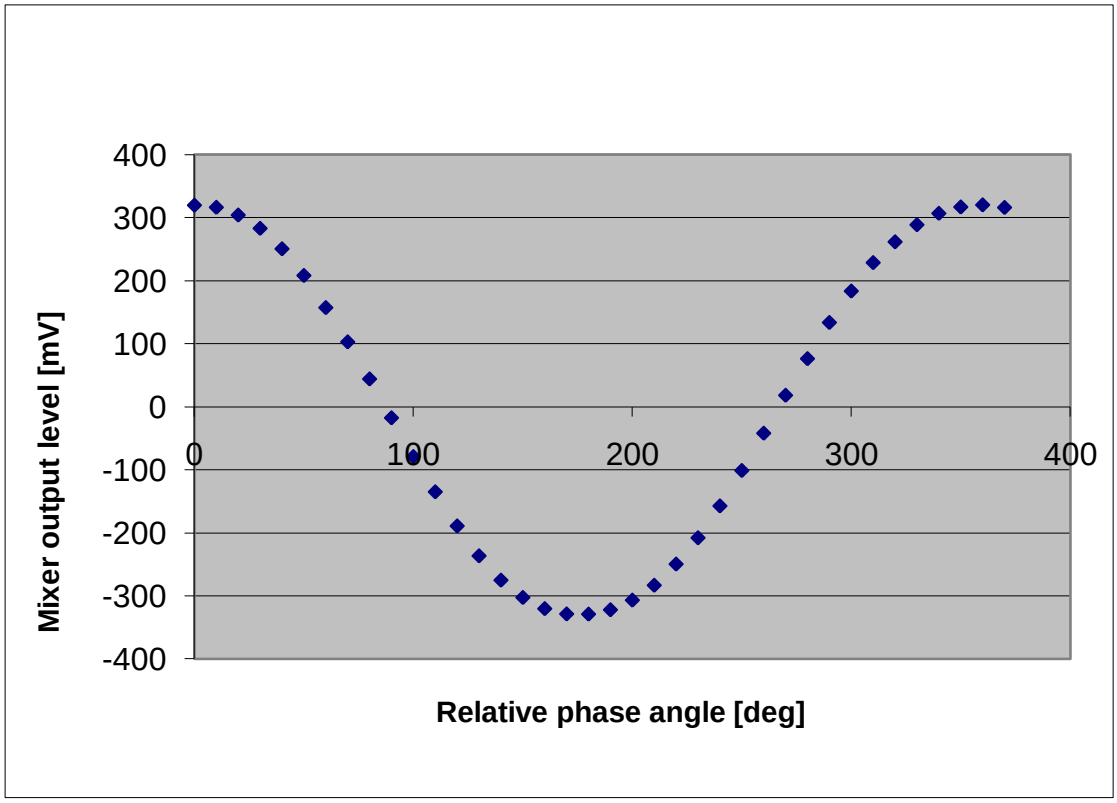

Figure 4.16: Mixer output versus input phase

So even if the two inputs are of the exact same frequency but are phase offset from each other. the mixer will produce a non zero output. According to the Allan deviation equation mentioned in chapter 2, $y(\tau)$ (frequency offset) could be formed as the value that $\Delta\emptyset$ (phase offset) divided by ΔT (time interval). Suppose we now have a raw phase offset data in voltage as figure 4.16. The $y(\tau)$ value will be $y(\tau)$ is equal to the differential of two raw samples, which is the tangent of curve between sample i and $i+1$. In other words, $y(\tau)$ could be rewritten as,

$y(\tau) = \frac{\emptyset_{i+1} - \emptyset_i}{T_{i+1} - T_i}$ (see figure 4.17)

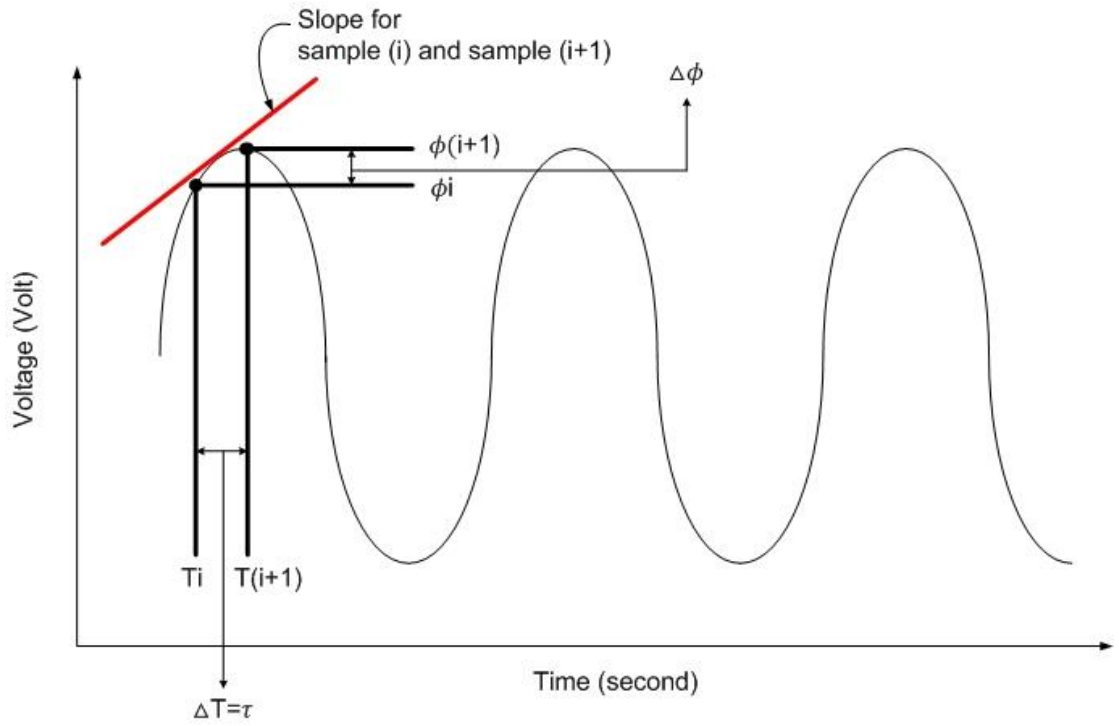

Figure 4.17: conversion from phase offset to frequency offset

Thus, additional code will have to be put in the program to perform frequency conversion for alternative calculation as following.

```
y=diff(y)/tau0;
```

With this addition the program for doing AVAR calculation is approximately done. However, as we are calculating all possible Allan deviation for every single bin, the execution time is much longer than expected. Therefore, it is necessary to go back to the principle of Allan deviation, calculating some significant Allan deviation for specific tau. Hence, we decided to calculate the AVAR for values of tau equal to the powers of 2. For instance, suppose the sampling interval is 0.001 second,

tau(0) will be $2^0 \times \text{tau} = 1 \times 0.001 = 0.001$,
tau(1) will be $2^1 \times \text{tau} = 2 \times 0.001 = 0.002$,
tau(2) will be $2^2 \times \text{tau} = 4 \times 0.001 = 0.004$,
and so on.

In order to achieve this, some codes in the AVAR calculation program were modified as follows;

```
maxi=floor(log2(n));
    for i=1:maxi
        nt=2^(i-1);
            for j=0:floor(n/nt)-1
                Bin(j+1)=(sum(y((i*j)+1:(j+1)*i))/nt)/1e+7;
            end
        m=length(Bin);
        %Calculate the AVAR for particular tau
        AVAR(i)=sqrt(0.5*(sum(diff(Bin).^2))/(m-1));
        %Calculate the tau
        tau(i)=nt*tau0;
        %Once one AVAR calculated, print one "$" mark
        fprintf('$');
        %Delete the Bin array
        clear Bin;
    end
```

(Note: maxi is the number of calculations of AVAR)

4.2.2.2 Streaming

Another major achievement of this program is to allow sampling of the data stream as fast as 1 KHz rate. To achieve this, we must make our Matlab® program communicate directly with the Labjack. As the Labjack driver is a common windows universal driver, in Matlab environment, we should create a prototype m file to perform as the header file LabjackUD.h in c:\program files\Labjack\Drivers folder. Also, codes in LabjackUd.h file should be modified as following to obtain required data in correct format.

```
LJ_ERROR _stdcall eGets(LJ_HANDLE handle, long IOType, long Channel, double *pVaule, double *Array);
```

By executing the following command, a prototype m file called loaddriver.m was established.

```
>>Loadlibrary('C:\windows\system32\labjackud.dll','C:\progra~1\Labjack\drivers\labjackud.h','mfilename','loaddriver.m');
```

Then, the function library will be loaded with the following code in stream.m.

```
Loadlibrary('C:\windows\system32\labjackud.dll', @loaddriver, 'alias','labjackud');
```

The streaming program worked well for short time period streaming. However, the buffer we configured had an overrun problem if we attempted long period streaming. We found it necessary to increase the buffer size to save the data acquired from the Labjack. The goal for total sampling time will be around 2 days to 3 days. By repeatedly trying, we finally set the value of buffer to 200 samples to avoid memory overruns.

Also, for the steaming program, we need to select a particular file that will be used to store the data. Due to the fact that the data read from Labjack is in the form of a floating point number and the resolution of Labjack U3HV is around 488 *m*Volt, we saved the data by using following codes.

```
fid=fopen(FileToRead,'a+');
fprintf(fid,'%1.4f\n',return_array);
fclose(fid);
```

Note that a .CSV file format is preferred for storing the data because it has better compatibility with other applications that may be used to further process the raw data, for example, the Microsoft Office family.

4.2.2.3 GUI

For the purpose of facilitating the use of this program, the idea of incorporating a GUI was proposed. A GUI in Matlab® consists of two components for its operation, the .m file and .fig file. In this case, the design structure is displayed as follows. An interface is created as the home interface when executing the program. On this interface, users will have two options, plotting AVAR from existing data, or start streaming new data and plot the result by using the data loaded from Labjack. An interface for plotting the AVAR is designed as in figure 4.18.

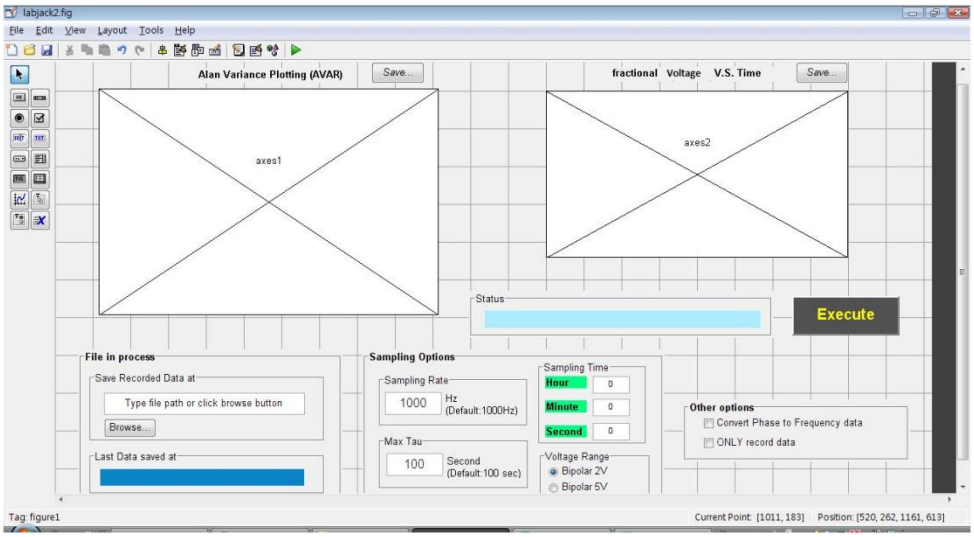

Figure 4.18: GUI design for plotting AVAR from existing data

The user will need to load any existing data file and select appropriate variables for AVAR plotting to proceed. In addition, functionality is included that allows the user to command the Labjack to begin sampling a new data stream. The sampling rate is set to 1 KHz by default. User can also decide if they need to plot the result out or just do the streaming job. The GUI of streaming function is designed as figure 4.19.

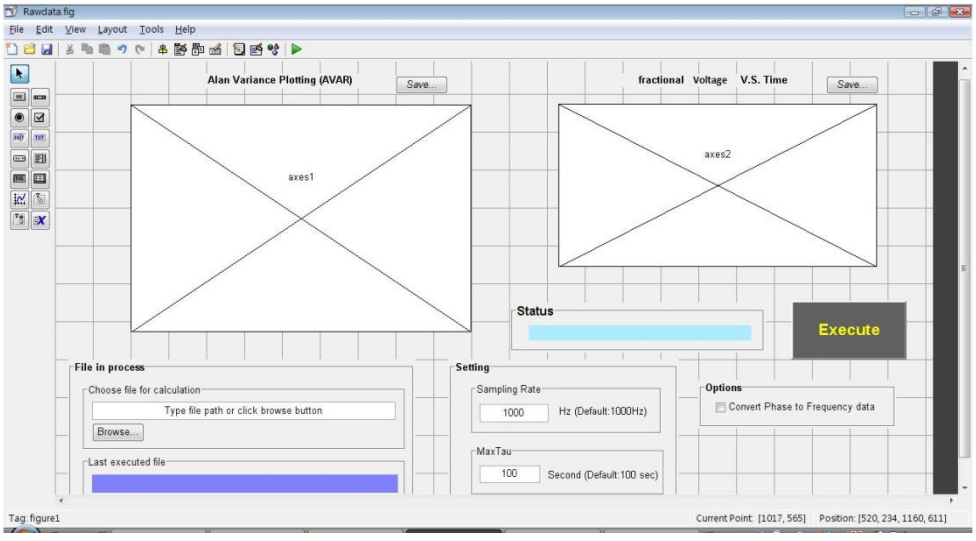

Figure 4.19: GUI design for data streaming and automatically AVAR plotting after the streaming

Both of these GUI will allow users to save the plotting result as jpg file and support a printing function.

