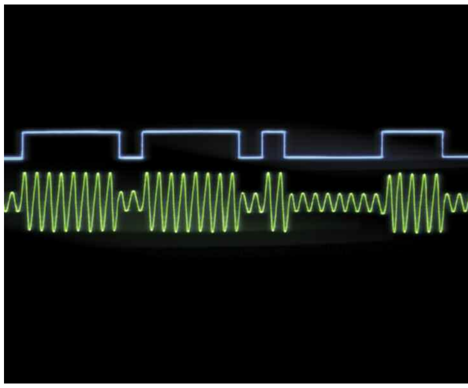
Symmetricom®

Measuring Software Based IEEE 1588/PTP Slave Accuracy



WHITE PAPER

Measuring Software Based IEEE 1588/PTP Slave Accuracy

Abstract

This paper describes how to enable a computer to accurately measure the performance of a local clock steered by an IEEE 1588 PTP software slave or NTP client. It reviews the synchronization problem and presents procedures for test setup, calibration and clock measurement. The results show that clock accuracy may not be as good as one might think, due to factors related to asymmetric path delays and scheduling. Methods to improve clock accuracy using a PTP slave are also discussed.

The Problem With Measuring Computer Clock Time Accuracy

Generally speaking, accurately measuring the time inside a server or workstation is like trying to measure the accuracy of a clock someone sets every day based on the noon church bell. The daily adjustments to the local clock may be very well known. But how long it took for the sound to travel from the bell to the ears of the one setting the clock is usually unknown. Thus, the local clock may be relatively correct (precise), but accuracy (offset in time) from the church bell is unknown due to the propagation delay of the sound waves.

What is needed to determine accuracy is a parallel path of measurement. In the case of the church bell, by measuring the distance from the bell and by knowing the speed of sound, one can compute a clock offset correction. To correctly measure the accuracy of a computer clock, a measurement clock is needed that is placed very close to the computer clock and that is far more accurate than the computer clock. It also cannot be subject to the same errors as the computer clock.

Offset and Path Delay

The difference between the times on two clocks is known as the offset. In timekeeping we strive to keep the offset below a particular value so that we can assign an accuracy value to one of the clocks. Generally in synchronization schemes one clock is more accurate than the other so the offset is relative to the more accurate clock.

The process of setting one clock to another is a matter of computing the offset, say, between a PTP slave clock and a grandmaster clock. Low accuracy applications, like the church bell, simply broadcast out the time from the master and the slaves set their clocks when they receive it. The time the sound takes to travel from the master clock to the slave is called delay, or path delay.

Symmetric and Asymmetric Delay

There are two main sources of error in transferring time from one clock to another: time transfer delay and the errors associated with eliminating it. In PTP packet based networks, timing packets are exchanged between the grandmaster and the slave for the purpose of computing the time offset. If the packet exchanges were instantaneous then there would be no delay and the offset could be computed perfectly. Also, if the packet exchange delay on the master-to-slave path and slave-to-master paths were identical, the offset could again be easily and precisely computed since the delays would cancel each other mathematically. The case where path delay is the same both ways between master and slave is called symmetric delay and time transfer over packet networks most often assumes symmetric delay.

Unfortunately, however, path delay is different between master-to-slave and slave-to-master and this case is called asymmetric delay. Furthermore, not only is delay different along the two paths, but the difference also varies. See Figure 1.

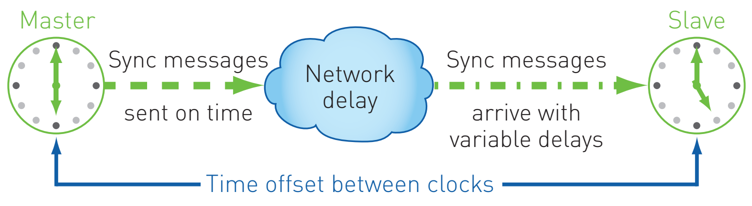


FIG 1 Timing messages sent from the master to the slave (and vice versa) experience variable delays caused by the switches in the network and application delays leading to timing errors at the slave clock.

