Software development for the Hobby-Eberly Telescope's segment alignment maintenance system using LabVIEW

Hall, Drew, Ly, William, Howard, Richard, Weir, John, Rakoczy, John

Drew Hall, William Ly, Richard T. Howard, John T. Weir, John M. Rakoczy, "Software development for the Hobby-Eberly Telescope's segment alignment maintenance system using LabVIEW," Proc. SPIE 4848, Advanced Telescope and Instrumentation Control Software II, (13 December 2002); doi: 10.1117/12.461269

Event: Astronomical Telescopes and Instrumentation, 2002, Waikoloa, Hawai'i, United States

Software development for the Hobby-Eberly Telescope's Segment Alignment Maintenance System using LabVIEW

Drew Hall, William Ly, Ricky Howard, John Weir, John Rakoczy
NASA Marshall Space Flight Center
Huntsville, Alabama

ABSTRACT

The software development for an upgrade to the Hobby-Eberly Telescope (HET) was done in LabVIEW. In order to improve the performance of the HET at the McDonald Observatory, a closed-loop system had to be implemented to keep the mirror segments aligned during periods of observation. The control system, called the Segment Alignment Maintenance System (SAMS), utilized inductive sensors to measure the relative motions of the mirror segments. Software was developed in LabVIEW to tie the sensors, operator interface, and mirror-control motors together. Developing the software in LabVIEW allowed the system to be flexible, understandable, and able to be modified by the end users. Since LabVIEW is built using block diagrams, the software naturally followed the designed control system's block and flow diagrams, and individual software blocks could be easily verified. LabVIEW's many built-in display routines allowed easy visualization of diagnostic and health-monitoring data during testing. Also, since LabVIEW is a multi-platform software package, different programmers could develop the code remotely on various types of machines. LabVIEW's ease of use facilitated rapid prototyping and field-testing. There were some unanticipated difficulties in the software development, but the use of LabVIEW as the software "language" for the development of SAMS contributed to the overall success of the project.

Keywords: LabVIEW, software development, segmented mirror, mirror control, HET

INTRODUCTION

The Hobby-Eberly Telescope (HET) is a 9.2-m fixed elevation telescope with a segmented primary mirror. It is located at McDonald Observatory in West Texas at an elevation of 2,008m. Descriptions of the telescope and its operation may be found at the indicated references[1,2]. During initial testing of the telescope after first light, composite star image spots formed by individual mirror segments were observed to "de-stack", or move with respect to each other, over a period of time after the segments had been "stacked", or aligned with each other. While minor segment motion over a period of an hour or more had been anticipated in the original design, misalignment of the segments on a time scale of tens of minutes under some conditions was unexpected.

In November 1999, the University of Texas at Austin entered into a Space Act Agreement with NASA's Marshall Space Flight Center (MSFC) to procure a Segment Alignment Maintenance System (SAMS) for the HET[3,4]. The objective of the SAMS is to correct the effects of the de-stacking phenomenon, maintaining primary mirror segment alignment. MSFC teamed with Blue Line Engineering of Colorado Springs, Colorado. Blue Line provided the edge sensing system and electronics. MSFC developed control algorithms and control system software. MSFC also managed system integration and verification testing.

The SAMS consists of inductive edge sensors, sensor electronics, and a central control computer hosting a suite of software applications. Details concerning SAMS hardware, architecture and control system design are reported elsewhere in the literature[3,4]. The main function of the SAMS software is to acquire edge sensor measurements from the edge sensor electronics, utilize optimal control algorithms to compute tip, tilt and piston correction commands, and communicate the tip, tilt and piston commands to HET's Primary Mirror Computer (PMC). The SAMS software has ancillary functions of monitoring edge sensor health, reporting and correcting errors, archiving engineering data and supporting calibration and troubleshooting. This paper describes the software applications developed for the SAMS central control computer. The first section will describe the justification for the technical approach chosen to develop the software. The second section will describe the design and implementation of the software that was developed.

1. JUSTIFICATION FOR TECHNICAL APPROACH

The technical approach employed in developing the SAMS software was driven by HET's requirements. Among the drivers for software development was low-bandwidth operation. The dynamic environment of HET had a bandwidth sufficiently low that a high-speed system was not required. Another driver was ease of use. The software had to be easy to use with minimal interaction on the part of the telescope operator (TO), because the HET TO has many subsystems to operate and maintain each night. Reliability was important because observing time is precious and nobody wanted to waste observing time on recovering from software faults. Maintainability of the software was essential so that HET staff could easily follow the code and modify it to accommodate facility upgrades and system improvements. A final and very important driver was the requirement for rapid prototyping. The SAMS project was on a very ambitious schedule which required most functional software to be operational and verified in order to support the seven-segment Sub Array Test (SAT). The SAT was scheduled only 36 weeks after the start of the program. So it was essential that the development environment supported rapid development and verification.

1.1 Low-bandwidth operation

The SAMS systems requirements placed no challenging real-time requirements on the SAMS software itself. The only requirement driving system bandwidth was that the HET's Primary Mirror Computer required mirror command updates from SAMS once every 10 seconds. That requirement flowed down to accessing the edge sensor electronics for edge sensor measurements once per second. The relatively low bandwidth of the data being processed (1Hz) and the computation of the required data to be used (~0.1 Hz) by the PMC allowed flexibility in the choice of the development software and the design of the necessary applications. The applications needed to be multi-threaded to accommodate the multiple tasks the software would be required to perform. The software did not require any unique process-scheduling code or real-time code to insure the successful operation of the SAMS control system.

