MRO FTT/NAS & FLC

Software Release Notes

MRO-MAN-CAM-1160-0163

John Young <jsy1001@cam.ac.uk>

rev 1.5

23 February 2016

Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE
UK

Change Record

Revision	Date	Author(s)	Changes
0.1	2013-12-23	JSY	Initial version
0.2	2014-01-06	JSY	Added Thread API memory leak issue
0.3	2014-01-07	JSY	Small updates
1.0	2014-01-08	JSY	Corrections from EBS
1.1	2014-01-13	JSY	Expanded build instructions, updated issue list
1.2	2015-11-18	EBS	Documented separation of environmental controller onto embedded computer, added ssh forced command implementation, removed known issues that are now fixed, many minor edits to bring document up to date. Bumped release number to 2.
1.3	2015-11-19	JSY	Documented installation of the Python Analysis GUI; minor edits.
1.4	2016-01-07	JSY	Updated implementation status and known issues.
1.5	2016-02-23	EBS	Updated known issues for release 2.

Objective

To describe the FTT/NAS software release.

Scope

This document outlines the FTT/NAS software development status, describes the content of the latest release, and explains how to build and test the software.

Reference Documents

RD1 FTT/NAS to ISS ICD (MRO-ICD-CAM-1100-0112) rev 1.13, November 13th 2015

RD2 FLC to ISS ICD (MRO-ICD-CAM-1200-0113) rev 1.13, November 13th 2015

RD3 FTT/NAS Software Preliminary Design Report (MRO-TRE-CAM-1160-0143) – rev 1.0, April 30th 2012

Acronyms and Abbreviations

API	Applications Programming Interface
EMCCD	Electron Multiplying Charge Coupled Device
FTT	Fast Tip-Tilt
FLC	First Light Camera
GSI	Generic System Interface
ICD	Interface Control Document
ISS	Interferometer Supervisory System
MROI	Magdalena Ridge Observatory Interferometer
NAS	Narrow-field Acquisition System
NMT	New Mexico Tech
SDK	Software Development Kit
SSH	Secure Shell
TBC	To be confirmed
TBD	To be determined

Table of Contents

1 Introduction

These notes describe release 2 of the MROI FTT/NAS software, and should be read in conjunction with the Interface Control Documents (RD1, RD2). The overall architecture of the software is described in the PDR report (RD3).

2 Release Content

The release comprises source code and makefiles for the system controller, environment controller, FLC and FTT/NAS editions of the control GUI, and the preliminary Analysis GUI coded in Python. Source code for the MRO-supplied Generic System Interface software framework (version 1.8) is included, and gets built as part of the FTT/NAS software build process.

The supplied software depends on the Andor Software Development Kit (SDK). This is needed to build the software, even if the intention is to run it configured to emulate a real EMCCD camera. We recommend that the SDK is purchased from Andor before attempting to build the software, though it is possible to build a subset without it.

The Matlab Analysis GUI is *not included* in this release. The user manual for the software exists only as an incomplete draft.

The following system variants (RD1) are *not included* in this release:

FTTCamSystem, FTTEnvSystem Require a network-capable GSI release to initialize and control the system

FTTCamSystem_ucamcontrol, FTTEnvSystem_ucamcontrol Require a network-capable GSI release to initialize the system and enable the control GUI interface

FTTCamSystem_ucamdisplay, FTTEnvSystem_ucamdisplay Require a network-capable GSI release to initialize the system and enable the display-only control GUI interface

test_FTTCamSystem, test_FTTEnvSystem We have opted not to use standalone test scripts, as the scripting environment does not support image-type command parameters

3 Implementation Status

The system and environment controllers can be configured to emulate the real FTT/NAS hardware (EMCCD camera and camera thermal enclosure), hence the software can be tested without connecting these devices. We have also implemented all of the functionality with real FLC hardware needed for UT commissioning, but this has not been specifically tested for this release.

