Constellation Program Electrical Ground Support Equipment Research and Development

Keegan S. McCoy[1]
The Pennsylvania State University, University Park, PA, 16802

At the Kennedy Space Center, I engaged in the research and development of electrical ground support equipment for NASA's Constellation Program. Timing characteristics play a crucial role in ground support communications. Latency and jitter are two problems that must be understood so that communications are timely and consistent within the Kennedy Ground Control System (KGCS). I conducted latency and jitter tests using Allen-Bradley programmable logic controllers (PLCs) so that these two intrinsic network properties can be reduced. Time stamping and clock synchronization also play significant roles in launch processing and operations. Using RSLogix 5000 project files and Wireshark network protocol analyzing software, I verified master/slave PLC Ethernet module clock synchronization, master/slave IEEE 1588 communications, and time stamping capabilities. All of the timing and synchronization test results are useful in assessing the current KGCS operational level and determining improvements for the future.

I. Introduction

This spring semester, I interned in the Electrical Design Branch at the Kennedy Space Center (KSC) through the Undergraduate Student Research Program (USRP). Working in the Controls Lab of the Engineering Development Lab, my individual project was focused on the research and development of electrical ground support equipment for NASA's Constellation Program. Particularly important for ground support equipment are timing characteristics. The tests that I performed measured the timeliness and consistency of programmable logic controller (PLC) communications, verified test procedures, and demonstrated the ability to time stamp data packets and synchronize the PLC network. All of these tasks are crucial to the successful operation of ground support equipment that aids in launch processing and operations.

My first task was to understand the role of PLCs in command and control applications, since KSC employs a programmable control system to operate the thousands of valves, switches, regulators, and sensors on the launch tower necessary for launch operations.

Programmable control systems were first developed as an alternative to relay systems. Programmable control systems hold many advantages over electromechanical relays, including enabling easier access for maintenance personnel, as well as only requiring edits to a project file for system changes rather than rewiring. Furthermore, programmable control system solid-state parts are capable of surviving harsh, industrial environments, their component lifetimes are long, and the components are modular and easy to replace.[1]

Information in a programmable control system flows in the following order:

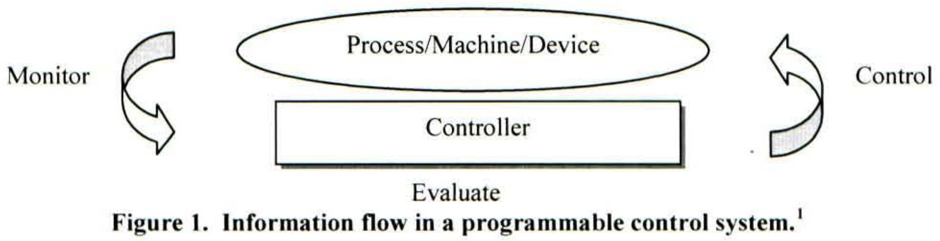


Figure 1. Information flow in a programmable control system.[1]

1. The system monitors input information from a process, machine, or device (e.g., a sensor).

[1] Student, Department of Electrical Engineering and Department of Astronomy and Astrophysics, The Pennsylvania State University. Student Member AIAA.

2. A controller evaluates the input information based on a given set of rules and then generates output information (e.g., shut a valve if the sensor detects a certain pressure).
3. The output information is used to control the process, machine, or device (e.g., the valve closes).

II. Programmable Control System Components

All programmable control systems consist of four main components:
- Programming system
- Communications network
- Controller
- I/O (input/output) system

A. Controller

A controller is a solid-state device with user-programmable memory and a central processor, serving as the brain of a programmable control system.[1] Controllers perform the following functions:
- I/O control
- Logic
- Timing
- Report generation
- Communications
- Data manipulation

For the Constellation Program's KGCS, NASA is using Allen-Bradley ControlLogix controllers produced by Rockwell Automation.

B. Input/Output (I/O) System

An I/O system consists of the following components:[1]
- I/O modules, which are part of the programmable control system
- I/O devices, which are part of the process/machine

An I/O system sends input and output information between a controller and a process/machine. An input device, such as a pressure transducer on the launch pad, supplies signals through an input module to a controller. An output device (e.g., a valve, switch, regulator, sensor, etc.) is actuated or energized by a controller through an output module.

NASA is utilizing a wide array of Allen-Bradley I/O modules, ranging from analog and digital modules used to collect and send information, to motion control modules capable of controlling the movement of end items. These I/O modules are modular, allowing for easy addition or removal from the PLC chassis.

C. Programming System

Composed of a programming device (e.g., personal computer, laptop) and software, a programming system is used to program and monitor the operation of a controller.[1]

RSLogix 5000, a software program developed by Rockwell Automation, allows NASA to write code in project files to control the vast PLC network. The program has a flexible instruction set, enabling users to write code in ladder logic, structured text, function block diagrams, or sequential function charts, depending on the best fit for the specific application.

D. Communications Network

A communications network is the physical connection between a series of components or devices. This connection is used to transfer data between the components, such as a computer in the Launch Control Center (LCC) and a controller, using a cable system.[1]

The KGCS employs a few different communications networks. The ControlNet network is a real-time control network that provides high-speed transport of both time-critical I/O and messaging data, including uploading and downloading of programming and configuration data. The ControlNet network is highly deterministic and repeatable, remaining unaffected as devices are connected or disconnected from the network. This robust quality results in dependable, synchronized, and coordinated real-time performance. ControlNet connects remote I/O (RIO) modules contained in separate chassis with the main controller contained in a local chassis, allowing for scheduling of all module communications.

Several timing values play significant roles in ControlNet communications. The Requested Packet Interval (RPI) specifies the period at which an I/O module updates to the backplane. Typically, RPIs fall in the millisecond range, from 0.2 ms to 750 ms. The RPI reserves a slot in the stream of data flowing across the ControlNet network, as all I/O modules have individual RPIs. Although the timing of the slot may not coincide with the exact value of the RPI, the control system guarantees that the data transfers at least as often as the RPI.

Analog input modules have a real-time sampling (RTS) value that specifies the module's sampling interval of the particular data being collected by the connected I/O device. The RTS value must always be greater than or equal to the RPI. If the module's RTS is lower than its RPI, samples will be lost, since the backplane is not being updated quickly enough to keep pace with the sample rate.

The Network Update Time (NUT) determines the time interval at which data from the I/O modules is collected and sent to the controller through the backplane. The NUT is set using RSNetworx, an add-on program to RSLogix 5000 that allows network scheduling and bandwidth calculations. In general, all module RPIs must be greater than or equal to the NUT or information will be lost from the individual module updates. The ideal choice is often to set the module RPIs equal to the NUT, which not only prevents losing information, but also redundant values.

Another communications network supported by the Allen-Bradley PLCs is EtherNet/IP, which is carried on an RJ-45 cable. The main functions of the EtherNet/IP network are to send information back and forth from the LCC Gateway and to enable master/slave clock synchronization. The EtherNet/IP network is unscheduled, allowing for information to be sent or received as needed.

III. Kennedy Ground Control System

A. Operational Description

The Kennedy Ground Control System (KGCS) is one of the key components of the Constellation Program's Launch Control System (LCS) at KSC. The KGCS will provide ground controls and interfaces to allow monitoring and control capability of Ground Support Equipment (GSE), thereby facilitating launch processing. The GSE will include field devices such as sensors, valves, heaters, and motors necessary for subsystem processes. The system will further facilitate testing and maintenance of the GSE end items, as well as providing fault tolerance, safety, and health management. The KGCS architecture utilizes PLCs for local subsystem control, while local and RIO modules provide direct connectivity to GSE end items. I/O points are addressed using tag names defined through RSLogix. Figure 2 provides a broken down view of the KGCS, with the LCC, a local controller, and end items. The LCS Gateway computer sends commands through Ethernet cable and an LCC switch out to local controllers. The local controllers are connected to RIO chassis on the launch pad with ControlNet busses. The individual end items are wired to input and output modules in the RIO chassis.

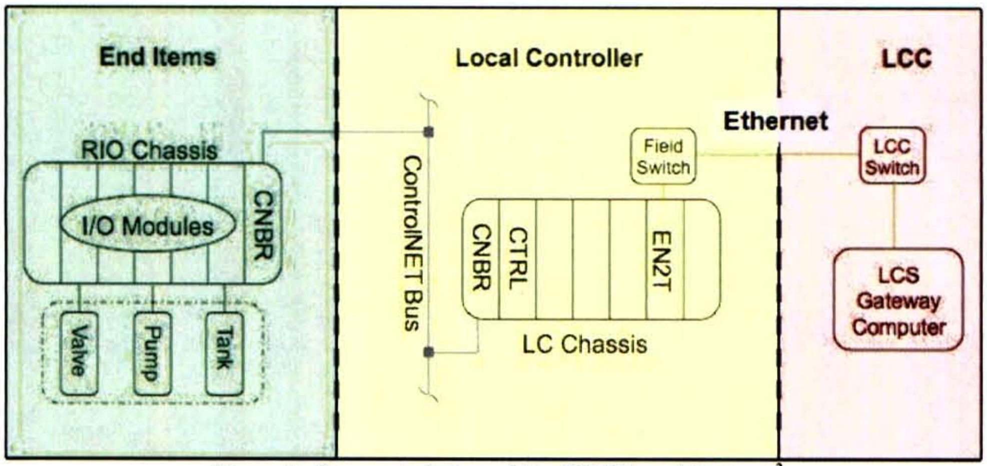


Figure 2. Segmented view of the KGCS architecture[2]

Figure 3 displays a block diagram of the KGCS, which primarily consists of the Command and Control Network, the Health Management Network, and the Emergency Safing Network.

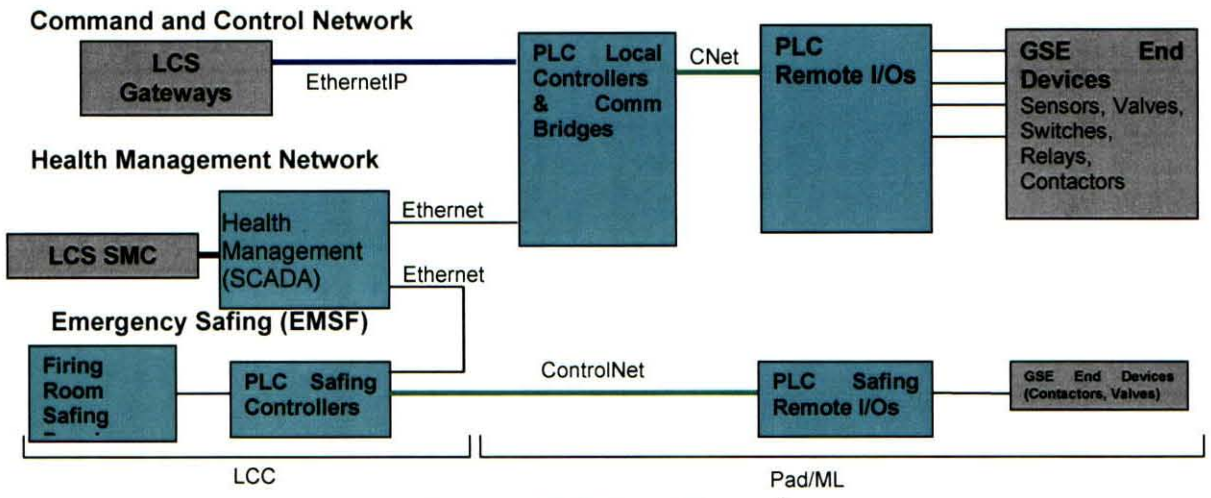

Command and Control Network
LCS Gateways
EthernetIP
PLC Local Controllers & Comm Bridges
CNet
PLC Remote I/Os
GSE End Devices Sensors, Valves, Switches, Relays, Contactors
Health Management Network
LCS SMC
Health Management (SCADA)
Ethernet
Ethernet
Emergency Safing (EMSF)
Firing Room Safing
PLC Safing Controllers
ControlNet
PLC Safing Remote I/Os
GSE End Devices (Contactors, Valves)
LCC
Pad/ML

Figure 3. KGCS Block Diagram[3]

B. Command and Control Network

The KGCS Command and Control Network (CCN) provides the real-time monitoring and control of GSE. It provides the direct interface to the end items for subsystems and facilitates local and remote monitoring and control by the users through a local human-machine interface (HMI) or through the LCS user interface. The RIO performs the signal conditioning and digitization of signals, which are then sent to the local controller for processing and further transmission to the LCS Gateway. The local controller also has the capability to perform closed-loop control and reactive-control logic for time-critical responses to the end items.[3] The CCN provides the critical data from field instrumentation and control devices to other LCS components, such as application servers and record and playback recorders. These data contain critical information that is used to measure the subsystem process health and control subsystem process.

C. Health Management Network

The KGCS will utilize a Health Management Network (HMN) that provides system monitoring and supervisory control for subsystem maintenance, troubleshooting, and configuration. This will provide health status of the PLCs and other KGCS components, such as sensors and transducers, to the GSE engineers who will be responsible in maintaining the field hardware. The HMN will also assist engineers in the initial verification of the GSE, troubleshooting, and maintenance. It contributes to a more efficient process without requiring the LCS control room to be running when performing a quick instrumentation snapshot. The HMN will interface with historical servers and provide automatic reporting and trending of process data. The HMN will also interface to the LCS System Management Console (SMC) for configuration management of PLC application software and IT security. Software that will be archived and will follow configuration management includes application programs, PanelView software, and PLC firmware. The HMN receives application programs and PLC firmware from the SMC to update the PLCs on the field.

D. Emergency Safing Network

The Emergency Safing Network (EMSF) will provide safing capabilities of end items in the event of an emergency independent from the Command and Control Network. EMSF sequences are manually initiated by the system engineers on the console from a dedicated emergency safing panel. The EMSF shall override all other controls and exert final control of the equipment when activated. No other systems shall be capable of overriding the EMSF. The EMSF employs redundancy for PLC devices, RIO, instrumentation whenever feasible, communications, and power distribution.[3] EMSF will be independent of the Command and Control Network and is an integral part of the KGCS.

