FLAO System operator manual

1. OS installation and configuration 6
   1.1. Network configuration 6
   1.2. Kernel parameters 6
   1.3. Add non-default repositories 7
   1.4. Add Zeroc-ice repository 7
   1.5. Install additional packages 7
   1.6. Install IDL 8
      1.6.1. IDL related notes 8
   1.7. Add shared libraries to the runtime path 9
   1.8. Support for NFS mounts 9
   1.9. User accounts 9
      1.9.1. flao account configuration 9
      1.9.2. Development and maintenance account configuration 9
   1.10. 4D and IDL 10
2. Software installation 10
   2.1. Bulding a test version of FLAO Supervisor 10
      2.1.1. Checkout FLAO Supervisor source tree and prepare for building 10
      2.1.2. Building and installing the source code 11
      2.1.3. Installing configuration ad calibration data 11
   2.2. Tools for automatic installation, deployment and usage 11
      2.2.1. prepare.py 12
      2.2.2. deploy.py 13
      2.2.3. flao.py 14
      2.2.3.1. Housekeeping commands 15
      2.2.3.2. AO commands 15
   2.3. Implementation Notes 15
      2.3.1. AdOpt servers 15
      2.3.2. Runtime accounts on AdOpt servers 15
      2.3.3. More details about deploy.py 15
      2.3.4. More details about flao.py 16
   2.4. Configuring AO data backup 16
3. Software overview 17
   3.1. Control computers 17
   3.2. Complete start/stop/restart 17
   3.3. Overall software scheme 18
      3.3.1. System processes 18
      3.3.2. AdSec control processes 19
      3.3.2.1. IDL issues 19
      3.3.3. WFS control processes 19
      3.3.4. Arbitrators 19
      3.3.5. AOS 19
   3.4. Engineering interface levels 20
4. AOS GUI 20
   4.1. Starting the GUI 20
   4.2. Status information display 20
      4.2.1. Connection to the AO system 21
      4.2.2. Overall AO system status 22
      4.2.3. Wfs status and commands 23

4.2.4. Adaptive Secondary status 23
4.2.5. Adaptive Secondary on/off/set/rest 24
4.2.5.1. Safety locks 24
4.2.6. Command execution reporting 24
4.3. Command GUI 25
4.3.1. AO commands 26
5. Engineering GUIs 27
5.1. Starting the Engineering GUIs 27
6. Wfs board status GUI 28
6.1. Starting the GUI 28
6.2. GUI description 29
7. Wfs Arbitrator GUI 29
7.1. Starting the GUI 31
7.2. GUI description 31
7.2.1. Status indicators 31
7.2.2. Startup/Shutdown commands 31
7.2.3. AO parameters 32
7.2.4. AO loop open/close/pause 33
7.2.5. Rotator tracking 34
7.2.6. Camera lens tracking 34
7.2.7. ADC tracking 34
7.2.8. Anti drift 35
7.2.9. Dark frame and slopenull acquisition 35
7.2.10. Disturbance 35
7.2.11. Offsets 36
7.2.12. WFS displays 37
7.2.13. CCD display 37
7.2.13.1. Controls 38
8. WFS Hardware GUI 39
8.1. Starting the GUI 39
8.2. GUI description 39
8.2.1. Power controller 40
8.2.2. CCD39 41
8.2.2.1. Status 41
8.2.2.2. Controls 42
8.2.3. CCD47 43
8.2.4. Filter wheel #1 43
8.2.5. Filter wheel #2 44
8.2.6. Status check 44
8.2.7. Temperatures 45
8.2.8. Tip-tilt 46
8.2.9. Tip-tilt low level 48
8.2.10. Pupil rerotator 48
8.2.11. Cube stage 49
8.2.12. Cube rotator 50
8.2.13. Bayside stages 51
8.2.13.1. Displays 52
8.2.13.2. Controls 53
8.2.14. Source lamp 53
8.2.15. Camera lens 53
8.2.16. ADC wheels #1 and #2 54
8.2.17. ADC high-level 55

8.2.18. Board setup 56
8.2.19. System tests 58
8.2.20. Quick selection 58
9. AdSec operation 58
9.1. Safety Remarks 58
9.2. Quick start with AOS GUI (from BP4 built ahead) 59
9.3. Quick start with Engineering GUI 62
9.4. Quick recovery from Failure 64
9.5. Adaptive Secondary startup and shutdown 65
9.5.1. With Engineering GUIs 65
9.6. More on GUIs 66
9.6.1. AdSec Mirror GUI 66
9.6.2. AdSec Housekeeper GUI 67
9.7. Housekeeper configuration files 73
10. Arbitrator GUI ..... **Error! Bookmark not defined.**
10.1. Starting the GUI ..... **Error! Bookmark not defined.**
10.2. GUI description ..... **Error! Bookmark not defined.**
10.2.1. Status indicators 64
10.2.2. Startup/Shutdown command ..... **Error! Bookmark not defined.**
10.2.2.1. Safety locks ..... **Error! Bookmark not defined.**
10.2.3. Shape control ..... **Error! Bookmark not defined.**
10.2.4. Shape offloads ..... **Error! Bookmark not defined.**
10.2.5. Zernike application ..... **Error! Bookmark not defined.**
10.2.6. Reconstructor control and gain ..... **Error! Bookmark not defined.**
10.2.7. Disturbance control ..... **Error! Bookmark not defined.**
10.2.8. Focal station selection ..... **Error! Bookmark not defined.**
11. AdSec Arbitrator engineering displays ..... **Error! Bookmark not defined.**
11.1. AdSec display (AdSec Mir GUI) ..... **Error! Bookmark not defined.**
11.1.1. Starting the GUI ..... **Error! Bookmark not defined.**
11.1.2. GUI description ..... **Error! Bookmark not defined.**
11.2. AdSec housekeeper GUI ..... **Error! Bookmark not defined.**
11.2.1. Starting the GUI ..... **Error! Bookmark not defined.**
11.2.2. GUI description ..... **Error! Bookmark not defined.**
12. Low-level GUIs 73
12.1. System processes GUI 73
12.2. Variable inspector tool 74
12.3. Text-based tools 75
12.3.1. IDL terminal ..... **Error! Bookmark not defined.**
12.3.2. Python shell ..... **Error! Bookmark not defined.**
12.3.3. Thrdtest ..... **Error! Bookmark not defined.**
12.3.4. Consumer 75
12.3.5. BCUread ..... **Error! Bookmark not defined.**
12.3.6. Log files 75
12.3.6.1. Log file archiving 76
12.3.7. Telemetry files 76
12.3.8. Log viewer ..... **Error! Bookmark not defined.**
13. Common tasks 76
13.1. System preparation 76
13.1.1. Using the AOSGUI 76
13.1.2. Using the Arbitrator GUIs 76
13.2. System shutdown after observation 77
13.2.1. Using the AOSGUI 77

13.2.2. Using the Arbitrator GUIs 77
13.3. Seeing limited observation 77
13.4. AO observation sequence 77
13.4.1. PresetAO 78
13.4.1.1. Error conditions and recovery 78
13.4.2. AcquireRefAO 78
13.4.2.1. Error conditions and recovery 79
13.4.3. StartAO 79
13.4.3.1. Error conditions and recovery 80
13.4.4. PauseAO/ResumeAO 80
13.4.4.1. Error conditions and recovery 80
13.4.5. OffsetAO 80
13.4.6. Other failure modes 80
13.4.6.1. AdSec safety fault 80
13.4.6.2. Hardware failure 81
14. Calibration procedures 81
14.1. Pupil calibration ..... **Error! Bookmark not defined.**
14.1.1. Pupil acquisition ..... **Error! Bookmark not defined.**
14.1.2. Pupil optimization ..... **Error! Bookmark not defined.**
14.2. Interaction matrix calibration 81
14.2.1. Preparation 81
14.2.2. Measurement parameters 81
14.2.2.1. Modal basis 81
14.2.2.2. WFS CCD binning 82
14.2.3. Modal history generation 82
14.2.4. Interaction matrix measurement 83
14.2.5. Reconstructor matrix generation 86
14.2.5.1. Iteration 86
15. Saving diagnostic data 87
15.1. Data format description 87
15.2. Optical Loop Diagnostic GUI 89
16. Elaboration library (elab_lib) 90
17. Configuration files 90
17.1. File format 90
17.2. MsgD configuration file 91
17.2.1. Configuring peering 91
17.2.2. LBT setup 91
17.2.3. LTB configuration files 92
17.2.4. How to check if peering works correctly 92
17.3. Common keywords 92
18. Configuration keywords 93
19. Table of wfs, adsec and AO status values and commands accepted 103
19.1. Wfs command table 103
19.2. AdSec Command table 104
19.3. AO Command table 104

Modification Record

Version	Date	Author	Section/Paragraph affected	Reason/Remarks
1.0	19 Nov 2015	A. Puglisi		First release of the document as part of the FLAO document package
1.1	25 Aug 2016	A. Puglisi, F. Rossi	Added sections 1,2 Modified sections 13,15,17	Integrating relevant content scattered in other documents.
1.2	26 Aug 2016	A. Puglisi	Modified sections 14, 16, 17. Fixed layouts in section 2.	Added description of diagnostic data format, description of configuration files and several configuration keywords.
1.3	6 Sep 2016	A. Puglisi, F. Rossi	Added section 1.6, modified section 1.4	Integrated data backup setup instructions, added note about installation of portmap.
1.4	26 Sep 2016	A. Puglisi	Detailed section 16.2	MsgD peering.

1. OS installation and configuration

OS installation and configuration, following the steps described in the following paragraphs, must be done with root privileges.

Install selecting the "software development workstation" setting. This will install most required packages automatically. After the installation from a distribution media, a full update is suggested. The list of RPMs after OS installation + upgrade is in:

- rpmlist-centos6.txt for CentOS 6.x
- rpmlist-centos7.txt for CentOS 7.x

1.1. Network configuration

- Put in /etc/hosts the addresses of the devices as specified in IpNumbers.
- The firewall configuration must allow network packets from the BCU's. The easiest way is to declare the ethernet interface of the BCU subnet as trusted.

1.2. Kernel parameters

In order to full configure the Adaptive Secondary it's needed to expand (if not) the shared memory size provided by the Operating System.
To check if there is enough shared memory you can write (from root):

```
# sysctl -a | grep shm
```

and look for the lines:

```
kernel.shmall = 131072
kernel.shmmax = 536870912
```

If you have values for kernel.shmall and kernel.shmmax have lower values, please change /etc/rc.d/rc.localfile adding the two lines:

```
sysctl -w kernel.shmall=131072
sysctl -w kernel.shmmax=536870912
```

This will set the values at reboot. You may also give the commands at a command propmpt for immediate effect.

1.3. Add non-default repositories

Some packages (mainly related to Qt version 3) are not available in the default CentOS6.x repository, You must add both the epel and the atrpms repositories to the yum list as shown below:
EPEL CentOS 6

```
wget http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -Uvh epel-release-6-8.noarch.rpm
```

ATRPMS You must add a file: /etc/yum.repos.d/atrpms.repo with the following content:

```
[atrpms]
name=Red Hat Enterprise Linux $releasever - $basearch - ATrpms failovermethod=priority
baseurl=http://www.mirrorservice.org/sites/dl.atrpms.net/el$releasever-
$basearch/atrpms/stable
enabled=1
gpgcheck=0
```

EPEL CentOS 7:

```
http://dl.fedoraproject.org/pub/epel/7/e/epel-release-7-5.noarch.rpm
```

1.4. Add Zeroc-ice repository

Note: Ice 3.6 which is the latest available at the moment of this writing seems not to provide RPMs for python support. Thjere is a python package available from PYPI, but only for python 3. We currently choose to develop against Ice 3.5
The easiest way to retrieve the code is to add the proper file to the yum directory as follows:
CentOS 6.x:

```
cd /etc/yum.repos.d
sudo wget https://zeroc.com/download/Ice/3.5/el6/zeroc-ice-el6.repo
```

CentOS 7.x:

```
cd /etc/yum.repos.d
sudo wget https://zeroc.com/download/Ice/3.5/el7/zeroc-ice-el7.repo
```

1.5. Install additional packages

The following are required packages from the CentOS distribution:

```
yum install qt-devel PyQt gmp-devel pyfits qt3-config kdelibs3-devel libXpm-devel cfitsio-devel lrzsz
```

Depending on the initial selection of features when installing CentOS, it might be necessary to install some the following packages:

```
yum install gcc-c++ subversion boost-devel ncurses-devel python-devel readline-devel armadillo-devel mysql++-devel openmotif-devel xterm
```

Then we need the ICE packages:

```
yum install ice ice-libs ice-c++-devel ice-python ice-python-devel
```

Note: the FLAO supervisor build procedure requires that a "version independent" link is created as in the following example (Note: make the link to the actually installed Ice version):

```
ln -s /usr/share/Ice-3.4.2 /usr/share/Ice
```

1.6. Install IDL

You must follow directions provided by IDL vendor. Current tested version is IDL 7.1, but other releases may work also.

1.6.1. IDL related notes

Note 1: do not forget to install/configure the license
Note 2: make sure that the “portmap” program is installed (it is not on Centos7 default distribution):

```
sudo yum install portmap
```

Note 3: Under CentOS 6.x the idlrpcserver can only be started with root privileges. As an alternative Exelis suggests to also specify the port as follows:

```
idlrpc -port=0x20001000
```

This seems not to work in all circumstances. We found more stable a different solution:

- Change to root the ownership of the idlrpc executable and declare the same file setuid:

```
chown root <idl_rpc_executable>
chmod +s <idl_rpc_executable>
```

The location of the executable file depends on the IDL release. The FLAO installation procedure has a tool to find where IDL is installed, so you can delay this step after the preparation of the installation environment described below.

1.7. Add shared libraries to the runtime path

Create library path files as follows (Note: the IDL path may vary):

```
echo /usr/local/qwt-5.1.2/lib > /etc/ld.so.conf.d/qwt.conf
echo /usr/local/exelisvis/idl71/bin/bin.linux.x86_64 > /etc/ld.so.conf.d/idl.conf
```

Then refresh the path:

```
/sbin/ldconfig -v
```

1.8. Support for NFS mounts

- The ADSEC server must be set to export via NFS the directory **/local/aomeas** to the WFS server
- The WFS server must NFS mount the directory **/local/aomeas** exported by the ADSEC server.

Note 1: In order to allow proper access to files the **flao** user account and other accounts used for development should have the same UID on the two servers.
Note 2: To allow the expected propagation of UID in NFS mounted filesystems the option **vers=3** must be used in file **/etc/fstab** of the mounting client.

1.9. User accounts

The production FLAO software will run from the account flao. This account must not be used for any software development whatsoever.
Software tests and any other maintenance operation will be performed from any other suitable user accounts.

1.9.1. flao account configuration

The flao account must have read/write access to the FLAO working directories:
/local/aolog
/local/aomeas (Note: this is a real directory on ADSEC server and is NFS mounted on the WFS server.
The two directories must be created in advance and must have proper owner and permissions (suggested owner:group = **flao:flao**, suggested permissions: **drwxrwxr-x**)

1.9.2. Development and maintenance account configuration

Development and maintenance user accounts must have read/write access to the directory /local/aomeas. (Suggestion: add user accounts used for development to the flao group)

1.10. 4D and IDL

To use the 4D PhaseCam 4020 with the IDL wrapper:

- Add the **PYRO_CONFIG_FILE** environment variable (i.e equals to $(ADOPT_ROOT)/conf/left/Pyro_Client.conf)
- Set the variable **PY_VER** in Makefile.gen

2. Software installation

2.1. Bulding a test version of FLAO Supervisor

The FLAO Supervisor can (and must) be built and tested from any convenient account which will not be used for the production installation. The account needs (and must have) only normal user privileges. Here follows the description of main steps.

2.1.1. Checkout FLAO Supervisor source tree and prepare for building

You must checkout from the proper SVN repository to get the FLAO version you want to install into any convenient directory. The following example gets the source tree from the SVN trunk and checks it out onto ./source (You have to specify an authorized username and you'll be prompted for a password):

```
svn checkout "svn+ssh://username@adopt.arcetri.astro.it/aogroup/svn/AOSupervisor/trunk
source"
cd source
```

Then you must set up the environment to allow compilation:

```
python prepare.py
make
source flao_environment.sh
```

The prepare procedure creates some working directories on your HOME checks the availability of IDL and creates the file **flao_environment.sh** with the environment definition required for the compilation of FLAO Supervisor.
You may want to add the source command to your environment setup procedure at login (usually **.bashrc**) to have it executed at every login.
Note: If the prepare.py procedure has executed correctly, you now can find the location of IDL executable files as follows:

```
echo $IDLLIBDIR
```

And you can modify **idlrpc** properties as directed above (see paragraph on IDL)

2.1.2. Building and installing the source code

NOTE: please be sure you have sourced the environment definition procedure **flao_environment.sh** before attempting to build the Supervisor.
The FLAO Supervisor build process is in four steps:

1. Build the contributed software:

```
cd $ADOPT_SOURCE/contrib
make (see note below)
sudo make install
```

2. Compile and install the Supervisor

```
cd $ADOPT_SOURCE
make
make install
```

Note: verify that the install procedure has set proper ownership (root) and permissions (setuid) to the following executables: mirrorctl, masterdiagnostic

2.1.3. Installing configuration ad calibration data

Calibration and configuration data are maintained in a different SVN repository. To install them you must first checkout the latest version onto a suitable directory:

```
svn checkout "svn+ssh://username@adopt.arcetri.astro.it/aogroup/svn/AOSupervisor/confcalib
confcalib
```

Then to install configuration and calibration files:

```
cd .../confcalib
make install_conf
make install_calib
```

2.2. Tools for automatic installation, deployment and usage

In the following page we resume the usage of procedures to install FLAO software, deploy on the AdOpt servers and start stop the software subsystem.
The management of FLAO software is performed by three procedures (which can be found at the root of the source code directory tree as checked out from the SVN repository.

- **prepare.py**. Environment setup for creating an installation to be used for software development and tests. The environment is suitable to run a single subsystem (adsec/ wfs, right / left).

- **deploy.py**. To deploy a runtime build of FLAO software onto the four subsystem servers. The deployed build is named TEST and is suitable for running the AO system at the telescope. The same procedure will be used at the end of tests to release (freeze) the build after testing.

- **flao.py**. Procedure to be used by the TO to manage the FLAO software, i.e.: select an available build, start/stop processes and the like. This procedure is intended to be usable as a standalone tool, i.e.: it does not need any other FLAO software component and can be run on any computer with an SSH access to the FLAO servers.

Here follows a detailed description of the procedures and their use.

2.2.1. prepare.py

The purpose of the procedure is to setup a proper environment for FLAO software development (and/or maintenance) and is usually run once after cyhecking out a fresh version of the FLAO source code. Here follows the help page which comes out with **python prepare.py**.

```
 FLAO Supervisor environment setup procedure.  Version 2.4    L.Fini, April 2015.

Usage:

python prepare.py check                       Check environment
python prepare.py make                        Create environment
python prepare.py set adsec|wfs left|right    Set installation target

This procedure operated on the local environment to be used for build and tests.  The
actual installation of "science ready" FLAO system must be done with:
python deploy.py
```

The procedure provides three subcommands:

- **make**. The make subcommand creates (or checks) the environment for development, more in details:
    - Creates local runtime directories: ~/aoroot, ~/aolog, ~/aomeasures.
    - Creates a private/public key file pair to be used by **ssh** and stores them into **~/.ssh** directory of the current user.
    - Creates an identity code for the specific build.
    - Creates a bash compatible file for setting up environment variables: **flao_environment.sh==**

      The environment definition file must be explicitly executed to take effect. If

desired it can be executed from the bash startup procedure (usually =~/.bashrc=). This is enough to be able to make and install the FLAO code, configuration and calibration files.

Note 1: when installing and running the development build, executable, configuration and runtime files are all stored in users' specific directories.

Note 2: In order to be able to execute the FLAO software, the specific server identity (i.e.: adsec / wfs and left / right) must be established with the **set**subcommand.

- **check**. The check subcommand verifies the environment to check that it is properly set for compilation and installation of the FLAO software.

- **set**. The set subcommand defines the specific identity of the current build so that it can operate as one of the AdOpt servers (i.e.: either as **adsec** or as **wfs** and with the required side (**right** or **left**).

  The effect of the procedure is to create the required symbolik link in the ADOPT_ROOT runtime directory tree and to generate a new version fo the environment setup file flao_environment.sh suitable for running the code with the proper identity.

  Note: the environment setup procedure must be explicitly executed in order to have effect.

2.2.2. deploy.py

The deploy procedure has the purpose to deploy all required runtime files onto the four AdOpt servers. The procedure assumes that the servers have been already configured for the purpose. Each server must have an account flao which will be used for software deployment and to run the FLAO software.
The procedure is intended to be launched from the root of the FLAO software source tree after properly setting the environment as defined in the flao_environment.sh file generated by prepare.py. Moreover the full software generation sequence: make, make install, make install_conf and make_install_calib must have been completed.
Here follows the relevant items from the help page as displayed by **python deploy.py**:

```
 FLAO Supervisor deployement procedure. Version 1.6    L.Fini, May 15, 2015   Usage:
python deploy.py [-v] command [args]

-v: Verbose command mode (for debug)
Commands:
  key:   Send SSH public key to targets (to be done once)
  rel:   Release the TEST installation
  test:  Deploy a test installation (possibly overwriting a previous one)
```

The procedure provides four subcommands:

- key. Transmits the ssh public key specific for FLAO management to the four AdOpt servers. Usually this will require to specify the password defined for the flao user on the four servers. This operation must be done once, before a build is deployed for the first time.

- test. Deploy current build as TEST build. All the required files will be copied onto proper directories on the four targets and required links will be created. The deployment can be repeated, in which case the new build is written over the previous one.

  Note: After the deployment the current setup of server is not modified, i.e.: the previously active build (if any) is still active. The selection of active build can be done with the flao.py procedure.

- rel. Releases the TEST build as an science ready build. The name is generated automatically and is of the form: 2105X (current year plus a single letter).

- lnk. Redoes the links in the specified build (for test purposes)

The deploy.py also provides a copy of commands provided by flao.py.

2.2.3. flao.py

This procedure is intended to be used by Telescope Operators to manage the runtime FLAO software system. For this purpose it can be used standalone on any computer provided with a standard python (2.x) installation and allowed to connect via ssh to the FLAO servers.

Note: because the procedure spawns standard unix commands and requires an X11 server, it is not supported in MS Windows environments (and has never been tested on Mac OS).
Here follows the help page obtained with python flao.py:

```
flao.py      Vers. 1.3          Luca Fini, 25 May, 2015

This file contains standalone functions to manage the FLAO
supervisor procedures.

Usage:

      python flao.py [-v] command [options]
      -v:    Verbose mode (for debug)

Housekeeping commands:

      env node        Show environment at remote node
      list            Show available builds
      set  build      Set active build
```

```
      show            Show active build
      targets         Show the names of AdOpt servers and their roles.



AO commands:

      start adsec|wfs r||  Start the specified subsystem
      stop adsec|wfs r||   Stop the specified subsystem
      check adsec|wfs r||  Check the specified subsystem
      eng adsec|wfs r||    Starts the engineering GUI on the specified subsystem
```

2.2.3.1. Housekeeping commands

To be used for various check of the build installations.

- env. Show FLAO related environment at given server (for debug purposes)
- list. List available builds.
- show. Show currently active build
- set. Set the specified build as active
- targets. List the AdOpt servers and their roles.

2.2.3.2. AO commands

To be used for managing the FLAO Supervisor Software.

- start. Start the specified subsystem (e.g.: start adsec right)
- stop. Stop the specified subsystem.
- check. Check the specified subsystem
- eng. Start the engineering GUI of specified subsystem.

2.3. Implementation Notes

2.3.1. AdOpt servers

The procedures described above have the specification of AdOpt servers and their roles built in. If server names will change in the future, the related tables into the file flao.py must be changed accordingly. The relevant table name is: MTGRAHAM_HOSTS.

2.3.2. Runtime accounts on AdOpt servers

The procedures rely on proper configuration of a user account on the four Adopt servers. The userid must be: flao, with any suitable password. A specific public key will be stored in the .ssh directory of the account when using the key subcommand of deploy.py procedure. After that all remote operations will be done using the key, i.e. without the need to specify a password.

2.3.3. More details about deploy.py

The deploy.py procedure has some more options useful for developers and debuggers. Here follows the additional help output which is shown with: python deploy.py -h

```
ADDITIONAL INFO FOR DEVELOPERS

The procedure accepts an additional option:

   -a:   select a list of target servers available at Arcetri to be used for tests instead
of the servers used for science ready operations.

The procedure also accept additional commands:

   fake:  Make a FAKE build (for debugging)
   lnk:   Redo build links
```

Note: The FAKE build just contains a fake implementation of the FLAO process launching python scripts.

2.3.4. More details about flao.py

The flao.py procedure has some more options useful for developers and debuggers. Here follows the additional help output which is shown with: python flao.py -h

```
   ADDITIONAL INFO FOR DEVELOPERS

The procedure accepts an additional option:

 -a:   select a list of target servers available at Arcetri to be used for tests instead of
the servers used for science ready operations.
```

2.4. Configuring AO data backup

There are two levels of data mirroring active at LBTO:

- Local data mirroring: from server internal disks to local shared SAN disks.
- Remote data mirroring: from SAN disks to NAS in Arcetri (not yet finished)

Each level provides for mirroring log files and AO data

Log files:

adsecdx	/local/aolog	➡	/ao-data/adsecdx/aolog
wfsdx	/local/aolog	➡	/ao-data/wfsdx/aolog
adsecsx	/local/aolog	➡	/ao-data/adsecsx/aolog
wfssx	/local/aolog	➡	/ao-data/wfssx/aolog

Calibration data:

adsecdx	/local/towerdata/adsec_data	→	/ao-data/adsecdx/adsec_data
adsecsx	/local/aomeas/adsec_calib	→	/ao-data/adsecdx/adsec_data

Data mirroring is performed by the **archiver.py** procedure. This procedure is designed to be regularly run (for example once a day) from a crontab file. Each of the four AO computers should have this line in their crontab file:

MM HH * * * <YOUR_PATH_TO_ADOPT_ROOT>/py/archiver.py –r

Where MM and HH is a suitable time for data backup (usually, in the middle of the day, in order to avoid network traffic during the night), and the path should be set in order to point to the current AO software installation. The "-r" option tells the procedure to execute all the configured backup jobs.

Further information on the archiver.py procedure, how to configure backup jobs, etc. is available in the archiver.py file itself as a Python docstring.

3. Software overview

3.1. Control computers

The FLAO software runs on two workstations, called **wfsdx** and **adsecdx**, dedicated respectively to control the WFS and the Adaptive Secondary. These workstations are AO-specific and are not part of the TCS server farm, but are accessible via ssh from any TCS machine or operator/observer workstation. The software runs on these workstations as user **AOeng**.

3.2. Complete start/stop/restart

The FLAO software is not a single program, but a collection of processes dedicated to hardware control, plus several more processes which coordinate their actions to perform most AO operations. All processes are normally always running, but it may happen that the software must be shutdown and/or restarted (for example, in case of computer power failures).
A few commands have been implemented on the two workstations to start and stop the complete list of processes. These are:

- **w_start** (on wfsdx) to start all the wfs-related processes
- **w_stop** (on wfsdx) to stop all the wfs-related processes
- **w_restart** (on wfsdx) to execute a stop followed by a restart
- **w_check** (on wfsdx) to check whether the wfs software is running

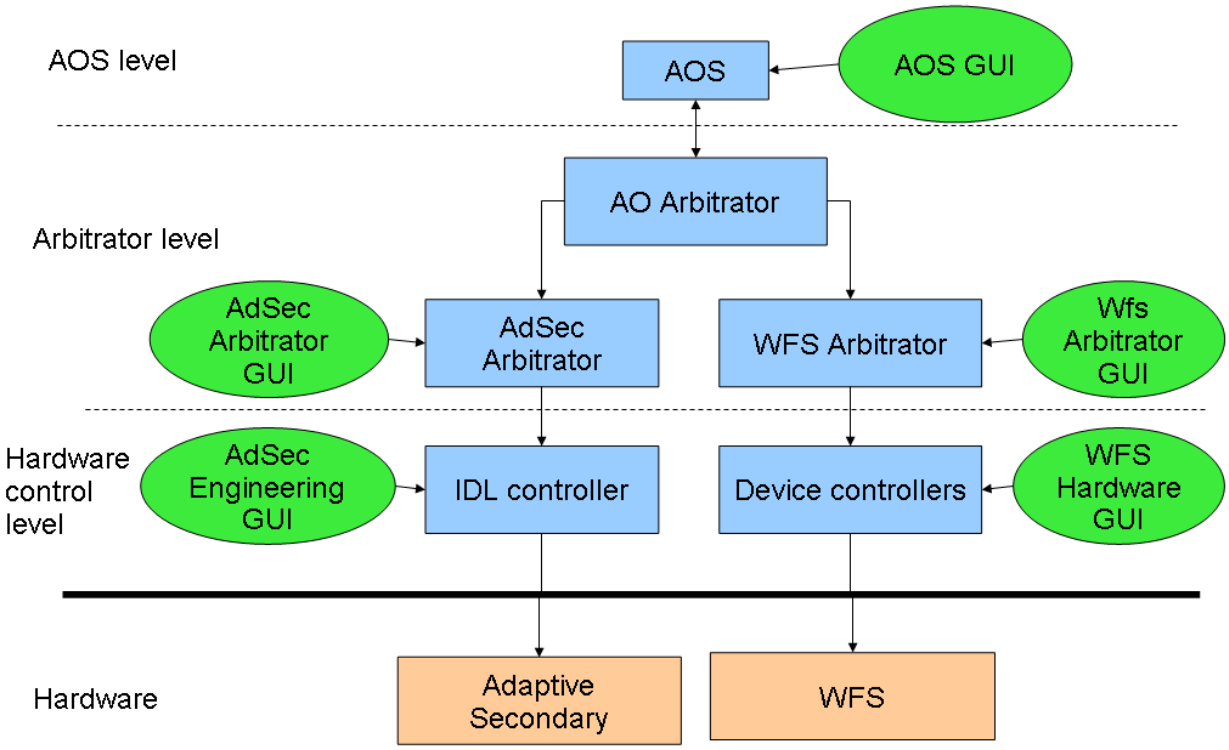
- **adsc_start** (on adsecdx) to start all adsec-related processes
- **adsc_stop** (on adsecdx) to stop all adsec-related processes
- **adsc_restart** (on adsecdx) to execute a stop followed by a restart
- **adsc_check** (on adsecdx) to check whether the adsec software is running

All these commands are text-based and can be issued by any text terminal on the control workstation. They will report the execution status and any errors which may arise. Multiple start commands will cause no harm.

The two software sets (wfs and adsec) are in constant communication but are independent and can be stopped/started separately in any order.

3.3. Overall software scheme

The following diagram gives a general description of how the AO software is structured:



Note the various horizontal levels: AOS, Arbitrator, Hardware control.

The following paragraphs give a short description of the various software components, including the three levels detailed above.

3.3.1. System processes

The lowest (and usually invisible) level is composed by the system processes which perform all housekeeping and message-passing tasks, and which maintain the overall AO status. A detailed knowledge of these processes is not required except for debug purposes, since at this level all operations are automatic.

However a list of the fundamental processes follows:

- MsgD-RTBD (message daemon and real-time database): one copy for each workstation is running. Manages message passing between all the other processes, maintains a central variable repository (similar to the telescope Data Dictionary), and manages shared memory

buffer for quick transfer of sizeable volumes of diagnostic data. Any kind of problem with the MsgD is usually fatal to the AO system, requiring a complete restart.

- MirrorController: the name is slightly misleading since this is the main hardware-communicator process also for the WFS. Manages communication with the Microgate BCUs onboard the Secondary Mirror and the WFS.
- MasterDiagnostic: manages the diagnostic data stream coming from the AO hardware
- Pinger: keeps an eye on the AO network and signals if something goes offline.

3.3.2. AdSec control processes

The actual AdSec control program is written the IDL language and is managed by the IdlCtrl process. The IdlCtrl allows command-line like access to the IDL control process for debug purposes, but the usual method of controlling the mirror is to use high-level interfaces like the AdSec Arbitrator (see chapters 3.3.4 and chapter 9). Various other processes handle housekeeping details of the mirror hardware, and perform continuous surveillance of the mirror safety. These safety mechanisms can shut down the mirror at any time, either in seeing limited or AO observations, if they detect some unsafe condition in the mirror shape or forces.

3.3.2.1. IDL issues

The use of IDL code requires an IDL license to be always available. Usually this is implemented with an IDL license server, to which the IDL program connects to verify that license validity. If this license server is not working or otherwise unavailable, the IDL program will not start (or stop in a short time if it was running). Failure of the IDL license server will cause the Adaptive Secondary to shut down for safety, and will cause some malfunctions, but not complete shutdown, on the WFS software.

3.3.3. WFS control processes

The WFS system does not have a single main hardware control process like the AdSec, but is instead distributed into a number of processes, each of which takes care of controlling a single hardware device. The coordination is then done in the Wfs Arbitrator process (see next chapter). IDL is used sparingly, but similar license problems exist as in the AdSec software.

3.3.4. Arbitrators

Coordination at the subsystem level (wfs and adsec) is done by the Arbitrator processes. The Adaptive Secondary has its own Arbitrator, as the WFS has. The Arbitrator hides the actual hardware implementation, and instead makes available a few high-level commands which implement the more common AO operations. The Arbitrator GUIs are the main interface to the AO system during engineering operations.

