SPIRou TipTilt & Viewing Software Infrastructure
Final Design
SPIROU-4900-CFHT-RP-00569-V1.3

Prepared by: Tom Vermeulen	**Date**: Feb 12, 2014

Keywords	Control/Command, Software, Hardware, Guider, Tip/Tilt, ISU, TCS, Cassegrain, Agent, Status Server, Director

Summary	This document presents a final design for the SPIRou software infrastructure that will be used for both target acquisition and tip/tilt guiding.

Version	Date	Comments
1.0	Sept 21, 2012	Preliminary Design Release.
1.1	Jan 3, 2014	Final Design Draft Release.
1.2	Feb 10, 2014	CFHT Internal Final Design Release.
1.3	Feb 12, 2014	SPIRou Project Team Final Design Release.

Table of Contents

List of Figures

1 Purpose

The purpose of this document is to provide a final design for the SPIRou software infrastructure that will be used for both target acquisition and tip/tilt guiding.

2 Scope

This document provides a detailed design for the software infrastructure that is used for target acquisition and tip/tilt guiding with SPIRou. In particular this document focuses on the details of the software designed by CFHT. This document does not contain the details of the actual guide image processing, FLIR camera control, or centroiding algorithms used to determine the voltage corrections sent to the Image Stabilization Unit (ISU). More details for this can be found in SPIROU-4200-ASIAA-RP-00527 and SPIROU-4200-ASIAA-RP-00646. This document also does not contain the details of the ISU control interface or the viewing analysis used to calculate the overall centering of the science object on the entrance aperture based on a long sequence of images. Please reference SPIROU-4100-IRAP-RP-00638 for more details.

3 System Architecture

3.1 Overall Software Architecture

Figure 1 illustrates the SPIRou software architecture spanning the full life-cycle from PI proposal submission to data distribution. The figure shows a highlighted breakout of the components associated with the guiding subsystem. This diagram is included in order to show where guiding system resides within the context of the overall software architecture. More details regarding the overall software architecture can be found in the SPIROU-4900-CFHT-RP-00590 high-level software preliminary design document.

3.2 Target Acquisition and Guiding Architecture

Figure 2 illustrates the SPIRou architecture used for target acquisition and tip/tilt guiding. This diagram shows not only the different subcomponents, but also which groups are responsible for each subcomponent. This document will focus primarily on those areas highlighted as being under the responsibility of CFHT. The following sections will provide a high level overview of each of the subcomponents identified in figure 2. Those items that are identified as the responsibility of CFHT are discussed in further detail in subsequent sections of the document.

3.2.1 Computing Hardware

The diagram in figure 2 contains two light blue boxes for the session host and the guide host computers. Both computers will run a Linux operating system known as “Sidious”. This distribution was created by Sidik Isani during his time as an employee of both CFHT and IfA. The session host will likely utilize an existing computer at CFHT while the guide host will be a new Linux computer. The session host computer will reside in the 4th floor computer room at the observatory.

The guide host computer will reside within an electronics rack mounted on the cassegrain location of the telescope. The current plan is to use a 1U form factor Casetronic C146 rack-mount casing with dimensions of 482.6mm x 44.5mm x 381mm (width x height x depth). The motherboard will be a VIA EPIA-M899 board. The peak real power usage at 100% with no GPU activity and a mechanical 2.5” HDD is 36.76 watts. Idle power usage is 18 watts. To improve the reliability of the system the hard disk drive will be replaced with a solid state drive. This change will also reduce the overall power usage. This computer will be populated with a PowerDAQ PCI card to communicate with

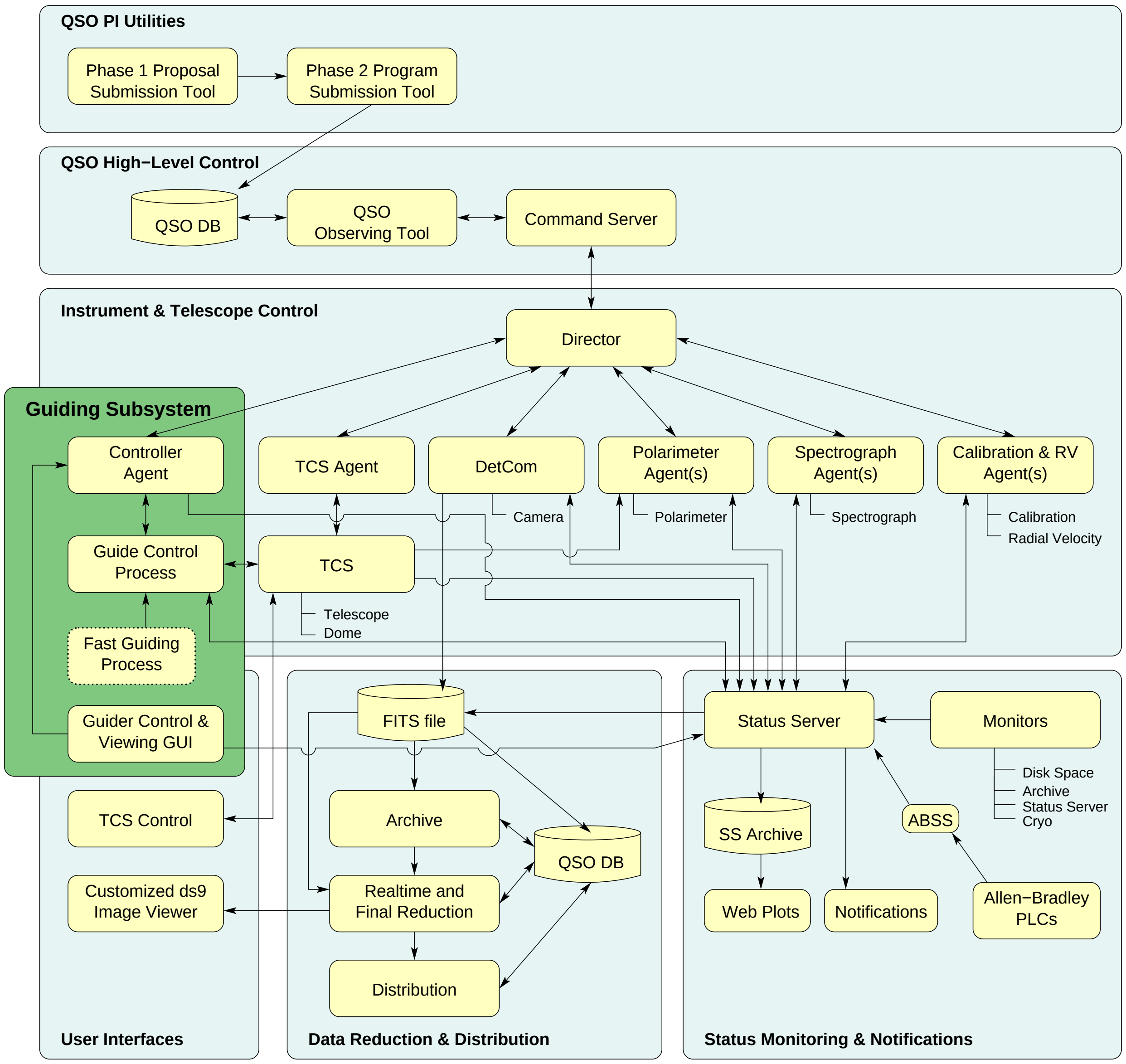


Figure 1: Overall Software Architecture for SPIRou

the ISU and an EDT CameraLink PCI express card to communicate with the Infrared detector. In order to calculate an approximate total thermal power load for the machine the following equation can be used: (36.76 + pci1 + pci2) * 1.25. The constant value of 1.25 is a reasonable factor to account for the inefficiency of the AC power supply.

The guide host computer will run a Linux operating system based on either the 2.6 or 3.2 kernel. CFHT used a Linux 2.4 kernel built with the RTAI real-time extension as the operating system for the guide host computer with WIRCam. Based on advancements in both the preemptive nature of recent kernels and hardware technology a real-time kernel should no longer be necessary with SPIRou. In addition, the EDT CameraLink card that is currently identified will not support real-time Linux. Given the speed of the currently identified guide host computer, as well as the available high resolution timer, using a non-real-time conventional Linux kernel should be sufficient to achieve a 50Hz to 100Hz guide rate with sufficiently accurate timing on this hardware.

In order to support the argument that a real-time kernel won't be necessary, a rackmount machine with the previously specified hardware was purchased. For initial testing the machine was loaded with Ubuntu Linux 10.04 LTS (2.6.32

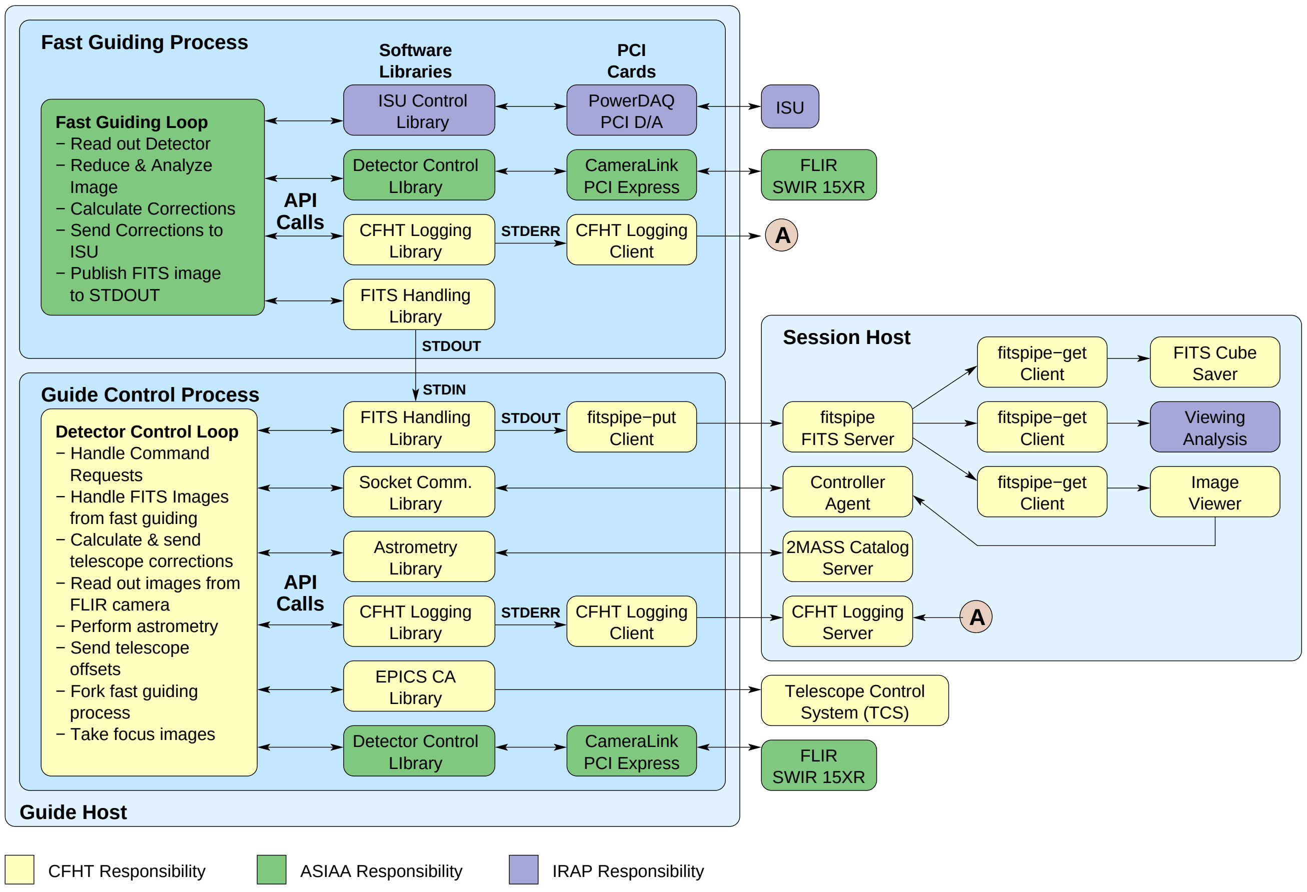


Figure 2: SPIRou Guiding Architecture

kernel). The Cyclictest (https://rt.wiki.kernel.org/index.php/Cyclictest) timer testing program was installed and run. Using a timer interval of 10 microseconds, the latency between the expiration of a timer and the time the interrupt was handled was on average 4 microseconds. More benchmarking and testing will be performed. However, the initial timer latency results appear to be well within the margin required for guiding intervals of 10 to 20 milliseconds.

While the guide host computer is used during normal operations there should be no need to write to the root disk on the machine. As a result, the root disk will be remounted read-only once the machine has finished its boot sequence. This will be done to prevent possible disk corruption in case power is cycled without the benefit of a clean shutdown. Additionally, since a solid state drive is used (more reliable at high altitude), preventing excessive disk writes will extend the life of the drive. This is normally done by adding the following line to the /etc/rc.local boot up script.

```
mount -o remount,ro /
```

3.2.2 Guide Control Process

This is the primary process running on the guide host computer. This process is responsible for the following activities. Each of the noted activities is described at a high-level here. More details are available in section 5.

- **Handle command requests** - Command requests are sent to the guide control process by the Controller Agent running under Director.

- **Read out images from FLIR camera** - For basic imaging, astrometry, and focus sequences images must be read from the FLIR camera. Based on the camera specifications in figure 22 the exposure time for the camera ranges from 50ns to 500ms. As a result, if images are requested for an exposure time longer than 500ms, these images will be created by stacking a sequence of images with a shorter exposure times.
- **Perform astrometry** - When moving to a new target from a science productivity standpoint it is important to quickly identify the field and place the science target on the entrance aperture. Given that the field of view (FOV) of the FLIR detector as measured on the sky is roughly 82.1 x 65.7 arcseconds this means that images may need to be taken at multiple telescope offsets in order to properly recognize the field. Using the combination of a stitched image from multiple offset positions, the science object RA, Dec, and magnitude along with 2MASS catalog query details, it should be possible to automatically position the requested science target on the entrance aperture for many requested pointings. Whenever this isn't possible, the observer must use the stitched acquisition image and PI supplied finding charts to manually move the telescope to place the science object on the entrance aperture.
- **Take focus images** - As with ESPaDOnS, two long exposures on a star will be taken approximately .5mm on either side of what is thought to be the current best focus. These images are analyzed using a parabolic fit in order to calculate the ideal focus for the instrument. Given the short exposure times offered with the FLIR camera the long exposure times will be achieved through a stacked sequence of short exposures over the space of 30 seconds or so.
- **Fork fast guiding process** - When the science target is accurately placed on the entrance aperture, the guide control process is forked to handle the fast guiding. Once the child process is forked, its STDOUT is piped to STDIN of the parent process so guiding images can flow from the Fast Guiding Process to the Guide Control Process. It is important to note that the lifetime of the Fast Guiding Process only spans the time that guiding is in progress.
- **Handle FITS images from Fast Guiding Process** - Raw guiding images from the guide camera will be packaged as FITS files and sent to STDOUT. These images will consequently be received via STDIN in the Guide Control Process. The Guide Control Process performs the necessary processing of the header information and sends the images to STDOUT where they are picked up by the fitspipe-put client.
- **Calculate and send telescope corrections** - The guide control process will use the ISU correction and current position information contained within the FITS headers of the images from the Fast Guiding Process as a basis for determining the corrections that must be sent to the TCS. Corrections are sent via an EPICS channel access interface to the Telescope Control System (TCS) at a 1Hz frequency.

In addition to the previously noted activities within the Detector Control Loop of the Guide Control Process, there are several libraries and piped processes that are referenced in figure 2. Each of these items is addressed in the following subsections.

3.2.2.1 FITS Handling Library (libfh)

This library, commonly referred to as libfh, was created at CFHT and is used for the bulk of all FITS file manipulation within the instrument software used at CFHT. It is CFHT's intention that this library will be used for all necessary FITS manipulation performed within SPIRou software as part of the Control Command subproject. Documentation for libfh can be found at http://software.cfht.hawaii.edu/libfh.

3.2.2.2 Socket Communication Library (libsockio)

This library is designed for clients and servers that must pass messages to each other over a socket. If used as intended, it is possible to create fast and responsive single-threaded multi-client socket servers with only a few lines of code needed to manage the socket communication. In this case, the libsockio library will be used as the communication core in order to add a server listening socket port for the Guide Control Process. The listening port will receive requests from the Controller Agent running under director on the session host. Documentation for libsockio can be found at http://software.cfht.hawaii.edu/sockio.