4.2.2.4 Additional development

- **Frequency variation plot**

In addition to the AVAR plot, it is often useful to be able to see the recorded signal in the time domain. This will quickly reveal the manner in which the frequency of a source under test varies with respect to the reference standard. In the beginning, we attempted to plot out all recorded raw data. There are several drawbacks to doing this. Firstly, it is difficult to see the frequency variation of raw data if the total data recorded is displayed in one plot. (See figure 4.20). Secondly, it takes too long to plot all raw data due to the same reason as above, the enormous data volume. This massive amount of data also contributed to the slow operation of the entire system. Hence, a decision was made to merely plot the frequency variation by fragmentary row data. A variable called Fre was created to store 500 samples of data to plot the frequency variation in axes 2 of GUI. (See figure 4.21) The coding for this is as follows;

```
Fre=y(5000:5500);
plot(handles.axes2,Fre);
xlabel(handles.axes2,'time', 'fontsize',8,'fontweight','b');
ylabel(handles.axes2,'Voltage', 'fontsize',8,'fontweight','b');
```

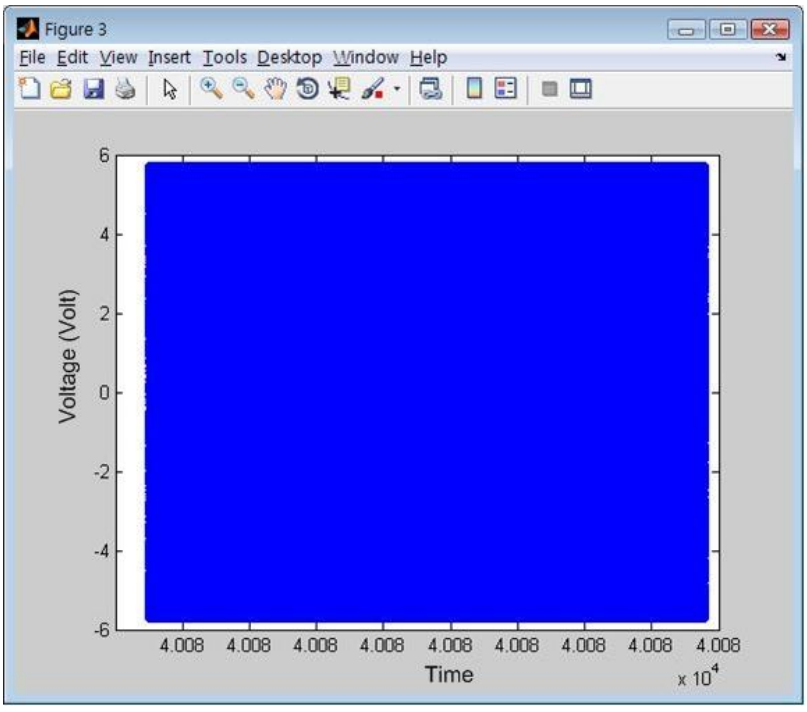

Figure 4.20: ambiguous voltage variation plotting

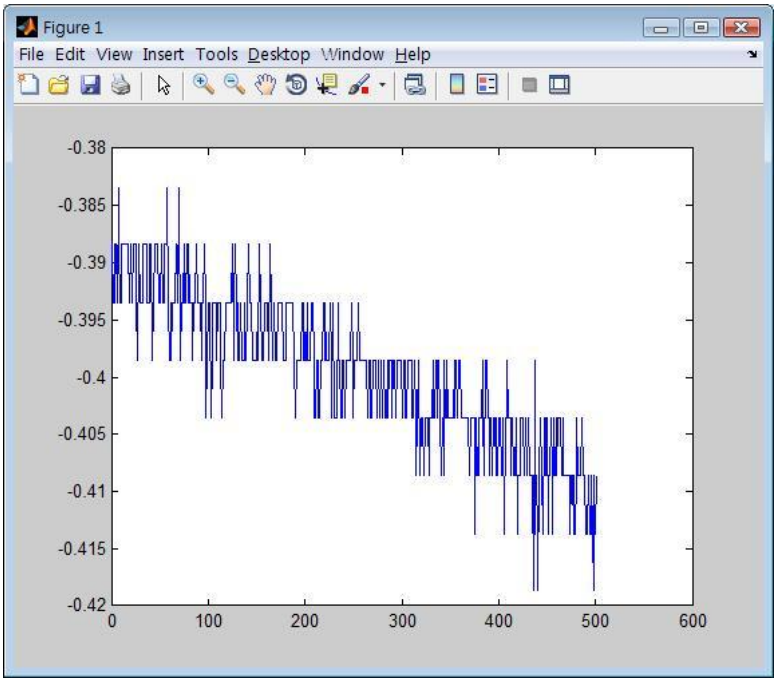

Figure 4.21: more clear voltage variation plotting with 500 data samples

- **Overlap Allan deviation, Modified Allan deviation and Allan deviation for all tau values**

After the successful implementation of Allan variance plotting, we planned to include some extra related plots, such as overlap Allan deviation, modified Allan deviation, and the Allan deviation for all tau values done before. A radio button box will be placed on the GUI for the selection of a particular plot type. (See figure 4.22)

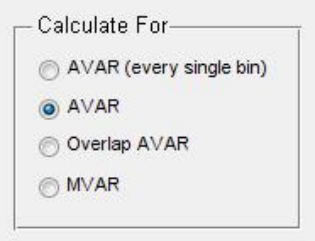

Figure 4.22: Radio button box group for different Allan deviation plotting

Each plot will be produced by running an individual program. Actually, the only code difference is in the calculation of deviation for the programs plotting different Allan deviation types.

Firstly, the modified calculation codes for plotting overlap Allan deviation is shown as follows.

```
for i=0:maxi-1
        nt=2^i;
        Bin=single(zeros(1,n-nt+1)');
        for j=1:n-nt+1
            Bin(j)=sum(y(j:j+nt-1))/nt;
        end
        %create the sequence for overlap Bin
        S1=Bin(nt+1:n+1-nt);
        S2=Bin(1:n+1-2*nt);
        u=sum((S1-S2).^2);
        e=length(S1);
        %calculate the Overlap AVAR
        AVAR(i+1)=sqrt(0.5*u/((n+1-2)*nt.^2));
        %Calculate the tau
        tau(i+1)=i*tau0;
        %Once one AVAR calculated, print one "$" mark
        fprintf('$');
        %Delete the Bin array
        clear Bin S1 S2;
end
```

Secondly, the calculation code for plotting Modified Allan deviation is written as following.

```
for i=0:maxi-1
        nt=2^i;
        Bin=single(zeros(1,n-nt+1)');
        for j=1:n-nt+1
            Bin(j)=sum(y(j:j+nt-1))/nt;
        end
        %MVAR
        T=single(zeros(1,n+2-3*nt)');
        for k=0:n+1-3*nt
            S1=Bin(1+L:nt+k);
            S2=Bin(1+nt+k:2*nt+k);
            T(k+1)=(sum(S2-S1))^2;
            clear S1 S2;
        end
        e=length(T);
        T=mean(T);
        AVAR(i+1)=sqrt(T/2)/nt;

        %Once one AVAR calculated, print one "$" mark
        fprintf('$');
        %Delete the Bin array
        clear Bin T;
end
```

Thirdly, those code previously described is retained to calculate Allan deviation for all possible tau values.

These four programs calculating four types of Allan deviation will be named independently to respond to the call back created in the GUI.

- **Error bar**

Error bars are used in statistical graphs to give an idea of the truthfulness for the plotting result. It will use the value of Allan deviation divided by the number of bins for each plotted value of tau. The Matlab® code to add this feature is as follows.

```
%calculate the Errorbar for AVAR, m is number of bins
erb(j)=AVAR(j)/sqrt(m);
```

- **Waiting bar**

Because the execution time for this program is normally long, it will be useful if the system has a waiting bar to inform users of progress during computation. Thus, the following codes are attached to display the percentage of the job done so far.

```
per=fix(i/maxi*100);
close(h);
waitbarstr=['Done ', num2str(per),'%'];
h=waitbar(i/maxi,waitbarstr);
pause(0.5);
```

4.2.2.5 Optimization

Towards the end of the development optimization is required to increase the system performance. Some key optimizations to coding were applied and are described below.

- **Pre-allocating arrays**

According to McGarrity (2007), code that used pre-allocation for arrays executed 580 times and 475 times faster than codes without pre-allocation on experimental machine A and B respectively. This situation will be especially obvious for large data volumes. In this work the data arrays required change size for each different value of tau, and if not preallocated memory space, would cause Matlab® to repeatedly search the available memory for space for each new value of tau In our case, the size of data required to be loaded in the system is very big, around 900 MB to 1G for 36hours of data recording. The demand for pre-allocating arrays is imperative in our case. Thus, the pre-allocation of arrays, such as tau, AVAR, and erb (errorbar) are implemented as per the following code.

```
maxi=floor(log2(n));
      erb= zeros(1,maxi);
      tau= zeros(1,maxi);
      AVAR= zeros(1,maxi);
       Bin= zeros(1,n-nt+1);
```

- **Store and Access Data in Columns**

McGarrity (2007) determined that storing data in columns could help reduce the execution time compared to storing data in rows. His survey shows storing data in columns consumes 33% and 55% less time than storing in rows on experimental machine A and B respectively. Hence, the code to store data in array will be re-written to:

```
maxi=floor(log2(n));
      erb= zeros(1,maxi)';
      tau= zeros(1,maxi)';
      AVAR= zeros(1,maxi)';
       Bin=zeros(1,n-nt+1)';
```

All arrays in the program were examined and modified by following this rule.

- **Avoid Creating Unnecessary Variables**

Any unnecessary variables created will both waste the storage space in memory and slow down the system. McGarrity (2007) indicated that calling operators in-place will enhance the execution time up to 40%. The idea of calling operators in-place is to replace the original value of existing variable by calculated result instead of creating a new variable containing the calculated results. As our system will process a very large volume of data, there is no point duplicating two variables containing similar data. For example, the newly converted frequency difference array will replace the original phase difference array as the input data for later calculation.

```
y= differ(y)/tau0;
```

Here, we replaced the original y array instead of creating a new array.

- **Reduction of size of data storage**

Because our program will process a large data volume, optimization is indispensable to reduce the amount of memory in use. In terms of recorded data, the raw data is a voltage reading with 4 decimal places between +0.8 and -0.8. We changed the data structure from double-precision array to single-precision array. For a double-precision array in Matlab® the system would require 8 bytes per element. If we load 32 hours data with 1 KHz sampling rate into Matlab®, 921600000 bytes (approximately equal to 900 Mb) of memory size will be needed for data storage. However, loading the data in a single-precision array format will only consume half the size of memory compared with using double-precision array format.

```
maxi=floor(log2(n));
        Bin=single(zeros(1,n-nt+1)');
        erb= single(zeros(1,maxi)');
        tau= single(zeros(1,maxi)');
        AVAR= single(zeros(1,maxi)');
```

4.2.2.6 Compiling

In order to make the program files created capable of execution outside the Matlab® environment, compilation is necessary. After finishing the program development, we will have several Matlab® script files (M file). By executing the following commands in Matlab® an executable file (.exe) will be made that compiles all m files and the GUI figure files (.fig).

```
>>mcc –m start.m –a ./*.m
```

After the compilation, seven files will be produced, start.exe, start_mcc_component_data.c, start.prj, start_main.c, start_delay_load.c, readme.txt, and mccExcludedFile.log. In term of these files, the executable file can then be run independently on another machine.

4.2.2.7 Packaging

Finally, we wish that the system can be installed on any Windows platform computer. Packaging software called Smart Install Maker was used to make a self-installation program. If we want to run the compiled executable file on another computer, the Matlab® function library will be required. For our development environment, Matlab® 2009a, Matlab® provides a self-installed executable file called MCRInstaller.exe which helps the user to install the Matlab® function library on the computer without the Matlab® environment. Hence, we include this file into the package and make it execute before copying all necessary files on the target computer.

Figure 4.23, 4.24, 4.25 are the snapshots of the packaging process.

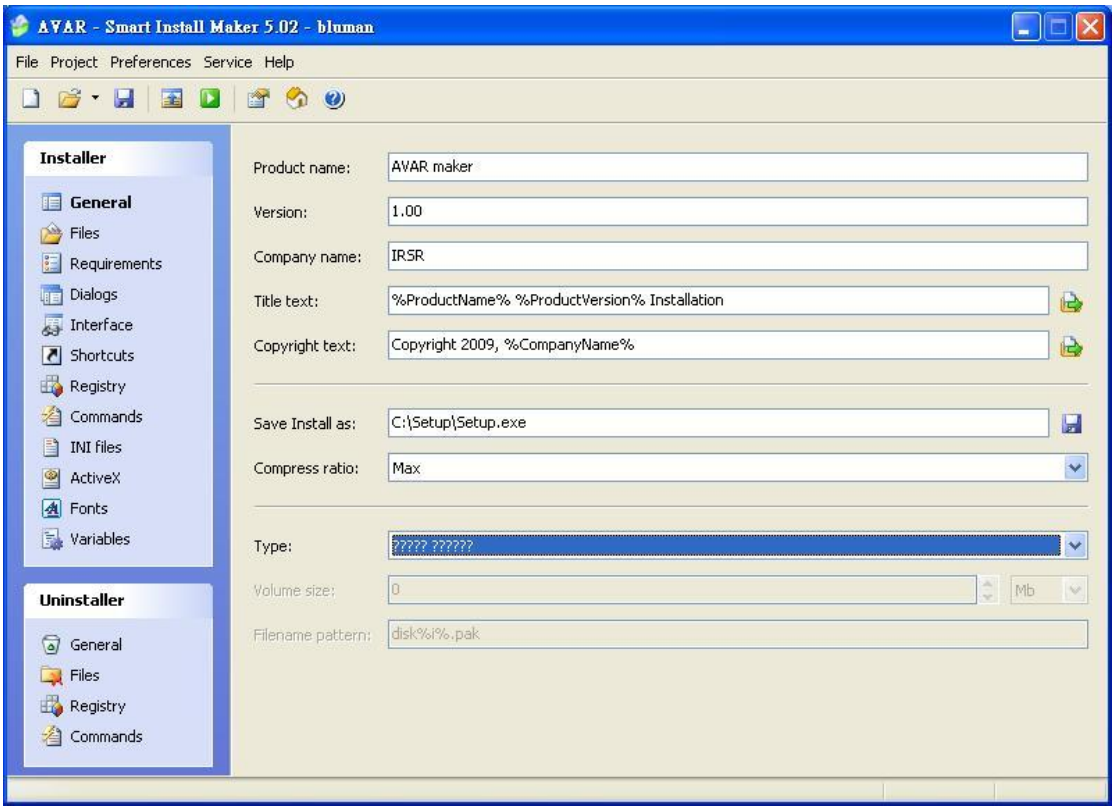

Figure 4.23: Snapshot of executing the Smart Install maker software

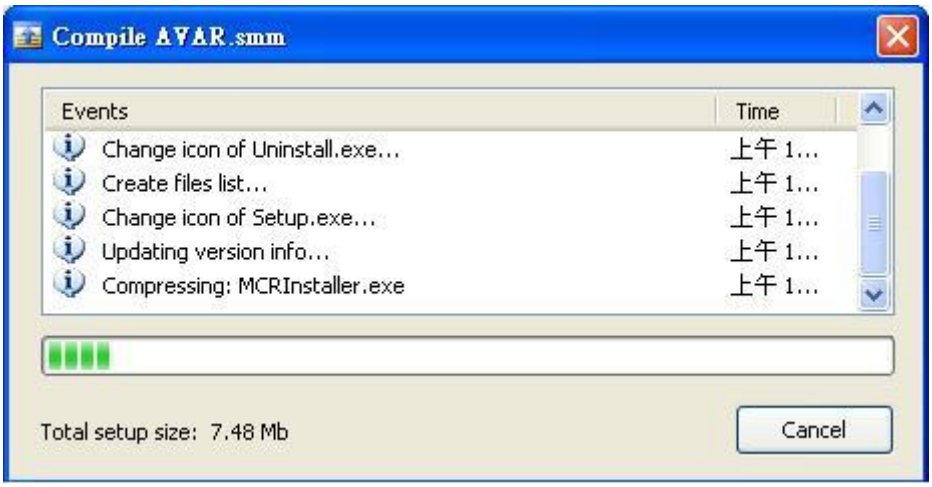

Figure 4.24: Snapshot of packaging files needed for installation

Figure 4.25: Snapshot for IRSR AVAR maker installation

4.2.3 Testing

After the finalization of system development, practical testing will be helpful to check if the system works to meet our expectation. Once the system is installed and executed in the machine, users will see the initial menu for the options of execution. These options are to execute the system with the existing samples (recorded) data or to acquire and calculate new data by using the Labjack. The initial menu snapshot is displayed as figure 4.26.

