

The LBT real-time based control software to mitigate and compensate vibrations.

Borelli J.^a, Trowitzsch J.^a, Brix M.^a, Kürster M.^a, Gässler W.^a, Bertram T.^a, Briegel F.^a

^aMax Planck Institute für Astronomie, Heidelberg, Germany

ABSTRACT

The Large Binocular Telescope (LBT) uses two 8.4 meters active primary mirrors and two adaptive secondary mirrors on the same mounting to take advantage of its interferometric capabilities. Both applications, interferometry and AO, are sensitive to vibrations. Several measurement campaigns have been carried out at the LBT and their results strongly indicate that a vibration monitoring system is required to improve the performance of LINC-NIRVANA, LBTI, and ARGOS, the laser guided ground layer adaptive optic system.

Currently, a control software for mitigation and compensation of the vibrations is being designed. A complex set of algorithms collects real-time vibration data, archiving it for further analysis, and in parallel, generating the tip-tilt and optical path difference (OPD) data for the control loop of the instruments. A real-time data acquisition device equipped with embedded real-time Linux is used in our systems. A set of quick-look tools is currently under development in order to verify if the conditions at the telescope are suitable for interferometric/adaptive observations.

Keywords: Vibration Monitoring System, Large Binocular Telescope, Real-time, Control Systems

1. INTRODUCTION

The analysis of vibration data taken in 2009 at LBT¹ and presented in Fig. 1, shows strong vibrations in the regime from 0 Hz to 5 Hz and some peaks around 8 Hz for all components. The spectral analysis also shows resonances at 9 Hz, 14 Hz, and 25 Hz. The tertiary mirrors present additional resonances between 17 Hz and 19 Hz.

This will clearly impact on the performance of the telescope, particularly on the interferometric capabilities of LBTI and LINC-NIRVANA⁵ and the overall performance of the ARGOS³ system.

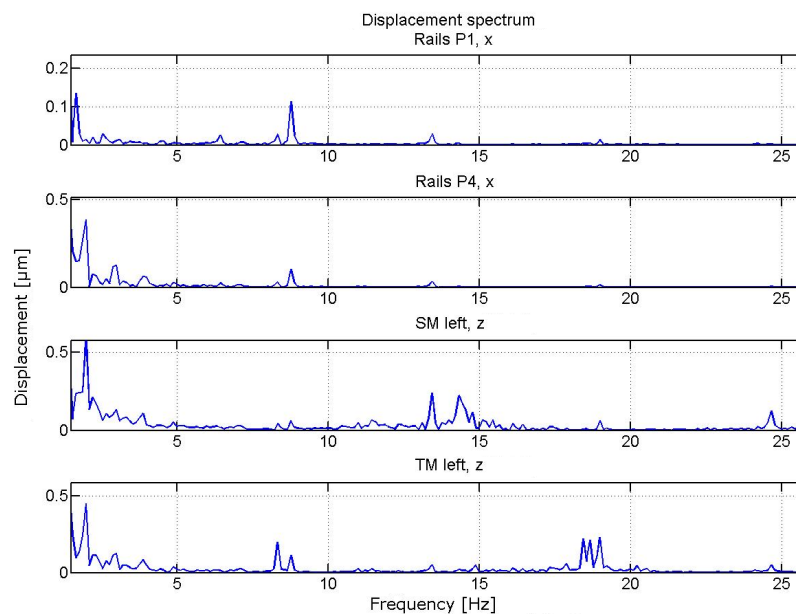


Figure 1. LBT vibration power spectra analysis showing the resonances in the interferometer rails, the secondary mirror(SM) and the tertiary mirror(TM)