3.3.5. AOS

The AO system is interfaced to the rest of the telescope through the AOS (AO Subsystem). This is a normal TCS subsystem running on the TCS server farm. The AOS exports the AO commands (a dozen or so) needed to perform seeing-limited and AO observations. The AOS GUI is the main interface to the AO system during normal observation.

3.4. Engineering interface levels

In order to setup, calibrate and debug the system, a number of engineering interfaces are provided. Each interface works at a certain level, and is independent of the others. Thus, they can override each other and care must be taken not to give conflicting commands. These conditions are noted where possible in this manual.

As a general rule, an interface for a low-level process (for example the WFS hardware GUI) will override commands given from a higher-level interface (for example, one of the Arbitrators).

It is therefore recommended to work with the highest available level. In addition, experience has shown that the complexity of the system is such that, when using the low-level GUIs, many details can be forgotten or overlooked even by experienced operators. Usage of the high-level interface make things easier, because most things are performed by scripts which will ensure that all details are properly taken into account.

4. AOS GUI

The AOS telescope subsystem makes available to the TCS and IIF all the AO commands needed for observation. These commands are intended to be sent from either the TCS command sequencer or the instrument observing block, but they can also be manually issued from the AOSGUI by the telescope operator if needed.

4.1. Starting the GUI

The AOSGUI can be started from any TCS machine. The syntax is:

```
AOSGUI [side]
```

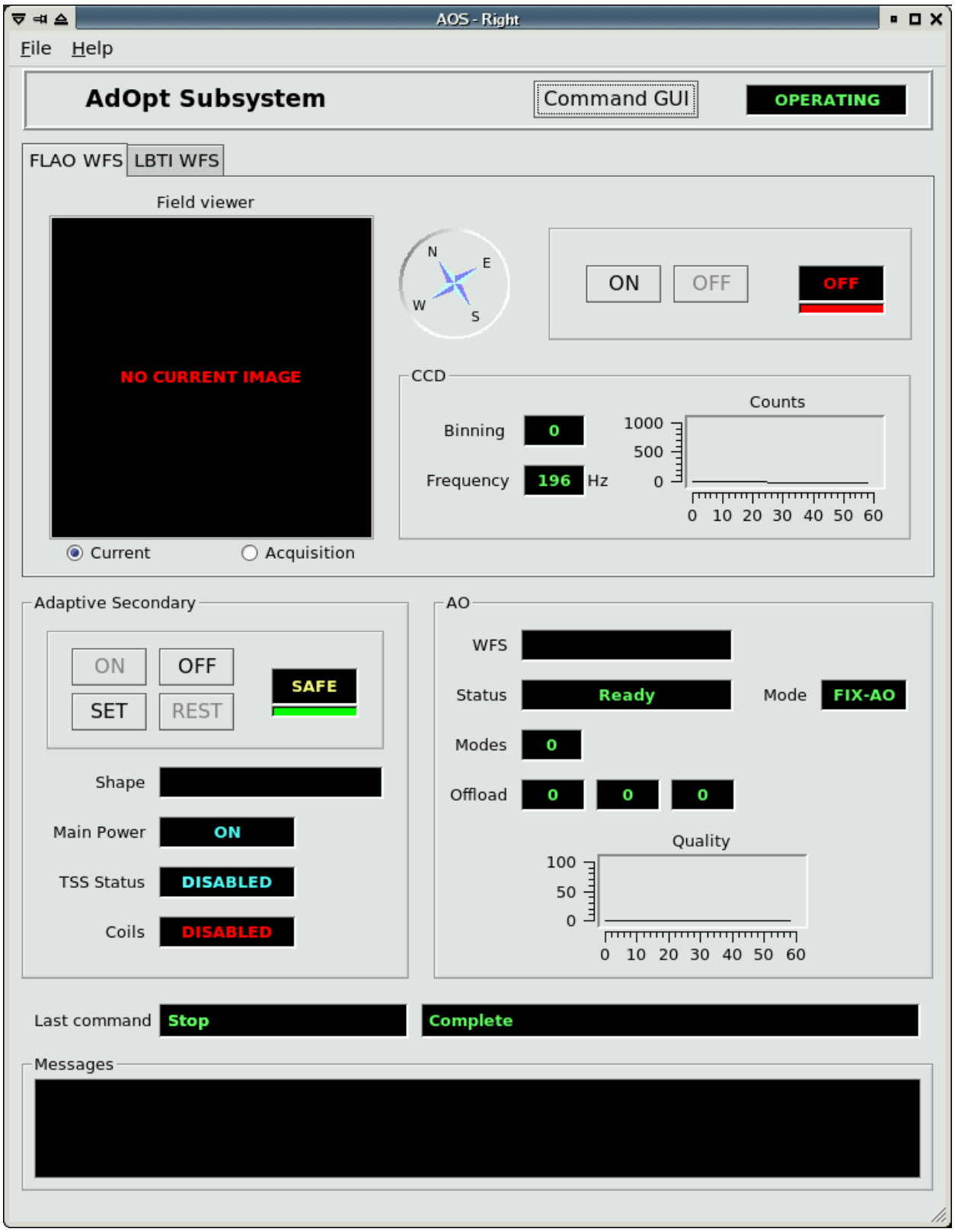
where [side] is either “left” or “right”. In case the side parameter is omitted, the left side is assumed by default.

The two purposes of this GUI are:

- display AO status information
- provide an interface to send commands to the AOS

4.2. Status information display

The main AOS GUI window is a status display with all main AO parameters. The window will become red if the AOS is not running properly.



4.2.1. Connection to the AO system

The connection status can have several values:

- **DISCONNECTED:** the AOS is not able to talk with the AO system. This may happen because the Adaptive Secondary software is not running on **adsecdx**;
- **NO ARBITRATOR**: the AOS is able to talk with the AO system, but the AO arbitrator is dwn or not answering. This usually means a problem on the network or on **adsecdx**;
- **STANDALONE**: the AOS is connected to the AO system, but the latter is indicating that it is not ready to receive commands from the AOS.
- **OPERATING**: the AOS is connected to the AO system and can send/receive commands.

4.2.2. Overall AO system status

The AO panel shows high-level parameters about the AO system:

- Mode: can assume several values:
  - FIX-AO: seeing-limited (“fixed”) mode
  - TTM-AO: tip-tilt only correction
  - ACE-AO: full AO correction
- WFS: shows which focal station has been selected
- Status: shows the overall AO state machine status
- Modes: shows how many modes are being corrected
- Offload: shows the magnitude of the current tip, tilt and focus offload
- Quality: …

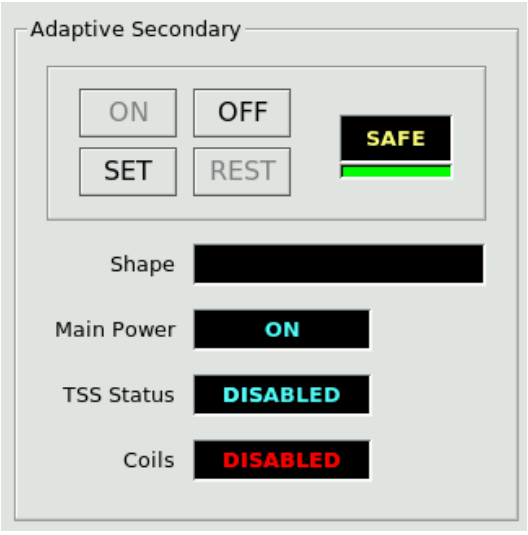
4.2.3. Wfs status and commands

The WFS status window shows the main WFS parameters:

- Field viewer: shows either:
  - The current technical viewer (ccd47) image
  - The technical viewer image used in the last source acquisition

  The compass right to the image shows the direction of the sky North
- CCD: show the current ccd binning, frequency (frame rate) and illumination level in counts in the last 60 seconds. Binning may be zero if the ccd is off.
- On/off: the two buttons ON and OFF control the power to the WFS unit. The ON button, in addition to simply turn on the power, will also perform a setup of the unit, thus taking some minutes to complete.
- Status label: the status label can show either ON or OFF depending on the WFS power status. The colored bar at the bottom represents the software status and will be either green or red, the latter case signaling a problem in the wfs software. There is currently no way of detailing or correcting such a problem from the AOS GUI, and the engineering interface must be used instead (see chapters 3.2 and 10.1 for how to check for software health)

4.2.4. Adaptive Secondary status



The Adaptive Secondary panel shows the following status information:

- Shape: name of the last loaded shape file (usually called a “flat”)
- Main Power: status (ON or OFF) of the three-phase power to the unit. Also controls power to the hexapod
- TSS Status: status of the wind protection system
- Coils: status of the voice coils of the adaptive secondary.

4.2.5. Adaptive Secondary on/off/set/rest

The adaptive secondary status indicator, right of the button group, can have three values:

- OFF: power to the unit is off
- SAFE: unit is powered on and in safe condition (shell rested)
- SET: unit is powered on and shell is set for observation.

Four buttons control the power and shell status of the Adaptive Secondary:

- On: turns on the power to the unit and goes to SAFE status. Takes about a minute to execute
- Set: sets the shell and goes to SET status. Takes about two minutes to execute.
- Rest: rests the shell from the set position and goes back to SAFE status. Takes a few seconds to execute
- Off: turns off the power to the unit and goes to SAFE status. Takes a few seconds to execute.

4.2.5.1. Safety locks

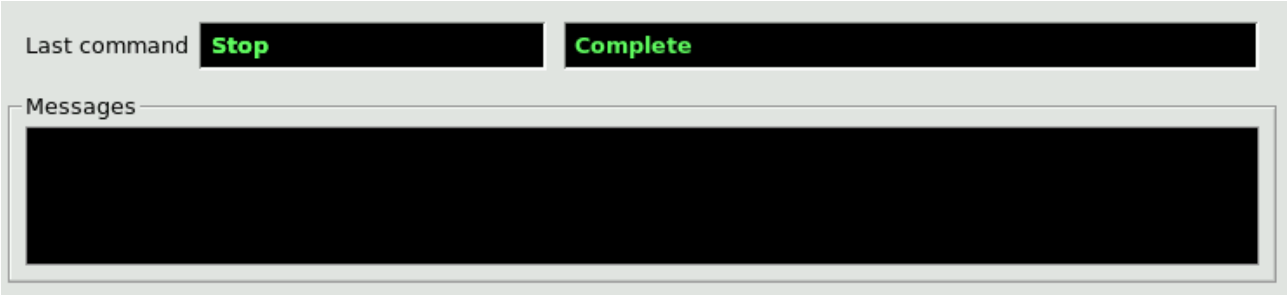
In order to ensure the safety of the Adaptive Secondary, the shell can be set only if the following conditions are met:

- Telescope elevation is 26 degrees or higher
- Swing arm is deployed
- Wind speed is under 8 m/s

If any of these conditions is not satisfied, the Set command will be refused. If the shell was already set, it will be rested immediately. The safety feature is fast enough even in the worst case of the telescope slewing down to zero degrees.
If for some reason any of these information do not reach the AO system (for example, the elevation value stops updating), it will be treated as an out-of-range condition and trigger the safety lock.

The colored bar at the bottom of the status indicator represents the software status and will be either green or red, the latter case signaling a problem in the AdSec software. There is currently no way of detailing or correcting such a problem from the AOS GUI, and the engineering interface must be used instead (see chapters 3.2 and 10.1 for how to check for software health)

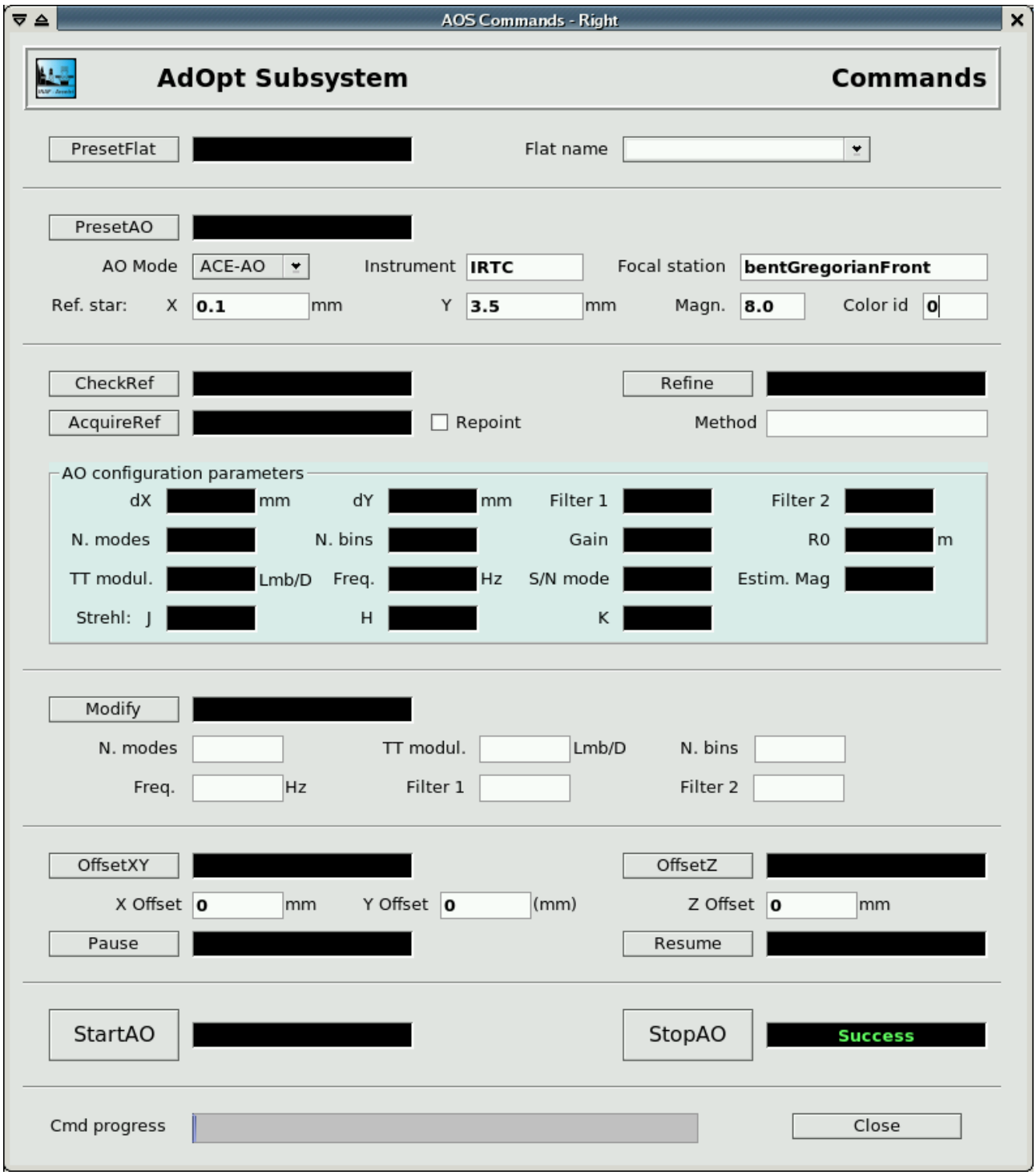
4.2.6. Command execution reporting



When any command is started, the name of the command is reported in the lower part of the GUI, along with the string “Running”. When the command completes correctly, the string “Complete” is reported. If there is any error, the error string is reported instead, and added to the message box below. Additional messages may appear in the message box during command execution, offloads, etc.

4.3. Command GUI

Clicking on the “Command GUI” button at the top of the AOS GUI opens the Command GUI sub-window:



4.3.1. AO commands

Each AO command has its own command button. Many command have parameters which appear next to the corresponding button.
The parameter input boxes serve both as input and as output: when a command is sent by the IIF, the corresponding parameters are written into the input boxes. Alternatively, the operator can input the parameter manually (or modify the ones written before automatically) and send the command manually.
Each command has a status indicator next to it which can have three values:

- Running: the command is currently executed
- Success: the command has completed successfully

- Failure: the command could not complete because of an error. Additional error information is available on the main AOS GUI window.

Error conditions include the refusing of a command because it was not allowed in the current AO status.

The progress bar at the bottom shows the command execution progress with respect to the command timeout. It is not possible at the moment to interrupt a command during execution.

5. Engineering GUIs

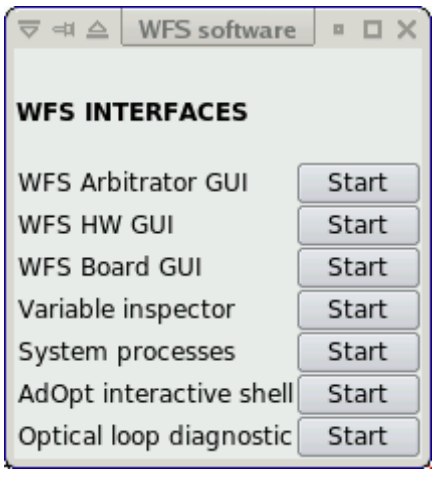
Unlike the AOSGUI, all the AO engineering GUIs must be started from the AO control computers (see chapter 3.1).

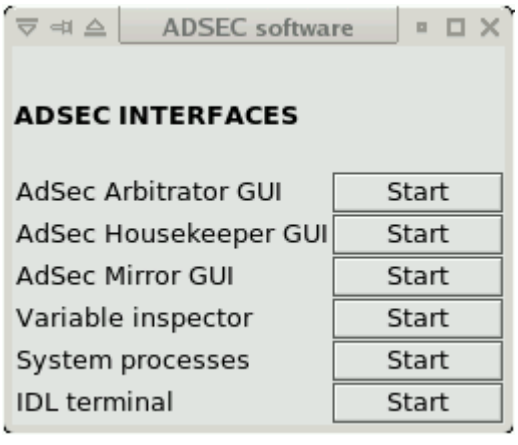
5.1. Starting the Engineering GUIs

A quick-start panel exists on both computers to start the relevant engineering interfaces. This panel is called:

- **wfseng** on wfsdx
- **adsceng** on adsecdx

The panels can be started typing their name on any terminal (an X connection must be present). Each panel shows the interfaces available for the system, which can be started clicking the “Start” button next to their name. Unless otherwise noted, multiple copies of the interfaces can be started without limitations.





WFS and Adaptive Secondary quick-start panels.

Alternatively, all interfaces can be started typing their name on a terminal. The correct program name is noted in the description of the interface, and is resumed here:

- WfsControl (on wfsdx) starts the Wfs Arbitrator GUI

- AdSecControl (on adsecdx) starts the AdSec Arbitrator GUI
- AdOptControl (on adsecdx) starts the AO Arbitrator GUI
- wfshw.py (on wfsdx) starts the Wfs Hardware GUI
- BoardGui (on wfsdx) starts the board status display
- AdSecMirGui (on adsecdx) starts the mirror status display
- ccd_viewer.py (on wfsdx) starts the ccd viewer
- vartool_AO.py (on either computer) starts the RTDB interface (viewer/editor)

These GUIs are described in detail in the following chapters.

6. Wfs board status GUI

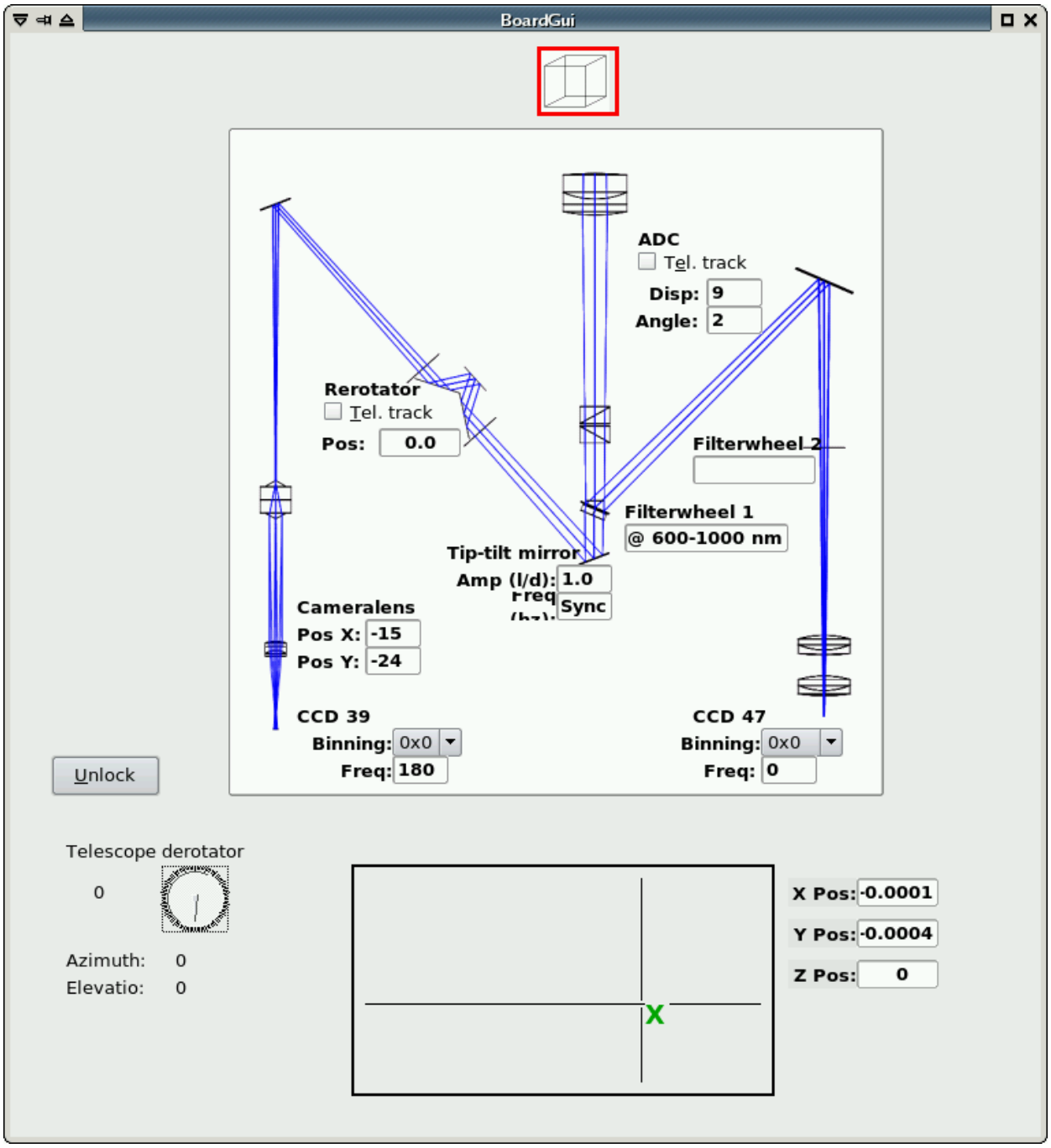
The Wfs board GUI shows the status of the various WFS devices.

6.1. Starting the GUI

The WFS Board Status GUI can be started from the wfseng panel (see []), or from a terminal on wfsdx with the following command:

```
BoardGui
```

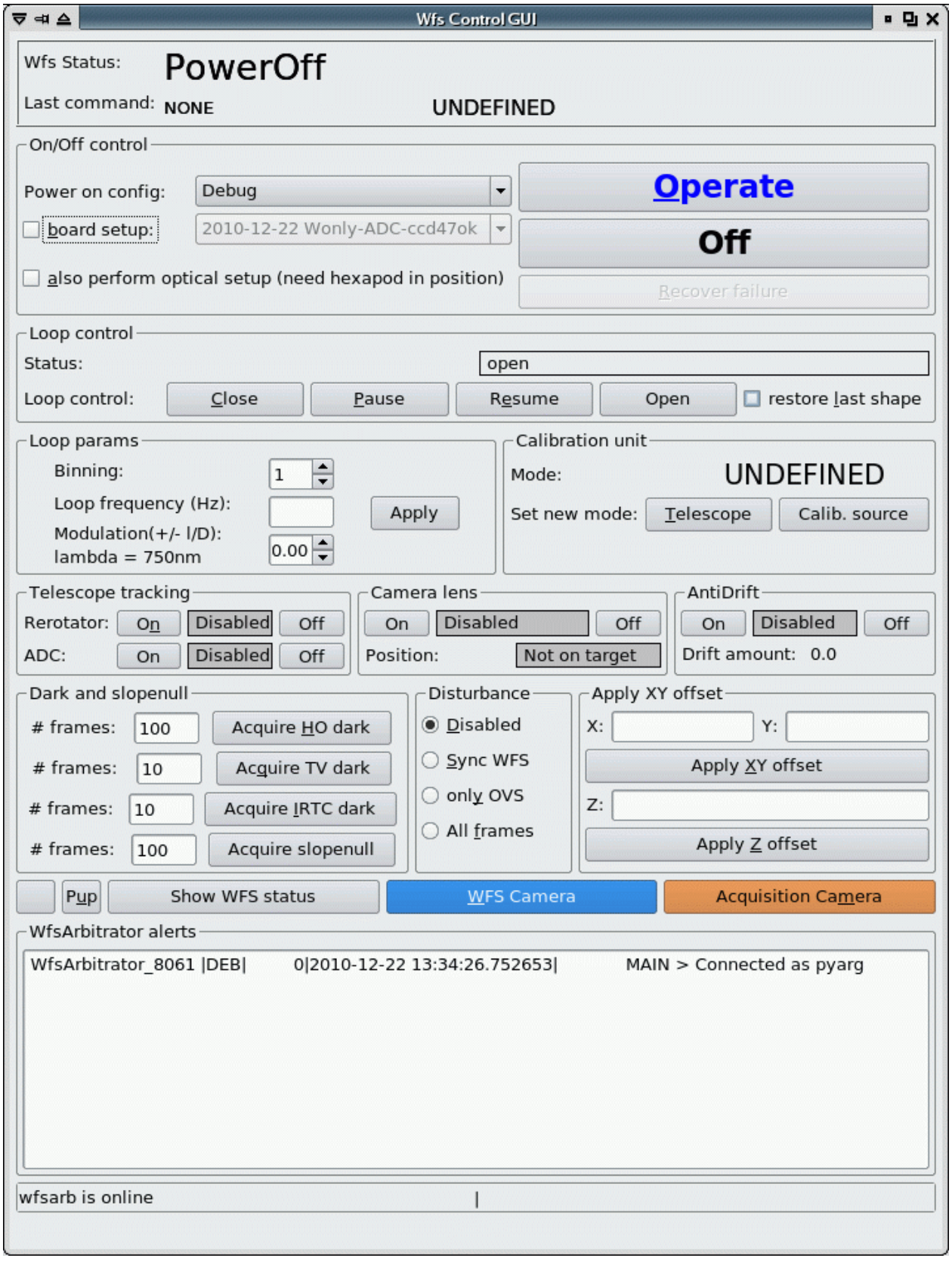
6.2. GUI description



The GUI is a read-only display of the position of the various WFS devices. For each device, only the most relevant information is shown (e.g. filterwheel position, ccd binning and integration frequency, etc). The Cube icon at the top moves to show the actual physical position of the cube beam splitter, and has a red outline when it is in the reference source beam path. The rectangular display at the bottom is a map of the focal plane FoV available to the WFS stages, and the current position is shown with a green ‘X’. When the stages are moving, a red circle is also drawn to indicate the target position. On the lower left, a resume of telescope position and rotation is shown.

7. Wfs Arbitrator GUI

The WFS arbitrator GUI is used to send commands to the WFS Arbitrator, which provides high-level commands to manage the WFS like startup/shutdown procedures, wfs configuration, dark frame acquisition, etc. Commands are implemented in the WFS state machine as described in CAN687f400.



7.1. Starting the GUI

The WFS Arbitrator GUI can be started from the wfseng panel (see []), or from a terminal on wfsdx with the following command:

```
WfsControl
```

7.2. GUI description

All GUI actions are implemented as one of the arbitrator commands described in CAN687f400. This has several consequences:

- only one command can be executed at a time. To send another command, one must wait for the previous command completion. The GUI will prevent the operator from sending multiple commands, graying out all buttons while a command is executing
- Not all commands are available at all times, depending on the state machine status. The GUI will either gray out buttons corresponding to unavailable commands, or display an error box if the command could not be received.
- Commands parameters are validated before execution. If an out-of-range or otherwise invalid parameter is entered, a “Validation failed” error will be displayed.

7.2.1. Status indicators

At the top of the GUI, the following status information is shown:

- Wfs Status: tells the operator in which state the WFS is at the moment, and therefore which commands are available. Also, in the GUI commands which are not available at the moment are grayed out.
  Note: if the Wfs Arbitrator program is not running, or not correctly responding, the Status will be “offline” and no commands will be executed.
- Last executed commands: shows the name of the last command executed by the WFS arbitrator
- Command execution status: shows whether a command is executing at the moment, or the result of the last command as described in []. When a command is executing, all GUI buttons are inactive.

7.2.2. Startup/Shutdown commands

Starting up the WFS requires turning on the various wfs devices in the correct order. This is managed by the WFS arbitrator command “Operate”, that takes two arguments: a list of devices to turn on (the “configuration”) and optionally a command file to setup each hardware device (the “board setup file”). The configuration files are pre-determined by the programmer and are not generally modifiable by the operator, while the setup files can be found in the WFS calibration directory as described in [] and may be modified as needed. For correct operation from the AOS, at least one setup file with the same name as the current instrument (e.g. “IRTC”) must be present.

To startup the WFS:

- select the configuration in the drop-down box
- (optionally) select the setup file in the drow-down box, checking the “apply setup” checkbox
- press the “Operate” button.

Execution of the Operate command will often take several minutes.

To turn off the WFS:

- press the “Off” button

Execution of the “Off” commands takes a few seconds.

The startup sequence turns on and prepares for operation all devices in the configuration list. Movements are homed, CCDs are configured to default values, etc.

7.2.3. AO parameters

The “parameters” section is used to configure the main AO-relevant parameters: ccd frame rate, binning and tip-tilt modulation. These parameters are applied together.

- enter the desired ccd frame rate in the “frame rate” input box. Frame rate is in Hz and can range from 100 to 1000.
- enter the desired binning in the “binning” input box. Available binnings are 1,2,3 and 4.
- enter the desired tip-tilt modulation in the “modulation” input box. This value will be the modulation radius in lambda/D. Modulation radius can range from 0 to 6 lambda/D.
- Press the “Apply” button.

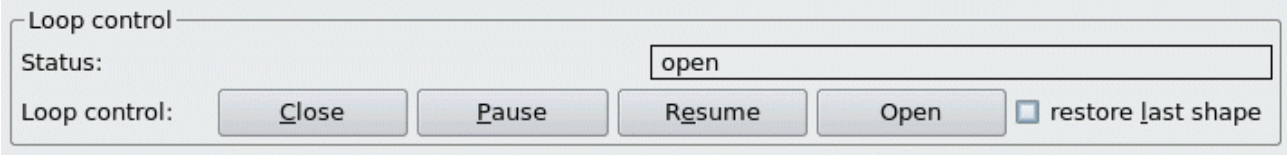
The WfsArbitrator will apply the new parameters and, if the ccd configuration is changed, take a new dark frame using the filter wheel #1 as described in the later chapter []. Execution of the command will be either very short (less than 1 second, if the ccd parameters are unchanged), or will take 20-30 seconds.

If the ccd or tip-tilt configuration is changed using the lower-level hardware GUI, the Wfs Arbitrator may not realize this. To avoid problems, after using the hardware GUI, always reapply a different value from the last command, to force the Wfs Arbitrator to reconfigure.

Note about the modulation: the modulation parameter is converted to tip-tilt voltage commands using a lookup-table, found in [], based on the ccd frame rate. Not all modulation values are available at all ccd frame rates, for example at a frame rate of 1000 hz the maximum modulation radius is 3 lambda/D. In general, higher frame rates will prevent to use the bigger modulation settings, but a precise specification cannot be given since it depends on the details of the lookup table. The GUI will give an error to the user when an incorrect modulation is entered.

During the setup, the ccd display may fluctuate wildly while the background is taken, and even stop for a minute or so if the ccd binning is changed. This is normal and the display is not to be considered valid until the command has completed.

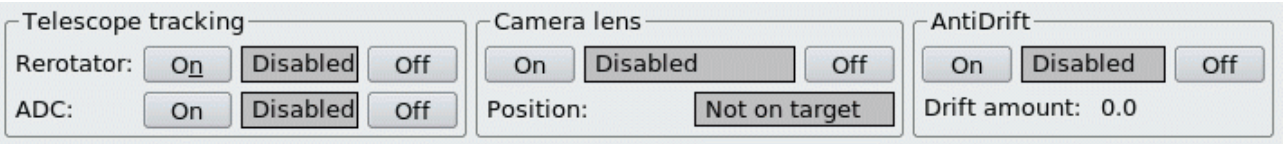
7.2.4. AO loop open/close/pause



Four buttons manage the AO loop status. Because of the need of coordination between the two systems, the Close and Open buttons will send commands to both the WFS and Adaptive Secondary.

- Close: start sending slopes to the Adaptive Secondary. The secondary must have been previously configured with the correct input port, reconstruction matrix, etc. The button first sends a command to the Adaptive Secondary to configure it with the expected frame rate, and then closes the loop on the WFS.
  This button will also configure the Adaptive Secondary with the disturbance setting, as described later in []
- Pause: suspends the loop stopping the slopes. The secondary mirror remains freezed in shape it had during the last loop iteration. A paused loop can be either resumed or stopped.
- Resume: resumes a previously paused loop. Before resuming, the illumination level on the ccd39 is checked to verify that it similar to the one present when the loop was paused, and the resume command may be refused if the illumination level is too low.
- Open: opens the loop stopping the flow of slopes to the Adaptive Secondary. After that, sends a command to the Adaptive Secondary to inform it that no more slopes are expected. If the “restore last shape” box is checked, the secondary mirror will re-apply the last shape it loaded before closing the loop. Otherwise, the mirror will remain in the position it had during the last loop iteration.

