MRO FTT/NAS & FLC

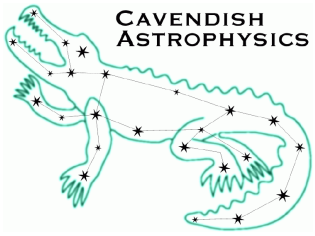
FTT/NAS Software Preliminary Design Report

MRO-TRE-CAM-1160-0143

The Cambridge FTT Team

rev 1.0

30 April 2012



Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE
UK

Change Record

Revision	Date	Author(s)	Changes
0.1	2012-03-26	JSY	Outline
0.2	2012-04-27	EBS	Draft
0.3	2012-04-29	EBS	Extensive modifications suggested by JSY and MF
0.4	2012-04-30	EBS	Revised figure captions, incorporated suggestions from JSY and MF
0.5	2012-04-30	EBS	Added objective and scope, merged suggestions from JSY, DFB and MF
0.6	2012-04-30	EBS	Minor changes suggested by JSY and MF
1.0	2012-04-30	EBS	First release

Objective

To present the preliminary design of the Fast Tip-Tilt / Narrow-field Acquisition System (FTT/NAS) software.

Scope

This document describes those aspects of the FTT/NAS design that are implemented in software. It also covers software communication with the FTT/NAS hardware and the Interferometer Supervisory System. Other aspects of the FTT/NAS hardware are discussed in the partner document RD4.

Reference Documents

RD1 Technical Requirements: Fast Tip-Tilt/Narrow-field Acquisition System (INT-403-ENG-0003) – rev 2.2, May 20th 2010

RD2 Technical Requirements: First Light Camera (INT-403-TSP-0107) – rev 1.0, May 20th 2010

RD3 Derived Requirements: MRO FTT/NAS & FLC (MRO-TRE-CAM-0000-0101) – rev 1.0, August 31st 2010

RD4 FTT/NAS Preliminary Design Report (MRO-TRE-CAM-1100-0142) – rev 1.2, April 30th 2012

Acronyms and Abbreviations

AMD Advanced Micro Devices

AMOS Advanced Mechanical and Optical Systems

API Application Programming Interface

CCD Charge Coupled Device

CSV Comma-Separated Values

DMA Direct Memory Access

EMCCD Electron Multiplying CCD

FITS Flexible Image Transport System

FLC First Light Camera

FTT Fast Tip-Tilt

FTTA Fast Tip-Tilt Actuator

FWHM Full Width Half Maximum

GUI Graphical User Interface

I/O Input/Output

ISS Interferometer Supervisory System

I^2C Inter-Integrated Circuit

MROI Magdalena Ridge Observatory Interferometer

NAS Narrow-field Acquisition System

NMT New Mexico Tech

NTP Network Time Protocol

PCI Peripheral Component Interconnect

PDR Preliminary Design Review

PID Proportional Integral Derivative

RMS Root Mean Square

ROI Region Of Interest

TBC To be confirmed

TBD To be determined

UT Unit Telescope

UTM Unit Telescope Mount

Table of Contents

1 Introduction

This document describes the design and functionality of the FTT/NAS software to be delivered by Cambridge, including the control software, user interfaces, and offline data analysis software. The design of the FTT/NAS system hardware is detailed in a separate report (RD4).

During the PDR phase, the design of the software has been developed from a conceptual level to a detailed design. The software has been partially implemented, and the challenging and unfamiliar aspects of the design have been successfully prototyped.

The prototyping work has been largely carried out by way of implementing the First Light Camera (FLC) control software, for which we chose an identical architecture (including real-time processing of camera data even though this was not needed to meet the FLC requirements) to the FTT/NAS software. The FLC software implements a subset of the FTT/NAS functionality, the main omission being the fast tip-tilt correction mode. This document describes the FTT/NAS software but also identifies the particular features omitted from the FLC software. Additional prototyping of the fast tip-tilt mode, including measurement of the real-time performance and algorithm prototyping using a simulation code, has been carried out as a parallel activity.

The FTT/NAS software has been designed to satisfy the following high-level use cases:

- Maintain a safe environment for the EMCCD camera and prevent the camera from being operated in adverse conditions.
- Provide functionality for science operations: in particular automated and manual target acquisition and fast correction of atmospheric wavefront tip-tilt perturbations using the UTM actuated secondary mirror. Secondary functions include:
  - Application of dispersion and off-axis offsets[1] in target acquisition and fast tip-tilt correction modes.
  - Operator GUI to control FTT/NAS in standalone mode.
  - Capability to record data in FITS format and interrogate previously-recorded data using a separate analysis GUI.
- Provide AMOS-requested functionality for UTM commissioning (via the FLC control GUI which it will be possible to use with the FTT/NAS controller software):
  - Live display of average and rms centroid coordinates.
  - Capability to record monitor data, including centroids, in Comma Separated Value (CSV) format.

Details of the software operational modes and functionality are given later in this document.

The software is obviously key to realizing many of the functional requirements that have been placed on the FTT/NAS system. It also plays a critical role in satisfying the fast tip-tilt correction performance requirements i.e. closed-loop bandwidth and limiting sensitivity. We have designed the control software to minimize the latency from camera readout to corrections being applied to the fast tip-tilt actuator, in order to maximize the closed-loop servo bandwidth for bright sources. Measurements of the latency achieved are presented in Sec. 2.2.1. The centroiding algorithms (including de-biasing of the raw CCD data using dark and flat-field frames) will need to be optimized in order to realize the low-light-level performance, and we have begun to address this by testing candidate algorithms on simulated data. However, further work on algorithms is planned for the final design phase.

[1] An off-axis offset allows the use of an off-axis reference star.

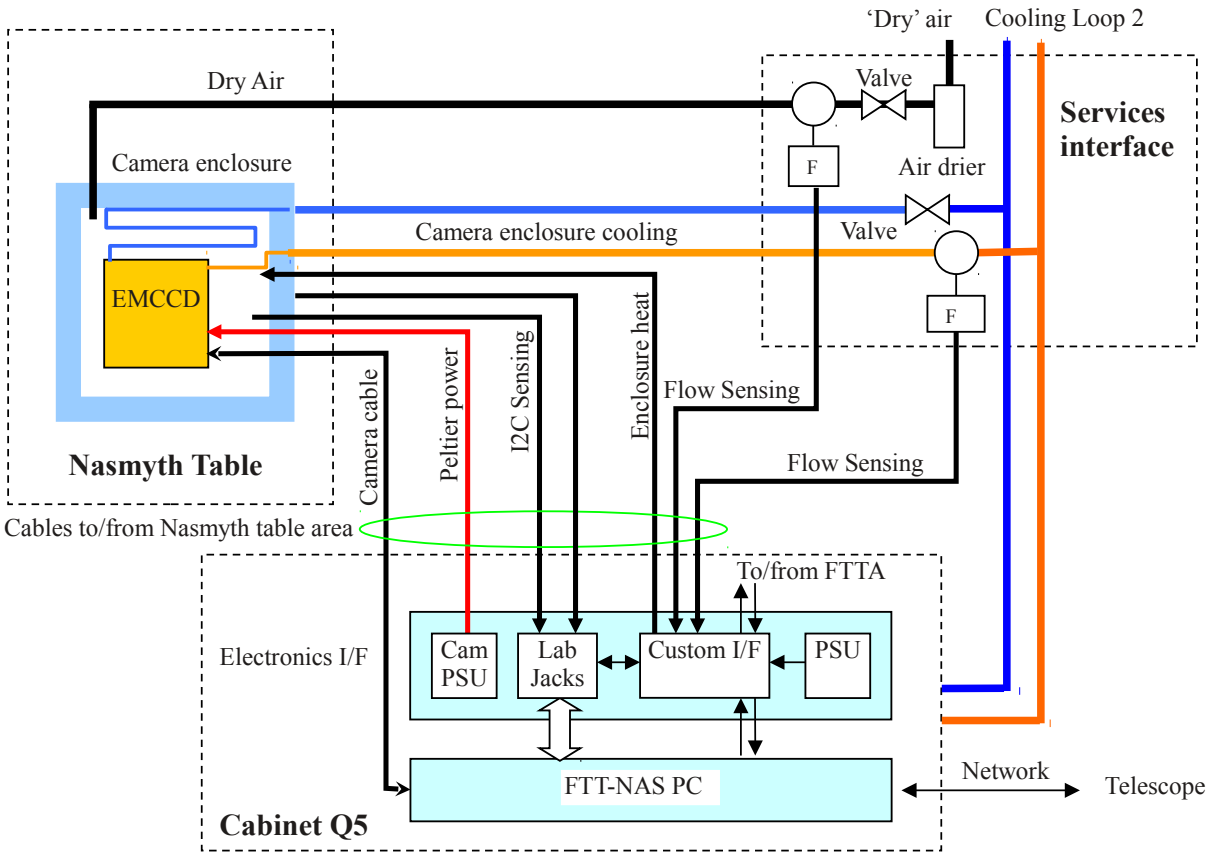


Figure 1: Overview of the control system.

1.1 Control system overview

A personal computer in Cabinet Q5 controls the system. It controls and collects data from the EMCCD camera, manages the camera's thermal environment, sends UTM offloads to the local telescope and moves the telescope's tip-tilt mirror. The control system is illustrated in Figure 1.

1.1.1 Camera control and monitoring

The camera converts sky images into electronic data for the purposes of source acquisition and compensation of atmospheric tip-tilt perturbations. The device is an Andor iXon X3 897 electron multiplying CCD camera, chosen for its high sensitivity and rapid readout. The camera's EMCCD sensor is cooled by a Peltier cooler and the EMCCD temperature is monitored by a sensor in thermal contact with the EMCCD.

Camera control and readout are via multicore cable connected directly to an Andor CCI-23 PCI card in the rack computer.

1.1.2 Environment control and monitoring

The camera is located in a thermal enclosure. The thermal control and monitoring system performs two important functions:

1. To protect the camera:
    - To prevent the camera from overheating
    - To prevent the camera from being operated below 0 °C
    - To prevent the camera from being operated in humid conditions
2. To comply with the thermal restrictions on components near the optical beam i.e. to keep the surface of the camera enclosure within 2 °C of the ambient air temperature.

Since the camera is not specified to operate below 0 °C the method used to remove heat from the enclosure (and also the camera peltier stage) is the electronics cabinet cooling loop. The temperature of the fluid in this loop tracks ∼2 °C below ambient temperature down to 1 °C (dew point allowing). The desired approximate flow rate is set using a simple manual valve. Flow sensing is not essential but is used for setting the flow and can be a useful diagnostic. It is possible that the camera enclosure may become too cool under certain circumstances and to prevent this (and help with any initial warming up to operating temperature) a resistance heat source is incorporated. A simple algorithm will be used to control this heat source.

Interface and signal conditioning requirements for all thermal control sensors and signals connected to the camera enclosure include:

1. Camera case external temperature sensor — for monitoring.
2. Camera enclosure air temperature sensor — for monitoring and control.
3. Cold plate temperature sensor — for monitoring.
4. Humidity sensor — for monitoring and control.
5. Heating element — for temperature control.

Interface and signal conditioning requirements for all thermal control sensors, signals and actuators connected to the services interface panel include:

1. Coolant flow rate sensor — for monitoring.
2. Dry air flow rate sensor — for monitoring.

Monitoring signals from the Nasmyth optical table include the following but might be extended if further, more accurate, temperature monitoring of the Nasmyth optical table is desired:

1. Nasmyth optical table temperature (in vicinity of camera).
2. Temperature of common base-plate.
3. Humidity above optical table.
4. Air temperature above optical table.

Additionally, the environment control system can enable or disable the camera in software according to environment criteria. The environment control criteria are listed in Sec. 2.5.8.

1.1.3 Telescope control and monitoring

The computer has the following control over the local telescope:

1. Movement of the telescope fast tip-tilt actuator via control voltages generated in an analogue I/O card. This is the control point of the fast tip-tilt servo loop.

2. Telescope steering errors detected by EMCCD image position errors (acquisition mode) or accumulated fast tip-tilt mirror pointing errors (fast tip-tilt mode) can be corrected via the network.

The computer also monitors the following telescope signals:

1. Fast tip-tilt mirror position via signal voltages into the analogue I/O card.
2. The rotation matrix that converts between EMCCD and fast tip-tilt mirror coordinate systems, via the network.
3. Other rotation matrices for the purposes of annotating images in the control/display and analysis GUIs.