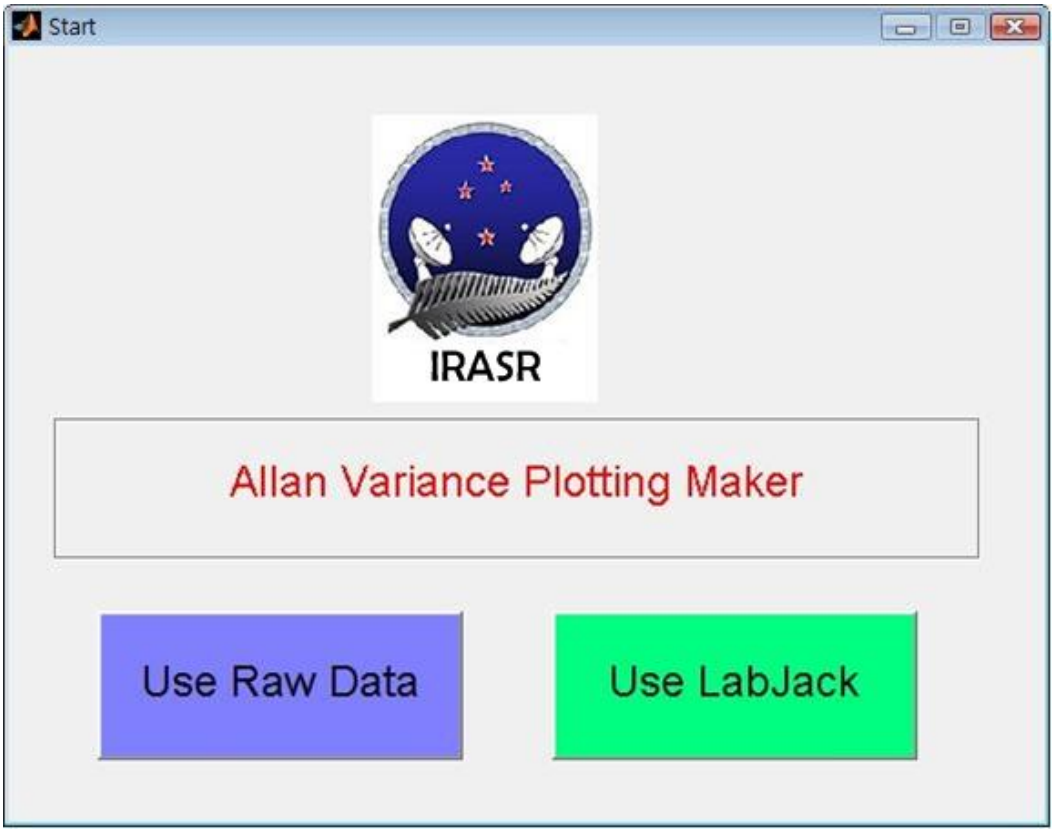

Figure 4.26: Home menu for the system

If users wish to calculate the AVAR from existing data file, the system will go to the interface as in figure 4.27.

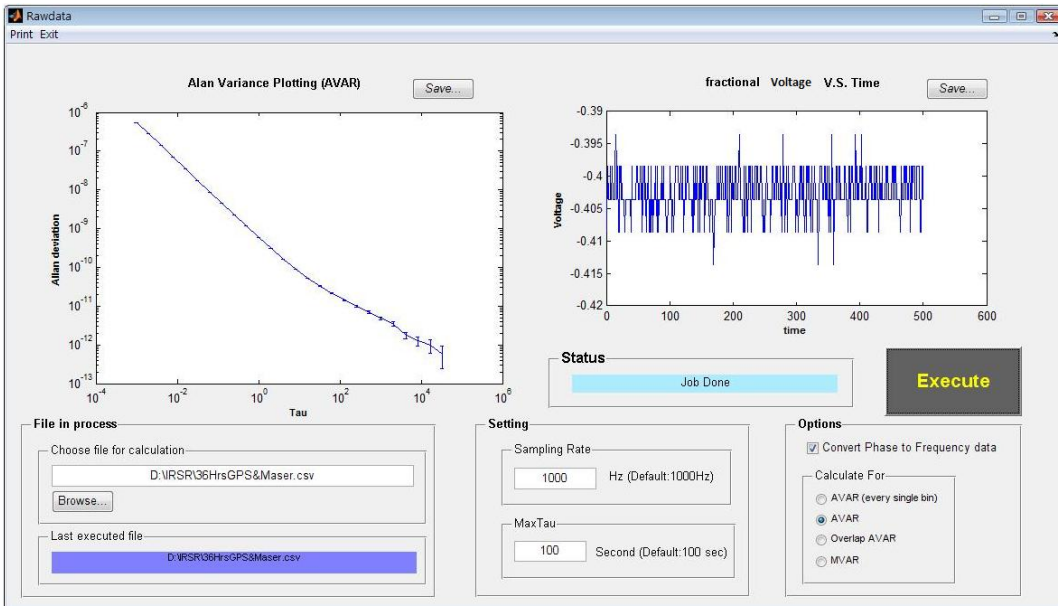

Figure 4.27: Example of using existing data for ADEV plotting

If users will acquire new data from the Labjack and calculate new AVAR values the system will go to the data acquisition page as figure 4.28.

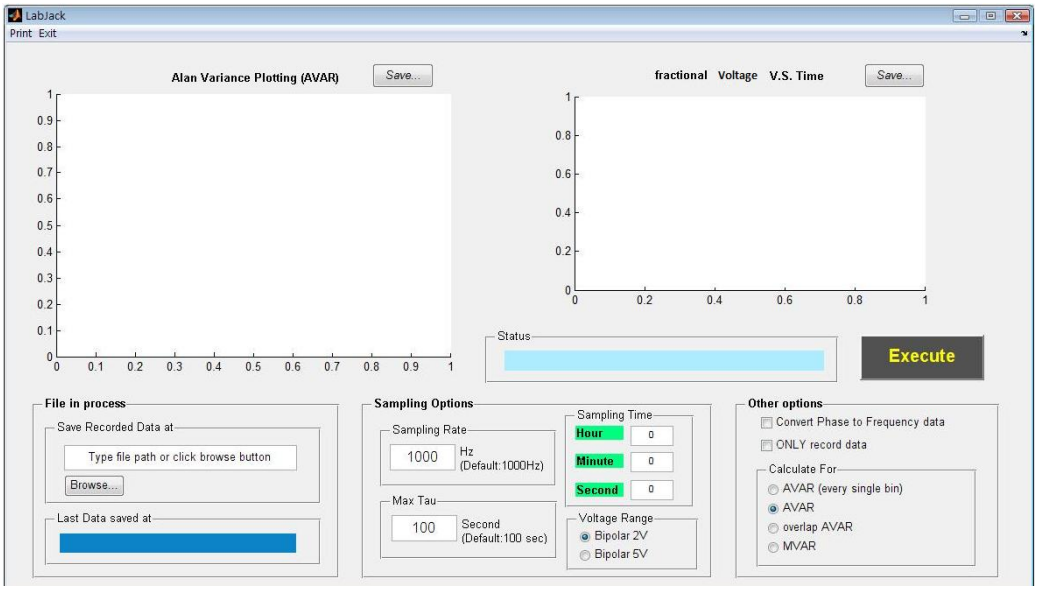

Figure 4.28: The streaming setting page of system

In this testing phase, firstly, we used this system to calculate the Allan deviation for a GPS reference against Hydrogen Maser. 17 hours, 36 hours and 72 hours recording were done separately. Then, Allan deviation with Rubidium atomic clock versus Hydrogen Maser was calculated. However, the AVAR between Rubidium atomic clock and Hydrogen Maser gave strange results. It was suspected that the Rubidium atomic standard may not be performing correctly as calibration had not been done for a long period of time. The Rubidium atomic clock used in the project is able to be calibrated by input of a 1PPS signal from the GPS.

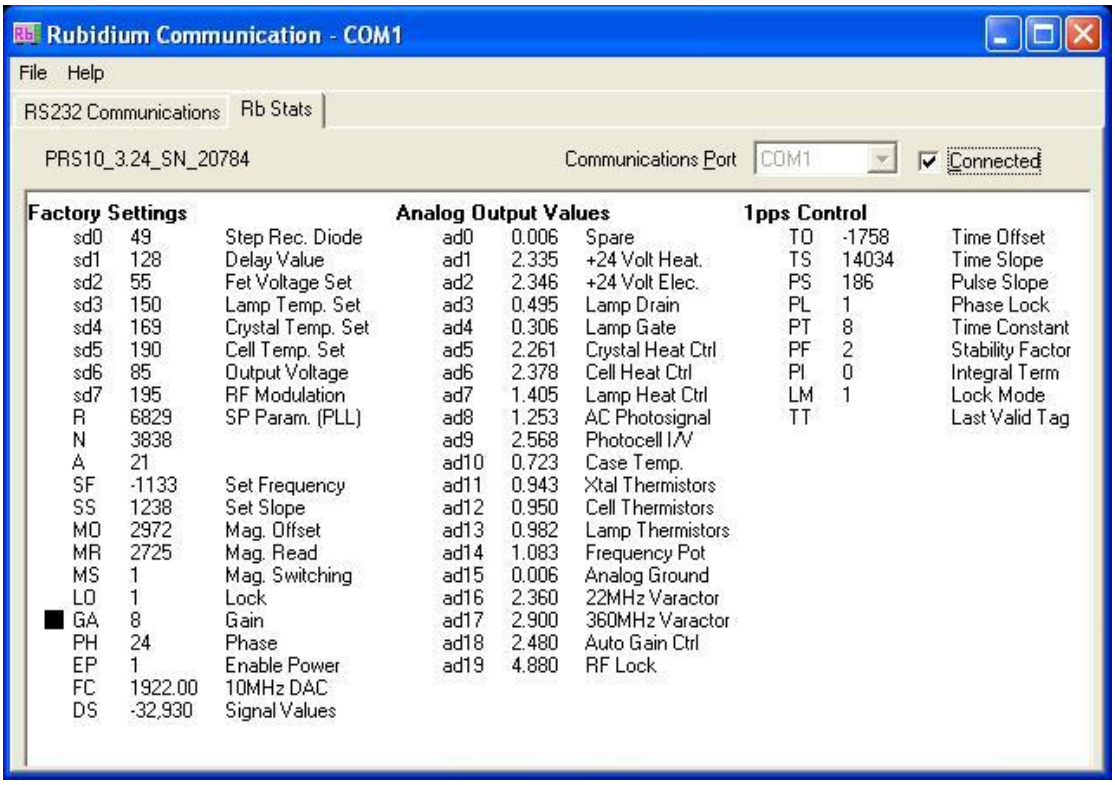

Figure 4.29: Software for monitoring Stanford Research Systems® FS725 Rubidium atomic clock

By using the monitor software of Rubidium atomic clock (see figure 4.29), we could adjust the frequency calibrated with GPS signal. After the correction, the 3 different duration recordings noted above were done and AVAR calculated for each.

In summary, the system testing gave satisfactory and reasonable performance. However, it should be noted that some confusing results for AVAR were seen. A more detailed discussion of these issues will be made in following chapter.

Chapter 5 Discussion and further work

5.1 The procedure of software development

In this research, we employed the rapid and incremental software development method because of limited time and cost considerations. After initial experiments, three main functionalities were decided for the system development, the calculation of AVAR, plotting of AVAR and streaming with appropriate rates. After those three main functionalities were finalized, some extra functions were added to improve the operation of system.

Regrettably there has not been enough time to write all tuition, maintenance and user documents that would normally accompany a completed software product.. Fortunately, the interface of this system is not difficult to become familiar with and the intended users are knowledgeable with regard to the application and therefore it is felt that these documents are not absolutely crucial. However, it is still expected that this documentation task could be done in any future development stages.

5.2 Comparison AVAR within Rubidium atomic clock, Hydrogen Maser and GPS

By using the system we developed, we finally come up with the results of the Allan variance comparisons of Rubidium atomic clock, Hydrogen Maser and GPS as shown as below, figure 5.1, 5.2, 5.3.

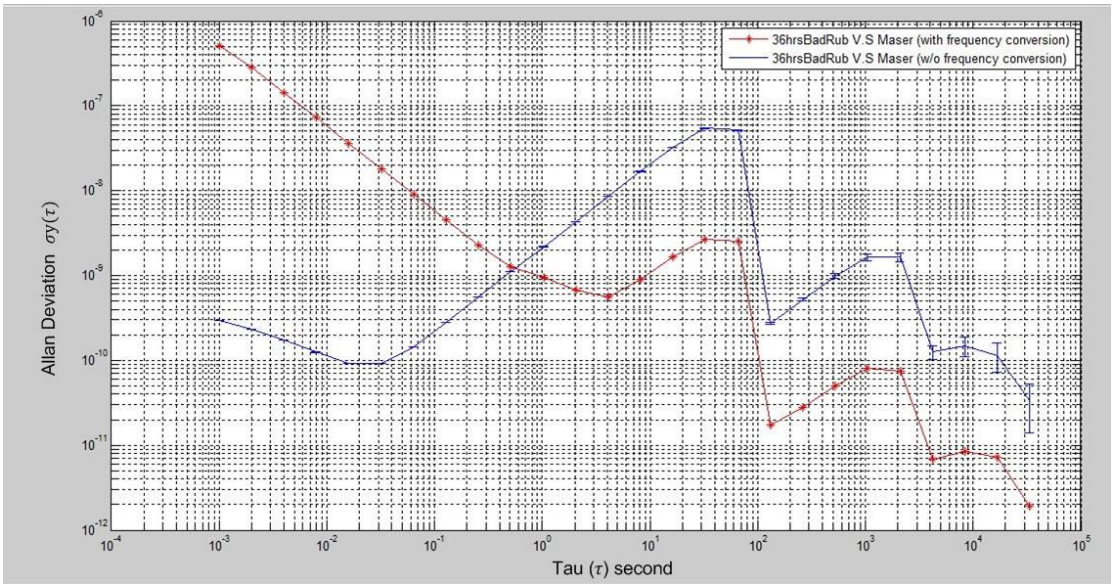

Figure 5.1 : AVAR plotting for 36hours recording of Rubidium atomic clock (without correction) versus Hydrogen Maser