In a simplified model, asymmetric delay adds a time offset correction error equal to one half the difference in delay times. For example, if it took one second for the timing packet to go from the slave to the master and three seconds on the reverse path, the offset computed by the slave would be off by one second. Fast forward to computer hardware that is time stamping packets that are being exchanged exactly every second between a master and a slave with the master having 50 nanosecond timestamp accuracy. These timing packets transit a LAN containing switches (or worse, routers) that may add asymmetric path delay in the 10s to 100s of microseconds, which result in similar size time offset computation errors.

The Crux of The Timing Problem

PTP software slaves time stamp PTP event packets at the application layer of the server or workstation operating system. The local clock offset is computed and the clock corrected. If millisecond time accuracy is all that is desired, the work is likely done. Unfortunately, in high-speed low latency networks, microseconds matter. And there are two key problems impeding microsecond-level synchronization: 1) the asymmetric path delays contributed by the intervening network between the PTP grandmaster and the PTP slave; and 2) the operating system, network and application delays inside the computer.

A software based PTP slave program computes an offset and adjusts the local clock. Often the size of the clock adjustment is mistaken for the clock accuracy, i.e., how far off the clock was before it was corrected back to "perfect." In fact, nothing could be further from the truth. The unknown asymmetric path delays add uncertainty in the time offset calculations. This is especially true if there is a constant delay present on one path that is not present on the other.

Measuring a Computer Clock

There are, however, clever ways to measure a local computer clock that mitigate both asymmetric network delays and inside-the-computer delays. "Measure" in this case means gauging the PTP software slave's ability to correct the local clock.

Suppose, for example, there is a single PTP grandmaster source of time. The PTP slave synchronizes the local computer clock to the grandmaster over the network. To create the "parallel path" as mentioned above, a second very accurate hardware based clock card is installed inside the computer and will also be synchronized to the grandmaster. This second more accurate clock card will be used to measure the PTP steered local clock. In this application the grandmaster can also measure the accuracy of the clock card installed inside the computer. By doing this the clock card becomes a known, accurate measurement platform from which to measure the PTP steered local clock.

The Test Setup

Figure 2 shows the test setup. A GPS referenced SyncServer S350 clock is the grandmaster. This clock is accurate to 50 nanoseconds to UTC. The PTP software slave synchronizes over the network to the grandmaster and steers the local computer clock (blue path). In parallel (green path) is a high accuracy Symmetricom bc635PCIe clock card inserted into a PCIe slot in the computer and also synchronized to the grandmaster.

In this case, the bc635PCIe card is connected via coaxial cables and synchronizes to a time code signal output by the S350. A very small program is run on the computer that interfaces to the bc635PCIe clock card, measures the local clock and saves the results. This local clock measurement is made once per second over a long period of time to obtain a good characterization of the PTP software slave's ability to steer the local clock.

High Accuracy Testing of PTP or NTP Software Based Clock Clients using PTP Grandmaster and Clock Card

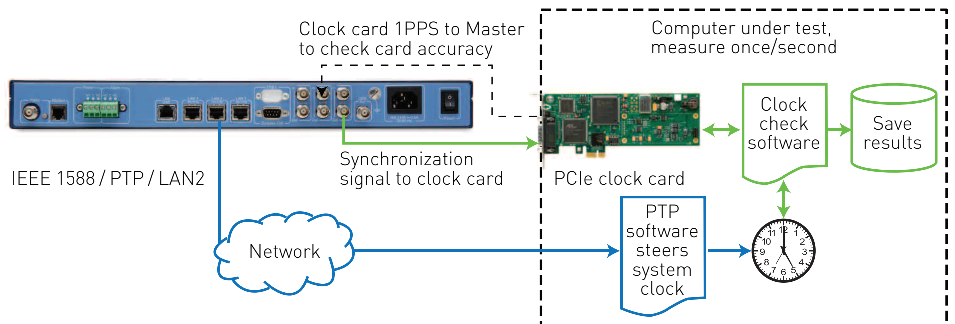

Clock card 1PPS to Master to check card accuracy
Computer under test, measure once/second
Clock check software
Save results
IEEE 1588 / PTP / LAN2
Synchronization signal to clock card
PCIe clock card
PTP software steers system clock
Network

FIG 2 The Test Setup

Verifying Measurement Accuracy

Most modern, high accuracy clocks output a once per second pulse called a 1PPS. This pulse is created when the fractional seconds of the time in the clock are all zeros, otherwise known as the "top of the second." To compare two clocks you compare their respective 1PPS signals. A counter, oscilloscope or in this case a time interval measurement built into the S350 SyncServer is used to measure the accuracy of the PCIe clock card. The simple setup involves the 1PPS output of the bc635PCIe card connected to the 1PPS input of the S350 via coaxial cable. In this demonstration the bc635PCIe clock card synchronized to the master to less than 380 nanoseconds, with an average offset of 149 nanoseconds. See Chart 1 below.