3.2.2.3 Astrometry Library

With WIRCam software was written to map stars detected on the focal plane with the 2MASS catalog in order to correct for errors in telescope pointing. The software utilities created with WIRCam were migrated to a library where they have been extended for use with both MegaCam and ESPaDOnS. This library will be re-used for SPIRou.

3.2.2.4 CFHT Logging Library

Any diagnostic messages generated by the software should be logged using the libcfht logging calls. Using this facility helps to provide consistency within the message logging and enables diagnostic messages to be captured in a central repository. Logging API calls use a standard printf-style message format. An example of a logging call is as follows.

```
cfht_logv(CFHT_MAIN, CFHT_WARN,
          "(%s:%d): unable to open %s directory in the Status Server"
          ": %s", __FILE__, __LINE__, ss_name, ssGetStrError());
```

Messages can be logged using any one of the following message types: CFHT_ERROR, CFHT_WARN, CFHT_STATUS, CFHT_LOGONLY, and CFHT_DEBUG.

When the cfht_logv API call is used on the guide host, this will cause formatted logging messages to be generated to STDERR. When the Guide Control Process is started, it will be initiated in such a way that the error messages are piped to STDIN of a separate CFHT logging client program.

3.2.2.5 EPICS Channel Access (CA) Library

Communication must occur with TCS in order to offset the telescope, enable guiding, and send guiding corrections. Both requests and status queries will occur via an existing CFHT EPICS Channel Access library interface.

3.2.2.6 fitspipe-put Client

This is a relatively simple process created by the Institute for Astronomy (IfA) PanSTARRS development team. It takes FITS images from STDIN and relays them via a socket connection to the fitspipe FITS server. The underlying socket communication is performed via the libsockio socket library. More information regarding the usage of the fitspipe-put client can be found in section 6.4.

3.2.2.7 CFHT Logging Client

This is a simple process that accepts line-based log message input and communicates with a logging server on the session host in order to place the log messages in a log file. Since the root file system on the guide host is mounted read-only, it is important that logging messages are offloaded to the session host.

3.2.2.8 Detector Control Library

This is a C API library produced by ASIAA. The library will provide an API to send commands to the FLIR camera via a CameraLink interface and provide the capability to retrieve image data from the camera. This library is used within both the Guide Control Process and the Fast Guiding Process. More details can be found in SPIROU-4200-ASIAA-RP-00646.

3.2.3 Fast Guiding Process

This process is created from a fork of the Guide Control Process whenever guiding is requested. This process is responsible for the following activities.

- **Read out the Detector** - Each iteration of the guide loop requires that a subraster of the FLIR detector be read out.
- **Reduce the Image and Calculate the Star Center** - It may not be possible to use the raw subraster as is. Instead, image processing techniques may need to be applied in order to accurately calculate the center of the point source.
- **Send Corrections to the ISU** - Once the center of the point source is determined, this can be compared to the X & Y null positions in order to calculate the ISU correction. This correction is then converted to an analog voltage and sent to the ISU.
- **Publish FITS Image to STDOUT** - The raw image read from the detector along with metadata is packaged as a FITS file and published to STDOUT.

This list of activities is performed within a single-threaded loop inside the Fast Guiding Process.

Design details for the Fast Guiding Process can be found in SPIROU-4200-ASIAA-RP-00646 and design details for the ISU interface can be found in SPIROU-4100-IRAP-RP-00638.

One of the advantages of using a modular architecture is so development within the various organizations can be performed without requiring that all the necessary dependencies are in place. For example, the Guide Control Process has a number of dependencies based on the observing environment at CFHT. However, these dependencies are not necessary for the development of the Fast Guiding Process. Figure 3 illustrates the software architecture that can be used for the development of the components within the Fast Guiding Process.

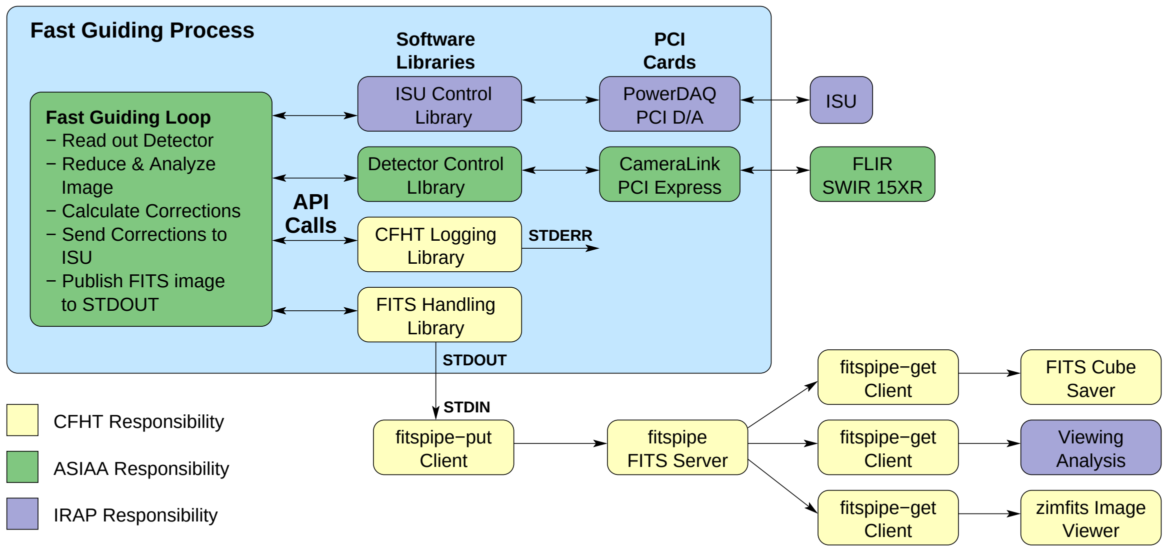


Figure 3: SPIRou Fast Guiding Development Architecture

As one can see from the diagram, the logging messages can be redirected to STDERR and the FITS files that are generated by the Fast Guiding Process can be redirected to the FITS Server directly instead of needing to flow through the Guide Control Process. In addition, aside from the FITS Cube Saver, noted in the diagram, all the other pieces of CFHT software within figure 3 already exist. FITS images can initially be viewed using the uncustomized zimfits image viewing utility developed by the Institute for Astronomy (IfA). Once the CFHT SPIRou image viewer is complete it can be used instead of zimfits.

In order to ensure that the interface works properly when the Fast Guiding Process is integrated with the Guide Control Process later in the development cycle, there are a few items that must be followed.

The Fast Guiding Process will be invoked as a forked copy (child process) of the Guide Control Process. This implies that the Fast Guiding Process will inherit an exact copy of the Guide Control Process. As a result, STDERR will already be properly set up to pipe error messages to the CFHT Logging Client. Once it is invoked, STDOUT of the Fast Guiding Process (child process) will need to be redirected to STDIN of the Guide Control Process (parent process).

The Fast Guiding Process must be designed in such a way that it can easily be run as a child process. First and foremost, this means that the Fast Guiding Process is written in C. Secondly, the Fast Guiding Process should be written in such a way that it has both a guideExecute() and guideCleanup() routine defined. When the process is forked, the Guide Control Process will ensure that it has closed its connection to the FLIR camera. As a result, when the guideExecute() routine is called, following the fork, the Fast Guiding Process should be able to initialize its internal state and establish a connection to the FLIR camera. When guiding is stopped, the Fast Guiding Process will be terminated. As a result, the Fast Guiding Process should be designed to intercept the SIGTERM signal and trigger the guideCleanup() routine in order to cleanly close all resources. For initial testing purposes as a standalone process, the main() routine should be set up to make a single call to guideExecute().

3.2.3.1 ISU Control Library

From a software perspective, the control of the ISU should be pretty simple. The ISU is driven by applying two analog voltages (one for each axis). These voltages are received by the ISU control hardware. The servo control system within the ISU control hardware initiates the actual movement of the tip/tilt plate. In order to generate the voltages, IRAP has specified the use of a PowerDAQ PD2-AO-8/16 PCI card. The data sheet for this card can be found in figure 20 within the appendix of this document. Both Linux drivers and a C API library are supplied along with the PCI card. The vendor supplied software comes in both source and binary format.

The Fast Guiding Process will utilize a C API library created by IRAP to send analog voltages to the ISU for setting the ISU position. The API is described in SPIROU-4100-IRAP-RP-00638.

3.2.4 Session Host

The session host computer contains several components that are part of the software infrastructure for SPIRou. These components are described at a high-level in the following subsections.

3.2.4.1 Controller Agent

This is a new agent that will be implemented for SPIRou. The role of the Controller Agent is to take single line instrument configuration and exposure commands and ensure that configuration operations are executed efficiently. The Controller Agent communicates with other agents, via director, in order to efficiently configure the instrument prior to exposure initiation. Below is a representative set of commands from ESPaDOnS that is similar to the type of commands that will be used with SPIRou.

```
tcoords 10:00:00.00 +20:00:00.0 2000.0 0.0 0.0 "Target Name"
gcoords select
go etype=OBJECT rmode=normal obsm=staronly stokes=I enum=1/1 etime=300.0
 raster="FULL" filename=odometer
```

In this command example, the “tcoords” command defines the coordinates of the target on the sky. The “gcoords select” command indicates that guiding is required. Finally, the “go” command specifies the desired instrument configuration as well as the exposure parameters. These three commands are used by the Controller Agent to perform the following sequence of operations.

- Initiate the necessary instrument configuration operations for the requested instrument mode.
- Follow the movement of the telescope and dome in order to know when both subsystems have finished moving.
- Initiate a target acquisition operation in order to place the PI requested target on the entrance aperture.
- Initiate a request to start closed loop guiding.
- Initiate an exposure request.

The controller agent communicates with the Guide Control Process on the guide host computer by sending commands over a socket connection.

3.2.4.2 fitspipe FITS Server

This is a process created by the Institute for Astronomy (IfA) PanSTARRS development team. It receives FITS images from a client and enables other clients to subscribe and receive FITS images. The underlying socket communication is performed via the libsockio socket library. More information regarding the usage of the fitspipe FITS server can be found in section 6.3.

3.2.4.3 fitspipe-get Client

This is yet another process created by the Institute for Astronomy (IfA) PanSTARRS development team. It connects to the fitspipe FITS server and retrieves FITS files. These images are then sent to STDOUT where they can be piped to clients that receive the images from STDIN. More information regarding the usage of the fitspipe-get client can be found in section 6.5.

3.2.4.4 FITS Cube Saver

This process will create a client connection to the fitspipe FITS Server via the fitspipe-get client and subscribe to FITS images. For diagnostic purposes FITS files will be generated for the images received from the public FITS server. Cubes and individual images will be saved for guiding, acquisition, and focus sequences. More information regarding the FITS Cube Saver can be found in section 6.6

3.2.5 Viewing Analysis

Hopefully the center of the entrance aperture will be well calibrated and the guider can accurately use this target position to calculate corrections. However, it is possible that the calibration of the entrance aperture center could shift. This component of the software is responsible for analyzing the flux surrounding the entrance aperture in an effort to measure any necessary shifts to the centering position reference used by the guider. This component of the software will be designed and implemented by IRAP. Design details for the viewing analysis can be found in SPIROU-4100-IRAP-RP-00638.

3.2.5.1 Image Viewing Graphical User Interface

A graphical user interface (GUI) will be created to show both target acquisition and guiding images as close to real-time as possible. The GUI will receive images from the fitspipe FITS image server on the session host via the fitspipe-get client. As FITS files are received via STDIN they will be displayed on the GUI. Any important metadata associated with the image will be transferred within the headers of each FITS frame. These values may be used by the GUI to update text fields or annotate images.

The GUI will also be used to handle special cases where the automated astrometry and target acquisition is not able to place the target on the entrance aperture. In this case, the observer will have the ability to click on the image in order to force the selected target at the pixel location on the GUI to be placed on the entrance aperture for manual acquisition. These positioning commands will be sent from the GUI to the Controller Agent. The Controller Agent will then forward the TCS centering command on to the Guide Control Process which will subsequently trigger the telescope movement via the TCS Channel Access interface.

It will be possible to launch more than one GUI. Launching additional "display only" GUIs can be helpful for diagnostic purposes. During normal observing, a single GUI will run on the display host used by the observer. Since the human eye is not capable of effectively processing guide images at 50Hz, the image display will likely be throttled to something more reasonable (perhaps 5-10 Hz).

A screen mockup of the proposed image viewing GUI for SPIRou is shown in figure 18. This display interface is similar to the GUI used with ESPaDOnS. More details regarding the design and usage of the GUI can be found in section 7.

3.2.5.2 2MASS Catalog Server

This server was created for use with WIRCam and it enables **scat** catalog requests to be processed with the results being sent back as a multi-line response to the client.

3.2.5.3 CFHT Logging Server

This server process receives client logging messages and stores these messages in the CFHT log file on the session host computer.

4 Controller Agent to Guide Control Process Communication

This section covers the communication between the Controller Agent on the session host and the Guide Control Process on the guide host as shown in figure 4.

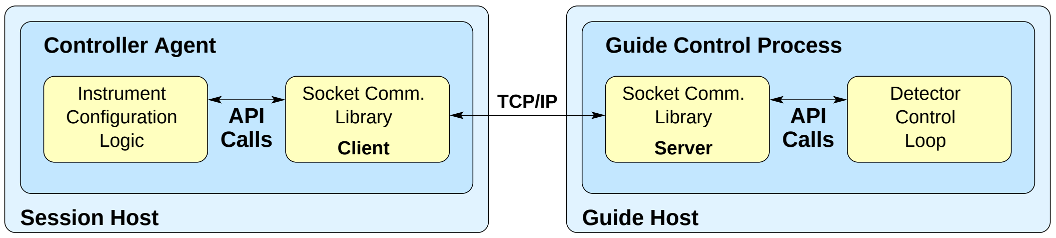


Figure 4: SPIRou Guide Control Process to Controller Agent Communication

4.1 Overview

The Controller Agent receives command requests from either the director interface or from the Image Viewing GUI. These command requests are processed by the Controller Agent and may require communication with the Guide Control Process on the guide host computer. The communication between the Controller Agent and the Guide Control Process will utilize a text-based TCP/IP protocol. The Guide Control Process will service each request received across the socket interface and send back a response. With the exception of a disconnect response, each client request will receive at least one response from the Guide Control Process. In some cases, the Controller Agent will receive an initial response indicating that the request has been received for processing and the interface is busy performing the request. In this case, the Controller Agent will receive a follow-up message when the requested operation is complete. The Controller Agent must not send any new commands until it has fully processed the current command. If, for some reason, the Guide Control Process receives a new command request from the Controller Agent before it has sent the client a response, it will inform the Controller Agent that a protocol error has occurred. At this point, the Guide Control Process will expect the Controller Agent to close its socket connection. If, however, the client sends another command, the Guide Control Process will close the client connection.

The Guide Control Process can only accept control commands from a single connection. This restriction is in place to present the receipt of instrument commands from multiple sources.

The Guide Control Process will utilize the sockio library to handle the low-level TCP/IP socket details. The sockio library uses a single-threaded, non-blocking approach to handling client connections. The interaction between the sockio library functions and the Guide Control Process will be discussed in more detail in the detailed design section for the Guide Control Process (see section 5). In addition, you can review the CFHT Sockio I/O Library (sockio) design document for more details at http://software.cfht.hawaii.edu/sockio.

Both the Controller Agent and the Guide Control Process will be designed in such a way that any data sent across the socket can be gracefully handled. This includes receiving binary data or unusually long messages, which may or may not be properly terminated with a newline (CR/LF or LF). If a client attempts to connect from outside the CFHT network, or a client violates the established message protocol, whenever possible its connection will be terminated.

Each request received by the Guide Control Process will be checked to make sure it is a valid command and does not contain any invalid characters. The Guide Control Process will only process requests that contain 7 bit ASCII printable characters terminated with a newline (CR/LF or LF). If a non-conforming request is received, it will be rejected with a “syntax error” response. The first character of the Guide Control Process response indicates the type of response message. This first character can be one of the following.