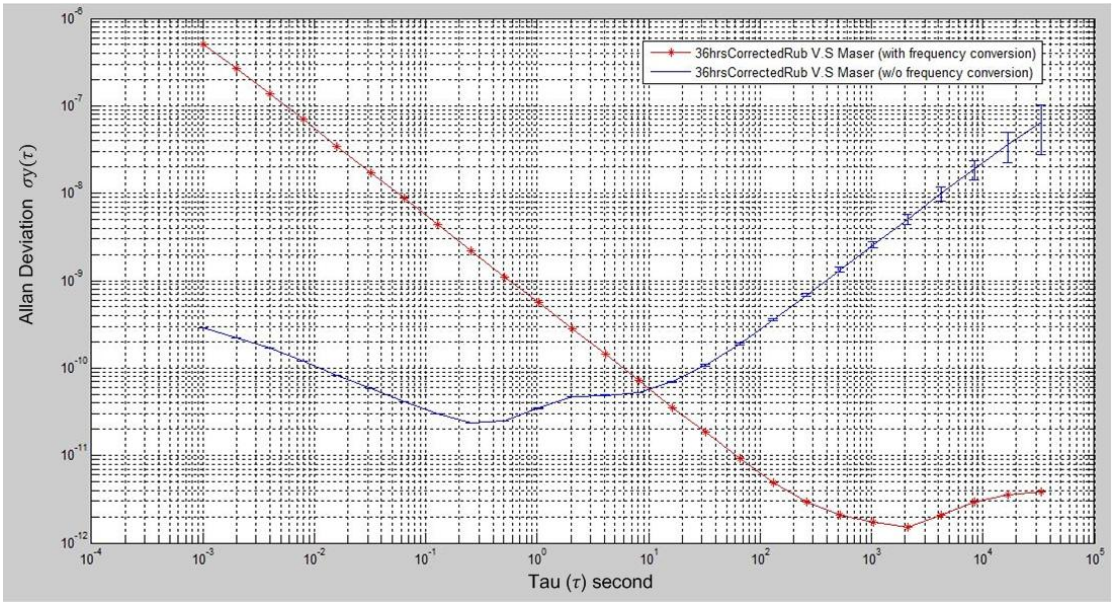

Figure 5.2: AVAR plotting for 36hours recording of Rubidium atomic clock (with correction) versus Hydrogen Maser

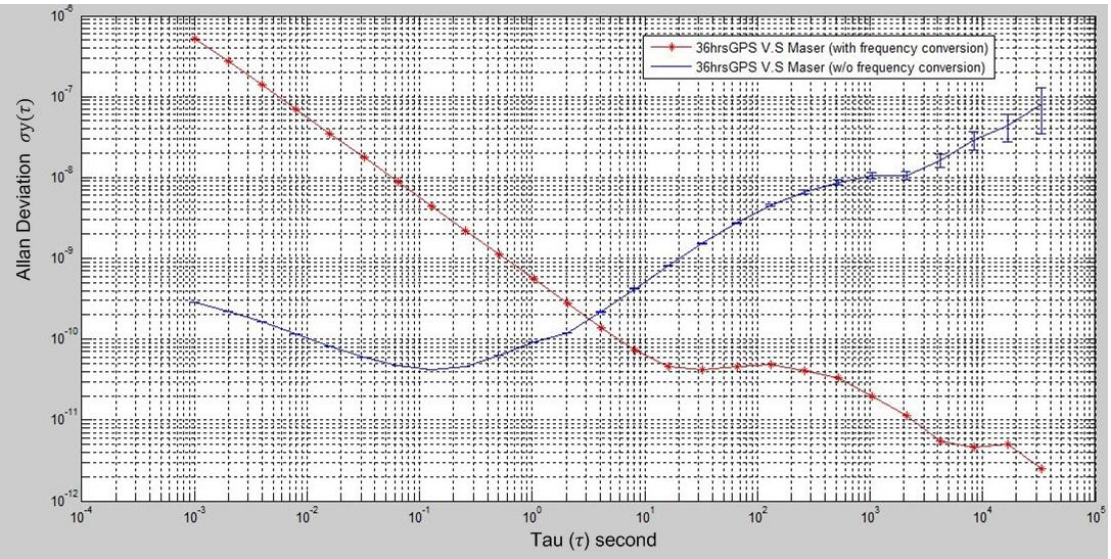

Figure 5.3: AVAR plotting for 36hours recording of GPS disciplined clock versus Hydrogen Maser

Because of the absence of a frequency counter it was not possible to do a direct comparison and there is still some confusion if the frequency conversion should be used or not. With frequency conversion applied, we could see the Allan deviation of the corrected Rubidium atomic clock against Hydrogen Maser reaches a floor of 10^{-12} at tau 1000 seconds. It was expected that it should have reached this floor at an earlier tau. However, the slope of this Allan deviation plot, around -1 for down going trend and +0.5 for up going trend are exactly as expected (Nunzi, Galleani, Tavella & Carbone, 2007). Without applying the frequency conversion, the Allan deviation between corrected Rubidium atomic clock and Hydrogen Maser reaches the bottom earlier between tau 0.1 second and 1 second, but has a higher level of minimum Allan deviation at 10^{-11}. Whilst this level is about what would be expected from the Rubidium manufacturers specifications, the slopes of the AVAR plot without doing the frequency conversion are not as expected at -1 for down trend and +1 for up trend.
In addition the results of non-corrected Rubidium atomic clock against Hydrogen Maser, both behave oddly at the end of curve whether the frequency conversion is applied or not. Both of them decrease with unexpected large scale fluctuations.

For the AVAR results of GPS disciplined clock against Hydrogen Maser, we could see the result is significantly different if frequency conversion is applied. For the AVAR curve without frequency conversion applied, it begins between 10^{-10} and 10^{-9} at tau 0.001 second whereas the AVAR curve with frequency conversion applied begins between 10^{-6} and 10^{-7}. The AVAR curve without frequency conversion applied is also represented as an initially downward-fluctuating signal followed by an-up trend which reaches the bottom level of 3e-10 at 0.1 second. However, with frequency conversion applied, the AVAR curve is typically a

down-stream which has two obvious break points; between 10^{-10} and 10^{-11} when tau is about 20 to 131 seconds, and between 10^{-11} and 10^{-12} when tau is around 4194 and 16780 seconds. It is possible that a contributing factor here is that the signal produced by the Caesium atomic clock onboard GPS satellites are regularly monitored by the GPS ground base station and calibration adjustment data is uploaded to the satellites regularly. (King, 2003) Thus, the expected random walk trend does not appear in this plot, it is in place of the AVAR curve which the AVAR is constantly revised to a smaller value while tau value increases.

Because of the odd performance of Allan deviation for these precise atomic standards, it will be sensible in the future to implement the experiment using a frequency counter. This would allow the frequency conversion involved for calculations to be bypassed and then it would be possible to determine the creditability of the results we observed in this research.

5.3 Limitation

1. Sampling resolution on ADC

In this project, a low cost ADC, the Labjack, was used to record the signals produced from highly precise references. According to the Labjack Corporation (2009), U3HV is a 12 bit analogue to digital convertor with a ±10 V range. The resolution is thus only about 0.00244140625 Volt ($20\,\mathrm{V}/2^{12}$) which means that only 3 decimal places are valid. One solution to tackle the resolution problem would be to make use of another instrument with higher resolution. However, the resolution is directly proportional to the cost of instrument, which means we may need to purchase more expensive instrument to enlarge the resolution of samples. Another way to solve this problem with original equipment setup is to set up the range of reading so that it corresponds more closely with the voltage output range from the mixer. For example, if the voltage output from mixer is between +0.8 and -0.8 volt, the reading voltage range could be configured as 2 volt in total range. As we discussed before, in this case using the Labjack, the resolution of a 2 volt range would be improved to around 0.488 mV ($2\,\mathrm{V}/2^{12} = 0.00048828\,\mathrm{V}$) from 4.88mV with 20 volt reading range (assuming 12 bits resolution as before). That would result in the valid number being enhanced from third decimal place to fourth decimal place. Last, a physical amplifier could also be used to increase the resolution of reading. The structure of configuration is presented as figure 5.4.

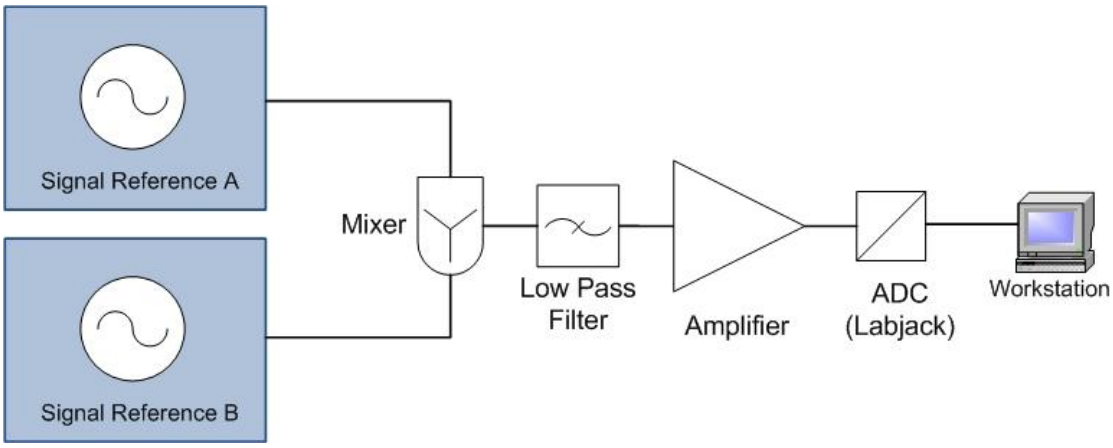

Figure 5.4: Experiment design of using amplifier to enlarge the signal

By using the amplifier, one more decimal place space will be allowed, for example, reading 0.234 Volt will become 2.345 Volt. Ideally, deploying both reading range adjustment and a physical amplifier will allow the resolution of data reading down to the fifth decimal place. Ideally, applying all of the solutions above will provide the best resolution for the sampling data.

2. Absence of frequency counter

In our experiments, we attempted to convert phase offset as measured by the output voltage of the mixer. It is feasible, but may be not the best solution for the issue. The best idea for doing the conversion would be to use a frequency counter. ROHDE & SCHWARZ Corporation (2009) explain that the experiment structure involving a frequency counter will be similar to figure 5.5.

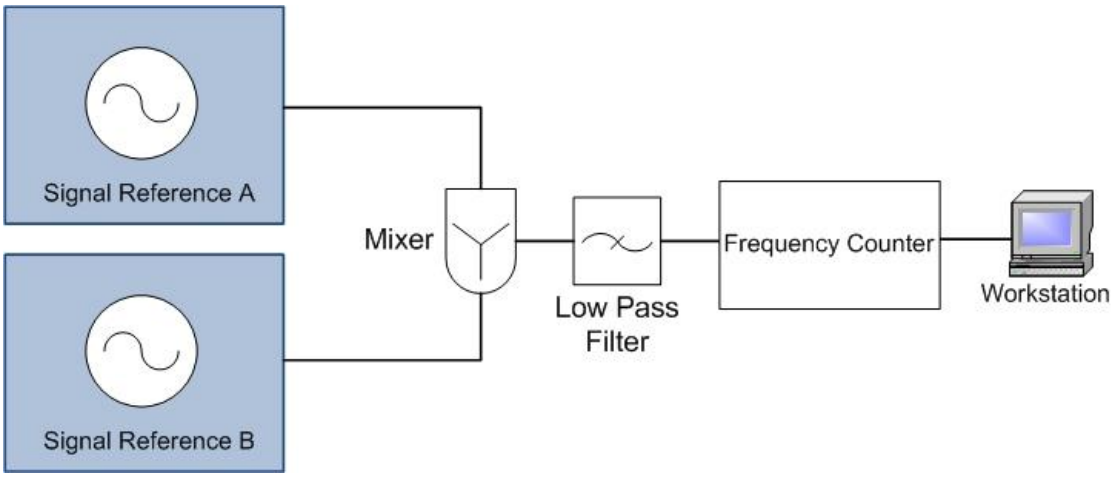

Figure 5.5: Experiment design for the participant of frequency counter

3. Execution time

From experience of running the system, we determined the execution time for various calculations with different input data as shown as table 5.1.

AVAR Type / Execute Time / Sampling Time	All possible Allan deviation	AVAR	Overlap AVAR	Modified AVAR
17 hours samples	28 hours	14 minutes	N/A	N/A
36 hours samples	38 hours	23 minutes	42 hours	58 hours
72 hours samples	46 hours	32 minutes	N/A	N/A

Table 5.1: the execution time for calculations of varies Allan deviation

We can clearly see that calculating all possible Allan deviation for every single tau takes a very long time for execution, about 38 hours for the 36 hour sample of data. The calculation of overlap AVAR has an even longer execution time of 42 hours for 36 hours recording data due to the necessity to compute overlapping data sequences. Furthermore, the Modified AVAR has the longest execution time because of complex computation, for example, 58 hours for the 36 hours samples. Impressively, the AVAR restricted to tau at only powers of 2 has a very short execution time. Apart from the AVAR, the rest of the execution time is not very good in practice. In following chapters, we attempt to provide some suggestions to solve this problem in the future.

4. Data storage

The 64-bit operating system will be suggested as the platform to run this program. There is no problem for the operation of the streaming function of this program. However, if the data used for calculating the Allan variance is too big, the program will encounter the problem of running out of memory. That is because the data used for this process is normally very large. As we mentioned previously, one array saving 36 hours data will demand nearly 500 Megabytes of memory size. In this program, there will be also several arrays requiring similar memory space. Thus, if a longer data recording is required, there is likely to be a problem of insufficient memory. Due to the limitation of memory allocation, a 32-bit operating system is only able to address 3 Gigabytes memory maximum. After launch some kernel programs of the operating system use up memory, in Windows XP for example, there are only around 2.3 Gigabytes memory available. However, running a 64-bit operating system could support a maximum 192 Gigabytes in physical memory for data processing.

5.4 Further work

There are some related developments that could be done in the future. Also, some further improvement could be undertaken for avoiding the limitations noted in this research. These possible further works can be classified in the following context.

1. Noise spectrum

Once the AVAR plot has been produced, each type noise spectrum can be plotted according to corresponding fragmentary frequency data. The frequency data in a particular time period could be computed using a Fourier transform (FFT) of the data, and then a specific noise spectrum could be obtained. It is expected that the white noise has a flat power spectral density (PSD) $S(f) = f^0$, frequency flicker noise has PSD $S(f) = f^{-1}$, and frequency random walk noise has PSD $S(f) = f^{-2}$ (Shin, Park & Lee, 2008). This information would provide additional insight into the noise processes that are occurring in atomic clocks and provide data for further advanced studies.

2. Frequency variation versus temperature

The frequency standards utilized in this project are very sensitive to the change of environment. Minor changes in the environmental temperature will appear as detectable variation in frequency. Thus, the analysis between the change of temperature and frequency could be implemented. As the result, it will aid the realization of the relationship between frequency and temperature. It is also possible to identify the best environmental temperature for the operation of atomic clocks that would help keep the best stability of these highly precise timing references.