It is expected that the telescope steering, rotation matrix data and sky coordinates will be conveyed via the interferometer's publish/subscribe system.

1.1.4 Derived real-time performance requirements

The camera, the computer and the fast tip-tilt actuator form a servo loop with the following performance requirements [RD3]:

1. A closed loop bandwidth of 40 Hz (50 Hz goal) for bright sources.
2. For the faintest (16th magnitude) science targets, a bandwidth of around 15 Hz is suggested.
3. The total delay (half the exposure time, plus the readout time, the centroid computation time and the mirror actuation time) should not introduce a servo phase lag of more than 25°.
4. It must operate within the derived tilt error budget, which allocates 34 milliarcsec RMS to detection noise centroiding error, 15 millarcsec RMS to speckle noise centroiding error and 30 milliarcsec RMS to residual seeing tilt. This is for "reference seeing" with $r_0 = 14$ cm and $V_{wind} = 10$ m/s.

These requirements translate into a maximum delay of 1.74 ms for bright sources (goal 1.39 ms) and 4.63 ms for the faintest sources. To ensure that the computational component of the lag is consistently the minimum possible, it must execute in "hard real time" context, that is, with guarantees that the response and computation always meet hard deadlines.

Other servos (thermal control, UTM offloads) run on time scales of seconds and do not require hard real time perfomance.

1.2 Software execution environment

The software execution environment is illustrated in Figure 2.

The system software will run on a rack mounted Intel-style computer, of height 3U in order to host the necessary interface cards. The currently preferred candidate computer is an Amplicon Ventrix 2008-P4.1. The hard real time operating system is expected to be Linux 2.6.32 (the MROI standard version) with Xenomai 2.6 kernel patches, but the source code is compatible with more recent releases of each, including the forthcoming Xenomai 3.

High bandwidth hardware interfacing is via two PCI cards: one is an Andor CCI-23 that interfaces with the Andor EMCCD camera, the other is a third party analogue card (the preferred candidate is an Advantech PCI-1716/L) for actuation of the fast tip-tilt mirror and monitoring of its current position. Some signal conditioning of the high bandwidth analogue signals may be required. Where access to each card is necessary to close the fast tip-tilt servo loop, Xenomai will be used to ensure minimal latency.

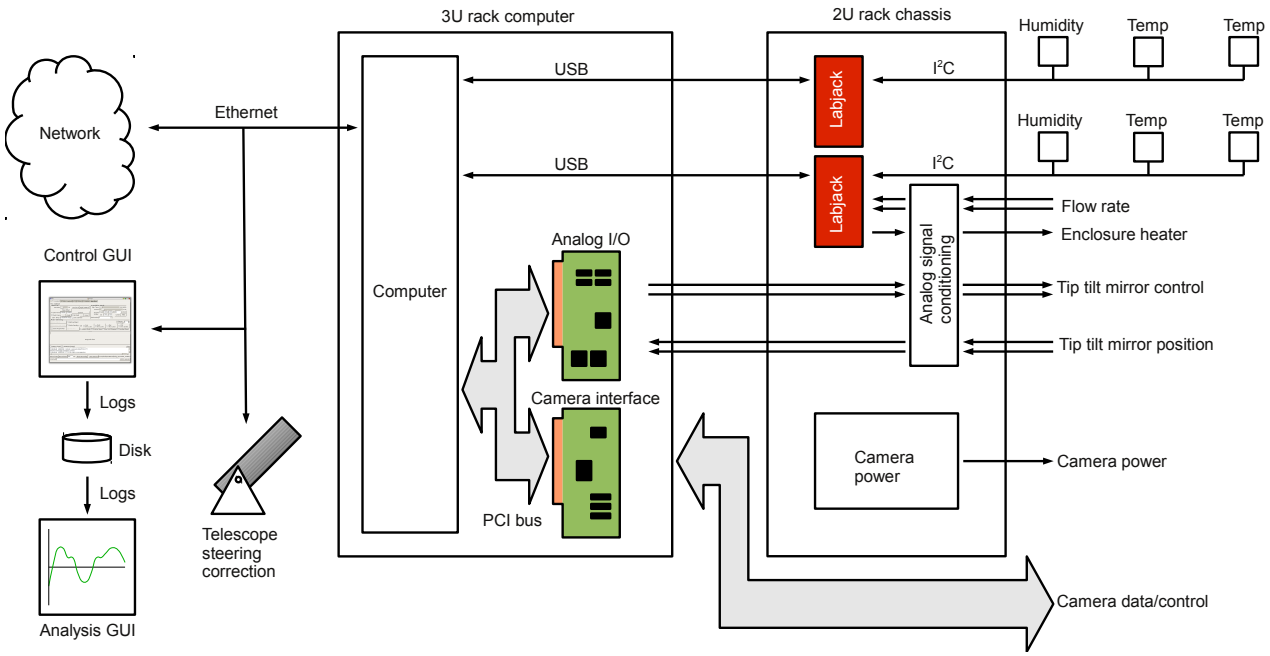


Figure 2: The software execution environment.

Local low bandwidth interfacing is via two of the computer's USB ports. Each port will connect with a Labjack U3 to provide access to an I^2C bus and a variety of digital and analogue input and output ports. Each I^2C bus will in turn connect to digital temperature and humidity sensors distributed through the system, while the analogue interfaces will connect to flow rate sensors and the camera enclosure heater via custom signal conditioning electronics. These interfaces do not have hard real time requirements and will run in Linux using libraries supplied by Labjack.

Low bandwidth UTM offloads are sent to the local telescope via the ISS. This interface does not have hard real time requirements and will use ordinary Linux network libraries.

Standalone control and analysis GUIs (Sec. 2.7 and 2.8) will also be provided. They allow a user to interact with the system software in circumstances when control via the ISS is not available or convenient, for example during commissioning. The control GUI runs under Linux and requires a network connection to the system and environment controllers (this can be a loopback socket on the controller computer). The analysis data runs in Matlab (available for a variety of operating systems) and needs only access to logs saved by the control GUI.

The network is also used to synchronise the computer's system clock using NTP (Network Time Protocol). This will allow data to be timestamped to an accuracy of $\sim$1ms.

In order to test the software when the hardware is not present (in Cambridge, for example), the system can be configured to generate simulated data. This allows all of the software (with the exception of the device drivers) to be exercised.

1.3 Software architecture

The FTT/NAS software consists of two components used to control the FTT/NAS hardware, a user interface software application (described in Sec. 2.7), and a fourth component used for offline visualization and analysis

of previously-recorded monitor data (see Sec. 2.8).

The control software is partitioned into two components because of the need to maintain the environmental conditions within the FTT/NAS camera enclosure at all times, not just when the system is operational. The component responsible for controlling the camera environment and enabling/disabling the camera is called the "environment controller", and the component that performs the primary FTT/NAS system functions such as target acquisition and fast tip-tilt correction is called the "system controller".

The environment controller will normally run continuously. The system controller runs when the interferometer is observing or preparing to observe. The system controller subscribes to monitor data (using the dlmsg interface also used by the control/display GUI) from the environment controller in order to detect whether the environment controller has given it permission to operate the camera. In all other respects, the two systems function independently.

The control/display GUI provides a graphical user interface for commanding the system and environment controllers and for live display of their monitor data (including camera images). It can also record monitor data to a set of FITS-format files when requested to do so by the operator. The control/display GUI may be used in one of two modes: a "standalone" mode in which it is fully functional, and a "display-only" mode which can safely be used when the FTT/NAS system is under the control of an ISS supervisor. In the latter mode only the monitor data display and recording functionality is available and no commands which change the system state are issued by the GUI. When the FTT/NAS system is under ISS control, the control/display GUI need not be used (although it provides the only method of recording data in the format accepted by the offline analysis GUI).

The analysis GUI is used to visualize and process monitor data previously recorded to FITS files by the control/display GUI. It provides a general diagnostic capability by means of functions for displaying image sequences, graphing scalar monitor data, and simple data analysis.

Each FTT/NAS system to be deployed at MROI will have its own instances of the environment controller, system controller, and control/display GUI. There are no interactions between the software components for different FTT/NAS systems.

2 Software Design and Implementation

2.1 Introduction

To assist in the description of the controllers, we begin with some nomenclature about the architecture of the operating system that they run in.

The operating system is Xenomai (`http://www.xenomai.org/`). It has two important properties that are exploited in this project:

- It runs in "hard real time". Consequently it reacts to hardware inputs and controls hardware outputs with minimal and predictable delays. This is a very useful property for introducing a computational element to servo loops, as is needed in the FTT/NAS.
- It coexists with Linux. This means that the richness of the Linux system infrastructure is available to the software whenever hard real time performance is not required.

Both Linux and Xenomai have a "user-space" and a "kernel space" component:

- Kernel space contains the core system functionality, such as hardware interfaces and memory management. Code running in kernel space has few restrictions on what it can do.

- User space is where the user interacts with the system. There are more restrictions on what code is allowed to do in user space, but it is much harder for user space code to crash the system.
- An interface is provided to allow applications to communicate across the barrier between user space and kernel space.

The system controller in particular makes extensive use of the user space and kernel space domains of both Xenomai and Linux.

2.2 System Controller

The system controller software performs acquisition functions, fast tip-tilt servo loop closure and transmission of system diagnostic information (including camera images) over the network. This software can be broadly divided into two parts:

- Code that implements the real time fast tip-tilt servo loop using Xenomai. This is implemented as one kernel space thread and three user space threads.
- Code that implements supporting functionality, including network control and monitoring, using non-real-time code in Linux. This is implemented as a single thread (apart from the threads generated by the system controller GSI code in the course of its interactions with an external client).

Note that the system controller code is used for both the FTT and FLC systems to simplify code maintenance. However in the latter the fast tip-tilt functionality will be disabled.

2.2.1 Fast tip-tilt servo loop

The core function of the software is closure of the servo loop between the EMCCD camera and the fast tip-tilt mirror. This closure must be implemented within a hard real time operating system in order to guarantee calculations meet deadlines on every servo cycle. Xenomai is used because it is free, open source, well integrated with Linux, and proven during development of the MROI delay line metrology system.

The core software reads a camera image, calculates a centroid and then a correction, and sends the correction to the mirror. All this occurs in real time context. An interface is also provided to the remainder of the fast tip-tilt software, which runs in Linux. The implementation is illustrated in Figure 3.

When the CCI-23 camera interface card reads out a camera image, it writes the data to computer memory using direct memory access (DMA) and then triggers an interrupt to notify the computer that the transfer is complete. This interrupt is intercepted by an interrupt service routine that has been ported to Xenomai from Andor's open source driver, thereby providing real time access to image data while maintaining compatibility with the remainder of Andor's code. The interrupt service routine copies the image data to a buffer and blocks a Xenomai user-space routine from reading the data until the copy is complete.

When the user-space routine is allowed to read the data, it does so immediately. It firstly reads the computer's NTP stabilised system clock to timestamp the data[2], then it calculates a centroid (Sec. 2.5.2), exploiting Xenomai's ability to use the computer's floating point hardware in real time context in user space.

A position error is then calculated from the centroid, which is used as the error input for the fast tip-tilt servo (Sec. 2.5.3). An appropriate correction is sent to a custom Xenomai driver for the analogue interface card, which applies scaled voltages to the FTTA controller inputs.

[2]Xenomai V2.6 or later is needed to read the system clock in real time context and hence provide an accurate timestamp.

