A High Resolution Performance Monitoring Software on the Pentium

Ong Cheng Soon*, Fadhli Wong Mohd Hasan Wong**, Lai Weng Kin*

* Software Lab, MIMOS Berhad
lai@mimos.my, csong@mimos.my
** Dept of Electrical & Computer, IIUM
fadw@yahoo.com

Abstract: *Measuring time accurately and precisely is essential for many applications in this modern era. Often, specialized and expensive hardware or software devices may be needed to obtain a high degree of accuracy. This paper proposes a time measuring performance-monitoring software, which is cost effective and accurate on the most ubiquitous of platforms, a Windows based PC. It utilises features of the Intel Pentium chip set to produce a high resolution and accurate measurement of the time period between events. Various experiments investigating the performance of the system were conducted, and the results of the experiments are discussed.*

Keywords: *Timers, Personal Computer (PC), High Resolution, Accurate, Pentium, Performance Monitoring*

I. INTRODUCTION

The history of electrical time keeping had existed since the first invention, in 1335, of a mechanical clock in Milan [1]. By 1889, a clock was produced, which had one pendulum swinging freely and another to control the clock mechanism that kept time to the accuracy of 1 second per year. This design of clock system was used all over the world until it gave way to the emergence of quartz crystal clock by 1942. Improvements in electronic technology enabled quartz clocks to become smaller. By 1961, portable quartz clocks were being produced for navigation purposes and today, quartz crystals are found in almost all timing devices ranging from a simple wrist watch to microchips and controllers found in electronic devices and systems.

It would have been difficult for our forefathers to imagine that we need to measure time so accurately but it is essential for many aspects of modern life, which we often take for granted. Modern navigation systems, mobile telephones, nuclear physics, real time measurement and automation, performance monitoring, digital television and many industrial and commercial activities need a high level of accuracy. Accuracy and precision are the main considerations when developing and designing a timing system. The level of accuracy and precision often depends on the type of time measurement or monitoring involved.

While a simple time keeping device such as a pendulum can be used to keep track of date and time, a stopwatch will need resolution of milliseconds (ms), testing software or network performance might need accuracy down to microseconds (μs). Higher resolution of timing devices will be needed in areas such as nuclear physics where accuracy of nanoseconds (ns) and picoseconds (ps) are required. Figure 1 depicts the various applications and fields with different need of timing resolutions. Different methodologies and approaches of time measurement apply to different systems. For example, a program using built-in functions *(Appendix A)* of high-level languages that utilizes the clock system in a personal computer (PC) can be designed to measure the performance of a network system with accuracy of milliseconds. Specially designed external hardware may be used when higher resolution in the range of ns or ps is needed.