- **’.’** - Request from the Controller Agent was processed successfully. It is now safe for the Controller Agent to send its next request.
- **’!’** - The request from the Controller Agent was not processed successfully. The reason for the failure is included following the ’!’.
- **’*’** - This is an out-of-band message from the Guide Control Process to the Controller Agent. For example, a request to the Guide Control Process may result in a successful response (’.’) to the request followed by a out-of-band message (’*’) whenever the requested operation has completed.
- **’+’** - This indicates a multi-line response from the Guide Control Process. Each non-final line of a multi-line response is prefixed with a ’+’. The last message of a multi-line response will either be prefixed with a ’.’ in case of a successful response or a ’!’ in case of an unsuccessful response. At this point, no multi-line message responses have been identified as necessary. However, if such support is required in the future, this is the way it will be handled.
- **’?’** - This indicates that a client protocol violation has occurred. A protocol violation is detected by the Guide Control Process whenever a new request is received before the Guide Control Process has sent either a positive (’.’) or negative (’!’) response to the previous request. The Guide Control Process expects the Controller Agent to close its connection whenever a protocol error message is sent to it. If the Controller Agent sends another request to the Guide Control Process without closing and re-establishing the connection, the Guide Control Process will close the client connection.

The client-server command protocol and rules described previously are based on the rules used within previously defined client-server projects implemented at CFHT. As a result, it will be possible to re-use existing code.

4.2 Message Flow Diagrams between Controller Agent and Guide Control Process

In order to understand the communications between the Controller Agent and the Guide Control Process, a set of message flow diagrams is included to illustrate the normal operational flows and error conditions. In each message flow diagram, solid lines indicate Controller Agent or Guide Control Process-initiated operations while dotted lines indicate low-level socket messages. Each of the flows show a generic description for the type of message that is being sent. A more detailed command syntax can be found in section 4.3.

4.2.1 Client Socket Connection

Figure 5 illustrates the underlying socket library connection handshaking and Guide Control Process checks when the Controller Agent attempts to establish a connection with the Guide Control Process.

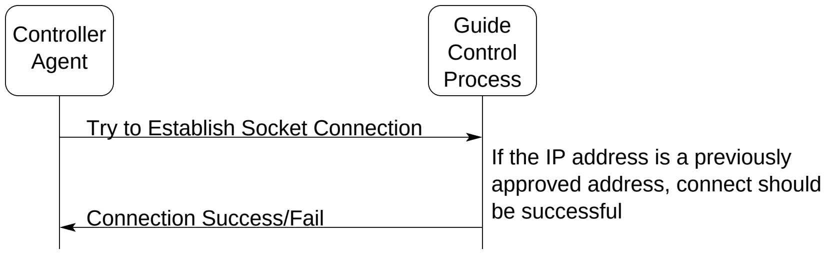

Controller Agent
Guide Control Process
Try to Establish Socket Connection
If the IP address is a previously approved address, connect should be successful
Connection Success/Fail

Figure 5: Client Socket Connection Message Sequence

4.2.2 Client Socket Disconnection

Figure 6 illustrates the normal sequence of messages when the Controller Agent disconnects from the Guide Control Process. It assumes that the Controller Agent sends a message to disconnect from the Guide Control Process. It is also possible for the Controller Agent to close its end of the socket connection in order to trigger a disconnect. It should be safe for the Controller Agent to exit without sending a message to the Guide Control Process before exiting. Once the Controller Agent process exits, the socket should automatically be closed.

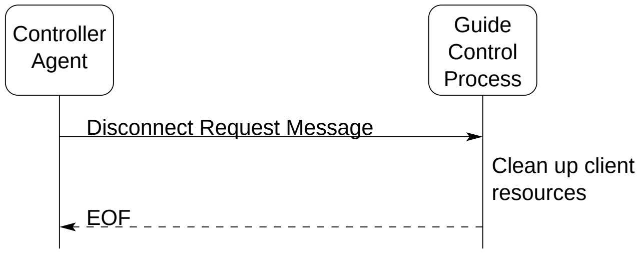


Figure 6: Client Socket Disconnection Message Sequence

4.2.3 Successful Command Request

Figure 7 illustrates the message sequence when the Controller Agent sends a command request to the Guide Control Process that is processed successfully. The return message from the Guide Control Process to the Controller Agent will be prefixed with a '.' character indicating that the request was successfully processed. The contents of the message will indicate whether the request was completed or whether the request was initiated.

4.2.4 Unsuccessful Command Request

Figure 8 illustrates the message sequence when the Controller Agent sends a command request to the Guide Control Process that is not processed successfully. The return message from the Guide Control Process to the Controller Agent is prefixed with a '!' character indicating that the request was not successfully executed.

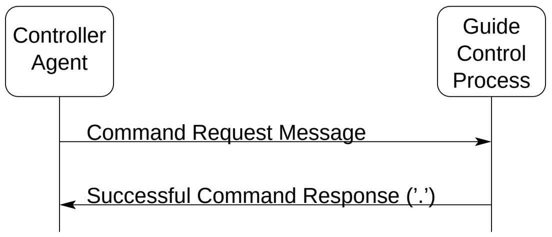


Figure 7: Successful Command Request Message Sequence

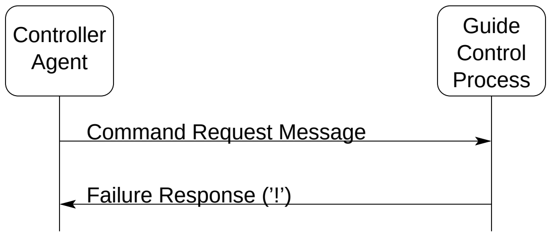


Figure 8: Unsuccessful Command Request Message Sequence

4.2.5 Command Execution

Figure 9 illustrates the message sequence when the Controller Agent sends a command request to the Guide Control Process that takes more than a few seconds to normally complete. In this case, the Guide Control Process acknowledges that the command received is being processed with an initial response. However, the notification that the command has completed is noted with a subsequent out-of-band command completion message. In addition, there may be several out-of-band progress messages sent from the Guide Control Process to the Controller Agent. These messages are used by the Controller Agent to display progress bar messages in director.

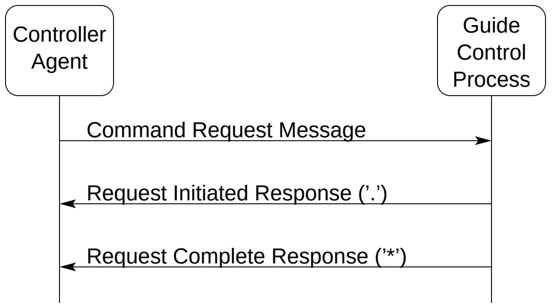

Controller Agent
Guide Control Process
Command Request Message
Request Initiated Response ('.')
Request Complete Response ('*')

Figure 9: Command Execution Message Sequence

4.2.6 Aborting a Command Already in Progress

From time to time it may be necessary to abort a previously executed command. Figure 10 illustrates the message sequence for an aborted sequence.

4.2.7 Out-of-Sequence Command Request (Protocol Error)

It is important that both the Controller Agent and Guide Control Process respect the messaging protocol that is outlined in this document. The protocol is designed in such a way that every client request will receive an associated server

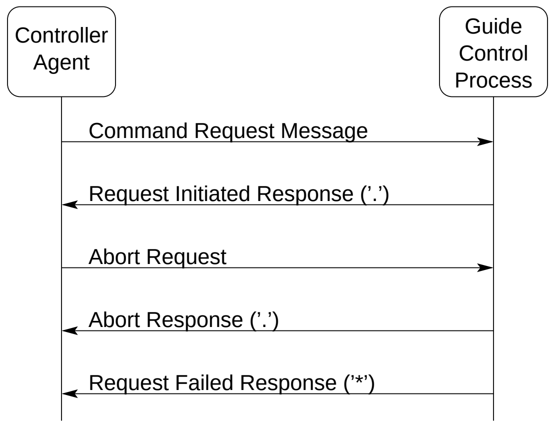


Figure 10: Aborted Message Sequence

response. The server response for a message request will be prefixed with either a '.' or '!' character. This character indicates whether the requested operation completed successfully or not. It is possible for a client to receive an informational “out-of-band” message at any time. An “out-of-band” message is always prefixed with a '*' character. It is important that the Controller Agent wait for a response message prefixed with either a '.' or '!' following a command request before sending the next command. Out-of-band messages don't count as a response to a client request.

If the Controller Agent sends another request to the Guide Control Process before waiting for the previous command response, this is considered a protocol error. When such a situation occurs, the synchronization between both client and server is broken. As a result, the Guide Control Process has no way of knowing how to respond. Following the existing standard used at CFHT for dealing with protocol error violations, the Controller Agent is informed that a protocol error has occurred and is expected to close its socket. Figure 11 illustrates two different protocol error scenarios. In the first scenario, the Controller Agent closes the socket as it should. In the second scenario, the Controller Agent sends another message request. Once the Guide Control Process receives additional messages from the client it closes the socket.

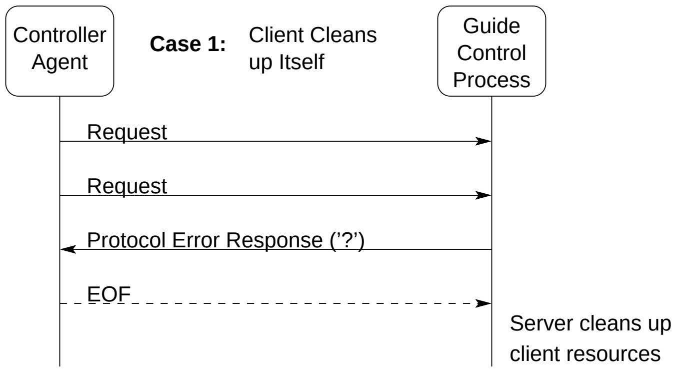
Protocol errors should never occur during normal operations. While it may seem extreme to close the socket, it is a clean way to handle the problem and ensure that both the client and server operate as designed.

4.3 Command Protocol

The previous section illustrated the high-level message sequences between the Controller Agent and the Guide Control Process. This section addresses the detailed syntax and format of the individual messages. As previously mentioned, each and every client request, with the exception of the “quit”, “exit”, or “logoff” requests, will receive a response. The Guide Control Process only handles single line 7-bit printable character messages terminated with a CR/LF or LF (“\r\n” or “\n”). If a request is received that doesn't conform to these rules, it is rejected.

As previously mentioned, the first character of a server response message indicates the type of response sent by the Guide Control Process. There are five different responses that the Guide Control Process can send to the Controller Agent.

- **Command Passed ('.')**
- **Command Failed ('!')**
- **Out-of-band Information ('*')**
- **Multi-line Command Response ('+')**



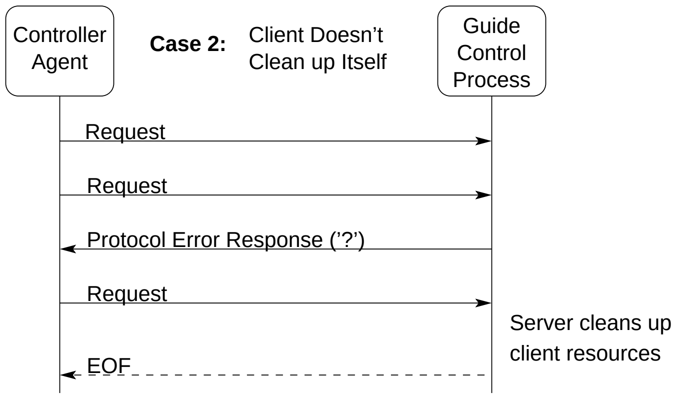

Controller Agent
Case 2:
Client Doesn't Clean up Itself
Guide Control Process
Request
Request
Protocol Error Response ('?')
Request
Server cleans up client resources
EOF

Figure 11: Protocol Error Message Sequence

- **Protocol Error ('?')**

The first character encoding scheme is defined in such a way that it is not specific to any particular command request sent across the socket interface from the Controller Agent.

Each command is defined with the required syntax and the expected response. Commands sent from the Controller Agent to the Guide Control Process are case insensitive. They can be sent in upper case, lower case, or mixed case. The commands in the following sections are shown in upper case. Mandatory command arguments are enclosed in '<' and '>' characters. Optional command arguments are enclosed in '[' and ']' characters. When the commands are sent the '<', '>', '[', and ']' characters must not be sent as part of the command text.

4.3.1 Disconnect from the Guide Control Process

The connection between the Controller Agent and the Guide Control Process will remain persistent until the client chooses to disconnect or the network connection between the client and server is broken. During initial development it may be more convenient to interact with the Guide Control Process using a telnet connection. When using a telnet connection, it is sometimes easier to type a short command than the CTRL - ']' quit sequence. The Guide Control Process will support several commands that enable the client to terminate the connection. These commands will cause the client to receive an EOF across the socket indicating that the Guide Control Process has closed the connection. The message sequence for this command can be found in figure 6.

Client Commands:

```
QUIT
```

```
EXIT
LOGOUT
```

Each of these commands will not generate a message response from the Guide Control Process.

4.3.2 Become the Master Control Client

Once the Controller Agent connects to the Guide Control Process it must send a “CONTROL” command in order to register itself as a connection capable of sending commands. If an existing connection already has control, the client requesting control will receive an error indicating that control could not be established. If the request is sent with an optional “FORCE” argument, the connection that was previously in control will be terminated and control is granted to the connection that sent the “CONTROL FORCE” request. The possible message sequences for this command can be found in figure 7 and figure 8.

Controller Agent Command:

```
CONTROL [ FORCE ]
```

Possible Guide Control Process Responses:

```
. CONTROL
! CONTROL "permission denied - connection from xxx.xxx.xxx.xxx has control"
```

In principle only one client should ever be connected to the Guide Control Process and that connection should be the Controller Agent. However, this command allows for the ability to identify if a connection already exists and, if necessary, terminate it. This may help if an existing connection is in an idle state and hasn’t been cleaned up by the underlying TCP keepalive. In addition, this capability enables a telnet connection to take over control for diagnostic purposes.

By allowing only a single client connection to establish control, this should prevent instances where multiple clients are sending commands to the Guide Control Process.

4.3.3 Trigger an Automatic Acquisition

Once the telescope and dome have settled into position for the requested target, it is necessary to ensure that the PI specified science target falls on the entrance aperture. Unfortunately, the accuracy of telescope pointing does not automatically allow this to happen without some help. In order to improve the science productivity of SPIRou, an attempt will be made to automatically identify the field and place the science target on the entrance aperture. This is accomplished via the following message sequence between the Controller Agent and the Guide Control Process. A visual representation for the possible message flows for this sequence can be found in figure 8 and figure 9.

Controller Agent Command:

```
ACQUIRE < RA=xx:xx:xx.x > < DEC=xx:xx:xx.x > [ 2MASS_ID=xxxxxxxx-xxxxxxx ]
```

Possible Guide Control Process Responses:

```
. ACQUIRE BUSY
! ACQUIRE "String with error details"
```

If the original response was an indication that the Guide Control Process was busy processing an acquisition request, the following messages may be received at a later time.

```
* ACQUIRE DONE  (Sent if an automatic acquisition completed successfully)
* ACQUIRE HELP  (Sent if manual positioning help by the observer is required)
* ACQUIRE FAIL "String with error details"  (Sent if acquisition failed)
```

4.3.4 Start Guiding

Once the science object is centered on the entrance aperture it is possible for guiding to be started. The following message sequence between the Controller Agent and Guide Control Process is used to trigger closed loop guiding. Once the Controller Agent receives an indicating that guiding has started successfully, it can trigger an exposure sequence. A visual representation for the possible message flows for this sequence can be found in figure 7 and figure 8.

Controller Agent Command:

```
GUIDE
```

Possible Guide Control Process Responses:

```
. GUIDE BUSY
! GUIDE "String with error details"

* GUIDE DONE  (Sent if guiding has been initiated and is stable)
* GUIDE FAIL "String with error details"  (Sent if guiding failed)
```

In order for the guiding to be complete the following items must have occurred.

- Science target is well detected within the entrance aperture
- Corrections are sent to the ISU
- TCS is receiving corrections every second in order to keep the ISU well centered within its dynamic range
- ISU is well centered within its dynamic range

4.3.5 Set the Guide Offset

In order to support guiding off of the entrance aperture, there is a command to define an offset to apply (in arcseconds). This support is added in order to use the flux from a guide star outside of the hole to accurately perform tip/tilt guiding. By offsetting the star it becomes possible to capture flux from the sky within the entrance aperture. The following message sequence illustrates how the guiding offset is defined. A visual representation for the possible message flows for this sequence can be found in figure 7 and figure 8.