IV. KGCS Testing

A. Latency and Jitter

My first hands-on task involved conducting tests measuring the latency and jitter of PLC communications. In general, latency is the amount of time it takes for a data packet to travel end-to-end through a network.[4] Several factors may contribute to a network's latency, including propagation, serialization, queue, and processing delays. Propagation delay is inherent in any network due to the finite nature of the speed of light in the transmission of electrical or optical signals. For example, the LCC is located about 3.4 miles (5.47 km) away from Launch Complex 39, where the Constellation Program's Ares I and V rockets will launch. Therefore, not taking into account propagation delay caused by the transmitting material itself, there will be an intrinsic speed of light delay each way of around 18.25 μs. Serialization delay refers to the time it takes a network device to encode and transmit a data packet, which is a function of packet size and transmission rate. Queuing delay measures the time that a data packet must wait in a buffer before it is sent.[4] This may become significant if data packets are sent to the transmitting device at a higher rate than the device is capable of transmitting packets. Processing delay is the time that it takes to handle data packets within a network node (e.g., a controller), which is dependent upon the speed of the device and the congestion in the network. In the past, processing delay has often been neglected; however, in some systems the processing delay can be quite significant, especially where routers are performing complex encryption algorithms and examining or modifying packet content.

Due to varying internal and external factors, latency may change in a network with time. Jitter is a measure of the resulting variation in latency. Jitter may be caused by a change in data packet size, variance in queue delay due to receiving packets from multiple sources, or data packets traveling over different paths to arrive at the same destination.

For the purposes of the KGCS, latency measures the delay between when a command is sent from the LCC to the time at which the end item (valve, switch, etc.) physically reacts. Jitter in the KGCS is caused by a variety of factors, including individual module RTS and RPI values, as well as the NUT.

I performed simple measurements of PLC latency and jitter in the Controls Lab, not taking into account the full propagation delay from the LCC to Launch Complex 39. To carry out the tests, I set up two chassis. The upper, local chassis contained a Logix 5563 controller, an EN2T Ethernet module, and a ControlNet module. The lower, remote chassis consisted of an EN2T Ethernet module, a ControlNet module, an IB16 discrete input module, and an OB8 discrete output module. The input module was connected to a function generator, which was meant to imitate an incoming signal from an end item, while the output module was hooked up to an oscilloscope to view the PLC's input and output signals. By measuring the time delay of the input and output signals over a number of samples, latency and jitter could be measured for each test. Figure 4 shows the test configuration with the local chassis, remote chassis, function generator, oscilloscope, and power rail.

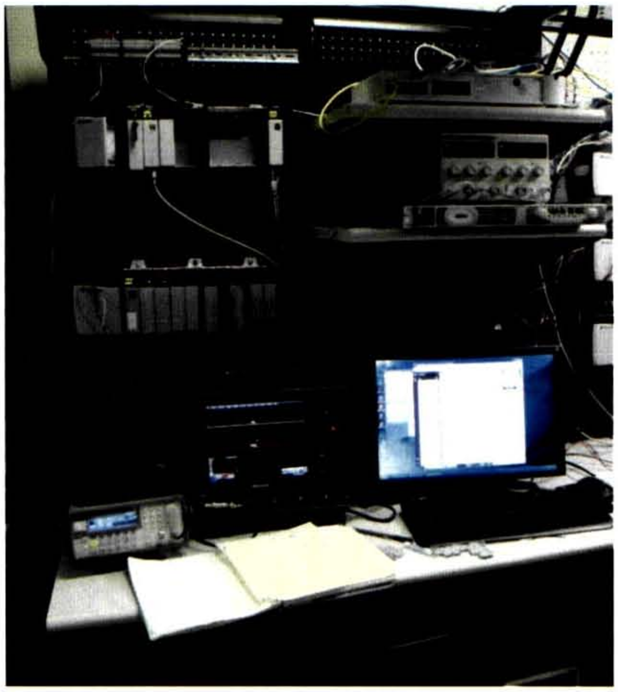


Figure 4. Latency and jitter test configuration.

For each test run, I set the function generator to input a 1 Hz, 20-V_{pp} square wave (0-V DC offset), which was viewed on Channel 1 of the oscilloscope. The resulting output of each test was observed on Channel 2 of the oscilloscope as a 28-V_{pp} square wave (14-V DC offset), due to the 28 volts supplied by the power supply. Figure 5 shows the RSLogix code used to implement the test, with the code consisting of a single rung of ladder logic that reads the function generator input and sends it unchanged through an output module to be displayed on the oscilloscope. For Test #1, the ControlNet RPI, IB16 RPI, OB8 RPI, and NUT were all set to 10 ms. The result of Test #1 is shown below in Fig. 6 (Channel 1 in yellow, Channel 2 in pink).

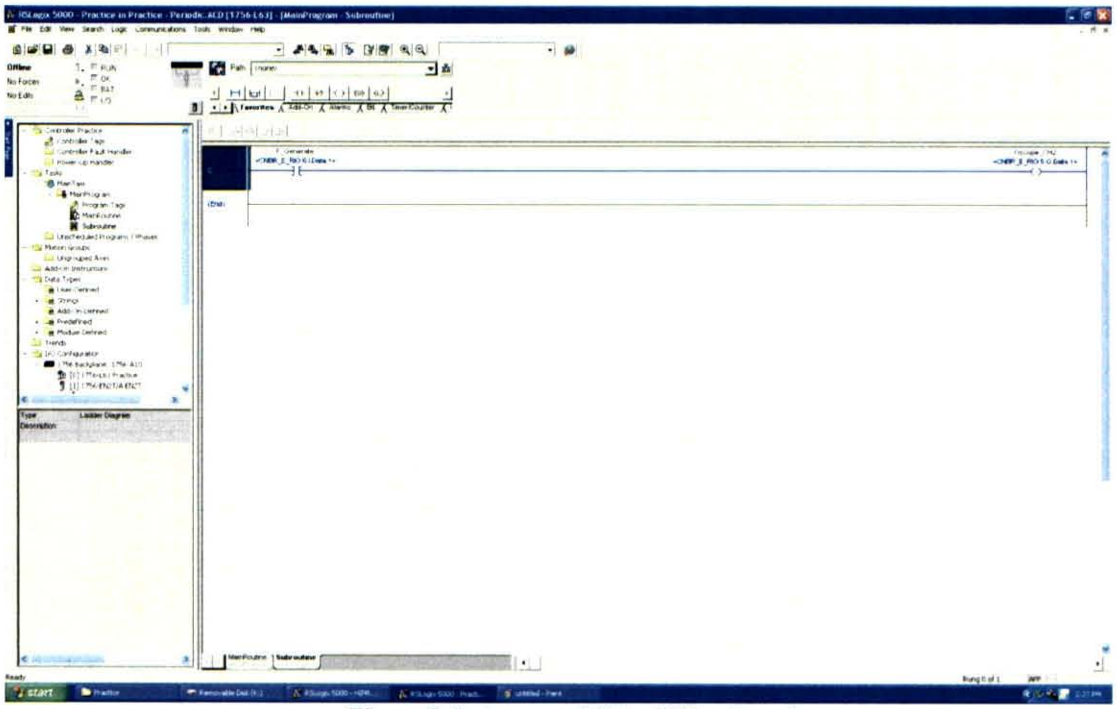
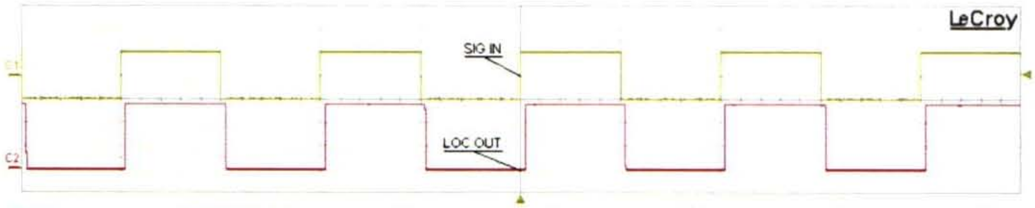


Figure 5. Latency and jitter RSLogix code.

Lab Notebook Entry from LeCroy DSO
DSO S/N: LCRY0312N48730
User: LeCroyUser
Time: 2/19/2010 2:35:05 PM

Latency/Jitter Test #1

LeCroy DSO Report



Measure	P1:skew(C1,C2)	P2:---	P3:---	P4:---	P5:---	P6:---
value	23.70841 ms					
mean	18.4977501 ms					
min	13.93789 ms					
max	23.94103 ms					
sdev	2.7438981 ms					
num	416					
status	✓					

C1: 10.0 V/div, 10.000 V
C2: 10.0 V/div, -30.000 V

Timebase 0.00 s, 500 ms/div, 5.00 MS, 1.0 MS/s
Trigger C1 DC, Stop 700 mV, Edge Positive

Figure 6. Latency and Jitter Test #1.

The skew function of the oscilloscope was used to measure the time delay between the rising edges of the input and output square waves. The min and max values displayed are the min and max values of this delay, representing latency measurements. The difference between the min and max values is the jitter for the test run. The standard deviation (sdev) and total number of samples (num) taken in the test run are also displayed. Table 1 of the Appendix contains the information from all 71 latency and jitter tests.

B. Synchronization and Time Stamping

Among its many challenges, the LCS must be able to time tag measurements taken in order to later facilitate event reconstruction, trend analysis, and fault detection. If data packets are properly time tagged, engineers will know the exact sequence of data packets sent to and received from every node on the network. This will be useful for verification and testing purposes, as well as analyzing trends over time to better understand the network's operations and to prevent future faults. In general, it is preferable to time tag as close as possible to where measurements are taken, hence we will time tag the packets at the local controllers. Time tagging will occur for every Ethernet packet before transmission to the LCC, rather than for every single measurement. This will not only reduce the size of the time stamp within the data packet, but also bandwidth consumption.

The KGCS design can be divided into two major ground control system parts: the components that reside at the LCC from where operations are managed by launch control personnel and the components that reside at the launch pad where PLCs receive LCC commands and return corresponding responses from end items. The PLCs that reside at the launch pad can be subdivided even further into local controllers and RIO chassis. Each chassis will maintain a Coordinated System Time (CST) that is local to each controller and consists of a free-running 64-bit time incremented in microseconds. As a result of this distributed nature, these individual chassis must have their CSTs synchronized in order to have a consistent time tagging mechanism across the entire KGCS. Once synchronization is achieved, each controller will be able to time tag measurement data consistently across the entire system before transmitting it to the LCC. One further complication is that even if perfect synchronization is achieved among all KGCS chassis, synchronization must also be achieved between the pad devices (local controllers and RIOs) and the LCC systems 5.47 kilometers away connected via high-speed (10 GB/s) Ethernet lines.

Therefore, synchronization is a two-fold engineering task: the field chassis controller clock times must be synchronized to each other and simultaneously to some master time source, which in our case is Coordinated Universal Time (UTC). UTC is a time standard based on International Atomic Time (TAI), with leap seconds added at irregular intervals to compensate for the Earth's slowing rotation. UTC replaced Greenwitch Mean Time (GMT) as the world timing standard in 1986. UTC is provided to KSC by means of Global Positioning System (GPS) signals sent down from GPS satellites to an antenna located on top of the Vehicle Assembly Building (VAB). The KGCS will receive this UTC time signal through an IRIG-B line inputted into a 1756HP-GPS (Hiprom) module, which will act as a Grandmaster (GM) time source for the KGCS. Code B of the inter-range instrumentation group time code (IRIG) sends timing information at a 100 Hz bit rate, with 100 bits per frame, for a frame rate of 1 Hz. IRIG time code consists of repeating frames, each containing 60 or 100 bits with information on the year, day of the year, hours, minutes, and seconds. The Hiprom module reports UTC as a 64-bit unsigned integer representing the number of microseconds elapsed since January 1, 1970, making UTC independent of the configured time zone. Time synchronization with the input IRIG-B signal and the internal clock of the Hiprom module is within a millisecond.[5]

Note that installing the Hiprom module in the local controller chassis and not in the RIO chassis, increases the accuracy of the time tag on the Ethernet packet, but decreases the accuracy of the time tag on the measurements taken by the RIO modules. The latter is because of the ControlNET network with its associated NUT and I/O modules with an equivalent RPI.

Version 2 of the Precision Time Protocol, defined in the IEEE 1588-2008 standard "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," establishes a high precision time synchronization protocol for synching the clocks of the local controllers and RIO chassis down to the microsecond. Many companies in industry, including Allen-Bradley, have developed products capable of synchronizing their clocks using PTP over Ethernet networks. When multiple EN2T modules are connected via an Ethernet network, their clocks are synchronized automatically. The IEEE 1588 standard specifies that this synchronization occurs without the need for any administration or initial setup.

IEEE 1588 synchronization operates by creating a master/slave hierarchy within a network. Each slave node is synchronized to the time, frequency, and phase of the grandmaster clock source. By extension, each slave node is effectively synchronized to all the other nodes on the network. Synchronization requires two steps: (1) determine which device serves as the master clock, and (2) measure and correct skew caused by clock offsets and network delays.[6] When the system first goes online, the IEEE 1588 protocol uses a Best Master Clock algorithm to automatically determine which clock in the network is the most precise. This clock is then set as the master, with all other clocks becoming slaves and synchronizing their clocks with the master. Since the time difference between the master and slave clocks is a combination of the clock offset and message transmission delay, correcting the clock skew is done in two phases -- offset correction and delay correction.

The PTP algorithm calculates the differences in time and frequency, then adjusts the slave clock times and frequencies so that they match the master. The master clock initiates offset correction using "sync" and "follow-up" messages (see Fig. 7). When the master sends a sync message, the slave uses its local clock to time stamp the arrival of the sync message and compares it to the actual sync transmission time stamp in the master clock's follow-up message. The difference between the two time stamps represents the offset of the slave plus the message transmission delay. The slave clock then adjusts the local clock by this difference at point A. To correct for the message transmission delay, the slave uses a second set of sync and follow-up messages with its corrected clock to calculate the master-to-slave delay at point B. The second set of messages is necessary to account for variations in network delays. The slave then time stamps the sending of a delay request message. The master clock time stamps the arrival of the delay request message. It then sends a delay response message with the delay request arrival timestamp at point C. The difference between the time stamps is the slave-to-master delay. The slave averages the two directional delays and then adjusts the clock by the delay to synchronize the two clocks. Since the master and slave clocks drift independently, periodically repeating offset correction and delay correction keeps the clocks synchronized. According to one of Rockwell's representatives, Mark Schillace, the master EN2T should send a sync message to its slaves once per second.