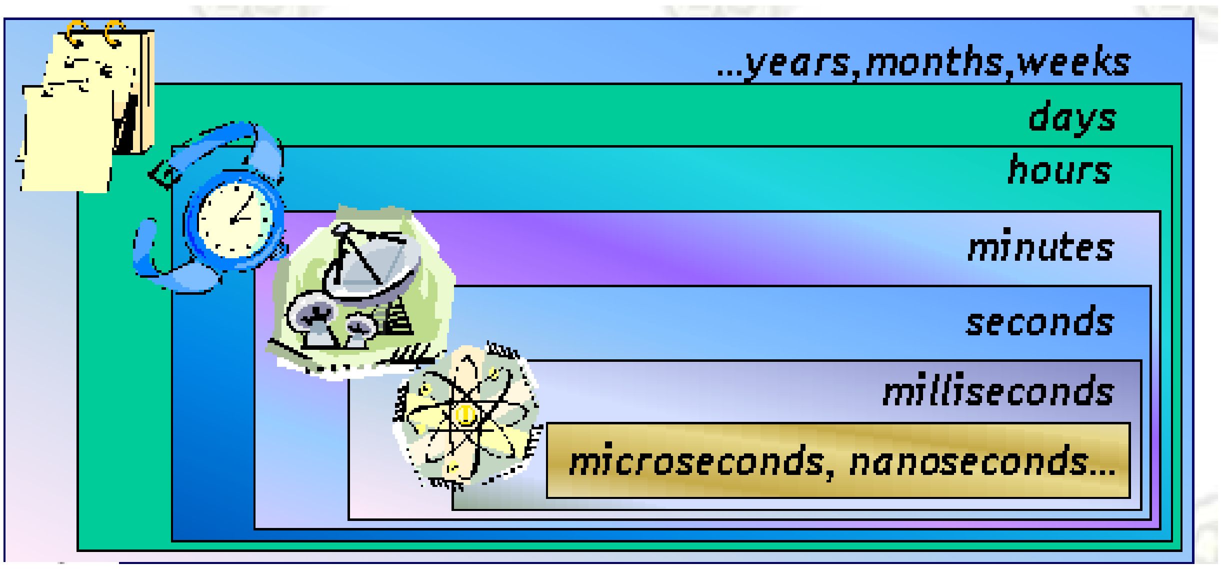

Fig 1: Samples of applications with different timing resolutions

In this research paper, the various stages of design and development of a high resolution Pentium based performance-monitoring software are discussed. The main usage of this system would be timing of instruction codes for optimization of software run-time, and real time measurement and control. In Section II, the sources of hardware timing system in a PC will be introduced. Section III will highlight some of the important issues and considerations taken into account when developing the software. The experimental results and analysis of the proposed software system will be reported in Section IV and a brief summary and conclusion are discussed in Section V.

II. PC-BASED TIME MEASUREMENTS

A. Sources of Hardware Timing System

There are three main sources of hardware timing system that can be derived from the computer system namely the Intel 8253/8254 counter/timer chip (CTC) form the IBM PC family, the real time clock (RTC) - *(Figure 2)* and the processor itself.

The first option for timing is to use the CTC. The frequency of the CTC depends on the size and cutting of the quartz crystal. Cutting crystals of quartz in different ways can make them to vibrate at almost any frequency [1]. The standard frequency of 14.31818 MHz was firstly introduced by IBM and this frequency has been used in every PC's system clock ever since. The 14.31818 MHz system clock is then divided by 12 to give a 1.193182 MHz clock (period is 0.8381 μs) which clocks the three channels of the 8253/8254 counter/timer chip (CTC) [2,3]. The CTC, then divides this frequency to lower frequencies of 18.2065 Hz using programmable divisors (a 16-bit counter), and produces three output signals. It is possible to read the actual count in progress in the CTC. In combination with the tick count variable, this can give an absolute time value, in units of 0.8381 μs, for time stamping, elapsed time calculation, etc.

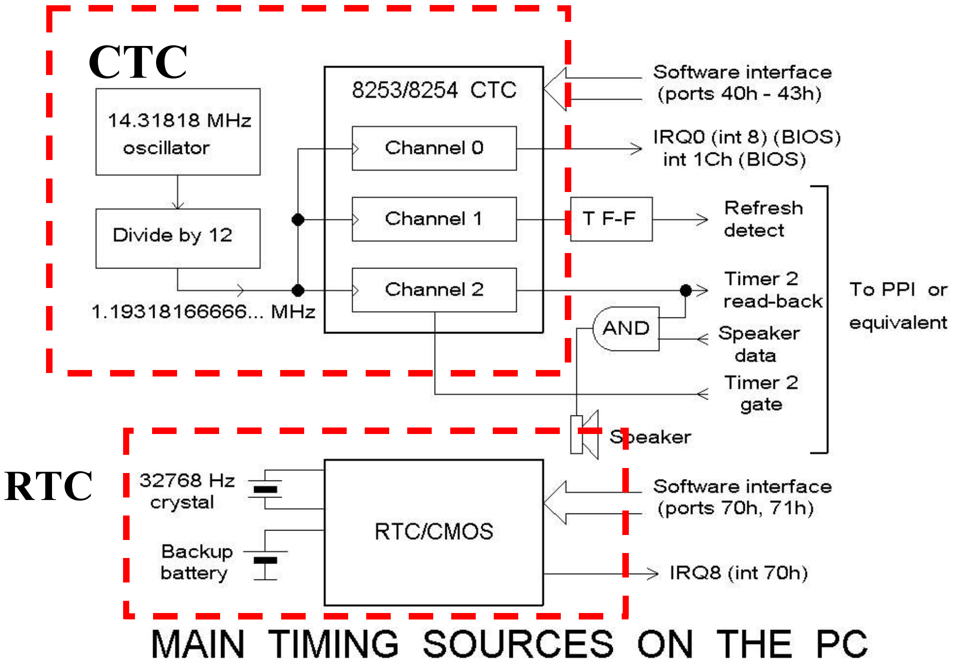

Figure 2 (RTC and CTC timer)