1.2 Ease of Use

Creating a set of easy to use and functional applications was one of the main goals of the SAMS software developers. To accomplish this, the applications were designed to be graphical-based and would require a minimal amount of telescope operator interaction. The SAMS software needed to provide the telescope operator an intuitive graphical user interface. This required the interface to provide the ability to point and click through the setup and operation of the SAMS software. Each mode of the software's operation required a graphical interface with command buttons containing descriptive labels. This approach eliminated the need to remember or look up text-based commands to use the SAMS software. Once operating, the software needed to provide the operator the ability to determine the overall health of the SAMS system but not require the operator to continuously adjust any aspect of the software in order to maintain the SAMS system performance. The SAMS software was designed in such a way that the telescope operator could provide the SAMS software an initial stacked mirror configuration and place the SAMS software into the operating mode. The software would then operate without any further interactions until a new mirror stack was required.

1.3 Reliability

In designing the SAMS system it was extremely important that its operation be reliable during nightime observations. To achieve the reliability needed, the system and software were designed around established and well-supported commercial off-the-shelf (COTS) hardware and software components. The COTS products that were used in the design and development of the SAMS system and software were the SUN/Solaris computer configuration, 100Base-T Ethernet communication medium, Transmission Control Protocol/ Internet Protocol communication protocol (TCP/IP) utilizing the Berkley socket abstraction, and National Instruments LabVIEW software development package.

The SUN/Solaris host configuration was chosen, in part, because it was one of the acceptable platforms identified in the SAMS specification. This configuration also supported the communication medium requirements contained in the SAMS system requirements. The SUN/Solaris configuration provided a proven and reliable hardware platform and operating system combination. The maturity of, and support available for, this configuration provided confidence in the design and development of the software and operation of the final system.

The SAMS specification required that the SAMS software communicate over either 10Base-T or 100Base-T Ethernet. The HET facility utilizes the 10/100Base-T Ethernet for its network communications. The 100Base-T Ethernet, or Fast Ethernet, network communication medium has been in widespread use since the late 1990's[5]. This communication medium provides a fast and reliable communication link for network traffic. The SUN/Solaris host computer provided the hardware connection and software support for a 100Base-T network interface. The software design relied on the 100Base-T interface for all communications.

The specification required that the SAMS software communicate using the TCP/IP protocol utilizing the Berkley socket abstraction. This TCP/IP communication protocol has been in service since 1977. The Berkley socket abstraction has been a part of the TCP/IP protocol since 1982[6]. The use of TCP/IP socket communications provided a standard, reliable communication mechanism on which to build the software interfaces needed to create a reliable set of applications.

While there were several software development environments available for the SUN/Solaris platform, the SAMS software design team chose National Instruments' LabVIEW software development package. The LabVIEW package was chosen because of its ease of use, strong graphical resources, and network support. LabVIEW offers an intuitive graphical programming language in which to develop the flow of a software application. The language does not require knowledge of obscure text-based function calls. Programming is accomplished using graphical programming objects. These objects are made available to a developer on a palette containing all valid programming resources. The function of each object is graphically depicted with a textual description also available. The LabVIEW development environment also provides the developer with a suite of graphical objects that can be used to build a user interface. These graphical controls and indicators are customizable and provide a rapid visualization capability. LabVIEW provides programming objects that support the TCP/IP socket network communications.

The SAMS software development team had to insure that no catastrophic or spurious mirror corrections could be sent from the SAMS system to the PMC. Extremely large mirror commands implemented by the PMC would, at the very least, send segments to the actuator limits, requiring extensive time for recovery. At worst, spurious commands could damage the mirror segments. Damaging a segment would cost the observatory money and result in lost observation time. The software design incorporated checks to insure that sensors providing erroneous data be detected and that the subsequent mirror correction commands not be used by the PMC.

1.4 Maintainability

The SAMS software applications needed to be maintainable. This maintainability re quired that the software applications be expandable, provide adequate troubleshooting resources, and be readable by personnel from other engineering disciplines. These capabilities needed to be available during development and operation.

During development the software design allowed for new programming objects to be developed and inserted in the program path where needed. This design allowed the software to be expanded with minimal impact. The LabVIEW development language itself provided many troubleshooting resources during development. Some of the resources provided by LabVIEW were the ability to set breakpoints, monitor variables, compute resources used, and observe program execution.

During operation the design allowed for multiple legal configurations of the HET mirror array and the SAMS electronic sensors. The SAMS software was designed to provide the user access to, and control of, the SAMS resources while in operation. The software design provided the user the ability to monitor and control the HET mirror array configuration, all SAMS sensors, and the resulting corrections being computed. This capability was intended to provide the user the ability to troubleshoot all aspects of the SAMS system during operation.

To illustrate the impact LabVIEW's readability has on the software's maintainability refer to Figures 1 and 2. Figure 1 is a depiction of the control flow developed by the SAMS control engineer. LabVIEW's graphical programming language provides the ability to implement control diagrams and present them in a way that is understandable by those unfamiliar with conventional text-based programming languages. Figure 2 contains the LabVIEW implementation of the control flow diagram. The LabVIEW code clearly captures the original control flow diagram's correlation between the control variables and the summing junctions. During the development and the early phases of initial operation for the SAMS software this feature of LabVIEW contributed to the success of the SAMS development. This feature enabled the software developers to quickly answer and explain questions concerning control algorithm implementation in LabVIEW. This readability feature will also benefit those who were not members of the original development team. This feature will allow those who may have to maintain the software in the future to better "see" and understand the flow of the software.