7.2.5. Rotator tracking



When enabled, the Wfs arbitrator will move the pupil rerotator to follow the telescope derotator and keep stable the pupil image on the ccd39. The tracking is applied once per second and has a total delay of 1-2 seconds, which gives an error <0.1 degrees in all observing conditions up to 87° degrees of elevation. When first activated, or when the telescope is slewing, the tracking may take some time to reach the correct position.

The pupil rerotator position is computed with the following formula:

rerotPos = derotPos/2.0 * rerotSign + rerotOffset + trackingOffset

The <derotPos> values comes from the AOS and is the DD derotator position. The <rerotSign> and <rerotOffset> values are read from the Wfs Arbitrator configuration file, using the keywords "RotatorSignBin1", "RotatorOffsetBin1" and similar for the other binnings. If the Sign keyword is missing, a sign of -1 is assumed.
The <trackingOffset> parameter is read each time from the RTDB (see []) and is intended to allow small corrections by the operator.

This tracking is always activated by the AOS when starting an observation.

7.2.6. Camera lens tracking

When enabled, the Wfs arbitrator will measure the current pupil position using feedback from the "pupilcheck" process, and move the camera lens to keep the pupil centers in a predefined position. The status indicator has three values:

- disabled: tracking loop is off and camera lens is not moving
- enabled (not on target): tracking loop is on, but the pupils are off the predefined position, and the camera lens will be moved to recenter them;
- enabled (on traget): tracking loop is on and the pupils are on the predefined position within 0.1 pixels; the camera lens will be left where it is.

Regardless of the "enabled" status, camera lens corrections are only applied during closed loop. This happens because, in open loop, the pupils are too aberrated to have an accurate measure of their position. A consequence of this is that, when the loop is closed, the indicator will temporarily go to "not on target", because the pupil position is only computed every four seconds, and it will take one or two iterations before the actual pupil position is reflected in the target indicator. For this reason, the first camera lens loop iteration is skipped by the WfsArbitrator after closing the loop.

This tracking is always activated by the AOS when closing a loop.

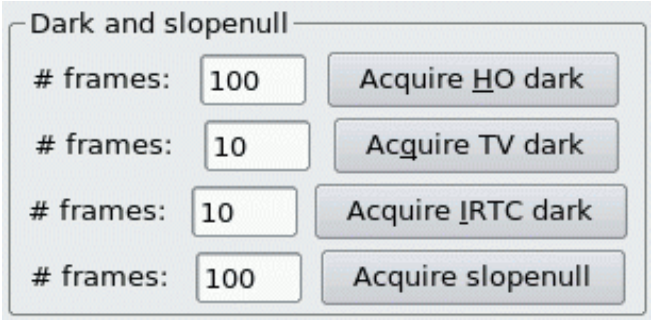
7.2.7. ADC tracking

The ADC tracking will follow the telescope elevation and field orientation moving the ADC wheels to have the correct atmospheric dispersion correction. Operation is similar to the Rotator tracking, except that instead of using a formula, the ADC position is computed using a lookup table found in [] based on the telescope elevation.

7.2.8. Anti drift

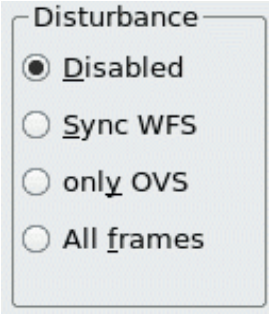
The antidrift loop tries to correct for temperature-dependent drifts of the ccd39 background levels. It will do so checking the current background level outside the pupils (in the ccd corners), and adjusting the current background frame so that the background-subtracted levels are zero. This is done independently on the four ccd quadrants.
Since the BCU can lockup if the background frame is overwritten while the loop is closed, an antidrift correction will temporarily stop the ccd integration, overwrite the background frame, and restart the ccd integration. Because of the slow serial connection to the ccd, this will result in pause of about 0.1 seconds. The antidrift correction will be applied at a maximum rate of 1 Hz, and will slow down as the ccd temperature stabilizes.

7.2.9. Dark frame and slopenull acquisition



These buttons allow the operator to take a dark frame for either ccd, or a slope null frame. To take a dark frame, enter the number of frames to average in the input box next to the button.
When the button is pressed, the filter wheel #1 is rotated to the “silver mirror” position (for the ccd39 background), or to the “empty” position (for the ccd47 background). For the other two buttons, no rotation occurs. After that, the ccd bias levels are equalized (see hardware GUI []) the specified number of frames are integrated, averaged, saved on disk with a tracking number, and sent to the correct BCU as the new background. The ccd display may fluctuate wildly during integration. The ccd39 background acquisition is a sub-procedure of the AO parameter apply command (see []).

7.2.10. Disturbance



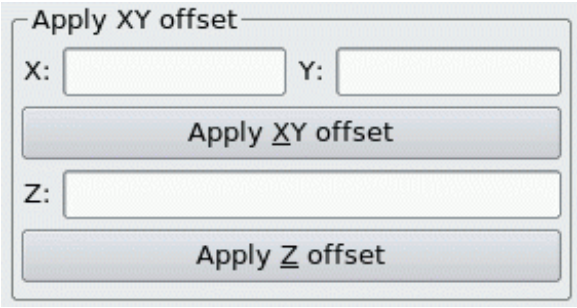
Disturbance application is used to manage the digital disturbance feature of the adaptive secondary. The disturbance commands are loaded on the secondary BCUs, but their application is commanded by the WFS with a bitmask sent together with the slopes. Therefore, to change the disturbance setting, the system must be in closed loop. There are four possible settings:

- Disabled: no disturbance is applied
- Sync WFS: one disturbance frame is applied at each optical loop iterations
- Only OVS: disturbance frames are only applied on oversampled frames (see [])
- All frames: disturbance frames are applied on both optical loop and oversampled frames.

The last setting allows the operator to have a disturbance "running" at a multiple of the optical loop speed, for example the optical loop can be at 200 Hz while the disturbance is applied at 800 Hz. Since oversampled frames are applied at a maximum rate of 890 Hz, this feature is only useful for an optical loop speed up to 445 Hz. Over this speed, the disturbance can only be synchronous with the AO loop.

The disturbance setting can be changed at any time, but will only be applied when in closed loop. If changed in open loop, it will be applied at the first loop iteration.

7.2.11. Offsets



The WFS Arbitrator can execute XY and Z offsets moving the stages which support the optical board. All offsets are specified in millimeters on the focal plane (for the XY offset) or along the optical axis (Z). Offsets commands are relative to the current position.

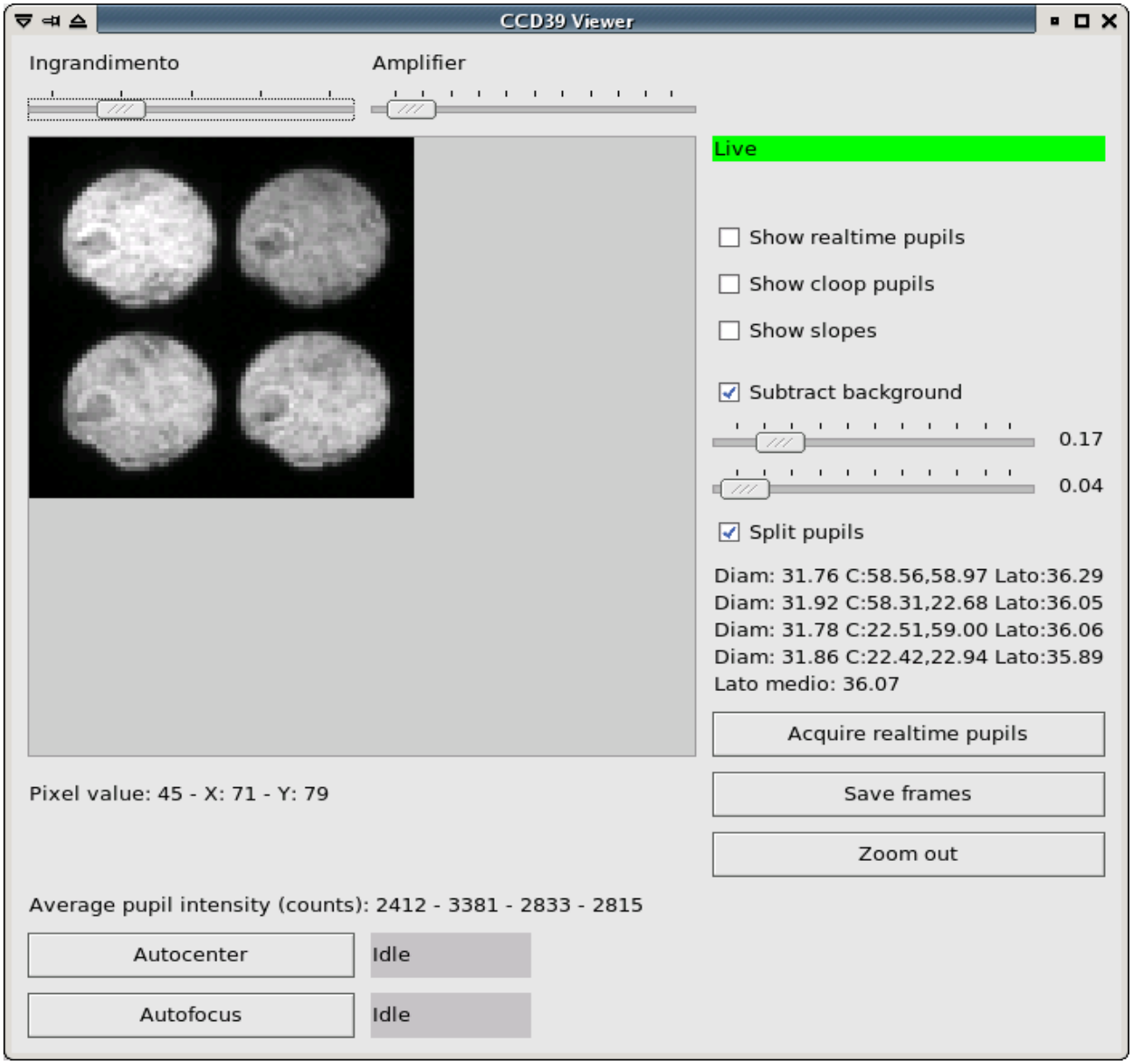
Offsets can be executed regardless of the loop status, but care must be taken not to exceed the adaptive secondary tilt or focus range if the loop is closed. By rule of thumb this means about 0.5 millimeters in either X or Y, and 5 millimeters in Z. If the low-order offload is active, the adaptive secondary will offload these tilts to the hexapod in a few seconds, and the offset can be repeated (this coordination is done automatically by the AOArbitrator when long closed loop offsets are requested by the AOS).

XYZ stages are normally braked. The brake is opened when an offset is requested, and closed again when it is completed. This can cause very small jitters (in the order or microns) in the actual stage position. Multiple offset commands are applied using the target position, and not the actual stage position, so that these errors are not accumulated.

7.2.12. WFS displays

From the WFS Arbitrator GUI, the two ccd viewers can be started pressing the “WFS Camera” and “Acquisition camera” buttons (for ccd39 and ccd47 respectively). The display program is the same for all ccds, with a few additional features for the ccd39

7.2.13. CCD display



- live indicator: the indicator can be either “Live” (green) or “Not live” (red). “Not live” only means that the display is not receiving frames, and may result from a variety of causes. For example, a “live off” display is normal while the ccd is changing binning.
- image: the image is always shown at a rate of 20 Hz (or slower if the ccd is going slower), to avoid using excessive amount of CPU. This means that, at high AO loop frame rates, the image is heavily decimated. Saturated pixels are shown in red, to avoid confusion with pixels which are white due to lookup table effects.
  The ccd39 image is rotated 90° from the raw one to correct for ccd orientation.
- Pupil positions (ccd39 only): radius, center X and Y position, and interpupil distance are shown for each pupil. This information is refreshed every few seconds.
- Pixel value and position: when moving the mouse over the image, the pixel value of the pixel under the mouse, and its X and Y positions, are shown. The zero position is in the upper-left corner.
- Intensity value (ccd39 only): shows the total intensity value, averaged over the four pupils, and rescaled to be in photons/subaperture/frame. This value is a running mean over the last 100 displayed frames (5 seconds), so can be incorrect in case of rapid fluctuations or while the background frame is being acquired. This value is also incorrect the background frame is missing or outdated.
- Slope rms plot: the plot is continously updated wi the current slope rms and shows the last 20-30 seconds of data.
- Stages on/off: this is a simple indicator to remind the operator that the stage motors are enabled, and may compromise the loop injecting electrical noise. This can be an issue when using the hardware GUI, but is automatically managed when using the AOS or Wfs arbitrator GUI.

7.2.13.1. Controls

- magnification slider: changes the ccd display magnification
- amplifier slider: changes the display lookup table. Higher (towards the right) slider settings cause the lookup table to shift towards low value pixels, while higher value pixels are saturated to white.
- “show realtime pupils” checkbox: when checked, four red circles are drawn on the image to show the current pupil position and diameter as calculated by the “pupilcheck” process.
- “show cloop pupils” checkbox: when checked, four red areas on the ccd are highlighted in red to show the pixels selected for the AO closed loop. The illuminated pupils must coincide with these areas
- “show slopes” checkbox: when checked, the ccd image is replaced with a slope map which show the current slopes as calculated by the BCU. Two maps, for X and Y slopes, are shown. The pixel value indicator shows the value of the slopes under the mouse cursor position.
- “Save frames” button: opens an interface to save frames from the ccd, described in [].
- “Autocenter”/”Autofocus” buttons: start the automatic autocenter/focus scripts. These scripts use the current ccd image as feedback and move the XY or Z stage to center the light on the four pupils, or bring them into focus. Green arrows are drawn over the ccd image to show the stage movement. The scripts will exit when a correct position is reached, signaled by a small green circle display on the ccd image. The “Stop autocenter/focus” button can be used to stop the scripts manually.

8. WFS Hardware GUI

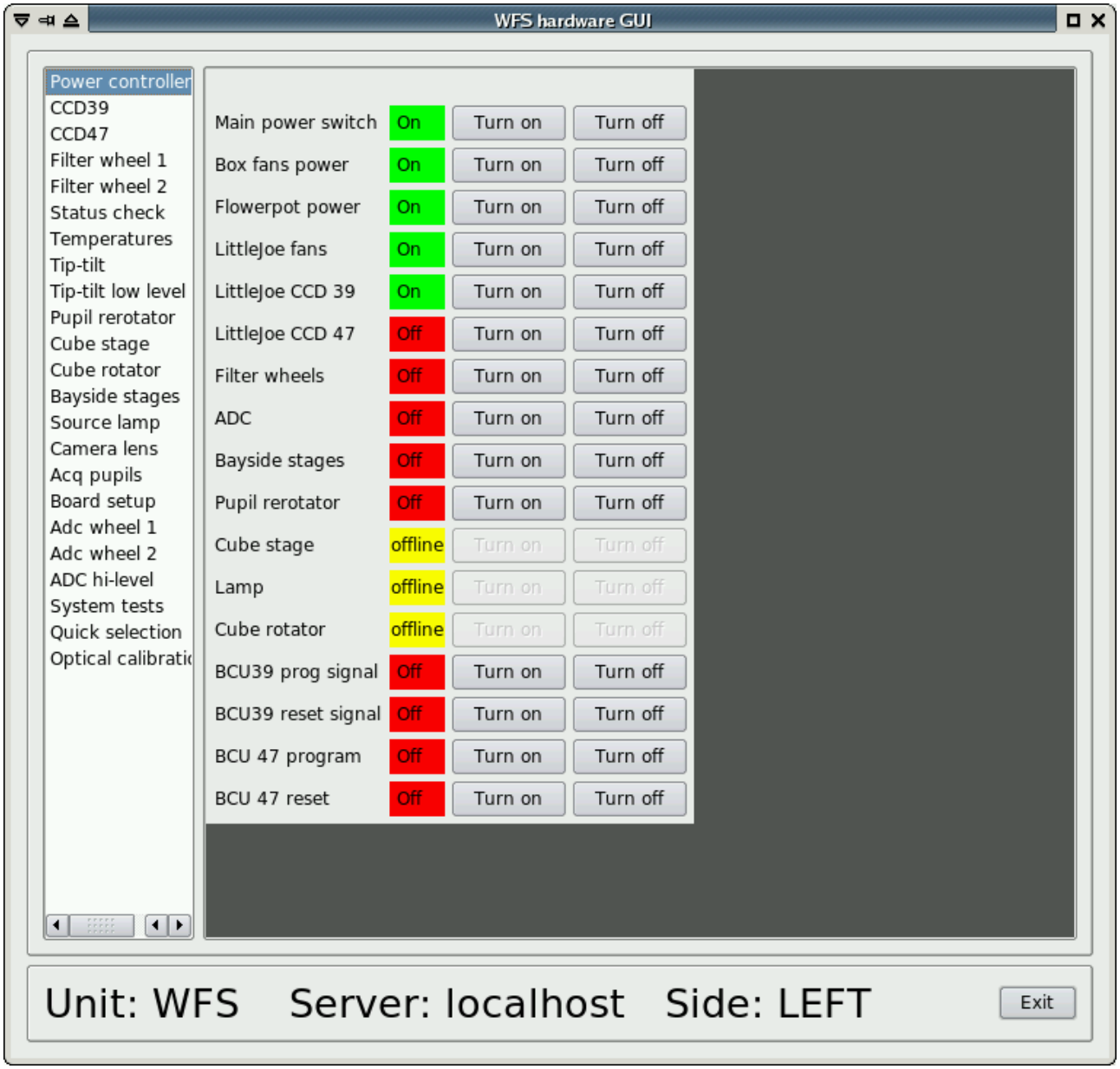
The WFS hardware GUI allows low-level control of the wfs devices.
Warning: if the higher-level Arbitrators and AOS software are running, their command can conflict with those sent by the operator. Always use the higher possible GUI level, unless a specific reason exists.

8.1. Starting the GUI

The WFS Hardware GUI can be started from the wfseng interface or started from a terminal on wfsdx with the following command:

```
wfshw.py
```

8.2. GUI description



The Hardware GUI has a list of devices on the left side. Clicking on a device name will display the corresponding window on the right side. At the bottom, three displays help identify which WFS is being operated on:

- Unit: identifies the WFS by name (may be W1, W2, etc.)
- Server: identifies the computer operating the WFS. Generally “localhost”, meaning that it is the same computer where the GUI is running, but can be different as the GUI can run somewhere else if properly configured.
- Side: shows the telescope side (either “RIGHT” or “LEFT”).

8.2.1. Power controller

(pictured above)
This panel shows all the on/off switches in the wfs system. The switches can be controlled by different hardware devices, and so some or all of them may be unreachable if the controlling device is powered off. The GUI shows this condition graying out the on/off buttons, and marking “offline” their status.
A minimum set of devices is kept always on, as long as the input 110VAC line is active. These are:

- the MiniMC fiber/copper Ethernet converter
- the internal 5-port Ethernet switch
- the left-box TS8 Ethernet/serial converter
- the PIC-based power board

Correspondingly, a few (on W#1) or most (on W#2) power switches are always available because they are located on the PIC-based power board.

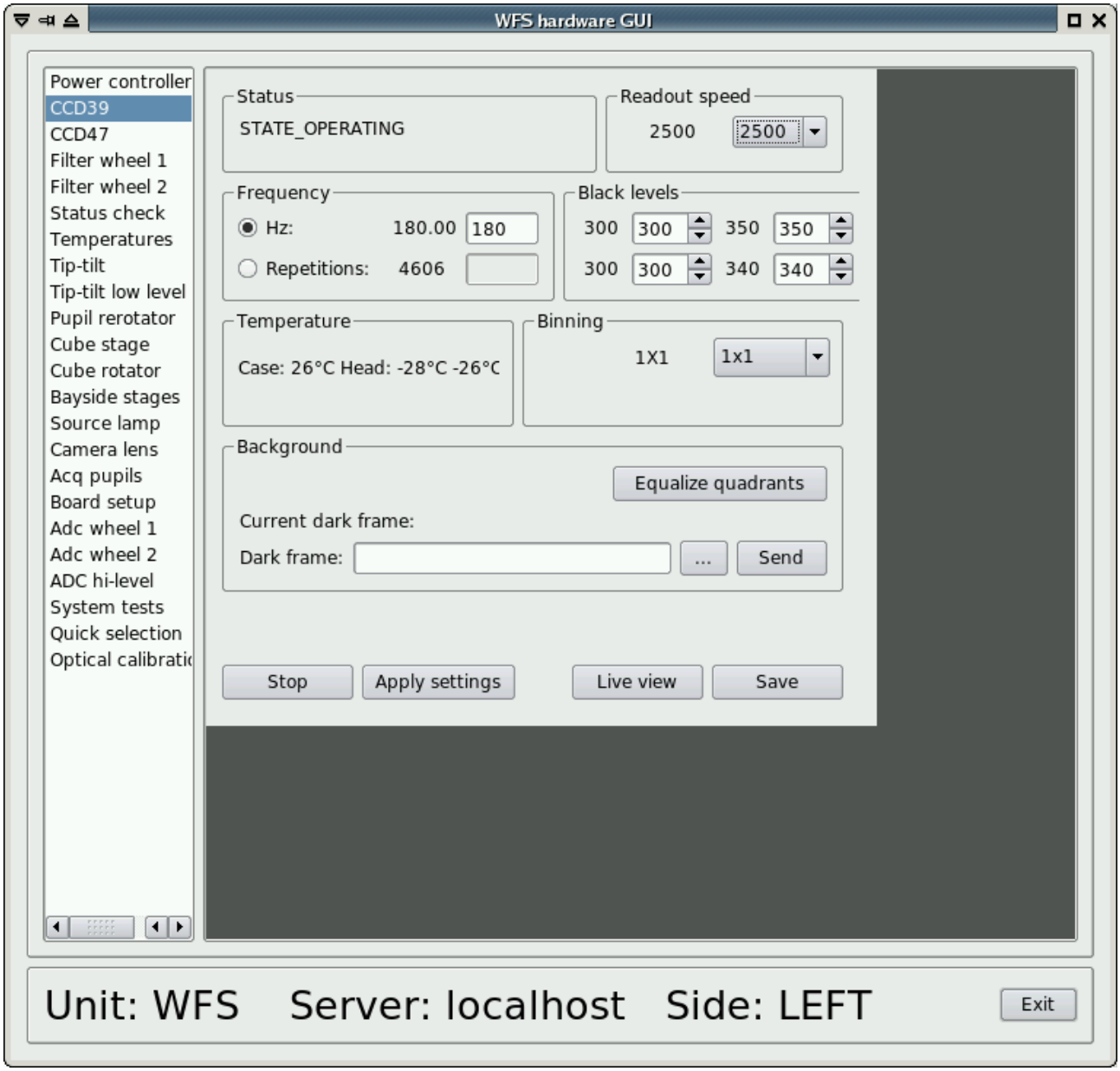
A list of the switches and what they do follows:

- Main power switch: controls the internal power supply for CCDs and BCUs.
- Box fans: controls the fans on the electronics boxes covers.
- Flowerpot: controls the flowerpot board, which in turn will allow control of the cube and reference lamp.
- Little joe fans: controls the fans on the little joe ccd controllers.
- Ccd39 and 47: controls the two little joe controllers.
- Filter wheels: controls the two filterwheels. The wheels will move to the home position upon starting.
- ADC: controls the two adc wheel motors. The wheels will move to the home position upon starting.
- Bayside stages: on W#1, controls the 110 V power supply for the stages motor. The stages will not move and will need to be homed manually from their panel (see [])
- Pupil rerotator: controls the pupil rerotator. The movement will move to the home position upon starting.
- Cube stage and rotator: controls the two motor controlling the cube position and rotation. Both movements will move to the home position when starting.
- Lamp: controls the reference lamp on (an additional intensity control is available separately)

Four more switches control the “reset” and “program” lines of the two BCUs. The “reset” lines will hold the BCU in reset status for as long as they are on. The “program” lines are sampled by the BCU when starting (or when the “reset” line is turned down) to select one of two internal memory

banks from which to load their program. These four switches are not normally needed for operation, unless a hardware problem on the BCU arises.

8.2.2. CCD39



The ccd39 panel shows the current ccd39 status and parameters, and allows the operator to change those parameters.

8.2.2.1. Status

Can have the following values:

- NOCONNECTION: ccd controller is either turned off or not reacheable over the network
- CONFIGURING: configuration parameters are being loaded through the serial line
- READY: ccd controller is ready, but not integrating frames.

- OPERATING: ccd is integrating frames

8.2.2.2. Controls

When first starting, only the binning drop-down box is available, because the ccd must be configured with one of the available binning. Selecting a binning from the drop-down box causes a configuration program to start, running in a separate xterm to display debug information. While this xterm is open, the hardware GUI is freezed. Binning configuration takes about 30 seconds.
Once a binning is configured, the other parameters are set to some default value and can be adjusted by the operator.

Parameters are set from the panel input boxes and then applied together when the "apply settings" button is pressed.

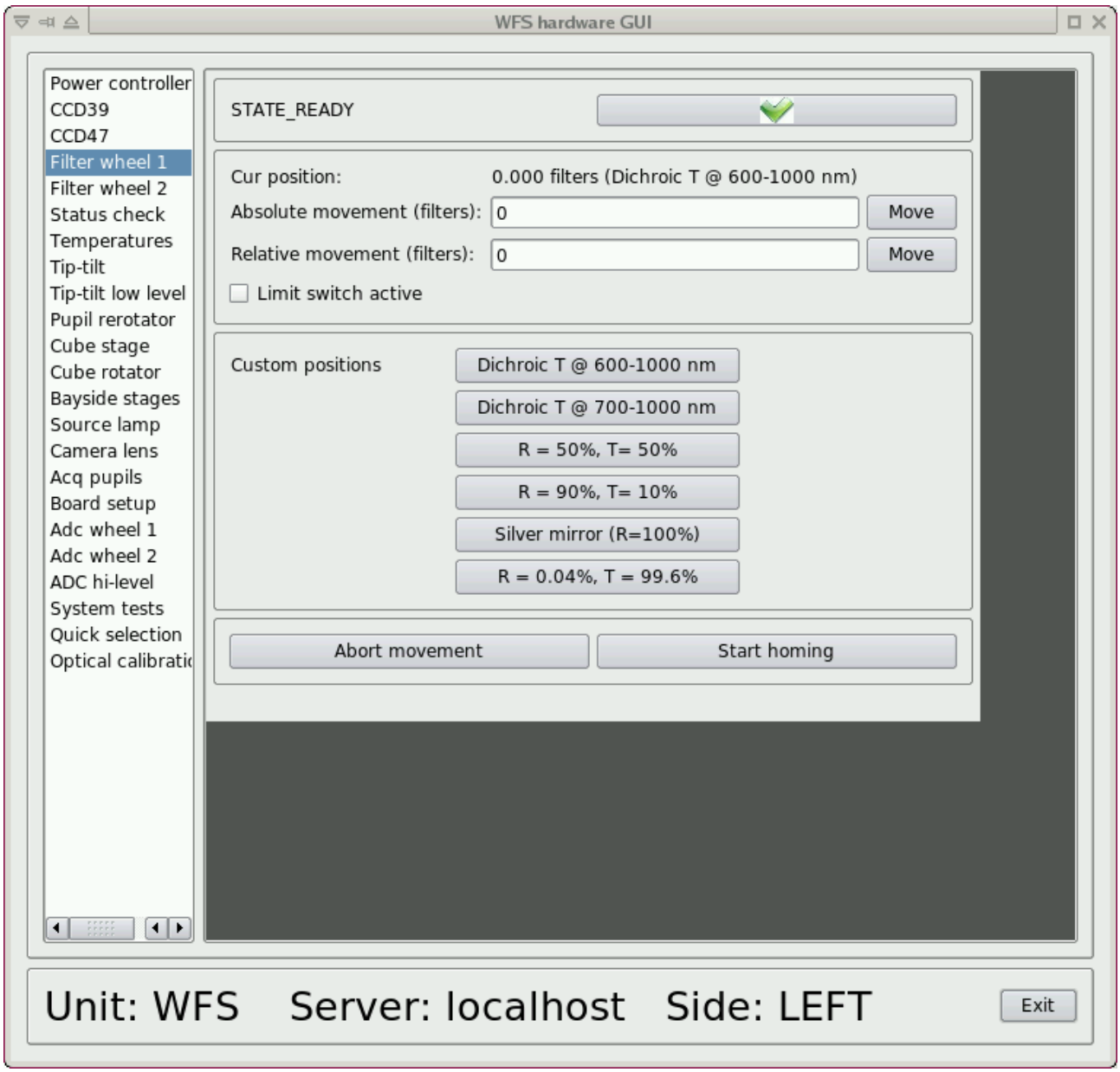
- **Frequency/repetitions**: either a frequency (frame rate) or "repetitions" number can be entered. The little joe controller cannot integrate a specified frame rate, but has instead a delay for integration with a minimum time defined by the frame readout time, plus a delay computed as the number of "repetitions" multiplied by a base delay interval (variable for each readout speed). When a frequency value is entered, the control program will approximate it with the closest possible value as allowed by the current delay interval. Typical errors range from a fraction of a Hz at low speeds up to 1 or 2 Hz at high speeds (1000 hz or more). The maximum possible repetition number is 65535.
  SAFETY WARNING: the tip-tilt mirror is often hardware-locked to the ccd frame rate (see tip-tilt section). The ccd39 has the capability of going much faster than 1 Khz, thereby entering the tip-tilt resonating frequency range and possibly breaking it. For this reason, the GUI will refuse frequency settings over 600 Hz, and the Wfs Arbitrator GUI must be used (the Wfs Arbitrator will perform the necessary safety checks and refuse unsafe settings). However, the repetition value has no similar checks and using it, especially at binnings from 2 up, is dangerous as it can result in frequencies of 3Khz or more. Always use the frequency setting and not the repetitions settings, unless you know what you are doing.
- **Black levels:** the four (ccd39) or two (ccd47) quadrants each have their independent bias level, called "black level". A higher black level corresponds to a lower pixel value for the same illumination level. A black level "unit" corresponds to 20 counts, or about 10 photons.
- **"Equalize quadrants" button**: starts an automatic procedure to adjust the black levels so that every ccd quadrant has an average level of 200 counts (prior to background subtraction). The procedure takes a few seconds to converge. This button should be used when ambient light, articificial illumination and internal wfs lamp are off, otherwise an incorrect level will be reached. In addition, if the artificial illumination is flickering at 120 Hz, the procedure will not converge since the ccd is seeing a variable amount of light, and it will giveup after a while.
- **Temperature display**: three temperatures are given: "case" is a sensor inside the electronics case of the little joe controller. "Head" are two sensors on the ccd chip itself. The first sensor has a lower limit of 19°C, the other two of -40 °C. Hence, during winter observation they are often pegged to the lower limit.
- **Background:** shows the current dark frame loaded on the BCU, and allows the operator to select another file and send it to the BCU.
- **"Start"/"Stop" button**: stars and stops ccd integration. Integration is started by default after a binning is applied.
- **"Live view" button**: starts the ccd viewer described in []
- **"Save" button:** starts an interface to save frames from the ccd, described in [].

8.2.3. CCD47

This window is identical to the one for the ccd39, except for a few differences:

- the ccd47 has only 2 quadrants, so only 2 black levels are available
- available binings are 1,2,4 and 16.
- frame rates are much lower and the repetition setting changes little. Frame rate is essentially fixed by the chosen binning.

8.2.4. Filter wheel #1



The filterwheel panel shows the current filter wheel status and position. The position is not valid until the motor has been turned on and homed, and the GUI shows this graying out the position display.

The movement is in “filters” unit (1.0 correspond to the angle between two filters) and can be either absolute or relative:

- absolute movement is an offset from the home position.
- Relative movement is an offset from the current position

Position can be decimal, for example a movement to 0.5 will position the wheel halfway between two filters.