The Basic Input Output System (BIOS) tick count variable is a 32-bit unsigned long word or DWORD, stored at the lower memory address 0040:006C (can also be addressed as 0000:046C), maintained by the BIOS's (*interrupt 8)* handler. It contains the number of timer ticks (units of 54.9254 ms) since midnight, in the current day. The maximum value in this variable is 1800AF hex, so only the bottom 21 bits can ever be nonzero.

The other option for timing is to use the Real Time Clock that is also called RTC/RAM or CMOS. A Motorola MC146818 or compatible contains real-time date and time registers and battery-backed-up storage for BIOS parameters (CMOS). It was introduced with the IBM PC/AT machine, and all hardware compatible ATs and newer machines have one. The RTC is completely independent of the CTC. It uses a 32.768 kHz watch crystal for timekeeping and has a DC battery to maintain its operation when the computer is turned off. It can be used to generate a periodic interrupt, usually at 1024 Hz (1024 interrupts per second).

The third option is to use the processor itself. Beginning with the Pentium® processor, Intel processors (Pentium I 150 MHz and higher) allow the programmer to access a time-stamp counter [4]. The time-stamp counter keeps an accurate count of every cycle that occurs on the processor. The Intel time-stamp counter is a 64-bit MSR (model specific register) that is incremented at every clock cycle, (low 32bits will not overflow for seconds; full 64bits will not overflow for centuries). On reset, the time-stamp counter is set to zero. To access this counter, programmers can use the RDTSC (read time-stamp counter) instruction.

This paper will emphasize on the usage of the Intel time stamp counter as the timing system for an accurate timing system for performance monitoring. The reasons for choosing this time stamp counter as the timer for our system are as follows:

a. Resolution

- Intel time stamp counter supports higher resolution that is only dependent on the processor itself. It provides at least a resolution of 6.6667 ns for a 150 MHz processor compared to the CTC, which only allows a maximum of 0.8381 μs.

b. Simplicity

- Programming a clock system via CTC and RTC needs a lot of interrupt handling processes and the exact address and register of the clock tick storage has to be known.

c. Portability

- Different systems and architectures have different interrupts and register usage. A piece of timing software developed on the experimental system might not be applicable to other systems.

B. Timing Methodologies

There are three basic approaches to timing [2]. Often two approaches may be used simultaneously to provide an accurate measurement.

- ABSOLUTE TIME REFERENCE
  A function may be written for any program that returns a value representing the absolute time, with units and resolution of one tick (54.9254 ms), or 977 μs (the RTC regular interrupt rate), or one CTC clock (0.8381 μs), or one clock cycle (Intel RDTSC).

- RELATIVE TIME REFERENCE
  We may use the CTC to measure short time duration, for example to generate a short pulse on an I/O port pin or measure an external signal.

- REGULAR INTERRUPTS
  An interrupt handler may be called at regular (or sometimes, irregular) intervals, e.g. the default rate of once every 54.9254 ms, or 1024 times per second using the RTC, or at a user-selectable rate. This user selectable rate can be achieved by reprograming the CTC. The interrupt handler can perform operations in the background and/or maintain an absolute time variable.

The Intel time stamp counter can be used to measure the absolute or relative time. It has a very high resolution that depends on the speed of the processor. The details will be explained in later sections.

III. PENTIUM-BASED SYSTEM

A. Intel Pentium Time Stamp Counter

Historically, the time stamp counter has been used for measuring code performance. One such case occurs when a very fine cycle measurement is needed for a section of the code. For example, in the case of optimizing the software run-time, any sets of instructions may be measured to find out the number of cycles it takes. Another use is to get an average time estimate for a function or section of code.