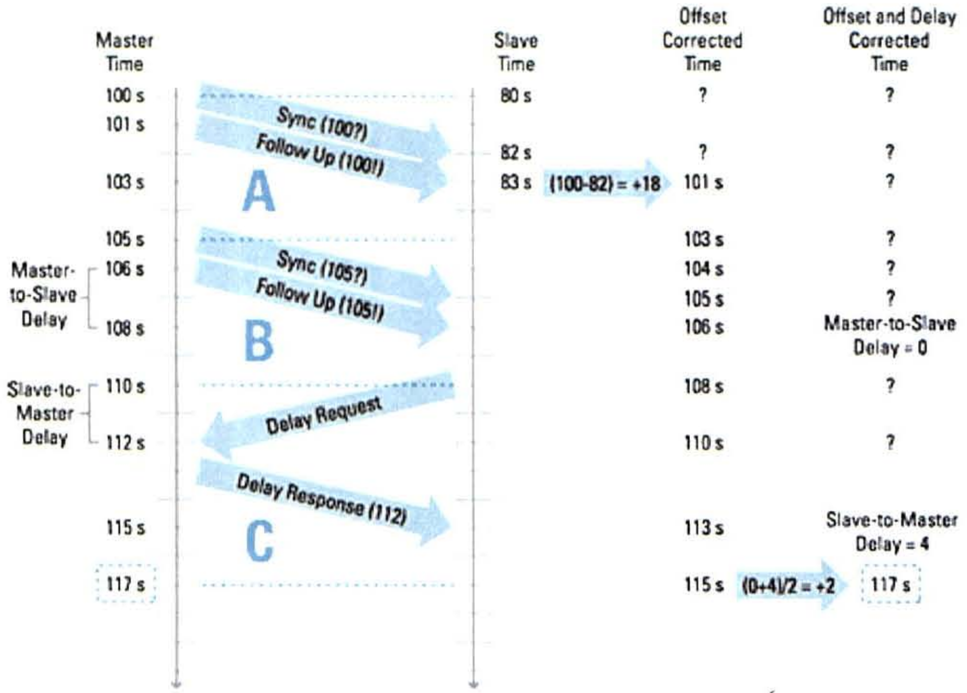


Figure 7. Master/Slave synchronization messages.[6]

Figure 8 displays the master/slave network topology that allows the KGCS to utilize IEEE 1588 synchronization. A GPS timing signal is received by the VAB antenna and sets the 64-bit microsecond UTC time on the Hiprom module. The Hiprom acts as a Grandmaster clock, broadcasting its timing signal to all of the connected local controllers. The local controllers serve as master clocks, which synchronize all of their slave RIO chassis through the IEEE 1588 PTP version 2, which uses an 80-bit timestamp. The most significant 48 bits represent the number of seconds elapsed since January 1, 1970, while the least significant 32 bits characterize the nanosecond fraction of seconds. However, within RSLogix, the time stamp is displayed as the 64-bit microsecond UTC time broken up into two separate double integers (DINTs). This is the time stamp that will be included in the KGCS data packets.

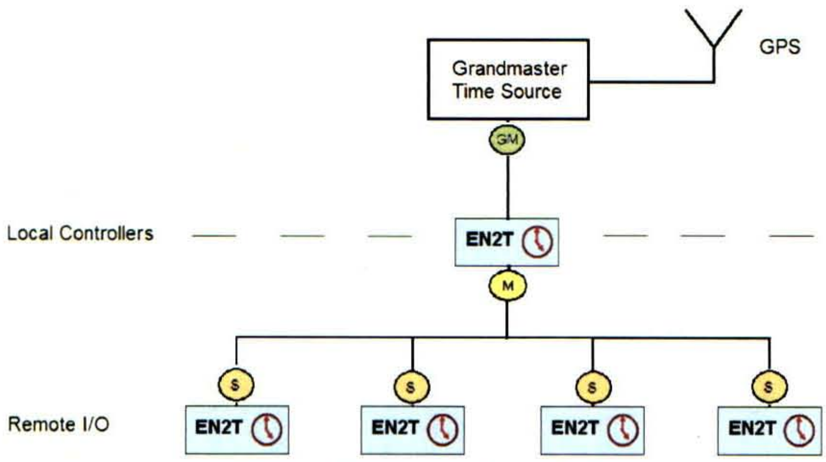


Figure 8. Master/Slave Network Topology.

For our KGCS timing and synchronization testing, I connected five separate ControlLogix chassis with EN2T modules to an Ethernet switch. Figure 9 shows the setup, with the chassis containing the local controller (Logix 5563) and master EN2T in the upper righthand corner and the chassis with the slave EN2Ts and controllers (Logix 5563) located below. A Hiprom module was included in the master chassis to input a coax cable containing the IRIG-B time from KSC's GPS antenna. The computer in the lower right was used to download the master and slave RSLogix project files into the respective controllers, as well as to monitor the real-time synchronization. The laptop in the lower left of the picture was used to analyze the IEEE 1588 communications using Wireshark, a network protocol analyzer software program. Figure 10 displays a simplified schematic of the setup.

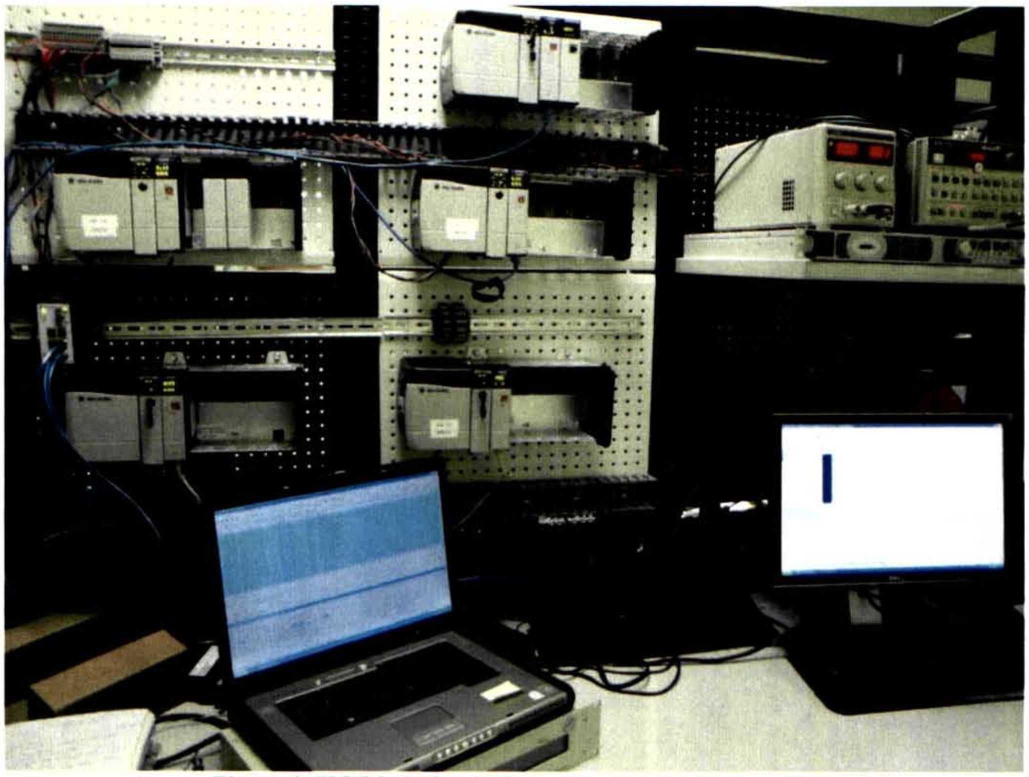


Figure 9. KGCS timing and synchronization test configuration.

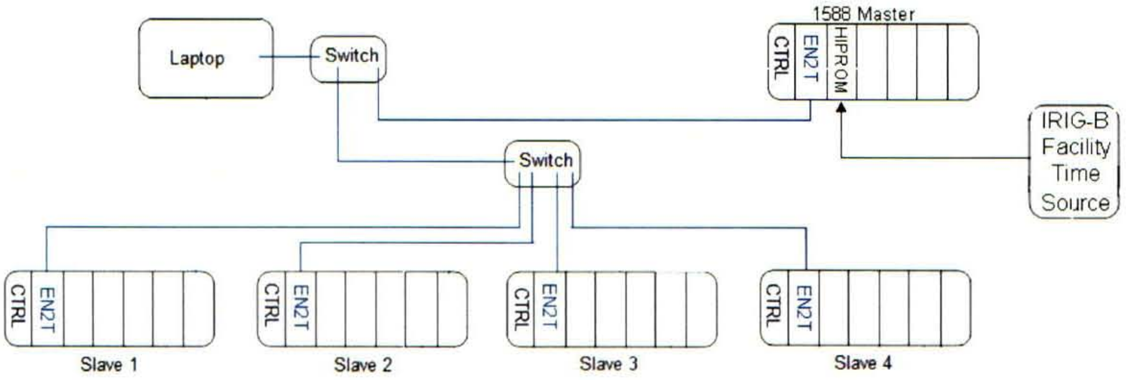


Figure 10. KGCS timing and synchronization test schematic.

In order to set the master EN2T and analyze the time stamps, I used modified RSLogix project files sent by Mark Schillace of Rockwell Automation. Figure 11 shows the master RSLogix project file that was downloaded into the master chassis' local controller. This master project file first copies the 64-bit microsecond UTC time inputted by IRIG-B through the Hiprom module (via the GPS antenna) and stores this value in a user-defined tag structure within the local controller. The code then contains an option to either use this inputted UTC time or the controller's Wall Clock Time (WCT) as the real-time clock source for the slaves. The WCT is simply the CST value with an offset that is relative to a system defined point in time. For my testing purposes, I set the controller's WCT as the real-time source so that I could control the inputted time. Next, the code sends either the inputted UTC time or the WCT to the local EN2T, which is subsequently set as the master EN2T by setting its priority to 1 (on a scale of 1-128). The slave EN2Ts have a lower priority number and thus becomes slaves.

After setting the master EN2T, IEEE 1588 synchronization automatically takes place, sending the time set in the master EN2T to the slave EN2T clocks. Figure 12 presents an example of the slave project file downloaded into each slave controller. The code retrieves the time set in the slave EN2Ts so that it can be viewed in RSLogix and compared with the master EN2T time for synchronization verification.

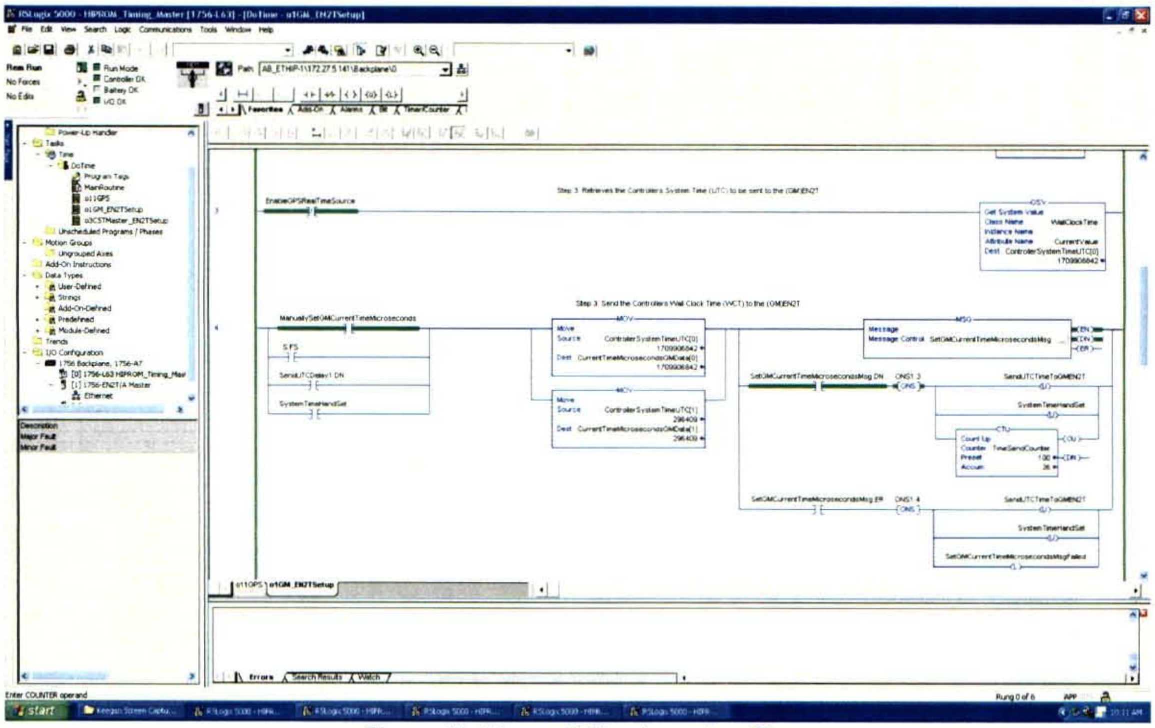
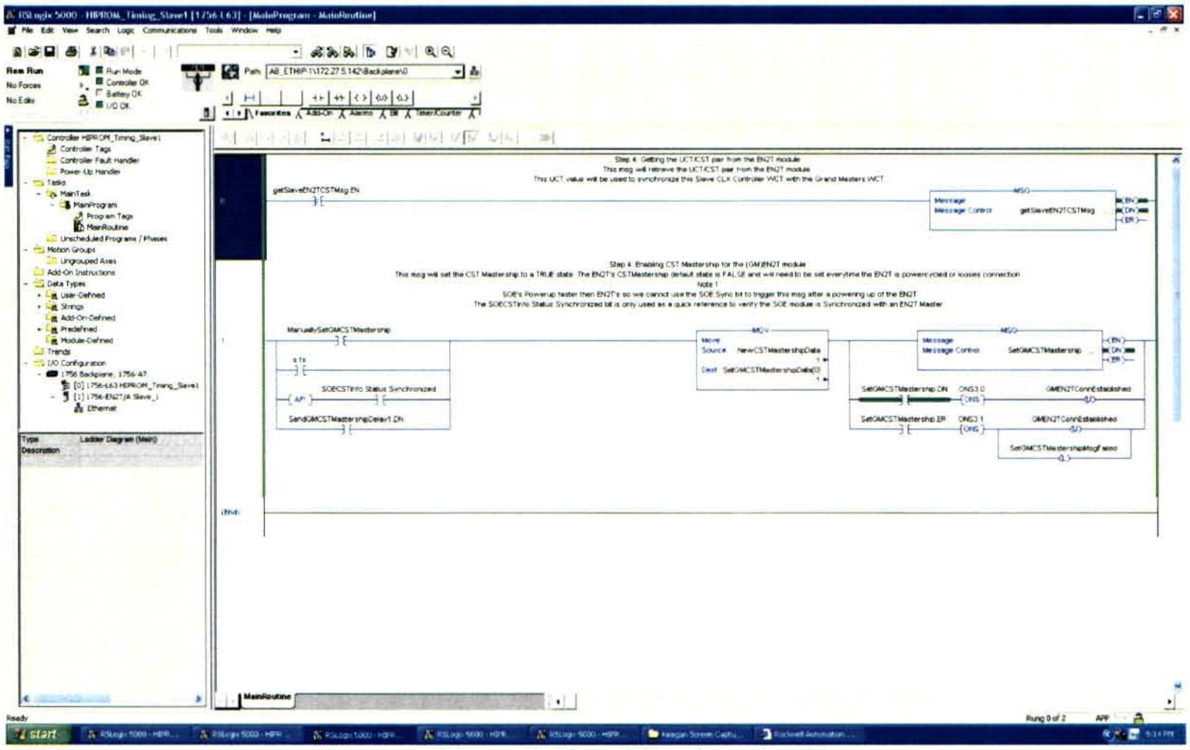


Figure 11. Master RSLogix project file.