Controller Agent Command:

```
GOFFSET < X_POS > < Y_POS >
```

Both the X position and Y position are provided in arcseconds. X is a position corresponding to a correction in RA and Y is a position corresponding to a correction in Dec. Adjustments in RA should be sent with a cosine declination adjustment already factored in. Both the X and Y positions are defined as absolute positions with respect to the center of the entrance aperture. If a “GOFFSET 0 0” command is sent, an offset is not used for guiding.

Possible Guide Control Process Responses:

```
. GOFFSET
! GOFFSET "String with error details"
```

It is important to note that this command does not cause the telescope to move. Instead, only the subraster and NULL position used for guiding are modified.

4.3.6 Center Star on the Entrance Aperture

It may not always be possible to automatically acquire a science target and place it on the entrance aperture. In some cases, it may be necessary for the remote observer at CFHT to manually select an object from the image viewing GUI and trigger the object to be centered on the entrance aperture. The following message sequence illustrates how the telescope can be moved via the Guide Control Process in order to center an object on the entrance aperture. A visual representation for the possible message flows for this sequence can be found in figure 8 and figure 9. This command would typically be sent after the Controller Agent receives an “* ACQUIRE HELP” response to a previous ACQUIRE command request.

Controller Agent Command:

```
GSTAR < X_POS > < Y_POS >
```

Both the X position and Y position are provided in arcseconds. X is a position corresponding to a correction in RA and Y is a position corresponding to a correction in Dec. Adjustments in RA should be sent with a cosine declination adjustment already factored in.

Possible Guide Control Process Responses:

```
. GSTAR
! GSTAR "String with error details"
```

It is important to note that this command will move the telescope, but not change the internal position of where the TCS believes the telescope to be. For example, if the telescope is commanded to move to a particular RA and Dec Position on the sky this command will not change this position. As a result, this command is useful to manually correct the pointing of the telescope.

4.3.7 Move Star to the Focus Position

As with ESPaDOnS, two long exposure on a star will be taken approximately .5mm on either side of what is thought to be the current best focus. These images are analyzed using a parabolic fit in order to calculate the ideal focus for the instrument. In order to best calculate the focus of the star, the star is moved to a previously defined “clean” region of the detector. This region should contain as few cosmetic defects as possible. The following message sequence illustrates how the telescope can be moved via the Guide Control Process in order to center an object on the previously defined region used for focus sequences. A visual representation for the possible message flows for this sequence can be found in figure 7 and figure 8.

Controller Agent Command:

```
FSTAR < X_POS > < Y_POS >
```

Much like the previously defined GSTAR command, both the X position and Y position are provided in arcseconds. X is a position corresponding to a correction in RA and Y is a position corresponding to a correction in Dec. Adjustments in RA should be sent with a cosine declination adjustment already factored in. The correction values are represented as the number of arcseconds in both X and Y to move an object from the entrance aperture to the focus position.

Possible Guide Control Process Responses:

```
. FSTAR
! FSTAR "String with error details"
```

It is important to note that this command will move the telescope and, unlike the GSTAR command, it will change the internal position of where the TCS believes the telescope to be.

4.3.8 Trigger a Focus Exposure Sequence

As part of a focus sequence, it is necessary to trigger a focus exposure .5 mm from what is thought to be the current best focus. Each focus exposure will need to be long enough to average out the effects of seeing. With ESPaDOnS an exposure time of 30 seconds is used. However, since it isn't possible to take an exposure longer than 500 milliseconds with the FLIR camera, a stack of 60 500 millisecond exposures can be used to generate a 30 second exposure. Since the focus exposures will be analyzed outside of the Guide Control Process, the following message sequence is implemented to provide the capability for triggering exposures. A visual representation for the possible message flows for this sequence can be found in figure 8 and figure 9.

Controller Agent Command:

```
GO < ETYPE=FOCUS > < ETIME=XX.X > < RASTER=XC,YC,XS,YS >
```

The parameters specified for the "go" command are as follows.

- **etype** - focus exposure type
- **etime** - exposure time in seconds (for the full sequence)
- **raster** - subraster specification (XC = X center, YC = Y center, XS = box width in X, YS = box height in Y)

Possible Guide Control Process Responses:

```
. GO BUSY
! GO "String with error details"
```

If the original response was an indication that the Guide Control Process was busy processing the exposure request, the following messages may be received at a later time.

```
* GO DONE  (Sent if the exposure sequence completed successfully)
* GO FAIL "String with error details"  (Sent if the exposure sequence failed)
```

While the exposures are taken by the Guide Control Process, the images are ultimately saved on the session host by the guide cube saver.

4.3.9 Trigger an Open-Ended Imaging Sequence

For diagnostic purposes and on-sky testing, it is necessary to provide a mode where the guiding camera is put into an imaging mode. At this point, the Guide Control Process will continue to take images from the FLIR camera and send these images to STDOUT. Since the maximum exposure time on the FLIR camera is 500 milliseconds, if an exposure time is requested that is larger than this duration, multiple images will be stacked before they are sent to STDOUT. A visual representation for the possible message flows for this sequence can be found in figure 8 and figure 9.

Controller Agent Command:

```
GO < ETYPE=IMAGING > < ETIME=XX.X > < RASTER=XC,YC,XS,YS >
```

The parameters specified for the "go" command are as follows.

- **etype** - imaging exposure type

- **etime** - exposure time in seconds (for the full sequence)
- **raster** - subraster specification (XC = X center, YC = Y center, XS = box width in X, YS = box height in Y)

Possible Guide Control Process Responses:

```
. GO
! GO "String with error details"
```

4.3.10 Set the ISU Operating Mode

Normally the ISU will always be in an active mode. However, as we've seen with MegaPrime, a temporary issue may arise where the ISU can't be controlled due to an electro-mechanical issue (such as a defective D/A interface). As a result, a mode is proposed with SPIRou to preset the ISU position and only use telescope guiding. Obviously this is not a mode that SPIRou would normally be operated in. However, it may provide the ability to operate the instrument in a temporarily degraded mode, as opposed to being down completely. The default mode will always be "active".

Controller Agent Command:

```
ISUMODE < ACTIVE | FIXED >
```

Possible Guide Control Process Responses:

```
. ISUMODE
! ISUMODE "String with error details"
```

The ISU mode can't be changed during a guiding sequence. If the mode is changed, it must be set before a guiding sequence is initiated.

4.3.11 Abort a Previously Initiated Sequence

In some cases it may be necessary to abort a previously initiated sequence. This abort can be sent to those commands that respond with an initial BUSY state followed by a subsequent out-of-band DONE state. This applies to the ACQUIRE, GUIDE, GO, and DONE commands. An abort is one way that guiding can be terminated when guiding is no longer necessary. The following sequence illustrates the typical messages as part of an abort. A visual representation for the possible message flows for this sequence can be found in figure 10.

Controller Agent Command:

```
ABORT
```

Possible Guide Control Process Responses:

```
. ABORT
! ABORT "String with error details"
```

5 Guide Control Process Design

5.1 Design Overview

The Guide Control Process will be implemented using the C programming language. The process itself will run on the Linux guide host computer installed in the Cassegrain environment on the telescope. While the Guide Control Process is responsible for performing a number of different guide camera related activities, much of the underlying complexity is handled by the various libraries shown in figure 2. As a result, the control logic of the Guide Control Process is primarily defined within a series of flow charts and message flow diagrams.

5.2 Guide Control Process Flowcharts and Message Flow Diagrams

5.2.1 Main Loop

Figure 12 contains a flowchart that shows the initialization and main loop for the Guide Control Process. If fast guiding is in progress, the Guide Control Process will receive FITS images from STDIN that must be handled. Otherwise, the entire process is driven by command requests received from the Controller Agent.

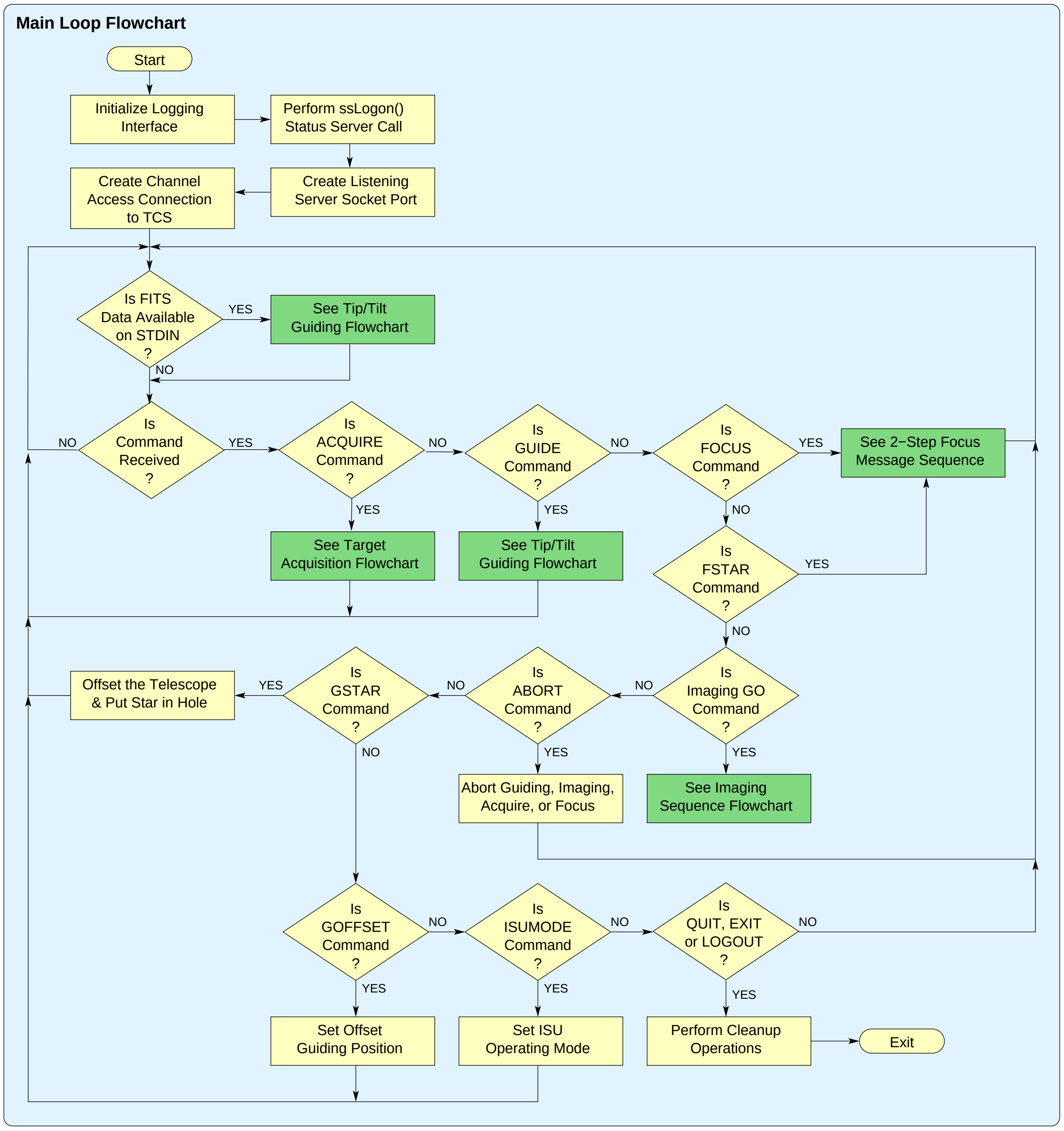

Main Loop Flowchart
Start
Initialize Logging Interface
Perform ssLogon() Status Server Call
Create Listening Server Socket Port
Create Channel Access Connection to TCS
Is FITS Data Available on STDIN ?
YES
See Tip/Tilt Guiding Flowchart
NO
Is Command Received ?
NO
YES
Is ACQUIRE Command ?
NO
Is GUIDE Command ?
NO
Is FOCUS Command ?
YES
See 2−Step Focus Message Sequence
YES
See Target Acquisition Flowchart
YES
See Tip/Tilt Guiding Flowchart
NO
Is FSTAR Command ?
YES
NO
Is Imaging GO Command ?
NO
Is ABORT Command ?
NO
Is GSTAR Command ?
YES
Offset the Telescope & Put Star in Hole
NO
YES
Abort Guiding, Imaging, Acquire, or Focus
YES
See Imaging Sequence Flowchart
Is GOFFSET Command ?
NO
Is ISUMODE Command ?
NO
Is QUIT, EXIT or LOGOUT ?
NO
YES
Set Offset Guiding Position
YES
Set ISU Operating Mode
YES
Perform Cleanup Operations
Exit

Figure 12: Guide Control Process Main Loop Flowchart

5.2.2 Target Acquisition and Centering

Section 3.2.2 described the goal for SPIRou to automatically identify targets and offset the telescope in order to place the science target on the entrance aperture. In order to reduce target acquisition overheads an automated acquisition will be attempted whenever possible. Figure 13 contains a high-level flowchart showing the basic operations that will be performed during the field acquisition and target centering process.

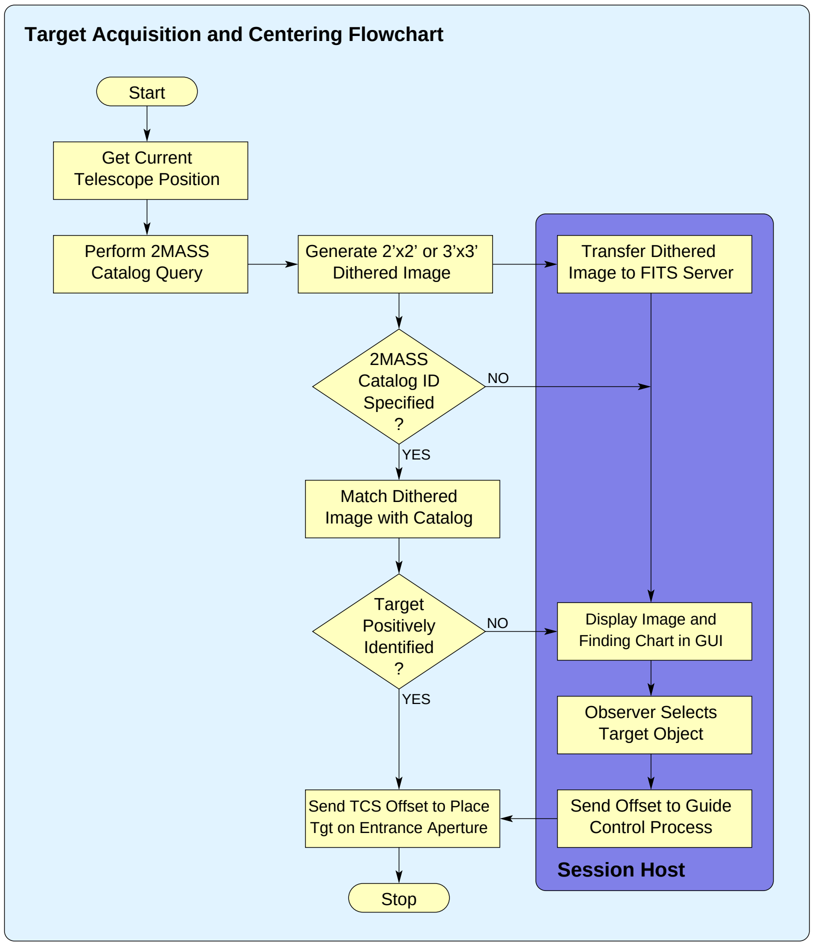

Target Acquisition and Centering Flowchart
Start
Get Current Telescope Position
Perform 2MASS Catalog Query
Generate 2'x2' or 3'x3' Dithered Image
Transfer Dithered Image to FITS Server
2MASS Catalog ID Specified ?
NO
YES
Match Dithered Image with Catalog
Target Positively Identified ?
NO
YES
Display Image and Finding Chart in GUI
Observer Selects Target Object
Send TCS Offset to Place Tgt on Entrance Aperture
Send Offset to Guide Control Process
Session Host
Stop

Figure 13: SPIRou Target Acquisition and Centering Flowchart

5.2.3 Guiding Initiation

In order to achieve the radial velocity stability required by the science requirements, the science target must be well centered on the entrance aperture and the motion of the science target on the fiber must be minimized. Figure 14 contains a flowchart showing the sequence of operations performed when a request is made to initiate guiding. Once guiding is initiated within the Fast Guiding Process, the Guide Control Process doesn't do much beyond sending telescope corrections every second. Design details for the Fast Guiding Process can be found in SPIROU-4200-ASIAA-RP-00646.