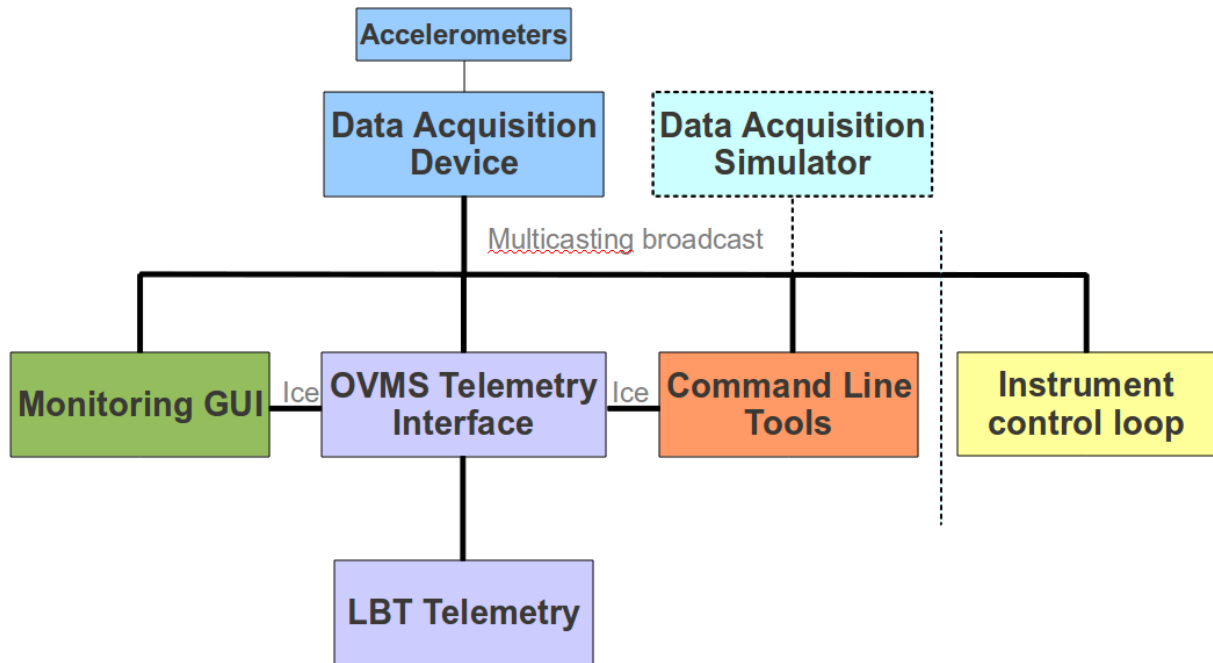


Figure 2. The OVMS architecture: The real-time acquisition service/simulator broadcasts the vibration data and the corresponding time stamp. The monitoring GUI presents this information to the astronomers and operators. The telemetry interface stores the data in the LBT archive for off-line analysis. The command line tools are mainly used for configuration and scripting purposes. The real-time instrument control loop calculates the OPD and compensates the vibrations using a piston mirror.

The LBT OPD and vibration monitoring system (OVMS)⁴ is designed to detect vibrations on the different components of the telescope at a level sufficient to calculate the displacements and variations of its optical path. The distributed system is composed of a real-time acquisition device, 45 accelerometers distributed all over the telescope, and a set of tools to monitor, setup, and control it. Figure 2 shows all these components and the interaction between them.

A real-time service collects vibration data from the accelerometers and broadcasts this information through the network using a multicasting protocol. Different clients join the multicast node in order to get the vibration data and perform their tasks. This mechanism allows to have several clients receiving exactly the same information, reducing the network traffic and any possible timing issue. The foreseen clients for the OVMS are the monitoring GUI, the telemetry interface, and the instrument control loops, all of them under development process at the moment.

The command line tools are mainly used for configuration and scripting purposes. In addition, a data acquisition simulator was designed to detach the software development from the hardware, giving some independence to the developers, especially in the initial phase of the project. It generates and distributes artificial acceleration data to the clients.

2. SYSTEM COMPONENTS

The software development is divided into two main work-packages, the real-time control and acquisition software (RTCS), and the monitoring and storage modules. Both applications run under Linux platforms.