RSLogix 5000 - HPROM_Timing_Slave1 [1756-L63] - [MainProgram - MainRoutine]
Path: AB_ETHIP-1\172.27.5.142\Backplane\0
getSlaveEN2TCSTMsg.EN
MSG
Message
Message Control getSlaveEN2TCSTMsg
ManuallySetGMCSTMastership
MOV
Move
Source NewCSTMastershipData
Dest SetGMCSTMastershipData[0]
MSG
Message
Message Control SetGMCSTMastership
SOECSTInfo Status Synchronized
SendGMCSTMastershipDelay1.DN
SetGMCSTMastership.DN ONS3.0
SetGMCSTMastership.ER ONS3.1
GMEN2TConnEstablished
(End)
MainRoutine

Step 4: Getting the UCT/CST pair from the EN2T module
This msg will retrieve the UCT/CST pair from the EN2T module
This UCT value will be used to synchronize this Slave CLX Controller WCT with the Grand Masters WCT

Step 4: Enabling CST Mastership for the (GM)EN2T module
This msg will set the CST Mastership to a TRUE state. The EN2T's CSTMastership default state is FALSE and will need to be set everytime the EN2T is powercycled or looses connection
Note 1
SOE's Powerup faster then EN2T's so we cannot use the SOE Sync bit to trigger this msg after a powering up of the EN2T
The SOECSTInfo Status Synchronized bit is only used as a quick reference to verify the SOE module is Synchronized with an EN2T Master

Figure 12. Slave RSLogix project file.

V. Results

A. Latency and Jitter

A few interesting results can be gleaned from analyzing the data in Table 1 of the Appendix. Latencies were generally in the range of 8 – 34 ms, while jitter was typically 5 – 20 ms. These latency values indicate a significant time delay, considering the fundamental speed of light delay for the trip from the LCC to the launch pad (and vice versa) is in the tens of microseconds. When conducting the tests, in almost every case the latencies consistently decreased from the max to the min value, before jumping back to the max value and repeating the process. This demonstrated a periodic alignment pattern of the RPIs and NUT, which could be expected from the intrinsically periodic nature of the RPIs and NUT.

Most of the tests where the ControlNet RPI was increased from its nominal value of 10 ms resulted in an increase in the max and mean latency values, with the min value staying constant. For example, in Test #5 the ControlNet RPI was increased to 50 ms, producing a 40 ms jitter due to the raising of the max latency value to almost 54 ms. This is intuitive, as increasing RPIs should result in longer time delays, especially for the ControlNet module, which regulates the communication speed between local and remote chassis.

Keeping the RPIs constant while reducing the NUT produced mixed results. As Test #8 displays, decreasing the NUT to 5 ms while keeping the RPIs at 10 ms produced a lower mean, min, and max latency, though the jitter stayed at the same 10 ms value as the nominal case (all RPIs and the NUT set to 10 ms). However, keeping the RPIs at 10 ms while changing the NUT to lower values, such as 2 ms in Test #18, actually produced higher mean, min, and max latencies. Discrepancies like this proved the unpredictable nature of changing RPIs or the NUT. RSNetworx scheduling only guarantees that the set RPIs and NUT are the maximum update times, but that lower times can be permitted depending on scheduling parameters. There is also no way to observe the actual update rates on the backplane to construct graphs detailing the patterns giving rise to the max to min latency precession.

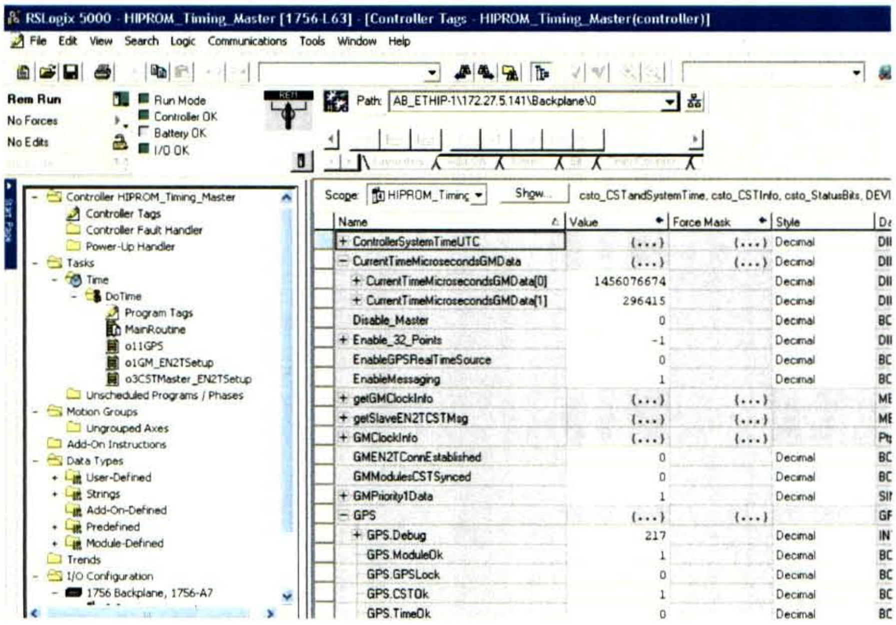
My latency and jitter testing was more of a lesson on how to measure these values in a network than a concrete examination of solutions for the timeliness and consistency of KGCS PLC communications. Without the ability to directly observe the actual RPIs and NUT rather than the values set in RSLogix, no formal conclusions can be reached on the best method to reduce the KGCS' latency and jitter. The most noteworthy result of my latency and jitter testing was that the PLCs themselves produced the most time delay and unpredictability in the network. Millisecond latency is important to understand in time-critical applications such as the KGCS, where the motion and condition of thousands of valves, switches, and heaters must be continuously monitored and controlled. Further latency and jitter testing within a more realistic scenario will provide more accurate insights into the timing characteristics of the actual KGCS PLC network.

B. Time Stamping and Synchronization

Upon downloading the master and slave project files to their respective controllers, I set the local controller's WCT to the current local time, and then observed the timing and synchronization results in RSLogix, Wireshark, and the EN2T webpages. The first step was to check the master EN2T's webpage and RSLogix project file to confirm that the local controller and master EN2T had, in fact, been set to the local time (5:37 p.m. ET on May 5, 2010).

Figure 13 shows the two tags (CurrentTimeMicrosecondsGMData[0] and CurrentTimeMicrosecondsGMData[1]) that store the 64-bit microsecond UTC time, that in this case, comes from the local controller's WCT (rather than the typical GPS signal). A rough calculation shows the number of microseconds that had elapsed since January 1, 1970 (as of May 5, 2010):

(60 seconds/minutes) x (60 minutes/hour) x (24 hours/day) x (365.25 days/year) x 40 years + (86,400 seconds/day) x 125 days = 1.273104 x 10^9 seconds or 1.273104 x 10^{15} microseconds



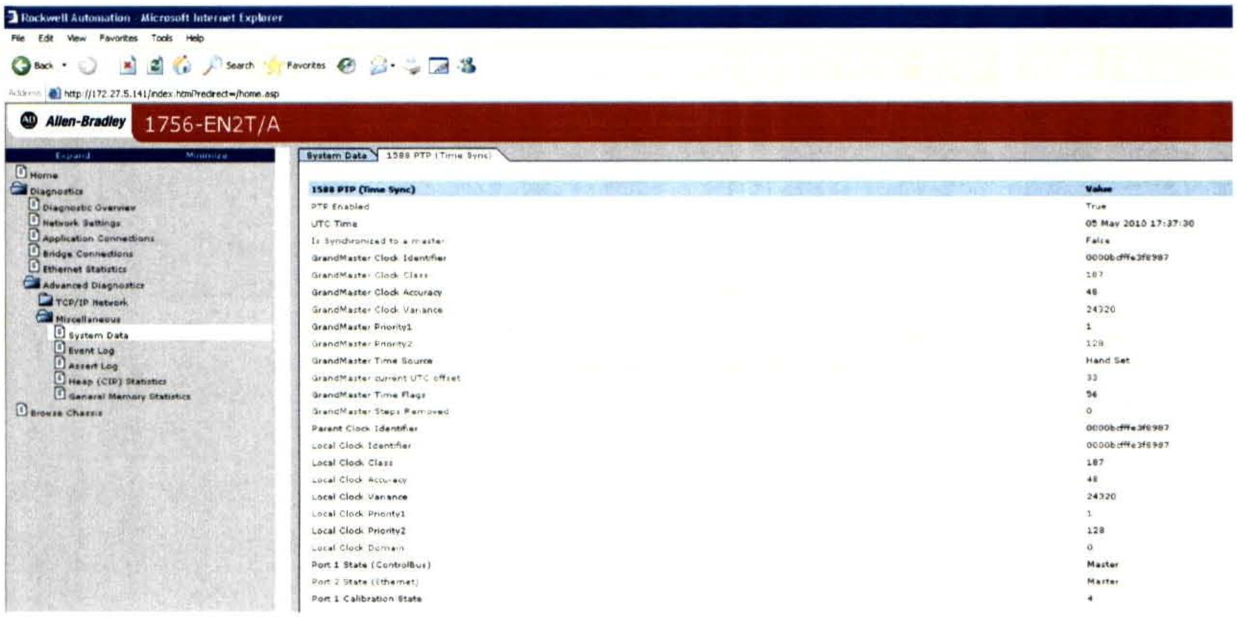
Name	Value	Force Mask	Style	Da
+ ControllerSystemTimeUTC	{...}	{...}	Decimal	DII
− CurrentTimeMicrosecondsGMData	{...}	{...}	Decimal	DII
+ CurrentTimeMicrosecondsGMData[0]	1456076674		Decimal	DII
+ CurrentTimeMicrosecondsGMData[1]	296415		Decimal	DII
Disable_Master	0		Decimal	BC
+ Enable_32_Points	-1		Decimal	DII
EnableGPSRealTimeSource	0		Decimal	BC
EnableMessaging	1		Decimal	BC
+ getGMClockInfo	{...}	{...}		ME
+ getSlaveEN2TCSTMsg	{...}	{...}		ME
+ GMClockInfo	{...}	{...}		Pt
GMEN2TConnEstablished	0		Decimal	BC
GMModulesCSTSynced	0		Decimal	BC
+ GMPriority1Data	1		Decimal	SI
− GPS	{...}	{...}		GF
+ GPS.Debug	217		Decimal	IN
GPS.ModuleOk	1		Decimal	BC
GPS.GPSLock	0		Decimal	BC
GPS.CSTOk	1		Decimal	BC
GPS.TimeOk	0		Decimal	BC

Figure 13. Local controller time tag.

Figure 13 shows a value of 296415 in the most significant DINT and 1456076674, representing elapsed microseconds, in the least significant DINT. Due to the decimal format in which the tags display their values, the least significant DINT counts through 4,294,000,000 microseconds before incrementing the most significant DINT by 1. Therefore, the total number of microseconds that are displayed in this 64-bit value:

296415 x 4294000000 + 1456076674 = 1272807466076674 microseconds or 1272807466 seconds

This compares favorably to my rough calculation above, within the accuracy of not accounting for leap years and conducting an exact calculation. However, an exact calculation does not need to be conducted. The webpage for the master EN2T, shown below in Figure 14, correctly displays the UTC time set in the master EN2T. As can be seen, the UTC time is listed as "05 May 2010 17:37:30," indicating that the master EN2T's clock had, indeed, been changed to the local controller WCT that I had previously set to May 5, 2010 at 5:37 p.m. ET. Note that the "Is Synchronized to a master" line on the master EN2T webpage is "False," since the master EN2T is not connected to a Grandmaster source. I suspect that actually connecting the facility IRIG-B coax line to the Hiprom module will change this value to "True," as the Hiprom module would then act as the Grandmaster time source. The "Grandmaster Time Source" line lends further credence to this theory, as it displays "Hand Set," which was what I did by manually setting the local controller WCT. The "Port 2 State (Ethernet)" line says "Master," delineating this as the master EN2T and the "Local Clock Identifier" line labels master clock's identity as "0000bcfffe3f8987."

