

LBT-AdOpt control software

Luca Fini, Alfio Puglisi and Armando Riccardi

INAF - Osservatorio Astrofisico di Arcetri, L.go E.Fermi, 5. 50125 Firenze (Italy)

ABSTRACT

The LBT-AdOpt subsystem is a complex machine which includes several software controlled parts. It is essentially divided into two parts: a real-time loop which implements the actual adaptive optics control loop, from the wavefront sensor to the deformable secondary mirror, and a supervisor which performs a number of coordination and diagnostics tasks. The coordination and diagnostics task are essential for the proper operation of the system both as an aid for the preparation of observations and because only a continuous monitoring of dynamic system parameters can guarantee optimal performances and system safety during the operation. In the paper we describe the overall software architecture of the LBT-AdOpt supervisor and we discuss the functionalities required for a proper operation.

Keywords: Adaptive optics, Control software, LBT

1. INTRODUCTION

1.1. The LBT adaptive subsystem in brief

As soon as the Large Binocular Telescope will be provided with a secondary mirror (for first light a prime focus camera will be used^{8,9}), it will also have adaptive optics. The LBT optical design is in fact based on a unique* adaptive secondary mirror.

Many references describing the design of the LBT Adaptive Optics subsystem (LBT-AdOpt in the following) and its development can be found in this conference^{3,5-7} and elsewhere^{1,2,4}; for the purpose of this discussion we will only briefly cover the overall structure of the system.

The LBT-AdOpt consists essentially of two parts (see Figure 1):

- The **wavefront sensor** (WFS), which includes the optics, the sensing CCD with its control electronics, the slope calculator and many auxiliary devices.
- The **adaptive secondary** (AS), which includes the deformable mirror with the related control electronics and the wavefront reconstructor, which is actually implemented by using the spare computing power of the mirror control electronics.

Most of the electronics of both parts is based on a custom developed board named BCU (Basic Computational Unit) which is characterized by a very flexible design so that a BCU may be used either for CCD data acquisition or as a controller for many DSP boards controlling the mirror actuators or as a generic fast data communication switch.^{4,5} The real-time adaptive loop is completely implemented in the control electronics of the two parts, which communicate through a dedicated fiber-optics channel by means of a custom protocol. The related software was specifically developed for this application and is described elsewhere.⁴

The BCU's are also equipped with Gigabit Ethernet connections which are used for all the functions not directly related with the real-time control loop: mainly all the supervision and diagnostics operations.

The LBT-AdOpt system is managed by a supervisor software system (SSS) which runs on a workstation (the Supervisor Workstation) located in the control room.

An overall description of the SSS is the main topic of this paper.

Send correspondence to Luca Fini: E-mail: lfini@arcetri.astro.it, Tel: +39 055 2752 307

*It is not actually unique because the first adaptive secondary is currently operating on the revamped MMT¹⁰⁻¹²

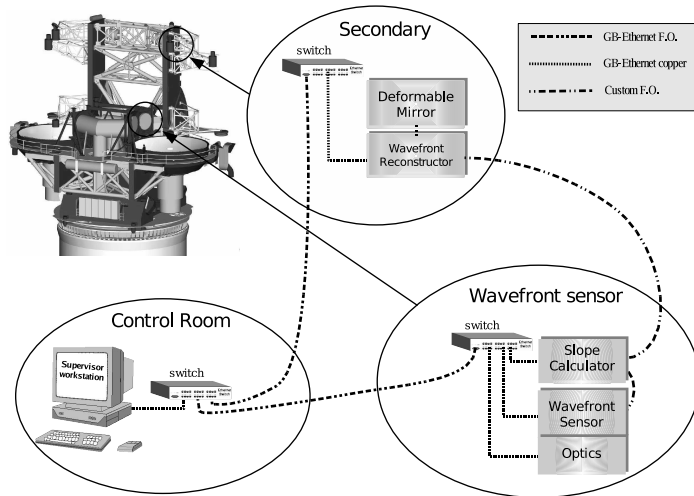


Figure 1. LBT-AdOpt system layout

1.2. Software design drivers

From the architectural point of view the LBT adaptive optics system has some aspects which make it peculiar with respect to traditional adaptive optics systems and thus affect its design and the design of the supervisor software:

- The secondary mirror cannot be removed from the optical path[†]. So the mirror control must be operational even in non adaptive observations.
- Safety issues are particularly important for the adaptive secondary: its thin shell can be easily damaged if not properly controlled.
- Optics and control electronics are located in two far apart places: the secondary on top of the telescope and the focal plane (see Figure 1). The supervisor workstation is located downstairs in the control room (but this is quite usual on many telescopes).
- Although first light configuration uses a single mirror, the LBT is a double mirror telescope, so in the end the SSS must also coordinate two adaptive systems.

In order to better understand some of the design choices made, we must also stress a few more points.

- The LBT-AdOpt system is rather a complex machine: it has many movable mechanical components, it integrates some commercial instrumentation (such as an interferometer) which must be remotely controlled. The functioning of the system is affected by many parameters, including local environment.
- It is also intrinsically an evolutionary system: some aspects concerning the deformable mirror will be only discovered during lab tests; some of the discoveries might lead to modifications in the design. The evolution of the system will also continue after commissioning, e.g.: to optimize its performances.
- The same SSS is also used during lab tests, when additional capabilities and functions not required for the final version at the telescope may be needed.

[†]The adaptive secondary is the only secondary available on LBT to feed the four+four Gregorian focal stations.

- Portions of the code, namely most of the calibration procedures, are derived from existing procedures developed for the MMT adaptive secondary and they must be merged into the SSS.

As a means to cope with all these conflicting requirements the software design has been strongly oriented towards modularity: the whole SSS is essentially based on many independent processes which cooperate by exchanging messages either internally to the same processor or through the network.

The overall idea behind the design is to be able to assemble very easily a number of loosely coupled building blocks (referred to as *components* in the following) so that when a block must be modified or some new block is needed this can be done with the lowest possible impact on the whole software system. This also allows the easy integration of modules developed with different language, which is one of the requirements of the project.

2. SUPERVISOR SOFTWARE FUNCTIONS

The SSS has the purpose to manage and coordinate the functioning of the entire LBT-AdOpt both during development and lab testing and at the telescope. Its main functions include:

- **Housekeeping:** which includes orderly startup of hardware devices, uploading programs and configuration data in the various BCU's, checking proper functioning of devices, performing safe shutdown of devices.
- **Setting seeing limited observation mode:** the deformable mirror must be maintained optically flat by applying proper command patterns to the actuators. During observation this may require periodic updates of mirror shape to compensate for gravity variations, taking input from the focal station in use.
- **Activating chopping mode for IR observation:** chopping is controlled by the SSS by applying proper commands to the deformable mirror.
- **Performing calibration:** calibration procedures consist essentially in applying proper sequences of command patterns to the deformable mirror and acquiring the CCD image. The resulting data are used to evaluate various sets of coefficients to be uploaded to the real-time loop subsystem. They are used for the evaluation of many sets of parameters: the feed-forward matrix; the modal/single actuator step responses and the transfer functions of the internal loop control (used for internal filter optimization); the commands required to "flatten" the shell (the calibration script during this operation uses measurements obtained from the on-board interferometer); the modal/zonal interaction matrix between (for this operation frames from the WFS CCD must be taken).
- **Activating adaptive optics mode:** while doing adaptive observations the actual mirror shape is controlled by the real-time loop electronics. In the meanwhile the SSS will continuously gather diagnostic data, including: capacitive sensor positions pattern, to detect quasi-static deformations to be off-loaded; actuators force pattern; position RMS in a short temporal window to detect shell self-oscillations; position command pattern; various alarm flags fired by dangerous conditions detected by the real-time loop software.
- **System health checks:** in addition to diagnostic checks directly related with the adaptive operation, other data must be acquired in order to assess the system good shape; e.g.: temperatures of DSP's and current drivers in the secondary mirror, inlet and outlet temperatures of the cooling fluid, temperatures of the various heat sinks, humidity, currents generated by current drivers, power fault flags, etc.
- **Communicate with the Telescope Control System:** during operation the SSS will need both to receive high-level commands from the TCS and to send back requests to be executed by some TCS subsystem (e.g.: adjust pointing).