2.1 The real-time acquisition service

This application written in Ansi-C runs embedded in the data acquisition device. The hard real-time operating system of this unit allows to digitize and distribute the accelerometer signals at approximately 5 kHz, using a

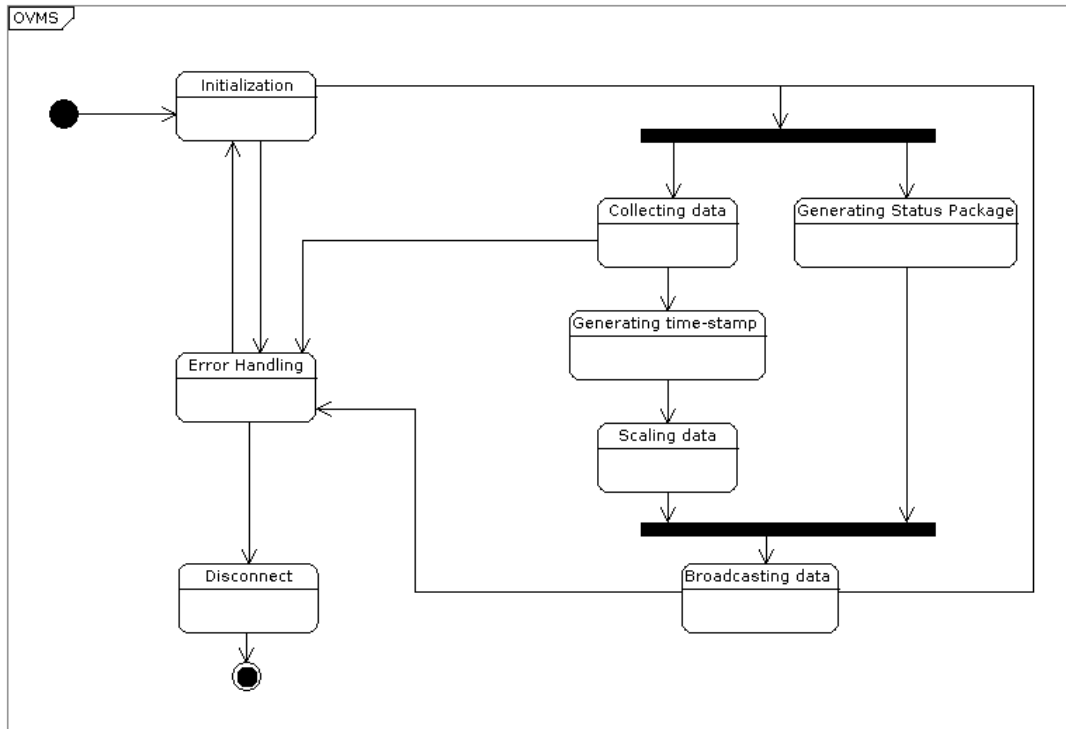


Figure 3. The real-time acquisition service state diagram.

network bandwidth of about 2MB/sec. As shown in Fig. 3, the RTCS also generates the corresponding time stamp and scales the signals before broadcasting them.

Making use of another thread the service periodically generates a status package. The clients will recognize it analyzing the field corresponding to the time stamp. If this value is negative, the received package is either a status or an error package. Table 1 shows the different codes and their meanings.

During initialization the service opens a network socket and sets the specified interface for outbound multicast datagrams.

Table 1. RTC status information.

Code	Description
-1	Status package: Indicates the number of devices, number of channels, and sampling frequency.
-10	Error package: Initialization error.
-20	Error package: Error setting the refresh rate of the devices.
-30	Error package: Error reading or scaling the accelerometer signals.

The overall latency of the system is within 1.5 ms, comprising the time needed to digitize and scale the analog signal, send it to the clients, swap the bits to Little-Endian format, and put it into a buffer.

2.2 The Monitoring Tool

The monitoring GUI is written in C++ using Qt, Qwt, and Ice libraries. Figure 4 shows the preliminary version of the OVMS monitoring GUI and its main components.