5.2.4 2-Step Focus

For each of the primary instruments at CFHT two images are taken at +/- 0.5mm from what is thought to be the best focus for the instrument. These images are analyzed and a parabolic fit is performed in order to identify the optimal

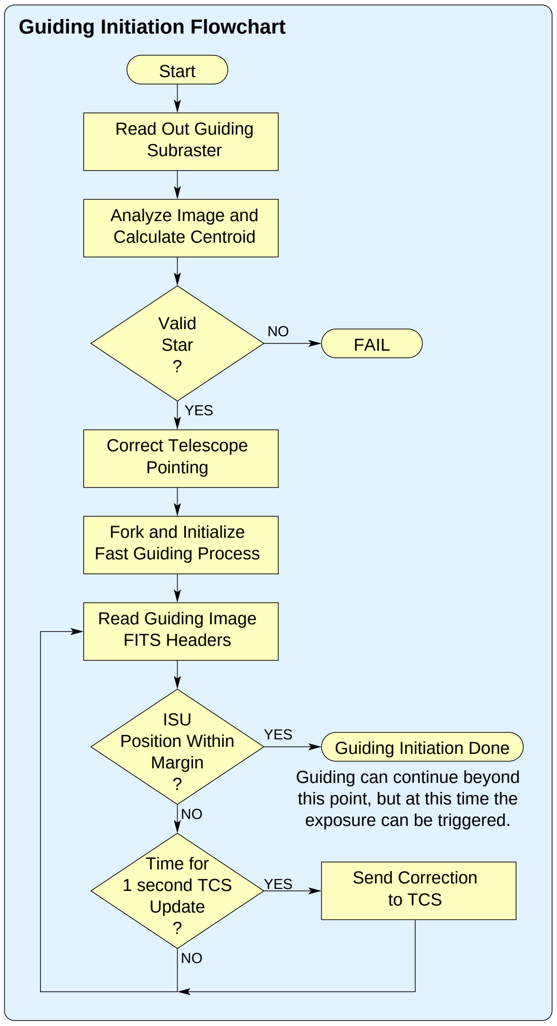

Guiding Initiation Flowchart
Start
Read Out Guiding Subraster
Analyze Image and Calculate Centroid
Valid Star ?
NO
FAIL
YES
Correct Telescope Pointing
Fork and Initialize Fast Guiding Process
Read Guiding Image FITS Headers
ISU Position Within Margin ?
YES
Guiding Initiation Done
Guiding can continue beyond this point, but at this time the exposure can be triggered.
NO
Time for 1 second TCS Update ?
YES
Send Correction to TCS
NO

Figure 14: SPIRou Tip/Tilt Guiding Flowchart

focus position for the instrument. In order to perform a 2-step focus sequence the following steps must be performed.

- Choose the star to perform the focus on
- Offset the telescope so the chosen star falls on a pre-defined area of the detector that is free from any major defects
- Move the focus stage to be -0.5mm from the current focus position
- Trigger a 30 second exposure sequence
- move the focus stage to be +0.5mm from the original position
- Trigger a 30 second exposure sequence
- Restore the focus stage it its original location
- Offset the telescope back to its original location
- Generate the final focus cube and perform a parabolic fit to determine the best focus. NOTE: At this point the focus stage could be moved to the ideal calculated location instead of restoring it to its original position.

In order to understand better where each of the previously defined steps occurs, please reference the message sequence in figure 15.

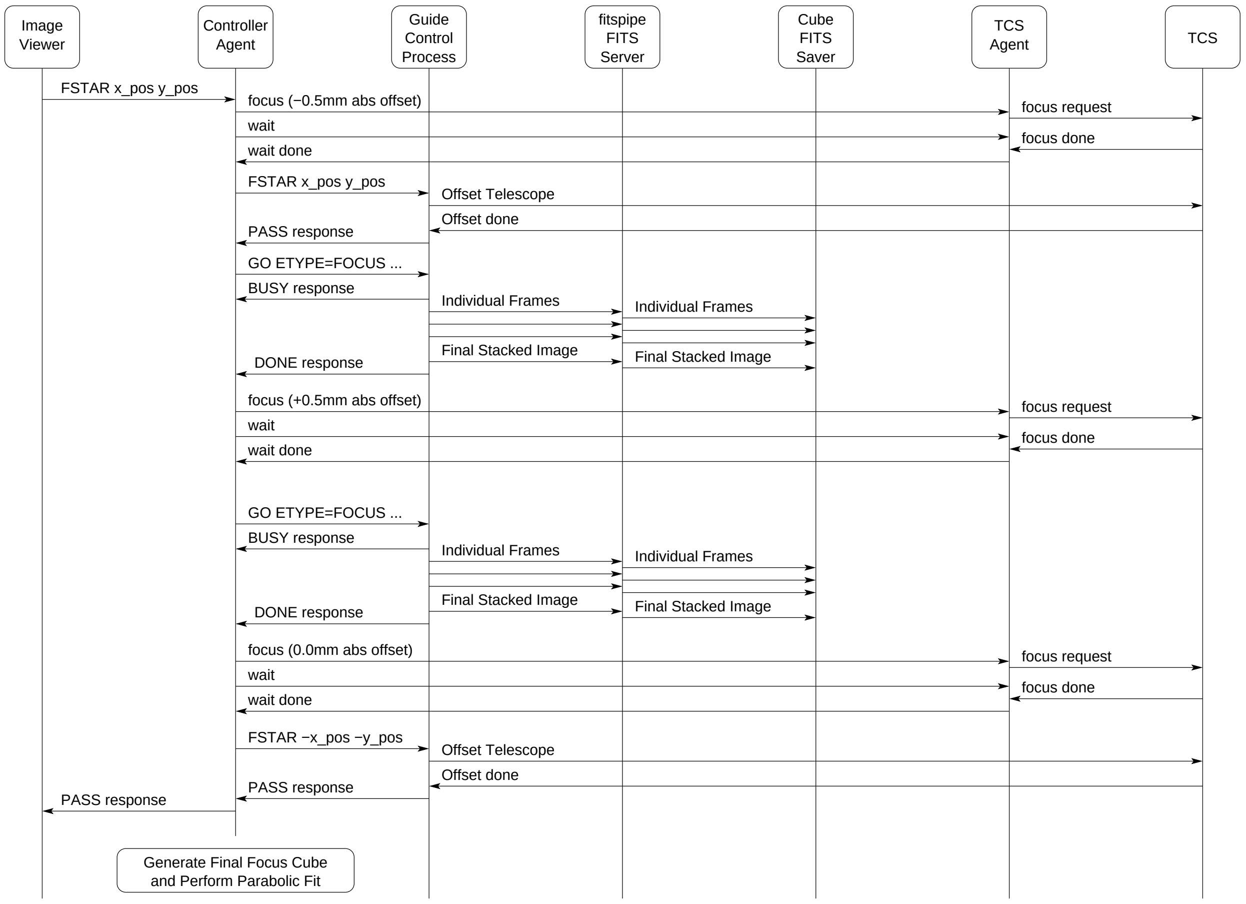


Figure 15: 2-Step Focus Message Sequence

5.2.5 Imaging Sequence

As previously noted, for diagnostic purposes and on-sky testing, it is necessary to provide a mode where the guiding camera is put into an imaging mode. Figure 16 illustrates the flowchart for this sequence. The Guide Control Process will utilize the Detector Control Library to set the location of the subraster and the exposure time. If the exposure time is greater than the maximum supported by the camera (500 ms), the maximum exposure time will be used and images will be stacked to achieve the target exposure time requested by the observer. Once an image meeting the requested exposure time of the observer is ready, it will be packaged into a 16-bit FITS file and sent to STDOUT. As long as imaging mode is still active, additional images will be taken indefinitely.

5.2.6 Command Line Syntax

The Guide Control Process is started using the following command line syntax.

```
> { spirou_gcp 2>&3 | fitspipe-put server=spirou-session port=9999 feed=default; } \
  3>&1 1>&2 | cfhtlogclient
```

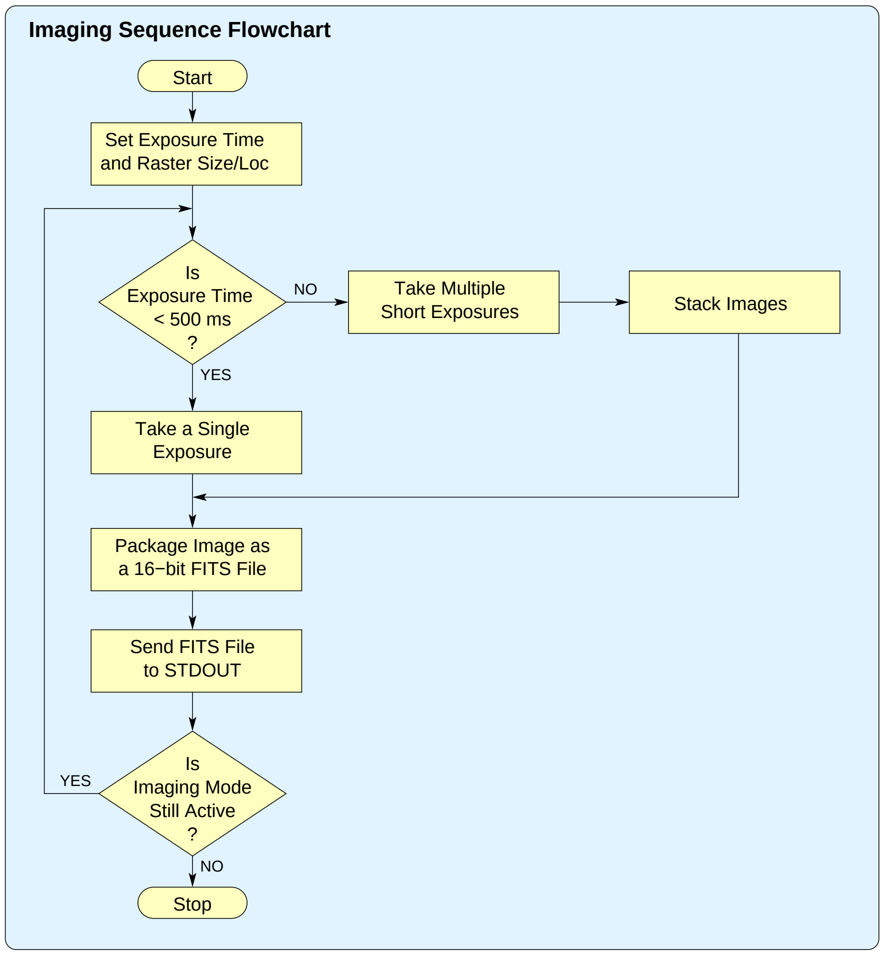

Imaging Sequence Flowchart
Start
Set Exposure Time and Raster Size/Loc
Is Exposure Time < 500 ms ?
NO
Take Multiple Short Exposures
Stack Images
YES
Take a Single Exposure
Package Image as a 16−bit FITS File
Send FITS File to STDOUT
Is Imaging Mode Still Active ?
YES
NO
Stop

Figure 16: Imaging Sequence

This syntax ensures that STDOUT is redirected to a fitspipe-client and that STDERR is redirected to the CFHT logging client. Normally, the Guide Control Process will be started automatically in the /etc/rc.local script on the guide host computer.

6 FITS Image Handling

Throughout this document there have been numerous references to FITS images and associated metadata. FITS data is an important aspect of the tip/tilt and viewing software infrastructure. This section details the flow of FITS data and identifies the image format as well as the metadata that should be contained within the FITS data at each stage of the software architecture.

6.1 Design Overview

Figure 17 illustrates the the data flow between the guide camera and the various components that use the Infrared data. At this point, there are a number of different ways the data from the camera is used.

- **Guiding** - Image data is retrieved from the SWIR camera and used to calculate ISU corrections by the Fast

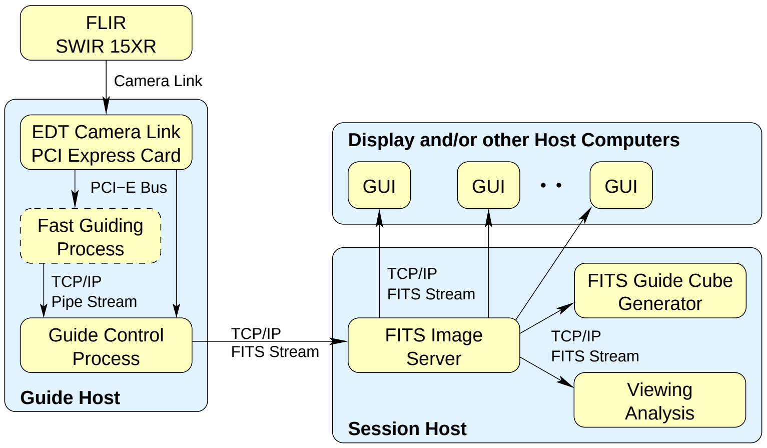


Figure 17: FITS Image Data Flow Diagram

Guiding Process. This data is then forwarded on to the Guide Control Process where the metadata within the FITS headers is used to send periodic telescope corrections to the TCS at a 1 Hz rate.

- **Acquisition** - Individual images at various telescope offset positions are retrieved from the SWIR camera by the Guide Control Process. These images are stitched together and used to match up with the 2MASS catalog in order to perform target acquisition.
- **Focus** - Two sequences of images spanning roughly 30 seconds per sequence at different telescope focus positions are used to calculate the best focus for the instrument.
- **Viewing Analysis** - By stacking a long sequence of guiding images it is possible to identify the overall guiding accuracy with respect to centering the science target on the entrance aperture. This analysis can be used to determine necessary changes to the NULL position used by the Fast Guiding Process for target centering. In addition, these images can be used to calculate an estimate for the current seeing conditions.
- **Real-Time Image Visualization** - Images can be displayed in a GUI as they are received by the FITS Image Server. A GUI will be used by the observer during the normal observing process. Additional GUIs can also be launched for diagnostic purposes. In addition, the GUI can be used by the observer to manually select targets and initiate focus sequences.
- **Diagnostics** - Each image will be saved to disk by the guide cube generator. These images are available for diagnostic purposes.

6.2 FITS Image Headers and Image Specification

All image data shown in the data flow diagram in figure 17 must be 16-bit 2D image data. In addition, each image must have metadata stored within the FITS headers. The image data and associated metadata becomes a critical information interface between subcomponents. As a result, this section will specify exactly what data must be available at each location within figure 17.

6.2.1 Guiding FITS Images and Metadata from Fast Guiding Process

Images retrieved from the FLIR Camera by the Fast Guiding Process must be packaged into FITS files using the CFHT FITS handling library (libfh). The image data must be 16-bit 2D raw image data with the following FITS headers. The images should packaged as they received from the FLIR camera without any enhanced image processing.

- **SIMPLE** - Set to ”T” indicating a standard FITS file
- **BITPIX** - Set to ”16” indicating 16-bits per pixel
- **NAXIS** - Set to 2 indicating a 2D image
- **NAXIS1** - Number of pixel columns (int)
- **NAXIS2** - Number of pixel rows (int)
- **BZERO** - Set to 32760.0 for the zero factor
- **BSCALE** - Set to 1.0 for the scale factor
- **PIXSCALE** - Pixel scale in arcseconds per pixel
- **UNIXTIME** - The number of seconds since 1970-01-01 00:00:00 UTC with 3 decimal points of precision (ex: 1388791475.001). This is the easiest format to use when processing dates and performing calculations downstream. (float)
- **SEQNUM** - Incrementing sequence number starting at 0 when the Fast Guiding Process is started (int)
- **WIN_X0** - X0 window coordinate for the guide raster (int)
- **WIN_Y0** - Y0 window coordinate for the guide raster (int)
- **WIN_X1** - X1 window coordinate for the guide raster (int)
- **WIN_Y1** - Y1 window coordinate for the guide raster (int)
- **NULL_X** - Null position (center of the aperture hole in X) using a detector reference (float)
- **NULL_Y** - Null position (center of the aperture hole in Y) using a detector reference (float)
- **CENTER_X** - Calculated position of the centroid in X using a detector reference (float)
- **CENTER_Y** - Calculated position of the centroid in Y using a detector reference (float)
- **SVOLT_X** - Analog voltage sent to the ISU in X (float)
- **SVOLT_Y** - Analog voltage sent to the ISU in Y (float)
- **RVOLT_X** - Analog voltage read from the ISU in X giving the actual ISU position (float)
- **RVOLT_Y** - Analog voltage read from the ISU in Y giving the actual ISU position (float)
- **ETYPE** - Set to ”GUIDE” indicating a guiding exposure.
- **ETIME** - Exposure time in seconds (float)
- **GDSTATE** - Indicates the guiding system state (ACQUIRE, GUIDING, OFF, or ERROR)

6.2.2 Guiding FITS Images and Metadata from Guide Control Process

As indicated in figure 17 guiding FITS images flow from the Fast Guiding Process to the Guide Control Process. The image data will remain unchanged, however additional headers will be added to the FITS images. Below is a breakdown of the FITS headers that will be associated with each of the guide images produced by the Guide Control Process.