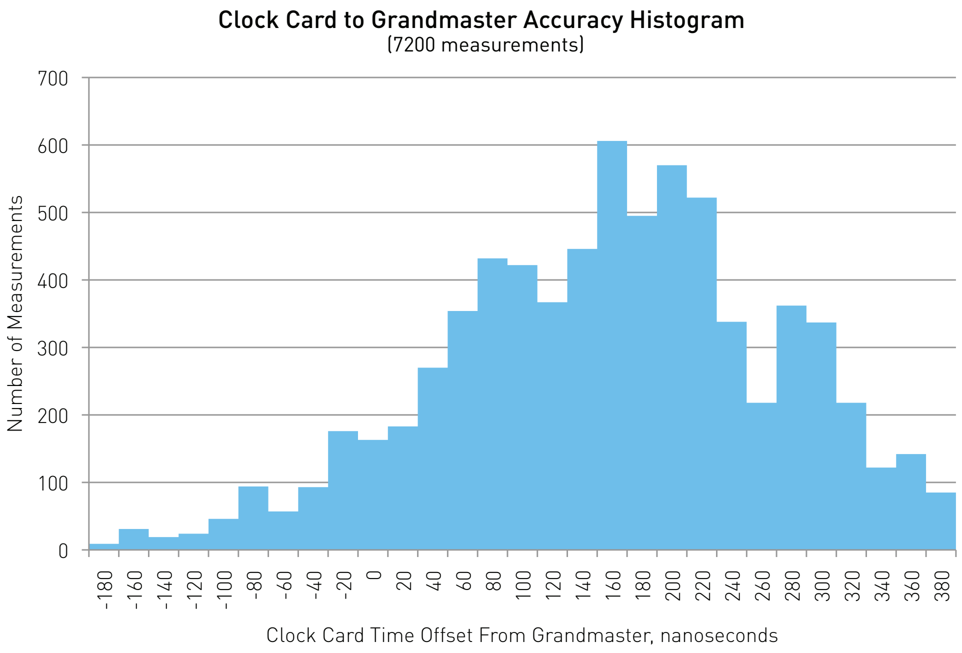


CHART 1 Clock Card Time Offset from Grandmaster Histogram

Measuring the Local Computer Clock

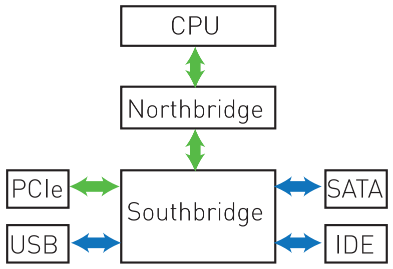


FIG 3 Location of the PCIe card/bus relative to the CPU running software applications

Figure 3 shows a simplified view of part of computer architecture between the clock card and the CPU (green path), showing the Northbridge and Southbridge interconnects. The bc635PCIe card connects to the PCIe bus in the Southbridge. The Southbridge connects to the CPU via the Northbridge. The PTP slave software and the simple clock check software (CCS)[1] run in the CPU. (The fastest, most modern server architectures configure these components differently, but the fundamentals are the same).

Once per second, at the top of the second, the bc635PCIe card places an interrupt on the PCIe bus to the CPU. In response to the interrupt the CCS program reads the local clock. Next, the CCS reads the time from the bc635PCIe card. The two values are then saved to be compared later and the measurement repeats at the top of the next second. Ideally, if the local clock is perfectly synchronized to the S350 (or at least to within 380 nanoseconds), the local clock time value read by the CCS program will be before the bc635PCIe clock time value based on the order of the measurements (interrupt→ local clock read→ clock card read).

PTP Slave Clock Steering Test Results

Charts 2 and 3 show the effectiveness of the PTP software slave steering the local clock relative to the S350 grandmaster. These results are not meant to show how well the software PTP slave can steer a local clock, but rather to show with some degree of certainty the accuracy of the local clock relative to the grandmaster over time.