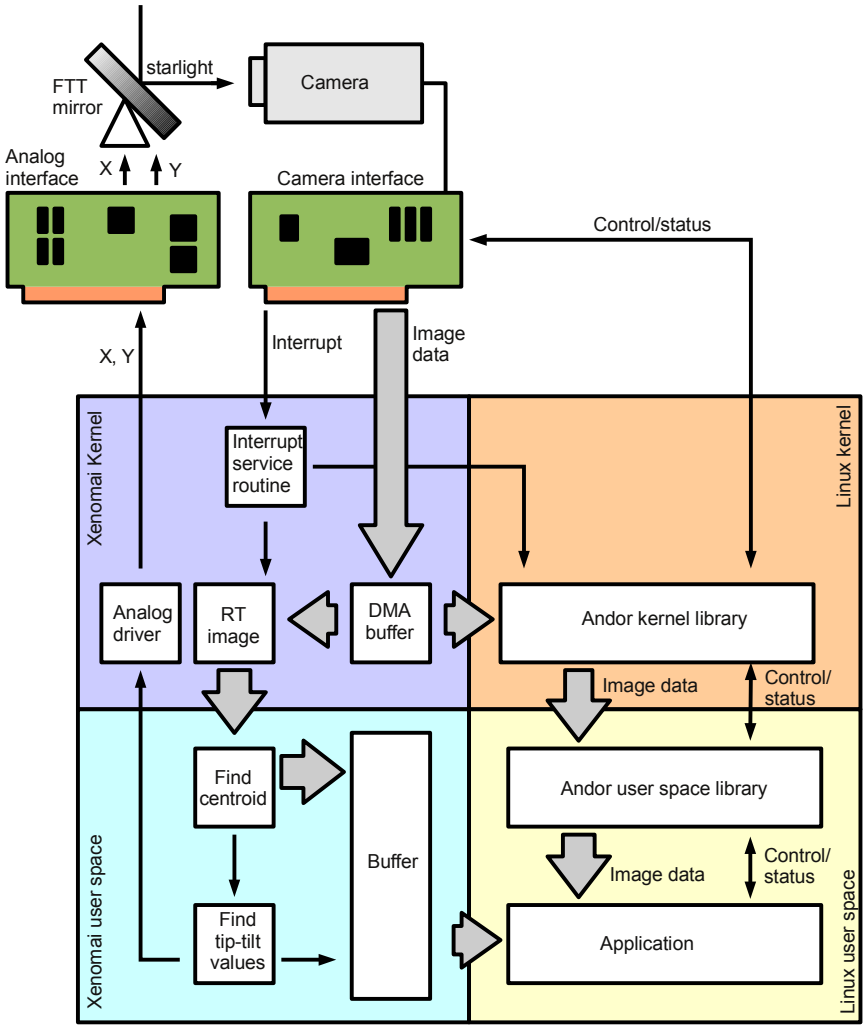


Figure 3: The implementation of the core servo loop in Xenomai and its connection with non-real-time Linux code.

Meanwhile the image data and corrections are buffered into an ordinary Linux application, where they are used for non-real-time tasks and logging. This buffering serves as an interface between the Xenomai code, which must meet strict deadlines, and the Linux code, which only needs to be able to keep up "on average".

To measure the delay introduced by the software, we have written a test program in Xenomai. When it receives an interrupt, it generates fake camera data in Xenomai kernel space , transfers it to user space, calculates a floating point centroid and a correction, and then emits a pulse on the test computer's printer port to simulate an interaction with an analogue PCI card. Effectively this simulates the Xenomai portion of Figure 3. The interrupt is generated with a signal generator, and the latency between the interrupt and the printer port pulse is measured with an oscilloscope. For a computer with a dual core 3 GHz AMD Athlon II processor, over many trials the maximum latency was measured to be 38 µs, an insignificant fraction of the requirement discussed in Sec. 1.1.4.

The Xenomai user space code is structurally complete and tested, with a temporary centroiding algorithm in place. The Xenomai port of the Andor interrupt service routine is also complete and has been successfully tested, as has the buffer code.

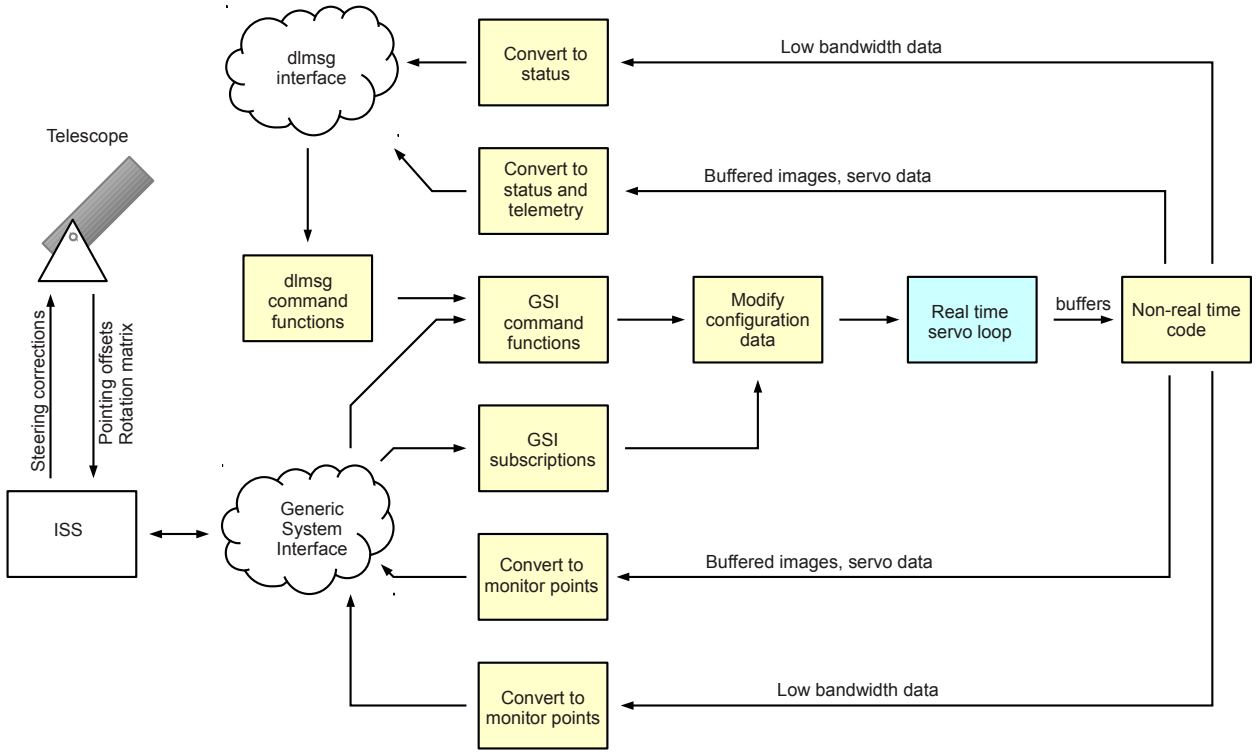


Figure 4: The implementation of control and monitoring functionality. The box labelled "Real time servo loop" encapsulates the core functionality illustrated in Figure 3.

The key remaining task is to write the analogue driver for Xenomai. The candidate analogue PCI card has been carefully chosen to simplify this task: it has a simple interface whereby analogue voltages are straightforwardly controlled or measured via access to specific registers which are well documented in the manual. Hence this driver is expected to be similar to the low latency link driver in the MROI metrology system, which was not difficult or time consuming to write.

2.2.2 Supporting functionality

In addition to closing the fast tip-tilt servo loop, the system controller software must also set up and initiate readout of the camera, send UTM offloads to the local telescope, execute commands and produce monitor data. These tasks are largely network dependent and vulnerable to network buffering and contention, hence there is no advantage to implementing them in real time context and it is much better to make them delay tolerant. The implementation is illustrated in Figure 4.

Two interfaces are provided for monitoring and control. These are the interferometer's Generic System Interface (GSI) and Cambridge's dlmsg interface. There is potential for conflict in that commands can arrive from both sources, hence the system controller will only accept commands that change the system state from one source. The nominated source is currently set via a flag at compile time.

The GSI is the primary interface to the system and is intended to connect the system to the Interferometer Supervisory Subsystem via the network. The code is implemented using the C threads version (currently version 1.4a) of the GSI.

When a GSI command arrives, GSI code decodes it and checks the parameters before calling a local function to deal with that command. If the command passes contextual checks (for example, it is forbidden to initiate a

camera readout while one is in progress), it will write the appropriate data into a configuration structure shared with the real time servo code. The real time code sees the changed values and reconfigures itself appropriately. Xenomai mutexes are used to prevent real time and non-real-time code from accessing configuration values simultaneously.

The system controller will also subscribe to ISS data via the GSI publish/subscribe system when this becomes available. As with commands, when subscribed data arrives it will cause configuration data to be modified. GSI subscriptions include telescope dispersion offsets, telescope off-axis offsets, the telescope rotation matrix used to convert between EMCCD and fast tip-tilt mirror coordinates, other rotation matrices for display purposes and permission to use the camera from the environment controller.

As mentioned above, the real time servo loop periodically produces data that must be shifted to non real time functions where immediate processing is not guaranteed. The transition is implemented with circular buffers and synchronisation is achieved with blocking Xenomai message queues [3]. The buffers ensure that the real time servo never has to wait if the non real time code pauses execution, while the message queues ensure that the non real time code waits for fresh data once it has "caught up".

Some data is then "decimated", that is, only one item in every n is used for the purposes of conserving network bandwidth and disk storage. Data generated at the camera frame rate and decimated camera data (such as images) are converted into monitor point data and sent back to the Interferometer Supervisory System via the GSI. Data are manipulated to calculate a UTM offload and sent to the GSI for forwarding to the telescope via the ISS. Exceptions and faults are also handled via the GSI framework.

The dlmsg interface is the secondary interface to the system, and is used to connect the system with the control GUI. Commands and data are handled on separate network ports, mirroring the GSI architecture. When a dlmsg command arrives from the control GUI, the receiving code calls the relevent GSI function with the decoded command parameters. Program execution then continues as if a GSI command had been issued. Data generated by the program are converted into the dlmsg format and sent to the control GUI for display and logging. The dlmsg interface allows the system controller to get permission to use the camera from the environment controller, however the GSI publish/subscribe system will be required, even in standalone mode, to communicate with the telescope. The design facilitates testing, commissioning and operation of the fast tip-tilt system when the Interferometer Supervisory System is unavailable.

The dlmsg code has been written so that it can easily be excised when it is no longer needed: almost all of the dlmsg code is kept in separate files, while the few remaining hooks in the system code are clearly marked and can be excluded at compile time by modifying a single line of code.

This code is almost complete. It includes a generator of fake camera data for when a real camera is not available, and testing using this generator is now underway using the GSI standalone test environment and the dlmsg control GUI. The remaining tasks are the integration of the Xenomai camera driver code and the interface to the telescope. The former has been proven in separate testing. The latter will be implemented using the interferometer's publish/subscribe system.

2.3 Environment Controller

The environment controller operates independently from the system controller so that it can function even when the camera is not in use. It is a simple sensing application which reads and reacts to data on timescales of the order of one second (the exact period has not yet been decided). Hence the environment controller has no hard real time requirements and runs in ordinary Linux. The environment controllers in the first light camera and fast tip-tilt systems are identical.

[3] Originally this was implemented using Xenomai mutexes and condition variables but it was found that the test system became unstable when a request to lock a Xenomai mutex was issued while an ordinary Linux mutex was being held in the same process.

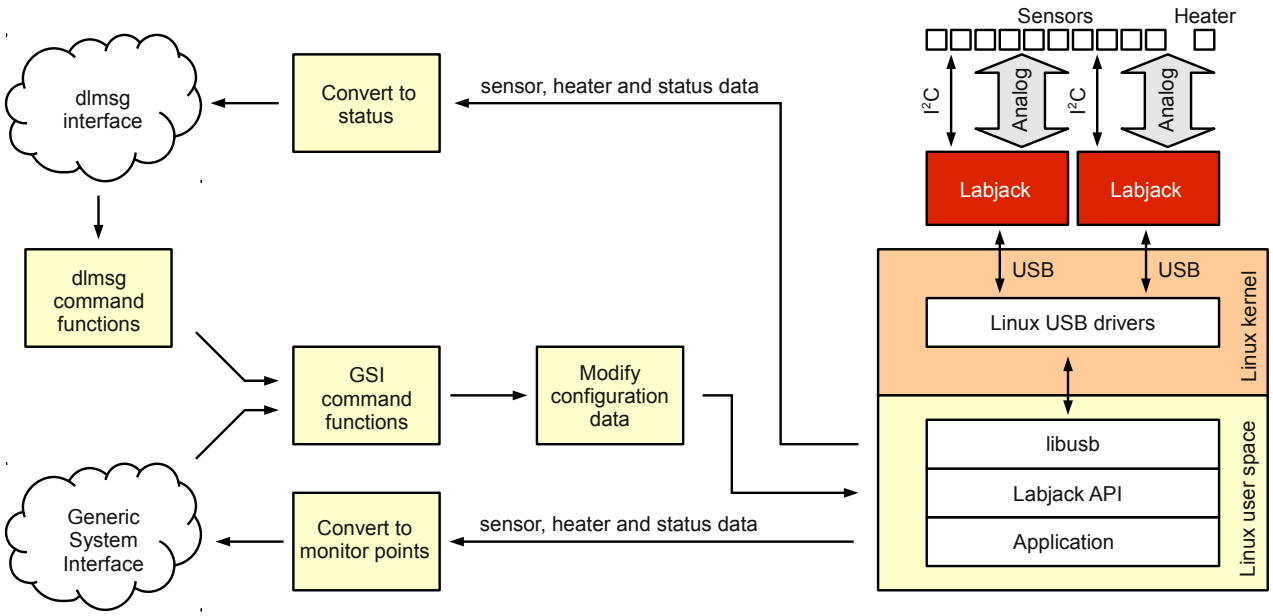


Figure 5: The implementation of the environment controller.