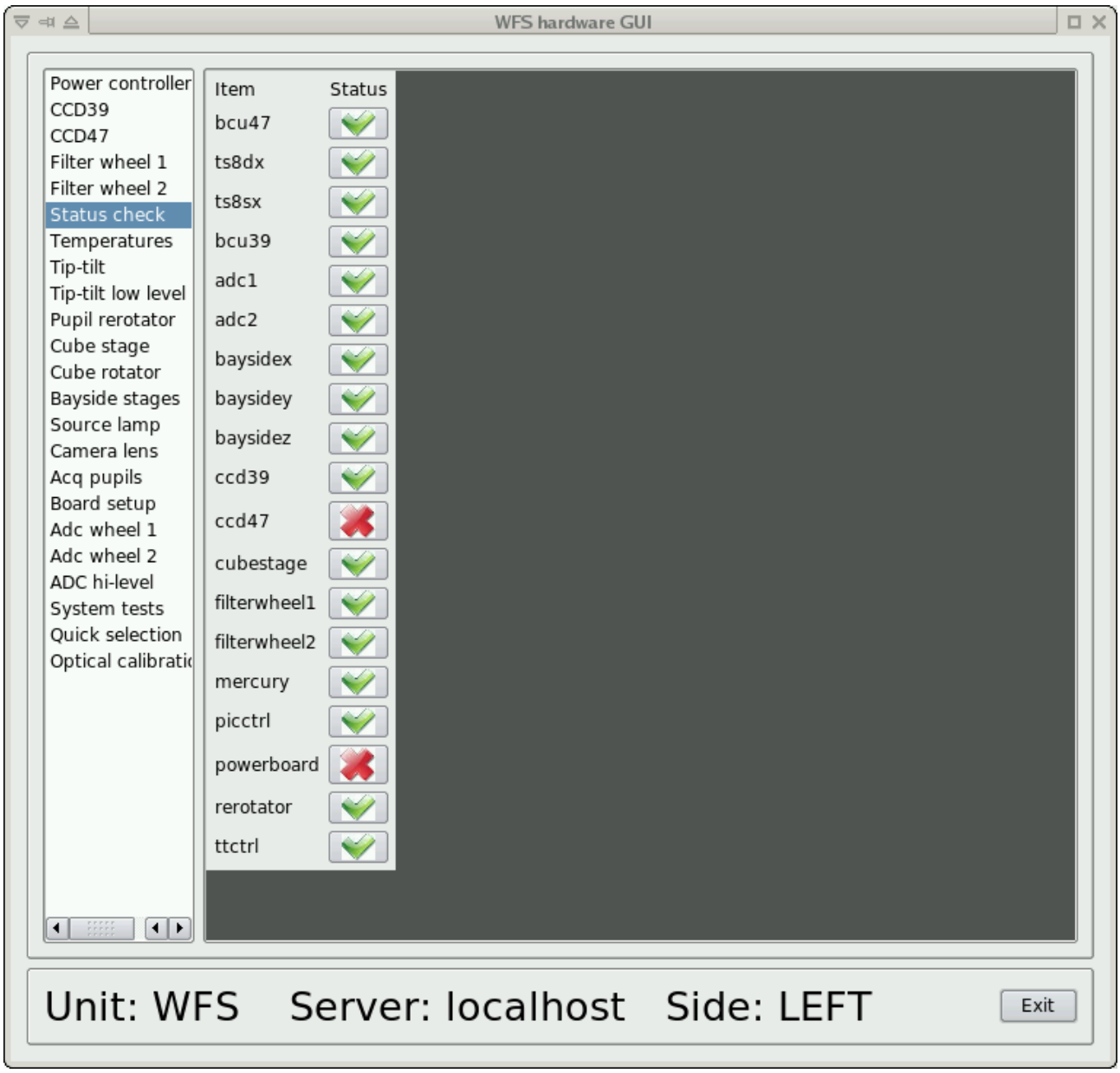
The custom position buttons are dynamic and are built from the custom positions defined in the filter wheel's configuration file. Clicking on those buttons will immediately move the filter wheel to the custom position. When the filterwheel is on a position defined as custom position, the name will be displayed along the numeric position (as in the screenshot above).

- “Abort movement” will stop any current movement of the filterwheel
- “Start homing” will start the home position search procedure. This is automatically triggered when the filterwheel is powered up.

8.2.5. Filter wheel #2

This panel is functionally identical to the Filter wheel #1 panel.

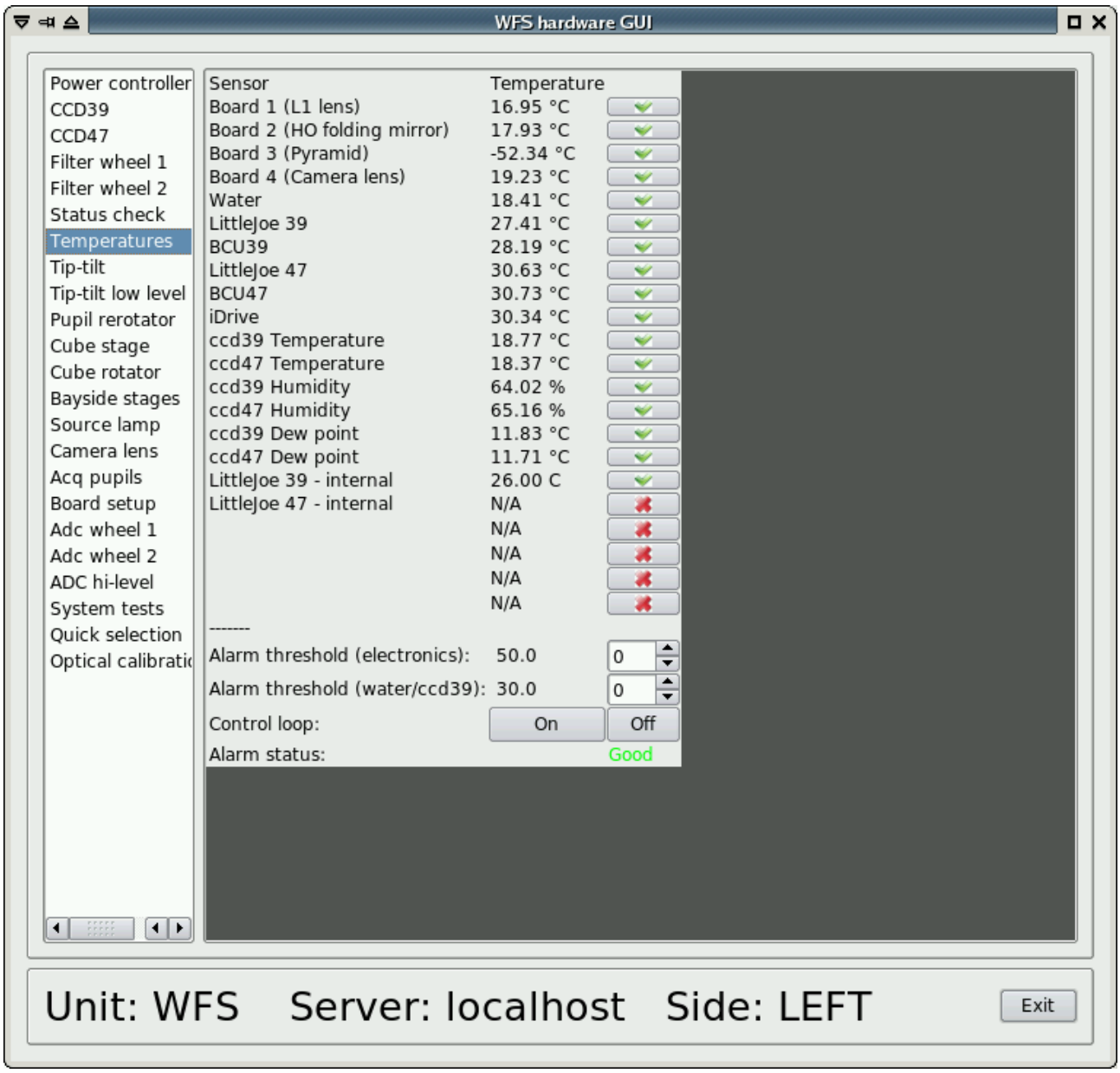
8.2.6. Status check



This panel shows the status of all wfs devices. Status are either red or green. A green status means that both the software and the hardware for the specified devices is ready. In normal wfs operation, everything is green

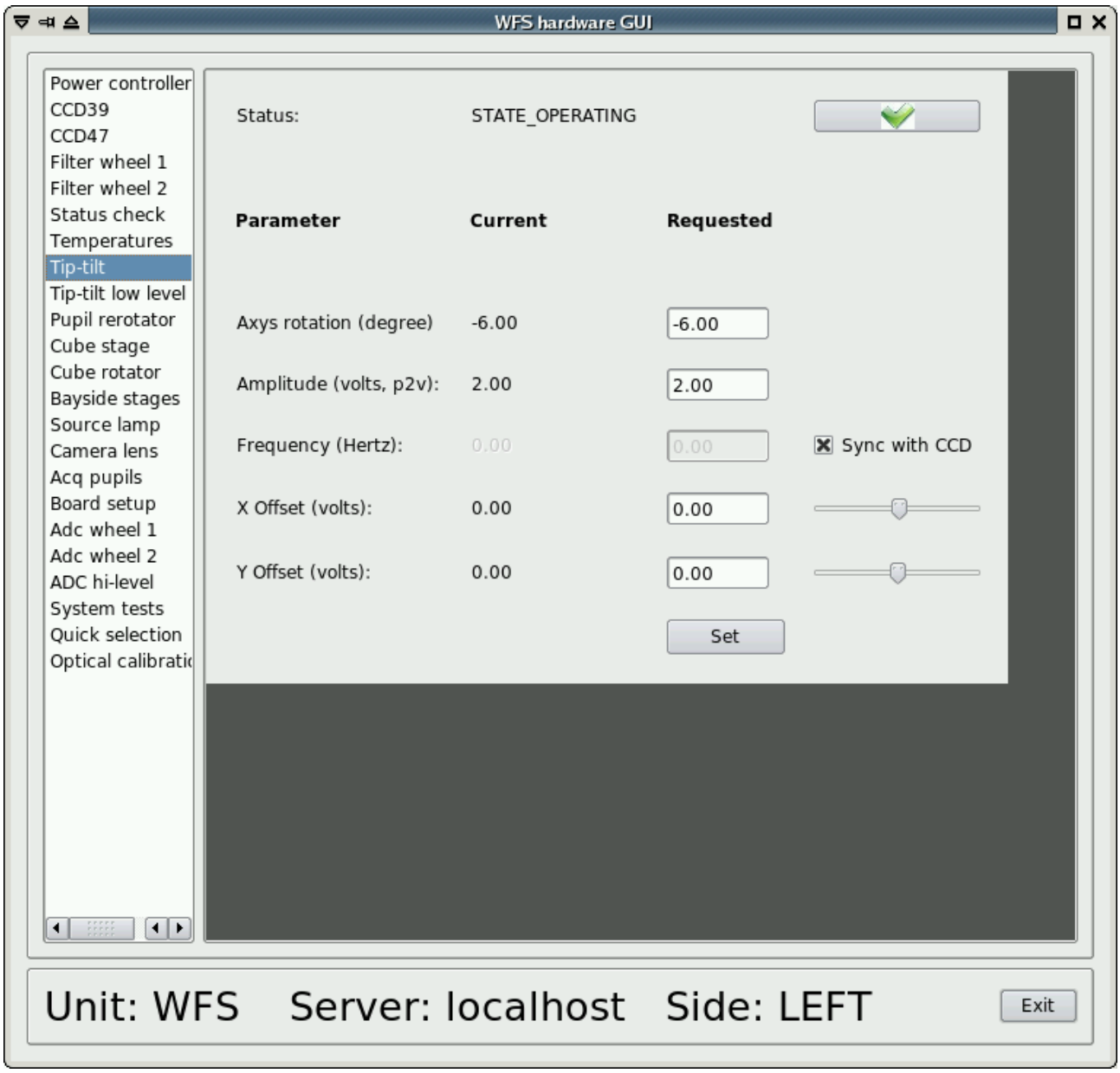
If the power on configuration was incomplete (as in this example, where the configuration was excluding the ccd47), it is normal that the excluded devices appear in red.

8.2.7. Temperatures



Shows all the temperature readings collected from various parts of the wfs. Update rates varies from once every few seconds to once every 30 seconds depending on the sensor. If a sensor does not answer for more than a minute, “N/A” (not available) is shown. Sensors can also be not available if the corresponding devices are off or otherwise unreachable.
The lower section controls the over temperature protection system: the PIC-based control board will automatically shut the WFS off if any of the sensors goes over the specified thresholds. Two different thresholds are available: one for the sensors on the electronics, and one for the water intake and CCD temperature.

8.2.8. Tip-tilt

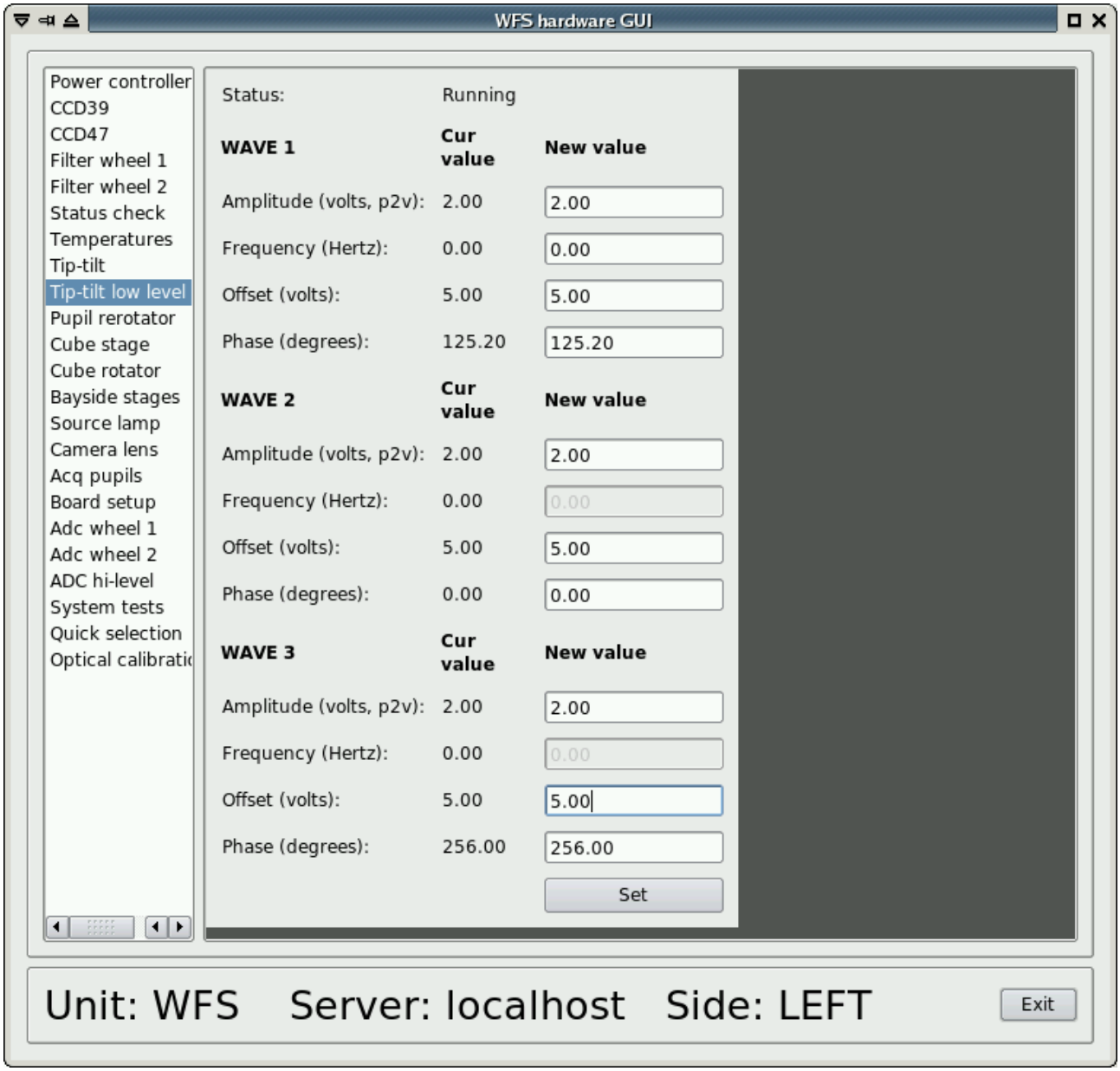


The tip-tilt panel shows both the current values and the input boxes for the requested values. All requested values are applied when the “Set” button is pressed (empty values default to zero). The tip-tilt is a three-axis device, but the control is done over two virtual axes that are remapped on the three real axis.

- axis rotation: controls the orientation of the XY reference system
- amplitude: controls the overall modulation amplitude, specified in volts (0-10V)
- Frequency: controls the modulation frequency. If the “Sync with ccd” checkbox is set, this value will be ignored (a zero is sent to the BCU), and the frequency will be synced to ccd frame rate using the tip-tilt fiber.
  Note: it is not possible to sync with the ccd frame rate when the ccd is turned off or otherwise unavailable. This condition is detected and the checkbox will be unavailable in this case.
- X and Y offsets: change the modulation center, from -5 to +5V, with zero being the nominal range center. Either the input boxes or the sliders can be used. Slider changes are applied immediately without pressing the “Set” button.
  Note: When applying X and Y offsets, the tip-tilt controller may reduce the modulation amplitude so that the maximum voltage applied does not exceed 10 V, as sum of offset plus

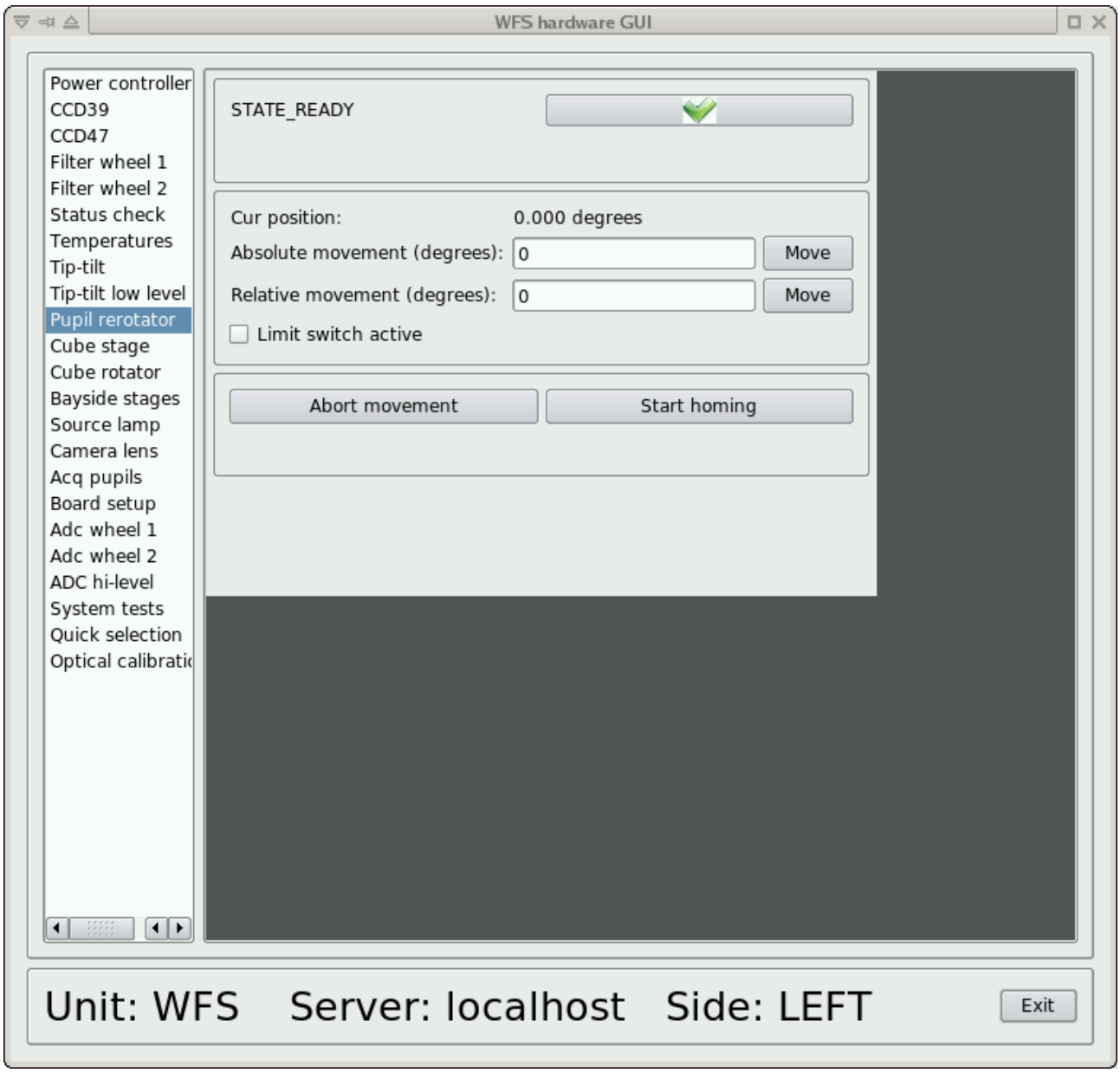
semi-amplitude. This is shown in the GUI as a reduction of the current amplitude with respect to the requested value.

8.2.9. Tip-tilt low level



This panel allows individual control of the three tip-tilt axes. For each axis, amplitude, offset and phase can be set independently, while the frequency is locked to be the same. The nominal offset center here is 5 volts, and ranges from 0 to 10V.

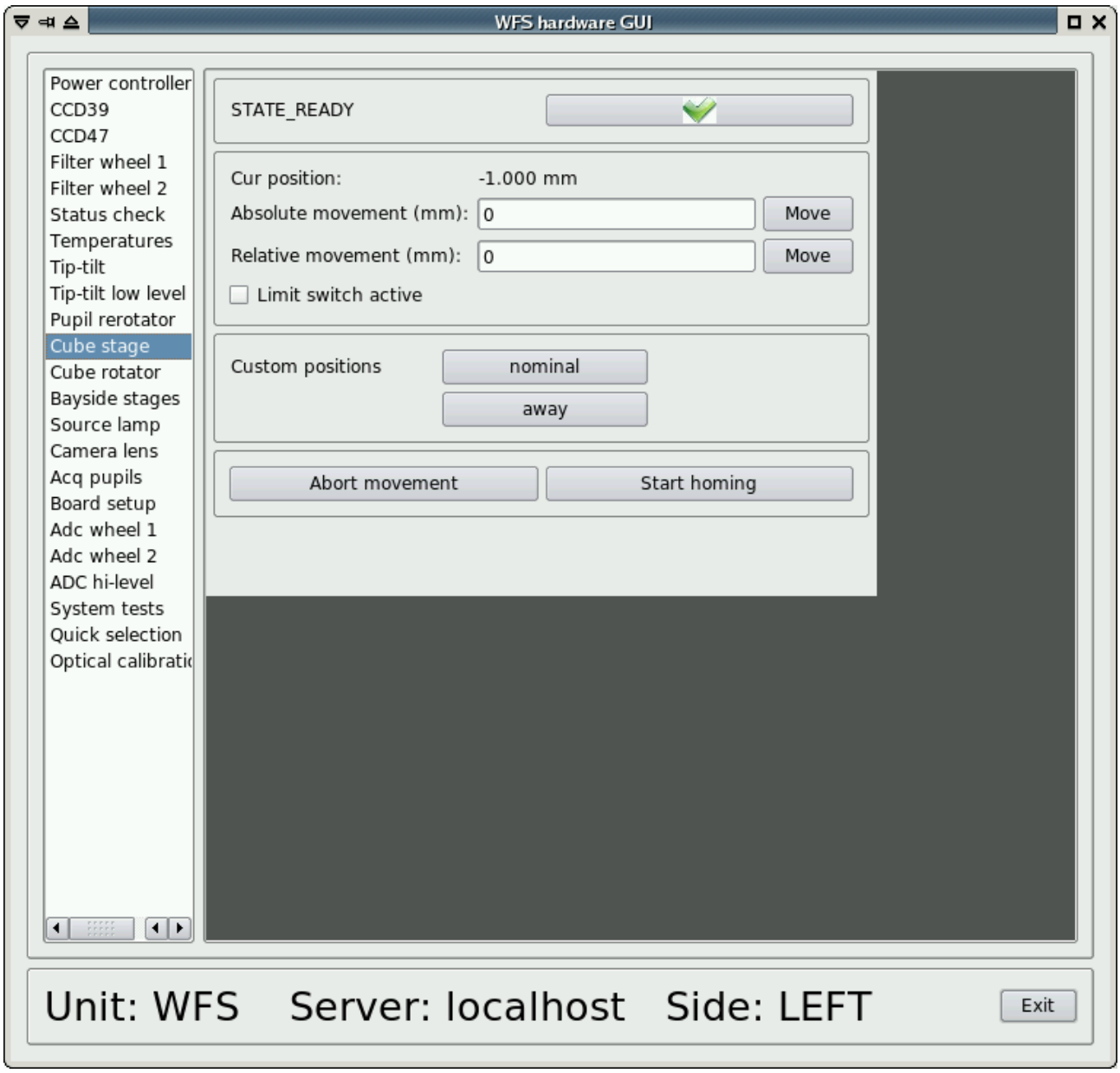
8.2.10. Pupil rerotator



This panel commands the pupil rerotator and is functionally similar to the one of the filter wheel #1, except that the movement unit is in degrees. Also, the pupil rerotator has a limit switch that will prevent movement lower than zero degrees, or any other backward movement crossing a 360° threshold. So, once the position has advanced over 360°, it will not go lower than that until the movement is power cycled.

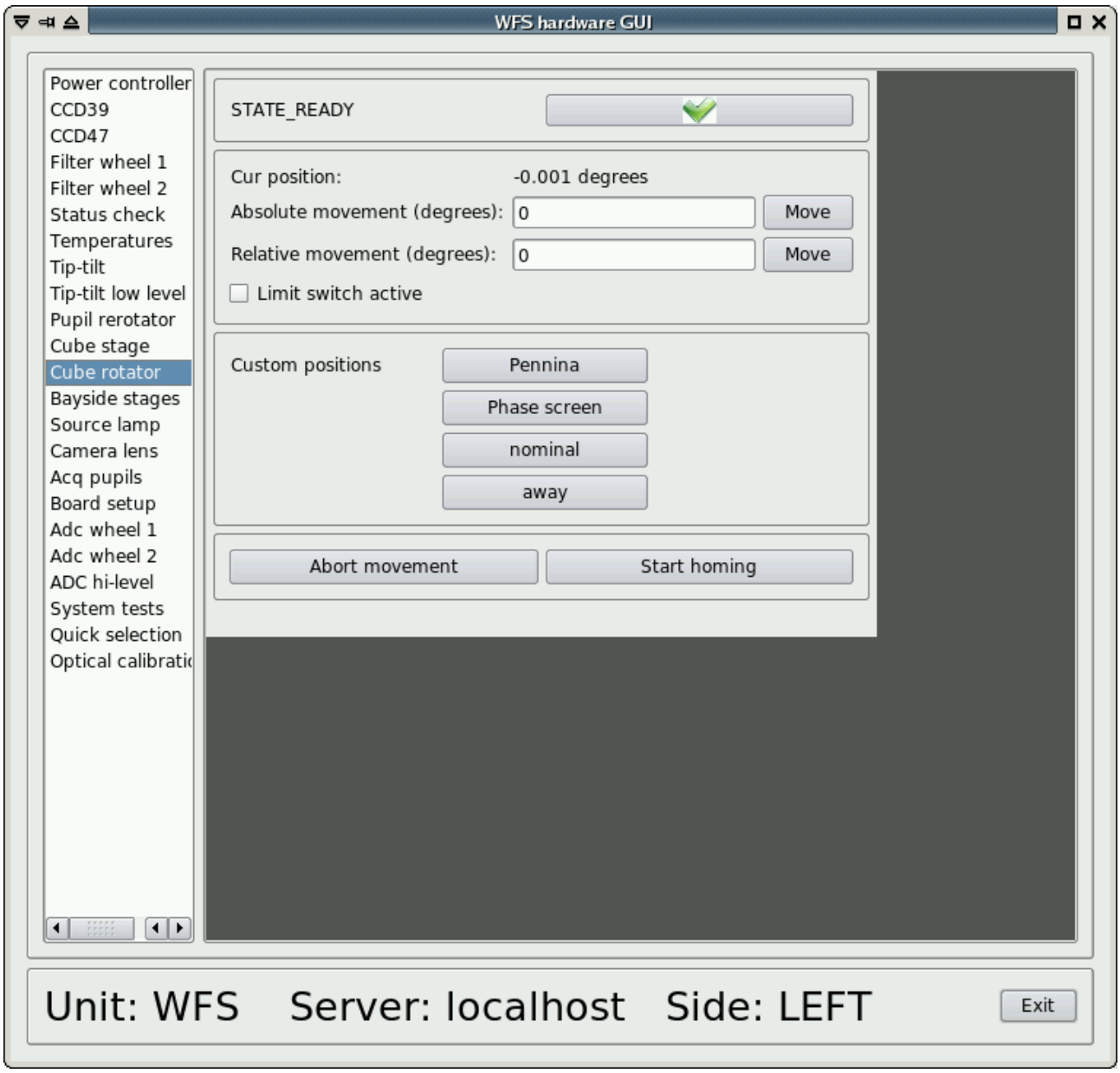
When using the AOS or WfsArbitrator, this condition is automatically avoided. Furthermore, the rotator tracking is on at virtually all times, and it will continuously send commands, rendering this panel basically useless until the tracking is stopped.

8.2.11. Cube stage



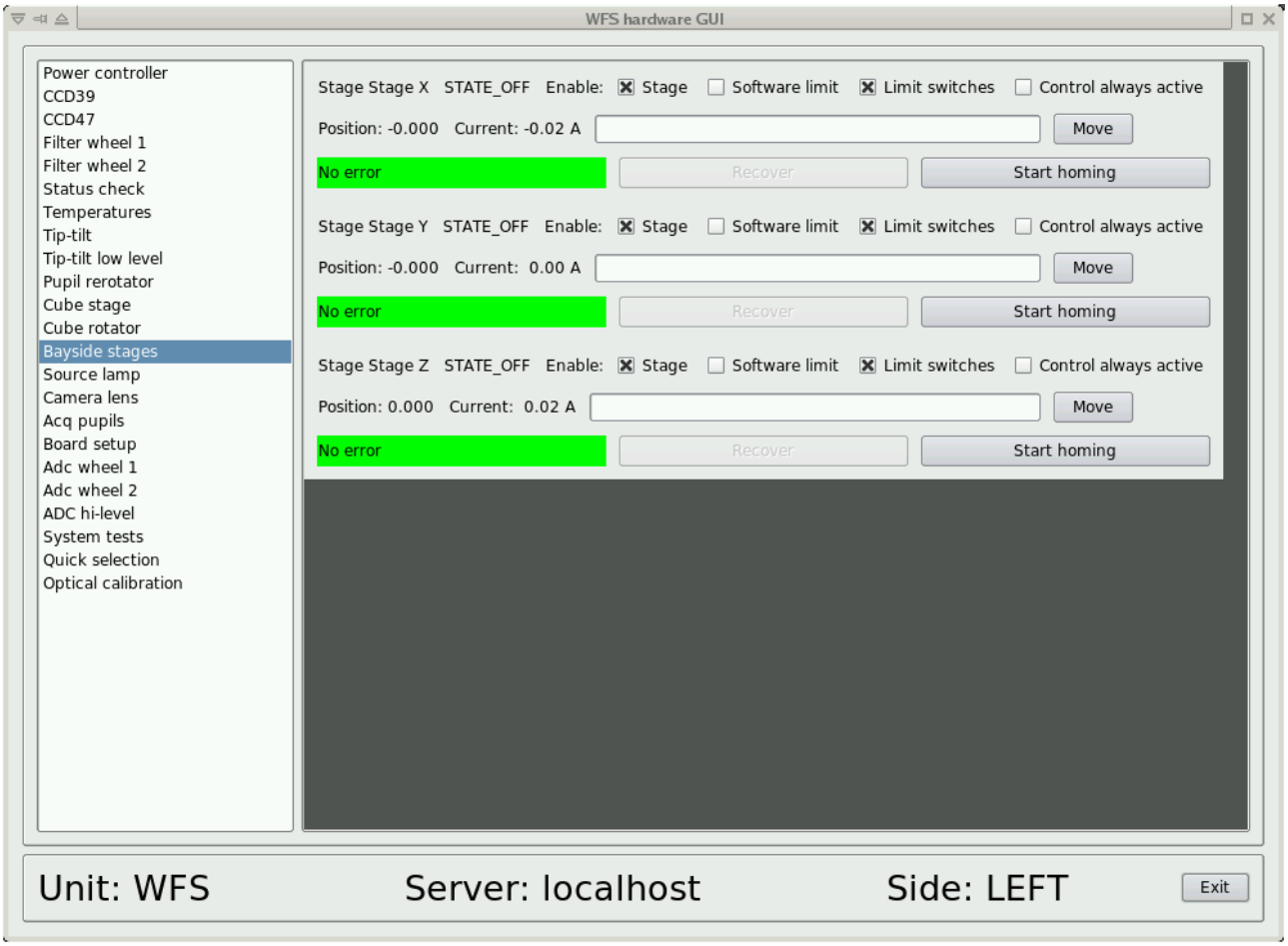
This panel commands the cube stage, which is a linear movement with two limit switches. Units is in millimeters. Apart from this, it is functionally identical to the filter wheel #1 panel.

8.2.12. Cube rotator



This panel commands the cube rotator, which is a rotary movement without limitations on movement. Unit is in degrees and precision of movement is on the order of 0.01 degrees. The panel is functionally identical to the filter wheel #1 panel.

8.2.13. Bayside stages



8.2.13.1. Displays

The bayside stage panel shows the status of each of the three XYZ stages. For each stage a status is reported:

- NOCONNECTION: stage is off or otherwise unreachable
- CONNECTED: stage is reachable, connection in progress
- OFF: stage is ready, motor disabled, brake set
- OPERATING: stage is moving and/or actively mantaining a position

Position display: the current position is shown, as an offset from the homing position. If the homing procedure has not been performed, the position where the stage was when it was turned on is assumed as zero. The stage positions use the following reference system:

[diagram]

Note that the X stage has a home position towards the left movement limit, and thus valid positions range from 0 down to -120 mm. Y and Z stages instead move from the home position up to about +88 and +70 mm respectively.