While performing the above functions, the SSS must cope with a very heterogeneous set of devices and subsystems:

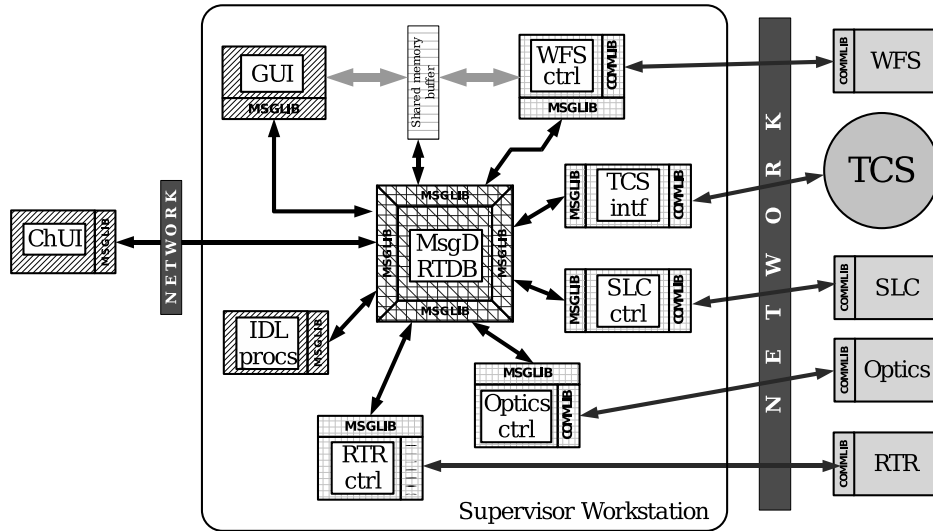


Figure 2. Supervisor Software System architecture

- **In the WFS:** from 8 to 11 moving optical parts; two CCD cameras: the wavefront sensor CCD and the technical viewer CCD; one commercial interferometer; one tip-tilt mirror; and, for laboratory purposes only, two commercial deformable mirrors.
- **On the adaptive secondary:** the BCU's of the mirror controller and the wavefront reconstructor; and the many optical bench devices required during lab tests.
- **The Telescope Control System.**

3. SUPERVISOR SOFTWARE ARCHITECTURE

3.1. Generalities

The main key point in the SSS design is an extreme level of modularity. The system is made up by assembling small components (processes) each dedicated to a single and relatively simple task. The components interact with each other by means of messages implementing a very simple protocol. Messages can be exchanged between components running on the same CPU or through the network and this is completely transparent: the actual location of a component may be set at run-time. Should the need arise, the supporting network architecture can be modified in any way; e.g.: we might decide for testing purposes to run any specific component in Italy while the rest of SSS is running on the supervisor workstation located on Mt. Graham (provided that network speed is enough and network security policies would not prevent this!).

Figure 2 shows the overall architecture of the SSS. All the managed equipment is shown to the right of the figure and is accessed through the network. The SSS components are independent processes running on the supervisor workstation in the control room. In order to stress the modularity of the system one of the software components (the character oriented user interface) is shown as running on a different CPU and interacting with the rest of the components through the network, although this might not reflect the actual running configuration[‡].

The figure also shows a special component (the **MsgD-RTDB**) and two different classes of standard components: User Interfaces and Controllers which we will describe in some deeper detail.

[‡]It must also be noticed that the figure doesn't show the actual network layout: at the telescope many independent subnet will be used in order to balance the network traffic, but this is completely transparent from the point of view of the architecture of the SSS.

3.2. The Message Dispatcher and Real-Time Database

The central component of the SSS architecture is the Message Dispatcher and Real-Time Database (**MsgD-RTDB**). This is the main process of the system and provides the glue which keeps all other components together. It has three main functions:

- **Message Dispatcher - MsgD.** All communications among components are routed through the **MsgD**. This solution has the advantage of simplifying very much the architecture of single components because each one has to cope with a single network connection independently on how many other components it must communicate with. With this choice many synchronization problems which are typical of distributed software systems disappear.