[1] Source code listing in the appendix

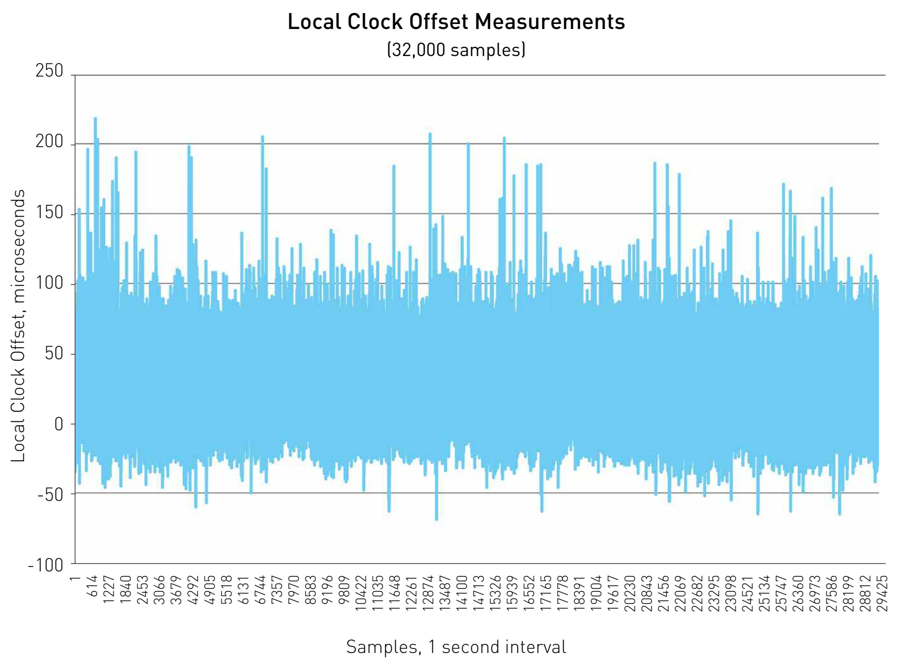
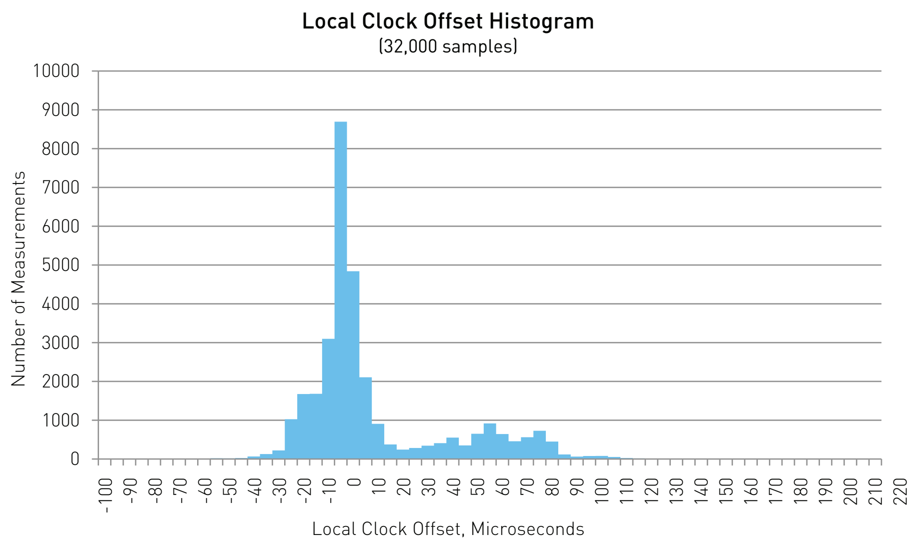


CHART 2 Local Clock Offset from Grandmaster



Measurement Set Statistics	Microseconds
Maximum Offset	218
Minimum Offset	-69
Average Offset	7
Standard Deviation	33

TABLE 1 Clock Offset Measurement Set Statistics

CHART 3 Local Clock Time Offset Histogram

The PTP software slave in this example is steering the local clock to better than 200 microseconds. Most of the time it is within 40 microseconds. Clearly the clock is moving around a mean and there is likely some application preemption that is occasionally delaying the reading of the time by the clock check software. This is explored further in the next section.