3. Parallel computation

One of the difficulties found in this research is the program execution time, especially for processing the large data volumes associated with long-time recording. In the future, parallel computation could be one of the solutions. In our experiments, we noticed that the time for Allan variance calculation varied by the volume of data.This is understandable, for smaller tau there will be more bins required for the process. By dividing the calculation into many pieces of small program, for instance, the Allan variance for each different tau (time interval), each node could pick up one of those tasks when they are idle. The result of each task will be sent back to the master node to generate the final result. However, there may be more factors involving in using distributed computation, such as the performance of network, the complication of system design, the effectiveness and efficiency of the program, the unpredictable errors that may occur during the calculation, and so on. These will need to be considered when constructing

the parallelization. Another possible solution is to deploy the operation of a high performance super computer. Nevertheless, it is normally much more expensive and easy to exceed the budget. Although using a super computer sounds like a good alternative solution, it may be not realistic in reality.

4. Sampling over network

By utilisation of advances in networking, we may be able to transmit sampled data to a remote host. In most scenarios, the radio telescope stations are usually located in remote rural areas to prevent the interference of massive signals. Thus, it may not be convenient for a researcher to travel around. The idea here is to transmit the samples acquired from a clock located at a radio telescope station immediately to a specific host in a research centre. The ADC supporting data transmission over network will be vital to this system. Furthermore, it could be also useful to use static IP address instead of dynamic IP. Ideally, the users could run a program locally to acquire the sampling data on remote host.

5. Semi real-time representation of the atomic clock

A more complete system could be implemented for the representation of the status among the various atomic clocks in IRASR. The idea is to record the frequency offset across many atomic clock references and save those data in a workstation. There will be another program to calculate the data in fixed regularity and to represent the result in semi real-time. For example, suppose there are three stable time references, S1, S2 and S3. We could design the structure as figure 5.6.

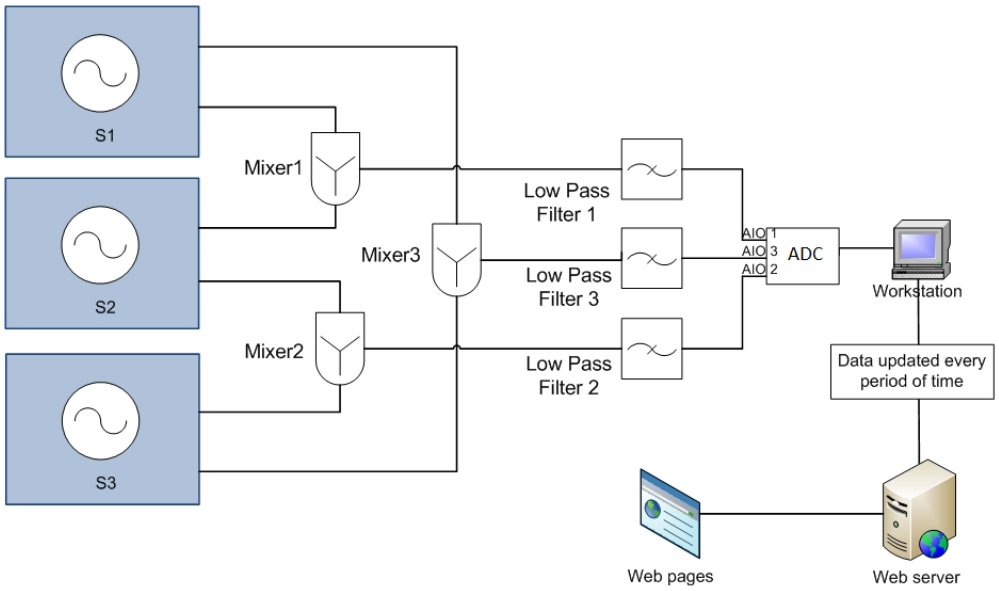

Figure 5.6: Possible structure of the semi-real time system for atomic clock status

As the result, we will be able to attain frequency offset for anyone of the three references against the others. We may set a fixed period of time for data recording, and let the program do calculations once the data recording is done. Suppose we need 72 hours for data recording (3 sets of data recording at the same time), and 1 hour for AVAR calculation (either calculate three data on same workstation or distributed computation over several workstation). Hence, the fresh cross-analysis for AVAR plotting between these three references will be updated every 73 hours. The result could be displayed locally on a particular workstation or alternatively uploaded to a website

6. Calculation for other variance

In the future, calculation for other variances could be added into the system, for instance, the dynamic Allan deviation (DAVAR), time variance (TVAR), Hadamard variance (HVAR), modified Hadamard variance (MHVAR), and so on. Another system optimisation will be required for the best execution corresponding to any new variances involved.

Chapter 6 Summary and Conclusion

Atomic clock time references are highly related to our living context. However, no oscillator can behave perfectly. A means to characterize the behaviour of a frequency reference or oscillator is the Allan deviation. For the purpose of characterization of extremely stable atomic time reference, the goal of this research is aimed to the development of a system which has the ability to produce the Allan variance from direct measurements of a clock compared to a master reference. By achieving the concepts based on ease-used and economic-efficiency the development followed the rule of rapid software development with iterative examination. The plots produced by using this system shows that there is still some confusion of how the input data should be interpreted. By computing the recorded data from various atomic clocks, the plot without frequency conversion applied showed the approximately correct slopes for white noise, flick noise and random walk noise phases. However, the level of Allan deviation for them did not meet our expectation. In contrast, by using the same recorded data, the plot applied with frequency conversion indicated that the Allan deviation stayed at reasonable level for particular tau as we expect. But, the slope and curve for them are out of our anticipation. Thus, some further experiments are strongly recommended to give more confidence on the way to choose suitable algorithm to produce the Allan variance plotting, for example, an experimental setup with use of a frequency counter.

All chapters in this research could be summarized as following.

Chapter 1 briefly introduced the importance of the atomic clock reference. The objective and contribution of this research objective are mentioned in this chapter. Also, the structure of this research is designed and procedures determined for proceeding.

Chapter 2 is presented as the background knowledge of this research. A systematic literature review of three fields, atomic clocks, Allan variance and software development, was conducted. For atomic clocks, we started from a brief history of atomic clock development, the introduction of varied atomic clocks and then to applications in which atomic clocks are crucial. In the Allan variance section, we simply introduced the theory, variants and application of Allan variance. In the final sections, we described several common software development models that were used to select an appropriate system development method for this research.

Chapter 3 discussed the methodologies applied in this research. A design science methodology was used to solve the problem of characterizing the behaviour of the atomic clock references in IRASR by developing a software-based system to calculate the Allan variance. The development process followed the principle of rapid application development with iterative and incremental factors. The reason of choosing RAD as the main approach is the limitation on the development timeframe and budget.

In Chapter 4, this research has reached the system development stage. Firstly, the equipment utilised for the experiments was briefly introduced along with their specifications. In the system development section, three main processes were organized, such as initialization, developing, and testing. In the initialization phase, plans for the functionalities, requirement and operating environment of this system were reviewed and it was decided that Matlab® should be selected as the development environment for this system. In the development phase, the programming required for producing AVAR, streaming, GUI, frequency variation plot, other variance calculations, error bar plotting, and waiting bar were all done. All codes had been iteratively revisited and examined to ensure it could meet the requirement of performance. Moreover, an optimization of codes was undertaken to improve the system performance. The finalized codes were compiled to an executable file and packaged as a self-installed program in order to allow use on any Windows operating system based machine. Finally, the system was actually used to test three atomic clocks references to test its performance.

Chapter 5 contains an in-depth discussion of the software development, the comparison of gained AVAR results, and the limitations noted in this research. Furthermore, a series of useful recommendations and potential implementations have been noted as suggestions for future

developments.

Chapter 6 is the conclusion of this research. We concluded that the software development in this research is able to meet the requirements planned initially, however, the AVAR plotting result on characterization of the behaviour on various atomic clock give us some puzzlement. A robust suggestion was made that involving a frequency counter in future experiments with same atomic clock references will clarify the uncertainty happening in this research.

Reference

Abrahamsson, P. (2003). *Extreme programming: first results from a controlled case study.* Paper presented at the 29th Euromicro Conference.

Adams, L. A., & Courtney, J. F. (2004). *Achieving relevance in IS research via the DAGS framework.* Paper presented at the 37th Annual International System Sciences, Hawaii.

Adikari, S., McDonald, C., & Collings, P. (2006). *A design science approach to an HCI research project.* Paper presented at the 18th Australia conference on Computer-Human Interaction, Design, Activities, Artefacts and Environments, Sydney, Australia.

Agarwal, R., Prasad, J., Tanniru, M., & Lynch, J. (2000). Risks of rapid application development. *Commun. ACM, 43*(11es), 1.

Allan, D. W., Ashby, N., & Hodge, C. C. (1997). The Science of Timekeeping. *Hewlett-Packard Application Note 1289*. Retrieved from http://www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf.

Allan, D. W., & Barnes, J. A. (1981). *A Modified "Allan Variance" with Increased Oscillator Characterization Ability.* Paper presented at the 35th Annual Frequency Control Symposium.

Allan, D. W. (2004). The Allan variance. Retrieved July 23, 2009, from http://www.allanstime.com/AllanVariance/

Audoin, C., & Guinot, B. (2001). *The measurement of time : time, frequency, and the atomic clock*. Cambridge; New York: Cambridge University Press.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer, 21*(5), 61-72.

Bregni, S., & Primerano, L. (2004). The modified Allan variance as time-domain analysis tool for estimating the Hurst parameter of long-range dependent traffic. Paper presented at the Global Telecommunications Conference (GLOBECOM '04).

Bruegge, B., & Dutoit, A. H. (2003). Object-oriented software engineering : using UML, patterns, and Java. Boston: Prentice Hall.

Bruggemann, T. S., Greer, D. G. and Walker, R. (2006) *Chip Scale Atomic Clocks: Benefits to Airborne GNSS Navigation Performance.* Paper presented at International Global Navigation Satellite Systems Society Symposium (IGNSS Symposium '06)

Calhoun, M., Shouhua, H., & Tjoelker, R. L. (2007). Stable Photonic Links for Frequency and Time Transfer in the Deep-Space Network and Antenna Arrays. *Proceedings of the IEEE, 95*(10), 1931-1946.

Cosart, L. D., Peregrino, L., & Tambe, A. (1997). Time domain analysis and its practical application to the measurement of phase noise and jitter. *Instrumentation and Measurement, IEEE, 46*(4), 1016-1019.

Dalcher, D., Benediktsson, O., & Thorbergsson, H. (2005). *Development life cycle management: a multiproject experiment.* Paper presented at the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS '05).

Fairley, R. E., & Willshire, M. J. (2005). Iterative rework: the good, the bad, and the ugly. *Computer, 38*(9), 34-41.

Fletcher, N., & Witt, T. J. (2008). *Some applications of times series analysis techniques to coaxial AC bridges.* Paper presented at the Precision Electromagnetic Measurements Digest (CPEM '08).

Forman, P. (1985). Atomichron: The atomic clock from concept to commercial product. *Proceedings of the IEEE, 73*(7), 1181-1204.

Galleani, L., & Tavella, P. (2003). *The characterization of clock behavior with the dynamic Allan variance.* Paper presented at the Frequency Control Symposium and PDA Exhibition Jointly with the 17th European Frequency and Time Forum.

Grantham, B. E., & Bailey, M. A. (2006). *A Least-Squares Normalized Error Regression Algorithm with Application to the Allan Variance Noise Analysis Method.* Paper presented at the IEEE/ION Position, Location and Navigation Symposium (PLANS '06).

Greenhall, C. A. (1992). *A shortcut for computing the modified Allan variance.* Paper presented at the 46th Frequency Control Symposium.

Groves, L., Nickson, R., Reeve, G., Reeves, S., & Utting, M. (2000). *A survey of software development practices in the New Zealand software industry.* Paper presented at the 12th Australian Software Engineering Conference, Australia.

Guimarães, L. R., & Vilela, P. R. S. (2005). *Comparing software development models using CDM.* Paper presented at the 6th conference on Information technology education, Newark, NJ, USA.

Han, J., Ge, S., Shen, Y., & Li, X. (2006). *Modeling and Simulation of Digital Closed-loop Fiber Optic Gyroscope.* Paper presented at the 6th World Congress on Intelligent Control and Automation (WCICA '06).

Hashmi, S. I., & Baik, J. (2007). *Software Quality Assurance in XP and Spiral - A Comparative Study.* Paper presented at the 5th International Conference on Computational Science and its Applications (ICCSA '07).

Highsmith, J. (2002). *Agile software development ecosystems*. Boston: Addison-Wesley Longman Publishing Co., Inc.

Howard, A. (2002). Rapid Application Development: rough and dirty or value-for-money engineering? *Commun. ACM, 45*(10), 27-29.

Howe, D. A., & Tasset, T. N. (2004). *Theo1: Characterization of very long-term frequency stability.* Paper presented at the 18th European Frequency and Time Forum (EFTF '04).

Hunt, J. (2005). *Agile Software Construction*. London: Springer.

Jau, Y. Y., & Happer, W. (2008). *All-photonic clock: Laser-atomic oscillator.* Paper presented at the IEEE International Frequency Control Symposium (IEEE IFCS '08).

Jeanmaire, A., Rochat, P., & Boudy, C. (1997). *Ultra-stable crystal and Rubidium oscillators for space applications.* Paper presented at the IEEE International Frequency Control Symposium (IEEE IFCS '97).

King, C.-Y. (2003). *Virtual instrumentation-based system in a real-time applications of GPS/GIS.* Paper presented at the International Conference on Recent Advances in Space Technologies (RAST '03).

Kitching, J., Knappe, S., Liew, L., Schwindt, P., Shah, K., Moreland, J., et al. (2005). *Microfabricated atomic clocks.* Paper presented at the 18th IEEE International Conference on Micro Electro Mechanical Systems (MEMS '05).

Knappe, S., Gerginov, V., Shah, V., Brannon, A., Hollberg, L., & Kitching, J. (2007). *Long-term stability of NIST chip-scale atomic clock physics packages.* Paper presented at the MOEMS and Miniaturized Systems VI, San Jose, CA, USA.

Labjack Corporation. (2009). *Labjack U3 User's Guide (-LV & -HV)*. Retrieved 14 Aug, 2009, from http://labjack.com/sites/default/files/2009/07/LabJack_U3_Users_Guide_0.pdf

Lombardi, M. A., Heavner, T. P., & Jefferts, S. R. (2007). Primary Frequency Standards and the Realization of the SI Second. *Measure: The Journal of Measurement Science, 2*, 74-89.

Lorini, L., Ashby, N., Brusch, A., Diddams, S., Drullinger, R., Eason, E., et al. (2008). Recent atomic clock comparisons at NIST. *European Physical Journal Special Topics, 163*, 19-35.

Lutwak, R., Vlitas, P., Varghese, M., Mescher, M., Serkland, D. K., & Peake, G. M. (2005). *The MAC - a miniature atomic clock.* Paper presented at the IEEE International Frequency Control Symposium and Exposition (IEEE IFCSE '05).