Current display: the current adsorbed by the motor is shown in Ampere. An adsorption of 1 or 2 amperes is normal. If the display is stuck at 4 amperes, it means that some mechanical obstruction is present and the current limiter is in action.

8.2.13.2. Controls

For each stage, the following checkboxes are available:

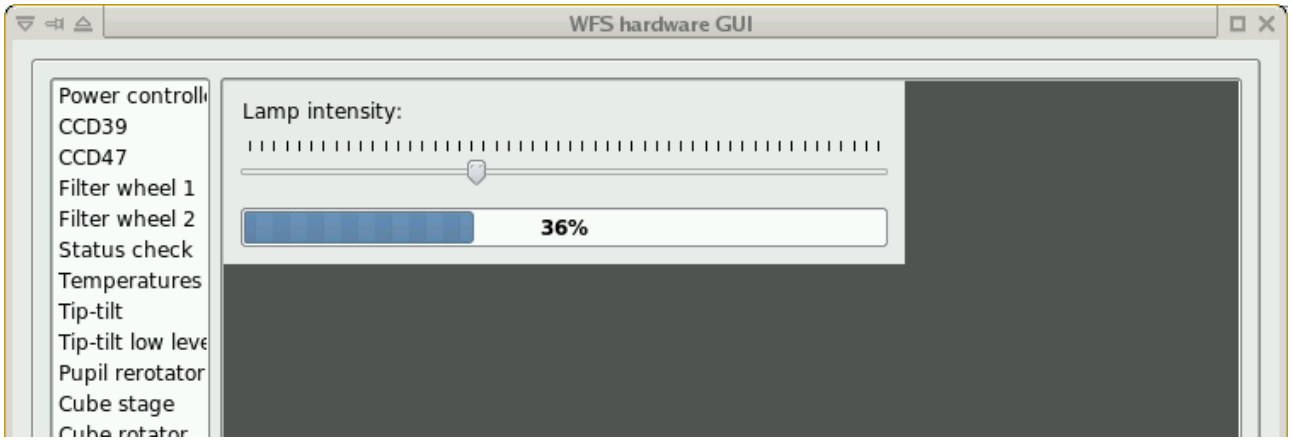
- **“enable stage”** this must be checked to allow the stage to move. Unchecking this checkbox will cause the stage to stop immediately and brake in the current position
- **“software limit”** enforces the sofware movement limits, defined in the stage configuration files.
- **“limit switches”** enables the electrical limit switches that prevent the stages from reaching the mechanical movement limits. This is usually always enabled.
- **“control always active”** avoids braking the stage once a movement is done, allowing the motor to actively keep the position. If the checkbox is activated before a movement, the setting will be applied on the next movement. If unchecked while the motor is maintaining a position, it will cause the stage to set the brake and stop there.

Other controls:

“Move” button: the move button will move the stage to the position specified in the main inputbox. Only absolute positioning (offset from the home position) is available.

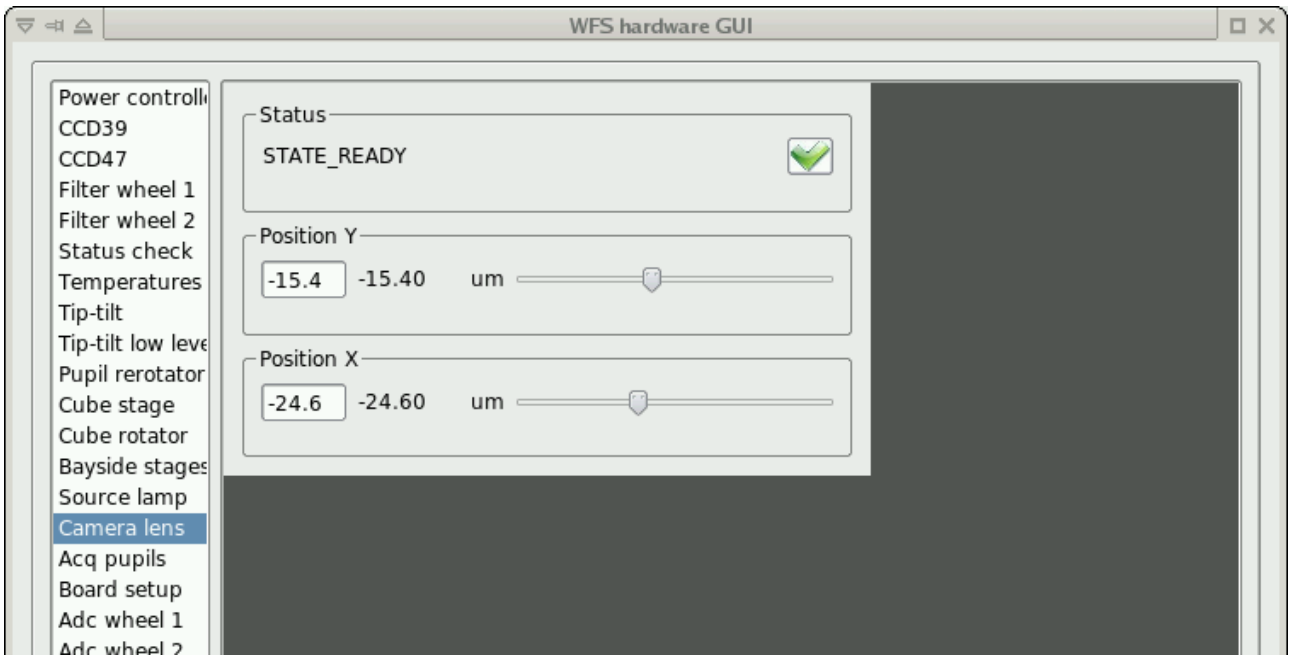
“Start homing”: starts the homing procedure. X stage homes towards the left limit (positive coordinates), while Y and Z stages home toward the lower and backward limit (negative coordinates).

8.2.14. Source lamp



This panel allows the operator to change the reference lamp intensity moving the slider. This is in addition to the lamp on/off control described in []. The actual lamp intensity is displayed as a percentage and will take a few seconds to follow the slider due to delays in the PIC-based flowerpot controller.
Note: the lamp behavior is highly non-linear. At the lowest settings, the lamp is virtually off, then ramps up quickly. Over 50% or so the lamp increases luminosity only marginally.

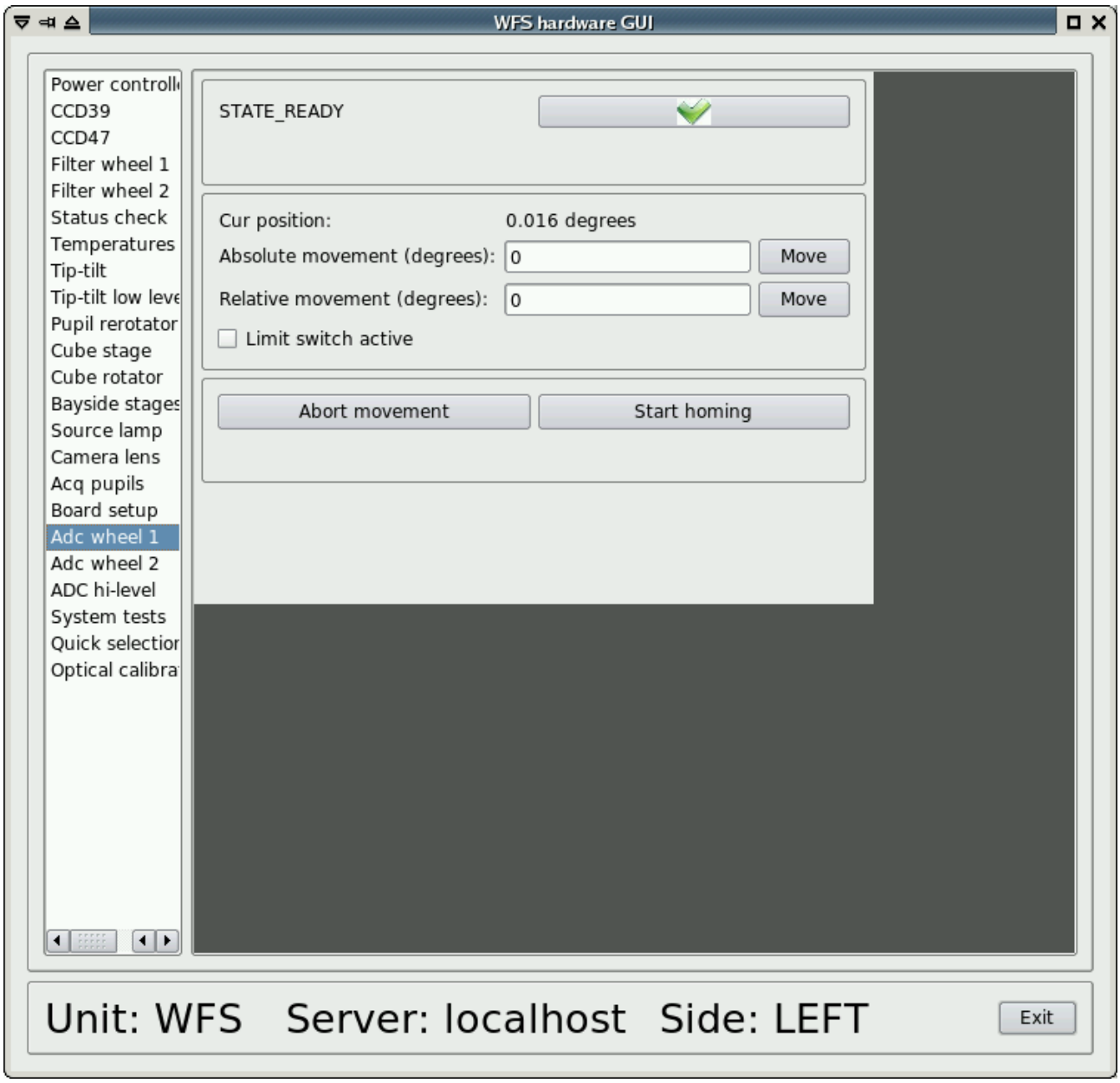
8.2.15. Camera lens



This panel controls the two axis of the cameralens. The cameralens outputs of the BCU are slaved to the ones for the tip-tilt: if the tip-tilt has not been configured, the outputs are not enabled and the cameralens is in the default rest position. Before using the cameralens, the operator must first set the tip-tilt, even giving zero as amplitude and offset.

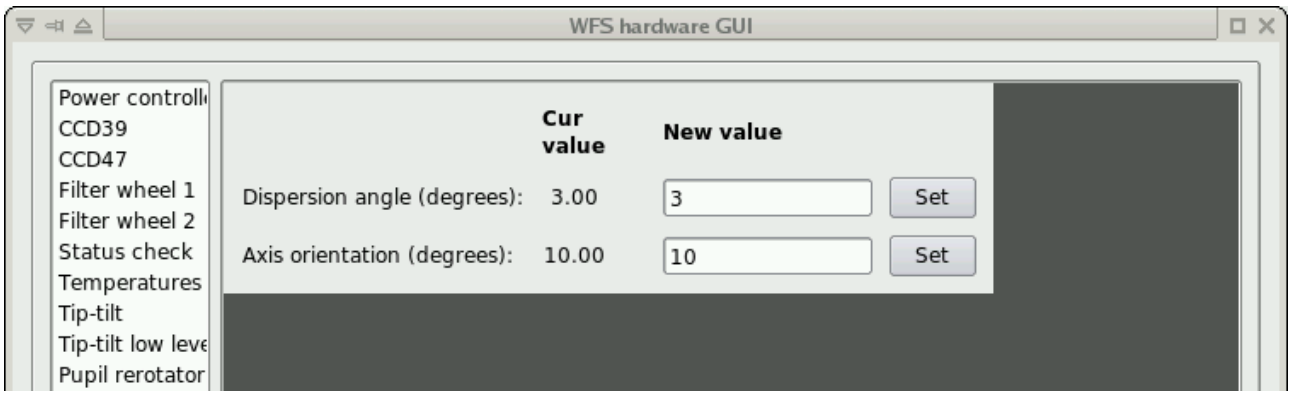
Once the outputs are active, the two sliders control the X and Y camera lens position. The current position is shown, and the position can also be changed writing the new values in the two input boxes and pressing Enter. This action will set both axes at the same time.

8.2.16. ADC wheels #1 and #2



These two panels control the two ADC prism wheels. The unit of movement is in degrees. Apart from this, they are functionally identical to the filter wheel #1 panel.
When using the Wfs Arbitrator ADC tracking, the ADC position is continuously updated and this panel cannot be used.

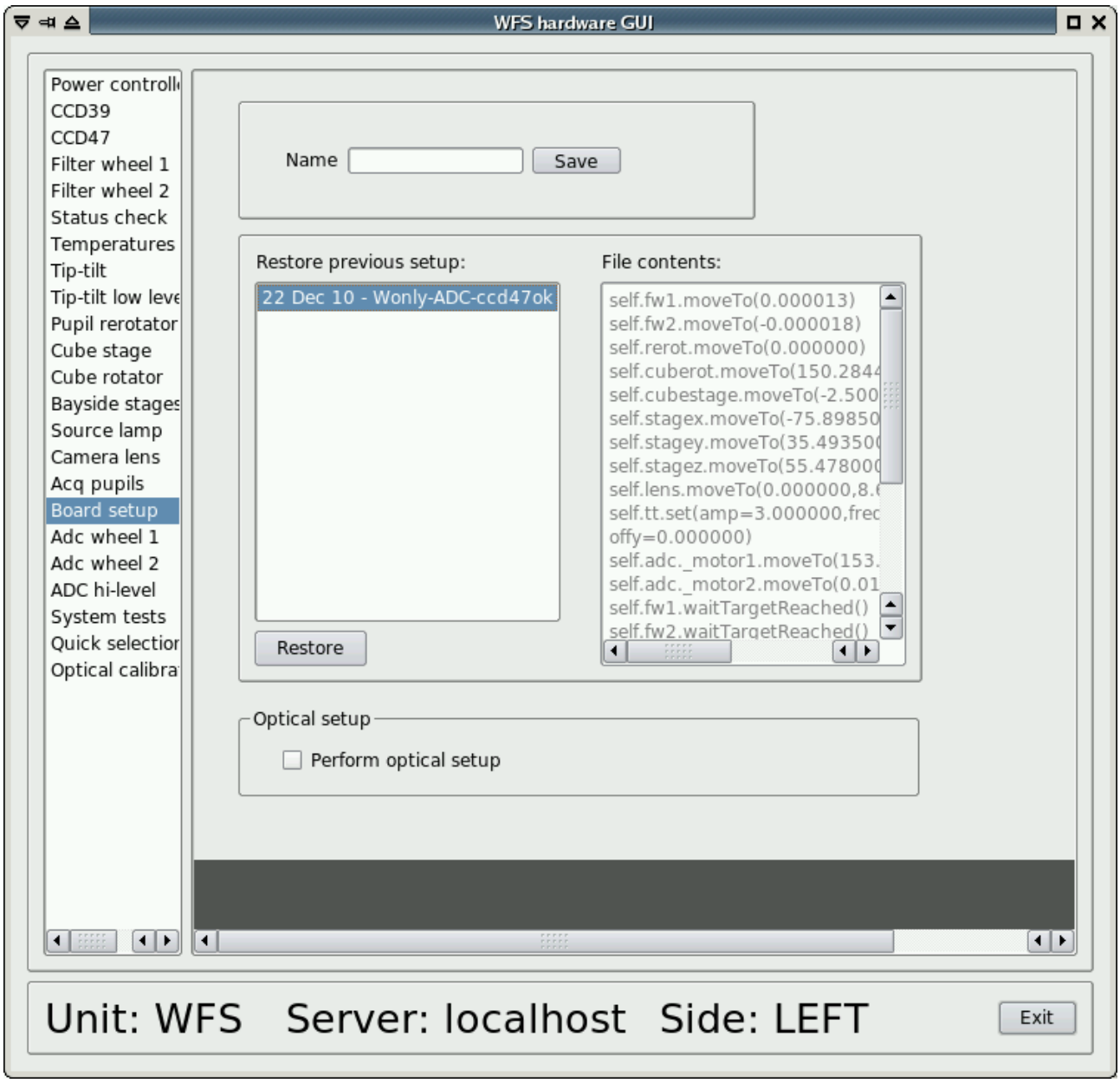
8.2.17. ADC high-level



This panel allows the control of the ADC as a function of two high-level parameters: dispersion angle and axis orientation. There is a direct mapping of these two parameters to the ADC wheels position.

When either of the Set buttons is pressed, both values are sent to the ADC controllers.
When using the Wfs Arbitrator ADC tracking, the ADC position is continously updated and this panel cannot be used.

8.2.18. Board setup



The operator can save the current board configuration using this panel to save a “board setup” file. A board setup is a text file containing commands to set the wfs movements positions. To save the current configuration, enter a descriptive name in the top input box and press “save”. A text file will be generated and saved in a predefined directory (see []), and shown in the list with the date on which it was saved.
Clicking on a filename on the list will display its contents in the right box for inspection. To load a board setup file, select it on the list and press the “Restore” button. An xterm will appear where the script will execute.
Warning: if some devices are off, they will not respond to commands, and the xterm will wait for their response with a timeout which may be quite long. It is safe to simply close the xterm, because all commands are sent in less than one second, and the rest is only waiting for the devices to report their status.

The following parameters are saved/restored:

- filter wheels #1 and #2 positions
- cube stage and rotator positions

- X, Y and Z stages positions
- Camera lens X and Y positions
- Tip tilt modulation amplitude, frequency and offsets
- ADC wheels #1 and #2 positions

8.2.19. System tests

[screenshot]

This panel contains shortcut buttons to start various system tests scripts. Their output, when available, is shown in the box below the buttons. The output text can be copied and pasted. The available scripts are:

- ccd39 RON test: starts the ccd39 RON (ReadOut Noise) test. It will cycle the ccd between the four available readout speeds, and report the results in the box below.

8.2.20. Quick selection

[screenshot]

This panel contains a list of the available calibration files for the BCU (background frames and slopenull frames), displays the currently selected file and allows the operator to select a different file and send it to the BCU. Loading the file is immediate whether in open or closed loop. While operational, use of the WfsArbitrator GUI for the background is recommended.

9. AdSec operation

9.1. Safety Remarks

The system is particularly sensitive to condensation problems: condensation inside the gap between the Reference Body and the Thin Shell not only does not allow operating the system, but also requires long time to be fixed.

For this reason the system is **maintained fully powered** up also during day and night time. If a switch off of the system is mandatory, the dew-point level has to be taken carefully in account.

Moreover, if the **wind speed** is over 8 [m/s], for thin shell safety reasons, only the system can be powered off **but not the AO software** framework.

In order for the mirror to work for observation the telescope **elevation** has to be **greater than 25 degrees**, the **swing arm** has to be **deployed** and the **wind speed** at secondary level has to be **below 22 m/s**. All these info has to be available from the TCS in order to permit to the Adaptive secondary to properly working.

For safety reasons, if the **wind speed** becomes **unavailable**, the **TSS** safety extra current is **enabled**.

In the same way, if the **elevation** or the **swing arm status** becomes **unavailable**, **the shell is rest** against the reference body and **any operation will be stopped**.

Please take care about the AOS process: since the AOS is routing the information from the TCS to the AOSupervisor , **any shutdown of the AOS will cause the TSS to be enabled and the shell to be rest**.

In the same way a mis-functioning of the TCS processes in charge of collecting the wind speed data, the elevation and the swing arm status will cause an AOSupervisor reaction as described above.

9.2. Quick start with AOS GUI (from BP4 built ahead)

Open the AOSGUI. If the GUI is RED, the upper right label is not operating or if the Adaptive Secondary fields marked not looks like the ones here reported please refer to the chapter 9.5

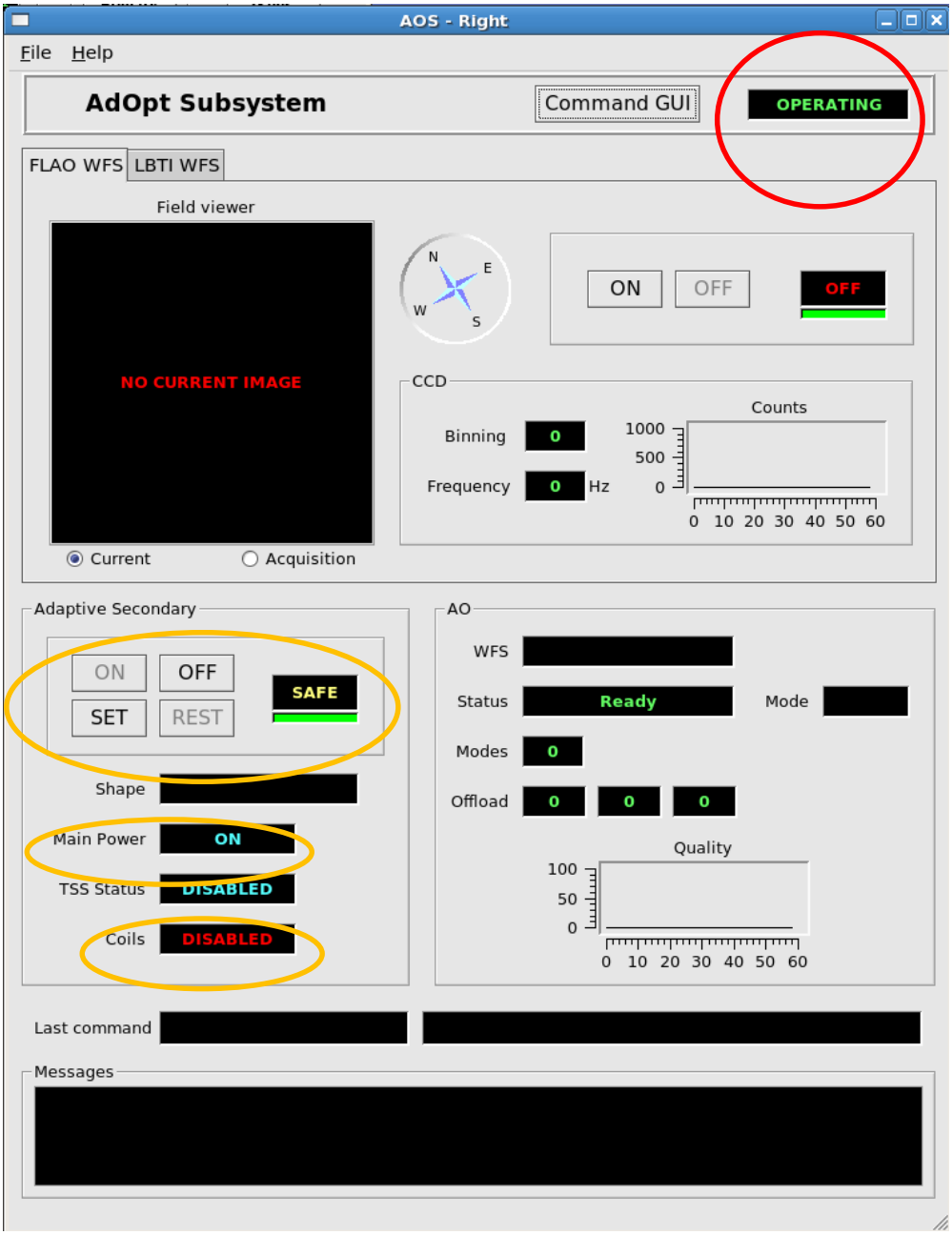
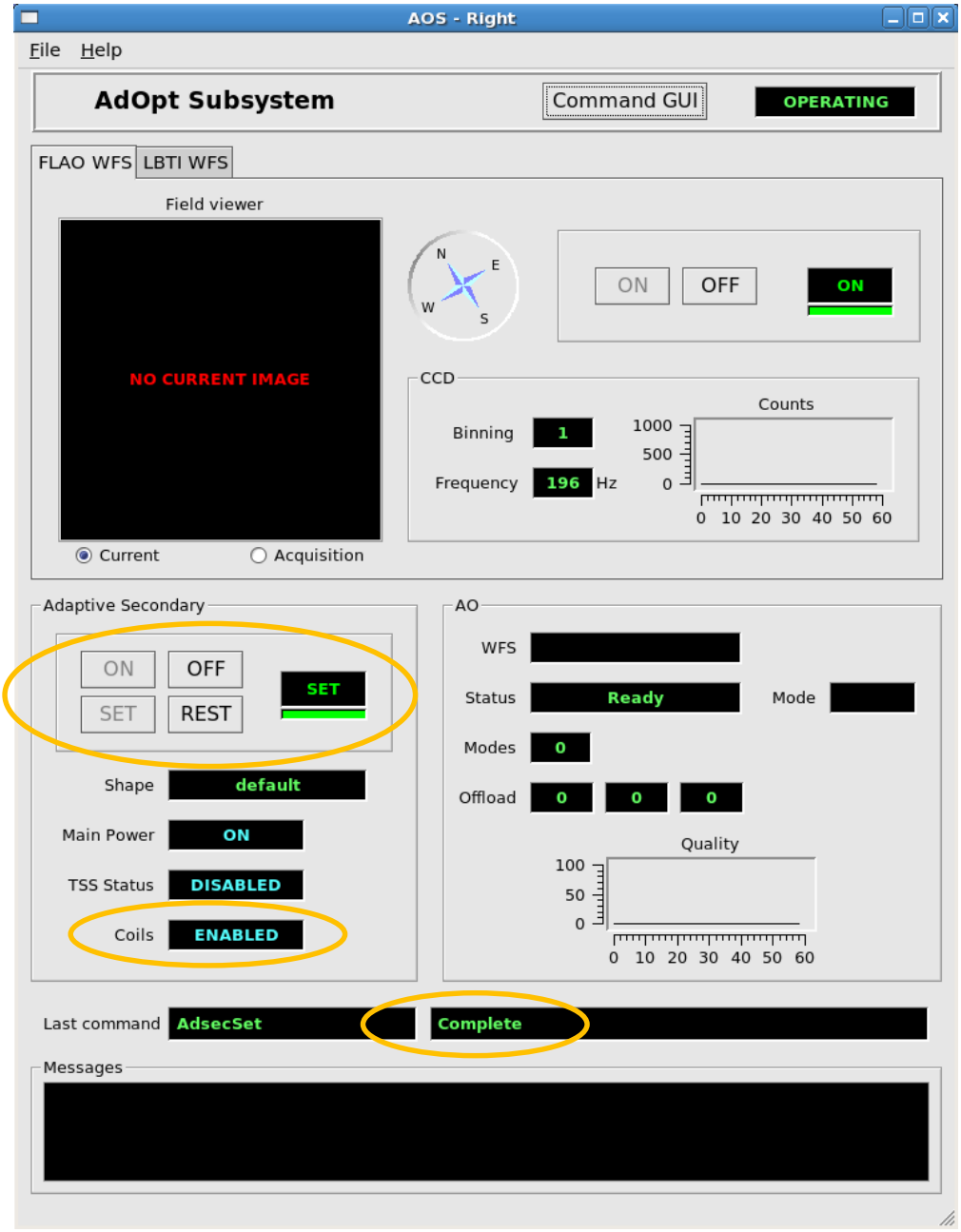


Figure: 9-1 AOS GUI

Press the **SET** button and wait. After a couple of minutes, if everything is ok, in the Adaptive Secondary frame, the label should change between ***SAFE*** to ***SET***.



At the end of the observation night, please put the mirror in the SAFE position BEFORE going horizon to close the dome pressing **REST**. If you REALLY want to power the system off you can press, after the REST, the OFF button. (About that read carefully the chapter 9.1).

In case of mirror FAILURE please use the engineering GUI as shown below.

9.3. Quick start with Engineering GUI

Start the adsceng panel (See chapter 5.1):

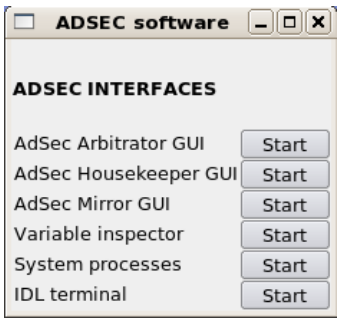


Figure: 9-2 Adsec Engineering GUI panel

Press for starting the **AdSec Arbitrator GUI**.

Note: the AdSec Arbitrator GUI can also be started from a terminal with this command:

```
AdSecControl
```

Once the GUI opens, check the fields circled in RED in the following screenshot.

If the fields marked in RED are not showing values OR have a RED background please check that the AOS is up and running. If that values are not correctly updated the Adaptive Secondary cannot be operated. Check the "AdSec Arbitrator Status" label (circled in orange in the following screenshot). It must show a **Ready** state. If not, refer to chapter 7 to set the proper arbitrator status.

To set the shell, press the "**SetFlatAo**" button (on the left column, circled in orange in the following screenshot). The command will take about 2 minutes to execute. At the end, the status will be "**AOSet**".

At this point, the secondary mirror is flat and ready for seeing limited operation and no further action is necessary.

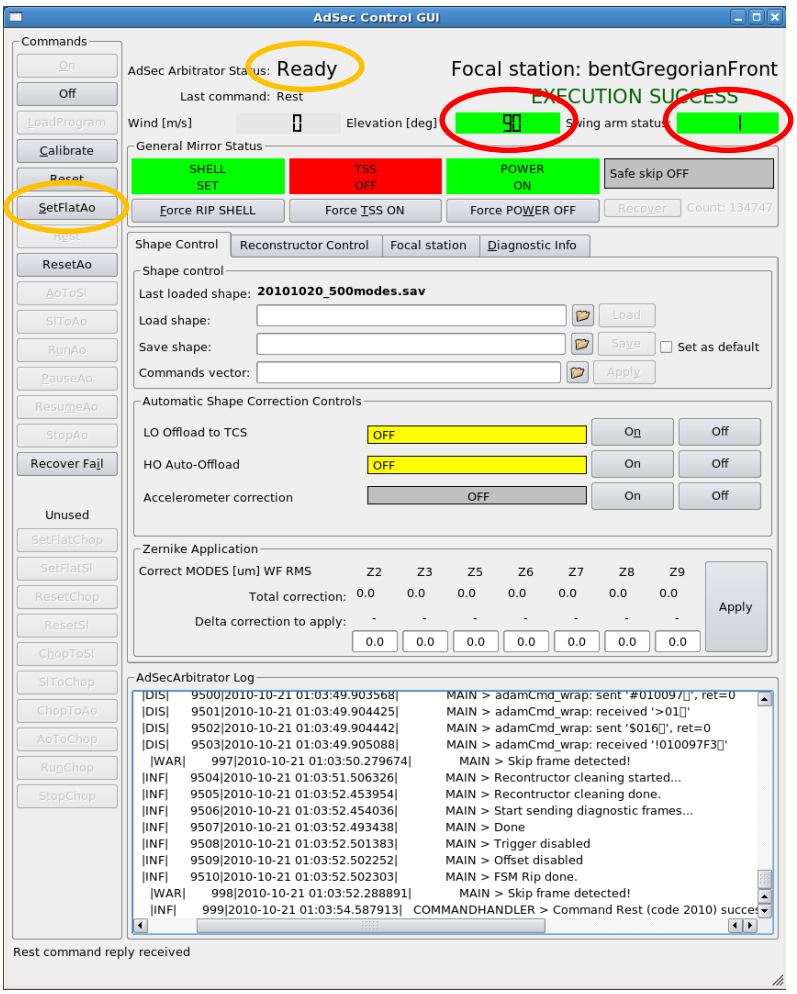
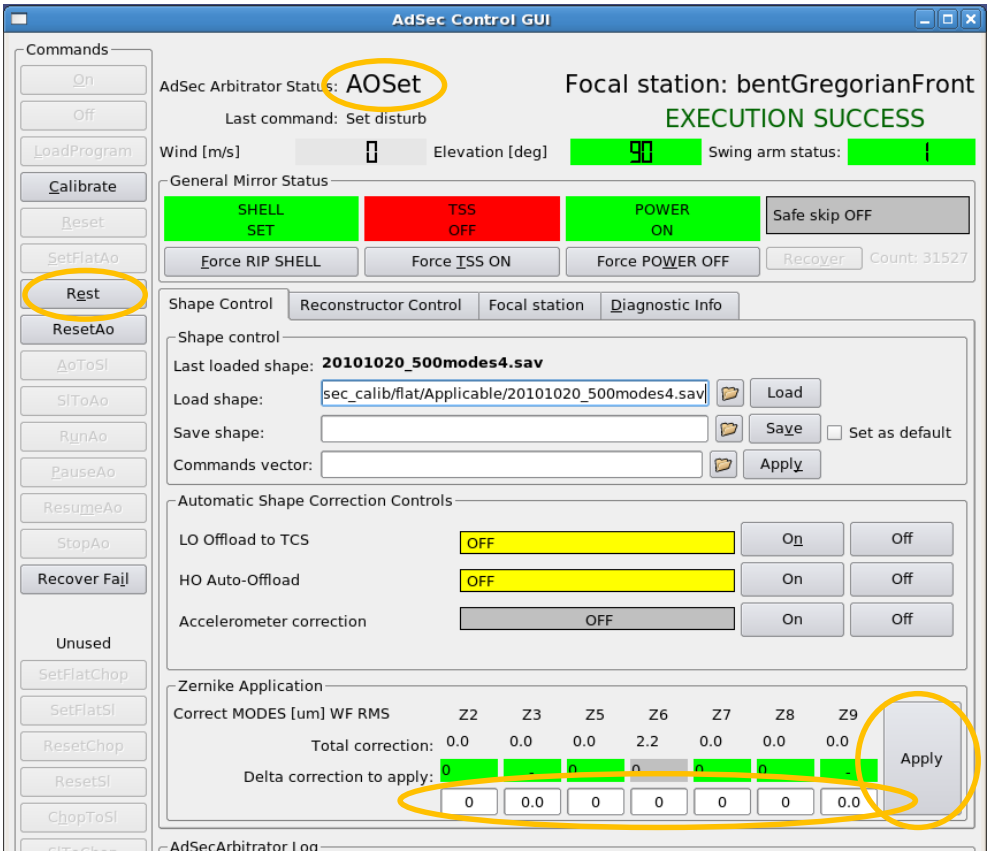


Figure: 9-3 Adsec Arbitrator GUI



Once the shell is set, you can apply low order Zernike correction. **Write** a value (um wavefront), press **Enter** and once the field is updated press **Apply**.