SSS components communicate with each other (and with the **MsgD-RTDB** itself) by means of messages sent to the **MsgD-RTDB** which either directly receives and manages the message or resends it to its final destination.

- **Real-Time Database - RTDB.** The **MsgD-RTDB** also operates as a central repository for *variables*, i.e.: data items which can be read and written by components. Variables are essentially arrays of either integer, double precision or character values with an associated name. They also have a last modification time-stamp, an owner and a rough form of access control.
- **Shared Memory Manager.** Although this is not reflected in its name, the **MsgD-RTDB** has also the function to allow creation and management of shared memory buffers to be used by components to exchange bulky data items. Whenever the use of routed messages is not handy (e.g.: for downloading CCD frames or arrays of actuator positions) standard shared memory buffers are used. The complexity of management of such buffers is hidden to component programmers by the management functions provided by **MsgD-RTDB**.

The **MsgD-RTDB** also provides some auxiliary functions which are useful for the coordination of the components, for diagnostic purposes and so on: synchronization among components (a components may wait for the starting of another component, or for the creation of a variable), notification of variable change (a component may register itself to get notification of modification of selected variables), selective logging (the **MsgD-RTDB** may log messages received or routed according to various selection criteria).

3.3. Interface components

The current SSS design includes three different User Interfaces[§]: a Graphical UI, a command based UI which also provides scripting capabilities and a special user interface from an IDL[¶] environment, needed to allow the reuse of legacy software.

The Graphical UI is the usual way to interact with the system. It is made up of a number of screens that allow an operator to monitor and change all the operating parameters of each subsystem. GUI's communicate with the rest of the system by **MsgD-RTDB** variables, showing their status as graphical information and changing them at user request, e.g.: as the result of pressing a button. The only exception to this protocol is the GUI interacting with the BCU controller described below, which uses a different interface. Currently, there are screens for:

- displaying the CCD window
- displaying/changing the CCD exposure parameters
- displaying/changing the position of each movable part
- displaying/changing the position of the laboratory deformable mirrors

[§]The functionality of the various UI's do not totally overlap. E.g.: some functions are only available through the character based user interface.

[¶]IDL is a data visualization and image analysis language by RSI Inc.

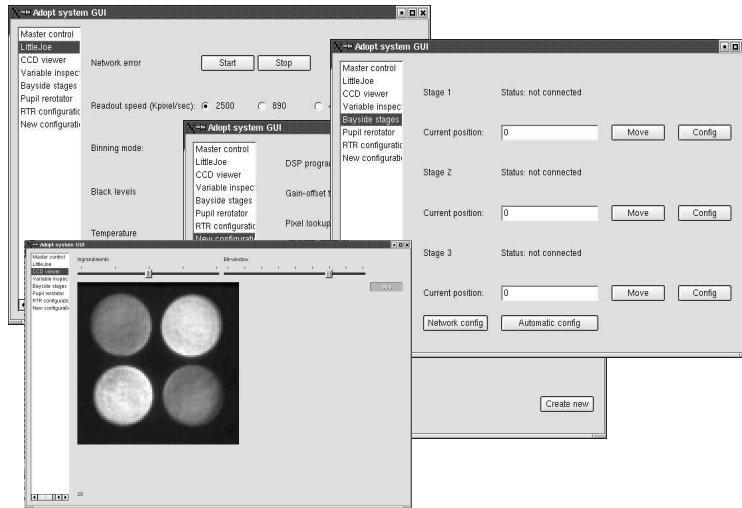


Figure 3. LBT-AdOpt Graphical User Interface

- starting and stopping the overall adaptive optics loop
- selecting the adaptive optics parameter files like dark frames and slope offsets.

All system functions are also accessible with a command based UI, either with dedicated commands or through the general variable interface. The command UI is a full-featured Python interpreter that connects to the `MsgD-RTDB` like any other component. The whole Python language (including modules which are part of the SSS) is available at the command line. The interface with the system is provided as a series of Python functions and classes which exchange messages with the `MsgD` and and and operate on `RTDB` variables. A hierarchical set of Python classes is being written to control most of the hardware from the command line (or as internal commands of the GUI) at various levels of abstraction.