Macias, F., Holcombe, M., & Gheorghe, M. (2003). *A formal experiment comparing extreme programming with traditional software construction.* Paper presented at the 4th Mexican International Conference on Computer Science (ENC '03).

McGarrity, S. (2007). *Maximizing Code Performance by Optimizing Memory Access* Retrieved 23 Sep, 2009, from http://www.mathworks.com/mason/tag/proxy.html?dataid=9305&fileid=41984

McDonald, K. D. & Hegarty, C. (2000). *Post-Modernization GPS Performance Capabilities.* Paper presented at the IAIN World Congress and the Institute of Navigation 56th Annual Meeting.

Nunzi, E., Galleani, L., Tavella, P., & Carbone, P. (2007). Detection of Anomalies in the Behavior of Atomic Clocks. *IEEE Transactions on Instrumentation and Measurement, 56*(2), 523-528.

Peik, E. (2006). The Measurement of Time with Atomic Clocks In I. I. Bigi & M. Faessler (Eds.), *Time and Matter: Proceedings of the International Colloquium on the Science of Time* (pp. 3-16). Singapore: World Scientific.

Prestage, J. D., Sang, C., Thanh, L., Beach, M., Maleki, L., & Tjoelker, R. L. (2003). *One-liter Hg ion clock for space and ground applications.* Paper presented at the IEEE International

Frequency Control Symposium and PDA Exhibition Jointly with the 17th European Frequency and Time Forum.

Pyhäjärvi, M., Rautiainen, K., & Itkonen, J. (2003). *Increasing understanding of the modern testing perspective in software development projects.* Paper presented at the 36th Annual International Conference on System Sciences, Hawaii.

Ramsey, N. F. (1991). The past, present, and future of atomic time and frequency. *Proceedings of the IEEE, 79*(7), 921-926.

Ramsey, N. F. (2005). History of early atomic clocks. *Metrologia, 42*(3), S1-S3.

Ramsin, R., & Paige, R. F. (2008). Process-centered review of object oriented software development methodologies. *ACM Computing Surveys (CSUR), 40*(1), 1-89.

Rigden, J. S. (2002). *Hydrogen: the essential element*. Cambridge, Massachusetts: Harvard University Press.

Riley, W. J. (2007). Handbook of Frequency Stability Analysis. *NIST Special Publication 1065*. Retrieved from http://tf.boulder.nist.gov/general/pdf/2220.pdf

ROHDE & SCHWARZ Corporation (2009). *Time Domain Oscillator Stability Measurement - Allan variance*. Retrieved 18 Aug, 2009, from http://www2.rohde-schwarz.com/file/1EF69_E1.pdf

Services., U. S. D. o. H. H. (2008). *Selecting a development approach*. Retrieved 23 July, 2009, from http://www.cms.hhs.gov/SystemLifecycleFramework/Downloads/SelectingDevelopmentApproach.pdf

Sesia, I., & Tavella, P. (2008). Estimating the Allan variance in the presence of long periods of missing data and outliers. *Metrologia, 45*(6), S134-S142.

Shin, M. Y., Park, C., & Lee, S. J. (2008). *Atomic clock error modeling for GNSS software platform.* Paper presented at the IEEE/ION Position, Location and Navigation Symposium (PLANS '08).

Sommerville, I. (2004). *Software engineering*. Harlow: Addison Wesley.

Sterr, U. (2007). Atomic Frequency Standards and their Impact on the Past, Present and Future of the Second. Paper presented at the IEEE and LEOS Summer Topical Meetings.

Succi, G., Stefanovic, M., & Pedrycz, W. (2001). *Quantitative assessment of extreme programming practices.* Paper presented at the Canadian Conference on Electrical and Computer Engineering (CCECE '07).

Theiss, A., Yen, D. C., & Ku, C. Y. (2005). Global Positioning Systems: an analysis of applications, current development and future implementations. *Computer Standards & Interfaces, 27*(2), 89-100.

Tomatis, A., Orgiazzi, D., & Mulassano, P. (2008). *Innovative Strategy for Vehicle Position Certification on the basis of GNSS reference time.* Paper presented at the IEEE/ION Position, Location and Navigation Symposium (PLANS '08).

Vaishnavi, V. K. & Kuechler, W. (2008). Design Science Research Methods and Patterns: Innovating Information and Communication Technology. New York: Auerbach Publications.

Venable, J. (2006). *The role of theory and theorising in Design Science research.* Paper presented at the 1st International Conference on Design Science Research in Information Systems and Technology, Claremont, California.

Venkataraman, P. (2009). *Applied Optimization with MATLAB Programming*: Wiley Publishing.

Vernotte, F., Addouche, M., Delporte, M., & Brunet, M. (2004). *The three cornered hat method: an attempt to identify some clock correlations.* Paper presented at the IEEE International Frequency Control Symposium and Exposition (IFCSE ‘04)

Yao, J. F., Ying, S., Luo, J. B., Xie, D., & Jia, X. Y. (2006). *Reflective Architecture Based Software Testing Management Model.* Paper presented at the IEEE International Conference on Management of Innovation and Technology (ICMIT '06).

Zhang, X. M., Liut, A. Q., Cai, H., Lu, C., & Tang, D. Y. (2005). *Theory and experiment of miniaturized laser source for accurate timing applications.* Paper presented at the 7th IEEE Conference on Sensors.

Appendix A: Other AVAR plotting

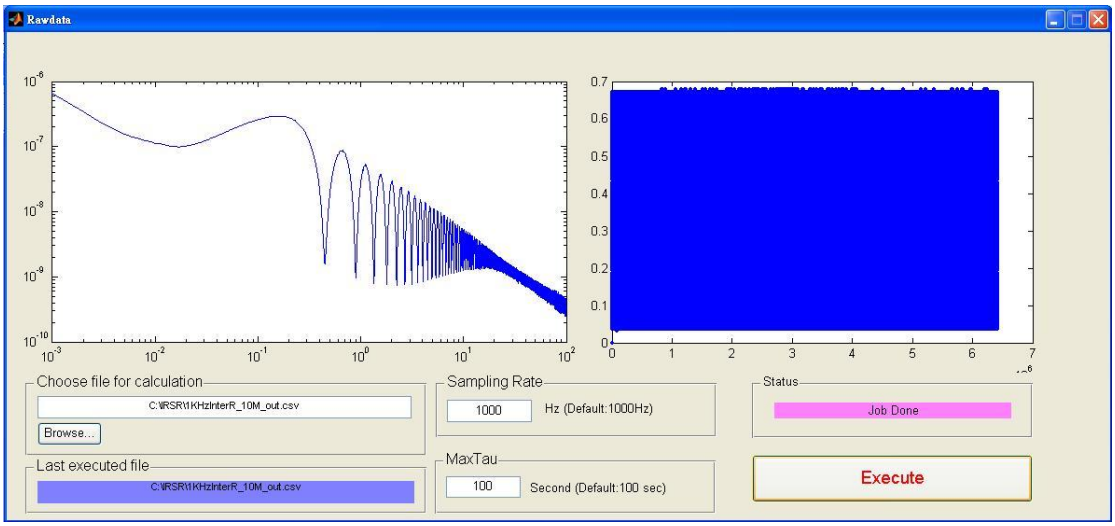

Figure A.1: ADEV and voltage variation plotting for 2 hours recording of two 10 MHz references produced by signal generators internally at 1KHz sampling rate

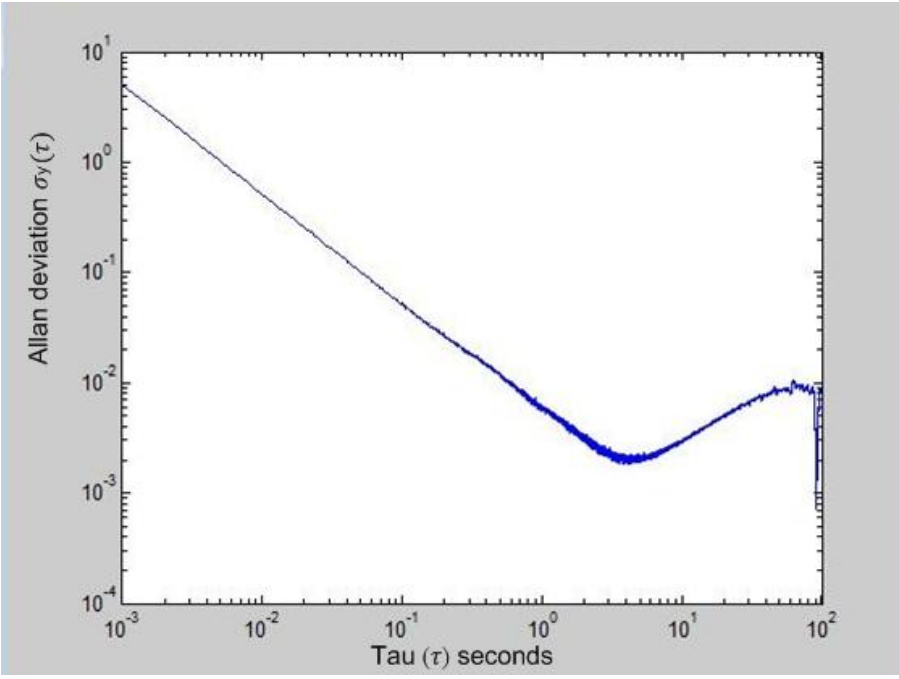

Figure A.2: ADEV plotting for 17 hours recording of two rubidium atomic clocks at 1 KHz sampling rate (with frequency conversion)

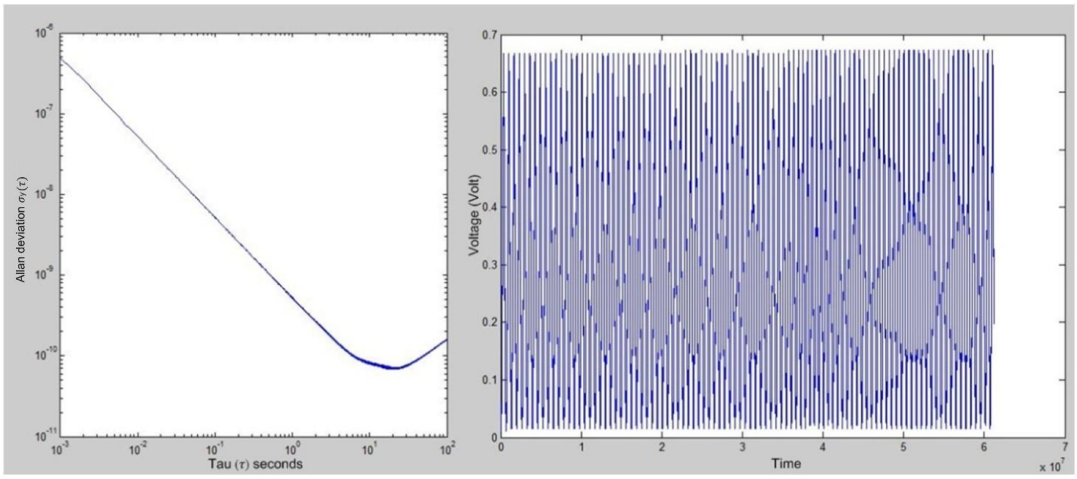

Figure A.3: ADEV and voltage variation plotting for 17 hours recording of two rubidium atomic clocks running 20 MHz at 1 KHz sampling rate (with frequency conversion)

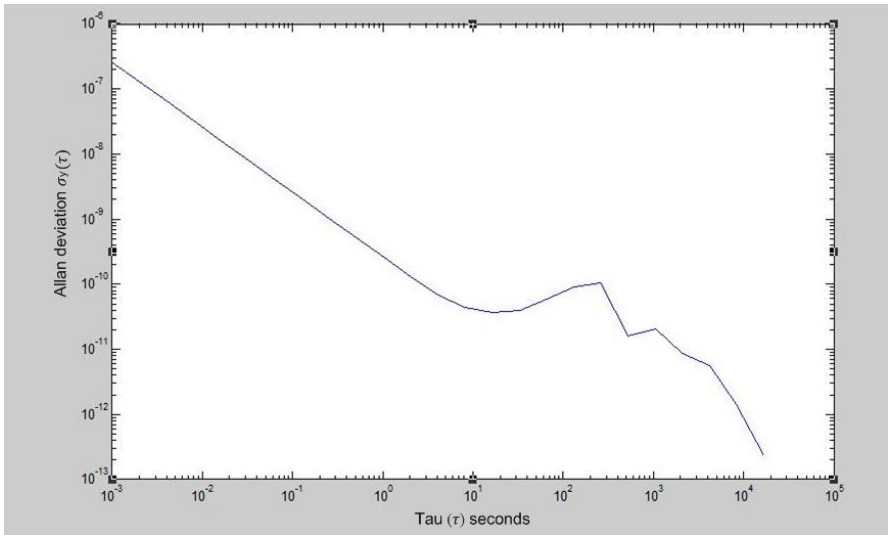

Figure A.4: ADEV plotting at maximum tau 32768 seconds for 17 hours recording of two rubidium atomic clocks running 20 MHz at 1 KHz sampling rate (with frequency conversion)

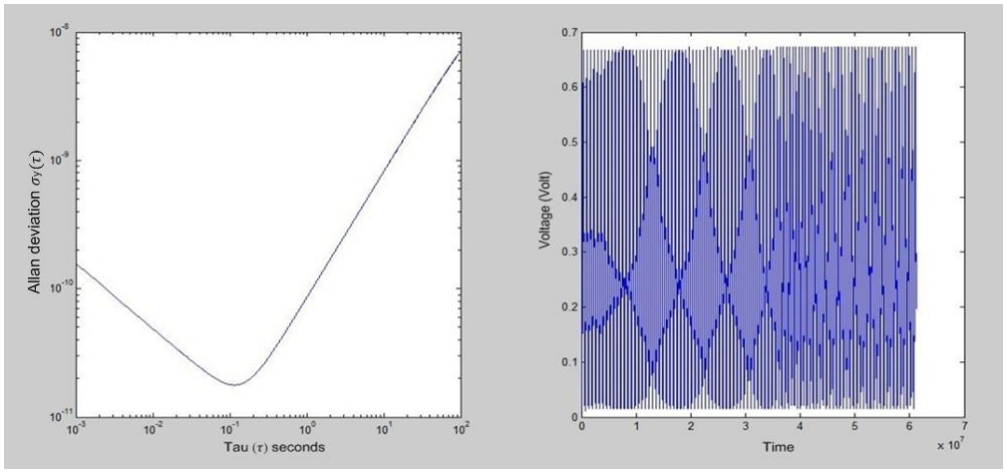

Figure A.5: AVAR and voltage variation plotting for 17 hours recording of two rubidium atomic clocks running 20 MHz at 1 KHz sampling rate (without frequency conversion)