Figure 4. Preliminary version of the OVMS monitoring GUI. 1: Telescope component list. 2: OPD status. 3: Telemetry service status. 4: Vibration/PSD plots. 5: Data acquisition device status. 6: Logging window 7: Time range adjustment. 8: Threshold alarm.

Following the sequence described in Figure 5, it connects to the multicast node of the real-time service and provides the operator with a tool for online monitoring of the vibration data. The data plots are constantly updated by a thread reading data from the multicast node. Each plot shows either the raw vibration (acceleration) value data or the corresponding power spectral density (PSD) data.

The operator can select a set of accelerometers (Fig. 4/1) for which the vibration data and/or PSD data should be shown in a plot (Fig. 4/4). The time window used for the plots can be modified and range from 1 minute up to 1 hour (Fig. 4/7) while for each accelerometer the incoming vibration data for the last hour is buffered.

Thresholds are defined for each accelerometer for the vibration data and for the PSD data. These thresholds specify critical data values that should not be reached. In case a threshold is reached the operator is informed immediately about it so that direct troubleshooting can be started (Fig. 4/8).

In the final version additionally the combined effect of all vibration data over all accelerometers in terms of OPD is shown.

Besides the online data monitoring the GUI periodically polls the status of the data acquisition device (Fig. 4/5) and the status of the telemetry interface (Fig. 4/3). The status are presented to the operator. Furthermore the GUI includes a logging system where all relevant information and events are logged so that the operator can investigate what might have caused a current event (Fig. 4/6).

2.3 The Telemetry Interface

The telemetry interface service, written in C++ and using Ice as a middleware, is in charge of collecting the accelerometer measurements from the multicast node and store them into the LBT telemetry subsystem at a maximum rate of 2.4 kHz. This means a data rate of about 1 MB/sec. This stand-alone service is basically a

Table 2. Set of commands of the OVMS Telemetry Interface.

Command	Description
getStatus	Returns the status of the finite state machine and the current sampling frequency.
pause	Pauses the archive of the vibration data into the telemetry subsystem.
resume	Resumes the archive of the vibration data into the telemetry subsystem.
setSampleRateHz	Sets the Sampling rate to archive the vibration data into telemetry.
shutdown	Shutdown the OVMS telemetry interface.

finite state machine that runs in the background and follows the behavior described in Figure 5. It provides a simple set of commands that can be issued either from the OVMS monitoring GUI or using the corresponding command line tool. Table 2 describes this set of commands.

2.4 The Command Line Tools

In order to easily setup and control the system, a set of command line tools have been developed. They can be used either by the telescope operator or grouped in scripts to run automatically every night.

The main purpose of these tools is to control the telemetry interface service and to check the connectivity with the data acquisition system and/or the simulator. The vibration data can be redirected from the standard output to a file, and later on, analyzed with GNUPlot, Matlab, or similar tools.