Python can thus be used also as the scripting language to perform complex and repetitive tasks that must coordinate the different parts of the system. For example, changing the CCD binning requires stopping the loop in an ordinate way, changing the binning, preparing the involved hardware devices (mostly BCU's) to the change, and restarting the loop. Such scripts can be executed while the system is running or, as is the case of the calibration script, they can take complete control of the system, including the deformable mirror (a test one in the lab, and the actual adaptive secondary at the telescope). Scripts can use any valid Python instructions and can be built using different Python modules - e.g.: the `numpy` module may be used to perform the mathematical analysis that some commands require.

A special mention must be made to the calibration component. It is based on a set of IDL procedures originally written during the development the MMT adaptive secondary, and reshaped to fit the architecture of LBT adaptive secondary^{||}. Calibration procedures run as independent processes which interact with the `MsgD-RTDB` through a message interface, like any other client. Most of them require direct access to the deformable mirror, and this is done by tunneling requests through the proper BCU controller (see below). Some of the IDL procedures can also be embedded in the scripting interface, but in this case parameter passing is done through FITS files. A tighter interface is planned and will be developed for the final version of the SSS.

^{||}The use of IDL was originally driven by the need of a language that is oriented to heavy processing of data with large data formats and simple use of linear algebra.

3.4. Controller Components

Controller components are processes which interact with the LBT-AdOpt hardware devices (and with the Telescope Control System, which is considered as a very complex device). They have no user interface on their own, but are completely managed via messages issued by some of the UI components. Some of them are actually controlled in a transparent way by simply modifying the value of some variables and using the "variable change notify" function of the `MsgD-RTDB`. Some of them also use shared memory buffers to transfer bulky data blocks.

Most of the controllers have two partially independent functions: one is to listen to requests and commands from the `MsgD-RTDB`, the second is to communicate with the controlled devices. All device control is performed through the network by means of various protocols according to requirements dictated by each specific device. E.g.: the CCD controllers within the WFS and the deformable mirror use a specialized BCU board which includes networking capabilities and supports a dedicated protocol for data exchange.^{4,5} Some devices which only support control through a serial line are transparently managed by using Ethernet/RS-232 converters. Thus the controllers may, if needed, run anywhere on the network and still be able to interface with their hardware.

Just to show the capabilities of the software design here follows the description of some controller components:

- The **BCU controller** is one of the most complex: it must manage the communications with a BCU (Basic Computing Unit), the modular block from which the wavefront sensor and secondary mirror electronics are built. Each BCU has an IP address of its own, and as such is seen as an independent network device. The BCU controller acts as a server with respect to the other processes. A BCU in turn acts as a server serving write and read requests coming from the controller. The BCU controller can serve low-level read/write requests or more high-level requests such as downloading diagnostic data, CCD frames, etc. Data are exchanged by using a custom developed protocol, which is actually the same protocol used for the high-speed communication links of the real-time loop, transported over UDP packets.

Some of the main functions of the BCU controller are:

- Configuring the internal BCU parameters as needed for the observations
- Allow arbitrary read/write into the various BCU memory buffers from any external program
- Serialize multiple incoming requests
- Split requests that are too big to fit into one communication block into a series of requests and reassemble the results
- Provide basic diagnostic about the BCU health state

Operations that require the exchange of large amounts of data are carried out using shared memory buffers, while operations, like simple commands, involving smaller data blocks can use `MsgD` messages. Given the complexity of the operation, no `RTDB` variables are used for the BCU controller.

- The **Stage controller** is used to control the three large Bayside stages that move the entire wavefront sensing unit along the three spatial axes. The main purpose of the controller is to drive the stages at the required position, tune the hardware control loop so that the error is not larger than a fraction of a micron, handle the complex startup phase of the stages, which consists of various steps of brake/unbrake, motor tuning, homing, etc. The complexity of the stage is hidden behind the software interface, which is simply a set of `RTDB` variables (E.g.: target position, required speed, etc.) that can be written by any process. Other non-modifiable `RTDB` variables are set by the Stage controller during starting up to specify the operating range possible for each parameter.