Measurement Error

Consider the order of the Clock Check Software program loop in Figure 4 (actual source code is listed in the appendix).

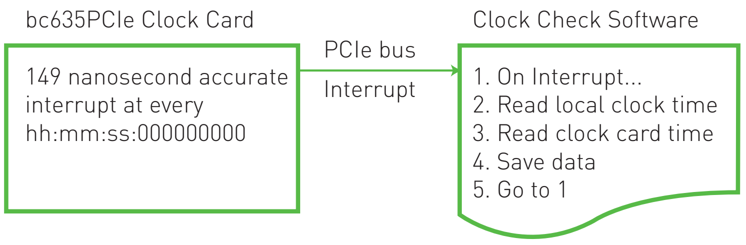


FIG 4 Clock Check Software Loop

The interrupt is initiated on average within about 149 nanoseconds of the same exact time on the S350 grandmaster. (The same clock the PTP slave is synchronizing to). The CCS program responds to the interrupt, reads the local PTP slave steered clock and then reads the time on the bc635PCIe clock card. Ideally if the local clock time is accurate the time read in step 2 (reading the local clock) should be before the time is read from the bc635PCIe card in step 3.

Since the interrupt is at the top of the second, the time read from the bc635PCIe card is the duration of the entire clock check process. The value of this duration represents the worst-case measurement error of the local clock being steered by the PTP slave. Charts 4 and 5 show the measurement duration for the data set.