The function call of time stamp counter [4], will return a value in *number of cycles.* For example, one hundred million cycles on a 100 MHz processor is equivalent to one second of real time, while the same number of cycles on a 200 MHz processor is only one-half second of real time. Thus, comparing cycle counts only makes sense on processors of the same speed. Obviously, to compare processors of different speeds, the cycle counts should be converted into time units, where:

Number of seconds = number of cycles / frequency

Since the system time stamp counter returns the number of cycles, high resolution can be obtained. For a 200MHz-speed processor, it will result in an accuracy of 50ns.

B. Pentium-Based Performance Monitoring Software

Using an instruction from the Intel instruction set called RDTSC, Read Time Stamp, a program can be written to tap into the processor counter. To be able to compare measurements between different systems, results obtained must be converted from number of cycles into real time. Hence, the speed of the processor has to be known before hand. The program that had been developed will firstly determine the processor's speed. It will then multiply the number of cycles obtained with the time taken for 1 cycle and gives the real time in seconds. With this calibration, the program can be used on different PCs with different Intel processor chips without the user having to provide the processor speed manually. A flowchart depicting the steps taken in our system is shown in Figure 3.

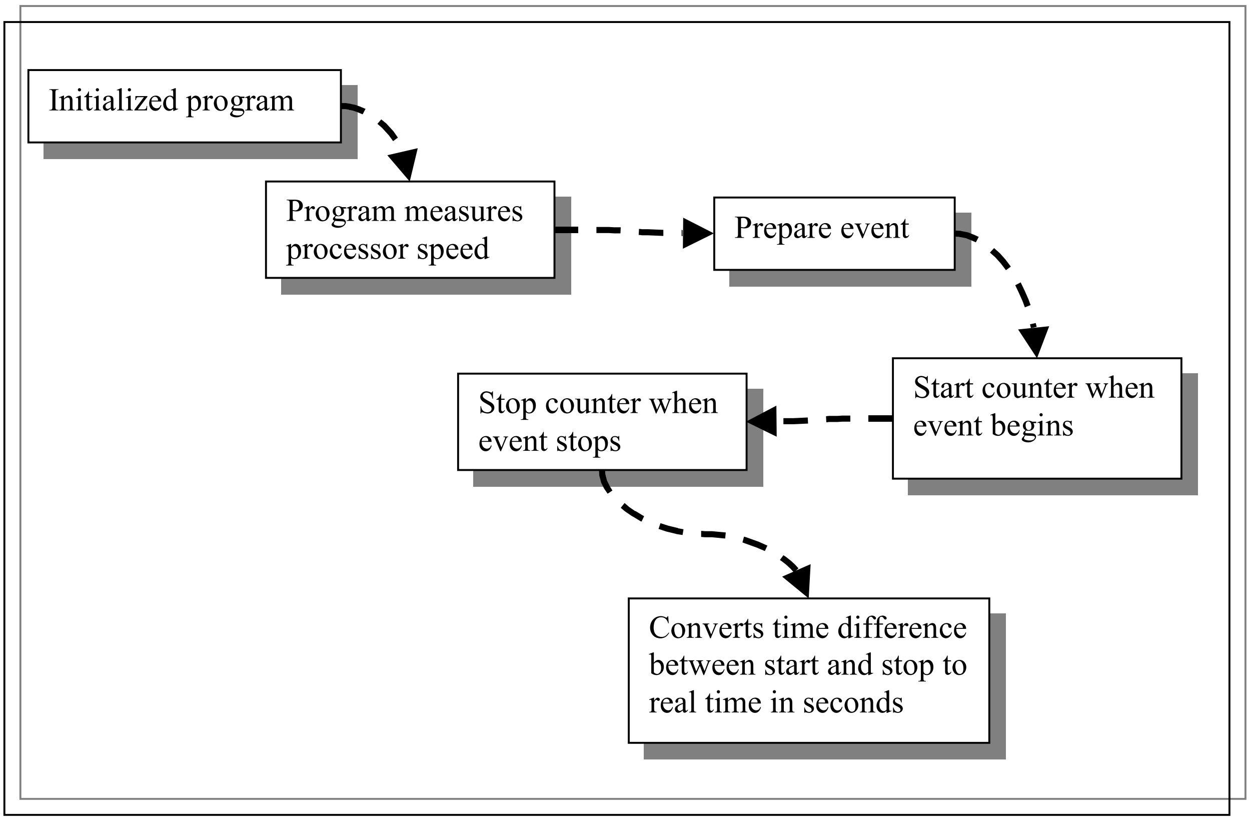
Initialized program
Program measures processor speed
Prepare event
Start counter when event begins
Stop counter when event stops
Converts time difference between start and stop to real time in seconds