The ISS interfaces for the system (`FTTCamSystem`) and environment (`FTTEnvSystem`) controllers have been fully implemented, including all of the commands, monitor points and system properties enumerated in RD1. We have tested that the system and environment controllers can be started and initialized in standalone (i.e. without a network interface to the ISS) mode using emulated or real hardware, and that all of the commands defined in the interface spreadsheet can be executed. We have tested publishing of the defined monitor points within the standalone execution environment.

The environmental controller has been moved to an embedded processor (a Raspberry Pi 2) within the existing environmental electronics box in the control rack. In adverse environmental conditions, this allows it to maintain environmental awareness while completely removing power from the FTT camera in an orderly fashion: it shuts down the FTT software, then the FTT computer, then cuts power to the FTT computer and the camera's Peltier cooler via instructions to the rack's power board.

It is still possible to run the environmental controller on the same computer as the other software components, but this is no longer a recommended configuration when real hardware is used.

The FTT/NAS system controller does not subscribe to any UTCS monitor points. Hence coordinate rotations have not been implemented and dispersion/off-axis offsets are not applied to the tip-tilt zero point. These capabilities will be implemented once a test environment for the ISS publish-subscribe system is made available.

4 Build Requirements

- A computer with an X86 architecture running Xenomai. The software release has been successfully built and run on the following 64-bit development computers in Cambridge:

  1. Debian amd64 testing ("stretch"), 3.14.17 kernel, Xenomai 2.6.4 patches.
  2. Ubuntu amd64 14.04.3 LTS ("Trusty Tahr"), 3.14.17 kernel, Xenomai 2.6.4 patches
  3. Ubuntu amd64 14.04.3 LTS ("Trusty Tahr"), 3.16.7 kernel, Xenomai 2.6.4 patches

  Additionally, the environmental control software has been successfully built and run on a Raspberry Pi 2 with its default operating system of Raspbian Linux (Debian "wheezy"), 4.1.7 SMP PREEMPT kernel. There is no need to install Xenomai, the documentation tools, the driver building tools, the second set of python packages, the Andor SDK or fv (all mentioned below) to build and run the environmental controller on this device.

  There is some flexibility with respect to choice of kernel, but only Xenomai 2.6.4 is known to work. Older Xenomai versions have been found to behave unpredictably when the software is run within the gdb debugger, and the recent 3.0 release has not yet been tested.

  Installing Xenomai (`http://www.xenomai.org`) involves patching Linux kernel source and then compiling and installing the resulting kernel. Instructions are beyond the scope of this document.

- The Gnu Compiler Collection (GCC). The versions used for development were 4.6.3, 4.8.4 and 5.2.1.
- Gnu Make. The versions used for development were 3.81 and 4.0.
- The subversion client software. This is needed because some of the makefiles invoke the svnversion utility.
- A Java Development Kit. This is needed to build the Generic System Interface framework.
- Driver building tools: linux-headers and linux-kbuild for the Linux kernel you're running. If you were able to compile and install Xenomai, you already have these.
- Python 2.6 or 2.7 (recommended). This is used for the preliminary Analysis GUI and to generate some test code from the system interface spreadsheets. A number of third-party packages are also required. The following are needed to build the system and environment controllers:

  – python-dev Debian package (needed for pip to build packages from source)

  – setuptools (install python-setuptools Debian package or follow the instructions at `http://pypi.python.org/pypi/setuptools`)

  - odfpy 0.9.x or 1.3.x (python-odf Debian package, or run `pip install odfpy`)
  - numpy (python-numpy Debian package, or `pip install numpy`)
  - astropy 1.0.2 or later (python-astropy Debian package, or `pip install --no-deps astropy`)

  To run the Analysis GUI, you will also need to install the following:

  - pygtk (python-gtk2 Debian package; note this cannot be installed using pip on Linux)
  - scipy (python-scipy Debian package, or `pip install scipy`)
  - matplotlib (python-matplotlib Debian package, or `pip install matplotlib`)
  - python-dateutil (python-dateutil Debian package, or `pip install python-dateutil`)

- The appropriate development libraries. On a Debian system, the following packages and their dependencies were used to compile the source code:
  - libgtk2.0-dev
  - libglib2.0-dev
  - libgtkimageview-dev
  - libcfitsio3-dev
  - libxenomai-dev
  - libc6-dev
  - libusb-dev
  - libgsl0-dev
  - libsnmp-dev

  Package names might vary for Linux distributions other than Debian and Ubuntu.