At the end of the observation, put the thin shell in safe position with pressing the **Rest** button.

The Adsec Arbitrator status will change **from AOSet to Ready.**

9.4. Status indicators

At the top of the AdSec Control GUI, the following status information is shown:

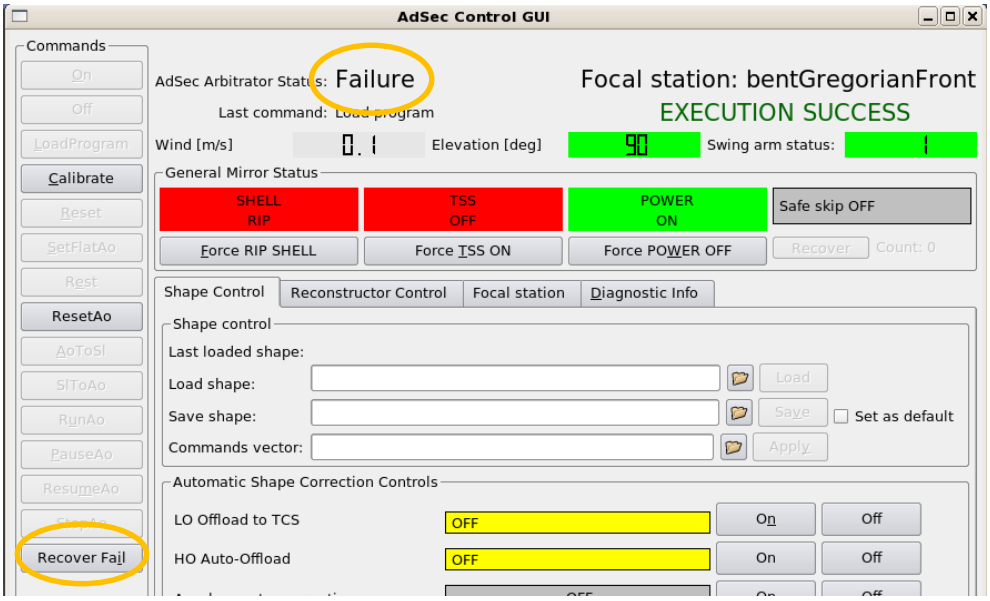
- AdSec Arbitrator Status: tells the operator in which state the AdSec is at the moment, and therefore which commands are available. Also, in the GUI commands that are not available at the moment are grayed out.
  Note: if the AdSec Arbitrator program is not running, or not correctly responding, the Status will be "offline" and no commands will be executed.
- Last executed commands: shows the name of the last command executed by the AdSec Arbitrator
- Command execution status: shows whether a command is executing at the moment, or the result of the last command. When a command is executing, all GUI buttons are inactive.
- Focal station: shows the currently selected focal station (which may be "None" if no focal stations have been selected since the last adsec startup). The selected focal station is the only one allowed to send slopes to the secondary mirror.
- Lab mode: the red label "lab mode enabled" is shown when the arbitrator is in the so-called "lab mode". In this mode, some safety features are disabled. Intentionally, it is not possible to enable the lab mode using the GUI.
- Telescope data (wind, elevation, swing arm status).
- Safe skip indicator. Becomed RED when the AO loop is skipping more than 10% of the frames.

9.5. Quick recovery from Failure

The Adaptive Secondary arbitrator can reach a **Failure** state in case of some hardware or software fault occurred. The software will try to **recover automatically** from the failure and go back to Ready state. In case this is not working, the operator has to press the "**RecoverFail**" button on the AdSec Arbitrator GUI to recover the system.

The fail recovery procedure takes about one minute to complete.

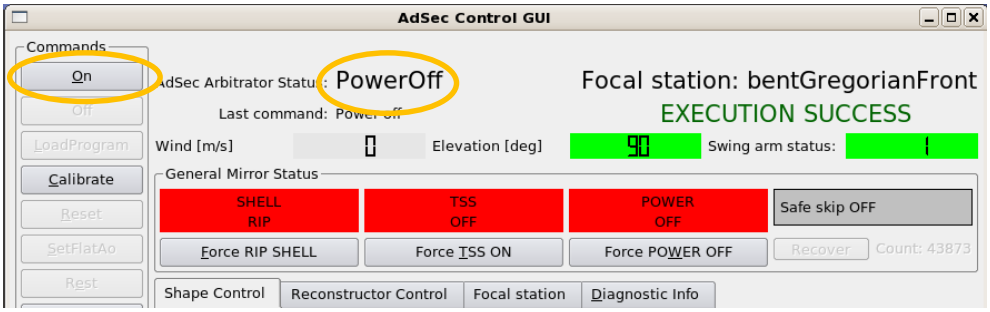
In case the Recover Fail procedure does not work, try first to restart the software from scratch. If it is still impossible to start the mirror, you will need technical assistance from Arcetri.



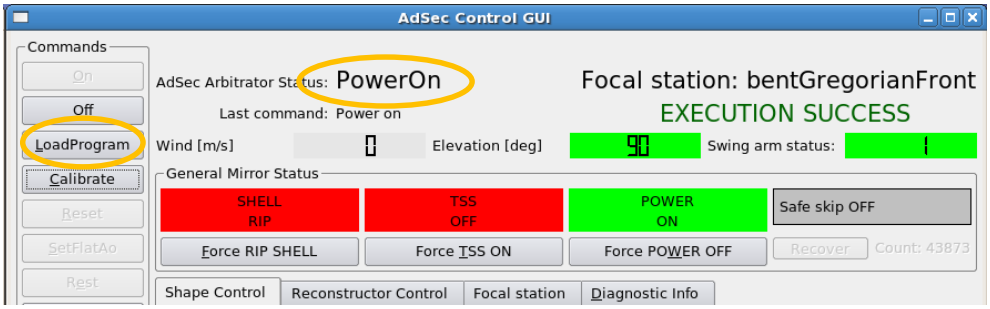
9.6. Adaptive Secondary startup and shutdown

9.6.1. With Engineering GUIs

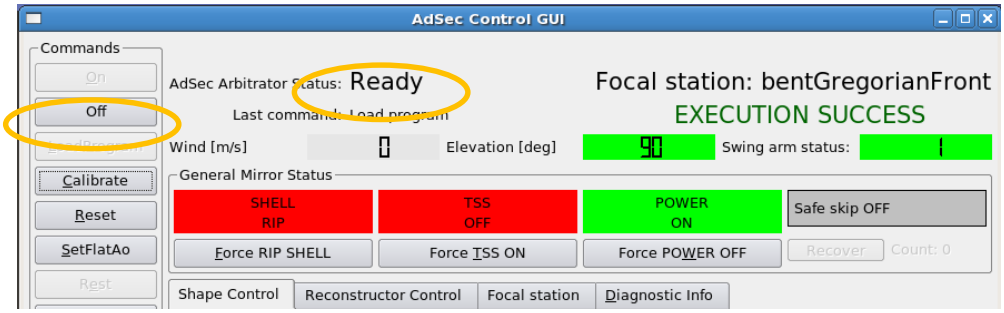
Log on the adsecdx pc and start the engineering GUI panel and the Adsec Arbitrator GUI (see chapter 9.3). The top of AdSec arbitrator GUI should look like the following one. In particular the Adsec Arbitrator Status should be **PowerOFF** and the **On** command button should be enabled.



Now press the **On** button and wait until the system is switched on and the Adsec Arbitrator Status changes **from PowerOff to PowerOn**. After that proceed pushing the **LoadProgram** button.

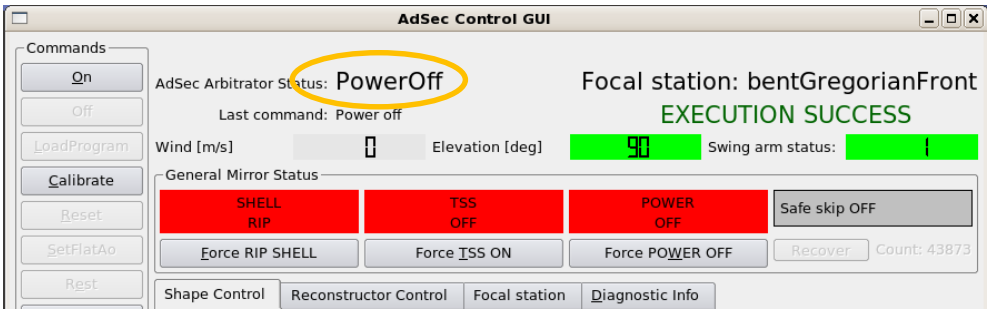


At this point the Adsec Arbitrator Status should change **from PowerOn to Ready**. Now the mirror is ready to operate as shown in chapter 9.2. When you end the observation **this is the right status** in which put the Adaptive Secondary.



If you need to PowerOff the Adaptive Secondary, please check the Safety Remarks (9.1). After that, you can simply power off the system pushing the **PowerOff** button.

When the Adsec Arbitrator Status changes to PowerOff, you can shutdown also the AO software framework as described in 3.2.



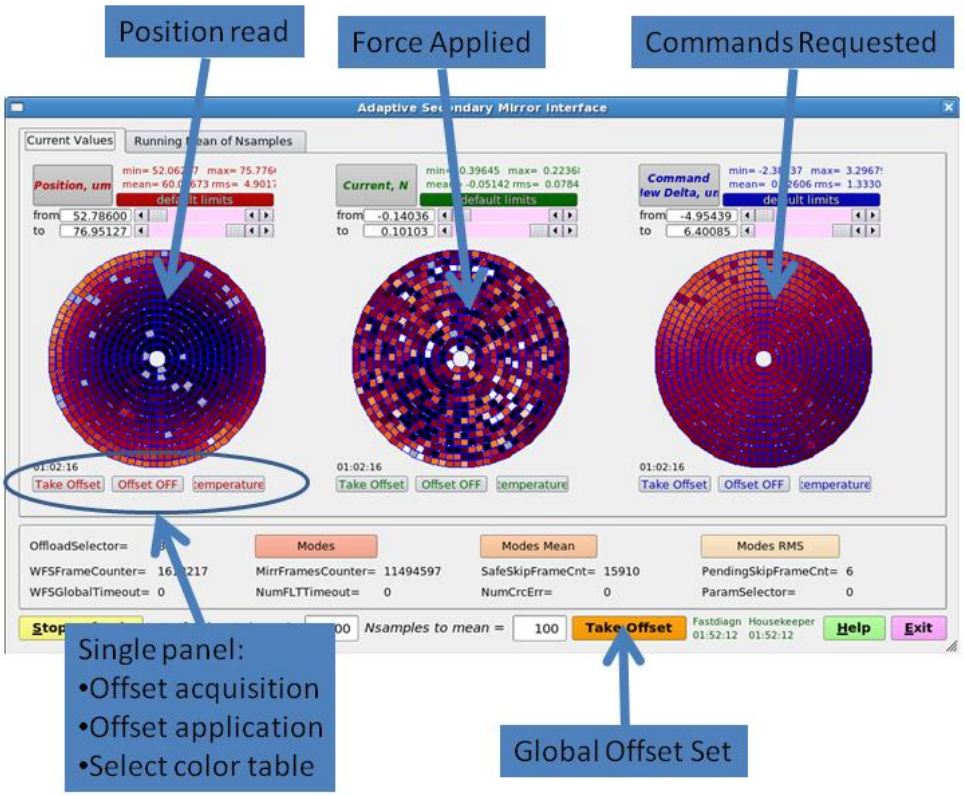
9.7. More on GUIs

In the Adsec engineering GUI panel you can select two more useful GUIs that allow to you to have a more detailed look to the Adaptive Secondary mirror status.

9.7.1. AdSec Mirror GUI

This GUI can be used if the Adsec Arbitrator status is **Ready**, **AOSet** or **AORunning**. It shows in a quick look the capacitive sensors readings, the force applied from each voice coil and the command sent by the Slope Computer.
This GUI can be started from the engineering panel, or from a terminal with the command:

```
AdSecMir_GUI
```



9.7.2. AdSec Housekeeper GUI

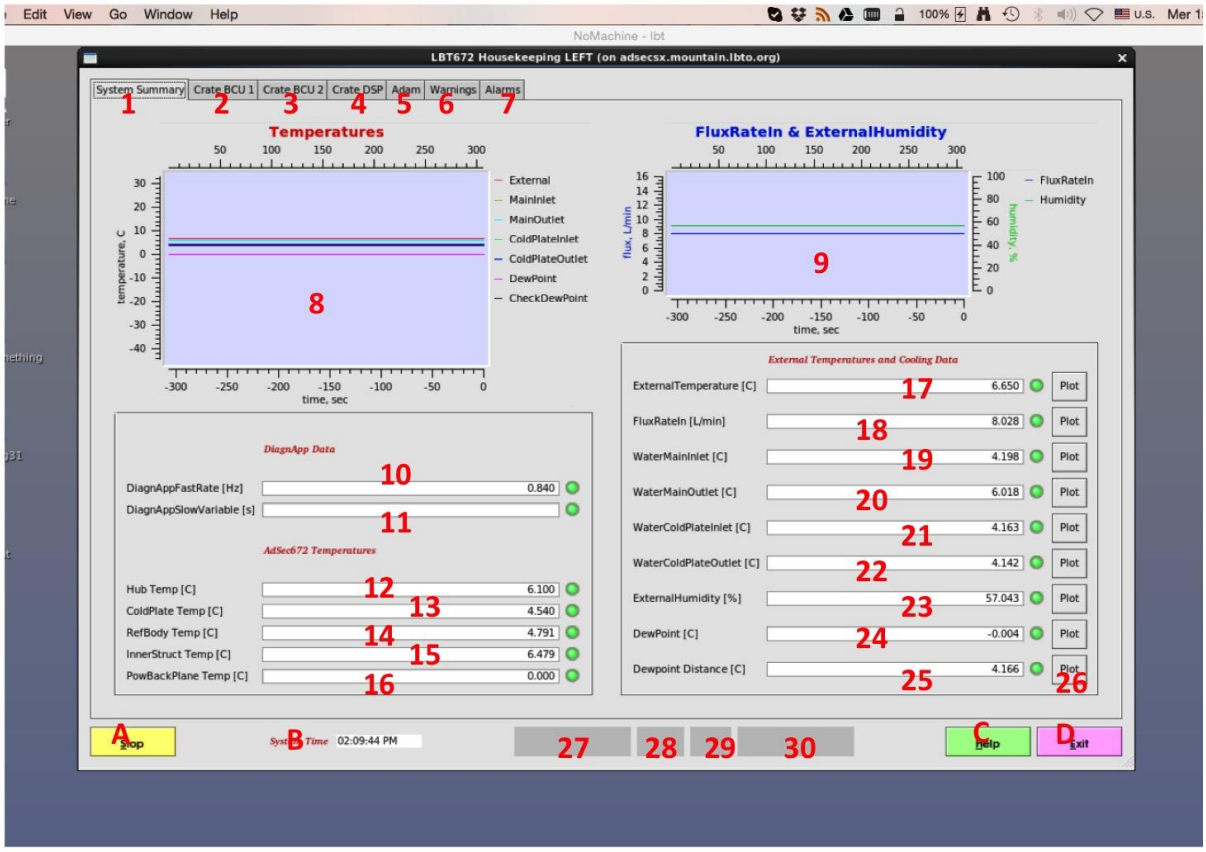
This GUI shows all slow diagnostic data of the adaptive secondary, essentially temperatures of various component of the Adaptive Secondary.

This GUI can be started from the engineering panel, or from a terminal with the command:

```
Housekeeper_gui
```

For the boolean values, dark or green means 0, white or red means 1. The STATUS FLAG on the right of each variable with a range can be GREEN (nothing to report), YELLOW (variable with warning values) or RED (variable out of the alarm thresholds). For the thresholds refers to 641a017.

NB: The ADAM panel STATUS FLAG are not properly working



A. Stop: start/stop acquisition data from Housekeeper process

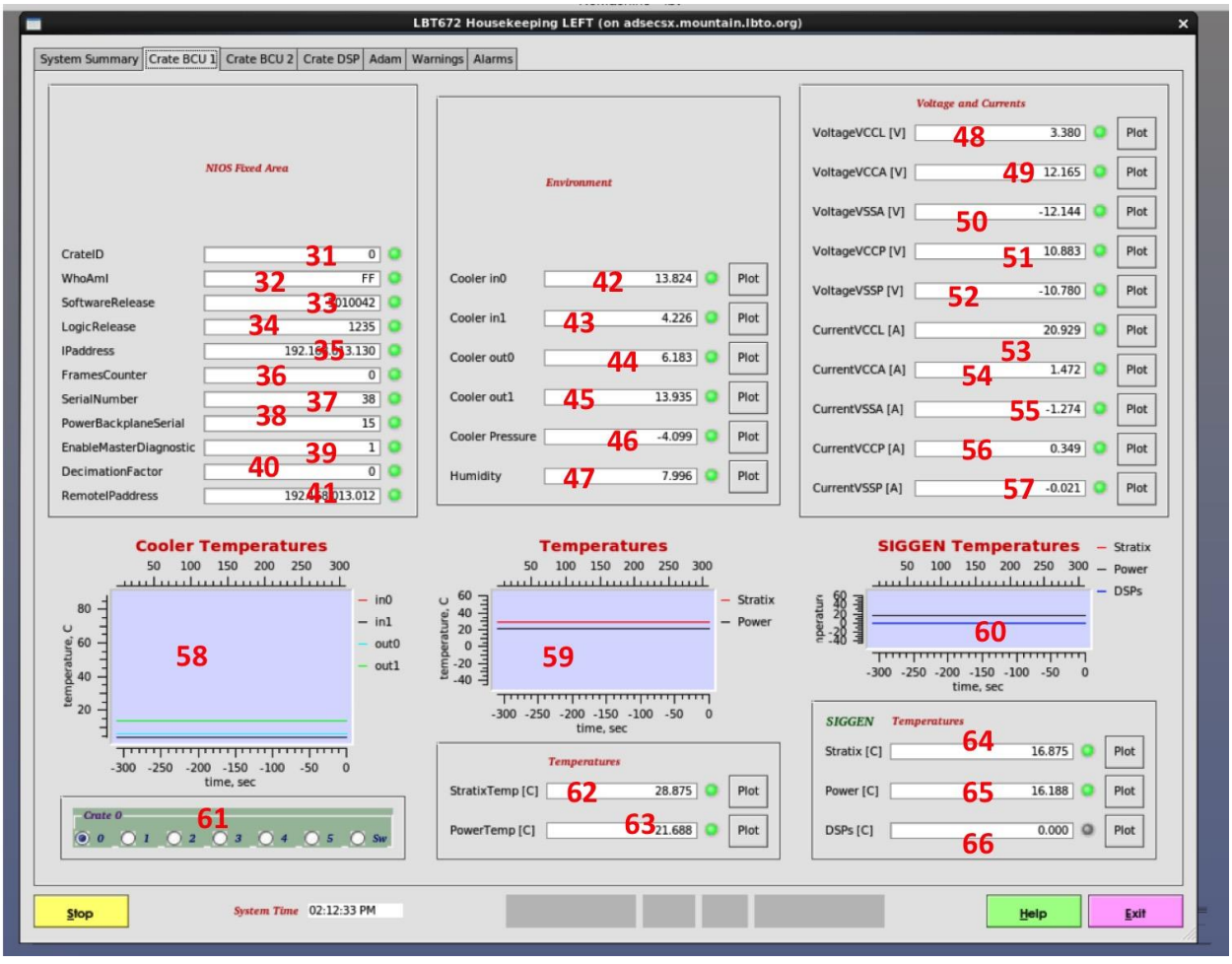
B. Last update timestamp

C. Help: help (if any ☹)

D. Exit: close Housekeeper GUI

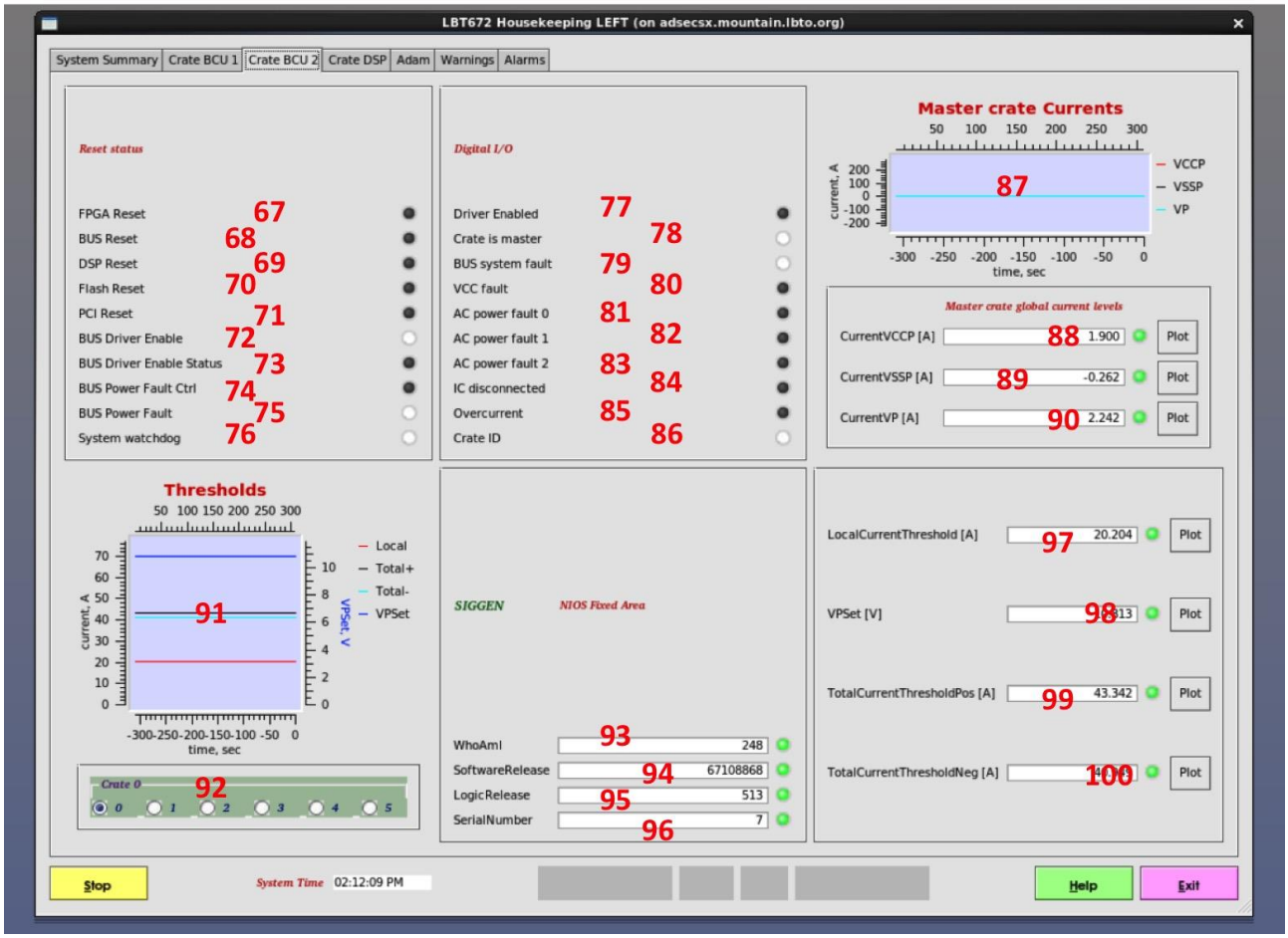
- 1. System summary Panel
- 2. Crate BCU Panel (1)
- 3. Crate BCU Panel (2)
- 4. Crate DSP Panel
- 5. Adam Panel
- 6. Warning Panel
- 7. Alarm Panel
- 8. Panel values summary plot
- 9. Panel values summary plot
- 10. Housekeeper frame rate
- 11. Unused
- 12. Hub temperature
- 13. Coldplate temperature
- 14. Reference Body temperature (see map)
- 15. Inner structure ASM temperature (see map)
- 16. Power Backplane temperature

- 17. External Temperature Probe
- 18. Fluxmeter value
- 19. Water temperature main inlet probe
- 20. Water temperature main outlet probe
- 21. Water temperature coldplate inlet probe
- 22. Water temperature coldplate outlet probe
- 23. External Humidity percentage
- 24. Dewpoint value
- 25. Dewpoint distance from lowest temperature
- 26. Separe plot: if available, a running plot can be requested for the corresponding variable
- 27. Passing the mouse on a value, here you will se the thresolds on minimum values
- 28. number of variables in warning
- 29. number of variables in alarm
- 30. Passing the mouse on a value, here you will se the thresolds on maximun values



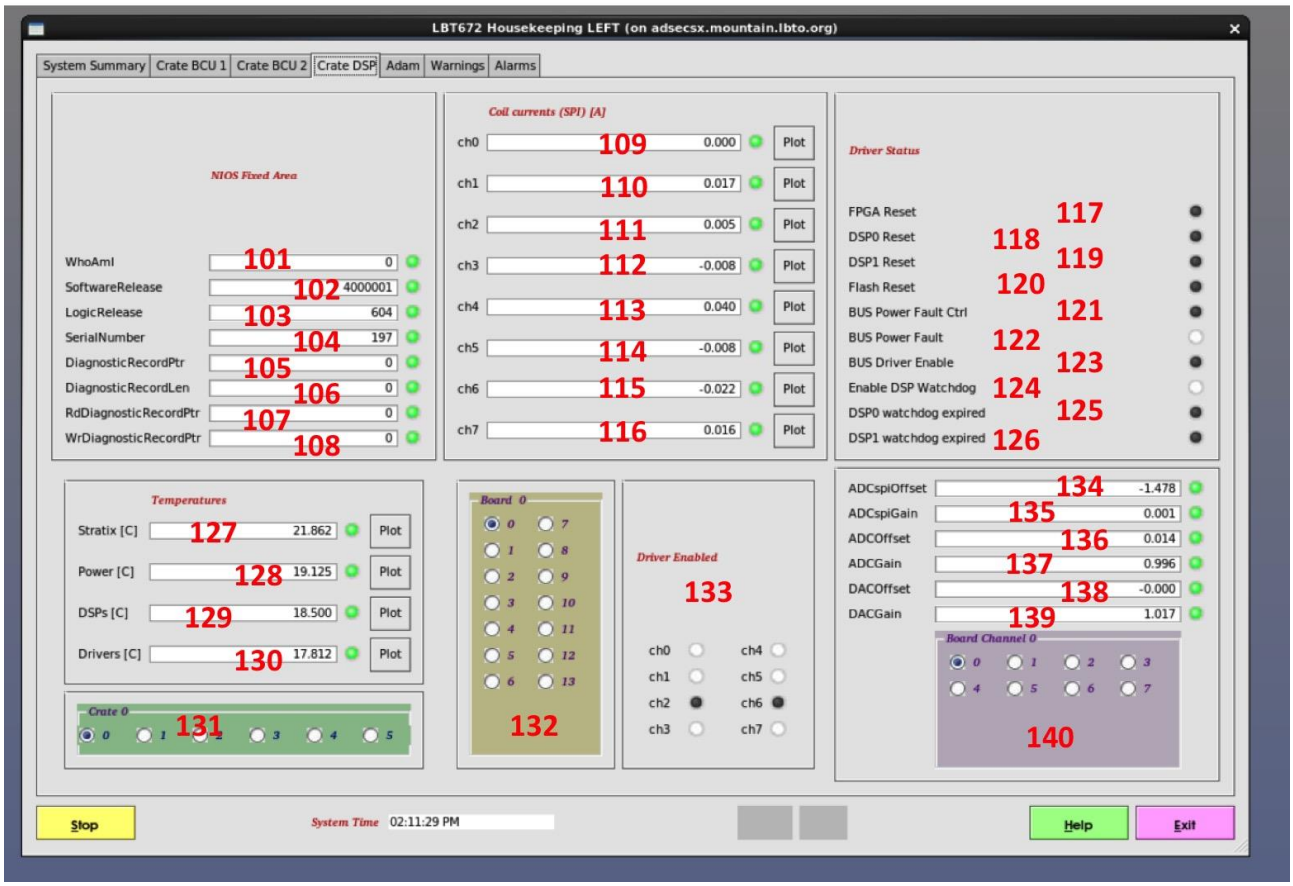
- 31. Crate identification number (0 to 5, Switch/Slave BCU 0)
- 32. Board identification inside crate (FF for BCUs, FC for siggen, F8 accelerometer)
- 33. NIOS identification number
- 34. Logics identification number
- 35. IP address of the BCU

- 36. Diagnostics frame counter
- 37. Board serial number
- 38. Power backplane serial number
- 39. Not relevant here. For details see 641a006
- 40. Not relevant here. For details see 641a006
- 41. Not relevant here. For details see 641a006
- 42. to 47. Analogic sensors. For details see map
- 48. to 57.Analogic parameters of the unit
- 58. Analogic sensors plot
- 59. BCU temperatures plot
- 60. Signal generator/Accelerometer board temperature plot
- 61. Crate selection
- 62. BCU FPGA temperature reading
- 63. BCU Board temperature reading
- 64. Signal generator/Accelerometer board FPGA temperature reading
- 65. Signal generator/Accelerometer board temperature reading
- 66. Signal generator/Accelerometer board DSP temperature (if any)

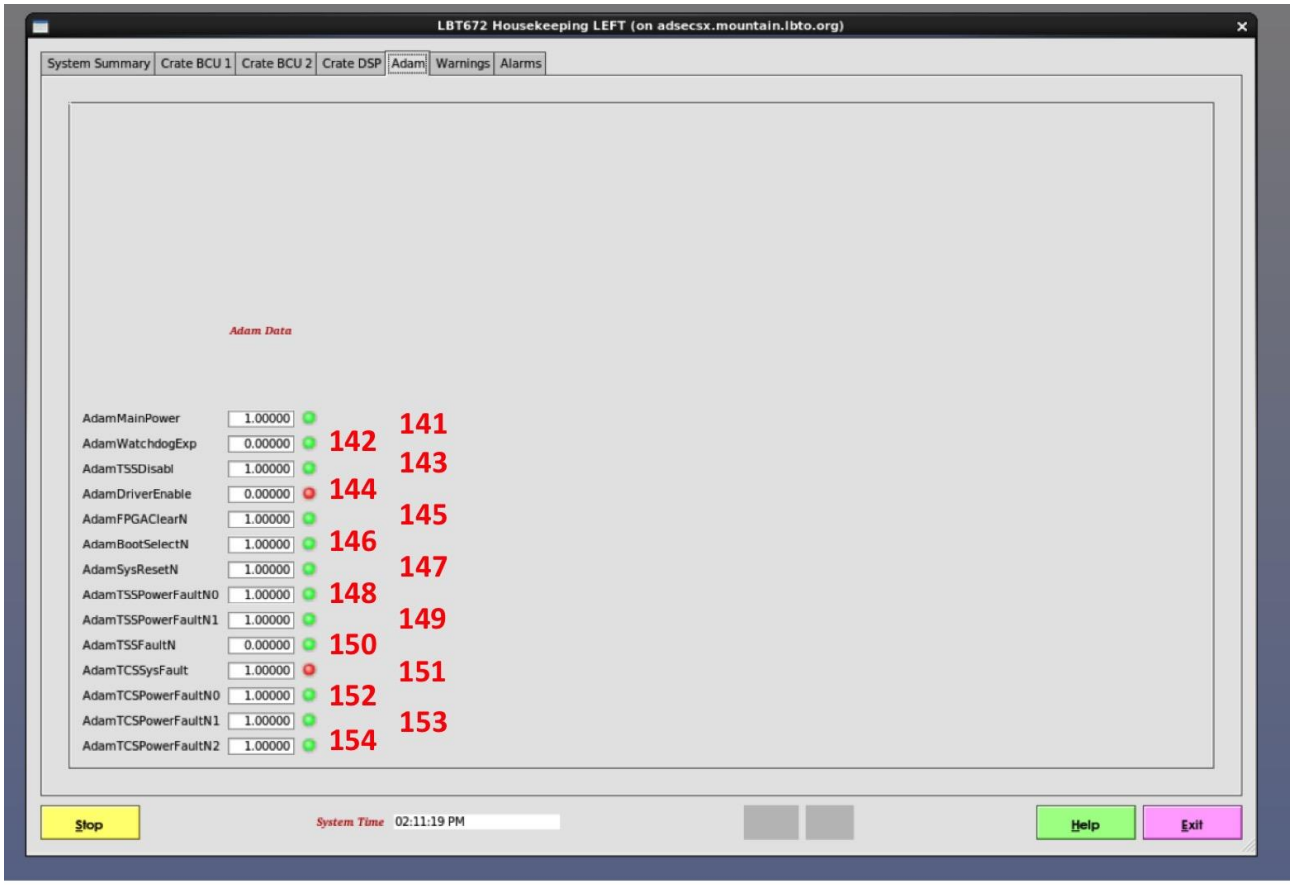


- 67.to 76. Reset status signals (see 641a017)
- 77. Driver enable boolean

- 78. Master crate boolean
- 79. Fault line boolean
- 80. Not relevant here.(for details see 641a017)
- 81. Not relevant here.(for details see 641a017)
- 82. Not relevant here.(for details see 641a017)
- 83. Not relevant here.(for details see 641a017)
- 84. Not relevant here.(for details see 641a017)
- 85. Overcurrent protecton boolean (for details see 641a018)
- 86. Not relevant here.(for details see 641a017)
- 87. to 90. Global current absorption on master crate plot and values (for details see 641a017)
- 91. Overcurrent protection thresholding plot
- 92. Crate selection
- 93. Signal generator/Accelerometer ID
- 94. Signal generator/Accelerometer NIOS ID
- 95. Signal generator/Accelerometer Logics ID
- 96. Signal generator/Accelerometer Serial Number
- 97. Single Crate absolute value current threshold
- 98. Not relevant here.(for details see 641a017)
- 99. System total positive current (valid only for crate master)
- 100. System total negative current (valid only for crate master)



- 101. DSP Board ID
- 102. DSP Board NIOS ID
- 103. DSP Boad Logics ID
- 104. DSP Board Serial number
- 105. to 108. Not relevant for here.
- 109. to 116. Coil current SPI reading
- 117. to 123. Not relevant here.(for details see 641a017)
- 124. Watchdog enable boolean
- 125. DSP 0 Watchdog expiration (for details see 641a018)
- 126. DSP 1 Watchdog expiration (for details see 641a018)
- 127. DSP board FPGA temperature
- 128. DSP board temperature
- 129. DSP board DSPs temperature
- 130. DSP board driver temperature
- 131. Crate selection
- 132. Board selection
- 133. Driver enable status for each actuator
- 134. to 139. DAC and ADC calibration paramenters (see 641a015)
- 140. Actuator selection



- 141. Main power (for details see 641a018)
- 142. Ethernet watchdog expiration (for details see 641a018)

- 143. TSS Status (for details see 641a018)
- 144. Driver Enable (for details see 641a018)
- 145. Not relevant here (for details see 641a018)
- 146. Firmware configuration selection (for details see 641a018)
- 147. Not relevant here (for details see 641a018)
- 148. Failure on TSS power supply number 0
- 149. Failure on TSS power supply number 1
- 150. Not relevant here (for details see 641a018)
- 151. System general fault signal
- 152. Failure on power supply number 0
- 153. Failure on power supply number 1
- 154. Failure on power supply number 2

9.8. Housekeeper configuration files

All the Housekeeper variables limits are listed in the configuration file; **housekeeper.param.** The file is in the directory: **$ADOPT_ROOT/conf/adsec/current/processConf/housekeepe**r. In the Housekeeper GUI, the option **Crate DSP** shows the temperature values of the 4 sensors for;Stratix, Power, DSPs and Drivers for every crate the boards For every crate (0 to 5) we have 14 boards (0 to 13).