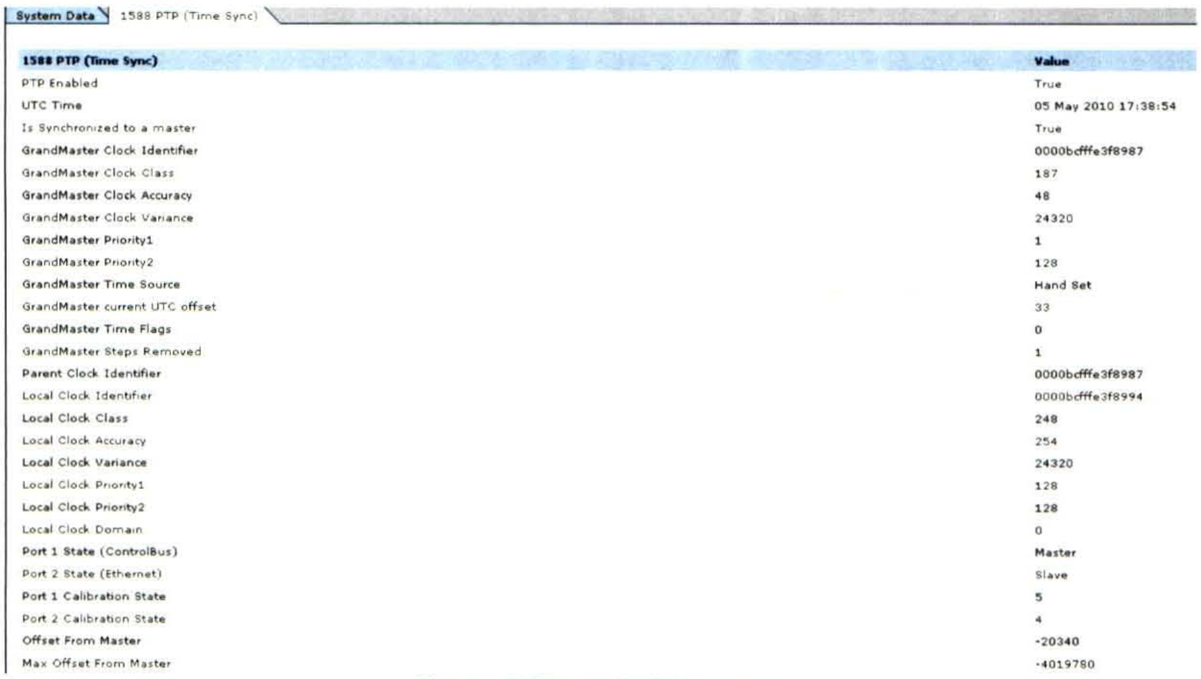


1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 17:37:30
Is Synchronized to a master	False
GrandMaster Clock Identifier	0000bcfffe3f8987
GrandMaster Clock Class	187
GrandMaster Clock Accuracy	48
GrandMaster Clock Variance	24320
GrandMaster Priority1	1
GrandMaster Priority2	128
GrandMaster Time Source	Hand Set
GrandMaster current UTC offset	33
GrandMaster Time Flags	56
GrandMaster Steps Removed	0
Parent Clock Identifier	0000bcfffe3f8987
Local Clock Identifier	0000bcfffe3f8987
Local Clock Class	187
Local Clock Accuracy	48
Local Clock Variance	24320
Local Clock Priority1	1
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Master
Port 1 Calibration State	4

Figure 14. Master EN2T webpage.

Now that it was clear the master EN2T had been correctly set, I needed to verify that the slave clocks had been synched to this time. Figures 15-18 show the webpages for the slave 1 EN2T, slave 2 EN2T, slave 3 EN2T, and slave 4 EN2T, respectively. All of the UTC times are within minutes of the master EN2T UTC time, which is within the speed that I could obtain screen captures of the webpages (in real-time, the master and slave UTC times exactly matched). The "Is Sychronized to master" lines changed to "True," indicating that the slave EN2T clocks were synched to the master EN2T clock. The "Grandmaster Clock Identifier" and "Parent Clock Identifier" lines for each slave changed to "000bcfff3f8987," which was the unique identifier of the master EN2T clock. The "Port 2 State (Ethernet)" lines were all listed as "Slave," as was expected for the slave EN2Ts.

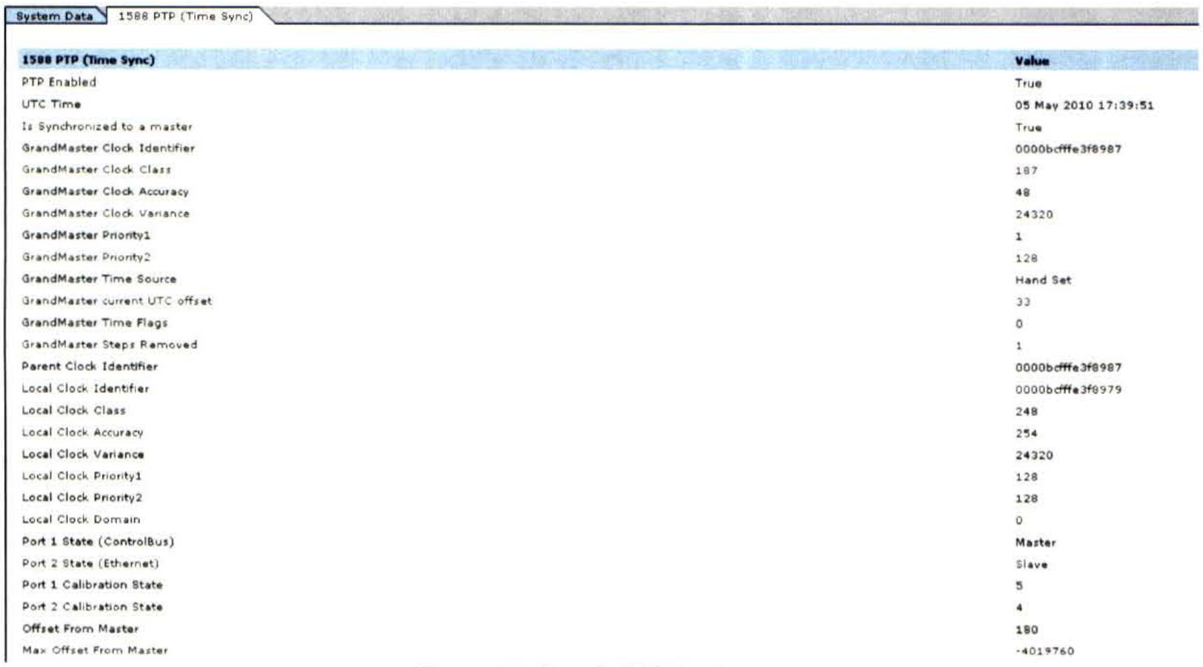
The next step was to verify that the two "CurrentTimeMicrosecondsGMData" DINT tags contained in each of the slave RSLogix project files were set to the correct microsecond value. Figure 19 displays the two DINTs for the slave 1 controller; the most significant DINT is 296415, while the least significant DINT is 1502028365. This is within seconds of the master EN2T 64-bit UTC time in Fig. 13, again acceptable given the limits of the speed at which I could obtain screen captures of the tags. The slave 2, slave 3, and slave 4 project files also contained values matching the time in the master EN2T. All of these results indicate that the slave EN2T clocks were accurately synched to the master EN2T. The next step was to verify the actual IEEE 1588 communications.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 17:38:54
Is Synchronized to a master	True
GrandMaster Clock Identifier	0000bcfffe3f8987
GrandMaster Clock Class	187
GrandMaster Clock Accuracy	48
GrandMaster Clock Variance	24320
GrandMaster Priority1	1
GrandMaster Priority2	128
GrandMaster Time Source	Hand Set
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	1
Parent Clock Identifier	0000bcfffe3f8987
Local Clock Identifier	0000bcfffe3f8994
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Slave
Port 1 Calibration State	5
Port 2 Calibration State	4
Offset From Master	-20340
Max Offset From Master	-4019780

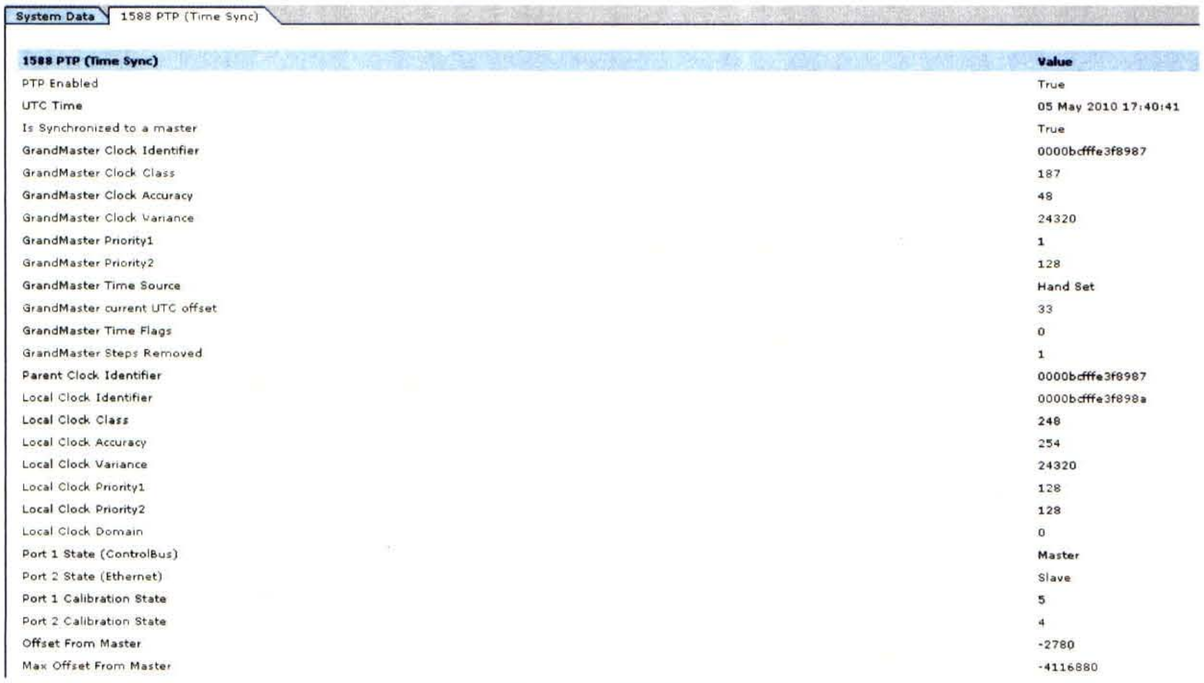
Figure 15. Slave 1 EN2T webpage.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 17:39:51
Is Synchronized to a master	True
GrandMaster Clock Identifier	0000bcfffe3f8987
GrandMaster Clock Class	187
GrandMaster Clock Accuracy	48
GrandMaster Clock Variance	24320
GrandMaster Priority1	1
GrandMaster Priority2	128
GrandMaster Time Source	Hand Set
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	1
Parent Clock Identifier	0000bcfffe3f8987
Local Clock Identifier	0000bcfffe3f8979
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Slave
Port 1 Calibration State	5
Port 2 Calibration State	4
Offset From Master	180
Max Offset From Master	-4019760

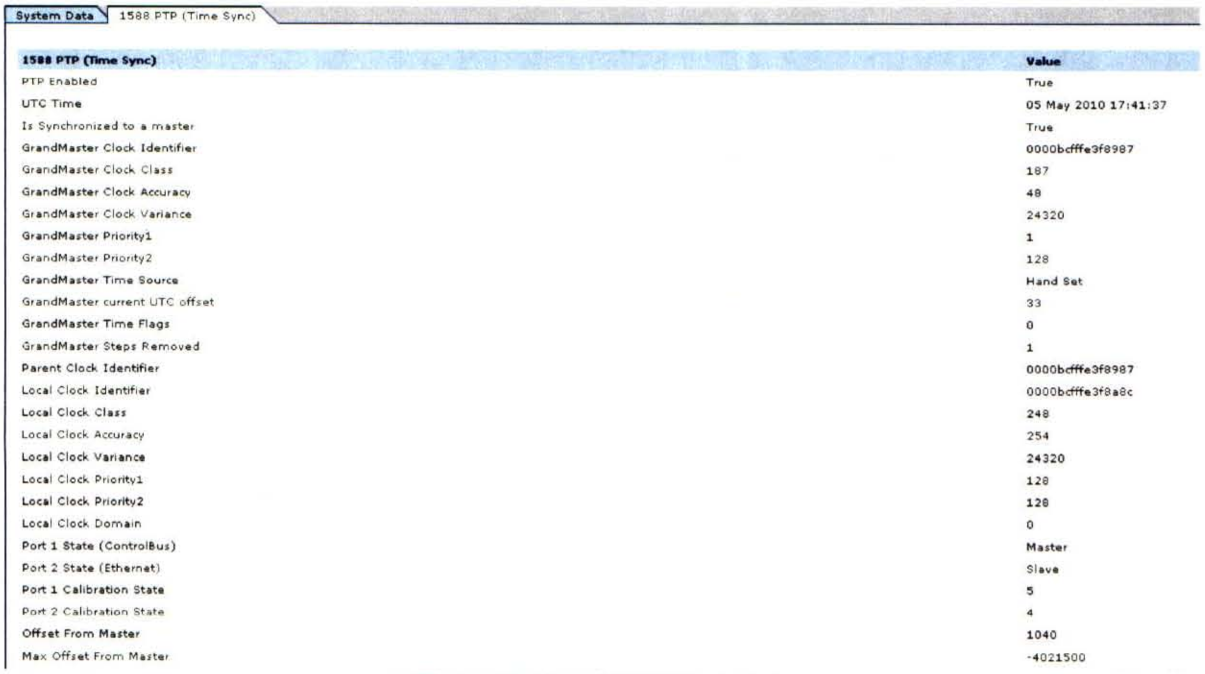
Figure 16. Slave 2 EN2T webpage.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 17:40:41
Is Synchronized to a master	True
GrandMaster Clock Identifier	0000bcfffe3f8987
GrandMaster Clock Class	187
GrandMaster Clock Accuracy	48
GrandMaster Clock Variance	24320
GrandMaster Priority1	1
GrandMaster Priority2	128
GrandMaster Time Source	Hand Set
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	1
Parent Clock Identifier	0000bcfffe3f8987
Local Clock Identifier	0000bcfffe3f898a
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Slave
Port 1 Calibration State	5
Port 2 Calibration State	4
Offset From Master	-2780
Max Offset From Master	-4116880

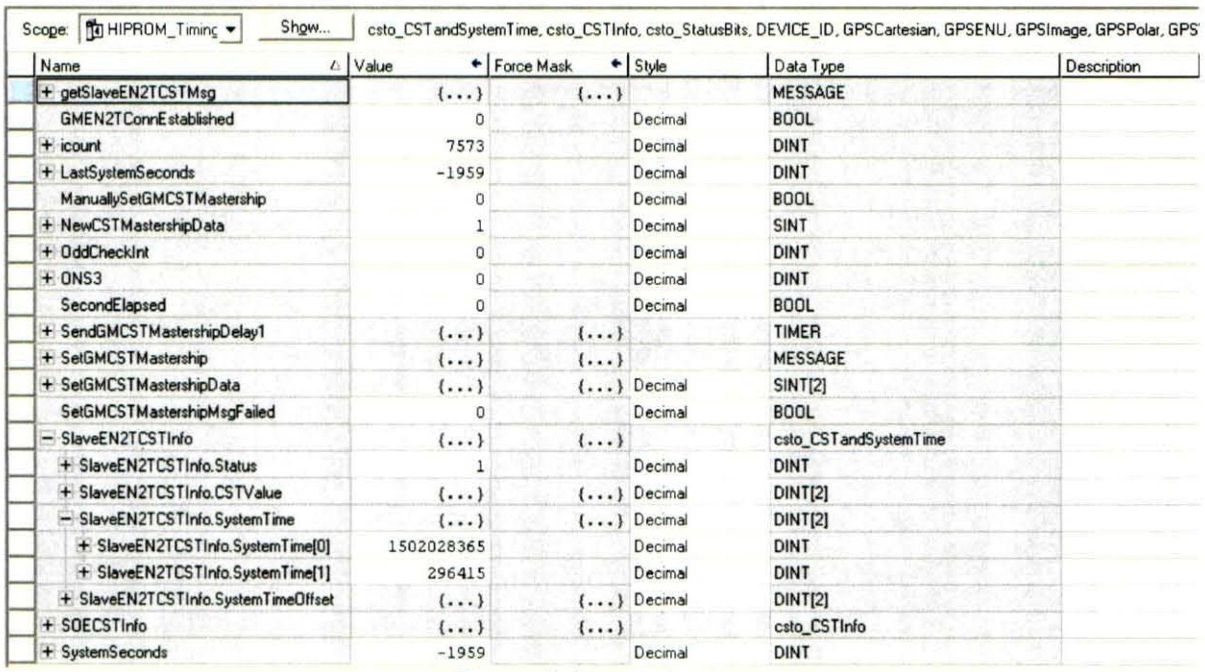
Figure 17. Slave 3 EN2T webpage.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 17:41:37
Is Synchronized to a master	True
GrandMaster Clock Identifier	0000bcfffe3f8987
GrandMaster Clock Class	187
GrandMaster Clock Accuracy	48
GrandMaster Clock Variance	24320
GrandMaster Priority1	1
GrandMaster Priority2	128
GrandMaster Time Source	Hand Set
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	1
Parent Clock Identifier	0000bcfffe3f8987
Local Clock Identifier	0000bcfffe3f8a8c
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Slave
Port 1 Calibration State	5
Port 2 Calibration State	4
Offset From Master	1040
Max Offset From Master	-4021500

Figure 18. Slave 4 EN2T webpage.