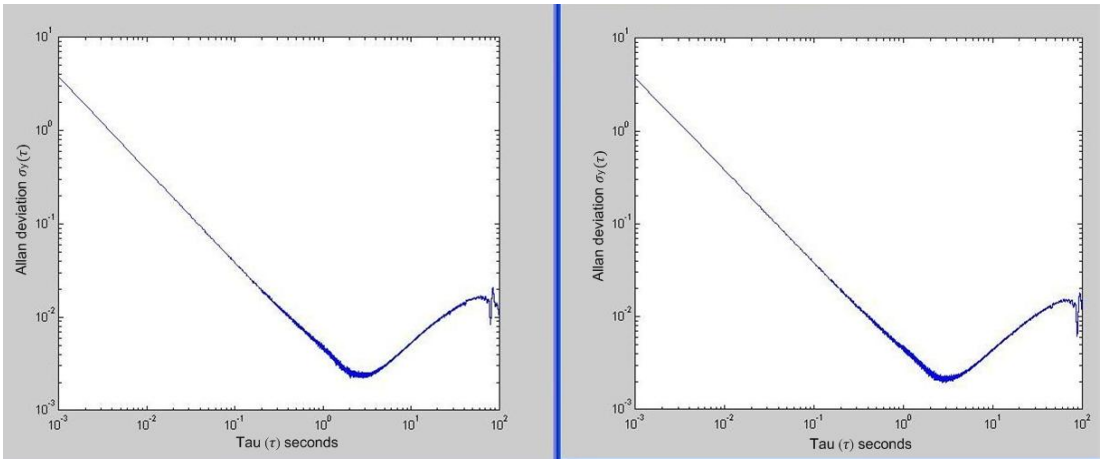

Figure A.6: Difference of ADEV plots for 17 hours recording of two rubidium atomic clock if signal generator involved

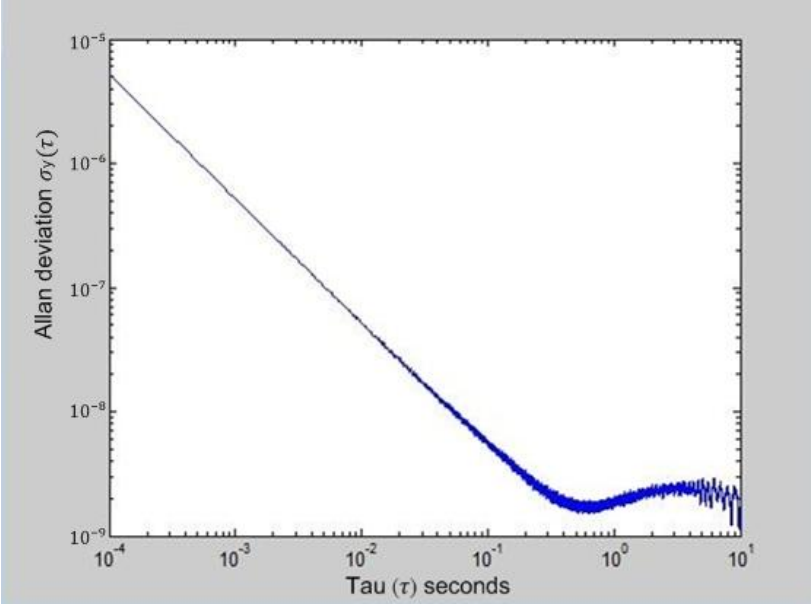

Figure A.7: ADEV plot for 3 hours recording of GPS disciplined clock versus hydrogen maser at 100 Hz sampling rate

Appendix B: Voltage variation plots for GPS vs Hydrogen Maser, Rubidium atomic clock vs Hydrogen Maser, and adjusted Rubidium atomic clock vs Hydrogen Maser

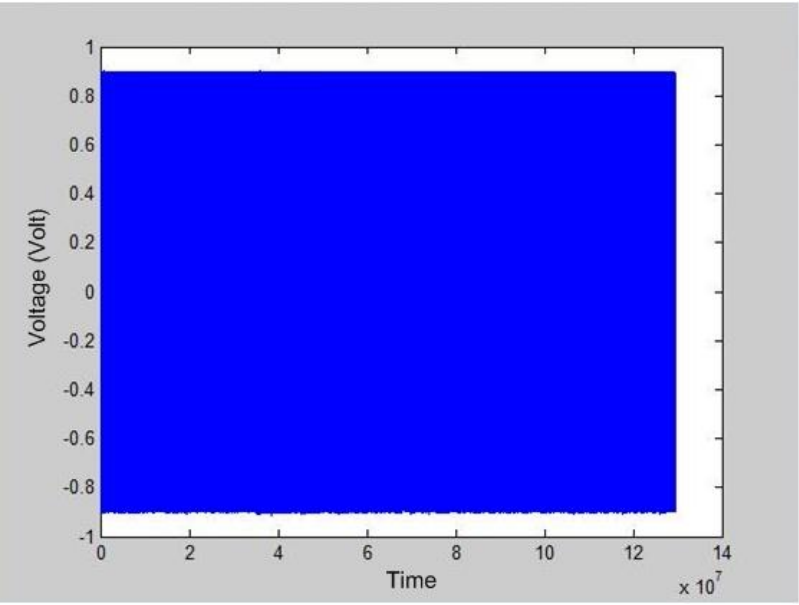

Figure B.1: Voltage variation of Rubidium atomic clock (without correction) versus hydrogen maser

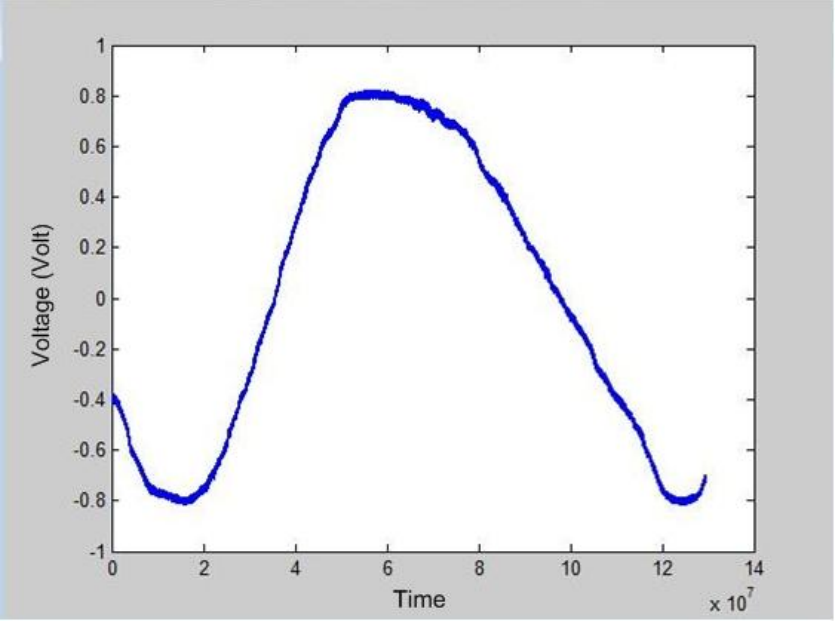

Figure B.2: Voltage variation of Rubidium atomic clock (without correction) versus hydrogen maser

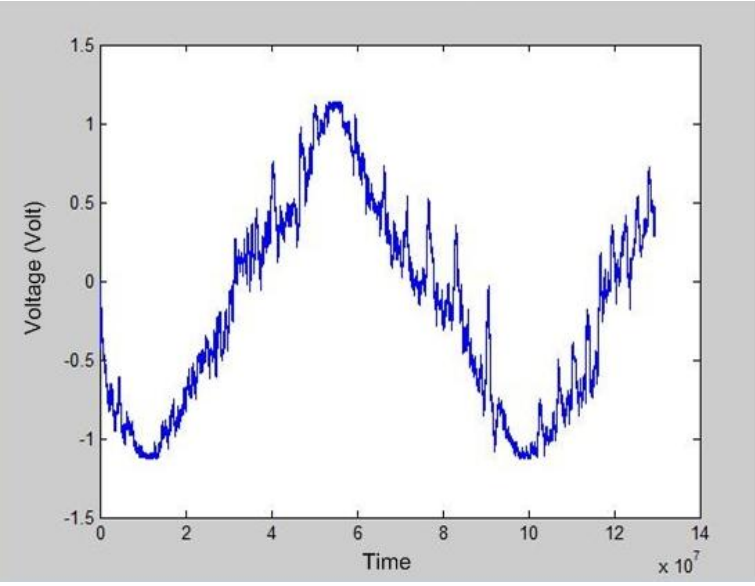

Figure B.3: Voltage variation of GPS disciplined clock versus hydrogen maser

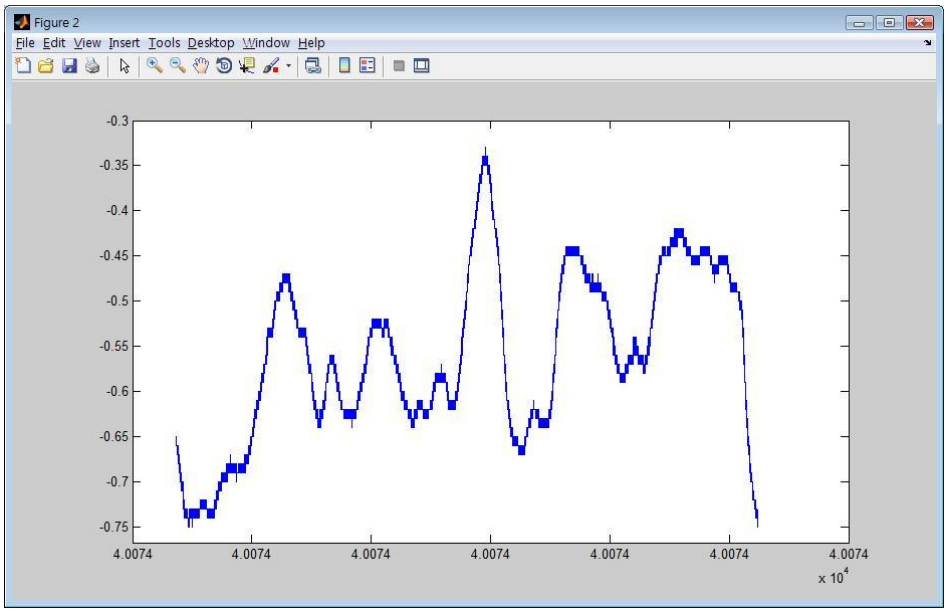

Figure B.4: Voltage variation of GPS disciplined clock versus hydrogen maser at 100 Hz sampling rate

Appendix C: Codes for Rawdata.m that creates GUI for AVAR calculation

```
function varargout = Rawdata(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',           mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn', @Rawdata_OpeningFcn, ...
                   'gui_OutputFcn',    @Rawdata_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Rawdata_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for Rawdata
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Rawdata_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
```

```
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% get the path of file for execution
function pushbutton2_Callback(hObject, eventdata, handles)
global FileToRead
[filename,pathname] = uigetfile('*.*','select file');
FileToRead=fullfile(pathname,filename);
set(handles.edit1,'string',FileToRead);
guidata(hObject,handles);

function pushbutton1_Callback(hObject, eventdata, handles)
global FileToRead
ScanRate=str2num(get(handles.edit3,'string'));
MaxTau=str2num(get(handles.edit2,'string'));
set(handles.text4,'string','Job Begin');
%=========================================
    if get(handles.checkbox1,'value')==1
        conversion=1;
    else
        conversion=0;
    end
    if get(handles.radiobutton1,'value')==1
        cal=1;
    elseif get(handles.radiobutton2,'value')==1
        cal=2;
    elseif get(handles.radiobutton3,'value')==1
        cal=3;
    elseif get(handles.radiobutton4,'value')==1
        cal=4;
    end

    [AVAR, tau, Fre, erb] = avar(FileToRead,ScanRate,MaxTau,cal,conversion);
```

```
    errorbar(handles.axes1,tau,AVAR,erb);
    set(handles.axes1,'xscale','log');
    set(handles.axes1,'yscale','log');
    xlabel(handles.axes1,'Tau','fontsize',8,'fontweight','b');
    ylabel(handles.axes1,'Allan deviation','fontsize',8,'fontweight','b');

    plot(handles.axes2,Fre);
    xlabel(handles.axes2,'time','fontsize',8,'fontweight','b');
    ylabel(handles.axes2,'Voltage','fontsize',8,'fontweight','b');

    set(handles.text1,'string',FileToRead);
    set(handles.text4,'string','Job Done');
    guidata(hObject,handles);

function edit2_Callback(hObject, eventdata, handles)

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function pushbutton3_Callback(hObject, eventdata, handles)
[filename,pathname] = uiputfile('*.jpg','select file');
FileToSave=fullfile(pathname,filename);
f = getframe(gcf,[14 208 552 385]);
f = frame2im(f);
imwrite(f,FileToSave);
```

```
function pushbutton4_Callback(hObject, eventdata, handles)
[filename,pathname] = uiputfile('*.jpg','select file');
FileToSave=fullfile(pathname,filename);
f = getframe(gcf,[580 295 540 320]);
f = frame2im(f);
imwrite(f,FileToSave);

function checkbox1_Callback(hObject, eventdata, handles)

function Untitled_1_Callback(hObject, eventdata, handles)
print(handles.figure1);

function Untitled_2_Callback(hObject, eventdata, handles)
close;
```