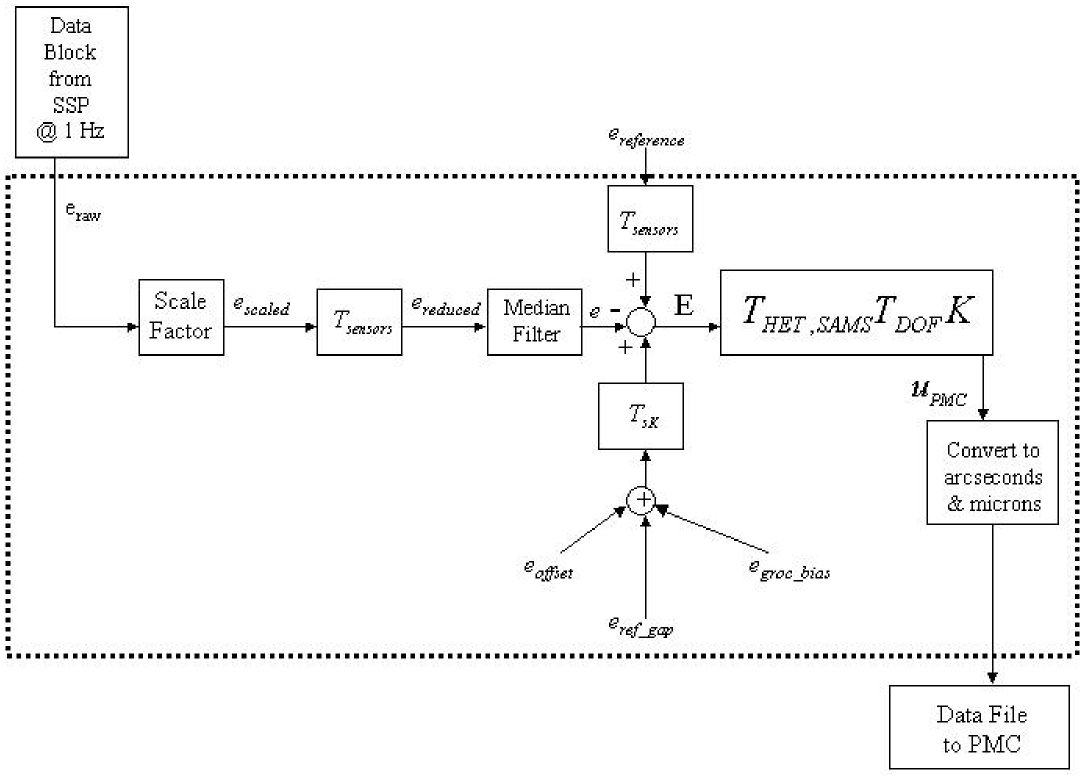


Figure 1: Control System Flow Diagram

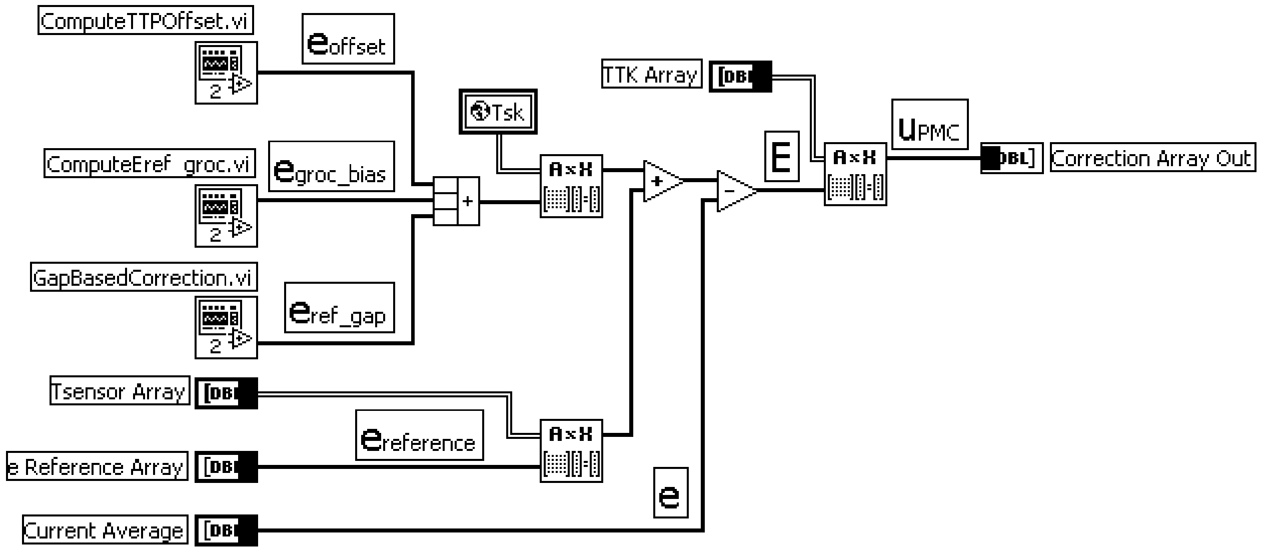


Figure 2: Control Loop Implementation in LabVIEW

The development schedule of the SAMS system required that a functional version of the software be available to support Sub-array test (SAT) before the critical design review was held. The program schedule mandated a preliminary design review (PDR) 24 weeks into the program, the SAT 36 weeks into the program, and a critical design review (CDR) 40 weeks into the program. At a typical CDR, 80% of software design has been completed, not necessarily developed and verified. The SAT was a seven-segment demonstration of the SAMS system. SAT required functional software to demonstrate control system operation. The same functional software would be extended to the full 91-segment array, except that, for the SAT, the size of the control matrices would be much smaller. The position of the SAT in the critical path of the project necessitated that 80% of all functional software be developed and verified prior to CDR. This required the software development team to rapidly develop a prototype version of the SAMS software applications. The resources available within LabVIEW allowed the software developers to efficiently develop the network communications and graphical visualization necessary to successfully support this pre-CDR test.