Figure 3: Flow Chart for Time Measuring of Two Events

Interrupts, pre-loading cache, inserting and executing instructions that will disrupt cache and buffers have to be considered. These factors may cause the measured

time to vary non-deterministically from the correct time. In the design of the system, we had also considered buffer and cache clearing where experiments had shown some inaccuracies occurring if those factors are neglected.
System interrupts may be immobilized by using the CLI instruction to disable all interrupts. Note that after issuing a CLI, the machine (mouse, keyboard, etc.) will be frozen until STI is issued, or until program flow returns to the operating system, (OS). Ideally, this will give more accurate readings. The enable/disable of interrupts are totally dependent on the nature of the measurements being conducted. For example, if the performances of two different software are being examined, all interrupt can be disabled until the process has complete.

However, if the time measurement concerned is based upon external inputs to signal start and completion, not all interrupts are allowed to be disabled. For example if the timing system is being applied to measuring the time differences between two keystrokes, the keyboard must not be disabled.

Unfortunately, without disabling all interrupts, the results may be inaccurate. The errors can be due to system interrupts (from OS task switching, etc) that happens while the program is running. Then again, the system interrupts will take only dozens of cycle. If a 150Mhz processor is used, a dozen cycles will yield 80ns. Thus, for applications that need accuracy in the range of μs and below, an error in the range of hundreds of nanoseconds is acceptable.

The final version of the program is tested using an external device as a benchmark for the system's accuracy.

IV. TESTING AND VERIFICATION

The accuracy of the timing system developed was verified through an external frequency source. The external source can be either a clock controlled from a high quality crystal in a temperature-controlled environment or something derived from an external clock source (such as frequency generators or radio time signals, etc.), that provides a signal into the input port (parallel, serial port, etc.), which can generate an interrupt.

This approach was chosen mainly because a comparison can be conducted between the software and benchmarked against a highly reliable external frequency source (refer to Figure 4). A 15MHz Function/Arbitrary Waveform *Generator (Hewlett Packard 33120A)* was used as an external signal source. The signal generated was fed into the system via the parallel port. The software was used to measure the time for every cycle and the results will be compare with the frequency generated. The experiment was conducted in room temperature under normal conditions. The speed of the processor used was Intel Pentium 200 MHz, with 64 MB SDRAM running in Windows 95 mode.

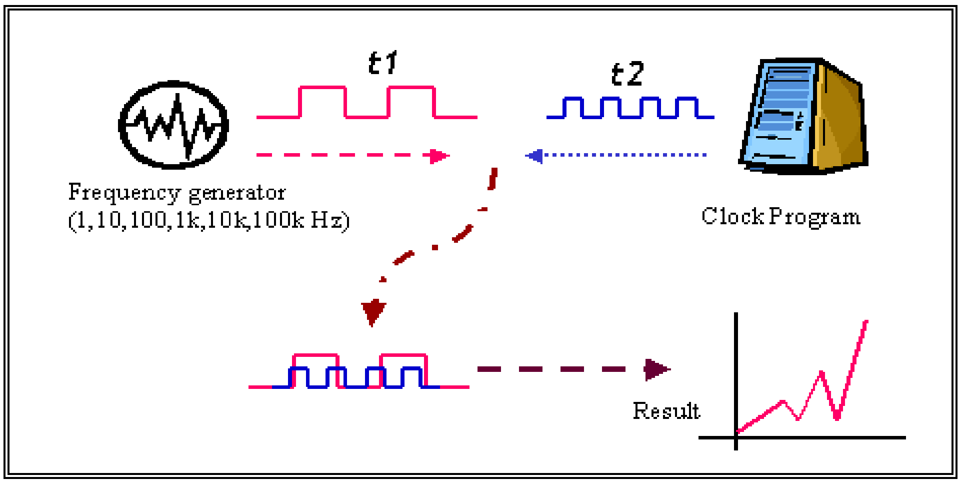

Figure 4: Measuring signal generated by waveform generator

The testing was done with frequencies of 1, 10, 25, 50, 100, 1k, 10k, 100k and 1M Hz. Each frequency testing was evaluated for 1000 cycles that will give a total of 1000 measurements of data. The results obtained are plotted into a histogram with 50 classes against the number of samples. An example of the result is shown in Figure 5(a). As shown, the maximum value obtained was 0.0099994s and the minimum value was 0.0000069s with 50-width range of 0.0099926s. 98% of the data are within 2% deviation from the actual signal of the frequency generator while 2% of the data are in the extreme regions.