- **SIMPLE** - Set to ”T” indicating a standard FITS file
- **BITPIX** - Set to ”16” indicating 16-bits per pixel
- **NAXIS** - Set to 2 indicating a 2D image
- **NAXIS1** - Number of pixel columns (int)
- **NAXIS2** - Number of pixel rows (int)
- **BZERO** - Set to 32760.0 for the zero factor
- **BSCALE** - Set to 1.0 for the scale factor
- **PIXSCALE** - Pixel scale in arcseconds per pixel
- **UNIXTIME** - The number of seconds since 1970-01-01 00:00:00 UTC with 3 decimal points of precision (ex: 1388791475.001). This is the easiest format to use when processing dates and performing calculations downstream. (float). This time is not modified by the Guide Control Process. It still contains the original timestamp populated by the Fast Guiding Process.
- **SEQNUM** - Incrementing sequence number starting at 0 when the Fast Guiding Process is started (int)
- **WIN_X0** - X0 window coordinate for the guide raster (int)
- **WIN_Y0** - Y0 window coordinate for the guide raster (int)
- **WIN_X1** - X1 window coordinate for the guide raster (int)
- **WIN_Y1** - Y1 window coordinate for the guide raster (int)
- **NULL_X** - Null position (center of the aperture hole in X) using a detector reference (float)
- **NULL_Y** - Null position (center of the aperture hole in Y) using a detector reference (float)
- **CENTER_X** - Calculated position of the centroid in X using a detector reference (float)
- **CENTER_Y** - Calculated position of the centroid in Y using a detector reference (float)
- **SVOLT_X** - Analog voltage sent to the ISU in X (float)
- **SVOLT_Y** - Analog voltage sent to the ISU in Y (float)
- **RVOLT_X** - Analog voltage read from the ISU in X giving the actual ISU position (float)
- **RVOLT_Y** - Analog voltage read from the ISU in Y giving the actual ISU position (float)
- **TCS_X** - Last correction value sent to the TCS (float)
- **TCS_Y** - Last correction value sent to the TCS (float)
- **ETYPE** - Set to ”GUIDE” indicating a guiding exposure.
- **ETIME** - Exposure time in seconds (float)

- **GDSTATE** - Indicates the guiding system state (ACQUIRE, GUIDING, OFF, or ERROR). In this case, the state would either be ACQUIRE, GUIDING, or ERROR.
- **RA** - Telescope right ascension
- **DEC** - Telescope declination
- **EQUINOX** - Equinox of coordinates

6.2.3 Acquire FITS Images and Metadata

When an acquisition sequence is requested, the Guide Control Process will trigger several images at different telescope offset positions. These images are stitched together to form a composite image that can be used for target identification in order to correct for telescope pointing and place the telescope on the entrance aperture. The Guide Control Process will output both the individual acquire exposures as well as the stitched composite image. The stitched composite image will be a processed image while the individual acquire exposures will be in a raw format. The FITS headers associated with the images are as follows.

- **SIMPLE** - Set to ”T” indicating a standard FITS file
- **BITPIX** - Set to ”16” indicating 16-bits per pixel
- **NAXIS** - Set to 2 indicating a 2D image
- **NAXIS1** - Number of pixel columns (int)
- **NAXIS2** - Number of pixel rows (int)
- **BZERO** - Set to 32760.0 for the zero factor
- **BSCALE** - Set to 1.0 for the scale factor
- **PIXSCALE** - Pixel scale in arcseconds per pixel
- **UNIXTIME** - The number of seconds since 1970-01-01 00:00:00 UTC with 3 decimal points of precision (ex: 1388791475.001). This is the easiest format to use when processing dates and performing calculations downstream. (float). This time is not modified by the Guide Control Process in the case of individual acquire exposures. This time is modified and set to the current time for a stitched acquisition image.
- **SEQNUM** - Incrementing sequence number starting at 0 when an acquisition sequence is initiated (int)
- **WIN_X0** - X0 window coordinate for the image raster (int). This field is removed for the stitched acquisition image.
- **WIN_Y0** - Y0 window coordinate for the image raster (int). This field is removed for the stitched acquisition image.
- **WIN_X1** - X1 window coordinate for the image raster (int). This field is removed for the stitched acquisition image.
- **WIN_Y1** - Y1 window coordinate for the image raster (int). This field is removed for the stitched acquisition image.
- **NULL_X** - Null position (center of the aperture hole in X) using a detector reference (float). This field is removed for the stitched acquisition image.
- **NULL_Y** - Null position (center of the aperture hole in Y) using a detector reference (float). This field is removed for the stitched acquisition image.

- **ETYPE** - Set to ”ACQUIRE” indicating an individual guiding exposure or ”ACQUIRE_STITCH” for a stitched acquire image
- **ETIME** - Exposure time in seconds (float)
- **GDSTATE** - Indicates the guiding system state (ACQUIRE, GUIDING, OFF, or ERROR). This would be set to ”OFF” in this case.
- **RA** - Telescope right ascension
- **DEC** - Telescope declination
- **EQUINOX** - Equinox of coordinates

6.2.4 Focus FITS Images and Metadata

A sequence of images spanning roughly 30 seconds per sequence at different telescope positions is used to calculate the best focus for the instrument. Since it is likely that a 30 second exposure can't be triggered without saturating the detector, a sequence of images is taken and stacked. The Guide Control Process will output both the individual focus exposures used to generate the stack as well as the stacked image. The individual exposures will be in a raw format while the stacked image will be processed. The FITS headers associated with the images are as follows.

- **SIMPLE** - Set to ”T” indicating a standard FITS file
- **BITPIX** - Set to ”16” indicating 16-bits per pixel
- **NAXIS** - Set to 2 indicating a 2D image
- **NAXIS1** - Number of pixel columns (int)
- **NAXIS2** - Number of pixel rows (int)
- **BZERO** - Set to 32760.0 for the zero factor
- **BSCALE** - Set to 1.0 for the scale factor
- **PIXSCALE** - Pixel scale in arcseconds per pixel
- **UNIXTIME** - The number of seconds since 1970-01-01 00:00:00 UTC with 3 decimal points of precision (ex: 1388791475.001). This is the easiest format to use when processing dates and performing calculations downstream. (float). This time is not modified by the Guide Control Process in the case of individual focus exposures. This time is modified and set to the current time for the stacked focus image.
- **SEQNUM** - Incrementing sequence number starting at 0 when a focus sequence is initiated (int)
- **WIN_X0** - X0 window coordinate for the image raster (int)
- **WIN_Y0** - Y0 window coordinate for the image raster (int)
- **WIN_X1** - X1 window coordinate for the image raster (int)
- **WIN_Y1** - Y1 window coordinate for the image raster (int)
- **NULL_X** - Null position (center of the aperture hole in X) using a detector reference (float)
- **NULL_Y** - Null position (center of the aperture hole in Y) using a detector reference (float)
- **ETYPE** - Set to ”FOCUS” indicating an individual focus exposure or ”FOCUS_STACK” for a stacked focus exposure

- **ETIME** - Exposure time in seconds (float)
- **GDSTATE** - Indicates the guiding system state (ACQUIRE, GUIDING, OFF, or ERROR). This would be set to ”OFF” in this case.
- **RA** - Telescope right ascension
- **DEC** - Telescope declination
- **EQUINOX** - Equinox of coordinates
- **FOCUSPOS** - Focus position

6.2.5 Miscellaneous Imaging FITS Images and Metadata

Much like with ESPaDOnS, there will be times where the guiding camera is used to provide images that can be viewed on the GUI. During this time, the Guide Control Process is reading images from the guide camera and sending these images to STDOUT. These images are not associated with guiding, acquisition, or focus sequences. Instead, these images are used to provide imaging feedback to the observer. These images will have the following headers populated.

- **SIMPLE** - Set to ”T” indicating a standard FITS file
- **BITPIX** - Set to ”16” indicating 16-bits per pixel
- **NAXIS** - Set to 2 indicating a 2D image
- **NAXIS1** - Number of pixel columns (int)
- **NAXIS2** - Number of pixel rows (int)
- **BZERO** - Set to 32760.0 for the zero factor
- **BSCALE** - Set to 1.0 for the scale factor
- **PIXSCALE** - Pixel scale in arcseconds per pixel
- **UNIXTIME** - The number of seconds since 1970-01-01 00:00:00 UTC with 3 decimal points of precision (ex: 1388791475.001). This is the easiest format to use when processing dates and performing calculations downstream. (float).
- **SEQNUM** - Incrementing sequence number starting at 0 when the camera is turned on and put into imaging mode (int)
- **WIN_X0** - X0 window coordinate for the image raster (int)
- **WIN_Y0** - Y0 window coordinate for the image raster (int)
- **WIN_X1** - X1 window coordinate for the image raster (int)
- **WIN_Y1** - Y1 window coordinate for the image raster (int)
- **NULL_X** - Null position (center of the aperture hole in X) using a detector reference (float)
- **NULL_Y** - Null position (center of the aperture hole in Y) using a detector reference (float)
- **ETYPE** - Set to ”IMAGING” indicating that these images are miscellaneous images used for viewing
- **ETIME** - Exposure time in seconds (float)
- **GDSTATE** - Indicates the guiding system state (ACQUIRE, GUIDING, OFF, or ERROR). This would be set to ”OFF” in this case.

6.3 FITS Server (fitspipe)

This is a server process created by the Institute for Astronomy (IfA) PanSTARRS development team. It receives FITS images from a client and enables other clients to subscribe to and receive FITS images. The underlying socket communication is performed via the libsockio socket library. Source code is available for use by CFHT. High-level documentation for the FITS communication utilities can be found at http://www.stargrasp.org/wiki/GraspExtrasSoftwareFitspipe.

Figure 2 illustrates the overall SPIRou guiding architecture. The FITS Server (fitspipe) will run on the session host Linux host computer. It will receive FITS images from the Guide Control process and supply FITS images to the FITS Cube Saver, Viewing Analysis, and Image Viewer.

The FITS Server has already been developed for use by the IfA and, based on an evaluation of the existing documentation and source code, it can be used as is. As a result, this section will focus primarily on the usage of the FITS Server rather than the low-level design details. The FITS Server is essentially a merge between the libsockio socket library and the libfh FITS handling library. Clients connecting to the FITS Server can either supply 16-bit 2D FITS image data or retrieve 16-bit 2D FITS image data.

When the fitspipe server is started using the following command line option, the following help summary is provided.

```
> fitspipe -help

Network FITS Video Stream Server v2.02
usage: fitspipe <port> [daemon=false]

Listens on port for fitspipe network clients. Clients can either
"put" images into particular feed, or "get" images from one.
Protocol is handled by fitspipe-put and fitspipe-get respectively.
By default, fitspipe becomes a background process.  This can be
prevented by adding "daemon=false".  There are no other options
for the server.  Make sure your firewall protects <port> adequately.
```

Typing “help” on the socket of a running fitspipe FITS Server produces the following information.

```
help
+ fitspipe v2.02 Jan 23 2014 (compiled Jan 23 2014) commands:
+   <Dump in a FITS file>                  - Same as "put feed=default"
+   version                                - Show fitspipe server version
+   put feed=<feedname>                    - To submit a FITS file
+   get feed=<feedname> [frame=<frameno>] - To retrieve a FITS frame
+   list or ls                             - List available feeds
+   composite compenable=<true|false> [compgeom=wxh compfeeds=<feed1,feed2,
feed3...>] - Set up composite of multiple feeds
+   shutdown                               - Make this server exit
+   exit or quit or q                      - Close the connection
. OK
```

The “fitspipe” FITS Server will be started at boot time from the /etc/rc.local script on the session host computer. This can be accomplished by including the following line in the script.

```
fitspipe 9999
```

This line would cause the FITS Server to become a daemon/background process after the listening socket is open.

Feeds are created when a data provider provides the first FITS image. In order for the “fitspipe-get” program to work properly, the FITS Server must have received a first frame before the “fitspipe-get” program is started. In order to accomplish this, a frame will be populated in the FITS Server just after the server is started.

6.4 FITS Client (fitspipe-put)

As illustrated in figure 2 image data from the Guide Control Process will be sent to the fitspipe FITS server via the fitspipe-put FITS client. Although it is possible for the fitspipe FITS server to receive FITS images from multiple clients, only a single source will be used in the SPIRou guiding implementation.

The fitspipe-put client works by reading in concatenated FITS files from STDIN and passes these on to the fitspipe FITS server. Only 2D (NAXIS=2), Basic FITS (no extensions), at 16 bits-per-pixel are accepted. Using the previous example where the fitspipe FITS server was started on port 9999, the following usage indicates how the fitspipe-put FITS client will be started for SPIRou.

```
spirou_gcp | fitspipe-put server=spirou-session port=9999 feed=default
```

In this example, the SPIRou Guide Control Process is started by starting a process named "spirou_gcp". The STDOUT feed from the Guide Control Process is fed into STDIN of the fitspipe-put process that is pointing to the fitspipe FITS server that was previously started on the SPIRou session host computer.

6.5 FITS Client (fitspipe-get)

Once the fitspipe FITS server exists, the fitspipe-get client is used to handle making a socket connection to the fitspipe FITS server and converting the stream of FITS images to something that can be read from STDOUT. As a result, clients that want to make use of the FITS files, don't need to be concerned with the underlying socket communications to the fitspipe FITS server.

The fitspipe-get utility takes that same parameters as fitspipe-put plus an additional parameter called fullheader=true|false which is described below.

```
fitspipe-get server=spirou-session port=9999 feed=default fullheader=false | ...
```

By default, the fitspipe FITS server sends a minimal, abbreviated header with each video frame. This is fine when the connection is being used, for example, to display the images for the user with a graphical display program such as zimfits. For other connections, which may want a complete set of metadata and timestamps, the server should be sent the option "fullheader=true". It is expected that each of the components using data from the fitspipe FITS server with SPIRou will need the complete set of metadata.

6.6 FITS Cube Saver Design

This process will receive FITS images from STDIN. These images originate from the fitspipe FITS server and are provided to the cube saver via the fitspipe-get client. This process will generate the following products.

- **Raw Guide Cubes** - Cubes generated from images generated by the Fast Guiding Process
- **Raw Acquisition Cubes** - Cubes generated from the raw acquisition images during an acquisition sequence
- **Stitched & Processed Acquisition Images** - Single frame stitched and processed composite acquisition image
- **Raw Focus Cubes** - Cubes generated from the 20 to 30 seconds of individual focus frames
- **Stacked & Processed Focus Images** - Single image based on a stack of the previously noted raw focus frames along with relevant processing

Of the previously noted products, the guide cubes will require the greatest amount of disk space and processing overheads. Using a baseline guide frame size of 32x32 pixels at a guide rate of 50Hz will generate data at the rate over 6 Megabytes per minute. Over the course of many observing nights this will certainly add up. Since the purpose of gathering the guider data is for diagnostic purposes, it may only be necessary to keep the data for a maximum of a few weeks. Currently, guider data for ESPaDOnS is only kept for the one observing run. Once a new observing run starts, the guider data from the previous run is deleted. Using the same approach with SPIRou would probably be reasonable.

The data associated with acquisition and focus sequences will be permanently kept. This data doesn't require much space and has both diagnostic and historical value.

The FITS Cube Saver will also receive images with an exposure type of "IMAGING". There are no plans to save these images, since they are only designed to be viewed in the GUI.