- Documentation building tools. On a recent Debian system, the following packages and their dependencies are required:
  - texlive-latex-base, texlive-latex-extra, texlive-bibtex-extra, biber, latexmk
  - doxygen
  - graphviz
- The Andor SDK (Linux edition V2.94.30009.0 or later). This must be purchased from Andor.
- The LabJack Linux software library and driver (Exodriver) for U3 devices (`http://labjack.com/support/software`) and its dependencies (libusb-1.0-0-dev)
- fv (`http://heasarc.gsfc.nasa.gov/ftools/fv/`) is useful for viewing FITS files generated by the control GUI.

5 Building

Boot Linux, ensuring you select your Xenomai-patched kernel. It is necessary to boot the kernel you will be running the software under because the device driver makefiles use `uname`.

Change to the directory you want the source code to appear in, then type:

```
$ tar xf [path-to-source-tarball.tar.gz]
```

Change to the top level ftt directory, e.g.:

```
$ cd ftt-release1-svn1111
```

Note that the top-level directory name in the tarball reflects the release version.

Build the software and documentation:

```
$ make
```

Should you need to delete the executables and object files generated by `make`, run `make clean`. To remove the documentation as well, use `make distclean`.

5.1 Building without the Andor SDK

A subset of the software can be built without the Andor SDK. Replace the `make` command above with the following:

```
$ make systems/FTTEnvSystem
```

```
$ make controlgui
```

5.2 Building system variants

It is possible to build only the system executables corresponding to a particular interface spreadsheet. For example:

```
$ cd ftt-release1-svn1111
```

Build the libraries and drivers:

```
$ make ext/camlibs ext/build_gsi fttlibs systems/drivers
```

Build the system executables based on the `FTTCamSystem_ucamcontrol` spreadsheet:

```
$ cd systems/FTTCamSystem
```

```
$ make build_ucamcontrol
```

6 Installation

After building the software, from the top level ftt directory, acquire superuser privileges and type

```
$ make install
```

Executables including scripts get installed in `/usr/local/bin`. Configuration files for the control GUI applictaions get installed in `/usr/local/share/fttgui`, `/usr/local/share/flcgui` and `/usr/local/share/enggui`. Documentation, including source code documentation, gets installed in `/usr/local/share/doc/ftt`.

6.1 Driver installation

If emulating real FLC or FTT/NAS hardware, or only building the environmental controller, there is no need to install the drivers.

6.2 NTP

We recommend that you run NTP clients on the computers that run the system applications and the computer that runs the control GUI. This will ensure that published monitor data have accurate timestamps, and that FITS logfiles written by the GUI have correct timestamps in their filenames. To check if NTP is running you can print a list of active time servers with

```
$ ntpq -p
```

6.3 Running the camera controller as an ordinary user

By default, all Xenomai programs must be run as root. To allow an ordinary user to run the camera controller without superuser privileges, make the changes listed below on the FTT computer. The instructions in later sections of this document assume that this has been done, and that hence FTTCamSystem can be run without acquiring superuser privileges.

1. Look up the group ID of the xenomai group:

   ```
   getent group xenomai
   ```

   The group ID is the number in the result, for example, “110”.

2. As root, edit `/etc/default/grub` to tell the Xenomai kernel about the group ID using the `GRUB_CMDLINE_LINUX_DEFAULT` parameter, for example,

   ```
   GRUB_CMDLINE_LINUX_DEFAULT="quiet xeno_nucleus.xenomai_gid=110"
   ```

3. As root, commit the change for the next system boot:

   ```
   update-grub
   ```

4. As root, add the user to the Xenomai group. If the user is fttuser:

   ```
   adduser fttuser xenomai
   ```

5. Reboot the system.

6.4 SSH forced command integration

The environmental controller must have the ability to shut down the FTT computer during adverse environmental conditions, even when the ISS is not available (for example, during a network failure outside the control rack).