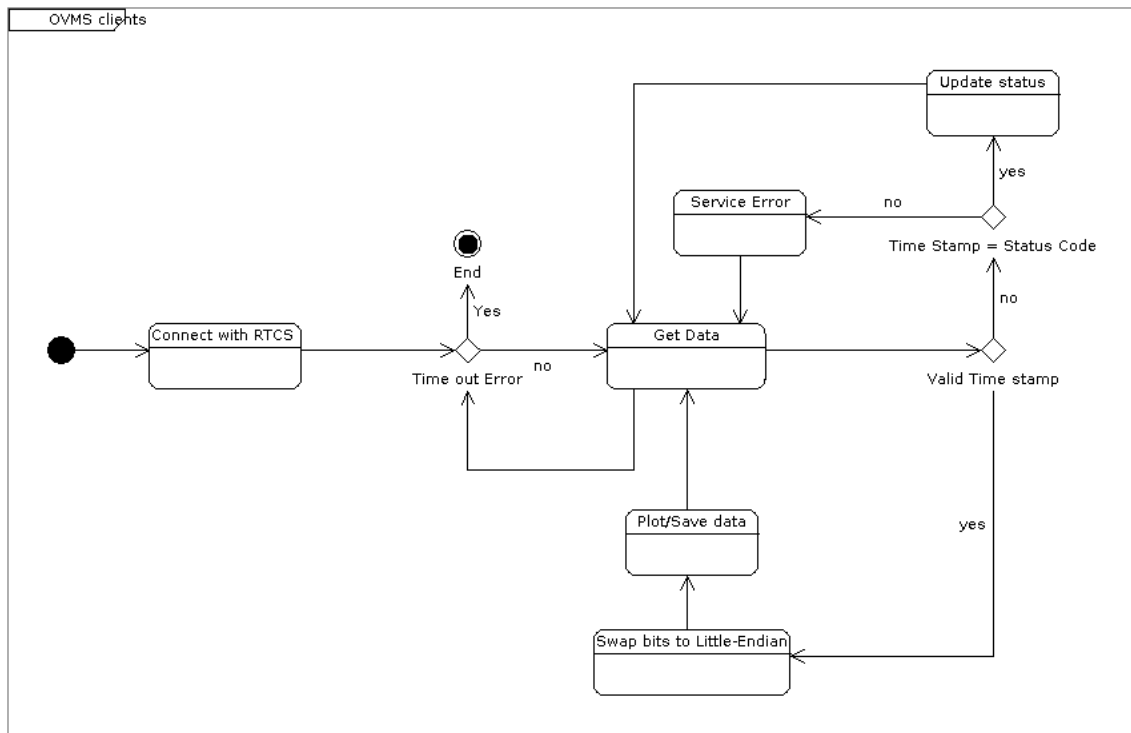


Figure 5. State diagram of the OVMS clients.

3. THE CONTROL LOOP OF THE INSTRUMENTS

For LINC-NIRVANA and LBTI a concept for compensation the OPD resulting from the combined effect of vibrations and atmospheric disturbances is needed to phase the right and left side of the LBT, achieving the maximum resolution of 22.8 m. The optical fringe and flexure tracker is designed and optimized to correct the atmospheric disturbances. The telescope vibrations will be corrected by means of feed forward of accelerometer measurements and OPD compensations.

The OPD value represents at a certain time stamp the instantaneous (path) difference related to the two telescopes. The calculation of the time discrete representation of the OPD has to be fast enough and transferred precisely. It will be used to drive a proper compensator element. The requirements to the dynamic behavior of this actuator are partly driven by the frequency spectrum of the vibration introducing disturbances. Therefore, a concept to control the Piston Mirror as the OPD compensating element is considered for LINC-NIRVANA.²

Unlike the interferometers which compensate for OPD, the ARGOS control loop is designed to mitigate variations in tip and tilt on the laser launch telescope. The launch folding mirrors are mounted on top of the secondary mirrors and the wind-braces. Here the vibrations of the telescope and the wind load have a big impact, generating tip-tilt movements on the mirrors which result in displacement of the laser spots on the sky.

As shown in Fig. 6, eight additional accelerometers mounted on the back plain of the folding mirrors will provide vibration information of the laser launch system in real-time. A data acquisition device will immediately process it to calculate the projection on the pupil mirror, and using an analog output voltage card, the corresponding tip/tilt correction will be applied to the piezo controller.