2. SOFTWARE IMPLEMENTATION

The SAMS software was developed on a SUN/Solaris platform using National Instruments' LabVIEW software development tool. The SAMS software consists of two applications, the SAMSServer and the SAMSGui. Figure 3 illustrates the relationship of the two SAMS software components and other systems within the Hobby-Eberly Telescope's Local Area Network (LAN). All data into and out of the SAMS software is transferred using socket port connections over a 100Base-T LAN and conform to the TCP/IP (IP version 4) protocol and the Berkeley Socket abstraction. The SAMSServer application acts as the data server within the SAMS system. The SAMSServer collects and processes data from the SAMS Stackable System Processor (SSP) and provides this data to client applications. Other SAMS clients, including the SAMSGui application and the HET's PMC, can request data from the SAMSServer application.

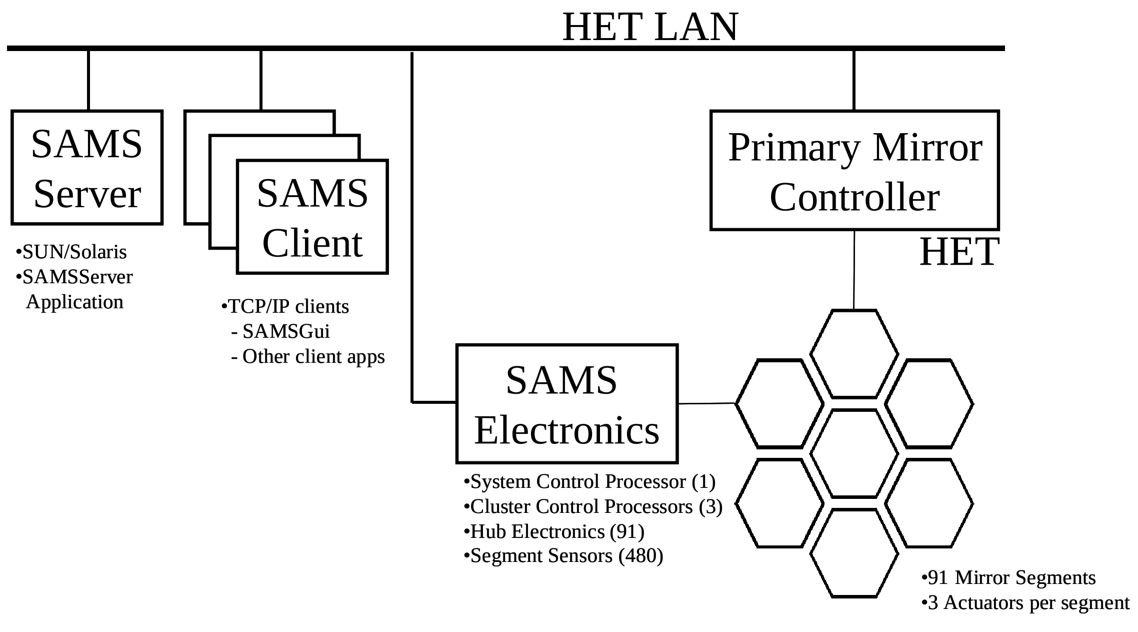


Figure 3: SAMS Software interfaces to SAMS and HET subsystems

2.1 SAMSGui Application and Mode Transition

The SAMS software operates within four distinct operating modes: Standby, Operate, Calibrate, and Diagnostic. The SAMS operating mode is managed by the SAMSServer application and controlled by the SAMSGui application. The SAMSGui can request the SAMSServer to transition from one mode to another. If appropriate, the SAMSServer will transition to the mode requested. The operating modes and their transition paths are depicted in Figure 4. A description of each mode follows.

The SAMS applications initially start in Standby mode. Standby mode is an executive mode which governs transitions to all other modes and also serves as a parking mode from which the SAMS system can be reconfigured. From Standby mode the application can transition to any of the three remaining modes of operation as shown in Figure 4. Standby mode provides the ability to modify and control the SAMS configuration. From Standby mode the operator can remove or install sensors, segments, or degrees of freedom for a segment. From Standby mode the operator can also set the current mirror configuration as the reference to be used by SAMS. During the transition to Operate mode, the SAMS software computes all the control system matrices based on the segment, sensor, and degrees of freedom configuration specified in Standby mode.