Scope: HIPROM_Timinc ▾ Show... csto_CSTandSystemTime, csto_CSTInfo, csto_StatusBits, DEVICE_ID, GPSCartesian, GPSENU, GPSImage, GPSPolar, GPS

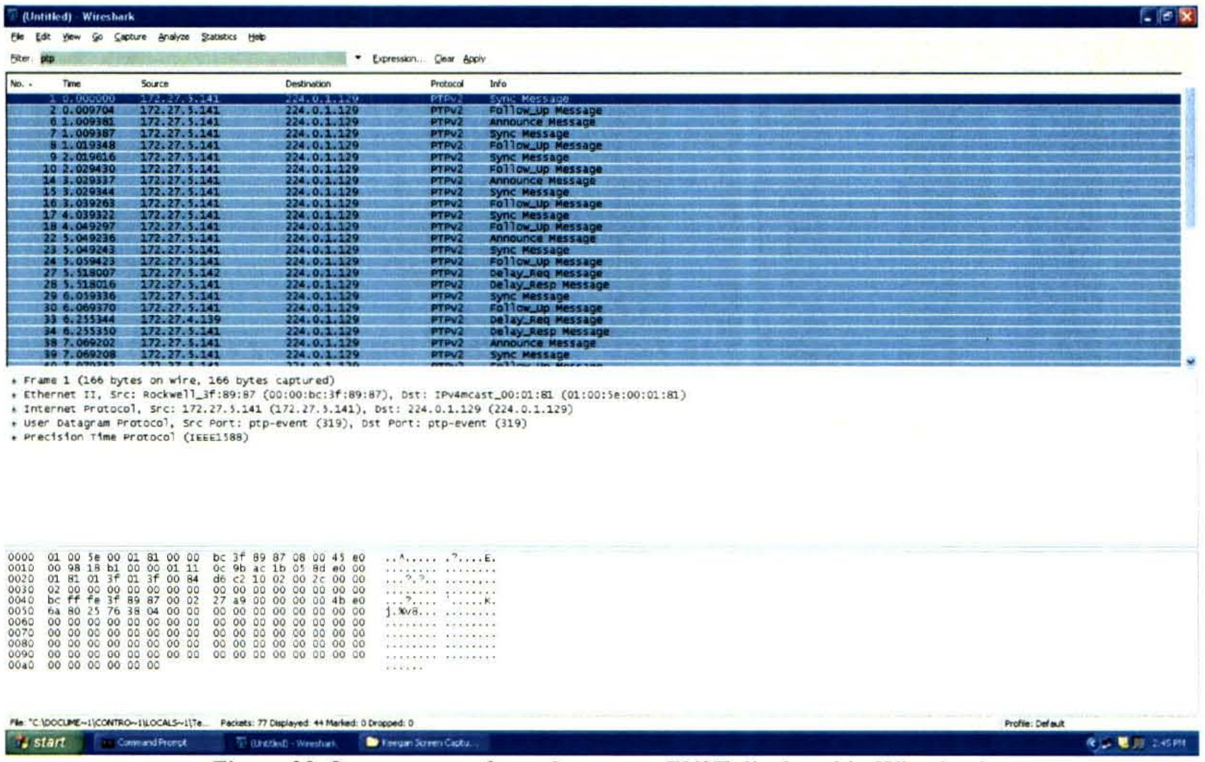
Name	Value	Force Mask	Style	Data Type	Description
+ getSlaveEN2TCSTMsg	{...}	{...}		MESSAGE	
GMEN2TConnEstablished	0		Decimal	BOOL	
+ icount	7573		Decimal	DINT	
+ LastSystemSeconds	-1959		Decimal	DINT	
ManuallySetGMCSTMastership	0		Decimal	BOOL	
+ NewCSTMastershipData	1		Decimal	SINT	
+ OddCheckInt	0		Decimal	DINT	
+ ONS3	0		Decimal	DINT	
SecondElapsed	0		Decimal	BOOL	
+ SendGMCSTMastershipDelay1	{...}	{...}		TIMER	
+ SetGMCSTMastership	{...}	{...}		MESSAGE	
+ SetGMCSTMastershipData	{...}	{...}	Decimal	SINT[2]	
SetGMCSTMastershipMsgFailed	0		Decimal	BOOL	
− SlaveEN2TCSTInfo	{...}	{...}		csto_CSTandSystemTime	
+ SlaveEN2TCSTInfo.Status	1		Decimal	DINT	
+ SlaveEN2TCSTInfo.CSTValue	{...}	{...}	Decimal	DINT[2]	
− SlaveEN2TCSTInfo.SystemTime	{...}	{...}	Decimal	DINT[2]	
+ SlaveEN2TCSTInfo.SystemTime[0]	1502028365		Decimal	DINT	
+ SlaveEN2TCSTInfo.SystemTime[1]	296415		Decimal	DINT	
+ SlaveEN2TCSTInfo.SystemTimeOffset	{...}	{...}	Decimal	DINT[2]	
+ SOECSTInfo	{...}	{...}		csto_CSTInfo	
+ SystemSeconds	-1959		Decimal	DINT	

Figure 19. Slave 1 time tag.

The Wireshark software program allowed me to view the packet-by-packet communications over the network's Ethernet line. By applying filters, I limited the visible traffic to PTP messages from the master EN2T's IP address. The results are shown in Fig. 20, with each line representing a separate data packet. The "Time" column displays the number of elapsed seconds since the beginning of the run, the "Source" column indicates the packets are sent from the master EN2T IP address, the "Protocol" indicates PTPv2 for version 2 of the PTP, and the "Info" column shows the type of message being sent. It can be clearly seen that a "Sync Message" is sent from the master EN2T about every 1.009 seconds, verifying Rockwell representative Mark Schillace's statement that sync messages are sent once per second.

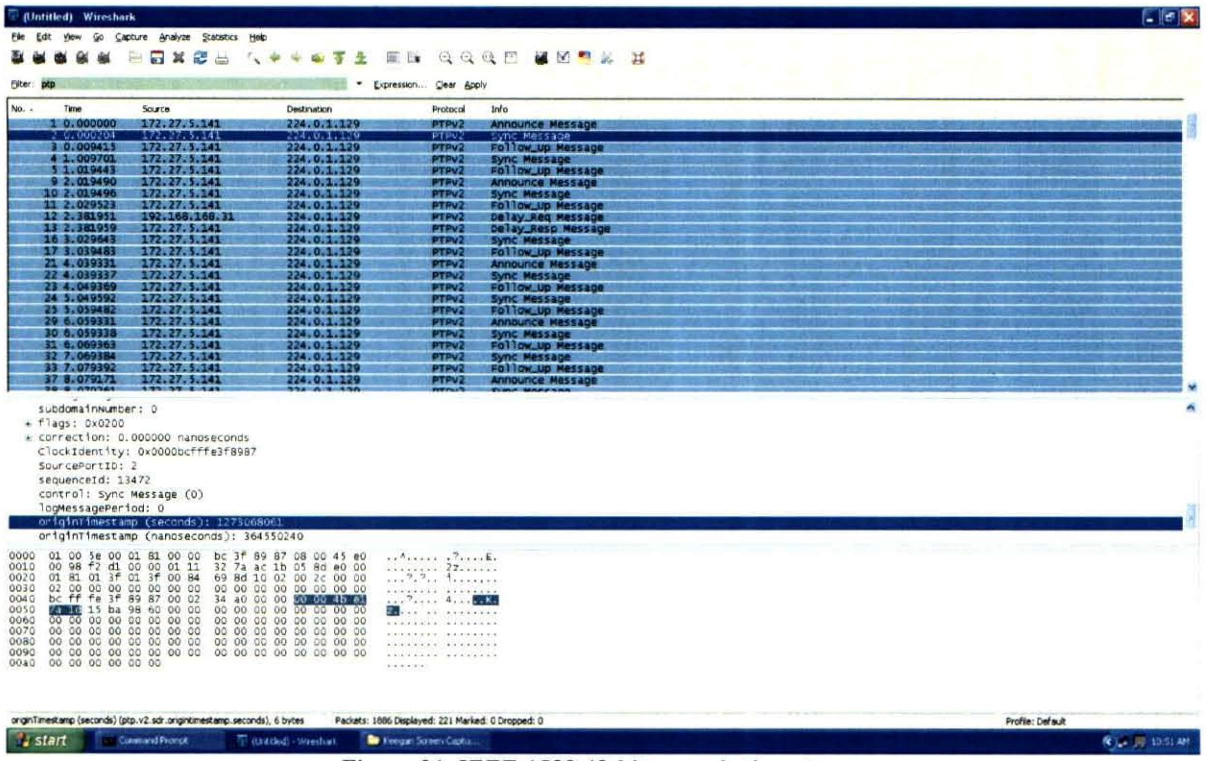
Figure 21 shows a breakdown of the sync message content, specifically the time stamp. The blue highlighted portion of the data packet indicates the 48-bit seconds portion of the time stamp, coded here in hex format. The "originTimestamp (seconds)" line lists 1273068061, which matches my rough calculation and the RSLogix values for the number of seconds elapsed since January 1, 1970. Figure 22 displays the 32-bit nanoseconds portion of the timestamp, which is stored contiguous to the 48-bit seconds within the IEEE 1588 data packet.

Observing Fig. 20-22 reveals an interesting characteristic of the IEEE 1588 communications. The lines labeled "12" and "13" in Fig. 22 are delay request and delay response messages, respectively. These two messages are sent much less frequency than the sync and follow-up messages. Figure 23 reveals the breakdown of the slave 1 EN2T's message count over a given time span. Over 100,000 sync and follow-up messages were sent/received, while only a little more than 3,000 delay request and delay response messages were sent/received. This demonstrates that the IEEE 1588 algorithm must be more concerned with correctly adjusting the master-to-slave message delay more than the slave-to-master delays.