This ability is implemented with an SSH forced command. The environmental controller attempts an SSH connection with a user account on the FTT computer. Normally, the account would respond with a login request, but instead is configured to run the shutdown command.

As this procedure requires passwordless access to a remote administration function, security measures are required: the user account has no login, cannot launch a shell, and can only run shutdown when an an SSH attempt is made from a computer with the environmental controller's internet protocol address.

Here is the procedure used to add this ability:

1. Using the environmental controller, generate an SSH key pair so that the environmental controller can log in to the FTT computer without a password:

   `ssh-keygen` (no passphrase when prompted)

2. Create a user account on the FTT computer especially for this purpose (say, "shutdownuser"). As root:

   ```
   adduser --disabled-password shutdownuser
   ```

   As it's not possible to log in as shutdownuser, you will need to edit files in shutdownuser's account as root, then change their ownership and group to shutdownuser.

3. Take the public key from the SSH key pair (the one in the file with a ".pub" suffix) and add it to shutdownuser's `.ssh/authorised_keys` file.

4. Take the private key file from the SSH key pair and place it in the environmental controller user's `.ssh` directory (this is the default location suggested by ssh-keygen).

5. Edit the `authorised_keys` file in shutdownuser's account to add a forced command. For example, if the line containing the public key begins with "ssh-rsa A...", put some text at the start of the line:

   ```
   from="ip.addr.of.env.controller",no-port-forwarding,no-X11-forwarding,
   no-agent-forwarding,no-pty,command="echo \"sudo /sbin/poweroff\" |
   /usr/bin/at now + 2 min >/dev/null 2>&1" ssh-rsa A...
   ```

   Substitute the actual IP address of the environmental controller for the placeholder text. It is recommended that the numerical IP address is used, so that the system will work even if there is a network failure to the default name server. Also, the line breaks have been added for clarity, don't type them.

   This says that when shutdownuser on the FTT computer receives an SSH attempt from the environmental controller, they should queue a poweroff command two minutes from the start of the current minute, and return immediately.

   The rest of the line is designed to prevent the environmental controller from gaining administrative privileges on the FTT computer.

6. On the FTT computer, use the `visudo` command as root to add the following line to `/etc/sudoers`:

   ```
   # shutdownuser can only shut down the computer

   shutdownuser    ALL=(ALL) NOPASSWD: /sbin/poweroff
   ```

   The line breaks are real this time. This allows shutdownuser to shut down the computer without a password.

7. Finally, remove shutdownuser from the sudo group, so that they can't do anything else with administration privileges. As root:

```
deluser shutdownuser sudo
```

8. As a final check, on the environmental controller issue the command

   ```
   ssh shutdownuser@ip.addr.of.ftt.computer
   ```

   (Substitute the actual IP address of the FTT computer for the placeholder text above). The FTT computer should shut down between one and two minutes later, depending on when in the current minute the command was issued.

6.5 Analysis GUI

To install the analysis tools and their prerequisite python packages (if not already installed), run the following commands:

```
$ cd analysisgui
```

```
$ python setup.py build
```

then acquire superuser privileges and type

```
$ python setup.py install
```

The tool for plotting time series and power spectra from FITS log files recorded by the control gui (either edition) is called `plotfttgui`. To run it, simply type:

```
$ plotfttgui
```

7 Orientation

In this and subsequent sections `[ftt-root]` stands for the top level directory of the source tree (as distributed – the corresponding directory in the Cambridge subversion repository is `trunk/mroi-ftt`), e.g. `ftt-release1-svn503`.

`[ftt-root]/analysisgui` contains analysis tools written in Python.

`[ftt-root]/controlgui` contains the control GUI (both the FLC and FTT/NAS editions).

`[ftt-root]/doc` contains the documentation (at the moment just this document and the API reference documentation).