Appendix D: Codes for labjack2.m that creates GUI for using Labjack

```
function varargout = labjack2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',            mfilename, ...
                   'gui_Singleton',    gui_Singleton, ...
                   'gui_OpeningFcn', @labjack2_OpeningFcn, ...
                   'gui_OutputFcn',    @labjack2_OutputFcn, ...
                   'gui_LayoutFcn',    [] , ...
                   'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function labjack2_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for labjack2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = labjack2_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function edit5_Callback(hObject, eventdata, handles)
if(isempty(get(hObject,'string')))
    set(hObject,'string','1000');
end
Hz = str2double(get(hObject,'string'));
if isnan(Hz)
    errordlg('Please enter an integer','Wrong Input','modal')
end
guidata(hObject,handles);

function edit5_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit4_Callback(hObject, eventdata, handles)
global FileToRead
if (isempty(get(hObject,'string')))==0
    FileToRead=get(hObject,'string');
    set(handles.text4,'string',FileToRead);
end
guidata(hObject, handles);

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function pushbutton1_Callback(hObject, eventdata, handles)
global FileToRead
[filename,pathname] = uigetfile('*.*','select file');
FileToRead=fullfile(pathname,filename);
set(handles.edit4,'string',FileToRead);
guidata(hObject, handles);
```

```
function edit1_Callback(hObject, eventdata, handles)
if(isempty(get(hObject,'string')))
    set(hObject,'string','0');
end
Hour = str2double(get(hObject,'string'));
if isnan(Hour)
    errordlg('Please enter an integer','Wrong Input','modal')
end
guidata(hObject,handles);

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit2_Callback(hObject, eventdata, handles)
if(isempty(get(hObject,'string')))
    set(hObject,'string','0');
end
Minute = str2double(get(hObject,'string'));
if isnan(Minute)
    errordlg('Please enter an integer','Wrong Input','modal')
end
guidata(hObject,handles);

function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit3_Callback(hObject, eventdata, handles)
if(isempty(get(hObject,'string')))
    set(hObject,'string','0');
```

```
end
Second = str2double(get(hObject,'string'));
if isnan(Second)
    errordlg('Please enter an integer','Wrong Input','modal')
end
guidata(hObject,handles);

function edit3_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit6_Callback(hObject, eventdata, handles)
if(isempty(get(hObject,'string')))
    set(hObject,'string','1000');
end
MaxTau = str2double(get(hObject,'string'));
if isnan(MaxTau)
    errordlg('Please enter an integer','Wrong Input','modal')
end
guidata(hObject,handles);

function edit6_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function checkbox1_Callback(hObject, eventdata, handles)

function pushbutton2_Callback(hObject, eventdata, handles)
  global FileToRead
  H=str2num(get(handles.edit1,'string'));
  M=str2num(get(handles.edit2,'string'));
  S=str2num(get(handles.edit3,'string'));
```

```
    SamplingTime = (H*60*60)+(M*60)+S;
    ScanRate=str2num(get(handles.edit5,'string'));
    MaxTau=str2num(get(handles.edit6,'string'));
        if get(handles.checkbox2,'value')==1
            conversion=1;
        else
            conversion=0;
        end

        if get(handles.radiobutton6,'value')==1
           range=1;
        elseif get(handles.radiobutton5,'value')==1
           range=2;
        end
if get(handles.checkbox1,'value')==1
        set(handles.text5,'string','Begin Streaming');
        Stream(FileToRead,range,SamplingTime,ScanRate,MaxTau);
        set(handles.text5,'string','Streaming Done');
        set(handles.text4,'string',FileToRead);
        set(handles.text5,'string','Job Done');
        guidata(hObject,handles);

else
        set(handles.text5,'string','Begin Streaming');
        Stream(FileToRead,range,SamplingTime,ScanRate,MaxTau);
        set(handles.text5,'string','Streaming Done & Begin Plotting AVAR');
        if get(handles.radiobutton1,'value')==1
            cal=1;
            elseif get(handles.radiobutton2,'value')==1
            cal=2;
            elseif get(handles.radiobutton3,'value')==1
            cal=3;
            elseif get(handles.radiobutton4,'value')==1
            cal=4;
        end
        [AVAR, tau, Fre, erb] = avar(FileToRead,ScanRate,MaxTau,cal,conversion);

        errorbar(handles.axes1,tau,AVAR,erb);
```

```
    set(handles.axes1,'xscale','log');
    set(handles.axes1,'yscale','log');
    xlabel(handles.axes1,'Tau','fontsize',8,'fontweight','b');
    ylabel(handles.axes1,'Allan deviation','fontsize',8,'fontweight','b');

    plot(handles.axes2,Fre);
    xlabel(handles.axes2,'time','fontsize',8,'fontweight','b');
    ylabel(handles.axes2,'Voltage','fontsize',8,'fontweight','b');

    set(handles.text4,'string',FileToRead);
    set(handles.text5,'string','Job Done');
    guidata(hObject,handles);
end

function Untitled_1_Callback(hObject, eventdata, handles)
print(gcf);

function Untitled_2_Callback(hObject, eventdata, handles)
close;

function pushbutton3_Callback(hObject, eventdata, handles)
[filename,pathname] = uiputfile('*.jpg','select file');
FileToSave=fullfile(pathname,filename);
f = getframe(gcf,[2 227 540 385]);
f = frame2im(f);
imwrite(f,FileToSave);

function pushbutton4_Callback(hObject, eventdata, handles)
[filename,pathname] = uiputfile('*.jpg','select file');
FileToSave=fullfile(pathname,filename);
f = getframe(gcf,[575 300 525 310]);
f = frame2im(f);
imwrite(f,FileToSave);

function checkbox2_Callback(hObject, eventdata, handles)
```

Appendix E: Codes for avar.m that calculates Allan deviation

```
function [AVAR, tau, Fre, erb] = avar(FileToRead,ScanRate,MaxTau,cal,conversion)
tic;
%Load the data from raw data file
y=single(load('-ascii',FileToRead));
Fre=y(5000:5500);
%Calculate the number fo the samples for raw data
%A=length(y);
%Use the following codes to round down the raw data value after 2nd decimal place.
%%rawdata=fix(y*100)/100;
%If doing the round down, re-calculate the number of the samples
%%A=length(y);

%Define the tau0 as the interval time of sampling
tau0=1/ScanRate;

%Convert the phase offset to frequency offset
   if conversion==1
      y=diff(y)/tau0;
   end
%Ccalculate the number of samples after conversion
n=length(y);

fprintf('"$" = finish the calculation for a particular tau\n');
h=waitbar(0,'Just begin calculation...Please wait');
if cal==1
      %Pre-define the size for tau
      tau=single(zeros(1,MaxTau/tau0)');
      %Pre-define the size for AVAR
      AVAR=single(zeros(1,MaxTau/tau0)');
      %%Pre-define the size for Errorbar
      erb=single(zeros(1,MaxTau/tau0)');
      for i=1:MaxTau/tau0
            %Calculate the tau
            tau(i)=i*tau0;
```

```
        %Pre-define the bin
        Bin=single(zeros(1,floor(n/i))');
        %Calculate the value for each bin
            for j=0:n/i-1
                Bin(j+1)=(sum(y(i*j+1:(j+1)*i))/i)/2e+7;
            end
        %Calculate the number of bins
        m=length(Bin);
        %Calculate every possible AVAR
        AVAR(j)=(sqrt(0.5*(sum(diff(Bin).^2))/(m-1)))';
        %calculate the Errorbar
        erb(j)=AVAR(j)/sqrt(m);
        %Once one AVAR calculated, print one "@" mark
        fprintf('@');
        %Delete the Bin array
        clear Bin;
        %show the percentage in process by waitbar
        if isinteger((j/MaxTau/tau0)*10)==1
            per=fix(j/MaxTau/tau0*100);
            close(h);
            waitbarstr=['Done ', num2str(per),'%'];
            h=waitbar(j/MaxTau/tau0,waitbarstr);
            pause(0.5);
        end
        end

    elseif cal==2
        maxi=floor(log2(n));
        erb=single(zeros(1,maxi)');
        tau=single(zeros(1,maxi)');
        AVAR=single(zeros(1,maxi)');
        for i=1:maxi
            nt=2^(i-1);
            Bin=single(zeros(1,floor(n/nt))');
                for j=0:floor(n/nt)-1
                    Bin(j+1)=(sum(y((i*j)+1:(j+1)*i))/nt)/1e+7;
                end
            m=length(Bin);
```

```
        %Calculate every possible AVAR
        AVAR(i)=sqrt(0.5*(sum(diff(Bin).^2))/(m-1));
        %Calculate the tau
        tau(i)=nt*tau0;
        %Once one AVAR calculated, print one "$" mark
        fprintf('$');
        %Delete the Bin array
        clear Bin;
        %calculate the Errorbar
        erb(i)=AVAR(i)/sqrt(m);
        %display(erb);
        %display(AVAR);
        %display(tau);
        %show the percentage in process by waitbar
        per=fix(i/maxi*100);
        close(h);
        waitbarstr=['Done ', num2str(per),'%'];
        h=waitbar(i/maxi,waitbarstr);
        pause(0.5);
    end
elseif cal==3
    maxi=floor(log2(n));
    erb=single(zeros(1,maxi)');
    tau=single(zeros(1,maxi)');
    AVAR=single(zeros(1,maxi)');
    for i=0:maxi-1
        nt=2^i;
        Bin=single(zeros(1,n-nt+1)');
        for j=1:n-nt+1
            Bin(j)=sum(y(j:j+nt-1))/nt;
        end
        %create the sequence for overlap Bin
        S1=Bin(nt+1:n+1-nt);
        S2=Bin(1:n+1-2*nt);
        u=sum((S1-S2).^2);
        e=length(S1);
        %calculate the Overlap AVAR
        AVAR(i+1)=sqrt(0.5*u/n+1-2*nt);
```

```
            %calculate the Errorbar
            erb(i+1)=AVAR(i+1)/sqrt(e);
            %Calculate the tau
            tau(i+1)=i*tau0;
            %Once one AVAR calculated, print one "$" mark
            fprintf('$');
            %Delete the Bin array
            clear Bin S1 S2;
            per=fix(i/(maxi-1)*100);
            close(h);
            waitbarstr=['Done ', num2str(per),'%'];
            h=waitbar(i/(maxi-1),waitbarstr);
            pause(0.5);
        end

elseif cal==4
        maxi=floor(log2(n));
        erb=single(zeros(1,maxi)');
        tau=single(zeros(1,maxi)');
        AVAR=single(zeros(1,maxi)');
            for i=0:maxi-1
            nt=2^i;
            Bin=single(zeros(1,n-nt+1)');
            for j=1:n-nt+1
                Bin(j)=sum(y(j:j+nt-1))/nt;
            end
            %MVAR
            T=single(zeros(1,n+2-3*nt)');
            for k=0:n+1-3*nt
                S1=Bin(1+k:nt+k);
                S2=Bin(1+nt+k:2*nt+k);
                T(k+1)=(sum(S2-S1))^2;
                clear S1 S2;
            end
            e=length(T);
            T=mean(T);
            AVAR(i+1)=sqrt(T/2)/nt;
            %calculate the Errorbar
```

```
            erb(i+1)=AVAR(i+1)/sqrt(e);
            %Calculate the tau
            tau(i+1)=i*tau0;
            %Once one AVAR calculated, print one "$" mark
            fprintf('$');
            %Delete the Bin array
            clear Bin T;
            per=fix(i/(maxi-1)*100);
            close(h);
            waitbarstr=['Done ', num2str(per),'%'];
            h=waitbar(i/(maxi-1),waitbarstr);
            pause(0.5);
            end
else
fprintf('System error!! Please double check\n');
end
close(h);
fprintf('\n Done calculation!!');
clear y;
%calculate the execution time
toc;
return
```

Appendix F: Codes for Stream.m that functions streaming

```
function Stream(FileToRead,range,ST,ScanRate,MaxTau)
display(ST);
display(ScanRate);
display(MaxTau);
pausetime = 1; % waiting time between each data reading from the buffer
Loops = round(ST/pausetime); % Number of data readings from the buffer
num_channels = 1; % Number of channels used for streaming, it is also used to setup the
buffer size
buffer = 200; %variable to setup the buffer size on PC-RAM

%The workflow of streaming is as follows:
%Load drivers.
%Define streaming variables
%Configure the labjack with appropriate variables
%Check if the configuration done properly
%Start the stream.
%Data will be loaded into an arrary
%Save the array on the file

% Load Function Library from Labjack driver
% check if the library is loaded
if (libisloaded('labjackud'))
    else
    loadlibrary('c:\windows\system32\labjackud.dll', @loaddriver, 'alias','labjackud');
end

ljud_Constants; % Load LabJack UD constants
%------I/O setting------
%Open Labjack
[Error ljHandle] = ljud_OpenLabJack(LJ_dtU3,LJ_ctUSB,'1',1);
Error_Message(Error) % Check if error occurs

%Reset all IO on Labjack
Error = ljud_ePut(ljHandle, LJ_ioPIN_CONFIGURATION_RESET, 0, 0, 0);
```

```
Error_Message(Error)

% Configure AIN0 as analog input
Error = ljud_AddRequest(ljHandle,LJ_ioPUT_ANALOG_ENABLE_BIT,0,1,0,0);
Error_Message(Error)
%------I/O setting------

% Configure Labjack as 12-bit resolution
Error = ljud_AddRequest(ljHandle,LJ_ioPUT_CONFIG,LJ_chAIN_RESOLUTION,12,0,0);
Error_Message(Error)
% Configure AIN0 for Bipolar 2V or 5V range depending on the value GUI callback return
if range==1
    Error = ljud_AddRequest(ljHandle,LJ_ioPUT_AIN_RANGE,1,LJ_rgBIP2V,0,0);
    Error_Message(Error)
elseif range==2
    Error = ljud_AddRequest(ljHandle,LJ_ioPUT_AIN_RANGE,1,LJ_rgBIP5V,0,0);
    Error_Message(Error)
end

% Configure Scan Rate
Error =
ljud_AddRequest(ljHandle,LJ_ioPUT_CONFIG,LJ_chSTREAM_SCAN_FREQUENCY,ScanR
ate,0,0);
Error_Message(Error)

% allocate a buffer on PC Ram for Labajck (ScanRate * Channels # * buffer variable)
Error =
ljud_AddRequest(ljHandle,LJ_ioPUT_CONFIG,LJ_chSTREAM_BUFFER_SIZE,ScanRate*nu
m_channels*buffer,0,0);
Error_Message(Error)

% Configure that no waiting time for data retrieve whatever the data is available or not
Error =
ljud_AddRequest(ljHandle,LJ_ioPUT_CONFIG,LJ_chSTREAM_WAIT_MODE,LJ_swNONE,0,
0);
Error_Message(Error)

% Clear stream channels
```

```
Error = ljud_AddRequest(ljHandle,LJ_ioCLEAR_STREAM_CHANNELS,0,0,0,0);
Error_Message(Error)

% Define the scan I/O
Error = ljud_AddRequest(ljHandle,LJ_ioADD_STREAM_CHANNEL,1,0,0,0);
Error_Message(Error)

% Execute the scan I/O.
Error = ljud_GoOne(ljHandle);
Error_Message(Error)

%Check if everything configured
% Get all results for errors checking
Error = ljud_GetFirstResult(ljHandle,0,0,0,0,0);
Error_Message (Error)

% Run while loop to ensure everything has been setup properly
while (Error ~= 1006) % 1006 Equates to LJE_NO_MORE_DATA_AVAILABLE
        Error = ljud_GetNextResult(ljHandle,0,0,0,0,0);
        if ((Error ~= 0) && (Error ~= 1006))
                Error_Message (Error)
                break
        end
end

% Start streaming
Error = ljud_ePut(ljHandle,LJ_ioSTART_STREAM,0,0,0);
Error_Message(Error)

for n = 0:Loops-1
        %define the number of data for every reading
        Scans = (ScanRate/1000) * (pausetime*1000);
        % create an array for data storage
        array(Scans)=double(0);
        % Wait a while for saving data on PC RAM buffer
        pause (pausetime)

        % Save the data samples data on the return array
```

```
        [Error Scans return_array] =
        ljud_eGet_array(ljHandle,LJ_ioGET_STREAM_DATA,LJ_chALL_CHANNELS,Scans,arr
        ay);
        Error_Message(Error)

        %Save the data on the array into the user-selected file
        fid=fopen(FileToRead,'a+');
        %save the data to fourth decimal place
        fprintf(fid,'%1.4f\n',return_array);
        fclose(fid);
        %Clear the array used
        clear return_array
        clear array
end

% Stop the stream
[Error] = ljud_ePut(ljHandle,LJ_ioSTOP_STREAM,0,0,0);
Error_Message(Error)

display('Streaming finished\n')
display('Done')
return
```