Observations have been made regarding the anomalies (data result with difference of 3 orders of magnitude) that occur, at the beginning of the measurement. Thus, the program has been modified to start off with clearing the buffers and cache repeatedly for about a second before timing begins. This decreases the occurrence of anomalies greatly but did not prevent them from occurring at all. This may be seen in Figure 5 (b).

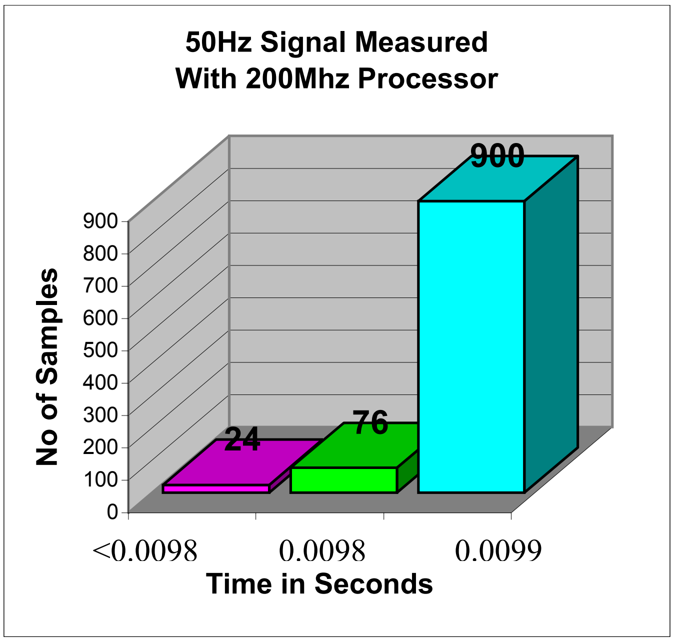

Figure 5 (a). Without buffer, cache clearing with expected value to be 0.010 sec. Max = 0.0099994 sec , Min = 0.0000069 sec

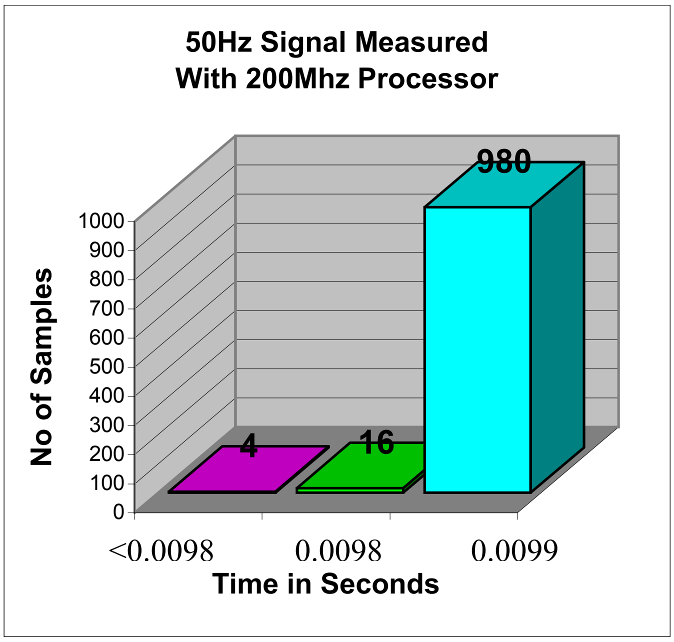

Figure 5 (b). With buffer, cache clearing with expected value of 0.010 sec. Max = 0.0099992 sec , Min = 0.0089895 sec