The program design is illustrated in Figure 5. As with the system controller, the GSI is the primary interface and dlmsg is the secondary interface. Both can call the GSI command functions, but commands that change the system state will only be accepted from one source. Commands modify application configuration data that is protected by mutexes. The environment controller changes its behaviour on the basis of the configuration values and publishes monitor data to both interfaces. Exceptions and faults are handled within the GSI framework.

The application communicates with sensors and a heater via two USB-driven Labjack U3 boards. The sensors are either analogue voltage or I^2C types (we do not yet have candidate sensors). The two Labjacks provide two independent I^2C buses, which will be helpful if two I^2C devices are needed that have the same I^2C address (this is a likely scenario with respect to humidity sensors).

When the application reads sensor data from the Labjacks or changes the power to the heater, it calls functions in the open source Labjack application programming interface. This interacts with the Linux libusb library, which in turn communicates with Linux USB kernel modules and the hardware. Hence the application need have no direct interaction with the Linux kernel.

The application reads the sensors at a regular rate (currently once per second). On the basis of the readings it decides what the heater power should be and whether the camera should be enabled. It then changes the heater power, and sends monitor data with the camera decision and the sensor and heater values. The system controller notices the published camera decision and acts accordingly.

The environment controller software is currently in the prototyping stage. A program has been written which reads temperature sensors at 1 Hz and publishes the results to the dlmsg interface. This program is in daily use monitoring temperatures during the fast tip-tilt system optical mount tests.

2.4 Testing and integration

Initial testing of the system controller is making use of an internal camera data generator that exercises most of the software functionality. When the software is stable, the EMCCD camera hardware will be integrated.

Testing will then involve placing the fast tip-tilt hardware in Cambridge's optical laboratory, imaging an artificial source that can be perturbed in a way that mimics atmospheric turbulence, and correcting for that perturbation. At present the source will most likely be an oscilloscope operating in X-Y mode and perturbations will be introduced by a digital signal generator emulating Kolmogorov turbulence. By that time the camera should be in its enclosure and that will allow the environment controller, and its interaction with the system controller, to be tested.

Application testing will use both the GSI standalone test environment and the dlmsg interface.

2.4.1 dlmsg testing

Initial testing will use Cambridge's dlmsg interface, because the control GUI already allows a high degree of interactivity with the system and environment controllers. It also saves data in a format that can be read later by the analysis GUI for closer examination. This will allow everything about the system and environment controllers to be tested with the exception of the GSI interface.

2.4.2 GSI testing

The GSI distribution provides a facility that allows the user to run a script file containing GSI commands. The application executes these commands as if they were coming from the Interferometer Supervisory System and both prints its monitor point data to the screen and writes it to a file.

This facility is already being used extensively to test application functionality and its interaction with the GSI. In particular, it is used to configure and initialise the system even when control and logging are carried out with the control GUI and dlmsg.

2.4.3 Integration

The software will work at the MROI just as it does in Cambridge, but will need to be integrated into the interferometer network to achieve its full potential. The major steps needed are:

- Integration with the networked version of the GSI. If the GSI file interface emulates the network interface sufficiently well, this should be trivial.
- Integration with the publish-subscribe system. Initially the environment controller and the system controller will communicate with each other via a dlmsg interface, but will be switched to the publish/subscribe system when it becomes available.
- The publish-subscribe system is to be used to communicate with the telescope. Hence target acquisition and fast tip-tilt functionality will not be available until the publish-subscribe system is commissioned.

2.5 Algorithms

2.5.1 Source and centroid window selection

During acquisition there will often be sources within the field other than the source of interest. A mechanism is needed to ensure that only the source of interest is selected and acquired.

If the source of interest (the "reference object") is the brightest in the field, then the user can request that the software select the brightest object in the entire acquisition image. A centroid window will then be placed around the reference object (the size has yet to be determined). Sources outside the centroid window are ignored. As the reference object is moved to the target position (the "objective point") the centroid window

follows it. Hence, if the act of acquiring the source moves an even brighter source into the field, the brighter source is ignored because it is always outside the centroid window.

If the reference object is not the brightest in the field, the user can manually select a centroid window around it to exclude other sources. In the control GUI they would draw a rectangle around the source to do this. Sources outside the centroid window are ignored. Once again, the centroid window tracks the source position as the source is acquired.

In fast tip-tilt mode, there is no source selection process because the centroid is calculated over the entire fast tip-tilt subframe.

2.5.2 Image centroiding

Centroiding determines the centre of the diffuse image of the source, to within a fraction of a pixel. The difference between the centroid position and the objective point on the EMCCD is then used to calculate an error signal. In fast tip-tilt mode the error is used to calculate how the fast tip-tilt mirror should be moved. In both fast tip-tilt and acquisition modes the error is also used to calculate a UTM offload (Sec. 2.5.7). The net effect of these operations is to cause the image to move toward the objective point.

The objective point itself will be updated during an observation according to lateral dispersion and off-axis offsets received from the telescope.

Currently a simple "centre of gravity" algorithm is used to calculate the centroid within a subframe:

1. Calculate the sum s of the intensities of all the pixels.
2. For every pixel, multiply the pixel x coordinate by the pixel intensity and calculate the sum of the multiplicands s_x.
3. For every pixel, multiply the pixel y coordinate by the pixel intensity and calculate the sum of the multiplicands s_y.
4. The centroid coordinates are $(s_x/s, s_y/s)$.

This is a placeholder algorithm and its performance in the servo has not yet been investigated.

One alternative which currently is under investigation is a median centre algorithm:

1. Sum the image rows together to form a single row.
2. Find the point in this row that has equal numbers of counts on either side with subpixel accuracy.
3. Sum the image columns together to form a single column.
4. Find the point in this column that has equal numbers of counts above and below with subpixel accuracy.
5. The centroid coordinates are formed by the two points found.

The performance of this algorithm is being tested in simulations related to servo loop performance and preliminary results are presented below (Sec. 2.5.3).

Whatever method is chosen, dark correction, flat field correction, and clock induced charge pattern noise correction will be available to increase the accuracy of the computed centroids.

2.5.3 Fast tip-tilt servo

The fast tip-tilt servo will be a proportional-integral-derivative (PID) controller. It must operate within the derived tilt error budget [RD3], which allocates 34 milliarcsec RMS to detection noise centroiding error, 15 millarcsec

RMS to speckle noise centroiding error and 30 milliarcsec RMS to residual seeing tilt. It must also meet this requirement for 16th magnitude sources (the interferometer design limit).

This servo is being studied in simulations using the median centre algorithm mentioned above (Sec. 2.5.2). The numerical model includes:

- Atmospheric turbulence, modelled as a phase screen with $r_0 = 14$ cm seeing at $\lambda = 500$ nm moving past the telescope at $10\,\mathrm{ms}^{-1}$ (that is, the reference seeing mentioned in Sec.1.1.4).
- EMCCD sensor noise.
- A readout delay of 0.7 ms and mirror response delay of 4 ms. This is pessimistic: although the readout latency is now expected to be about 1 ms, true mirror response is closer to 2 ms.

The simulation has found that this servo operates within the residual requirement for sources down to 15th magnitude. Adding Kalman filtering improves performance by another magnitude (thereby meeting the brightness requirement). Therefore Kalman filtering will be incorporated into the servo. Hence the simulation shows that this servo meets the requirement, and for a less pessimistic delay the performance can only improve further.

We aim to improve performance further by incorporating in the servo algorithm some compensation for latency. This is likely to be some kind of Smith predictor or, more likely, a Kalman predictor. Further simulation is needed to determine the benefit that would result.

Further validation will involve physical tests. The camera will be set up in a darkened optical laboratory and an image of an oscilloscope trace in X-Y mode will be focused onto the sensor. The glowing phosphor dot will be perturbed by a signal generator simulating atmospheric turbulence in real time and the system will attempt to correct for it. The correction might involve a tip-tilt mirror or adding a correction voltage to the oscilloscope inputs. Either way, this allows us to test system response to atmospheric turbulence. However, it cannot test speckle noise response, this will have to be addressed through simulation.

Optimal servo and Kalman filter parameters have not yet been determined. The software allows them to be modified at run time to suit the current observing conditions. This operation can be partially automated via the *setTipTiltBandwidth* command, which will set the closed loop bandwidth by choosing parameters that optimise servo performance. It might also choose which EMCCD clocking mode (conventional or custom) is optimal.

If the user chooses to override default servo parameters, they revert to their defaults whenever the servo bandwidth or the frame rate change.

2.5.4 Seeing estimates

The system controller calculates and publishes seeing estimates based on data received in acquisition and fast tip tilt modes. The manner in which this is done is yet to be decided, however the following is a possible outcome:

- In acquisition mode, a spatial seeing estimate is calculated using the size of the image on the EMCCD, possibly integrated over a standardised time period using multiple images.
- In fast tip-tilt mode, a spatial seeing estimate is calculated from the size of the movements demanded of the fast tip-tilt mirror in order to stabilise the image. Care is needed to ensure that offload corrections do not appear in this data.
- In fast tip-tilt mode, a coherence time estimate is calculated from the velocity of the movements demanded of the fast tip-tilt mirror in order to stabilise the image.

2.5.5 Dark and flat field correction

Commands are provided to acquire dark and flat field full frame images. These commands perform acquisitions with standardised parameters such as a fixed exposure time, shutter open or closed, and so on. Once an image is acquired, it is returned when the command returns and can be logged for later use.

Commands are also available to use previously acquired images as datasets for on-the-fly dark and flat field correction of images currently being acquired. A single full-frame image can be used to correct new images whether they are taken in acquisition mode or fast tip-tilt mode, with conventional or custom clocking.

2.5.6 Clock induced charge model

As discussed in RD4, the EMCCD is clocked out in an unusual way in order to meet the performance requirements. This results in charge from pixels outside the region of interest getting added to those within it during the readout process. Furthermore, for any given pixel, the number of pixels that dump their charge into it varies according to its position within the image and the position of the subframe on the EMCCD.

For an ideal EMCCD this is of no consequence because the field is dark and no charge is generated except in the vicinity of the source image. However, in practice there is a low level uniform background noise caused by clock induced charge transfer. The result is that the image contains pattern noise that depends on the subframe position on the EMCCD.

Fortunately this background is predictable and can be modelled and subtracted from data. To confirm this, a software model of the EMCCD pixels and registers has been written. It takes about a second to "clock out" a single subframe from the "uniformly illuminated" EMCCD model to discover the pattern noise in the image. This program will be incorporated into the system controller so that pattern noise can be subtracted from real data without having to acquire a large series of dark frames. Because the pattern noise is constant for a given subframe position, the model need only run once immediately prior to each fast tip-tilt observation.

The model was instrumental in choosing the improved 32×32 clocking scheme. It showed that the scheme had no pattern noise when the subframe centre was within a few pixels of the centre of the EMCCD (the on-axis guiding position) and even in the worst case off-axis guiding positions the pattern noise was less than that of other candidate schemes.

2.5.7 UTM offloads

UTM offloads are sent to the telescope at a maximum (provisionally) 1 Hz rate, although the software is capable of sending them at up to the camera frame rate. They are published as monitor points and the ISS forwards them to the telescope.

In acquisition mode, corrections rely on camera images but the camera frame rate is independent from the rate at which telescope corrections are sent and usually higher. Hence UTM corrections are determined in the following manner:

- If the camera frame period is greater than the telescope correction period, calculate an offload for every camera frame and send to the local telescope.
- If the camera frame period is less than the telescope correction period, average the images that were acquired since the last telescope period. Use this averaged image to calculate and send a correction.