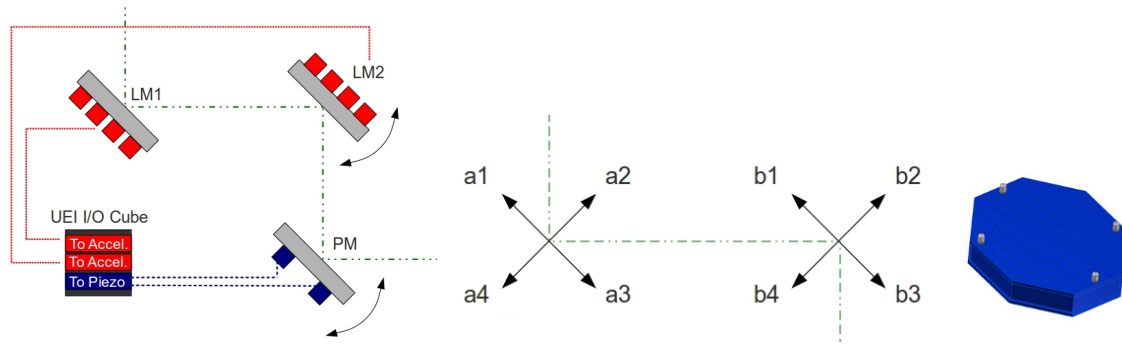


Figure 6. Left: schematic view of the ARGOS vibration control system. Right: position of the accelerometers over the folding mirrors

Knowing the distance from the accelerometers to the center of the mirrors (D_a and D_b), the initial angle of both mirrors (γ_a and γ_b), and the acceleration in μg of *accelerometer_i* we propose the following tip-tilt pseudo-algorithm:

- i. Convert the accelerations by integrating two times from μg to μm (a_i and b_i)
- ii. Calculate the change in tip and tilt angle $\Delta \Theta_x$ and $\Delta \Theta_y$ by doing:

$$\Delta \Theta_x = ((a_1 - a_3)/D_a) * \arctan(\cos \alpha_a - \gamma_a) + ((b_1 - b_3)/D_b) * \arctan(\cos \alpha_b - \gamma_b)$$

$$\Delta \Theta_y = ((a_2 - a_4)/D_a) * \arctan(\cos \alpha_a + 90 - \gamma_a) + ((b_2 - b_4)/D_b) * \arctan(\cos \alpha_b + 90 - \gamma_b)$$

where

$$\alpha_a = \arctan(-\tan(a_1 - a_3)/\tan(a_2 - a_4))$$

and

$$\alpha_b = \arctan(-\tan(b_1 - b_3)/\tan(b_2 - b_4))$$

- iii. Apply to the piezo the $\Delta \Theta_x$ and $\Delta \Theta_y$ corrections, multiplied by the respective gain.

4. NETWORK

To maximize the use of the network bandwidth and to minimize the overhead, the system broadcasts the data using a multicasting connectionless protocol at a rate of 5 kHz. Every slice of data sent includes the time-stamp, plus the scaled signal of the 45 accelerometers.

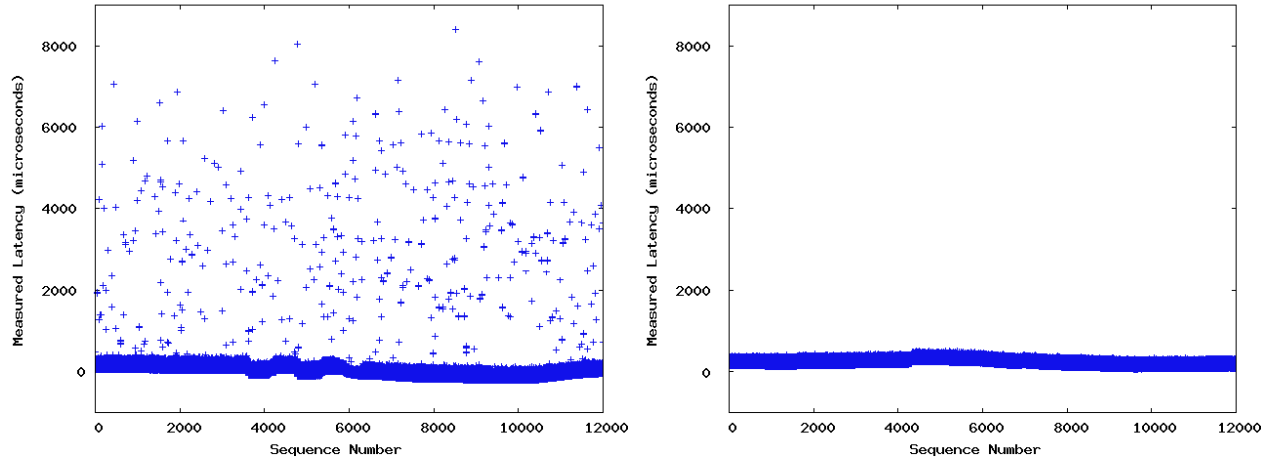


Figure 7. Process execution time to send and receive a datagram package. Left: Latency with maximum CPU priority. Right: Latency with a real-time kernel preemption patch.