`[ftt-root]/ext/build_gsi` contains a Makefile to build the Generic System Interface framework. This is used to integrate the framework into the FTT software build system.

`[ftt-root]/ext/camlibs` contains libraries shared with other Cambridge software projects.

`[ftt-root]/ext/gsi` contains the Generic System Interface framework itself.

`[ftt-root]/fttlibs` contains libraries specific to this project.

`[ftt-root]/systems/FTTCamSystem` contains the FTT/NAS system controller (all variants).

`[ftt-root]/systems/FTTEnvSystem` contains the FTT/NAS environment controller (all variants).

8 Testing

To run the entire suite of unit tests, simply run `make check` from the top level ftt directory. This can be done prior to installation.

The unit tests for the system controller and environment controller are likely to be of particular interest. These exercise all of the defined commands in standalone mode, by calling the command methods directly from C code. To run these separately from the other unit tests, type the following commands:

```
$ cd systems/FTTEnvSystem
```

```
$ bin/utest_FTTEnvSystem
```

and:

```
$ cd systems/FTTCamSystem
```

```
$ bin/utest_FTTCamSystem
```

There are further unit test executables `utest_*_ucamcontrol` which exercise the commands using the control GUI network interface over loopback sockets. The scripts `testdata_FTTCamSystem.py` and `testdata_FTTEnvSystem.py` run the appropriate system in order to collect data in a local file, then perform various checks on the file contents.

9 Control GUI

The control GUI is a software application that provides an operator interface to the system controller and environment controller systems. Images and scalar monitor data published by the systems are displayed to the operator, and these displays update automatically as new data arrive. The application can record the images and scalar data to a set of FITS files when requested to do so by the operator. The resulting files can be read by `plotfttgui`. If the system and environment controllers are not being directed by an ISS supervisor, the control GUI applicaion is also used to control the FTT system.

There are two editions of the GUI:

FTT/NAS GUI Provides access to the full functionality of the FTT/NAS system, including the fast tip-tilt correction mode.

FLC GUI Provides access to the FLC functionality as defined in RD2. Compared with the FTT/NAS GUI, has additional data handling capabilities requested by AMOS for the purpose of UTM commissioning:

- Ability to record scalar data (e.g. unaveraged centroid estimates) to CSV files as well as FITS files.
- Scrolling display of recent average and rms centroid estimates.

The two GUI editions are almost identical in look and feel and share much of their code. They may be used interchangably with the system and environment controllers. The intention is that the FLC GUI be used with the FLC system for integration and commissioning of the first UTM. We expect that subsequent UTMs will be commissioned using FTT/NAS system hardware and the FLC GUI software, with the FTT/NAS GUI used to test the fast tip-tilt loop and for subsequent interferometer operations.

9.1 Starting the FLC GUI

The FLC GUI requires a configuration file and several widget definition files, which are installed in `/usr/local/share/flcgui`. The application complies with the XDG Base Directory Specification, and will search for these files firstly in `~/.local/share/flcgui` and then `/usr/local/share/flcgui`. Thus a convenient method for modifying the configuration is to place a customised copy in `~/.local/share/flcgui`. Alternatively, the location of a configuration file may be specified with a command line option.

The GUI can be configured to prevent changes to the system state, by changing the relevant "Control" value to "false" in the configuration file. This is in addition to any restriction enforced by the system server. The configuration file also contains the address and ports on which the system and environment controllers are assumed to be listening for connections. The default addresses are "localhost".

Once the software has been installed and the configuration has been set appropriately, starting the GUI is simply a matter of:

```
$ flcgui
```

Or to specify an alternative configuration file:

```
$ flcgui -i /path/to/file.ini
```

9.2 Starting the systems

To run the standalone-mode FTTCamSystem and FTTEnvSystem variants which can be controlled by the FLC GUI, run the following commands (in separate terminals):

```
$ start_FTTCamSystem_ucamcontrol /path/to/property.txt
```

```
$ start_FTTEnvSystem_ucamcontrol /path/to/property.txt
```

Again, FTTCamSystem must be run as root. Pathnames of ASCII files containing the system property values must be specified on the command line as shown above. Example `property.txt` files can be found in the source tree at `[ftt-root]/systems/FTTCamSystem/test` and `[ftt-root]/systems/FTTEnvSystem/test` These are not installed by `make install`.