It is likely that the corrections will be calculated using a simple "centre of gravity" centroid as described above, enabling the generation of an error term for the UTM offload servo. The slow update rate ensures that there will be plenty of light available even for the dimmest of science targets.

In fast tip-tilt mode, the fast tip-tilt mirror demands are integrated over a long period (nominally the telescope correction period) to determine any long term drifts in fast tip-tilt mirror pointing that indicate a telecope tracking error. The error is converted into a telescope steering correction and once more fed into the UTM offload servo.

2.5.8 Environmental control

In the following discussion, the numbers are provisional.

The camera should be allowed to operate only if all the following conditions are true:

- The camera enclosure internal temperature is less than 30 °C.
- The camera enclosure air temperature is more than 0 °C.
- The camera enclosure humidity is between 5% and 95%.
- The cold plate temperature is above the dew point.
- The cold plate temperature is no more than 5 °C below ambient temperature.

The following events should issue warnings:

- The coolant flow is lower than (TBD) litres per minute.
- The air flow is lower than (TBD) litres per minute.
- The camera enclosure external surface temperature is not within 2 °C of ambient.

The following is the enclosure thermal control algorithm:

- If the enclosure air temperature is below 0 °C heat the enclosure until air temperature is above 1 °C. Provision will be made in the hardware and software for the heater power to be variable, however it is expected that constant heater power is all that is required.
- If the enclosure external surface temperature is more than 2 °C below ambient heat the enclosure until this temperature is less than 1 °C below ambient.

2.6 ISS interfaces

2.6.1 System classes and state model

The system controller is a standard GSI asynchronous system and there is nothing unusual about it in this respect. It implements the following GSI state change actions:

- *initializeFTTCamSystemAction*: Initialises the system, including the camera hardware.
- *shutdownFTTCamSystemAction*: Shuts down the system, including the camera hardware.
- *aboutToAbortFTTCamSystemAction*: Places the hardware in a state where it is safe without program supervision.
- *stopFTTCamSystemAction*: This action is almost empty, it just logs that the system has stopped.

The environment controller work has not yet progressed to the GSI class implementation stage, but there is no reason to believe it will be anything other than standard.

2.6.2 Deployment scenarios

We expect that most of the time the FTT/NAS software components will be deployed according to one of the following scenarios:

1. Control by an ISS Supervisor: the Supervisor sends commands to the FTT/NAS system and environment controllers.
    - There will eventually be two kinds of ISS Supervisor: one for sequenced observing and an alternative one for interactive control by an operator.
    - The FTT/NAS control/display GUI may be used as necessary to record data for the analysis GUI or to supplement the facilities for monitor data display provided by MRO-supplied software.
2. Control by an operator interacting with the FTT/NAS control/display GUI: the control/display GUI sends commands to the FTT/NAS system and environment controllers in response to the operator activating controls (e.g. butttons) on the GUI.
    - The control/display GUI may be running on a PC in the MROI control room or on a laptop located in the Unit Telescope enclosure.

In all scenarios the control/display GUI may be run on any convenient PC that is on the same network as the FTT/NAS computer.

The software has been designed to allow multiple instances of the control/display GUI to run simultaneously (for example one in the control room and one in the telescope enclosure). Only one such instance (or the ISS Supervisor) can control the system; the others must be in display-only mode.

In the initial version of the control/display GUI, the choice between "standalone" (full control) and display-only mode is made when the GUI is started. To change the computer from which the operator controls the system, the GUI instance(s) must be restarted; this can be done without interrupting the system or environment controllers or altering the system state.

2.6.3 Commands

The system controller implements the standard GSI state model. It is a complex application and has many commands.

Within the "Operational" state, it has three modes:

1. Idle. The camera is not reading out images, but commands are still accepted and monitor points are publishing.
2. Acquisition. The camera is reading out full frame images using the manufacturer's conventional clocking scheme. This state is intended for acquisition of celestial objects – that is, getting an object image onto the correct CCD pixels so that the fast tip-tilt mode can be activated.
3. Fast tip-tilt. The camera is continuously reading out a predefined subframe. The readout can use the manufacturer's conventional clocking scheme, or a customised scheme optimised for speed. This state is intended to acquire images with sufficient speed to close the servo loop.

Major commands involve changing between modes. These include:

- *startAcquireRun* Change to acquisition mode and read out full frame continuously. (Synchronous)
- *getObject* Change to acquisition mode, acquire and return when object is acquired. (Asynchronous)
- *aquireAndGuide* Change to acquisition mode and acquire until *stopRun* is called. (Synchronous)

- *acquireAndStartTipTiltRun* Change to acquisition mode, acquire object, then change to fast tip-tilt mode with servo closed. (Asynchronous)
- *startTipTiltRun* Change to fast tip-tilt mode with servo closed. (Synchonous)
- *startTipTiltRunNoServo* Change to fast tip-tilt mode with servo open. (Synchronous)
- *stopRun* Stop any ongoing readout. (Synchonous)

For example, a typical command sequence might be to place the system in acquisition mode and wait for a target to be acquired (*getObject*), then place the system in fast tip-tilt mode with the servo closed (*startTipTiltRun*), wait for science observations to complete, then place the system in idle mode (*stopRun*). If one was confident in the system's ability to acquire objects, one could call *acquireAndStartTipTiltRun* instead of the first two commands.

There are many other commands. These include the following:

- In any mode: Change the electron multiplying gain, the goal position on the EMCCD or inform the system whether the telescope is in tube offset mode or not. Provide a dark frame, flat field, or clock induced charge model to process camera frames with. Get the servo settings that are constant over the program run time, get the current servo parameter settings.
- In or switching to acquisition mode: Select or deselect the target object, change image decimation rate, change frame period and exposure time. Acquire a variety of standardised images for later use as dark frames or flat fields.
- In fast tip-tilt mode: Change which clocking scheme (custom or conventional) to use, the camera frame rate, whether the servo loop should be open or closed and what the servo parameters should be.

A complete provisional command list is included in Sec. 4.1.1.

By contrast, the environment controller has only one operational state, in which it publishes environment measurements, (optionally) controls a heater in the camera enclosure and (optionally) enables or disables camera operation. The command set can be summarised as follows:

- Set heater power control to manual or automatic. If manual, set the heater power.
- Set camera enable to manual or automatic. If manual, enable or disable the camera.
- Retrieve settings that are constant over the program run time.

A complete provisional command list is included in Sec. 4.1.2.

2.6.4 Monitoring

The monitor points used in the system and environment controllers are all scalars, arrays of scalars or images. There are no monitor points with complex data types or character strings. This is to maintain equivalence with dlmsg, which has a smaller set of transmissable data types than the Generic System Interface.

Monitor points are also all non-polled. This is for two reasons:

1. In the system controller, data is updated at a rate determined by the camera hardware. Polling this data could lead to aliasing, it is much better to push it to the system at the rate at which it is generated.
2. The dlmsg interface does not support polling and it simplifies the code if data is accessed with both interfaces in the same way.

Monitor points for the system controller can most conveniently be grouped by frequency of generation.

Information about the status of the system controller is sent at a rate that gives "instant" feedback on a human timescale (currently set to 10 Hz). Monitor points in this category include:

- The current camera settings and status.
- The camera settings that are to be used in acquisition mode and fast tip-tilt mode.
- Information on what state the system is in, what it is currently doing, and what it is about to do. For example, the *acquireAndStartTipTiltRun* command mentioned above will cause the *AcquireMode* monitor point to publish "true". It will also cause the *TipTiltPending* monitor point to publish "true". When an object is acquired and the system goes into fast tip-tilt mode, both monitor points start to publish "false" and the *FastTipTiltMode* monitor point starts to publish "true".
- Current parameter settings, such as the target position on the CCD, the rotation matrix used to convert between CCD and tip-tilt mirror axes, decimation settings, servo and filter parameters and whether a real camera instead of an internal simulation is in use.

High bandwidth data is sampled at the current camera frame rate, which is expected to be as high as 1kHz. If the current decimation rate is n, then data from every n frames is chunked together into arrays prior to publication to improve the network efficiency. The exception is the camera images themselves, only one in every n is published to save bandwidth and storage.

High bandwidth monitor points include:

- The currently calculated image centroid and the distance from the target position on the CCD.
- The current mirror demand and mirror position.
- An acquisition spatial seeing estimate, published only in acquisition mode.
- A fast tip-tilt spatial seeing estimate, published only in fast tip-tilt mode.
- A coherence time estimate, published only in fast tip-tilt mode.
- A raw camera image. Only one image is published for each chunk, the image contemporaneous with the earliest of the scalar data in the chunk.

If the system is idle, none of this information is published.

Finally, UTM offloads are published at the rate at which they are sent to the telescope, which is either a fixed rate (expected to be 1Hz) or the camera frame rate if that is lower.

The UTM offload monitor points are:

- The UTM offload values.

A provisional list of the system controller monitor points can be found in Sec. 4.2.1.

The Generic System Interface monitor points for the environment controller all generate scalars at a constant frequency, expected to be 1 Hz. The majority of these are environment data but there are a few status monitor points too. Environmental controller monitor points include:

- Temperature at locations on or within the camera enclosure, near the external optics and on the coolant entry and exit pipes.
- Humidity inside (and possibly outside) the enclosure, and calculated dew point.
- Flow rates of coolant and dry air.

- Heater power setting and whether the heater is on.
- Whether the system is enforcing a camera shutdown due to adverse environment conditions.

A provisional list of the environment controller monitor points can be found in Sec. 4.2.2.

2.6.5 System properties

Both the system and environment controllers have system properties that are set by the GSI during initialisation. These are values that remain constant during program execution. All of the system properties in these two programs are strings.

For the system controller, the system properties include:

- Preset bounds and standard values for electron multiplying gain, exposure time and period in acquisition and fast tip-tilt modes.
- The sizes of the subregions to be used in fast tip-tilt mode with conventional or custom clocking.
- Physical system parameters such as pixel size, EMCCD size and focal length of the system when the telescope is in or out of tube offset mode.
- Default servo and Kalman filter parameters.
- The period of UTM offloads.

A provisional list of system controller properties is in Sec. 4.3.1.

For the environment controller, the system properties include:

- Temperature and humidity bounds, both absolute and relative to other measurements, within which the camera is allowed to operate.
- Minimum air and coolant flow rates needed for the camera to be allowed to operate.

A provisional list of environment controller properties is in Sec. 4.3.2.

2.6.6 Publish/subscribe interface

Prior to publish/subscribe system support becoming available, the system controller and environment controller will communicate with each other via a dlmsg interface. However telescope subscriptions rely on the publish/subscribe system being available. Once the publish/subscribe system is running, the system controller will subscribe to the following:

- Telescope rotation matrix for conversion between EMCCD and FTTA coordinates.
- Other telescope rotation matrices and data needed for display annotation.
- Telescope lateral dispersion correction offset and off-axis offset.
- Permission to operate camera from environment controller.

The environmental controller will not subscribe to any publications.

2.7 Control/display GUI

The control/display 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 the separate offline analysis GUI (see Sec. 2.8). If the system and environment controllers are not being directed by an ISS supervisor, the control/display GUI is also used to control the FTT system.

There will be two versions of the control/display 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 (a subset of the FTT/NAS functionality, see Sec. 4.1.1. 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 versions will be almost identical in look and feel and share much of their code. Only the FLC GUI should be used with the FLC versions of the system and environment controllers (certain controls on the FTT/NAS GUI would cause unavailable commands to be issued). The two GUI versions may be used interchangably with the FTT/NAS versions of 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.

2.7.1 GUI Functionality

The preliminary FLC GUI is shown in Figure 6. The FTT/NAS GUI will be 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 window after they are acquired.

Text display of recent average and rms centroid estimates *(FLC GUI only)* This is not shown in the screenshot, but is likely to be positioned to one side of the image display.

Graph of recent centroid estimates This is not shown in the screenshot.

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.