Although the monitoring GUI and the telemetry interface have no constraints in terms of network latency, the interferometers need the data in real-time to calculate and compensate the corresponding OPDs.² For such reason, optimization of the network and kernel parameters are needed in the instrument control systems to achieve the minimum performance requirements. Fig. 7 shows the execution time of a normal Linux process with high CPU priority (left plot) that sends and receives a datagram (UDP) package from the data acquisition device versus the same process running with a real time kernel scheduler (right plot). The average for the first one is about 250 μ s, however some packages arrived after up to 9 ms. In the second case, the latency remains stable, about 150 μ s.

5. FAST CONVERSION FROM BIG-ENDIAN TO LITTLE-ENDIAN AND VICE VERSA

Intel defines Endianness⁶ as following: “*Endianness is the format to how multi-byte data is stored in computer memory. It describes the location of the most significant byte (MSB) and least significant byte (LSB) of an address in memory. Endianness is dictated by the CPU architecture implementation of the system rather than the operating system. Actually, the endian model of the CPU architecture dictates how the operating system is implemented.*”

Representing these two storage formats are two types of Endianness-architecture, Big-Endian and Little-Endian. There are benefits to both of these endian architectures. Big-Endian stores the MSB at the lowest memory address. Little-Endian stores the LSB at the lowest memory address. The lowest memory address of multi-byte data is considered the starting address of the data.

The chosen data acquisition device, the PowerDNR RACKtangle from United Electronic Industries, uses a PowerPC CPU to perform the analog-to-digital conversion. This computer architecture uses Big-Endian format to store the data in memory, and therefore, a conversion to Little-Endian on the client side is needed.

The most typical cases are the ordering of bytes within a 16-, 32-, or 64-bit word. There are many ways to convert from Big-Endian to Little-Endian. The functions deployed for the OVMS use only bit shifts and logic operations in order to minimize the time of CPU.

5.1 Swapping 16 bits numbers

Swap (x16) = (x16 >> 8) OR (x16 << 8)

5.2 Swapping 32 bits numbers

Swap (x32) = ((x32 << 24) AND 0xff000000) OR ((x32 << 8) AND 0x00ff0000)
OR ((x32 >> 8) AND 0x0000ff00) OR ((x32 >> 24) AND 0x000000ff)

5.3 Swapping 64 bits numbers

Swap (x64) = (bswap32((unsigned int)(x64 AND 0x00000000ffffffffULL)) << 32)
OR bswap32((unsigned int)((x64 >> 32) AND 0x00000000ffffffffULL))

ACKNOWLEDGMENTS

The authors would like to thank all their colleagues at the MPIA and the University of Arizona for their important input during the entire development process.

REFERENCES

- [1] Brix, M. et al., "Vibration measurements at the Large Binocular Telescope (LBT)," *Ground-based and Airborne Telescopes II*, 7012, 70122J-70122J-10 (2008).
- [2] Brix, M. et al., "Linc-Nirvana piston control elements," Proc. SPIE 7734, 65 (2010).
- [3] Rabien, S. et al., "ARGOS the laser guide star system for LBT," Proc. SPIE 7736, 13 (2010).
- [4] Kürster, M. et al., "OVMS: the optical path difference and vibration monitoring system for the LBT and its interferometers," Proc. SPIE 7734, 107 (2010).
- [5] Herbst, T. et al., "LINC-NIRVANA: the Fizeau interferometer for the Large Binocular Telescope," Proc. SPIE 7013, pp. 701326-701326-7 (2008).
- [6] Intel White Paper, "Endianness White Paper," (2004).