Other very similar controllers are needed for filter wheels and other movements. Their interfaces, whenever possible, are exactly identical to the interface for the stage movements. Thus all the movable parts of the optical board can be managed in a consistent way.

- The **WFS controller** is used to manage CCD cameras. The WFS optical board hosts two CCD cameras, and more may be added in the future. Given how the hardware is implemented, this process controls the

hardware in "open loop", because all the results (CCD frames) are sent to a BCU, which must be ready to handle the data in a correct way. Thus all CCD operations like choosing a binning mode, changing the frame rate and readout speed, should be coordinated by a high-level script which also reconfigures the appropriate BCU. As for the stage controller, the WFS controller has an interface consisting of a set of RTDB variables, which can be changed by other processes.

3.5. Additional functionalities

The LBT-AdOpt software will also include some other programs or procedures which are not directly related with the functioning of the system, but will provide auxiliary functions aimed to increase the reliability of the system. These will include:

- Operating System level scripts to launch and properly shutdown the SSS.
- Hardware checking procedures, to provide early warning related to hardware failures.
- Configuration checking procedure, to periodically check O.S. related parameters (such as temporary disk space availability).

All the checking procedures will be integrated in the global Telescope Control System so that proper warning and alarms may be directed to telescope operators.

4. IMPLEMENTATION AND SOFTWARE ENGINEERING DETAILS

4.1. Development and target environment

The Operating System of choice both for software development and as target environment is Linux. We are currently using kernel 2.4 which is a must because we actually use some of the features introduced with this particular release. Given the high number of processes and threads in the SSS, we are in the process of switching to kernel 2.6 because of its better scheduling policies.

SSS components extensively use advanced O.S. features such as threads, semaphores, shared memory. In the development process maximum effort has been devoted to carefully select O.S. features which may be considered stable and well established, giving preference to POSIX compliance wherever possible.

4.2. Programming languages

The choice of programming languages has been affected by conflicting needs: from one side we would obviously like to use as few languages as possible for coding, but we must also consider that using a language more suitable for the given task may considerably speed up the development. Moreover there is a wealth of legacy software to be reused.

As a result the SSS is currently using three programming languages:

- C/C++, for the time critical components (actually all the control components and the `MsgD-RTDB`).
- Python, for the User interfaces, both character based and graphical. The GUI is based on the PyQt widget library.
- IDL, for the legacy calibration procedures.

4.3. Libraries

All the functions common to various SSS components are gathered in libraries with well defined API's which are consistently used in the implementation:

- **msglib**. Used to assemble and disassemble `MsgD-RTDB` messages and to interface with O.S. networking functions.
- **rtplib**. Although operations on the variable repository are actually implemented via `MsgD` messages, this library defines some higher level function to create, read and write variables.
- **commlib**. Manages the communication with the BCU-based devices.
- **buflib**. Manages the shared memory operations.
- **configlib**. Interfaces with the system configuration files.

4.4. Documentation

Like most instrument related software, the LBT-AdOpt SSS will be presumably around for a long time and it will need to be upgraded and maintained by many different peoples, for different purposes. For this reason we believe that maximum effort must be dedicated to source code documentation. Other kind of documents (such as API definitions, user manuals, and the like) will be obviously produced, but have lower priority.

With respect to code documentation we have adopted the principles of *literate programming* as originally developed by D. Knuth¹³ and, in order to support multiple languages, we developed a small specific WEB-like application,¹⁴ written in Python, which processes source code files and generates source code documentation in various formats (HTML, PDF).¹⁵

5. CONCLUSIONS

The LBT-AdOpt Supervisor Software is a complex system with some peculiar characteristics. Its architecture is the result of two conflicting requirements: the need of a sound software engineering approach to guarantee the reliability and maintainability levels which are a must in this kind of projects, and the very experimental nature of the LBT-AdOpt which requires that functionalities are modified or added just-in-time, in the lab, as soon as a better understanding of the system is reached.