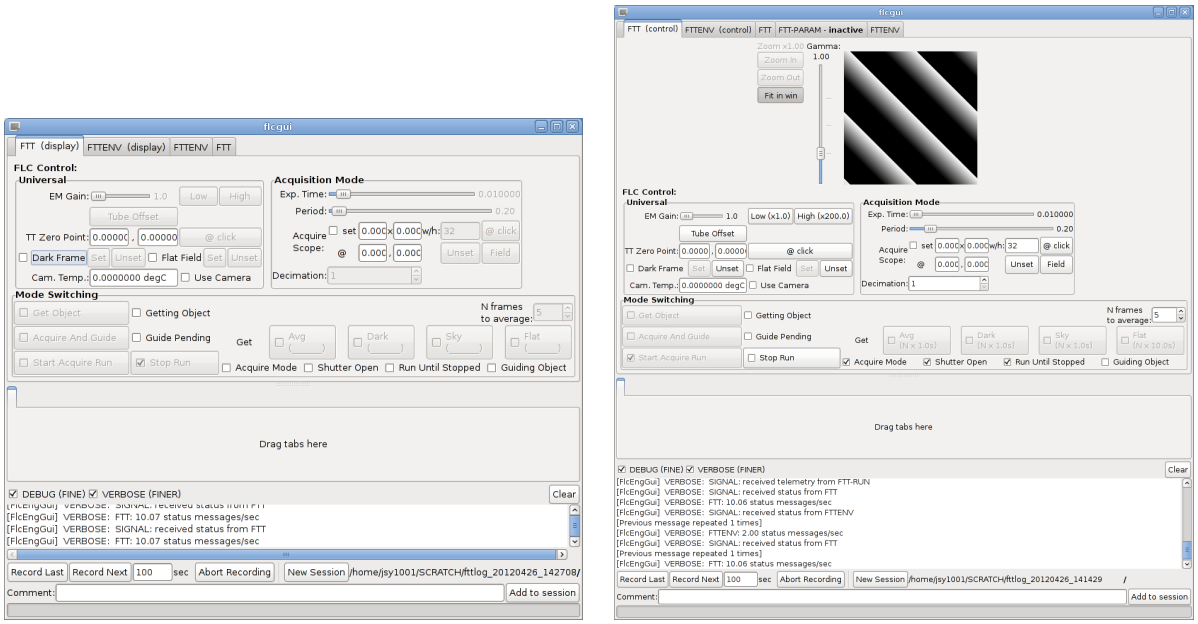


Figure 6: Screenshots of the current version of the FLC control/display GUI. Two views of the application main window are shown: on the left the GUI has been configured for display-only mode (note that most of the buttons are greyed-out) and the system controller is in idle mode; on the right the GUI is in standalone mode (buttons active) and the system controller is in acquisition mode (note that the greyscale image is a test pattern). We plan to 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.

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. The buttons correspond to the commands outlined in Sec. 2.6.3

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 (and eventually CSV for the FLC GUI) format. See below for more details.

Error 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 will 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 the enable state if manual.

2.7.2 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:

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

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.

Add (comment) to session : Append a user-specified comment string to the header of the current session group table (these comments are displayed by the analysis GUI).

2.7.3 GUI Implementation

The GUI applications are written in ANSI C and use the GTK+ widget library. Two third-party widget libraries are used to extend the core GTK+ functionality — **gtkimageview** (`http://projects.gnome.org/gtkimageview/`) is used to display greyscale images and **gtkdatabox** (`http://sourceforge.net/projects/gtkdatabox/`) will be used to plot graphs. The applications are structured as a collection of general-purpose GUI components (most are in common with the Cambridge delay line control GUI):

DataDisplay Displays dlmsg status and/or images for multiple messages sources in a Notebook widget.

- **StatusDisplay** Displays dlmsg status values from a single message source as text.
- **ImageDisplay** Displays greyscale images from a single message source.

MessageDisplay Displays human-readable log messages.

LogGui Data logging user interface.

The StatusDisplay component can either create a generic set of widgets to suit the content of the status messages, or can make use of a custom set of widgets created with the Glade interface designer software. The latter approach is used to implement the main system controller and environment controller tabs.

These in turn depend on a number of lower-level components which provide services and library functions. The GUI applications are single-threaded and use the GLib Main Event Loop to schedule socket input/output operations and user interactions with the GUI widgets. This approach has been used successfully for the MROI delay line user interface software.

2.7.4 Current Status

At PDR, a preliminary version of the FLC GUI has been written and tested using simple emulators of the system and environment controllers. The functionality for recording images and scalars to FITS and for commanding the systems has been fully implemented. The data display functionality is partially implemented: image display with a user-variable greyscale mapping works, as does textual display of monitor points. Annotation of images (with cross-hairs etc.), operator interaction with images (e.g. to select the target of interest), and graph plotting are yet to be implemented. We do not forsee any particular difficulties with completing the implementation.

2.8 Analysis GUI

2.8.1 Introduction

The FLC/FTT GUI is an analysis GUI which provides access to the FITS log files that can be generated by the user of the control/display GUI in order to record performance or analyse deficiencies in the operation of the FTT/NAS and telescope. In its simplest application it can be used to assess the images captured by the EMCCD camera during a particular log or in plotting and analysis of guiding corrections (useful for acceptance testing in FLC mode). Other uses include:

1. The review and plotting of seeing conditions;
2. The examination of image profile;
3. The measurement of image scales and distances between objects in the field of view;
4. The review of the FTT/FLC system parameters.

Although the GUI is not operating on real-time data it can be used as soon as the FITS log is available.

The GUI is run under the Matlab environment and is capable of spawning other GUIs to provide additional functionality. The main GUI deals with the selection and importing of FITS files or groups of recordings. These recordings are logs of FLC/FTT system operation, initiated by the user of the control/display GUI, and they are stored using the FITS Hierarchical Grouping Convention as described in Sec. 2.7.2. Basically a “Recording” is a collection of contemporaneous FITS log files associated with a particular log event. One or more recordings are stored in a “Session” which is a directory established at every start-up of the control/display GUI which can span up to one day.

Within the main GUI it is possible to examine parameters and tabulated data that are available in the selected recording and to plot data to Matlab figures or export the data to the Matlab workspace for further processing. In order to view images or sequences of images it is possible to plot them to a Matlab figure or to a movie figure as appropriate. To analyse images there is also a capability to spawn another Matlab GUI which specifically performs common analysis functions.

What is described in the following subsections is the current design and can be regarded as the minimum content that will be available.

2.8.2 Main GUI content and layout

A snapshot of a typical session on the main GUI is shown in Figure 7. There are three principal areas of the GUI. On the left of the GUI are selection list-boxes for loading an individual file or recording session together with information panels that give details of the file or recording group selected.

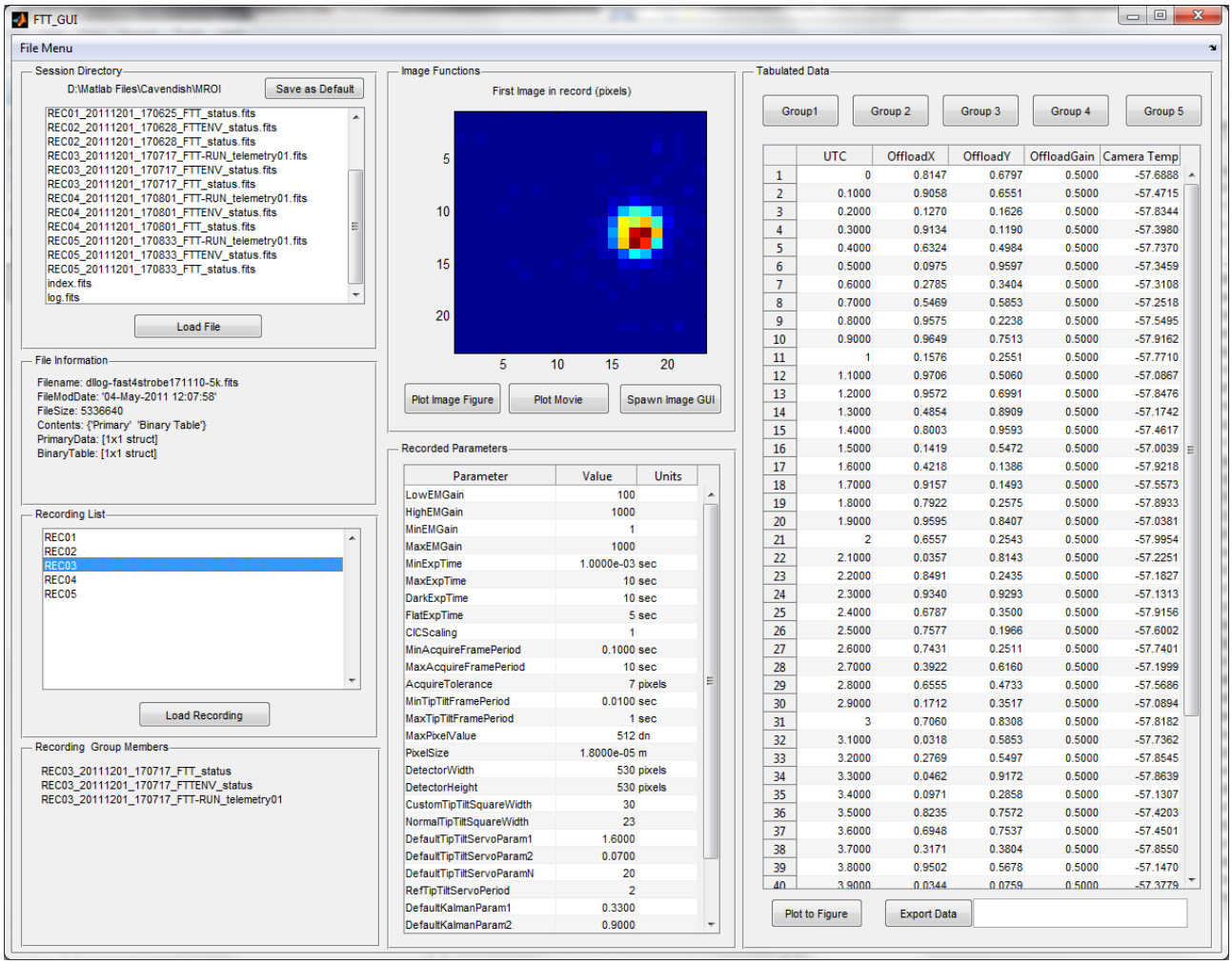


Figure 7: A snapshot of a typical session on the main GUI. On the left are selection list-boxes for loading an individual file or recording session and information from those files. The upper panel in the centre shows an image from the current file with options to plot it to a figure or to a movie or to spawn an image analysis GUI for more detailed processing. The lower centre panel is a scrollable table of current system parameters. The panel on the right displays a scrollable table of sample variables from the recording according to which group is selected by the push buttons above. Data can be selected in the table and plotted to a figure or exported to the Matlab workspace with a variable name provided by the user.

In the centre of the GUI there are two panels. The upper panel shows an image from the currently loaded file together with options to plot it to a figure or to a movie (in the case that the figure is the first in a sequence of frames) and there is also an option to spawn an image analysis GUI for more detailed processing. The lower panel features a scrollable table containing the static system parameters which are current for this recording. This is a list of fixed parameters as discussed in Sec. 2.6.5.

On the right hand side of the GUI there is a single panel for the display of sampled monitor points from the recording. As there may be many variables available they have been divided into groups for display in a scrollable table. The table always presents time in the left-most column and the remaining columns depend upon the group selected by the buttons above. It is possible for the user to select data from the table and plot it to a figure. It is also possible to export selected data to the Matlab workspace with a variable name provided by the user.

There is a menu bar at the top of the GUI which contains a number of useful commands such as *Print Setup*, *Print GUI*, *Close figures* and *Close GUI*.

2.8.3 Recording selection process

When the GUI starts the session directory list box is populated with the default directory listing and the user is able to navigate to and select a particular session (directory). Within each session is an index file which identifies all the recordings and files which belong to each recording. When a session is chosen, this index file is automatically selected and the Recording List-box is populated with the recordings available. The individual FITS files comprising the chosen recording are shown in the Recording Group Members panel while information on the file currently selected in the session directory is shown in the File Information panel. The user can choose to load a recording at this point or an individual FITS file can be selected in the session directory for viewing or processing a limited set of data. The act of loading a data file will clear the GUI of any data from a previous selection but will not clear figures previously spawned from the GUI nor any workspace variables created by the user. This allows comparison of the current data with that loaded previously.

2.8.4 Analysis GUI content and layout

This GUI, shown in Figure 8 is initiated from the main GUI and uses data loaded from the FITS log files. There are two panels: on the left is the full "Image" area and on the right is the "Region of Interest" area.

In the full image area, apart from the image itself, there are facilities for selecting the frame in the case that there is a sequence of frames and for playing the sequence at a specified target frame rate. Also there are facilities to select a specific colour map and to adjust background and contrast levels. An image histogram is also provided. For more flexible adjustments or analysis the selected frame can be exported to the Matlab workspace.