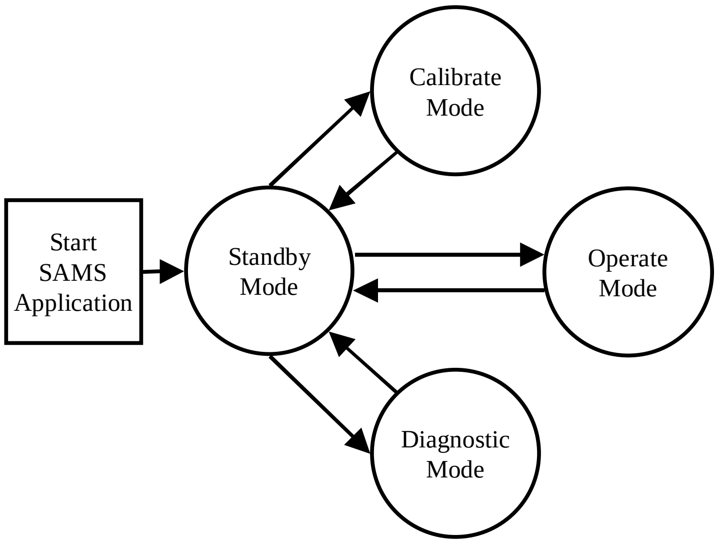


Figure 4: Conceptual operation of mode transitions with SAMSGui

In Operate mode, the SAMSServer acquires edge sensor measurements from the electronics stack over the SAMS electronics server port. Figure 1 illustrates data flow within Operate mode. Edge sensor measurements are sampled approximately once per second. The sensor data are time-tagged and archived on the system's hard disk. The Operate mode software then applies scale factors to the data, strips out missing sensors, identifies faulty sensors, and filters the remaining data. Next, a control error signal is computed by combining the current sensor readings with reference values that are either user-specified or obtained from the global radius of curvature (GRoC) compensation loops[8,9]. The control error signal is operated on by a series of matrix operations which produce mirror segment motion commands in the prescribed units in the correct reference system. The mirror commands are written to a file and updated every 17 seconds. It is during Operate mode that the PMC acquires current commands to maintain the HET's mirror array alignment.

Calibrate mode provides the ability to perform Influence Matrix and Edge Sensor calibrations, set calibration coefficients in the SAMS electronics, and command the SAMS electronics to save previously set calibration coefficients to non-volitile memory. Calibration tests are performed by sending actuator move commands to the PMC via the client port. Sensor data acquired during the calibration tests are time-tagged and archived on the SUN workstation's hard disk. Software external to the SAMS software post-processes the data to derive the edge sensor scale factors and empirically-determined influence matrix. Downloading coefficients to the SAMS electronics and saving coefficients to the electronics' non-volatile memory is accomplished through the SAMS electronics server port.

Diagnostic mode provides access to all SAMS sensor data and its intended use is for debugging SAMS sensor anomalies. Diagnostic mode allows users to get a quick look at current data values in real-time. Data available for viewing under Diagnostic mode include edge sensor measurements, local sensor temperatures, and segment electronics temperatures.

Within the SAMS software, the SAMSGui application provides the SAMS operator a graphical interface with which to control and monitor the operation of the SAMS system. SAMSGui provides a graphical interface for each mode of the SAMS system. Each of these mode-specific graphical interfaces contains clearly labeled buttons that can be selected by the operator. These buttons correspond to the commands defined in the SAMS interface control document[7]. Each button, when pressed, initiates a communication cycle with the SAMSServer. The SAMSGui communicates with the SAMSServer over a TCP/IP socket connection. Through the mode-specific graphical interfaces and the communication link with the SAMSServer the SAMSGui allows the operator to configure and monitor the SAMS system. Figure 5 shows the graphical display of the SAMSGui application in Standby mode.

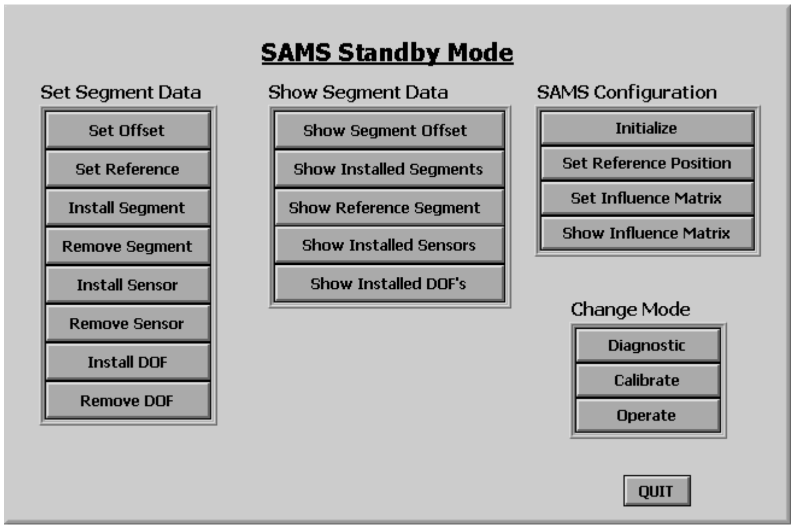


Figure 5: SAMSGui Standby mode graphical interface

In Figure 5, the SAMSGui display has four separate sections, each with special functionality. The “Set Segment Data” section allows the operator to install or remove sensors, segments or degrees of freedom besides setting control system offsets and references. The buttons in the “Show Segment Data” section bring up new displays with text boxes giving snapshots of current SAMS engineering data. The “SAMS Configuration” buttons allow the operator to initialize the SAMS system and set new sensor references when the mirrors are aligned. In the “Change Mode” section of the display there are three buttons. The Diagnostic, Calibrate and Operate buttons execute the mode transitions illustrated in Figure 4.