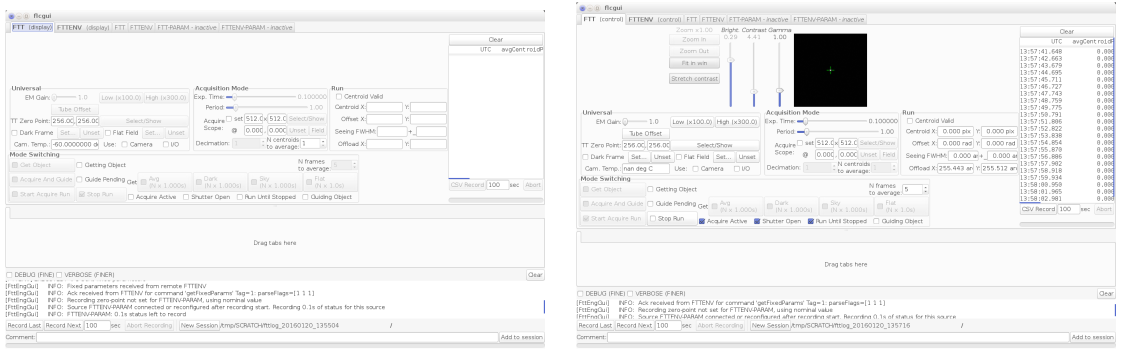
9.3 GUI functionality

The released FLC GUI is shown in Figure 1. The FTT/NAS GUI is very similar in appearance. The main window includes the following regions (differences between the FLC and FTT/NAS GUI are *highlighted* in this description):

Upper notebook User-selectable tabs for system controller ("FTT (control)" or "FTT (display)"), environment controller ("FTTENV (control)" or "FTTENV (display)"), and generic text displays of scalar status items (remaining tabs). Within the main system controller tab, related control and display widgets are grouped together as follows:

Image display Greyscale display of latest (decimated) acquisition or fast tip-tilt mode image, with zoom and pan controls and adjustable mapping from pixel values to displayed grey level. Flat-field and dark frame images are displayed in a separate tab after they are acquired.

Text display of recent centroid estimates *(FLC GUI only)* This is located at the far right of the tab, and includes buttons for saving the data to an ASCII file (either as displayed or as comma-separated values).



```
[FttEngGui]    INFO:  Fixed parameters received from remote FTTENV
[FttEngGui]    INFO:  Ack received from FTTENV for command 'getFixedParams' Tag=1: parseFlags=[1 1 1]
[FttEngGui]    INFO:  Recording zero-point not set for FTTENV-PARAM, using nominal value
[FttEngGui]    INFO:  Source FTTENV-PARAM connected or reconfigured after recording start. Recording 0.1s of status for this source
[FttEngGui]    INFO:  FTTENV-PARAM: 0.1s status left to record
```

Record Last | Record Next 100 sec | Abort Recording | New Session /tmp/SCRATCH/fttlog_20160120_135504

```
[FttEngGui]    INFO:  Ack received from FTTENV for command 'getFixedParams' Tag=1: parseFlags=[1 1 1]
[FttEngGui]    INFO:  Recording zero-point not set for FTTENV-PARAM, using nominal value
[FttEngGui]    INFO:  Source FTTENV-PARAM connected or reconfigured after recording start. Recording 0.1s of status for this source
```

Record Last | Record Next 100 sec | Abort Recording | New Session /tmp/SCRATCH/fttlog_20160120_135716

Figure 1: Screenshots of the FLC control GUI. Two views of the application main window are shown: on the left the GUI has been configured for display-only mode (note that most of the buttons are greyed-out) and the system controller is in idle mode; on the right the GUI is in standalone mode (buttons active) and the system controller is in acquisition mode. We plan to further optimize the layout of the GUI in order to maximise the area available for image display and to ensure the windows fit on a laptop screen. This may involve moving some controls to pop-up dialog boxes or menus.