Within the ROI panel the user has the facilities to select a region of interest using the cursor in the full image and display the region in the smaller window. There are push-buttons to initiate the ROI selection, convert the ROI image to a contour map and back, to centroid the ROI area and to generate profiles through the ROI image. The centroid function computes the barycentre of the image and then displays an X and Y profile through the centroid position. The profile buttons allow the user to generate a cursor-defined profile.

Coordinates of the calculated centroid (referred to the full image) are overlaid on the ROI area. If centroid data for the selected frame was available from the recording these are overlaid on the full image area together with cross-hairs.

3 Conclusions

The design of all components of the FTT/NAS software is essentially complete, including definition of the system properties, commands, and monitor points that comprise the interface to the ISS. Coding of the system controller and control/display GUI is well advanced, and a precursor to the environment controller is complete and has been used extensively for our opto-mechanical stability tests. The design of the analysis GUI(s) have been successfully prototyped.

We believe we have already mitigated the major technical risks associated with the software. The real-time performance of the code has been demonstrated, and we have shown by simulation that a candidate fast tip-tilt correction algorithm can meet the limiting sensitivity requirement (with potential improvements to the algorithm yet to be tried).

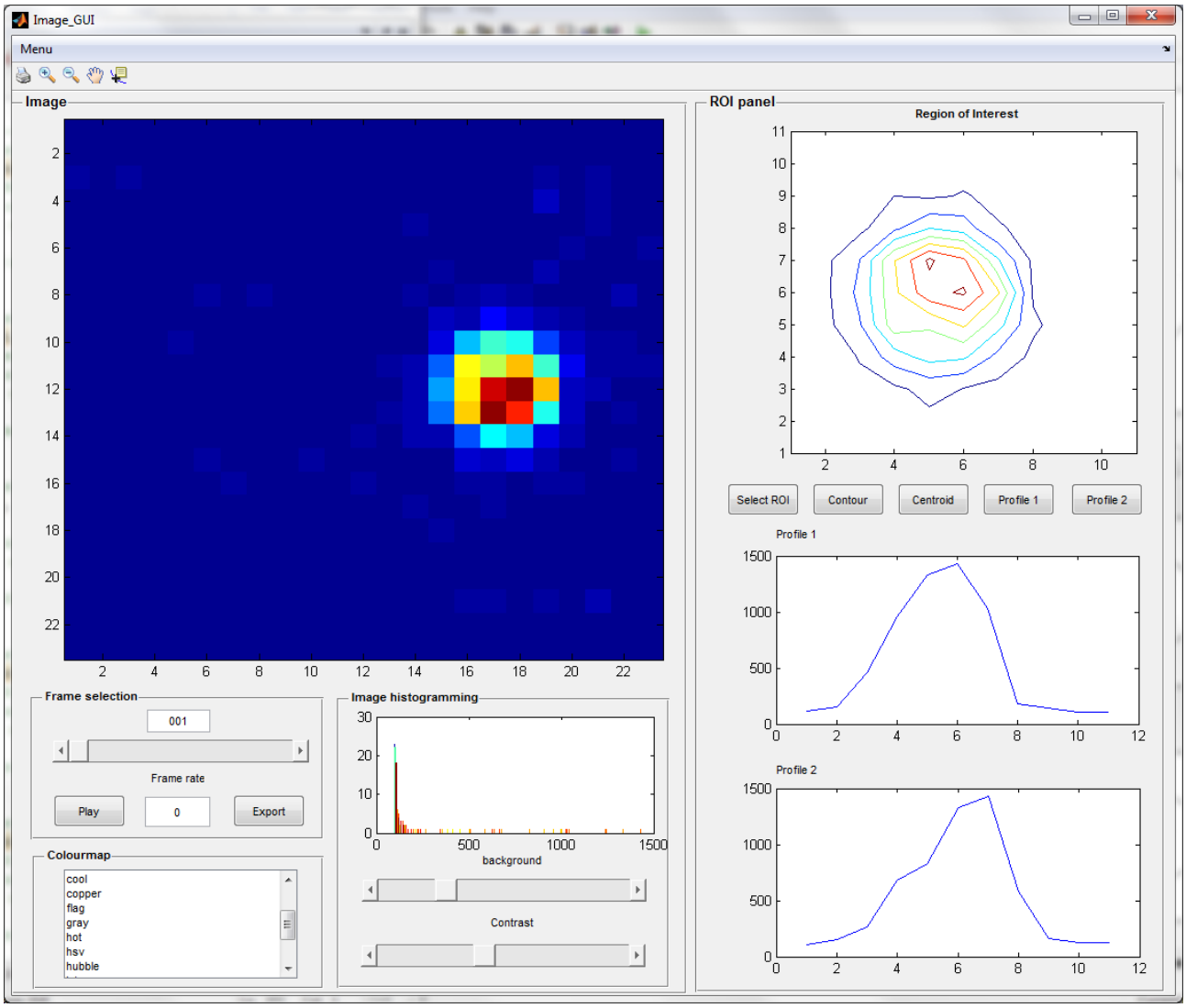


Figure 8: Snapshot of the image analysis GUI spawned from the main GUI. There are two panels: on the left is the full 'Image' area where there are facilities for selecting the frame or for playing a sequence of frames, to select a specific colour map, to adjust background and contrast levels and an histogram; on the right is the 'Region of Interest' area where there are push-buttons to initiate the ROI selection, to generate a contour map and to centroid the ROI area and to generate profiles.

The current version of our software is based on a preliminary standalone version of the MROI Generic System Interface framework. The most significant GSI changes we are anticipating over the coming weeks and months are as follows:

- Reconciliation of the C and Java versions of the GSI. We are not expecting this to result in major changes to the API.
- Changes to the exception handling behaviour and API.
- Definition and implementation of the Publish-Subscribe interface that will be used to interface the FTT control software with the UTCS (and to connect the system and environment controllers with each other, although an alternative interface will be provided initially).
- Implementation of the networked version of the C-language GSI.

- Revised and expanded documentation.

The uncertainties in the timescale and content of these changes may lead to an impact on the schedules for delivering the FLC and FTT releases of the software. However, the availability of the alternative dlmsg interface for controlling the system and capturing test data will allow many tasks to proceed while the GSI framework is in flux (the dlmsg interface is not completely independent though — the GSI standalone interface is needed to configure and initialize the system, and the command methods whose signatures are generated by the GSI will be wrapped by dlmsg equivalents).

In particular, we plan to mitigate the impact of future GSI releases by means of the following strategies:

- Implementation and testing of the fast tip-tilt and autoguiding algorithms does not depend on the details of the GSI interface and can proceed regardless (the dlmsg interface can be used for all such tests).
- Development of the control/display GUIs and the analysis GUI does not depend on the GSI interface.
- For the FLC release at least, we plan to deliver code based on the standalone version of the GSI. This release will be suitable for acceptance testing of the UTM. Integration with the networked version will be left as a task for MRO.

A question which could be discussed at the review (and has been raised previously) is whether there should be a separate acceptance/integration activity for the FLC software that takes place before delivery of the FLC hardware. MRO would need to purchase the FLC computer but not necessarily the EMCCD camera (owing to the emulation functionality that we have included). This would allow the software functionality to be demonstrated and provide MRO with an early opportunity to begin integration with the networked version of the GSI and with the UTCS. We would only be in favour of this if the integration work would actually proceed on the appropriate timescale.

4 Appendix: ISS interface command, monitor point and system property lists

Where an item is only applicable to the fast tip-tilt system, it is noted as such. Otherwise, items are common to both the first light camera and fast tip-tilt systems.

4.1 Commands

4.1.1 System controller commands

Mode parameters: universal

setEMGain Set electron multiplying gain.

setToLowEMGain Set electron multiplying gain to the LowEMGain preset value, see Sec. 4.3.1.

setToHighEMGain Set electron multiplying gain to the HighEMGain preset value, see Sec. 4.3.1.

setTubeOffsetOn Inform the system that the telescope is in tube offset mode.

setTubeOffsetOff Inform the system that the telescope is not in tube offset mode.

setTipTiltZeroPoint Set the target position (in EMCCD pixel coordinates, prior to application of telescope dispersion and off-axis offsets) for acquisition and fast tip-tilt. (Disabled in FLC edition).

Mode parameters: acquisition mode (full frame, conventional readout)

setAcquireScopeRect Set reference object to brightest object within predefined rectangle. A "reference object" is the image of the celestial object to be acquired and may not be the science target.

setAcquireScopeField Set reference object to brightest object in entire field.

unsetAcquireScope Unset reference object.

setAcquirePeriod Set the time between EMCCD acquisition frames.

setAcquireExpTime Set exposure time for each acquisition frame.

setAcquireDecimation Set the decimation n for acquisition frames, only one frame in every n will be sent over the network to conserve bandwidth.

Mode parameters: fast tip-tilt mode (pre-defined subframe, conventional or custom clocking)

setTipTiltCustomReadout Use custom clocking scheme for fast tip-tilt readout. (Disabled in FLC edition).

setTipTiltNormalReadout Use conventional clocking scheme for fast tip-tilt readout. (Disabled in FLC edition).

setTipTiltLoopOpen Open the fast tip-tilt servo loop. (Disabled in FLC edition).

setTipTiltLoopClosed Close the fast tip-tilt servo loop. (Disabled in FLC edition).

setTipTiltPeriod Set the time between fast tip-tilt frames. (Disabled in FLC edition).

setTipTiltDecimation Set the decimation *n* for fast tip-tilt frames, only one frame in every *n* will be sent over the network to conserve bandwidth. (Disabled in FLC edition).

setTipTiltBandwidth Set closed loop bandwidth by adjusting frame period and servo gain. Possibly also chooses between normal and custom clocking scheme. (Disabled in FLC edition).

kalmanFilterOn Turn on adaptive servo filter. (Disabled in FLC edition).

kalmanFilterOff Turn off adaptive servo filter. (Disabled in FLC edition).

setTipTiltServoParams Override current servo parameters. Parameters revert to defaults if bandwidth or tiptilt period change. (Disabled in FLC edition).

Mode Switching

startTipTiltRun Change to fast tip-tilt mode with servo closed. (Disabled in FLC edition).

startTipTiltRunNoServo Change to fast tip-tilt mode with servo open. (Disabled in FLC edition).

getObject Change to acquisition mode, acquire and return when object is acquired.

aquireAndGuide Change to acquisition mode and acquire until *stopRun* is called.

acquireAndStartTipTiltRun Change to acquisition mode, acquire object, change to fast tip-tilt mode with servo closed. (Disabled in FLC edition).

startAcquireRun Change to acquisition mode and read out full frame continuously.

stopRun Stops any ongoing readout.

getAvgFrame Returns a standardised camera frame that is an average of a preset number of frames.

getDarkFrame Like *getAvgFrame* but with the camera shutter closed.

getSkyFrame Like *getAvgFrame* but taken on the sky.

getFlatField Like *getAvgFrame* but taken against an (assumed) uniformly illuminated field.

Data Modifiers

setDarkFrame Use the provided image for dark subtraction.

unsetDarkFrame Do not do dark subtraction.

setFlatField Use the provided image for flat field correction.

unsetFlatField Do not do flat field correction.

useCICModel Use internal clock induced charge model to dark subtract when custom clocking. (Disabled in FLC edition).

ignoreCICModel Do not do clock induced charge subtraction. (Disabled in FLC edition).

Passive: system state is not affected

getFixedParams Retrieve settings that are constant over the program run time.

getTipTiltServoParams Retrieve servo parameters for current tip-tilt period (might use monitor points instead). (Disabled in FLC edition).

4.1.2 Environmental controller commands

Mode parameters: universal

setHeatAuto Allow the environment controller to control the heater state automatically.

setHeatManual Control the heater manually.

setHeatPower When manual heater control is on, set the heater power.

setCameraAuto Allow the environment controller to enable or disable the camera.

setCameraManual Manually enable or disable the camera.

setCameraEnable When manual camera control is on, enable the camera.

setCameraDisable When manual camera control is on, disable the camera.

Passive: system state is not affected

getFixedParams Retrieve settings that are constant over the program run time.

4.2 Monitor points

4.2.1 System controller monitor points

Low bandwidth data sent at 10Hz

TubeOffsetMode Boolean. True if the system has been told the telescope is in tube offset mode.