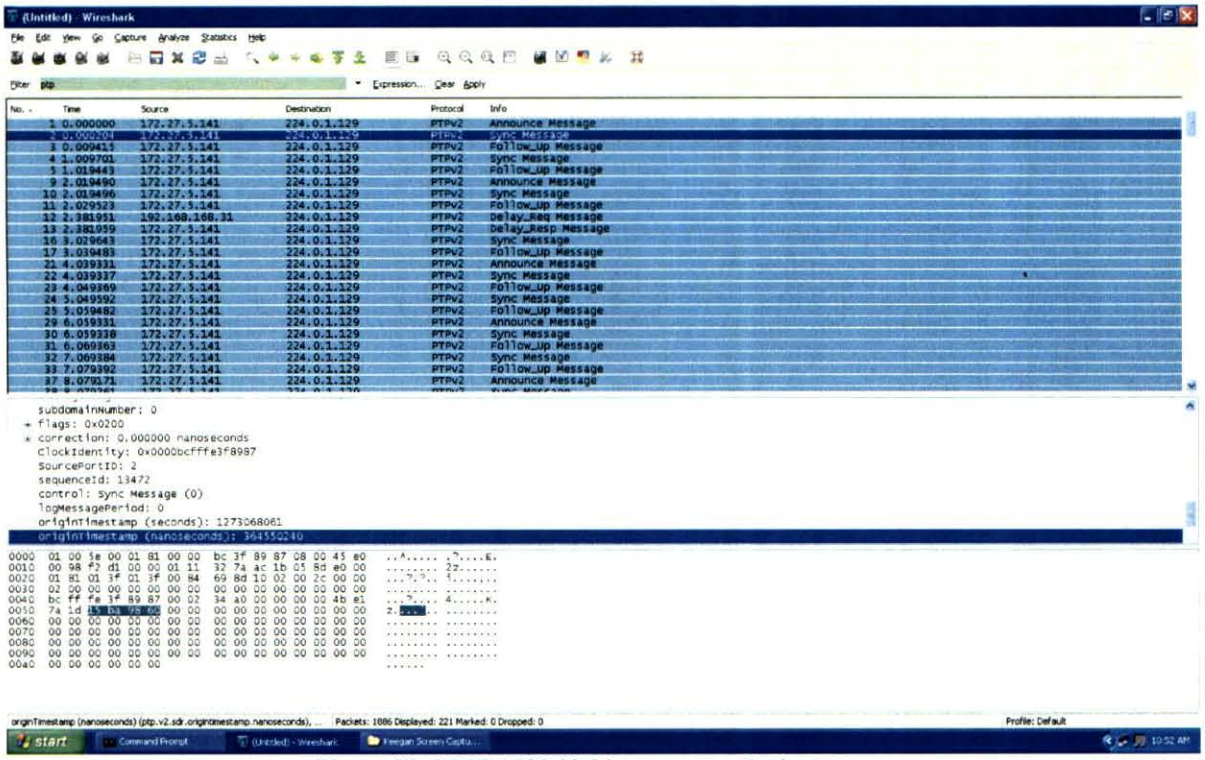
```
(Untitled) - Wireshark
File  Edit  View  Go  Capture  Analyze  Statistics  Help
Filter: ptp                                  Expression...  Clear  Apply

No. -  Time       Source          Destination    Protocol  Info
    1  0.000000   172.27.5.141    224.0.1.129    PTPv2     Sync Message
    2  0.009704   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
    6  1.009381   172.27.5.141    224.0.1.129    PTPv2     Announce Message
    7  1.009387   172.27.5.141    224.0.1.129    PTPv2     Sync Message
    8  1.019348   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
    9  2.019616   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   10  2.029430   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   14  3.029337   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   15  3.029344   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   16  3.039263   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   17  4.039322   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   18  4.049297   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   22  5.049236   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   23  5.049243   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   24  5.059423   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   27  5.518007   172.27.5.142    224.0.1.129    PTPv2     Delay_Req Message
   28  5.518016   172.27.5.141    224.0.1.129    PTPv2     Delay_Resp Message
   29  6.059336   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   30  6.069370   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   33  6.255344   172.27.4.139    224.0.1.129    PTPv2     Delay_Req Message
   34  6.255350   172.27.5.141    224.0.1.129    PTPv2     Delay_Resp Message
   38  7.069202   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   39  7.069208   172.27.5.141    224.0.1.129    PTPv2     Sync Message

+ Frame 1 (166 bytes on wire, 166 bytes captured)
+ Ethernet II, Src: Rockwell_3f:89:87 (00:00:bc:3f:89:87), Dst: IPv4mcast_00:01:81 (01:00:5e:00:01:81)
+ Internet Protocol, Src: 172.27.5.141 (172.27.5.141), Dst: 224.0.1.129 (224.0.1.129)
+ User Datagram Protocol, Src Port: ptp-event (319), Dst Port: ptp-event (319)
+ Precision Time Protocol (IEEE1588)

0000  01 00 5e 00 01 81 00 00  bc 3f 89 87 08 00 45 e0   ..^..... .?....E.
0010  00 98 18 b1 00 00 01 11  0c 9b ac 1b 05 8d e0 00   ........ ........
0020  01 81 01 3f 01 3f 00 84  d6 c2 10 02 00 2c 00 00   ...?.?.. ........
0030  02 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  bc ff fe 3f 89 87 00 02  27 a9 00 00 00 00 4b e0   ...?.... '.....K.
0050  6a 80 25 76 38 04 00 00  00 00 00 00 00 00 00 00   j.%v8... ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00                                  ......

File: "C:\DOCUME~1\CONTRO~1\LOCALS~1\Te...   Packets: 77 Displayed: 44 Marked: 0 Dropped: 0      Profile: Default
start   Command Prompt   (Untitled) - Wireshark   Keegan Screen Captu...      2:45 PM
```

Figure 20. Sync message from the master EN2T displayed in Wireshark



```
(Untitled) - Wireshark
File  Edit  View  Go  Capture  Analyze  Statistics  Help
Filter: ptp                                  Expression...  Clear  Apply

No. -  Time       Source          Destination    Protocol  Info
    1  0.000000   172.27.5.141    224.0.1.129    PTPv2     Announce Message
    2  0.000204   172.27.5.141    224.0.1.129    PTPv2     Sync Message
    3  0.009415   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
    4  1.009701   172.27.5.141    224.0.1.129    PTPv2     Sync Message
    5  1.019443   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
    9  2.019490   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   10  2.019496   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   11  2.029523   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   12  2.381951   192.168.168.31  224.0.1.129    PTPv2     Delay_Req Message
   13  2.381959   172.27.5.141    224.0.1.129    PTPv2     Delay_Resp Message
   16  3.029643   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   17  3.039483   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   21  4.039331   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   22  4.039337   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   23  4.049369   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   24  5.049592   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   25  5.059482   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   29  6.059331   172.27.5.141    224.0.1.129    PTPv2     Announce Message
   30  6.059338   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   31  6.069363   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   32  7.069384   172.27.5.141    224.0.1.129    PTPv2     Sync Message
   33  7.079392   172.27.5.141    224.0.1.129    PTPv2     Follow_Up Message
   37  8.079171   172.27.5.141    224.0.1.129    PTPv2     Announce Message

    subdomainNumber: 0
  + flags: 0x0200
  + correction: 0.000000 nanoseconds
    ClockIdentity: 0x0000bcfffe3f8987
    SourcePortID: 2
    sequenceId: 13472
    control: Sync Message (0)
    logMessagePeriod: 0
    originTimestamp (seconds): 1273068061
    originTimestamp (nanoseconds): 364550240

0000  01 00 5e 00 01 81 00 00  bc 3f 89 87 08 00 45 e0   ..^..... .?....E.
0010  00 98 f2 d1 00 00 01 11  32 7a ac 1b 05 8d e0 00   ........ 2z......
0020  01 81 01 3f 01 3f 00 84  69 8d 10 02 00 2c 00 00   ...?.?.. i.......
0030  02 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  bc ff fe 3f 89 87 00 02  34 a0 00 00 00 00 4b e1   ...?.... 4.....K.
0050  7a 1d 15 ba 98 60 00 00  00 00 00 00 00 00 00 00   z....... ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00                                  ......

originTimestamp (seconds) (ptp.v2.sdr.origintimestamp.seconds), 6 bytes   Packets: 1886 Displayed: 221 Marked: 0 Dropped: 0   Profile: Default
start   Command Prompt   (Untitled) - Wireshark   Keegan Screen Captu...      10:51 AM
```

Figure 21. IEEE 1588 48-bit seconds timestamp.


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter ptp
Expression... Clear Apply

```
No. -  Time       Source           Destination    Protocol  Info
    1 0.000000   172.27.5.141     224.0.1.129    PTPv2     Announce Message
    2 0.000204   172.27.5.141     224.0.1.129    PTPv2     Sync Message
    3 0.009415   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
    4 1.009701   172.27.5.141     224.0.1.129    PTPv2     Sync Message
    5 1.019443   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
    9 2.019490   172.27.5.141     224.0.1.129    PTPv2     Announce Message
   10 2.019496   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   11 2.029523   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   12 2.381951   192.168.168.31   224.0.1.129    PTPv2     Delay_Req Message
   13 2.381959   172.27.5.141     224.0.1.129    PTPv2     Delay_Resp Message
   16 3.029643   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   17 3.039483   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   21 4.039331   172.27.5.141     224.0.1.129    PTPv2     Announce Message
   22 4.039337   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   23 4.049369   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   24 5.049592   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   25 5.059482   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   29 6.059331   172.27.5.141     224.0.1.129    PTPv2     Announce Message
   30 6.059338   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   31 6.069363   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   32 7.069384   172.27.5.141     224.0.1.129    PTPv2     Sync Message
   33 7.079392   172.27.5.141     224.0.1.129    PTPv2     Follow_Up Message
   37 8.079171   172.27.5.141     224.0.1.129    PTPv2     Announce Message

    subdomainNumber: 0
  + flags: 0x0200
  + correction: 0.000000 nanoseconds
    ClockIdentity: 0x0000bcfffe3f8987
    SourcePortID: 2
    sequenceId: 13472
    control: Sync Message (0)
    logMessagePeriod: 0
    originTimestamp (seconds): 1273068061
    originTimestamp (nanoseconds): 364550240

0000  01 00 5e 00 01 81 00 00  bc 3f 89 87 08 00 45 e0   ..^..... .?....E.
0010  00 98 f2 d1 00 00 01 11  32 7a ac 1b 05 8d e0 00   ........ 2z......
0020  01 81 01 3f 01 3f 00 84  69 8d 10 02 00 2c 00 00   ...?.?.. i.......
0030  02 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  bc ff fe 3f 89 87 00 02  34 a0 00 00 00 00 4b e1   ...?.... 4.....K.
0050  7a 1d 15 ba 98 60 00 00  00 00 00 00 00 00 00 00   z....... ........
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00a0  00 00 00 00 00 00                                  ......

originTimestamp (nanoseconds) (ptp.v2.sdr.origintimestamp.nanoseconds), ...   Packets: 1886 Displayed: 221 Marked: 0 Dropped: 0   Profile: Default
```

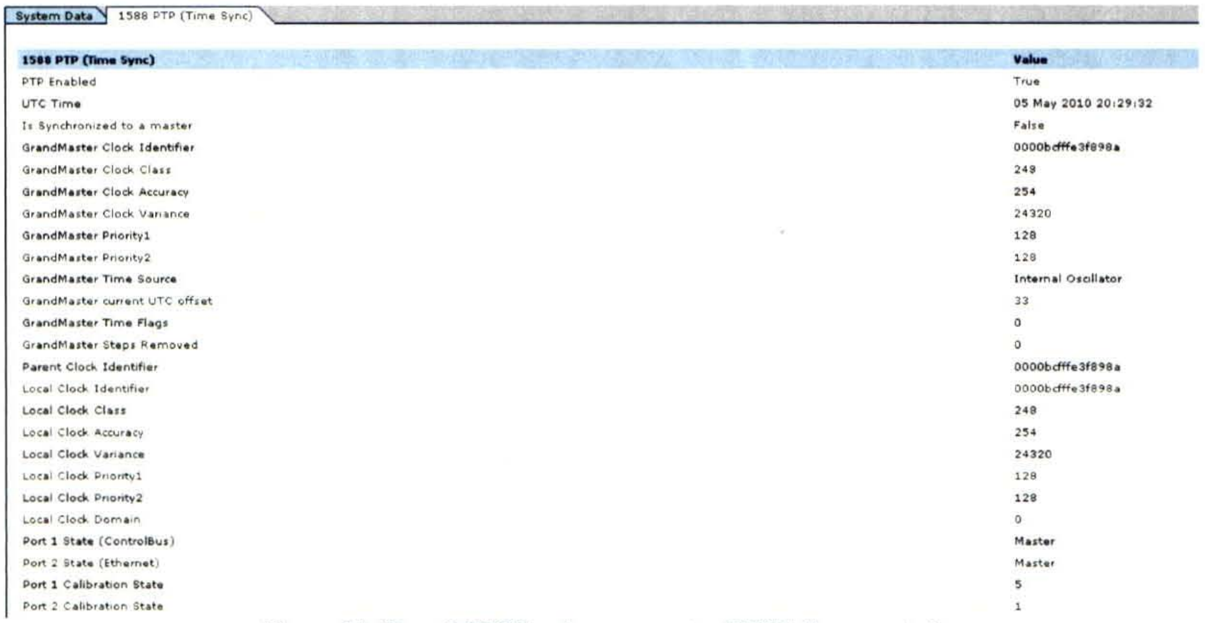
Figure 22. IEEE 1588 32-bit nanoseconds timestamp.

Port 1 Announce message count	0
Port 1 Sync message count	0
Port 1 Followup message count	103712
Port 1 Delay Req message count	0
Port 1 Delay Resp message count	0
Port 2 Announce message count	51968
Port 2 Sync message count	103511
Port 2 Followup message count	103500
Port 2 Delay Req message count	3291
Port 2 Delay Resp message count	3270

Figure 23. Review of slave 1 EN2T IEEE 1588 message count.

The final test to perform involved disconnecting the master EN2T and observing the communications that take place in establishing a new master. Once the master EN2T Ethernet cable was disconnected, various sync and follow-up messages were observed; however, these messages originated from the slave 1-4 EN2Ts rather than the master EN2T. Interestingly enough, after the negotiating back and forth, it was the Gateway server that consistently took over as the master (IP address 192.168.168.31). The master negotiation was usually completed within 10-13 seconds. Figure 24 shows the slave 3 EN2T webpage immediately after the master Ethernet cable was disconnected and before the Gateway server had taken over as master. The "Is Synchronized to a master" line now shows "False" and the "Grandmaster Clock Identifier," "Parent Clock Identifier," and "Local Clock Identifier" lines all show the same clock identity, namely, that of the slave 3 EN2T. Also note that the "Port 2 State (Ethernet)" line indicates "Master" as this EN2T is no longer connected to a designated master (in which case, this line would still be "Slave"). "The GrandMaster Time Source" has changed to "Internal Oscillator," which might suggest that the EN2T is serving as its own master.

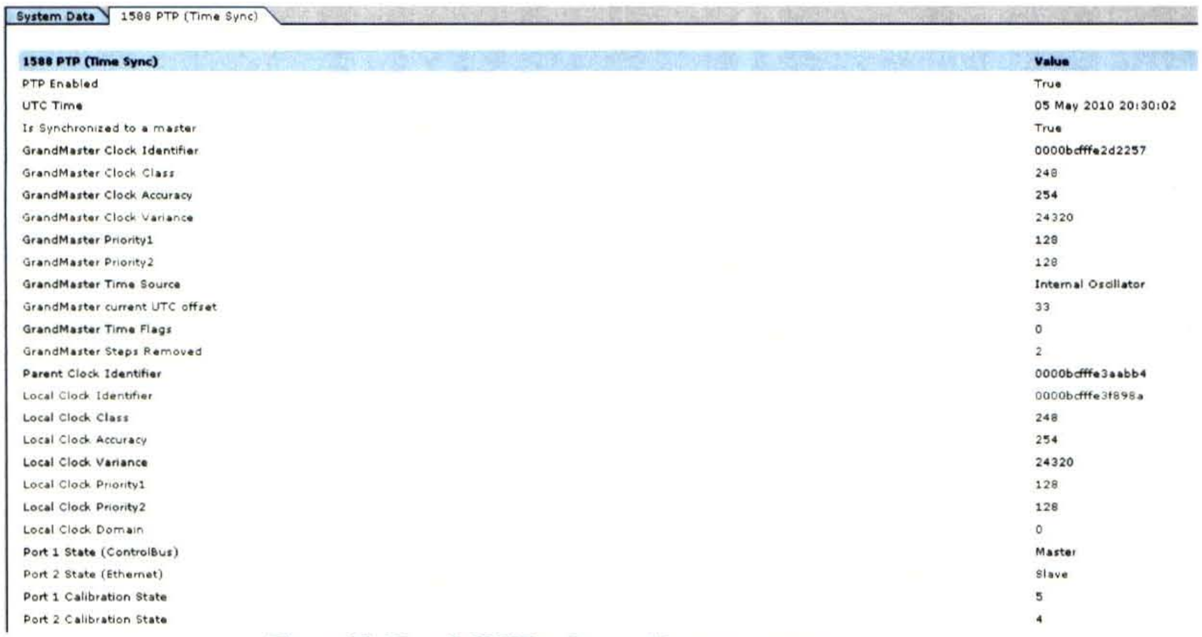
Thirty seconds later, Fig. 25 shows the result after the Gateway server has taken over as the master clock source. The "Is Synchronized to a master" line is now "True" and the "GrandMaster Clock Identifier" and "Parent Clock Identifier" lines have changed to different identities (the "GrandMaster Clock Identifier" is most likely that of the Gateway server), although the "Local Clock Identifier" has stayed the same (as expected, it is unique to the slave 3 EN2T). "Port 2 State (Ethernet)" has switched back to "Slave" now that the EN2T is once again synched to an outside master. For reasons not fully understood, the "GrandMaster Time Source" is still listed as "Internal Oscillator," even though the Gateway server is now the master clock source.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 20:29:32
Is Synchronized to a master	False
GrandMaster Clock Identifier	0000bcfffe3f898a
GrandMaster Clock Class	249
GrandMaster Clock Accuracy	254
GrandMaster Clock Variance	24320
GrandMaster Priority1	128
GrandMaster Priority2	128
GrandMaster Time Source	Internal Oscillator
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	0
Parent Clock Identifier	0000bcfffe3f898a
Local Clock Identifier	0000bcfffe3f898a
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Master
Port 1 Calibration State	5
Port 2 Calibration State	1

Figure 24. Slave 3 EN2T webpage, master EN2T disconnected.