When the operator selects the Operate mode button in Figure 5, the display in Figure 6 appears on the terminal screen. From the buttons in Figure 6, the operator may select one of several engineering data display buttons. The operator may view the latest tip, tilt and piston commands that SAMS has computed (“Show TTP”). The operator may view edge

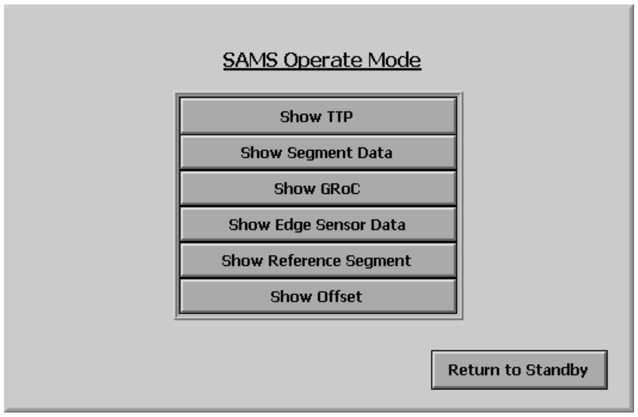


Figure 6: SAMSGui Operate mode graphical interface

sensor data by segment ("Show Segment Data"), or he may view the entire array of edge sensor values ("Show Edge Sensor Data"). The operator can also view current control system offsets ("Show Offset"), the current reference or boundary condition segments ("Show Reference Segment") or global radius of curvature sensor measurements ("Show GRoC"). The "Return to Standby" button on the lower right corner of Figure 6 transitions the SAMSServer application back to Standby mode. The SAMSGui application then returns to the Standby mode display shown in Figure 5.

2.2 SAMSServer Application

The SAMSServer application is the workhorse of the SAMS software package. Figure 7 depicts the different interfaces managed by the SAMSServer. The SAMSServer application serves as the communication hub between remote users (including the SAMSGui application), the SAMS SSP's System Control Processor (SCP), and the HET's PMC. The SAMSServer also oversees the archival of all SAMS data, computes mirror correction commands, and provides textual and graphical displays showing the health and status of the SAMS system.

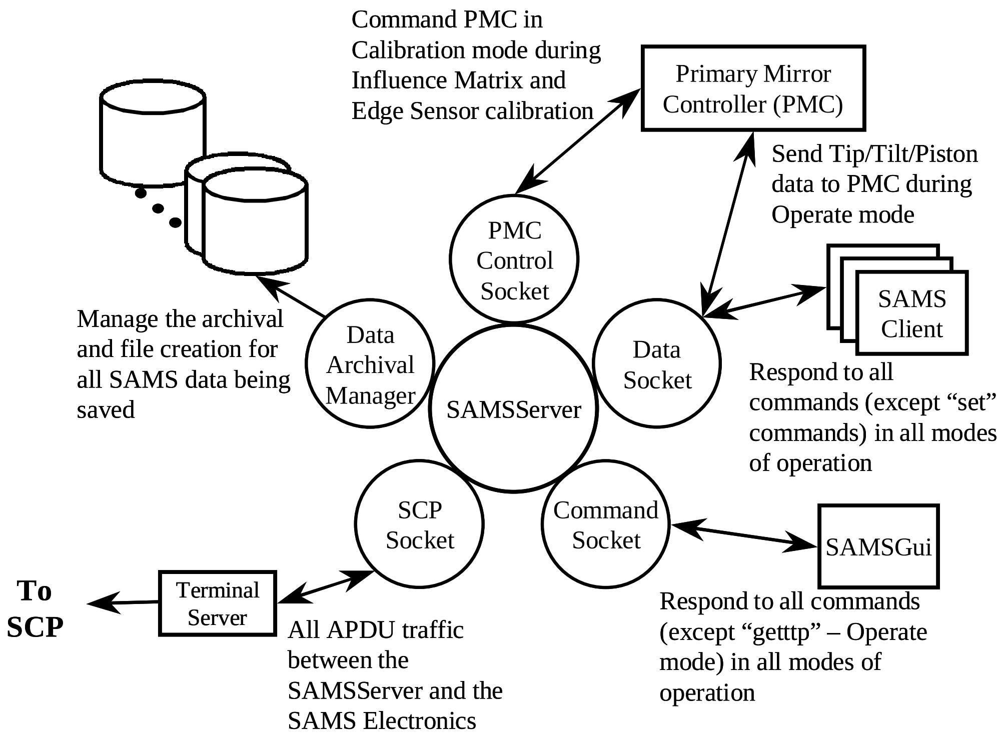
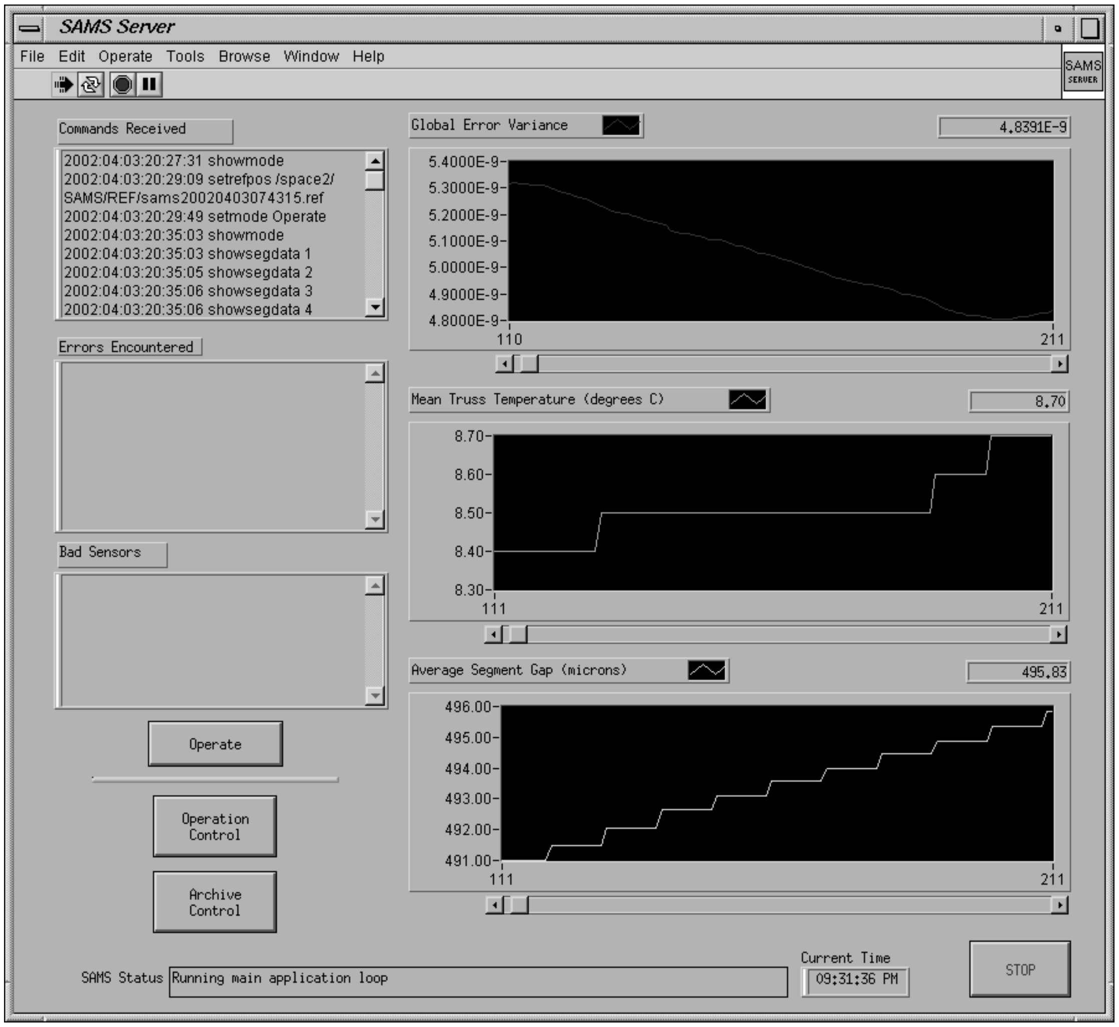