CameraReadout Boolean. True if the camera is reading out.

ShutterOpen Boolean. True if the shutter is open.

CameraTemp Floating point. The camera CCD chip temperature.

ExpTime Floating point. The currently set exposure time.

EMGain Integer. The electron multiplying gain, on an arbitrary scale set by Andor.

AcquirePeriod Floating point. The time between frames to be used in acquisition mode.

AcquireExpTime Floating point. The duration of exposures to be used in acquisition mode.

AcquireReadoutTime Floating point. The time required to read out an image in acquisition mode.

AcquireMaxExpForPeriod Floating point. Maximum possible exposure time available for *AcquirePeriod*.

AcquireDecimation Integer. The decimation rate, used to reduce the number of acquisition mode images sent.

TipTiltPeriod Floating point. The time between frames to be used in fast tip-tilt mode. (Disabled in FLC edition).

TipTiltExpTime Floating point. The duration of exposures to be used in fast tip-tilt mode. (Disabled in FLC edition).

TipTiltReadoutTime Floating point. The time required to read out an image in fast tip-tilt mode. (Disabled in FLC edition).

TipTiltDecimation Integer. The decimation rate, used to reduce the number of fast tip-tilt mode images sent and to determine how much high bandwidth data should be chunked together before it is sent. (Disabled in FLC edition).

ReadoutSquareX Integer. X pixel coordinate of fast tip-tilt square readout region. (Disabled in FLC edition).

ReadoutSquareY Integer. Y pixel coordinate of fast tip-tilt square readout region. (Disabled in FLC edition).

AcquireRectX Integer. X pixel coordinate of acquisition rectangle.

AcquireRectY Integer. Y pixel coordinate of acquisition rectangle.

AcquireRectWidth Integer. Pixel width of acquisition rectangle.

AcquireRectHeight Integer. Pixel height of acquisition rectangle.

OffloadGain Floating point. Multiplier for telescope offload values prior to being sent for telescope correction. (Disabled in FLC edition).

TipTiltZeroPointX Floating point. X pixel coordinate of target closed loop servo position, prior to application of telescope dispersion and off-axis offsets.

TipTiltZeroPointY Floating point. Y pixel coordinate of target closed loop servo position, prior to application of telescope dispersion and off-axis offsets.

ObjectivePointX Floating point. X pixel coordinate of target position, after application of telescope dispersion and off-axis offsets.

ObjectivePointY Floating point. Y pixel coordinate of target position, after application of telescope dispersion and off-axis offsets.

RotG11 Floating point. Element G_{11} of the rotation matrix that converts from CCD axes to tip-tilt mirror axes. This depends on the telescope pointing.

RotG12 Floating point. Element G_{12} of the rotation matrix.

RotG21 Floating point. Element G_{21} of the rotation matrix.

RotG22 Floating point. Element G_{22} of the rotation matrix.

AcquireScopeSet Boolean. True if acquisition scope is set.

AcquireMode Boolean. True if the system is in acquisition mode.

FastTipTiltMode Boolean. True if the system is in fast tip-tilt mode.

GettingObject Boolean. True if the *getObject* command has been issued (Sec. 4.1.1) and has not yet returned.

AcquiringAvg Boolean. True if the system is in the process of acquiring data for an average frame.

AcquiringDarkAvg Boolean. True if the system is in the process of acquiring data for a dark average frame.

AcquiringSkyAvg Boolean. True if the system is in the process of acquiring data for a sky average frame.

AcquiringFlatAvg Boolean. True if the system is in the process of acquiring data for a flat field.

GuidingObject Boolean. True if the system is in acquisition mode, has successfully acquired an object, and is continuing to keep it acquired.

RunUntilStopped Boolean. True if a command is issued that causes the camera to read out indefinitely, such as *startTipTiltRun*.

TipTiltPending Boolean. True if a command is issued that will eventually cause the system to change to fast tip-tilt mode (currently only *aquireAndStartTipTiltRun*). (Disabled in FLC edition).

GuidePending Boolean. True if a command will eventually cause the system to guide in acquisition mode (currently only *acquireAndGuide*).

DarkFrameSet Boolean. True if a dark frame (which could be a sky frame) is being used to do dark frame subtraction.

FlatFieldSet Boolean. True if a flat field is being used to do flat field corrections.

CICModelUsed Boolean. True if an internal clock induced charge model is being used to correct for clock induced charge noise when cuctom clocking mode is being used. (Disabled in FLC edition).

TipTiltBandwidth Floating point. The current fast tip-tilt loop bandwidth. (Disabled in FLC edition).

TipTiltLoopClosed Boolean. True if the fast tip-tilt loop is closed. (Disabled in FLC edition).

TipTiltCustomReadout Boolean. True if the custom clocking scheme is currently used to read out CCD images. (Disabled in FLC edition).

KalmanFilterOn Boolean. True if the Kalman predictive filter is on. (Disabled in FLC edition).

TipTiltServoParam1 Floating point. Placeholder for servo tuning parameters. (Disabled in FLC edition).

TipTiltServoParam2 Floating point. Placeholder for servo tuning parameters. (Disabled in FLC edition).

TipTiltServoParamN Floating point. Placeholder for servo tuning parameters. (Disabled in FLC edition).

KalmanParam1 Floating point. Placeholder for Kalman predictive filter tuning parameters. (Disabled in FLC edition).

KalmanParam2 Floating point. Placeholder for Kalman predictive filter tuning parameters. (Disabled in FLC edition).

KalmanParamN Floating point. Placeholder for Kalman predictive filter tuning parameters. (Disabled in FLC edition).

AllowCustomReadout Boolean. True if custom clocking is allowed in fast tip tilt mode. (Disabled in FLC edition).

UseCamera Boolean. True if a real camera (instead of an internal simulation) is in use.

Offload data

OffloadX Floating point. X value of UTM offload.

OffloadY Floating point. Y value of UTM offload.

High bandwidth data acquired at camera frame rate, sent in chunks at decimated frame rate

CentroidPixX Floating point array. X coordinate of calculated image centroid.

CentroidPixY Floating point array. Y coordinate of calculated image centroid.

CentroidValid Boolean array. True if the system thinks that the calculated centroid value is valid.

OffsetX Floating point. The X angle between the current centroid and the target position.

OffsetY Floating point. The Y angle between the current centroid and the target position.

MirrorTipDemand Floating point. The voltage being sent to the mirror tip axis. (Disabled in FLC edition).

MirrorTiltDemand Floating point. The voltage being sent to the mirror tilt axis. (Disabled in FLC edition).

MirrorTip Floating point. The mirror tip axis position. (Disabled in FLC edition).

MirrorTilt Floating point. The mirror tilt axis position. (Disabled in FLC edition).

AcquireSeeingFWHM Floating point. Spatial seeing estimate during acquisition.

AcquireSeeingErr Floating point. Estimated error in *AcquireSeeingErr*.

TipTiltSeeingFWHM Floating point. Spatial seeing estimate in fast tip-tilt mode. (Disabled in FLC edition).

TipTiltSeeingErr Floating point. Estimated error in *TipTiltSeeingFWHM*. (Disabled in FLC edition).

CoherenceTime Floating point. Estimate of the coherence time. (Disabled in FLC edition).

CoherenceTimeErr Floating point. Estimated error in *CoherenceTime*. (Disabled in FLC edition).

Frame Integer 2D array. Camera image.

4.2.2 Environmental controller monitor points

Data published at 1Hz

CameraCaseTemp Floating Point. Temperature of the camera case.

EnclosureAirTemp Floating Point. Air temperature inside the camera enclosure.

EnclosureSurfTemp Floating Point. Air temperature on the external surface of the camera enclosure.

ColdPlateTemp Floating Point. Air temperature on the cold plate inside the camera enclosure.

HeaterTemp Floating Point. Temperature of the heater inside the camera enclosure.

CoolantInTemp Floating Point. Temperature of coolant prior to entering the camera enclosure.

CoolantOutTemp Floating Point. Temperature of coolant as it exits the camera enclosure.

TableTemp Floating Point. Temperature of the optical table.

BaseplateTemp Floating Point. Temperature of the fast tip-tilt baseplate.

ShieldTemp Floating Point. Temperature of the shield protecting the optics.

Humidity Floating Point. Humidity inside the camera enclosure.

Dew Point Floating Point. Calculated dew point inside the cameras enclosure.

CoolantFlowRate Floating Point. Coolant flow rate.

AirFlowRate Floating Point. Flow rate of dry air blown onto the camera window.

HeaterOn Boolean. True if the heater is on.

HeaterPower Floating Point. The power used by the heater.

ForceShutdown Boolean. True if the environment controller is enforcing a camera shutdown.

4.3 System properties

4.3.1 System controller system properties

LowEMGain Standardised low electron multiplying gain.

HighEMGain Standardised high electron multiplying gain.

MinEMGain Minimum allowed electron multiplying gain.

MaxEMGain Maximum allowed electron multiplying gain.

MinExpTime Minimum allowed exposure time.

MaxExpTime Maximum allowed exposure time.

AvgExpTime Exposure time used for acquiring frames for averaging.

DarkExpTime Exposure time used for acquiring dark frames for averaging.

FlatExpTime Exposure time used for acquiring flat fields for averaging.

CICScaling Multiplier to apply to internal clock induced charge model prediction. (Disabled in FLC edition).

MinAcquirePeriod Minimum allowed frame period in acquisition mode.

MaxAcquirePeriod Maximum allowed frame period in acquisition mode.

AcquireTolerance Threshold that determines when an object is considered to be acquired.

MinTipTiltPeriod Minimum allowed frame period in fast tip-tilt mode. (Disabled in FLC edition).

MaxTipTiltPeriod Maximum allowed frame period in fast tip-tilt mode. (Disabled in FLC edition).

MaxPixelValue Maximum pixel value allowed by the camera hardware.

PixelSize Length of the edge of each pixel.

DetectorWidth Width of the sensitive area of the CCD.

DetectorHeight Height of the sensitive area of the CCD.

CustomTipTiltSquareWidth Width in pixels of the square used with the custom clocking scheme in fast tip-tilt mode. (Disabled in FLC edition).

NormalTipTiltSquareWidth Width in pixels of the square used with the conventional clocking scheme in fast tip-tilt mode. (Disabled in FLC edition).

DefaultTipTiltServoParam1 Placeholder for default fast tip-tilt servo tuning parameter. (Disabled in FLC edition).

DefaultTipTiltServoParam2 Placeholder for default fast tip-tilt servo tuning parameter. (Disabled in FLC edition).

DefaultTipTiltServoParamN Placeholder for default fast tip-tilt servo tuning parameter. (Disabled in FLC edition).

RefTipTiltServoPeriod Standardised frame period for the tip-tilt servo for which the default servo parameters apply. (Disabled in FLC edition).

DefaultKalmanParam1 Placeholder for default Kalman filter tuning parameter. (Disabled in FLC edition).

DefaultKalmanParam2 Placeholder for default Kalman filter tuning parameter. (Disabled in FLC edition).

DefaultKalmanParamN Placeholder for default Kalman filter tuning parameter. (Disabled in FLC edition).

CameraFocalLength Focal length of the local fast tip-tilt lens.

UTFocalLength Focal length of the combination of telescope and local lens.

OffloadPeriod Period of UTM offloads sent to telescope.

4.3.2 Environmental controller system properties

EnclosureAirMaxTemp Maximum allowed air temperature within the camera enclosure.

EnclosureAirMinTemp Minimum allowed air temperature within the camera enclosure.

EnclosureAirMinPassiveTemp Minimum allowed air temperature within the camera enclosure to allow the camera to be turned on.

EnclosureMinHumidity Minimum allowed humidity within the camera enclosure.

EnclosureMaxHumidity Maximum allowed humidity within the camera enclosure.

ColdPlateMaxBelowAmbient The cold plate must be at least this close to the enclosure air temperature.

ColdPlateMinAboveDewPoint The cold plate must be warmer than the dew point by this amount.

EnclosureSurfMaxAboveAmbient The external enclosure surface must be less than this amount above the ambient temperature.

EnclosureSurfMaxBelowAmbient The external enclosure surface must be less than this amount below the ambient temperature.

EnclosureSurfMaxPassiveBelowAmbient The external enclosure surface must be less than this amount below the ambient temperature before the camera can be turned on.

MinCoolantFlow The minimum coolant flow needed.

MinAirFlow The minimum air flow needed.