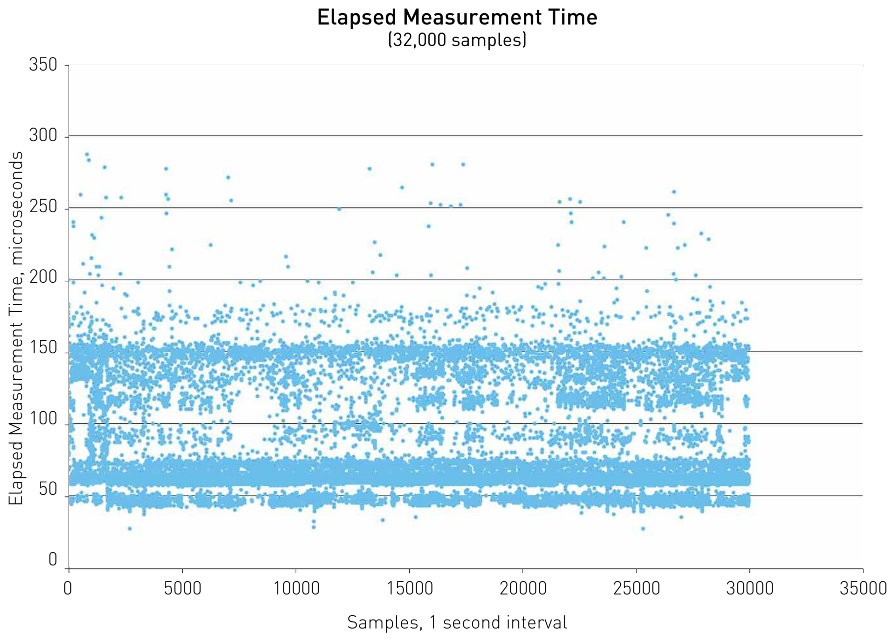


CHART 4 Elapsed Measurement Duration, microseconds

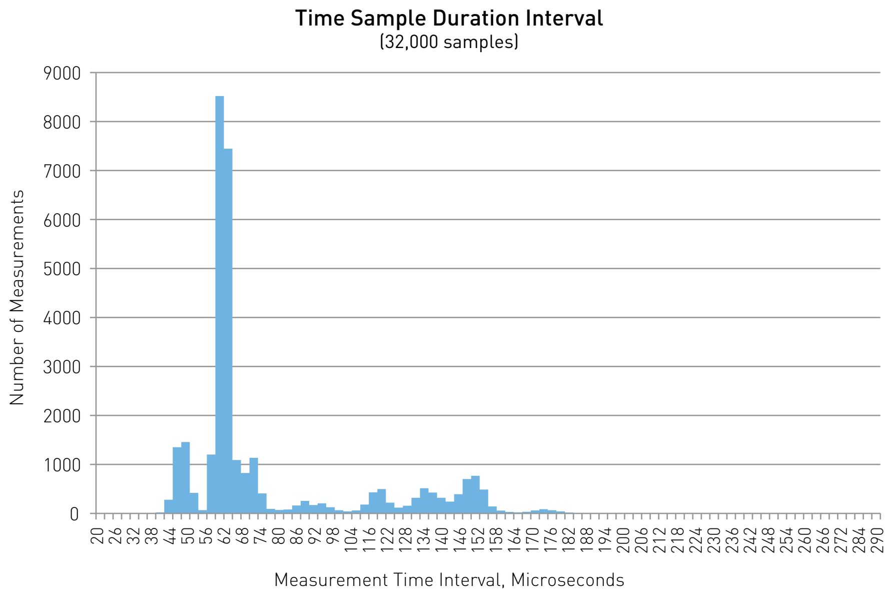


CHART 5 Measurement Duration Interval Histogram, microseconds.

Since it is not known exactly where in the measurement interval that the CCS program read the local clock, there is uncertainty up to the duration of the whole measurement process.

Table 2 shows the duration statistics of the clock measurement process.

Measurement Duration Statistics	Microseconds
Maximum Duration	287
Minimum Duration	27
Average Duration	78
Standard Deviation	34

TABLE 2 Measurement Duration Statistics

Generally speaking, two-thirds of all measurements were made in less than 112 microseconds. This value is added to the offset from the bc635PCIe card to the master. Thus, the measurement interval would generally be between 0.149 and 112 microseconds from the SyncServer grandmaster. From other tests performed retrieving time from the bc635PCIe clock card and reading time from the system clock, it is known that the time read of the local clock likely happens in the first 30 microseconds of the measurement interval.[2]

This test case intentionally represents a very basic clock check example. Preemption and scheduling are two very apparent sources of error in the measurements. Consider, however, the outliers in Charts 4 and 5 as well as the apparent uniformity in delay intervals of the elapsed measurement times. Clearly there are processes at work that are adding uniform delays as well as spurious delays. These factors are beyond the scope of this paper.

Improving PTP Software Slave Clock Steering Results

Improving synchronization accuracy focuses on one primary area, reduction of asymmetric path delays. The closer the path delays match, the more accurately the clocks can be synchronized. The first step is the reduction in the number of elements that introduce delay. These would be switches and routers. The fewer the hops and the faster the switches, the smaller the path delay asymmetries will be.

Another solution is to use a faster computer running the PTP slave software. This reduces preemptive delay duration and shortens the overall delay of many of the internal processes. Additional techniques like processor affinity for the PTP software slave can also improve local clock steering accuracy. Adjusting the PTP software slave priority may also be worth exploring.

PTP Sync and Delay_Request rates can also be adjusted upwards in combination with specialized algorithms to filter out packets that arrive over the network with excessive delay. This involves a good deal of expertise in packet filtering and clock modeling, something that Symmetricom does particularly well in its PTP slave implementations.

[2] Using the time stamp counter (TSC) as a relative timescale, reading the local clock is about 3x speed of reading the time over the 32 bit bus to the clock card.

Conclusion

The objective of this test was to demonstrate a technique of how to measure a local clock being steered by a PTP software slave. This same test technique is also applicable to a clock steered by NTP. The test results reveal not only the general sync effectiveness range of the PTP slave software, but also issues surrounding the transfer of precise time. Time transfer asymmetries, process preemption and scheduling are all apparent factors that result in time synchronization errors. All this of course is relative to the desired clock accuracy at the slave computer. After conducting a test like this one might find the accuracy is perfectly fine, or not. The beauty of such a test is that it forms the basis for further testing and refinements in improving clock synchronization accuracy.

Appendix: Test Setup

1. PTP Grandmaster: Symmetricom SyncServer S350 referenced to GPS; IEEE 1588 Option enabled; configured for the IEEE-1588 Annex J, Default Profile; Sync rate 4 Sync messages per second. 1PPS from bc635PCIe card measured against 1PPS of master using Time Interval Measurement function in the S350 (part of the PTP Option for the S350).