Image data will be saved in a directory on the session host using the following file naming convention.

```
YYYYMMDD-hhmmssSSS<ext>.fits
```

- **YYYY** - Four digit year
- **MM** - Month (01..12)
- **DD** - Day (01..31)
- **hh** - Hour (00..23)
- **mm** - Minute (00..59)
- **ss** - Second (00..59)
- **SSS** - Millisecond (000...999)
- **ext** - Extension (gc - raw guide cube, ac - raw acquisition cube, ap - stitched and processed acquistion image, xc - raw focus cube, xp - stacked and processed focus image)

Two dimensional images, such as the stitched acquisition image and processed focus images, will be stored as NAXIS=2 with no extensions. The headers for this images will be largely unchanged from what is produced by the Guide Control Process.

Three dimensional images, such as guide cubes, raw focus cubes, and raw acquisition cubes, will be stored as NAXIS=3 with an additional binary table extension. The binary table will be used to store the following types of header information that is normally stored in the individual FITS frames.

- UNIXTIME
- CENTER_X and CENTER_Y
- SVOLT_X and SVOLT_Y
- RVOLT_X and RVOLT_Y
- TCS_X and TCS_Y
- GDSTATE

Whenever the FITS Cube Saver is processing a stream of FITS frames, it will finalize the current cube and start a new cube whenever one of the following events occurs.

- ETYPE changes

- ETIME changes
- RA, DEC, or EQUINOX changes
- 60 seconds has passed since the current cube was started.

The previously defined rules will help ensure that the size of the cubes remains manageable and since the cubes are primarily designed for diagnostic purposes, it becomes easier to view and search the metadata in case issues occur.

7 Image Viewer Design

This section discusses the design of the image viewer for SPIRou. The image viewer is a GUI that enables the observer to manually initiate focus sequences, trigger acquisitions, and initiate guiding. During normal science operations, these activities are triggered automatically by commands received from the queue system. Whether manually initiated or automatically triggered, the image viewing GUI provides control and feedback to the observer.

7.1 Interface Mockup

Figure 18 illustrates a mockup of what the image viewer GUI will look like for SPIRou. This interface is designed to support both passive viewing from the observer while observations are performed via queue as well as active interaction during commissioning, engineering, and whenever human intervention is required. The mockup includes all the buttons and interface elements that may be visible. However, some of these items may be disabled or hidden from view depending upon the particular operation and state of the guiding/acquisition system. Each of the elements is described in detail in the following list. Elements are described in a left-to-right top-down manner.

- **ISU Correction Scatter Plot** - This GUI element will show an intensity scatter plot corresponding to ISU corrections when guiding is in progress. The plot will be reset each time guiding is restarted. The plot is generated much like an image where individual counts at pixel locations increase depending upon the SVOLT_X and SVOLT_Y FITS headers containing ISU sent voltage values. Ideally, the intensity scatter plot will resemble a gaussian distribution centered at the origin much like the flux from a star on an image. The goal of this interface element is to provide a visual view of the ISU corrections. In addition, it can help identify cases where the ISU is not kept at a position that maximizes its dynamic range. This plot is only available when guiding is in progress and the TCS is being fed corrections. This is indicated in the FITS headers for guide frames when the “GDSTATE” header indicates “GUIDING”.
- **TCS Correction Scatter Plot** - This GUI element will show an intensity scatter plot corresponding to corrections sent to the TCS when guiding is in progress. The plot will be reset each time guiding is restarted. The plot is generated much like an image where individual counts at pixel locations increase depending upon the TCS_X and TCS_Y FITS headers containing arcsecond corrections in RA and Dec sent to the TCS. This plot is only available when guiding is in progress and the TCS is being fed corrections. This is indicated in the FITS headers for guide frames when the “GDSTATE” header indicates “GUIDING”.
- **ISU X & Y Correction Strip Chart** - When guiding is in progress, this chart will show a strip chart of the X & Y correction values sent to the ISU. The information for this plot is extracted from the SVOLT_X and SVOLT_Y FITS headers contained within the guiding FITS frames. In order to reduce the scrolling speed of the chart, values may be averaged over time (perhaps average every 10 samples). This chart is only available for guide frames containing valid ISU corrections. This should be true whenever the FITS headers for guide frames has a “GDSTATE” header indicating either “ACQUIRE” or “GUIDING”.
- **TCS RA & Dec Correction Strip Chart** - When guiding is in progress, this chart will show the RA & Dec corrections in arcseconds that are being sent to the TCS. The information for this plot is extracted from the TCS_X and TCS_Y FITS headers contained within the guiding FITS frames. This plot is essentially the same

plot as one would see on the TCS interface. This chart is only available for guide frames containing valid TCS corrections. This is indicated in the FITS headers for guide frames when the "GDSTATE" header indicates "GUIDING".

- **Status** - This indicates the overall status for what is currently happening with regards to the guiding camera. Possible statuses include "Camera Off", "Imaging", "Focus Sequence", "Acquisition", "Guide Acquire", "Guiding", and "Error". This information is based on a combination of the GUI state as well as the "ETYPE" and "GDSTATE" FITS headers within the images.
- **Exposure Type** - This field is blank if the camera is off. Otherwise, the field is set to the value populated in the "ETYPE" FITS header.
- **Timestamp** - This field is set to a formatted timestamp based on the "UNIXTIME" FITS header. The time format will be set to HH:MM:SS.sss. If the camera is off this field will be blank.
- **Image Sequence** - This is an incrementing sequence number that is reset each time a particular exposure type is initiated. The value of the field is extracted from the "SEQNUM" FITS header. If the camera is off this field will be blank.
- **ISU Correction** - This field contains the X and Y voltage correction values sent to the ISU. These values are extracted from the SVOLT_X and SVOLT_Y FITS headers in the images whenever the ETYPE is set to "GUIDE". If the camera is off the the exposure type is something other than a guiding exposure, this field will be blank.
- **TCS Correction** - This field contains the RA and Dec corrections in arcseconds sent to the TCS. These values are extracted from the TCS_X and TCS_Y FITS headers when the ETYPE is set to "GUIDE" and the GDSTATE is set to "GUIDING". If the camera is off this field will be blank.
- **Seeing** - This field is calculated by the Viewing Analysis and should be periodically updated in the Status Server. A monitor will be placed on this Status Server object and it will be updated whenever it changes. If the value goes "EXPIRED" in the Status Server, it will be blank.
- **Position** - This field displays the X and Y position of the cursor on the main display. If the cursor is not over the main display area, this field will be blank.
- **Value** - This field displays the flux value of the pixel associated with the cursor on the main display. If the cursor is not over the main display area, this field will be blank.
- **Zoom** - This field will indicate the zoom factor that is currently used for the image data.
- **Guide Camera Image Data** - This is the display area for the pixel data within the FITS file. If the user right-clicks on the imaging area, a popup menu will appear upon which the user can choose the color palette and display algorithm (min-max, histeq, median adjust, etc.) for the pixel data. If the camera is turned off, the imaging field will be completely black.
- **Window Center** - This is the central location of the imaging raster. This is represented in terms of arcseconds where the origin is defined to be the aperture stop. This field is calculated based on the WIN_X0, WIN_X1, WIN_Y0, WIN_Y1, NULL_X, NULL_Y, and PIXSCALE FITS headers.
- **Window Size** - The is the size of the window in arcseconds. The size if represented as the width and hight of the raster. This field is calculated based on the WIN_X0, WIN_X1, WIN_Y0, WIN_Y1, NULL_X, NULL_Y, and PIXSCALE FITS headers.
- **Exposure Time** - This is the exposure time of the image. The value for this field is extracted from the ETIME FITS header.
- **Neutral Density** - This is the amount of neutral density filter applied to the guider. The value for this field is extracted from the Status Server. The neutral density wheel position is controlled by the Cassegrain Agent. This agent is also responsible for populating the position of the wheel in the Status Server.

- **Guide Button** - This button is used to set the subraster and exposure time for normal guiding. This button is only available when the guider is placed in “Imaging” mode.
- **Acquire Button** - This button is used to set the subraster and exposure time for a full raster frame where the observer may be looking for candidate guide stars. This button is only available when the guider is placed in “Imaging” mode.
- **Custom Button** - This button is used to bring up a dialog box where a customized subraster, exposure time, and neutral density wheel position can be set. This button is only available when the guider is placed in “Imaging” mode.
- **Imaging On/Off** - This button is used to either turn on or off imaging mode. Imaging mode is defined as receiving a stream of pixels with the subraster and exposure time defined by the previously described list items. This button is not visible when guiding, an acquisition sequence, or a focus exposure is in progress. The text of the button will change between “Imaging On” and “Imaging Off” depending upon what the current imaging state is.
- **Offset Guide Star** - This button allows the observer to choose a star for offset guiding. This button is only visible after an acquisition sequence or when imaging is turned on.
- **Center Star** - This button allows the observer to choose a star and place it on the aperture stop. This button is only visible after an acquisition sequence or when imaging is turned on.
- **Focus on Star** - This button allows the observer to choose a star and trigger a focus sequence to be performed on the chosen star. This button is only visible after an acquisition sequence or when imaging is turned on.
- **Perform Acquisition** - This button triggers an acquisition sequence to be performed. This button is only visible when guiding, an acquisition sequence, or a focus sequence are not already in progress.
- **Initiate Guiding** - This button is visible when imaging is turned on or after an acquisition sequence. Once the button is pressed, an attempt will be made to initiate guiding based on an object centered on the the aperture stop or a previously defined offset guide star.
- **Abort** - This button is visible when either guiding, an acquisition sequence, or a focus sequence is in progress.

7.2 Design Overview

The design for the SPIRou image viewing GUI is somewhat similar to the ESPaDOnS guider GUI with improvements. The GUI will be written in C using the Gtk+ toolkit. This is the same language and graphics toolkit library used for other image rendering projects at CFHT. As a result, the image rendering core can be reused from other projects. FITS images received over STDIN are rendered in the GUI and the metadata within the FITS headers is used to update various fields in the GUI. A connection is established to the Status Server to retrieve and update GUI state information. Finally, a socket connection is established to the Controller Agent to send commands to the instrument session. An overview of the architecture and interactions for the image viewing GUI is shown in figure 19. Each of the items within Image Display Process box in the figure is described in more detail within the subsequent sections.

7.2.1 STDIN FITS Image Handling Thread

This thread is basically set up as a loop. It will block waiting for image data to arrive on STDIN. If image data is available, it will read the image into memory using the FITS manipulation utilities in libfh. The metadata in the FITS header will be used to update the various fields in the GUI described in section 7.1 along with the scatter plots and strip charts. The image data will be used to update the main imaging area of the GUI. The image data will be processed according to the processing algorithm selected by the observer (contrast stretch, histogram equalization, zscale, etc.). Updates to the GUI will be wrapped within gdk_threads_enter() and gdk_threads_leave() calls so the updates are properly performed.

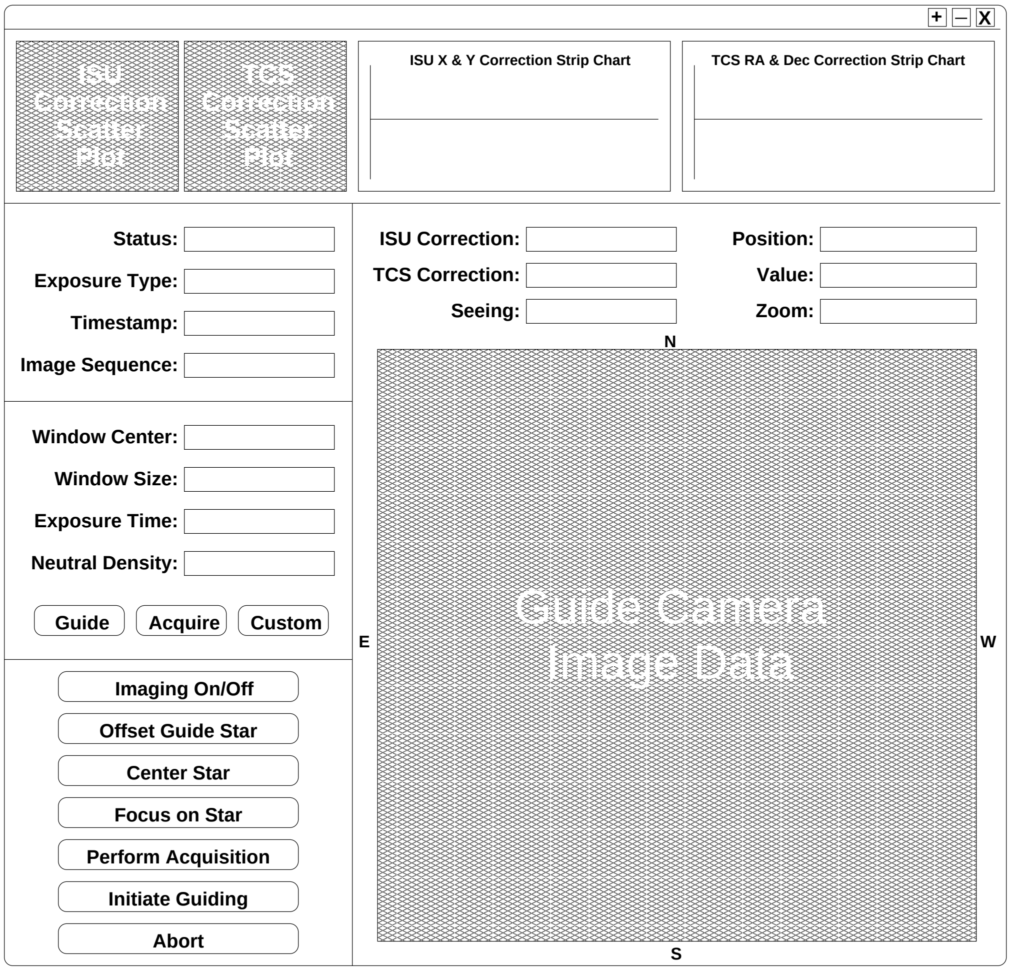


Figure 18: SPIRou Guider Display Mockup

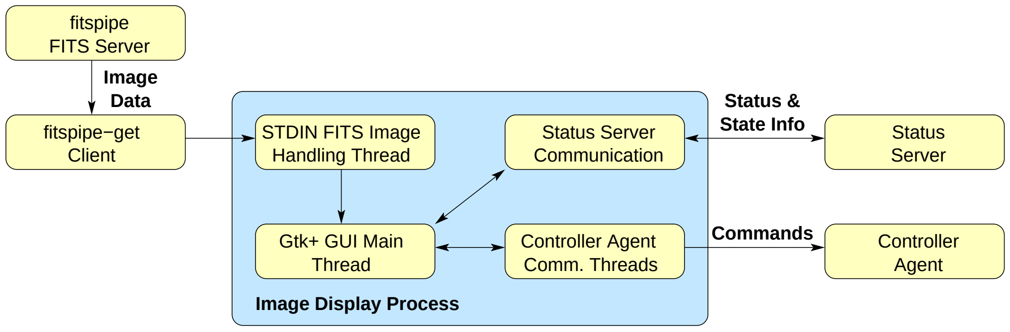


Figure 19: SPIRou Guider Display Architecture

It is possible that image data will be received more rapidly than the GUI can actually display it. In this case, the image handling thread will prioritize reading in the FITS metadata from each frame and only rendering the last frame available on STDIN. This way all the available data on STDIN is read before performing the updates to the GUI.

7.2.2 Gtk+ GUI Main Thread

This is actually the “main” thread of the process. Once the GUI is started, all observer interactions with the GUI will result in callback functions being triggered. These callback functions may result in Status Server API calls, commands to the Controller Agent, or modifications to the GUI.

7.2.3 Status Server Communication

The SPIRou GUI will interact with the Status Server in order to retrieve the status of external items such as the position of the neutral density wheel. A monitor will be placed on any items of interest in the Status Server. These monitors will be updated whenever an ssPoll() API call is made. This call will be made approximately every second and is performed within a timer callback routine that is set up with the gtk_timer_add() API call. Since the socket connection to the Status Server is persistent, these calls don’t require much overhead and are performed within the main thread.

7.2.4 Controller Agent Communication Threads

During normal operations there will be a single “master” control GUI. This GUI will be started under the “spirou” user account and is eligible to send commands to the create a connection to the Controller Agent. It is possible to start additional GUIs in a view-only fashion. These GUIs are considered view-only since they won’t send commands to the Controller Agent.

In order to simplify the interface between the image display GUI and the Controller Agent, commands to the agent will be sent using the normal **clicmd** interface. In this case, the command will be sent by executing a clicmd shell command using the cfht_exec() libcfht library call with the CFHT_EXECBGH parameter specifying a callback function to be invoked when the call is complete. At most, two calls to the Controller Agent could be active at the same time (request followed by an abort). Once the command is executed, the SIGCHLD signal will be blocked using the cfht_signal_block() API call until the main thread is ready to handle possible interruptions to service callbacks. At that time, the SIGCHLD signal will be unblocked using the cfht_signal_unblock() API call. Commands sent using the **clicmd** interface are handled by director just like any other commands.