Example: If a temperature sensor for one variable is giving bad readings, eg: DSPDRIVER-0008 =10e4 reported in the housekeeper.R/L.log, this sensor probably is bad functioning and to have the system to operate has to be disable it. The 8 means crate 0 board 8. To do so, we can edit the file housekeeper.param and for the DSPDRVERTEMP variable we just do:

FamilyName From To AlarmMin WarnMin WarnMax AlarmMax RunningMeanLen CAF Enabled Slow

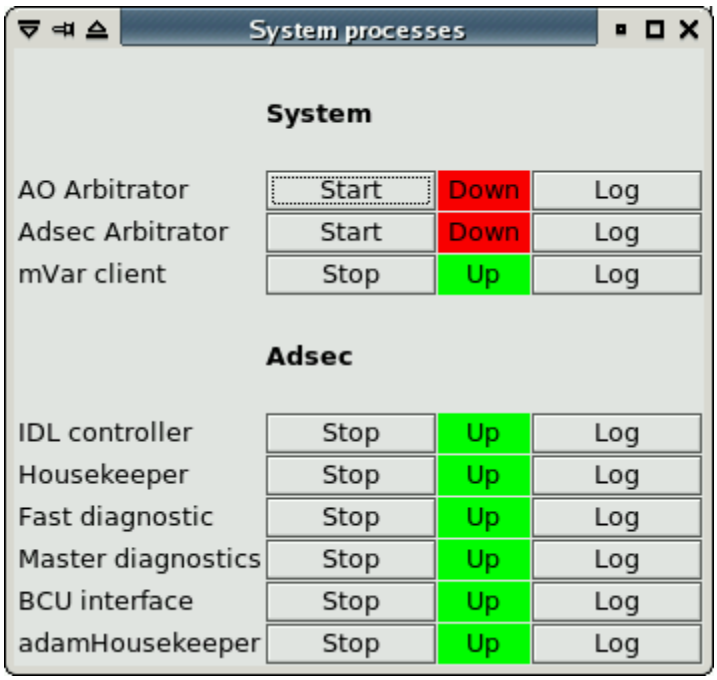
DSPDRIVERTEMP **8 8** -50 -15 inf inf 5 1 **dis** fast

From **8** to **8** is to specify the bad temperature sensor and the **dis** option means disabled

10. Low-level GUIs

10.1. System processes GUI

The System processes GUI lists all the necessary processes for the subsystem currently configured (either the Adaptive Secondary or the WFS), and shows whether each process is running or not, or if it is being initialized. It also provides buttons to start and stop each process, and to visualize their log file.



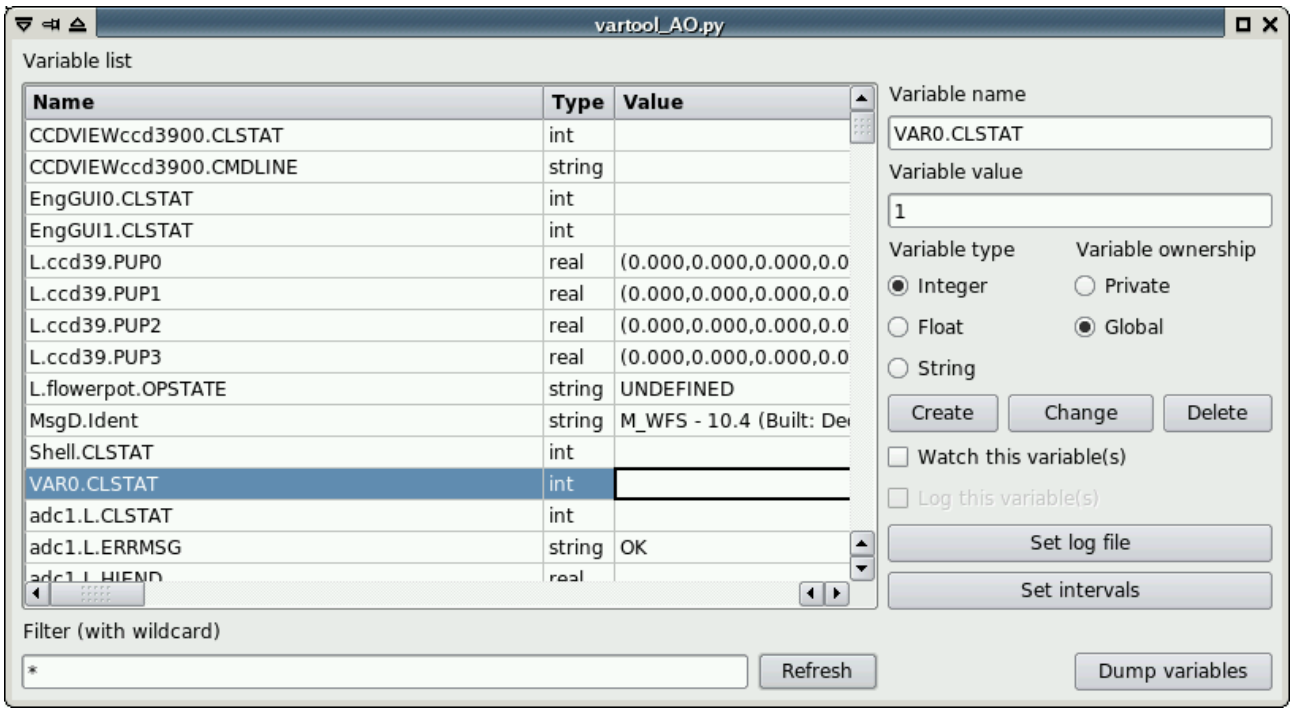
The status indicator of each process can have three values:

- Down (red): the process is not running or not connected to the MsgD
- Init (yellow): the process is running, but not yet correctly initialized
- Up (green): the process is running correctly

The initialization fase is generally very short, except for processes which must wait for some particular condition (for example, the adamHousekeeper will remain in “init” state until the secondary mirror unit is powered on). In normal operating conditions, all lights should be green.

The “log” button will open a window where the log file of the specified process is shown, with color-coded lines for normal logs, error logs, etc. The window does not show the entire log file, but only shows the last part, and follows the changes as they are written. If the process is restarted, the log will continue in the same window. The log window is only a viewer and can be closed at any time with no harm.

10.2. Variable inspector tool



The variable inspector tool is a viewer/editor of the central variable repository maintained by the RTDB. A different variable repository is maintained on each AO computer, even if some information of common interest is replicated.

10.3. Text-based tools

10.3.1. Consumer

"consumer" connects to the shared memory buffer that contains the realtime telemetry, and it has options to dump the data on disk, how many frames to save, etc, like this:

[sxwunit@lbti-sxwfs]$ ./consumer -w telemetry.dat -c 100000 1 masterdiagnostic.L:OPTLOOPBUF

-w <filename> is the output file
-c <count> is how many telemetry frames to store. If omitted, runs until interrupted with Ctrl-C.
"1" is just a stupid argument to have a number in the clientname in case multiple consumers are connected
"masterdiagnostic.L:OPTLOOPBUF" is the name of the shared memory buffer

The number and buffer name must be the last two arguments

10.3.2. Log files

All AO processes, except for graphical interfaces, write information on what they are doing in a log file. Log files are ASCII text and can be opened with any text viewer (see also the logviewer tool, chapter **Error! Reference source not found.**). Care must be taken not to modify the log file if they are opened with an editor such as Emacs or Vi.

The log file has a standardized path and filename:

$ADOPT_LOG/<processname>.<side>.log

Where <processname> is the name of the process, and <side> is the telescope side is it running on. The MsgD has a special name:

$ADOPT_LOG/M_<msgd name>.log

10.3.2.1. Log file archiving

When a process exits, its log file is “archived”, that is, is renamed to make room for the new log file which will be created when the process will restart. The archived filename has the format: $ADOPT_LOG/<processname>.<side>.<timestamp>.log

Where <timestamp> is a timestamp in Unix format, recording the time when the file was archived. Log files are automatically archived and re-openened when they reach a predefined length (around 200MB). Log files are also archived when a process, upon starting, finds out that the log file of the previous instance was not correctly archived. The process will then archive the old log file before opening a new one.

Graphical and text interfaces, which can run in multiple copies at the same time, do not write a log file.

10.3.3. Telemetry files

In addition to log files, a few processes also write telemetry files, which are a special case of log files containing mostly numerical data. They are still in ASCII format and have the following naming scheme:

$ADOPT_LOG/<processname>.<side>_TELEMETRY.tel

Telemetry files are archived in a similar manner as the log files.

11. Common tasks

11.1. System preparation

11.1.1. Using the AOSGUI

- Check on the AOSGUI that the software status is OK (AOS connected, and green light on the AdSec and Wfs software status). See chapter 4.2.1.
- Check if the hardware subsystems (again either the AdSec or both the AdSec and the WFS) are turned on. They will be off if the software was just started. Turn on the needed subsystems. See chapters 4.2.3, 4.2.3 and 4.3.1.
- Set the Adaptive Secondary shell (see chapter 4.3.1).
  Note: telescope conditions (like elevation < 25 degrees) may prevent the shell from setting up, or will cause it to rest it afterwards (see chapter **Error! Reference source not found.**). The shell will need to be set again after these conditions are resolved.

11.1.2. Using the Arbitrator GUIs

In case the AOS GUI is not available, or the AOS is not working properly, it is still possible to setup the system from the Arbitrator GUIs.

- Check that the software on adsecdx is up and running properly (see chapter 3.2)
- Bring up the **adsceng** panel on the adsecdx computer
- Start the AdSec Arbitrator GUI
- Setup the adaptive secondary using the command buttons in the left column. The complete sequence is:
    - "On" (goes to status Operating)
    - "LoadProgram" (goes to status Ready, corresponding to the SAFE label on the AOS)
    - "SetFlatAo" (goes to status AOSet, corresponding to the SET label on the AOS).

If the adaptive secondary is already halfway through this sequence, only the remaining steps need to be performed. Most of the time, the secondary should be in Ready state (SAFE label on the AOS).

To setup the WFS:

- Check that the software on wfsdx is up and running properly (see chapter 3.2).
- Bring up the **wfseng** panel on the wfsdx computer
- Start the WFS Arbitrator GUI
- Select the configuration "complete with ccd 47" and press the Operate button. The setup operation will take a few minutes to complete.

When the WFS Arbitrator GUI reports that the WFS is in state "Operating", the setup is done and observation can proceed.

11.2. System shutdown after observation

11.2.1. Using the AOSGUI

- Rest the Adaptive Secondary mirror shell (see chapter 4.3.1). Do not turn off the secondary mirror, leave it in the status marked SAFE on the AOS GUI.
- Turn off the WFS (if it was turned on initially)

11.2.2. Using the Arbitrator GUIs

- From the AdSec Arbitrator GUI on adsecdx, press the "Rest" button and verify that it goes to "Ready" state.
- From the WFS Arbitrator GUI on wfsdx, press the "Off" button and verify that it goes to "Off" state.

11.3. Seeing limited observation

11.4. AO observation sequence

AO observations are intended to be performed automatically by the instrument through the IIF. It is possible for the AO operator to intervene to repeat or modify a command using the AOS Command

GUI, where the original command parameters are displayed. In case of a command failure, or if a command must be repeated, the operator can modify the parameters on the AOS Command GUI and repeat the command.

This chapter gives a resume of the typical AO sequence, an overview of what happens during each command, and what is possible for the operator do to in each case.

11.4.1. PresetAO

An AO observation starts with a PresetAO command, which tells the AO system the main parameters of the following observation: which instrument and focal station will be used, and the reference star magnitude and position. The command is received by the AOS and forwarded to the lower-level arbitrators, where the following parameter checks are done:

By the AdSec Arbitrator:

- focal station name is among the ones defined for the Switch BCU input ports

By the Wfs Arbitrator:

- Star magnitude is within the limits of the AO parameters table
- Star position is within the AO field-of-view of the wfs stages
- A board setup file with the same name of the instrument is present

The AO parameters table and AO field-of-view are defined in two different WFS calibration files (see []). The board setup file will be searched in the board setup directory (see []).

If any of the checks fails, the command will report a “Validation failed” or “Retry” error. In this case, the command must be repeated with valid parameters before AO observations can go ahead.

When the parameters are successfully validated, the AdSec and WFS are configured, ccd darks are taken and the board setup file is applied. Tracking loops (rerotator, adc) are turned on. If the WFS CCD displays are active, they may stop for a while during the CCD reconfiguration.

A PresetAO command can be repeated any number of times without harm. Since it may take a certain amount of time (up to a couple of minutes if everything must be reconfigured), the PresetAO can be sent while the telescope is slewing to speed up AO operations.

11.4.1.1. Error conditions and recovery

- Focal station name is not recognized. *Solution*: repeat the command with a recognized focal station name
- Instrument name is not recognized. *Solution*: repeat the command with a recognized instrument name
- Star magnitude is too faint or too bright: *Solution*: repeat the command with a star magnitude within accepted bounds
- Star position outside FoV. *Solution*: repeat the command with a star position inside the accepted FoV
- Any other problem is a symptom of hardware failure. See chapter [].

11.4.2. AcquireRefAO

The AcquireRefAO tells the AO system to acquire the reference star and configure the system for close loop operations. It has no parameters since everything was specified by the previous PresetAO command. When the AcquireRefAO command is received by the AOS, the following sequence happens:

- A sky image is taken wih the ccd47 and the position of the reference star is measured
- WFS stages are moved to bring the reference star on the target position
- The magnitude of the star is measured on the ccd39 and compared with the one given by the system during the PresetAO. If there is a difference, the system is reconfigured (basically repeating a PresetAO command) for the new magnitude.
- The AO loop is temporarily closed with a special set of parameters to center the camera lens.
- Once the camera lens is centered, the temporary AO loop is opened and the system is configured with the final parameters.

A number of things may prevent the command from completing successfully. The following section details the most common problems encountered.

Note: the fact that a temporary AO loop is closed during this command means that telescope guiding and active optics must be stopped during command execution. This is done automatically by the telescope when in ADAPTIVE mode, but must be done manually if the telescope was preset in ACTIVE mode.

11.4.2.1. Error conditions and recovery

- Star not found on ccd47. It may happen that the telescope pointing was not accurate enough (the ccd47 field has a diameter of about 15 arcseconds), or that the star position given the previous PresetAO command was incorrect.
- Star found, but of very different magnitude. In this case the system will assume that the wrong star was found, and will stop.
- Camera lens position not reached: if the seeing is very bad, it may prevent a good measurement of the pupil position on ccd39, causing the camera lens centering to fail.
- AdSec safety failure during the temporary closed loop: see chapter [] regarding this condition.

In case of any error, since there are no parameters for the command, the only option for the operator is to solve the external problem and try again. If any of the parameters sent with the PresetAO command need to change (for example, the star position or magnitude needs to be changed), the operator must first send another PresetAO command and then repeat the AcquireRefAO.

An AcquireRefAO can be repeated any number of times.

11.4.3. StartAO

Once the AcquireRefAO has completed, the loop can be closed immediately with the StartAO command. This command has no parameters and no failure modes (apart from hardware failures), since it just enabled the “fastlink” fiber over which the slopes are transmitted.

Once the system is in closed loop, the realtime part will go on indefinitely until another command is sent, or until a safety failure occurs.

11.4.3.1. Error conditions and recovery

No errors are expected during the command. After that, the system is in closed loop and an AdSec safety failure can occur. See chapter [].

11.4.4. PauseAO/ResumeAO

The PauseAO command suspends the AO loop, while the ResumeAO command resumes a previously paused loop. Their operation include a check on the incoming light on ccd39 before resuming the loop:

PauseAO:

- Records illumination level on ccd39
- Disables the “fastlink” fiber and stops the flow of slopes to the adaptive secondary

ResumeAO:

- Checks that the illumination level is the same as recorded during the ccd39
- Enables the “fastlink” fiber and resumes the flow of slopes to the adaptive secondary

The check on the illumination level prevents resuming the loop if, during the pause, the reference star is not in the WFS field of view anymore. This may happen in case of tracking drifts, or if some incorrect offsets were executed during the pause.

11.4.4.1. Error conditions and recovery

- Illumination level check fails on ccd39 during resume. Solution: if the reference star position is known, fix the WFS position using an OffsetAO command and try again. Otherwise, the loop must be opened with a StopAO command and the AO sequence started again from the PresetAO.

11.4.5. OffsetAO

The OffsetAO can be executed in any condition (loop open, closed, or paused).

If the loop is open or paused, it will be executed simply moving the WFS stages by the specified amount. The reference star will be then lost, unless the same offset is executed by the telescope mount.

If the loop is closed, the WFS stages will be moved in small steps of 0.3 mm, waiting at each step for the tip-tilt offloading to recover the movement. The execution time for the offset is correspondingly greater.

11.4.6. Other failure modes

11.4.6.1. AdSec safety fault

When the loop is closed, the AdSec mirror shape and forces are under continous safety check by the FastDiagnostic process. If an out-of-range condition is detected, the power to the mirror actuators will be turned off, terminating immediately the AO loop and causing the shell to go back to rest position.
In this event, the AdSec mirror will execute its own "RecoverFailure" routine, which brings it back to the Operating state where it is ready to be set again. In the meantime, any AO operation in progress will have been cancelled, and the WFS has been notified of the event and stopped as well in order to stop the flow of slopes to the secondary mirror.

The operator must set the shell again (see chapter 11.1.1) and restart the AO observation from the PresetAO command.

11.4.6.2. Hardware failure

If a hardware failure happens, it will be generally impossible to continue the AO observation. It is not feasible to list all possible hardware failures. What will happen is that commands will start to fail randomly with specific error messages about the faulted hardware component. It will be necessary to look at the Arbitrator GUIs and log files to properly diagnose and fix the problem.

12. Calibration procedures

12.1. Interaction matrix calibration

12.1.1. Preparation

The measurement is done in daytime using the retroreflector. It is critical that the system optical setup is as similar as possible to the one used during night observation. The setup is described in detail in the "FLAO User Procedures" in chapter 7 (daytime AO closed loop). Basic setup of the AO system (software startup, power on, etc) is also described in the same document.

This document assumes that the system has been setup according to the User Procedures document, and that an AO daytime closed loop has been successfully closed.

12.1.2. Measurement parameters

When measuring a reconstructor matrix, a few parameters must be chosen before starting the measure:

- AdSec *modal basis*
- WFS CCD binning

12.1.2.1. Modal basis

The adsec modal basis (sometimes called modes-to-commands or M2C matrix) is a list of command vectors. Each vector contains the actuator commands that are needed to generate a certain modal shape on the mirror. The modal shapes definition, and the calculation of the corresponding actuator commands, is done with a specialized procedure that reduces data taken with the 4D interferometer.

In practice, a modal basis is a directory on the AdSec computer, which contains several files and subdirectories. Here is a typical layout:

```
[AOeng@adsecdx M2C]$ ls -l KL_v16
```

```
drwxr-xr-x   2 AOeng aoacct     32768 May 15 02:48 RECs
drwxr-xr-x   2 AOeng aoacct      4096 Nov 12  2013 disturb
drwxr-xr-x   3 AOeng aoacct      4096 Oct 15  2010 filtering
drwxrwxrwx   3 AOeng aoacct   1019904 Jul  2 04:55 gain
drwxr-xr-x 130 AOeng aoacct      4096 Oct 13  2013 intmatAcq
-rw-r--r--   1 AOeng aoacct   3617280 Sep 15  2013 m2c.fits
drqxr-xr-x   2 AOeng aoacct      4096 May  1 19:46 modesAmp
```

The name "KL_v16" identifies a modal basis, whose contents are found in the "m2c.fits" file. Several subdirectories exist, which contain data that is only valid when used together with this specific modal basis: reconstructors, time filtering matrices, gain vectors, etc.

This directory structure has been created when the modal basis was measured, and there is no need to setup it manually.

12.1.2.2. WFS CCD binning

The FLAO WFS can operate in four different binning modes, numbered from 1 to 4 inclusive, depending on the reference star brightness. In order to close the loop using a binning mode, a reconstructor matrix must have been measured with the WFS configured with the same binning. Thus, multiple measurements of the same modal basis may be needed, up to one for each binning.

12.1.3. Modal history generation

The first step is to generate a *modal history*, that is, a sequence of push-pull commands that will be loaded on the AdSec and "played" during the measurement. Such a sequence is also called a disturbance, because it uses the disturbance feature of the AdSec in order to work.

This step can be executed offline, as it does not need any input from the live system except for the presence of the modal basis on disk.

The parameters needed to generate a modal history are (please refer to the screenshot on the next page for parameters placement):

1. The modal basis to use (already decided before)
2. How many modes to measure: initially 10, will be increased with further iterations.
3. Type of modal history (push-pull or sinusoidal): only push-pull is currently allowed
4. No. of frames for each movement: currently fixed at 3 frames
5. Push-pull cycles: as many as possible, but without exceeding the AdSec disturbance capacity, which is 4000 frames. Thus the total product No. of frames * cycles *2 must be <= 4000. The GUI will show the total in green or in red (if it is over the threshold).
6. Amplitude file: this is a file containing a vector of amplitudes, one for each mode. Unless a hand-optimized one is available, one of the pre-defined ones like 672_0.2.fits is a good start.

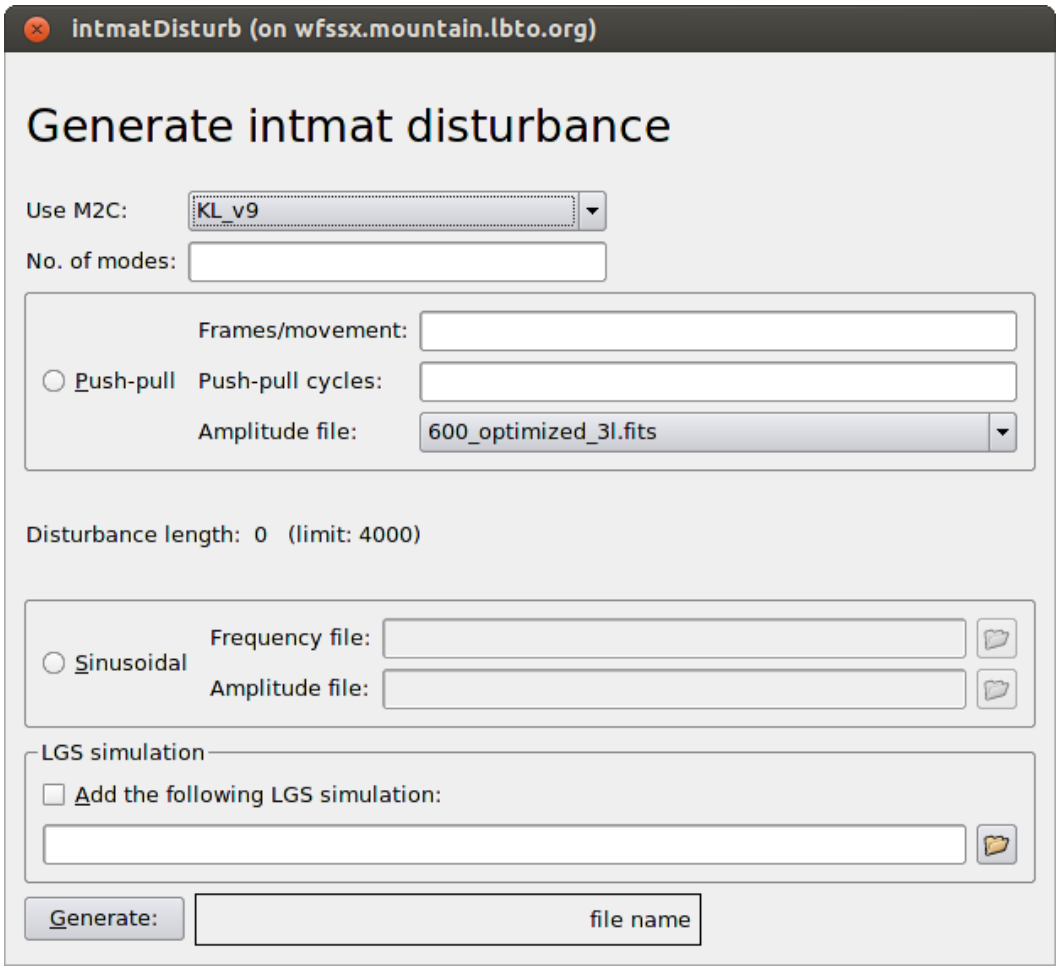
After parameters have been entered, click on "Generate". After a few seconds, a tracking number will appear on the right. This is the disturbance tracking number, which must be noted down for later use.

Prerequisites:

- WFS software must be up and running, as described in the User Procedures document in section 2 ("Start and check Software status").

Action	Procedure	Notes
• Start intmatDisturbGui	*[AOeng@wfsdx ~]$ intmatDisturbGui*	
• Enter main parameters	Enter modal basis, number of modes to measure.	
• Enter push-pull parameters	Click on push-pull radio button. Enter no. of frames/movement	

	and cycles Check that total is <= 4000	
• Generate disturbance	Click on “Generate”	
• Note down the tracking number		Format is YYYYMMDD_HHMMSS



Note:

A modal history can be re-used as many times as needed, and is independent from WFS CCD binning.

12.1.4. Interaction matrix measurement

After a modal history is available, the interaction matrix can be measured. It is critical that the measuring conditions are as similar as possible to the night-time conditions. This means:

1. No lights in the dome
2. No vibrations (as far as possible)
3. All WFS tracking loops (rotation, camera lens, anti drift) active, as described below.

The measurement is usually done in closed loop. A full system setup is needed, that can be obtained following the User Procedures manual in section 6.5 (“Close AO loop in daytime”). That procedure includes all vibration mitigation needed for daytime operation.

If no reconstructor matrix is available for the current WFS pupil and binning combination, a closed-loop measurement is not possible. It is suggested to do a low-order measurement (10 modes) with very long averating (100 cycles or more) in order to obtain a preliminary reconstructor matrix. The measurement can be then iterated (see the "iteration" chapter later on).

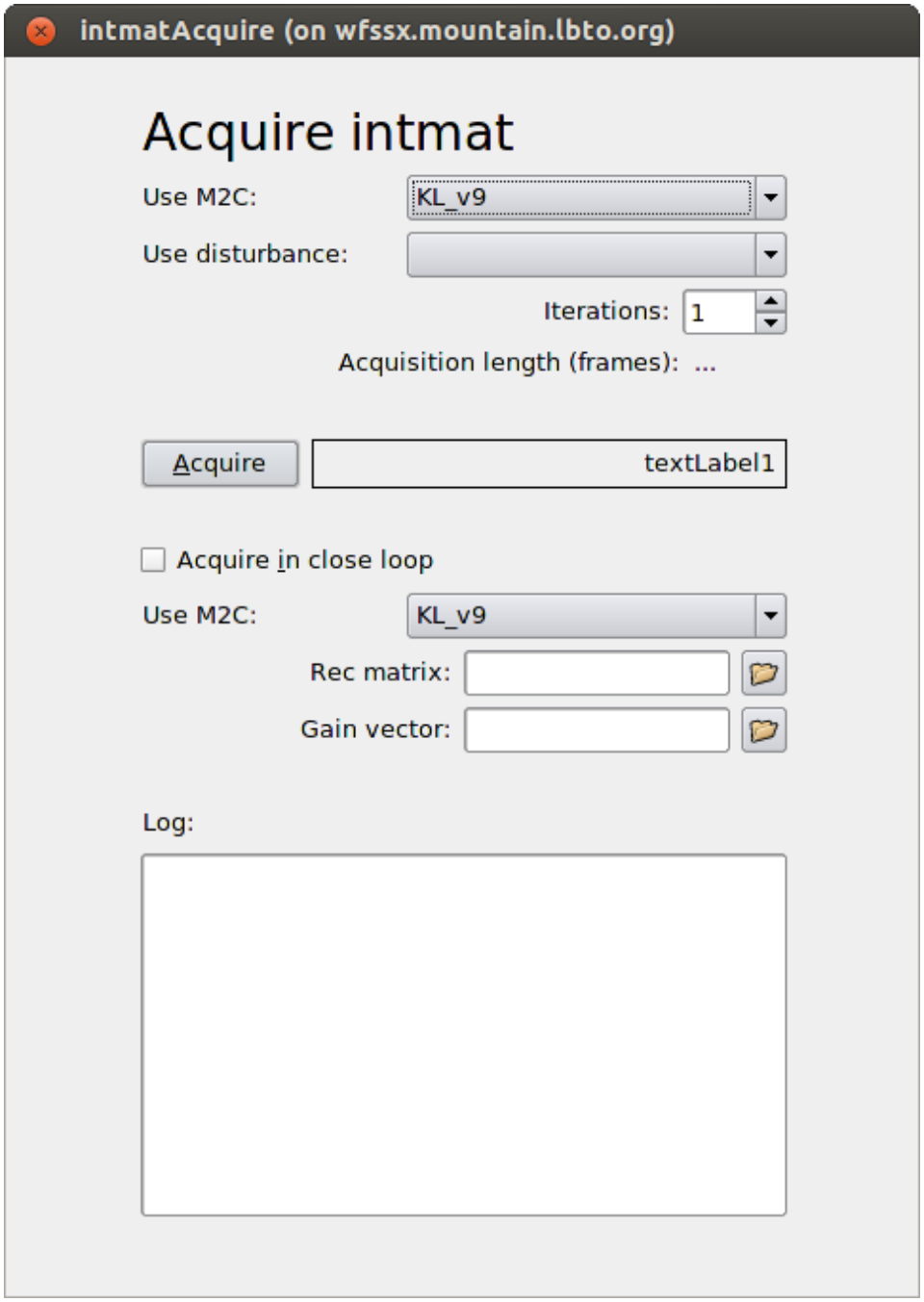
The measurement is done at a loop speed of 600 Hz. This is a compromise between the necessity of going as fast as possible, in order to minimize vibrations, and the mirror settling time. This speed should be entered in the WFS Arbitrator GUI as described below. Speeds different from 600 Hz have not been tested and may not work.