Command PMC in Calibration mode during Influence Matrix and Edge Sensor calibration
Primary Mirror Controller (PMC)
Send Tip/Tilt/Piston data to PMC during Operate mode
PMC Control Socket
SAMS Client
Manage the archival and file creation for all SAMS data being saved
Data Archival Manager
SAMSServer
Data Socket
Respond to all commands (except "set" commands) in all modes of operation
SCP Socket
Command Socket
SAMSGui
To SCP
Terminal Server
All APDU traffic between the SAMSServer and the SAMS Electronics
Respond to all commands (except "getttp" – Operate mode) in all modes of operation

Figure 7: SAMSServer application's interfaces and functions

All communication into and out of the SAMSServer is accomplished using TCP/IP socket connections. The SAMSServer manages four sockets which allow access to the SAMS data being collected and provide a communication link between the SAMSServer and other systems comprising the SAMS system. These sockets are the PMC Control Socket, the Command Socket, the Data Socket, and the SCP Socket. The PMC Control Socket allows the SAMSServer to send actuator commands to the PMC during Influence Matrix and Edge Sensor calibrations. This socket is also used during Operate mode to request the current truss temperature from the PMC. The Command Socket is the socket that the SAMSGui uses to control the SAMSServer operation. All SAMS configuration commands are sent to the SAMSServer through this connection. The Data Socket can be used by any SAMS client to acquire the current SAMS data. In particular, the PMC uses this connection to request mirror correction commands during Operate mode. The SCP Socket handles all communication between the SAMS electronics stack and the SAMSServer.

The SAMSServer provides graphical and textual displays that inform the SAMS operator of the health and status of SAMS (Figure 8). The graphical displays provided by the SAMSServer are stripcharts of the global error variance (GEV) metric, the average sensor gap metric, and the HET's mean truss temperature. The GEV is the control system's performance metric computed from the measured shear errors of all active edge sensors and is a useful metric for monitoring how well SAMS is maintaining the mirror figure[4,9]. The average sensor gap measurement is the average gap of all active edge sensors and indicates the magnitude of the HET's radius of curvature change due to temperature. The telescope's mean truss temperature is obtained from the PMC over the PMC Socket and also indicates the magnitude of the HET's radius of curvature change due to temperature. The textual displays provided by the SAMSServer are the Commands Received, Errors Encountered, and Bad Sensors text boxes. These text boxes are all scrollable and each entry is time-tagged. The Commands Received text box displays all commands received over the various socket ports. The Errors Encountered text box conatins all error conditions detected by SAMSServer. The Bad Sensors text box contains all sensors that have been identified as providing erroneous data.



```
2002:04:03:20:27:31 showmode
2002:04:03:20:29:09 setrefpos /space2/
SAMS/REF/sams20020403074315.ref
2002:04:03:20:29:49 setmode Operate
2002:04:03:20:35:03 showmode
2002:04:03:20:35:03 showsegdata 1
2002:04:03:20:35:05 showsegdata 2
2002:04:03:20:35:06 showsegdata 3
2002:04:03:20:35:06 showsegdata 4
```