System Data | 1588 PTP (Time Sync)

1588 PTP (Time Sync)	Value
PTP Enabled	True
UTC Time	05 May 2010 20:30:02
Is Synchronized to a master	True
GrandMaster Clock Identifier	0000bcfffe2d2257
GrandMaster Clock Class	248
GrandMaster Clock Accuracy	254
GrandMaster Clock Variance	24320
GrandMaster Priority1	128
GrandMaster Priority2	128
GrandMaster Time Source	Internal Oscillator
GrandMaster current UTC offset	33
GrandMaster Time Flags	0
GrandMaster Steps Removed	2
Parent Clock Identifier	0000bcfffe3aabb4
Local Clock Identifier	0000bcfffe3f898a
Local Clock Class	248
Local Clock Accuracy	254
Local Clock Variance	24320
Local Clock Priority1	128
Local Clock Priority2	128
Local Clock Domain	0
Port 1 State (ControlBus)	Master
Port 2 State (Ethernet)	Slave
Port 1 Calibration State	5
Port 2 Calibration State	4

Figure 25. Slave 3 EN2T webpage, Gateway server as master.

Upon reconnecting the master EN2T Ethernet cable, master renegotiation once again took place, although this time it was generally about 9 seconds before the master EN2T had reestablished itself as the master clock source. This verified that the RSLogix code succeeded in successfully establishing the local EN2T as the master every time that it was connected to the network (remember, the code sets the local EN2T priority to 1).

One extra test that I decided to perform highlighted an anomalous result that must be examined further. I changed the local controller's WCT to June 5, 2011 to see if the master and slave clocks correctly changed to this new time. In RSLogix, the "CurrentTimeMicroseconds" tags changed to 33371 for the most significant DINT and -1555315787 for the least significant DINT, indicating a large increase in seconds, which was to be expected for moving the data 13 months forward in time. However, the tags within the slave project files did not update to this value. The slave tags stayed at their previous values, counting on May 5, 2010. The EN2T webpages all still showed a UTC time on May 5, 2010. Within Wireshark, one of the sync message time stamps contained a 48-bit seconds portion of 127309661, indicating May 5, 2010 once again. If the RSLogix project files were working correctly, resetting the local controller's WCT should change all of these timing values to the new set time. This was what happened when I originally set the local controller's WCT to May 5, 2010, but is unclear why setting the time forward 13 months did not change all of the timing values again.

VI. Conclusion

These test results will prove useful for the Constellation Program's KGCS or any other ground support system at KSC, regardless of the Constellation Program's future. Further tests must be performed under more realistic conditions to accurately measure the latency and jitter with the KGCS' PLC network. The latency and jitter tests that I conducted merely demonstrated that the PLCs themselves produced the most time delay and unpredictability in the network. Since the actual KGCS network will have many local and RIO chassis connected, I would recommend performing tests with many RIO modules. Including more modules, each with their own RPIs that must be scheduled along with the NUT, may lead to further time delays that will be important to understand in the operation of the KGCS.

The timing and synchronization tests verified the IEEE 1588 communications between the master and slave EN2Ts. Synch messages were sent from the master EN2T to the slave EN2Ts once per second, with the time stamp stored in 48-bit second and 32-bit nanosecond portions of the data packet. The master RSLogix project file successfully set the local EN2T as the master, with the local EN2T establishing itself as the master any time that it was connected to the network. The 64-bit microsecond UTC time stored within the two DINTs was correctly synched between the master and all of the slaves. Code can be easily written to add these two DINTs to the data packets sent to and from the LCC. Considering the 64-bit microsecond UTC time was correctly synched to the slave RSLogix project files, time stamping could be done at the RIO chassis, though this would consume more bandwidth. Whether data packets are time stamped at the local controllers or the RIO chassis is also dependent on the desired level of time stamp accuracy. Further testing remains in connecting the actual IRIG-B line into the Hiprom module, rather than simply setting the local controller's WCT as done in this test scenario. That will be a more accurate simulation of the KGCS' operation, since it will actually use the GPS signal fed through the IRIG-B line.

It is unclear why the Gateway server consistently took over as the master rather than one of the slave EN2Ts whenever the master EN2T was disconnected. In actual operation, the KGCS will most likely contain a second Hiprom module with a redundant master EN2T. Testing must be performed to understand how the redundant master EN2T will ensure that it gains mastership in the event that the true master EN2T is disconnected from the network. Perhaps setting the redundant master EN2T's priority to a number such as 2 (anything greater than 1) will allow it to seamlessly gain mastership.

Appendix

Table 1. Latency and Jitter Test Measurements. *(All times in milliseconds)*

Test Run	CNet RPI	IB16 RPI	OB8 RPI	NUT	Mean	Min	Max	Jitter	Sdev	Num
1	10	10	10	10	18.49775	13.93789	23.94103	10.00314	2.743898	416
2*	10	10	10	10	18.85436	13.9798	23.9997	10.0199	2.9037	920
3	15	15	15	10	18.9922067	13.88858	23.99545	10.10687	2.9157617	240
4	15	10	10	10	18.8476676	13.94968	24.21920	10.26952	3.0826042	508
5	50	10	10	10	33.9632022	13.93435	53.97385	40.0395	11.5566852	932
6	10	10	10	10	19.0311232	13.93227	23.99451	10.06224	2.7911976	252
7	5	10	10	5	11.4743518	8.82689	13.92154	5.09465	1.488855	528
8	10	10	10	5	14.0318095	8.81429	18.87515	10.06086	2.9231947	248
9	5	10	10	5	11.3907800	9.00538	13.87184	4.86646	1.4178859	120
10	10	10	10	10	19.0485609	14.00528	23.96837	9.96309	2.8996222	244
11	10	10	10	10	18.9560578	13.92333	23.97040	10.04707	2.8866426	232
12	10	10	10	10	18.5818042	13.94683	23.98406	10.03723	2.8374690	208
13	10	10	10	10	18.5965723	13.94688	23.90098	9.9541	2.7612430	228
14	10	10	10	10	18.9779382	13.96821	24.02265	10.05444	2.9771858	248
15	10	10	10	10	18.8074736	13.89259	23.82361	9.93102	2.9145254	232
16	10	10	10	5	14.13174291	9.03429	18.85272	9.81843	2.8242465	222
17	10	10	10	2.5	21.4667808	16.48512	26.44103	9.95591	2.8478809	240
18	10	10	10	2	21.7063236	16.51443	26.45197	9.93754	2.9419118	208
19	10	10	10	2	21.4073218	16.52802	26.52379	9.99577	2.8835176	224
20	10	10	10	3	21.5254664	16.55163	26.48281	9.93118	2.8885815	232
21	10	10	10	4	21.5336290	16.56085	26.51604	9.95519	2.9022578	232
22	10	10	10	5	21.4067819	16.40067	26.46037	10.0597	2.9033321	240
23	10	10	10	6	21.5281259	16.52009	26.45726	9.93717	2.8916269	240
24	10	10	10	8	21.5392955	16.49247	26.58923	10.09676	2.8860247	236
25	10	10	10	8	18.8873087	13.91596	23.90899	9.99303	2.8553402	240
26	10	10	10	2	18.9072305	13.95129	23.90899	9.9577	2.5649835	300
27	10	10	10	5	19.5758522	13.89700	23.98612	10.08912	2.8703814	300
28	10	10	10	2	18.975891	13.98050	23.92894	9.94844	2.8623880	228
29	10	10	10	5	13.9291956	9.00599	18.90289	9.8969	2.8874951	232
30	10	10	10	2	17.4752607	13.46610	21.52191	8.05581	2.2457791	196
31	10	10	10	2	17.5598902	13.46006	21.42116	7.9611	2.3340320	192
32	10	10	10	3	10.5542447	6.88057	18.01098	11.13041	2.2290055	312
33	10	10	10	2	25.4306889	21.46565	29.0025	8.3346	2.3247264	184
34	10	10	10	5	24.0511452	18.85277	28.99734	10.14457	2.9608781	244
35	10	10	10	7	16.9995866	10.94310	24.95497	14.01187	4.0974194	252
36	10	10	10	9	21.0861460	12.85179	30.92694	18.07515	5.1963023	252
37	10	10	10	10	18.9297489	13.88071	23.95761	10.0769	2.8211689	252
38	10	10	10	2	21.4962566	13.46182	29.43626	15.97444	4.5284803	396
39	10	10	10	10	19.1141197	13.99129	23.91492	9.92363	2.8476010	252
40	10	10	10	2	19.5812915	13.46026	29.47004	16.00978	4.5842105	472
41	10	10	10	10	28.9992358	23.93201	33.98043	10.04842	2.8695171	484
42	10	10	10	2	17.6169534	13.46693	21.58592	8.11899	2.415361	500
43	10	10	10	5	13.6662618	8.86377	18.95187	10.0881	3.0168843	516
44	10	10	10	7	18.8335998	10.98122	24.98992	14.0087	3.4936607	372
45	10	10	10	9	20.9621513	12.88946	30.93100	18.04154	4.9594706	464
46	10	10	10	10	18.1544895	13.87298	23.99940	10.12642	2.8056858	360
47	10	10	10	2	18.6230866	13.46799	29.50015	16.0316	4.4887042	372

Test Run	CNet RPI	IB16 RPI	OB8 RPI	NUT	Mean	Min	Max	Jitter	Sdev	Num
48	10	10	10	5	24.0257228	18.82100	28.96982	10.14882	3.0921598	544
49	10	10	10	9.5	21.5212609	13.47557	32.42338	18.94781	5.5271953	452
50	10	10	10	5	23.8215433	18.89830	28.93302	10.03472	2.8462151	456
51	10	10	10	7	18.1406772	10.93419	24.99672	14.06253	3.995184	444
52	10	10	10	10	18.1547620	13.89016	24.01314	10.12298	2.9714172	312
53	10	10	10	9	20.9137687	12.98423	30.40336	17.91913	5.4331450	384
54	10	10	10	5	24.237853	18.90366	29.00009	10.09643	3.2427983	312
55†	10	10	10	10	21.2178960	13.98026	34.24797	20.26771	5.0642212	2,000
56	10	10	15	10	19.1362355	13.92755	24.06829	10.14074	2.9158596	2,000
57	10	15	10	10	27.4353915	14.02887	34.06390	20.03503	4.6629030	1,200
58	15	10	10	10	21.4012325	13.96736	34.09551	20.12815	5.0476602	2,000
59	15	10	15	10	19.0561108	13.97107	24.07084	10.09977	2.8850225	1,600
60	15	15	10	10	19.6009793	13.93634	28.68453	14.74819	3.4473884	1,200
61	10	10	15	10	24.1518155	13.91676	34.07583	20.15907	5.8138599	1,000
62	15	10	15	10	18.7998455	13.91356	24.00536	10.0918	2.7793589	800
63	15	10	20	10	23.9021478	14.03923	33.99573	19.9565	5.5980853	500
64	20	10	10	10	19.0278361	13.86605	24.00887	10.14282	2.8827965	1,560
65	25	10	10	10	29.0585773	23.87661	33.96900	10.09239	2.6668587	600
66	15	15	15	10	18.8415854	13.91499	24.02897	10.11398	2.8874074	1,100
67	10	11	12	10	18.9665261	13.95110	24.10566	10.15456	2.8022222	1,000
68	10	11	11	10	19.0798167	13.89130	24.04535	10.15405	2.8186088	1,000
69	10	10	10	10	19.0114637	13.91331	24.07885	10.16554	2.8904831	1,860
70*	10	10	10	10	18.95100281	13.913909	24.072793	10.158884	2.92819604	1,720
71	10	10	10	10	22.6325675	13.87375	34.05937	20.18562	5.6217632	3,828

*10 Hz input signal
† Tests #55-71 were performed within a 10 ms periodic task; the previous tests were conducted within a continuous task.

Acknowledgments

I would like to thank my KSC mentor, Reggie Martin, for assisting me in conducting these tests and Elias Victor, for providing me with valuable information and insights on previous PLC timing and synchronization testing. Finally, I would like to thank Mark Schillace from Rockwell Automation for originally providing the RSLogix master and slave project files needed for carrying out my tests.

References

[1]"ControlLogix System Fundamentals - Student Manual." Rockwell Automation, Inc. 2005.
[2]Victor, Elias. "Trade Study Report, PLC Time Tagging - 700ETD00004." Kennedy Space Center. 10 April 2009.
[3]"Kennedy Ground Control Subsystem Operational Concept Document (OCD)." Kennedy Space Center. 9 May 2008.
[4]"Latency and Jitter." NetQoS, Inc. 23 June 2008. <http://www.networkperformancedaily.com/2008/06/latency_and_jitter_1.html>.
[5]"1756HP-GPS Datasheet." HIPROM Technologies.
[6]McCarthy, Alex. " Special Focus: Understanding the IEEE 1588 Precision Time Protocol." National Instruments. 2010. < http://zone.ni.com/devzone/cda/pub/p/id/130>.