Prerequisites:

- Execute User Procederues section 6.5 - Close AO loop in daytime

Action	Procedure	Notes
1. Open the AO loop	Press STOP on the AOS command GUI	
2. Set 600 Hz loop speed	Open WfsControl GUI. In the "Loop params" panel enter the following parameters: Binning = the one chosen for the measurement Loop frequency = 600 Hz Modulation: 3 (bin 1-2) or 6 (bin 3-4) And click the "Apply" button	
3. Start tracking loops	On the WfsControl GUI, make sure that the three tracking loops: 1) rotator tracking 2) Camera lens tracking 3) Anti drift are enabled. Click on the corresponding Enable or On buttons if needed.	
4. Start intmatAcquireGui	*[AOeng@wfsdx ~]$ intmatAcquireGui*	
5. Select the M2C	Same one used during disturbance generation	
6. Select disturbance	If a different one is needed, select it.	The last generated disturbance is pre-selected
7. Enter the no. of iterations	Use at least 4 to fully use the AdSec internal buffers. If more averaging is wanted, use a higher multiple of four (i.e. 16, 48, etc).	Time needed is about 1 minute every 4 iterations including overheads.
8. Enter the reconstructor	If a reconstructor is available for the current pupil and binning combination, click the "Acquire in closed loop" checkbox and select M2C, reconstructor, and a low gain vector like 0.05.fits	An old reconstructor with a different M2C but the same WFS pupils may be used if it is still working fine.
9. Start measurement	Click the "Acquire" button on the	

		GUI	
10	Wait for completion	It will take about 1 minute for every 4 iterations.	
11	Note down tracking number	Tracking number appears next to the Acquire button	Same YYYY… format as before.



12.1.5. Reconstructor matrix generation

After an interaction matrix has been acquired, one or more reconstructor may be generated from it. Typically, a full interaction matrix is acquired with 500 or 600 modes, and a set of reconstructors with progressively more modes is generated (for example: 10, 100, 250 and 500).

The generation is done with a GUI, which however uses terminal input for some parameters. It is best to use a terminal dedicated to this GUI to avoid conflicts.

The generation can be done "offline", just starting the GUI. Only the WFS software is needed to be running.

Prerequisites:

- WFS software must be up and running, as described in the User Procedures document in section 2 ("Start and check Software status").

Action	Procedure	Notes
1. Start intmatAnalyseGui	*[AOeng@wfsdx ~]$ intmatAnalyseGui*	
2. Select acquisition	Select M2C and tracking number to analyse	Last tracking number is pre-selected
3. Select parameters	"skip frame" and "avg frames" are always "2". Check "remove tip-tilt" and uncheck "only check saturation"	
4. Start analysis	Click on the Analyse button	
5. Answer questions on terminal	Often the terminal pauses and asks to press Enter to continue, in order to look at a plot or graph	Sometimes the terminal is hidden behind big plots. Close or move them away.

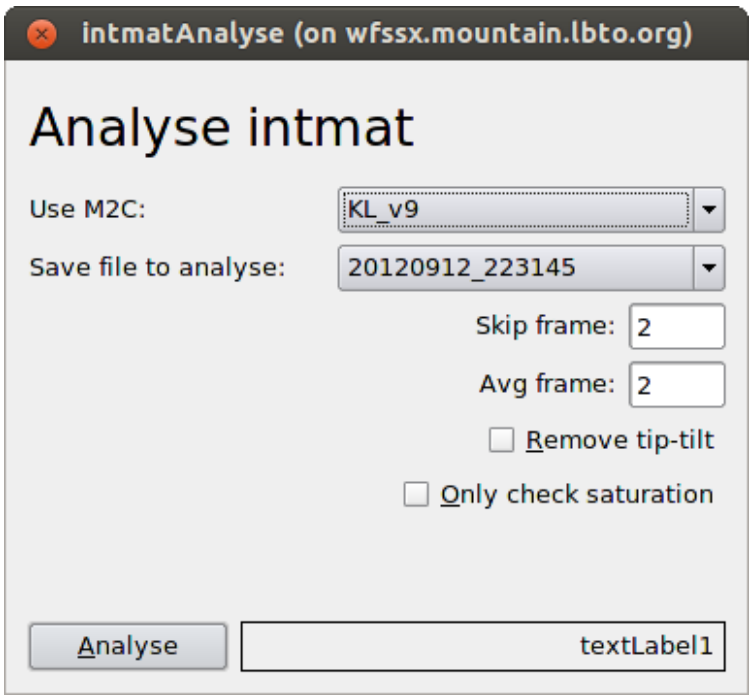
12.1.5.1. Iteration

Once a reconstructor is available, it is recommended to repeat the measurement in closed loop. Repeat all steps in section 5, and when configuring the acquisition GUI, click the "Acquire in closed loop" checkbox and select the M2C and reconstructor. A very low gain (like 0.05) is recommended.

The typical reconstructor iteration is as follows:

1. 10 modes (open loop)
2. 10 modes (in closed loop with previous 10 modes)
3. 50 modes (in closed loop with previous 10 modes)
4. 100 modes (in closed loop with previous 50 modes)
5. 400 modes (in closed loop with previous 100 modes)
6. 400 modes (in closed loop with previous 400 modes)

Only the last reconstructor is considered useful. In addition, a 10 modes reconstructor is generated from the last interaction matrix measurement, for use during the camera lens centering loop.



13. Saving diagnostic data

13.1. Data format description

Diagnostic data is stored into a directory on the adsec computer:

`/local/aomeas/adsec_data`

the wfs computer mounts this directory via NFS using the same name, as described in sections 1.8 and 1.9. The /local/aomeas prefix can be changed using the ADOPT_MEAS environment variable.

Data is organized into “tracking numbers”. Each tracking number is a directory containing a number of data files. The directory name is a timestamp of when the data saving started and, in order avoid having thousands of subdirectories, the adsec_data directory is further subdivided into directories for each day. Thus the full path for a given tracking number is:

`$ADOPT_MEAS/adsec_data/YYYYMMDD/YYYYMMDD_HHMMSS`

where YYYYMMDD is a date in year-month-day format (like “20151125”) and HHMMSS is a time in 24-hour-minute-format (like “115834”).

Typically, data inside a tracking number is not analyzed manually, but using the IDL elab_lib tool described in the next section, and the user only has to record the tracking number that identifies a particular acquisition of interest.

Each tracking number contains a number of files. Most files also contains the same timestamp in their filename, in order to allow quick identification during data analysis. In the following table, “xxx” stands for the full YYYYMMDD_HHMMSS. Thus for example, the name “Frames_xxx.fits” will be saved on disk with a name like “Frames_20160301_224946.fits”.

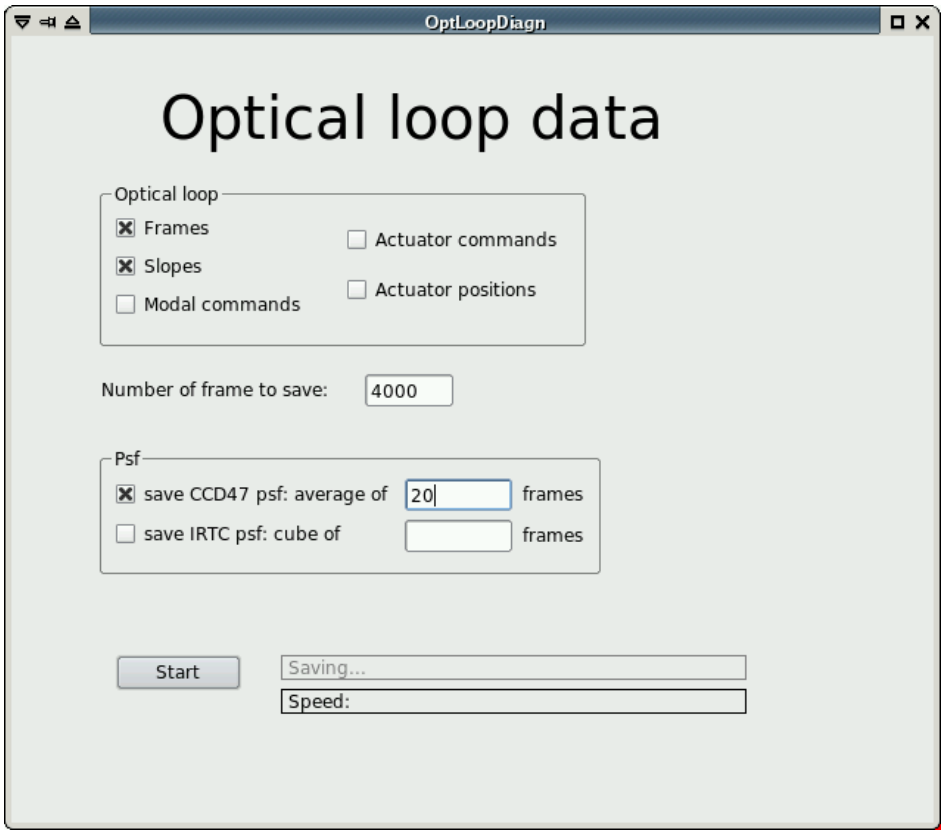
Table 1: List of files included in a tracking number

Filename	Format*	Description	Typical size**
adsec.sav	IDL SAV file	Contains a number of IDL variables with detailed AdSec status information.	15 MB
AntiDrift_xxx.fits	4xN FITS, FLOAT 32 bits	anti-drift" background corrections applied to the ccd39 background file	68 KB
Commands_xxx.fits	672xN FITS, FLOAT 32 bits	Mirror actuator commands	11 MB
CrcErrors_xxx.fits	N-elements FITS, INT 32bits	CRC error counter on fastlink fiber	20 KB
DarkApplications_xxx.txt	ASCII	Timestamps of ccd39 background changes	From zero to few KB.
Dimm_xxx.fits	Variable length FITS, FLOAT 32 bits	Dimm values received from TCS during data acqusition	3K
FlTimeout_xxx.fits	N-elements FITS, INT 32bits	Timeout counter on fastlink fiber	20 KB
Frames_xxx.fits	80x80xN FITS, INT 16 bits	Ccd39 pixel frames	49 MB
FramesCounter_xxx.fits	N-elements FITS, INT 32bits	Slope computer-generated frame counter	20 KB
GuideCam_xxx.fits	Variable length FITS, FLOAT 32bits		3 KB
LoopClosed_xxx.fits	N-elements FITS, INT 32bits	History of loop closed flag during data acquisition	20 KB
MirrorCounter_xxx.fits	N-elements FITS, INT 32bits	MirrorBCU-generated frame counter	20 KB
Modes_xxx.fits	672xN FITS, FLOAT 32 bits	Delta-Modes (reconstructor output)	11 MB
PendingCounter_xxx.fits	N-elements FITS, INT 32bits	mirror "pending" frame counter	20 KB
Positions_xxx.fits	672xN FITS, FLOAT 32 bits	Mirror actuator positions	11 MB
SkipCounter_xxx.fits	N-elements FITS, INT 32bits	"skip frame" frame counter	20 KB
Slopes_xxx.fits	1600xN FITS, FLOAT 32 bits	Slopes calculated by slope computer BCU	25 MB
Timestamp_xxx.fits	N-elements FITS, INT 32bits		20 KB
wfs.fits	FITS with no data, only header	Contains the WFS device status in the FITS header.	17 KB
WFSGlobalTimeout_xxx.fits	N-elements FITS, INT 32bits	Global timeout counter	20 KB
WindDir_xxx.fits	Variable length FITS, FLOAT 32bits	Wind direction data received from TCS during data acquisition	3K

WindSpeed_xxx.fits	Variable length FITS, FLOAT 32bits	Wind speed data received from TCS during data acquisition	3K

* In the format specification, N refers to the dataset length (usually 4000 frames)
**Typical size is given for a dataset length of 4000 frames.

13.2. Optical Loop Diagnostic GUI



The *Optical Loop Diagnostic GUI* is a tool to trigger saving of diagnostics data sets. The user has to choose which data must be saved using the checkboxes in the "Optical Loop" panel. If the system is in open loop, only Frames and Slopes are available (if the other data types are selected, they will be saved but will only contains zero). Data types not described in the checkbox, but listed in Table 1, are always saved.

The number of frames to save can range from a minimum of 2 to a maximum only limited by the amount of available RAM on the wfs computer. A typical value for AO data is 4000 frames, however values up to 20000 frames have been used with success.

Optionally, a PSF from one of the available instruments (ccd47, IRTC/LUCI) can be selected, and it will be saved as an additional FITS file in the tracking number directory. Use of the PSF checkbox will trigger specialized routines to trigger frame acquisition on the instrument, and may fail if the instrument has not properly setup.

Once all values have been initialized, click the "Start" button to star the data acquisition. The display will show the tracking number and the acquisition progress in terms of number of frames and data speed in Hz, refreshed about once per second. When the acquisition is complete, the

display will show “Saving…” while data is saved on disk. Once the saving is complete, the tracking number turns green.

14. Elaboration library (elab_lib)

The elaboration library is an IDL object-oriented library dedicated to the analysis of the FLAO diagnostic data described in section 13.

An introduction to the elaboration library can be found on the LBTO wiki site:

http://wiki.lbto.org/bin/view/AdaptiveOptics/Elab

The elab_lib has an internal help system that provides a concise description of (nearly) every procedure and function. See the “help” section in the Getting Started page of the above wiki site for more information.

15. Configuration files

15.1. File format

Configuration is stored into *configuration files*. A configuration file is an ASCII file, with a filename that by convention ends in “.conf”.
Configuration files are stored into this directory:

```
$ADOPT_ROOT/conf/<system>/current/processConf/
```

where <system> is one of “wfs” or “adsec”. “current” is a soft link to a specific subsystem name (like “W1” or “672a”) and is usually created by the prepare.py procedure (described in section 2.2)

There is typically one configuration file for each process running in the AO Supervisor. A process looks up the configuration file name using the identity provided by the “-i” command line switch. For example, if a SimpleMotorCtrl instance is started up with this command line:

```
$ADOPT_ROOT/bin/SimpleMotorCtrl -i adc2
```

it will try to load this configuration file:

```
$ADOPT_ROOT/conf/<system>/current/processConf/adc2.conf
```

Configuration files are a list of keywords. Each keyword occupies a text line with three components, separated by one or more spaces and/or tab characters:

```
<name>    <type>   <value>
```

<name>: the keyword name. Letters (case sensitive) and numbers are accepted.
<type>: one of the types listed in Table 2, in lowercase.
<value>: keyword value. See the type description for each value.

Table 2: List of keyword types

Type	Description	Examples
Int	Integer number	23, -400
String	String. Double quotes are needed if the string contains spaces	adc, “adc motor”, 127.0.0.1
Float	Generic floating-point number	0.2, -644.2
float32	Floating point number, limited to 32 bits	
double	Floating point number, double precision	
structure	A sub-configuration file. The keyword value is the filename relative to $ADOPT_ROOT/conf/xxx/current.	

15.2. MsgD configuration file

A special configuration file is reserved for the MsgD-RTDB process. This file always has the name “msgdrtdb.conf” and resides at the same level as the “processConf” directory. An example follows:

```
loglines    2000000
autodump    300

ident    FLAOWFS
peers    ADSEC:192.168.13.12
```

<loglines> is the maximum number of line to write in the MsgD log file before opening a new one.
<autodump> is the interval between information dumps in the log file
<ident> sets this MsgD identity
<peers> is a comma-separated list of <identity>:IPaddress pairs, that specify other MsgD to which this MsgD will try to connect to.

15.2.1. Configuring peering

Message daemons can communicate between them using a custom "peering" mechanism: each MsgD is identified by a string (called "identity"), connects to all other MsgDs and appears as an ordinary client. If one or more MsgD are not reachable, a polling loop at low frequency (one attempt every 5-10 seconds) is started.

Peering is configured with the “ident” and “peers” keywords described above. All MsgDs participating in a peering set must have unique identities.

15.2.2. LBT setup

The following identities has been defined at LBT:

adsec computer: ADSEC
Flao WFS computer: FLAOWFS
LBTI WFS computer: LBTIWFS

The ADSEC MsgD peers with all possible WFSs. Each WFS only peers with the ADSEC MsgD.

At LBT, right and left sides do not peer, that is, they are completely independent from each other.

15.2.3. LTB configuration files

on adsecdx: $ADOPT_ROOT/conf/adsec/current/msgdrtdb.conf

```
ident ADSEC
peers FLAOWFS:10.144.0.85,LBTIWFS:192.168.149.100
```

on wfsdx: $ADOPT_ROOT/conf/wfs/current/msgdrtdb.conf

```
peers ADSEC:192.168.11.12
ident FLAOWFS
```

Similar configuration is used on the left side, with the appropriate IP addresses.

15.2.4. How to check if peering works correctly

The easiest way is to use the "thrdtest" utility to connect to the local MsgD and list the attached clients. The others MsgDs will appear as a client marker with "(Peer)". For example:

```
[flao@wfsdx ~]$ thrdtest

THRDTEST: 6.14 [TH](Built: Sep 10 2015 14:46:04) - Dbg lev.:0, quiet, Line edit & history:Yes
Trying to connect to MsgD @ 127.0.0.1:9752

THRDTEST cmd: clist

THRDTEST MsgD client list:
 M_ADSEC (Peer) [Id=1 N.Conn=1] 11.6 (Built: Sep 10 2015 14:41:52) R NW @=193.206.155.42:9752 Start:2015-09-11 09:24:33.005752
```

15.3. Common keywords

All configuration files can contain the keywords listed in Table 3:

Table 3: List of common keywords

Keyword	type	Description	Notes
Server	string	IP address or hostname of MsgD server	Usually 127.0.0.1
LogLevel	string	General log level, allowed values are:	

		ERR (only log errors) WAR (log errors and warnings) DEB (log errors, warnings and debug) TRA (maximum log level)	
Simulation	int	If present and set to 1, the software will enter simulation mode (if available)	

16. Configuration keywords

The following tables list all the keywords accepted by the various configuration files. All keywords are mandatory unless otherwise noted. If a mandatory keyword is missing, the process will exit and write into its log file the missing keyword. If a file contains keywords not among the ones in the table, they will be silently ignored.

Table 4: List of keyword for SimpleMotorCtrl instances

Keyword	type	Accepted values	Unit	Description	Notes
MotorType	string	filterwheel adc rerotator mercury		Motor type. Correspond to a different C++ class implementing motor functionality.	
Name	string			Motor name. Only used for GUI displays	
IPaddr	string	Ip address or hostname		IP address or hostname to connect to	
IPport	Int			IP port to connect to	
Max	Float32	Any	mm or degrees	Maximum allowed position	
Min	Float32	Any	mm or degrees	Minimum allowed position	Movement commands outside min/max boundaries are rejected.
GoodWindow	Float32	Any	Mm or degrees	Threshold to accept position: if abs(current position – commanded position) <= good window, the device is considered “in position”	
StartingPos	Float32		mm or degrees	Position to move to at startup, (if homing is enabled, this position is assumed after homing has completed).	
Ratio	Float32			Multiplier to apply to translate user commands into encoder increments	

CircleSteps	Float32		Encoder increments	(only for circular commands) number of device increments to perform one cicle	
AutoHoming	Int	0-1		If 1, perform homing whenever the device is turned on or newly connected	
AutoHomingOffset	Int	0-1		If 1, apply an offset position after the homing procedure is completed	
HomingOffset	Float32		Encoder increments	Offset to apply to the position found by the homing routine	
HomingPosition	Float32			Special position: when commanded to this position, start the homing procedure	
AbortPosition	Float32			Special position: when commanded to this position, abort the current movement	The ability of aborting a movement is device-dependent
HomingSpeed	Int		Device units	Speed of homing movement	
CruiseSpeed	Int		Device units	Speed of all other movements	
Acceleration	Int		Device units	Acceleration/deceleration	
Unidirectional	Int		0-1	If set to 1, only move towards positive direction.	
customPositionNum	Int		0-N	Number of custom positions defined.	The definition of custom positions is optional. If this keyword is present and with a non-zero value, all posX_name and posX_pos keywords are mandatory.
posX_name	String			Name of position X (for GUI display)	Allowed values for X are from 0 to customPositionNum minus 1.
posX_pos	Float32		mm or degrees	Definition of position X	

Table 5: Keywords required for SimpleMotorCtrl instances of type "rerotator"

Keyword	type	Accepted values	Unit	Description	Notes
HomingType	String	“ncal” (homing to		Type of homing to perform	The keyword value is the

		negative) “nrm” (homing to positive)			name of the Pollux serial command.
Speed	float		Device units	Movement speed	
Accel	float		Device units	Movement acceleration	
nlimits	int		1 or 2	Number of limit switches	

Table 6: Keywords for the “Pinger” process

Keyword	**type**	**Accepted values**	**Unit**	**Description**	**Notes**
PollingPeriod	Float	0-1000	s	Interval between successive pings to the same device	
TimeoutPeriod	float	0-1000	s	Period after which a device is considered to be offline	Must be less or equal to PollingPeriod
device_XXXX	string	IP or hostname		IP address or hostname of device. The XXX part is the name of the device, and will be used in RTDB variable names. Any number of device_XXXX is allowed.	If a hostname is used, /etc/hosts must contain an entry to resolve it to an IP address

Table 7: Keywords for the “TTCtrl” process

Keyword	**type**	**Accepted values**	**Unit**	**Description**	**Notes**
ccd39process	string			Identity of the ccd39 controlling process. Used to generate the name of the RTDB variable containing the current frame rate.	
pingerProcess	string			Identity of the pinger process. Used to generate the name of the RTDB variable containing the bcu47 online status.	
ACT_NUM	Int	3		Number of tip-tilt mirror actuators.	Only “3” is currently supported!
TIMEOUT_MS	Int	1-Inf	ms	Timeout for MirrorCtrl replies when applying	Includes round-trip via

				values	Ethernet to the BCU47
ZV_TO_XV	double	Any		Ratio between user-coordinates volts and device voltage	Usually 1
MAX_FREQ	Double	0-1000	Hz	Maximum allowed value for user-defined frequency	
MIN_FREQ	Double	0-1000	Hz	Minimum allowed value for user-defined frequency	
MAX_VOLT	Double	0-10	V	Maximum allowed value for user-defined voltage	Used when controlling axes independently
MIN_VOLT	Double	0-10	V	Minimum allowed value for user-defined voltage	Used when controlling axes independently
DEFAULT_LL_FREQ	Double	0-1000	Hz	Default frequency value at startup	
DEFAULT_LL_AMP	Double	0-10	V	Default modulation amplitude value at startup	
DEFAULT_LL_OFFSET	Double	0-10	V	Default voltage offset at startup	
DEFAULT_LL_PHASE_1	Double	0-360	deg	Default phase for first actuator	Usually set at 0,120 or 240 but may be fine-tuned for a specific mirror head.
DEFAULT_LL_PHASE_2	Double	0-360	deg	Default phase for second actuator	
DEFAULT_LL_PHASE_3	double	0-360	deg	Default phase for third actuator	
DEFAULT_ROT_ANG	Double	0-360	deg	Default rotation applied when translating high-level XY positions into low-level commands	
DEFAULT_FREQ	Double	0-1000	Hz	High-level frequency at startup	If 0, uses sync signal from bcu39
DEFAULT_AMP	Double	0-10	V	High-level amp at startup	
DEFAULT_OFFSET_X	double	0-10	V	High-level X offset at startup	
DEFAULT_OFFSET_Y	double	0-10V	V	High-level Y offset at startup	

Table 8: Keywords for the Autogain routine

Keyword	type	Accepted values	Unit	Description	Notes
slopes_skip	Int	1-1000		Number of slope frames to skip after applying a gain value	
slopes_record	Int	1-1000		Number of slope frames to record (after skipping) for each gain value	
removeBadModes	Int	0-1		If 1, read the "high_force_modes.fits" file in the current M2C directory and set to zero the gain on those modes.	If the file does not exists, this keyword is silently ignored.
reduction_factor	Float	any		Apply this factor unconditionally to any generated gain value	Set to 1.0 for no reduction. Usually set to 0.8 or 0.7
interpolateGains	Int	0-1		If 1, interpolate gains between mid and high-order, instead of producing a step-like function	Gain interpolation is experimental.
RR	Int	0-1		If 1, retro-reflector operation is assumed and all gains are reduced by a factor of 2.	
sinusIM	Int	0-1		If 1, a sinusoidal IM is assumed and all gains are reduced by a further factor of 2	
max_iterations	Int	1-10		Maximum number of iterations to perform while exploring higher gain values, if a minimum is not found in the current range.	
repeat_th	Float	0-1		Repeat threshold in percentage (0..1): if the modal RMS at the optimal gain value is higher that the max modal RMS multiplied by this threshold, the measurement is iterated with 50% higher gains.	
safe_skip	Float	0-1		Percentage of safe-skip condition above which the autogain is aborted.	0=no check 1=wait for 100% safe skip before aborting. Check is done once per

					second.
bin1_min	Float	any		Minimum gain value to apply	All bin1 keywords are also repeated for bin2, bin3 and bin4.
bin1_start	Float	Any		Initial start of gain range	
bin1_end	Float	Any		Initial end of gain range	
bin1_step	Float	Any		Step to use while exploring the range	
bin1_cycles	Int	Any		Number of times the range is explored in a single measurement.	Multiple range explorations are averaged.
bin1_max_tt	Float	Any		Clip value for tip-tilt gain	
bin1_max_ho1	Float	Any		Clip value for mid-order gain	
bin1_max_ho1	Float	Any		Clip value for high-order gain	
bin1_ho_middle	Int	3-670		Mode number that divides mid-order modes from high-order modes.	

Table 9: Keywords for the "JoeCtrl" process

Keyword	**type**	**Accepted values**	**Unit**	**Description**	**Notes**
ccdName	string	"39", "47"		CCD name	Only used for GUI displays
ccdNum	Int	39, 47		CCD number.	Number is often used in code.
ccdXdim	Int	1-32767	px	Number of pixels in the X (horizontal) dimension, not binned.	
ccdYdim	Int	1-32767	px	Number of pixels in the Y (vertical) dimension	
ccdDefaultXbin	Int	1-5		Default X binning applied at startup	
ccdDefaultXbin	Int	1-5		Default Y binning applied at startup	
ccdDefaultSpeed	Int	any	Kpixel/sec	Default readout speed applied at startup	
ccdDefaultBlack	Int	0-1023	Arbitrary	Default black level applied at startup	
ccdBlacksNum	Int	2, 4		Number of blacks levels to configure: 4 for ccd39 and 2 for ccd47	Corresponds to the number of amplifier channels.
minRep	Int	0		Minimum allowed value for the "repetition" parameter	

maxRep	Int	1023		Maximum allowed value for the “repetition” parameter	
maxNumSpeeds	Int	1-8		Number of supported readout speeds	“supported” by the JoeCtrl process, not the LittleJoe hardware
maxNumBins	Int	1-8		Number of supported binning configuration	“supported” by the JoeCtrl process, not the LittleJoe hardware
num_programsets	Int	1-8		Number of programsets stored on disk	
programsetX	Structure			Filename of the configuration file describing program set #X	
startProgramSet	Int	-1 – up to the value of “num_ programsets”		Number of the programset to load at startup. A value of -1 means to load no programset.	
fanReqVar	String	Variable name		Name of the RTDB variable used to control the LittleJoe fan	It is assumed that setting this variable to “1” will start the fan, and to “0” will stop it.
fanCtrlActive	Int	0-1		If 1, try to control the fan using the previous variable. If 0, ignore it.	A limitation of the current FLAO hardware is that a single hardware switch controls both the LJ39 and LJ47 fans. Only one control loop should be active.
fanOnTemp	Int	Any	°C	Temperature over which the fan is turned on	
fanOffTemp	Int	Any	°C	Temperature under which the fan is turned off	

Table 10: Keywords for the “Gopt” (optical gain) process

Keyword	type	Accepted values	Unit	Description	Notes
gopt	Float	Any		Initial optical gain vaule, loaded at startup on the BCU	Usually 1.0
nframes	Int	1-10000		Number of frames to record for the optical	Initial value, may be changed from the GUI

				gain analysis	
mode	int	0-671		Mode to use for optical gain analysis	Deprecated, now automatically detected by the elab_lib
freq	Float	1-200	Hz	Modulation frequency for optical gain analysis	Deprecated, now automatically detected by the elab_lib
trackGain	Float	Any		Gain of the tracking loop	Initial value, may be changed from the GUI
delay	Float	Any	s	Delay between adjusting the optical gain and saving new data	

Table 11: Keywords accepted by WfsArb instances

Name	Type	Accepted values	Unit	Notes
OP_MODES	String array	Any		Defines the list of "operating modes" available to the WFS. These operating modes are described in $ADOPT_SOURCE/PyModules/AdOpt/cfg_W1.py.
WfsSpec	string	FLAOWFS, LBTIWFS		Defines the WFS name, should match the one used for the current MsgD.
MinLoopFreq	Float	1-1000.0		Minimum loop frequency accepted in the ModifyAO command
MaxLoopFreq	Float	1-1000.0		Maximum loop frequency accepted in the ModifyAO command
MaxOvsFreq	Float	1-1000.0		Maximum oversampling frequency accepted by the Adaptive Secondar (see [])
MinHODark	int	1-1000		Minimum no. of frames to average when taking a dark frame with ccd39
MaxHODark	Int	1-1000		Maximum no. of frames to average when taking a dark frame with ccd39
MinTVDark	Int	1-100		Minimum no. of frames to average when taking a dark frame with ccd47
MaxTVDark	Int	1-100		Maximum no. of frames to average when taking a dark frame with ccd47

MinIRTCDark	Int	1-1000		Minimum no. of frames to average when taking a dark frame with IRTC
MaxIRTCDark	Int	1-1000		Maximum no. of frames to average when taking a dark frame with IRTC
MinSlopenull	Int	1-1000		Minimum no. of frames to average when taking a slopenull frame.
MaxSlopenull	Int	1-1000		Maximum no. of frames to average when taking a slopenull frame.
MaxOffsetXYCloop	Float	0.6	mm	Maximum accepted offset in closed loop.
INITIAL STATE	String	State name		State from which the FSM is initialized. Usually “PowerOff” or “Operating”
TelElevationVar	String	Variable name		Name of the RTDB variable which contains the current telescope elevation.
TelRotatorVar	String	Variable name		Name of the RTDB variable which contains the current telescope derotator position.
RerotVar	String	Variable name		Name of the RTDB variable to write in order to move the pupil rerotator.
Adc1Var	String	Variable name		Name of the RTDB variable to write in order to move the adc wheel #1.
Adc2Var	String	Variable name		Name of the RTDB variable to write in order to move the adc wheel #2
RotatorOffsetBin1	Float	0-360.0	degrees	Offset to apply to the pupil rerotator at bin1.
RotatorOffsetBin2	Float	0-360.0	degrees	Offset to apply to the pupil rerotator at bin2.
RotatorOffsetBin3	Float	0-360.0	degrees	Offset to apply to the pupil rerotator at bin3.
RotatorOffsetBin4	Float	0-360.0	degrees	Offset to apply to the pupil rerotator at bin4.
cameralensTempCheck	Int	0 – 1		Activate(1) or deactivate (0) the cameralens temperature check

cameralensTempMin	Float	-50/+50	°C	Minimum temperature required for cameralens operation
cameralensTempNumber	Int	0-9		Position in the powerboard temperature array where the relevant cameralens temperature is found.

Table 12: Keywords accepted by CopleyCtrl instances

Name	Type	Accepted values	Unit	Notes
iDriveNetAddr	String	Ip address or hostname		Host to connect to.
iDriveNetPort	Int	1-65536		Port to connect to
mvSpeed	Float	1-100	Mm/s	Stage movement speed
mvHighEnd	Float	any	Mm	Maximum accepted target position
mvLowEnd	Float	Any	Mm	Minimum accepted target position
stepsRatio	Float	Any		Ratio between encoder steps and mm.
HomingPosition	Float	Any	Mm	Target position to trigger a homing sequence
AbortPosition	Float	Any	Mm	Target position to trigger an abort sequence
GoodWindow	Float	Any	Mm	Accepted positioning error
proportionalGain	Int	1-9999		Proportional gain of the internal PID loop. Defaults to 4500.
positiveLimitSwitch	Int	0-9		I/O line to use as positive limit switch. Zero means not used.
negativeLimitSwitch	Int	0-9		I/O line to use as negative limit switch. Zero means not used.

homeLimitSwitch	Int	0-9		I/O line to use as home switch. Zero means not used.
HomingMethod	String	POS or NEG		Use either the Positive or Negative limit switch to perform homing.

17. Table of wfs, adsec and AO status values and commands accepted

17.1. Wfs command table

State	Accepted commands
PowerOff	Operate
Operating	Off Operate SaveStatus SaveOptLoopData AntiDrift EnableDisturb AutoTrack CalibrateHODark CalibrateTVDark CalibrateIRTCDark StopLoop PrepareAcquireRef ModifyAO
AOPrepared	Operate PrepareAcquireRef AcquireRef StopLoop AntiDrift AutoTrack CalibrateHODark SaveOptLoopData SaveStatus CheckRef ModifyAO Off
AOSet	CloseLoop ModifyAO AcquireRef Operate PrepareAcquireRef StopLoop AutoTrack AntiDrift EnableDisturb

	SaveOptLoopData SaveStatus CalibrateHODark CalibrateTVDark CheckRef
Failure	RecoverFailure (Operate)
LoopClosed	PauseLoop StopLoop RefineLoop OffsetXY OffsetZ AntiDrift AutoTrack EnableDisturb SaveOptLoopData SaveStatus
LoopPaused	ResumeLoop StopLoop OffsetXY OffsetZ AntiDrift AutoTrack EnableDisturb SaveOptLoopData SaveStatus

17.2. AdSec Command table

17.3. AO Command table