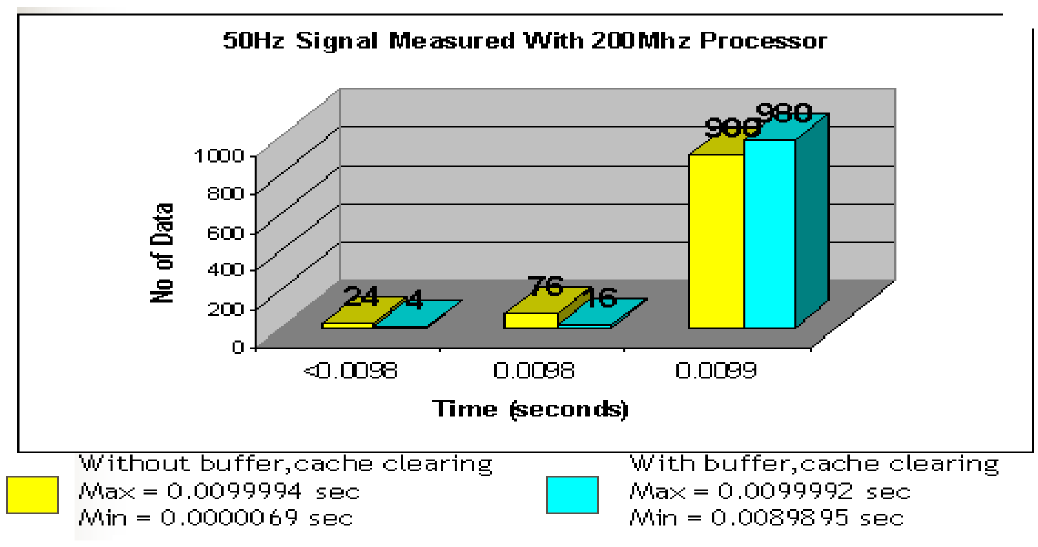

Figure 6: A comparison of results with and without cache clearing

V. SUMMARY AND CONCLUSION

Based on a literature search, we found that there is little research being conducted to prove the accuracy of the PC clock systems. Most applications could either settle for a low resolution (0.1 seconds) or needs an external timing device for extremely high resolution and accuracy [1]. By utilizing the PC clock system described here, certain real-time software performance and measurements can be monitored cost effectively without compromising its accuracy and reliability.

In this investigation, the Intel Pentium Time Stamp Counter has been found to be able to generate timing signals within a $\pm$ 99% accuracy in the range of 50Hz and below. Ideally, the accuracy of the system can be found within the range of nanoseconds or more. Nevertheless, there are still opportunities for fine-tuning and debugging to improve it. Additional testing can be carried out and comparison between systems using the CTC and RTC can be done. Further research can be done to find and eliminate the cause of anomalies that appear in the system. The accuracy in other higher regions of frequencies can also be explored.

In conclusion, we have developed an accurate and reliable software based timing system, which operates in a Windows environment. This is an extremely portable system, which can be used on most Pentium PCs for many real-time performance-monitoring applications.

ACKNOWLEDGEMENTS

We wish to thank Encik Rahadi from Real-Time Lab, MIMOS Berhad for helping us to set up the experiment.

REFERENCES

[1] The Science Museum, "*Time Measurement – Atomic Clocks*", http://www.nmsi.ac.uk/collections/exhiblets/atomclock/before.htm

[2] Kris Heidenstrom, "*FAQ / Application notes: Timing on the PC family under DOS*", http://home.clear.net.nz/pages/kheidens

[3] Kenneth L. Short, "*Microprocessors and Programmed Logic*", Prentice Hall, 1988.

[4] Intel's Developer Site, http://www.developer.intel.com

APPENDIX A

Some normal built in time function supported by C and C++.

1. The ***clock*** function tells how much processor time the calling process has used. The time in seconds is approximated by dividing the clock return value by the value of the ***CLOCKS_PER_SEC*** constant. In other words, clock returns the number of processor timer ticks that have elapsed. A timer tick is approximately equal to ***1/CLOCKS_PER_SEC*** second. In versions of Microsoft C before 6.0, the ***CLOCKS_PER_SEC*** constant was called ***CLK_TCK***.

2. The ***time*** function returns the number of seconds elapsed since midnight (00:00:00), January 1, 1970, coordinated universal time, according to the system clock. The return value is stored in the location given by timer. This parameter may be NULL, in which case the return value is not stored.