2. Network: Netgear DS108 hub.

3. Slave Computer: CPU Intel i7 @ 2.8 GHz, Quad-core, 4 GB Ram

4. Slave Software: Sourceforge PTPd IEEE 1588 v2 daemon, configured for 4 delay_requests per second.

5. Bus Card: Symmetricom b635PCIe PCI Express Time & Frequency Processor; synchronized via IEEE 1344 DCLS time code via coaxial cable. Symmetricom freeware SDK/driver for Linux for bc635PCIe.

6. Clock check software: custom written, see below for source code.

Appendix: Computer Clock Check Program

```
// Filename: ppsTime.c

#include "bcuser.h"
#include <stdio.h>
#include <sys/time.h>
#include <signal.h>

struct timeval timeP;
BC_PCI_HANDLE hBC_PCI;
DWORD major;
DWORD minor;
WORD nano;
BYTE stat2;
FILE * outFile;

void intHandler(int sig){
 signal(sig, SIG_IGN);
 printf("Stopping\n");
 fclose(outFile);
 bcStopPci(hBC_PCI);
 exit(0);
}

void bcIntHandlerRoutine(BC_PCI_HANDLE hBC_PCI, DWORD dwSource){
 gettimeofday(&timeP, NULL);
 bcReadBinTimeEx(hBC_PCI, &major, &minor, &nano, &stat2);
 fprintf(outFile,"%lu,.%06u,.%06u%03u\n", timeP.tv_sec, timeP.tv_usec,
minor, nano);
 fflush(outFile);
}

int main(int argc, char* argv[]){
 if (argc < 2){
  printf("Usage: %s fileName [runtime(seconds)]\n", argv[0]);
  return 1;
 }
 char * filename = argv[1];
 int runTime = 0;
 if (argc > 2)
  runTime = atoi(argv[2]);
 hBC_PCI = bcStartPci();
 if (!hBC_PCI){
  printf("Error Opening Device Driver\n");
  return 1;
 }
 printf("Device Open\n");
 signal(SIGINT, intHandler);

 gettimeofday(&timeP, NULL);
 bcReadBinTimeEx(hBC_PCI, &major, &minor, &nano, &stat2);

 printf("Setting input to timecode 1344 DCLS\n");
 bcSetMode(hBC_PCI, MODE_IRIG);
 bcSetTcInEx(hBC_PCI, TCODE_IEEE, TCODE_IRIG_SUBTYPE_NONE);

 printf("Opening output file: %s\n", filename);
 if (!(outFile = fopen(filename, "w"))){
  printf("Unable to open output file\n");
  return 1;
 }
 fprintf(outFile, "Sys Major,Sys Minor,IRIG Minor\n");

 bcStartIntEx(hBC_PCI, bcIntHandlerRoutine, INTERRUPT_1PPS);

 bcSetInt(hBC_PCI, INTERRUPT_1PPS);
 if (argc >2){
  printf("Running for %d seconds\n", runTime);
  sleep(runTime);
 }
 else{
  printf("Running until Ctrl-C\n");
  while (1);
 }

 fclose(outFile);
 bcStopPci(hBC_PCI);
 return 0;
}
```

End Notes

Time Interval is the elapsed time between two events. In time and frequency metrology, time interval is usually measured in small fractions of a second, such as milliseconds, microseconds, or nanoseconds.[1]. A time interval measurement is a measurement of the elapsed time between some designated START phenomena and a later STOP phenomena. [2]

In this application, the Symmetricom SyncServer S350 is able to perform a time interval measurement. The START time is the on-time internal 1PPS of the S350 referenced to GPS and was accurate to 50 nanoseconds to UTC. The STOP phenomenon was the 1PPS received from the PCIe slave. The 1PPS from each clock is at the "top of the second" (no fractional seconds) and is considered the on-time marker for the clock and useful for time offset calculations. Since the slave is referenced to the master, the time interval measurement is an indication of how accurately the slave can be synchronized to the master over the network. These measurements are made every second with a statistical analysis performed on a collected data set.

[1] Time & Frequency Division website (tf.nist.gov)

[2] Fundamentals of Time Interval Measurements, Application Note 200-3, Hewlett Packard

SYMMETRICOM, INC.
2300 Orchard Parkway
San Jose, California
95131-1017
tel: 408.433.0910
fax: 408.428.7896
info@symmetricom.com
www.symmetricom.com