Universal Widgets for displaying/setting universal parameters e.g. electron-multiplying gain and flat-field and dark frame corrections.

Acquisition Mode Widgets for displaying/setting acquisition-mode-specific parameters e.g. exposure time, frame period, decimation factor.

FTT Mode *(FTT/NAS GUI only)* Widgets for displaying/setting fast-tip-tilt-mode-specific parameters e.g. frame period, servo parameters, decimation factor

Mode Switching Widgets for displaying/setting the current operational mode.

Lower notebook Any of the tabs from the upper notebook may be dragged here so that two displays may be viewed simultaneously

Log message display Scrolling display of human-readable log messages. Checkbuttons toggle whether verbose messages are displayed.

Recording controls Controls to start and stop recording of data in FITS format. See below for more details.

Exception and fault messages are shown in a separate pop-up dialog box.

The widgets on the main environment controller tab are not shown in the figure but include displays of the temperature and humidity sensor readings and widgets for:

- Setting the heater power control to manual or automatic, and the power level if manual.
- Setting camera enable to manual or automatic, and disabling the camera if manual.

The buttons and other widgets on the GUI have tooltips giving brief help text. Further details on the commands executed by the buttons can be found in RD1.

9.4 Data Recording

Monitor data are transmitted continuously to the GUI application using the dlmsg protocols. These data are inserted into a circular buffer on arrival, so that the most recent 100 seconds of data (the amount can be changed at compile time) are available most of the time. Log and fault messages are always written continuously to FITS files, but recording of other kinds of data must be initiated by the user. All kinds of data are grouped into **sessions**, which are intended to group a related set of recordings (such as those from a sequence of tests) made during the same 24 hour period. A session consists of a directory containing multiple FITS-format files (this is for efficiency when writing the data). Within a session, data are grouped into **recordings**, each of which is initiated by the user and spans a continuous time period. Starting the GUI application automatically initializes a new session. The user may close the session and begin a new one by clicking a button on the GUI.

The FITS files use the Heirarchical Grouping Convention (HGC) to describe the relationship between the various FITS files in the session directory. In particular the recording application writes an index file which is subsequently read by the analysis GUI in order to discover the recordings belonging to the session and the files that comprise each recording.

The following recording functions are available:

Record Last Record the most recent N seconds of data to the current session.

Record Next Record the next N seconds of data to the current session.

Abort Recording Stop the current recording operation.

New Session Close the current session and start a new one.

Add (comment) to session : Append a user-specified comment string to the header of the current session group table.

10 Known Issues

1. The MROI Thread C API creates threads with the `PTHREAD_CREATE_JOINABLE` attribute set, but none of the functions in the API call `pthread_join()` or allow the joinable attribute to be unset. We have demonstrated that failing to call `pthread_join()` results in a small memory leak (even once the function running in the thread has returned). The application could work around this issue by making its own calls to `pthread_join()` (we have used this workaround in one case, accessing the `threadId` member of the `Thread` struct), or by unsetting the joinable attribute prior to starting the thread.
2. FTTCamSystem and FTTEnvSystem start publishing monitor data when they are initialized, whereas the standard behaviour (enforced by the GSI for polled monitor points) is to defer monitor publishing until the system is in the operational state.
3. The `setTipTiltBandwidth` command does not have the correct functionality.
4. Remote power socket switching does not work with the Eaton IPC3401-NET power distribution unit, which apparently does not support SNMP.
5. The algorithm used to calculate the `CoherenceTime` (temporal seeing estimate) monitor point hasn't been tested thoroughly and may be changed in a future release.
6. The Kalman filter has not been tested as part of the fast tip-tilt correction laboratory tests.
7. The FTT computer lacks sufficient resources to publish full acquisition frames at a rate greater than about 1Hz, whether real or emulated camera hardware is used (the maximum camera frame rate is about 30Hz). This is likely to be a software issue rather than an intrinsic limitation of the computer hardware.