SAMS Status: Running main application loop

Figure 8 SAMSServer display during Operate mode

During the HET's nighttime operations, Operate mode is the primary mode of operation. An illustration of the data and process flow within Operate mode is shown in Figure 1, and a brief description was given in section 2.1. A detailed discussion of Operate Mode operation and control system flow is given in the references[9]. On a typical night the telescope operator will start with SAMS in Standby mode. The operator stacks the HET mirrors and executes "Set Reference Position" from the Standby mode display. The operator then transitions to Operate mode where the system usually remains until the operator realigns the primary mirror again.

During Operate mode the SAMSServer monitors the health of each sensor. If a sensor is providing erroneous data the software will not compute a set of correction commands. When an error is detected, all correction commands will be set to a value of zero. The sensors that are identified as erroneous will be reported to the operator on the front panel of the SAMSServer. This insures that no correction commands based on erroneous data are sent to the PMC. Bad sensors are also identified and presented to the operator. SAMS will continue to send zero-valued correction commands until the operator takes corrective action by removing the sensor from within Standby mode or fixing whatever problem the sensor is having.

During operation there are several data files managed by the SAMSServer. Each entry into these files is time-tagged. Some files are continuously updated while others are only updated in specific modes. The SAMSServer updates the Log and Error files during all modes of operation. The Log file contains all the information relative to SAMS system startup and records all incoming commands received over the various socket ports. The Error file contains any errors that may have been encountered during SAMS operation. The Log and Error files time tag every communication which they record. Other data files are updated only during Operate mode. These files contain edge sensor shear data, edge sensor gap data, edge sensor temperature data, truss temperature data, tip/tilt/piston correction commands, and global radius of curvature (GRoC) compensation bias data.

CONCLUSION

By utilizing COTS hardware and software products, the SAMS development team was able to successfully develop the SAMS control system software. The use of the LabVIEW development environment was instrumental in the success of the SAMS system development. The benefits of LabVIEW were evident early on during the support of the pre-CDR sub-array tests and continued until final installation and verification. The resources provided by LabVIEW enabled the developers to create a set of applications that provided all the necessary capabilities to support the SAMS development.

REFERENCES

1. Ramsey, L.W., Adams, M.T., Barnes, T.G., Booth, J.A., Cornell, M.E., Fowler, J.R., Gaffney, N., Glaspey, J.W., Good, J., Kelton, P.W., Krabbendam, V.L., Long, L., Ray, F.B., Ricklefs, R.L., Sage,J., Sebring, T.A., Spiesman, W.J., & Steiner, M., 1998, "The early performance and present status of the Hobby-Eberly Telescope," S.P.I.E. Vol. 3352, Advanced Technology Optical/IR Telescope VI, p.34.
2. J.W. Glaspey, M.T. Adams, J.A. Booth, M.E. Cornell, J.R. Fowler, V.I. Krabbendam, L.W. Ramsey, F.B. Ray, R.L. Ricklefs, and W.J. Spiesman, 1998, "Hobby-Eberly Telescope: commissioning experience and observing plans," S.P.I.E. Vol. 3349, Observatory Operations to Optimize Scientific Return, p.50.
3. J. Booth, M. Adams, G. Ames, J. Fowler, E. Montgomery, J. Rakoczy, "Development of the Segment Alignment Maintenance System (SAMS) for the Hobby-Eberly Telescope," *Optical Design, Materials, Fabrication, and Maintenance*, SPIE **4003**, No. 20, Munich, Germany, March 27-31, 2000.
4. J. Rakoczy, D. Hall, R. Howard, J. Weir, E. Montgomery, G. Ames, T. Danielson, P. Zercher, "Demonstration of a Segment Alignment Maintenance System on a seven-segment sub-array of the Hobby-Eberly Telescope," *Adaptive Optics Systems and Technology II*, SPIE **4494**, pp. 69-80, San Diego, California, July 30 – August 1, 2001.
5. D.E. Comer, *Internetworking with TCP/IP Principles, Protocols, and Architectures - Fourth Edition,* p.27, Prentice Hall, Upper Saddle River, New Jersey, 2000.

6. J. Kurose and K. Ross, *Computer Networking A Top-Down Approach Featuring the Internet*, p. 322, Addison Wesley Longman, Inc., Boston, 2001.
7. "Statement of Work: Hobby-Eberly Telescope Segment Alignment Maintenance System," University of Texas at Austin, July 14, 1999, pp. B1-B14.
8. J. Rakoczy, D. Hall, W. Ly, R. Howard, E. Montgomery, "Global radius-of-curvature estimation and control for the Hobby-Eberly Telescope," *Large Ground-Based Telescpes*, SPIE **4837**, No. 79, Waikoloa, Hawaii, August 21-28, 2002.
9. J. Rakoczy, D. Hall, R. Howard, W. Ly, J. Weir, E. Montgomery, M. Adams, J. Booth, J. Fowler, G. Ames, "Primary mirror figure maintenance of the Hobby-Eberly Telescope using the Segment Alignment Maintenance System," *Large Ground-Based Telescpes*, SPIE **4837**, No. 81, Waikoloa, Hawaii, August 21-28, 2002.