The key design choice has been an higher level of modularity: the SSS consists of a collection of loosely coupled components interacting through messages so that modifications of functionalities or the addition of new and unforeseen capabilities have minimal impact on the whole system.

REFERENCES

1. S. Esposito, A. Riccardi, J. Storm, M. Accardo, C. Baffa, R. Biasi, P. Biliotti, G. Brusa, M. Carbillet, D. Ferruzzi, L. Fini, I. Foppiani, D. Gallieni, A. Puglisi, R. Ragazzoni, P. Ranfagni, P. Salinari, W. Seifert, P. Stefanini, A. Tozzi, C. Verinaud, *First Light Adaptive Optics System for Large Binocular Telescope*, Proc. SPIE 4839, pp. 164-173, Feb. 2003.
2. S. Esposito, A. Tozzi, A. Puglisi, L. Fini, P. Stefanini, P. Salinari, D. Gallieni, and J. Storm, *Development of the first-light AO system for the large binocular telescope*, in "Astronomical Adaptive Optics Systems and Applications". R. K. Tyson, M. Lloyd-Hart, Eds., vol. 5169 of Proc. SPIE, pp. 149-158, Dec. 2003.
3. S. Esposito, A. Tozzi, A. Puglisi, P. Stefanini, E. Pinna, L. Fini, P. Salinari, J. Storm, *Integration and test of the first light AO system for LBT*. This conference: 5490-16.
4. R. Biasi, M. Andrighttoni, D. Veronese, V. Biliotti, L. Fini, A. Riccardi, P. Mantegazza, D. Gallieni, *LBT adaptive secondary electronics*, Proc. SPIE 4839, pp. 772-782, Feb. 2003.
5. R. Biasi, M. Andrighttoni, A. Riccardi, V. Biliotti, L. Fini, D. Gallieni, P. Mantegazza, *Dedicated flexible electronics for adaptive secondary control*. This conference: 5490-84.

6. A. Riccardi, G. Brusa, M. Xompero, P. Salinari, R. Biasi, M. Andrighettoni, D. Gallieni, D. L. Miller, P. Mantegazza, *The adaptive secondary mirrors for the Large Binocular Telescope: a progress report*. This conference: 5490-187.
7. D. Gallieni, V. Anaclerio, A. Ripamonti, R. Biasi, M. Andrighettoni, D. Veronese, W. Ponzo, *LBT adaptive secondary units construction: a progress report*. This conference: 5490-192.
8. E. Diolaiti, R. Ragazzoni, F. Pedichini, R. Speziali, J. Farinato, D. Gallieni, E. Anaclerio, P. G. Lazzarini, R. Tomelleri, P. Rossettini, E. Giallongo, *Blue and red channels of LBC: a status report on the optics and mechanics*, in "Instrument Design and Performance for Optical/Infrared Ground-based Telescopes", M. Iye, A. Moorwood, Eds., vol. 4841 of Proc. SPIE, pp. 552-563, Mar. 2003.
9. A. Di Paola, A. Baruffolo LBT double prime focus camera control software. This conference: 5496-54.
10. G. Brusa, D. L. Fisher, M. Kenworthy, A. Riccardi, *MMT-AO: two years of operation with the first adaptive secondary*. This conference: 5490-3.
11. D. L. Miller, G. Brusa, M. Kenworthy, P. Hinz, D. L. Fisher, *Status of the NGS adaptive optic system at the MMT Telescope*. This conference: 5490-14.
12. M. Kenworthy, D. L. Miller, P. Hinz, G. Brusa, L. Close, D. McCarty, jr., D. L. Fisher, M. Lloyd-Hart, F. Wildi, *Scientific results from the MMT natural guide star adaptive optics system*. This conference: 5490-48.
13. D. E. Knuth, *Literate Programming*, The Computer Journal, 27, 2, May 1984, pp. 97-111.
14. D. E. Knuth, S. Levy, *The CWEB System of Structured Documentation*. Addison-Wesley, 1993.
15. L. Fini *LBT-AdOpt Document Processing Utility - Annotated Source Code*, LBT-ADOPT Technical Report, Sept. 2003.