The following set of commands can be sent to director and are handled by the Controller Agent.

- **ACQUIRE** - Used to trigger an acquisition sequence
- **GUIDE** - Used to initiate guiding
- **GOFFSET** - Used to set a guiding offset
- **GSTAR** - Used to center a star on the entrance aperture
- **FSTAR** - Used to trigger a focus sequence on the selected star
- **IMAGEGO** - Used to start imaging mode

7.2.5 Command Line Syntax

When the SPIRou image viewing GUI is started using the following command line option, the following help summary will be provided.

```
> spgui -help

SPIRou Image Viewing GUI
usage: spgui [-control]
```

8 Astrometry Library

In order to meet the accurate pointing requirements of WIRCam, a number of astrometry routines were created to map items detected on the focal plane to the sky and determine the scale, rotation, and translation between the sky and the focal plane. Once this was successfully used with WIRCam, the software was repackaged into a library so it could be used with other intruments at CFHT (MegaCam and ESPaDOnS). It should be possible to use the same library with SPIRou. The library is set up to provides API calls that enable the following sequence of steps.

1. Detect stars above a given SNR level over the image background. CFHT has developed its own star detection routines which are much faster than Sextractor and operate on image data already in memory.
2. Retrieve stars from the 2MASS catalog based on a given sky position and search size in arcseconds.
3. Convert the stars from the catalog to a position expressed as relative distances in RA and Dec from the catalog query position.
4. Convert the detected stars to a position in millimeters from the center of the detector or mosaic and finally to a position expressed as relative distances in RA and Dec from the center of the detector or mosaic. This transformation uses a user defined optical distortion map.
5. Perform a match between the relative RA-Dec catalog positions and detected star positions.

9 Reference Documents

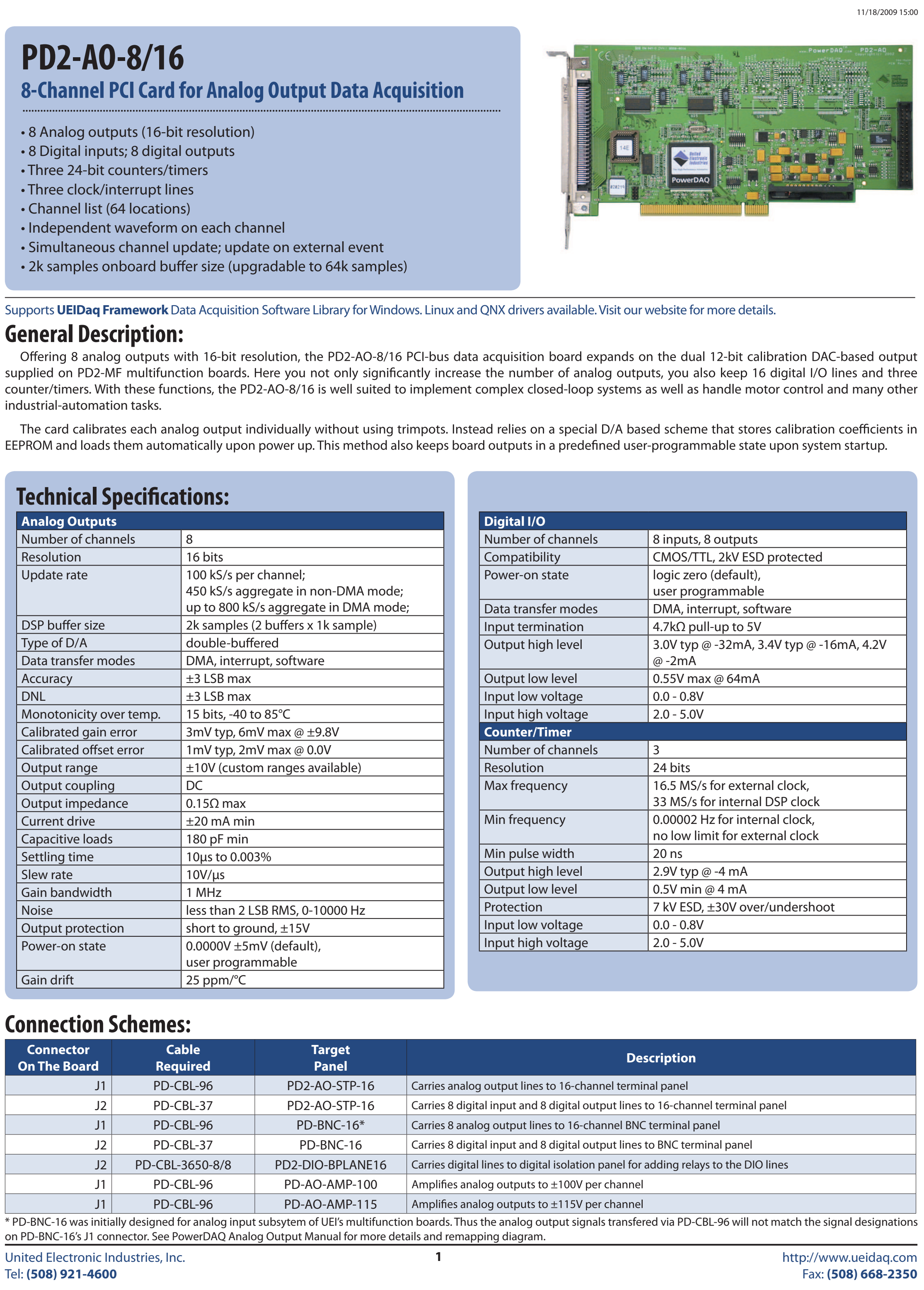
This section contains links to other CFHT reference documents that are not found in the SPIRou DocuShare system. SPIRou specific information can be found in the SPIRou DocuShare repository at http://ged.obs-mip.fr/omp/dsweb/HomePage.

- **Instrument Design Specification (IDS)** - http://software.cfht.hawaii.edu/InstrumentDesignSpecs.pdf
- **Director** - http://software.cfht.hawaii.edu/director
- **Status Server** - http://software.cfht.hawaii.edu/statserv
- **Client-Server Socket Library** - http://software.cfht.hawaii.edu/sockio
- **FITS Handling Library (libfh)** - http://software.cfht.hawaii.edu/libfh
- **Data Acquisition** - http://software.cfht.hawaii.edu/DataAcquisition
- **SPIRou High-Level Instrument Control Command** - SPIROU-4900-CFHT-RP-00590.
- **SPIRou TipTilt & Viewing Camera Control and Software Design** - SPIROU-4200-ASIAA-RP-00646.
- **SPIRou TipTilt & Viewing Interface Control Document** - SPIROU-4200-ASIAA-RP-00527.
- **SPIRou Cassegrain Software Control Command Design** - SPIROU-4100-IRAP-RP-00638.

10 Acronyms

Acronym	Description
ABSS	Allen-Bradley PLC to Status Server software process to map data from the PLC to the Status Server
API	Application Programming Interface
Dec	Declination
DetCom	Detector Communication Agent
EDT	Engineering Design Team
EPICS	Experimental Physics and Industrial Control System
ESPaDOnS	Echelle SpectroPolarimetric Device for the Observation of Stars at CFHT
FDR	Final Design Review
FITS	Flexible Image Transport System
FOV	Field of View
FLIR	Forward Looking Infrared (Company based its name on acronym)
GUI	Graphical User Interface
IDS	Instrument Design Specification
IfA	Institute for Astronomy
IOC	Input Output Controller
ISU	Image Stabilization Unit
NEO	New Environment for Observing
OAP	Observatory Automation Project
Pan-STARRS	Panoramic Survey Telescope and Rapid Response System
PCI	Peripheral Component Interconnect
PI	Principal Investigator
PLC	Programmable Logic Controller
PDR	Preliminary Design Review
QSO	Queued Service Observations
RA	Right Ascension
RTAI	Real-Time Application Interface
SPIRou	SpectroPolarimetre Infra-Rouge (Near Infrared Spectropolarimeter)
SWIR	Shortwave Infrared
TCS	Telescope Control System
2MASS	Two Micron All Sky Survey
WIRCam	Wide-Field Infrared Camera At CFHT

11 Appendix



11/18/2009 15:00

PD2-AO-8/16

8-Channel PCI Card for Analog Output Data Acquisition

- 8 Analog outputs (16-bit resolution)
- 8 Digital inputs; 8 digital outputs
- Three 24-bit counters/timers
- Three clock/interrupt lines
- Channel list (64 locations)
- Independent waveform on each channel
- Simultaneous channel update; update on external event
- 2k samples onboard buffer size (upgradable to 64k samples)

Supports **UEIDaq Framework** Data Acquisition Software Library for Windows. Linux and QNX drivers available. Visit our website for more details.

General Description:

Offering 8 analog outputs with 16-bit resolution, the PD2-AO-8/16 PCI-bus data acquisition board expands on the dual 12-bit calibration DAC-based output supplied on PD2-MF multifunction boards. Here you not only significantly increase the number of analog outputs, you also keep 16 digital I/O lines and three counter/timers. With these functions, the PD2-AO-8/16 is well suited to implement complex closed-loop systems as well as handle motor control and many other industrial-automation tasks.

The card calibrates each analog output individually without using trimpots. Instead relies on a special D/A based scheme that stores calibration coefficients in EEPROM and loads them automatically upon power up. This method also keeps board outputs in a predefined user-programmable state upon system startup.

Technical Specifications:

Analog Outputs	
Number of channels	8
Resolution	16 bits
Update rate	100 kS/s per channel; 450 kS/s aggregate in non-DMA mode; up to 800 kS/s aggregate in DMA mode;
DSP buffer size	2k samples (2 buffers x 1k sample)
Type of D/A	double-buffered
Data transfer modes	DMA, interrupt, software
Accuracy	±3 LSB max
DNL	±3 LSB max
Monotonicity over temp.	15 bits, -40 to 85°C
Calibrated gain error	3mV typ, 6mV max @ ±9.8V
Calibrated offset error	1mV typ, 2mV max @ 0.0V
Output range	±10V (custom ranges available)
Output coupling	DC
Output impedance	0.15Ω max
Current drive	±20 mA min
Capacitive loads	180 pF min
Settling time	10µs to 0.003%
Slew rate	10V/µs
Gain bandwidth	1 MHz
Noise	less than 2 LSB RMS, 0-10000 Hz
Output protection	short to ground, ±15V
Power-on state	0.0000V ±5mV (default), user programmable
Gain drift	25 ppm/°C

Digital I/O	
Number of channels	8 inputs, 8 outputs
Compatibility	CMOS/TTL, 2kV ESD protected
Power-on state	logic zero (default), user programmable
Data transfer modes	DMA, interrupt, software
Input termination	4.7kΩ pull-up to 5V
Output high level	3.0V typ @ -32mA, 3.4V typ @ -16mA, 4.2V @ -2mA
Output low level	0.55V max @ 64mA
Input low voltage	0.0 - 0.8V
Input high voltage	2.0 - 5.0V
Counter/Timer	
Number of channels	3
Resolution	24 bits
Max frequency	16.5 MS/s for external clock, 33 MS/s for internal DSP clock
Min frequency	0.00002 Hz for internal clock, no low limit for external clock
Min pulse width	20 ns
Output high level	2.9V typ @ -4 mA
Output low level	0.5V min @ 4 mA
Protection	7 kV ESD, ±30V over/undershoot
Input low voltage	0.0 - 0.8V
Input high voltage	2.0 - 5.0V

Connection Schemes:

Connector On The Board	Cable Required	Target Panel	Description
J1	PD-CBL-96	PD2-AO-STP-16	Carries analog output lines to 16-channel terminal panel
J2	PD-CBL-37	PD2-AO-STP-16	Carries 8 digital input and 8 digital output lines to 16-channel terminal panel
J1	PD-CBL-96	PD-BNC-16*	Carries 8 analog output lines to 16-channel BNC terminal panel
J2	PD-CBL-37	PD-BNC-16	Carries 8 digital input and 8 digital output lines to BNC terminal panel
J2	PD-CBL-3650-8/8	PD2-DIO-BPLANE16	Carries digital lines to digital isolation panel for adding relays to the DIO lines
J1	PD-CBL-96	PD-AO-AMP-100	Amplifies analog outputs to ±100V per channel
J1	PD-CBL-96	PD-AO-AMP-115	Amplifies analog outputs to ±115V per channel

* PD-BNC-16 was initially designed for analog input subsytem of UEI's multifunction boards. Thus the analog output signals transfered via PD-CBL-96 will not match the signal designations on PD-BNC-16's J1 connector. See PowerDAQ Analog Output Manual for more details and remapping diagram.

United Electronic Industries, Inc. 1 http://www.ueidaq.com
Tel: **(508) 921-4600** Fax: **(508) 668-2350**

Figure 20: PowerDAQ PD2-AO-8/16 Analog Output PCI Card Data Sheet

Tau SWIR 15XR

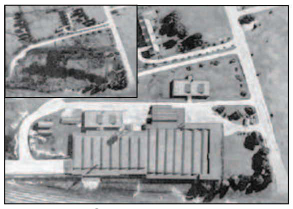
Low-Noise, Shortwave Infrared Camera (15 µm Pixels)

The Tau SWIR 15XR joins FLIR's Tau family of best-in-class small, light weight, low-power camera cores that deliver shortwave infrared imaging with very high sensitivity. A read noise of less than 35 electrons rms coupled with an extended spectral range of 600 – 1700 nm provides for outstanding low-light performance in a variety of applications. Designed for a variety of OEM applications, the Tau SWIR 15XR provides outstanding image quality and performance for machine vision, a variety of medical, agricultural, semiconductor/ solar panel inspection applications, as well as high temperature endpoint and defect monitoring.

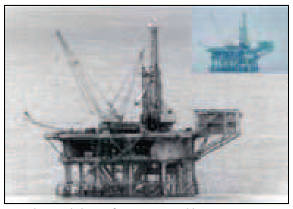
Tau SWIR 15XR cameras incorporate a high-resolution (640 × 512) Indium Gallium Arsenide (InGaAs) 15-micron pixel pitch focal plane array that features variable exposure control, high frame rates (in sub-windowing mode), nearly zero image lag, and high quantum efficiency. The camera features Camera Link and analog video outputs, and delivers exceptional image quality in a variety light levels using advanced automatic gain control and non-uniformity correction. Weighing only 130 g (4.5 oz) and measuring only 3.8 × 3.8 × 4.8 cm (1.5" × 1.5" × 1.9") Tau SWIR 15XR is ideal for OEM applications.

As the world's largest commercial infrared camera supplier, FLIR delivers the worldwide support, quality, and reliability that you demand at the pricing you need.

Features	Benefits
15 µm Pixels	Better compatibility with optics
Extended Spectral Range of 0.6 – 1.7 µm	Improved night-time performance and compatible with a variety of illumination sources
<1% Image Lag Frame-to-Frame	Eliminates smear (no persistence)
Low Power	Low power budget requirements
Compact Size and Lightweight	Fits into small space-claim applications
>High Frame Rate (sub window)	Works in specialized applications
Affordable SWIR Solution	Meet your cost targets
Digital and Analog Video Outputs	Flexible integration
Thinned InGaAs Detector	Reduced blooming when viewing intense light sources in high gain


Detect camouflage

See through many paints to visualize hidden details


Cut through haze for improved long range image quality

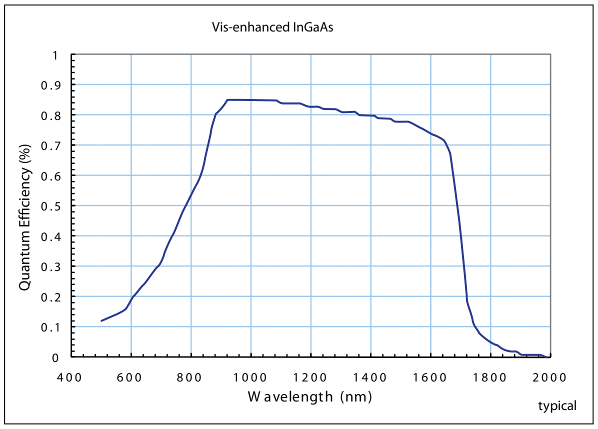
Passive night vision

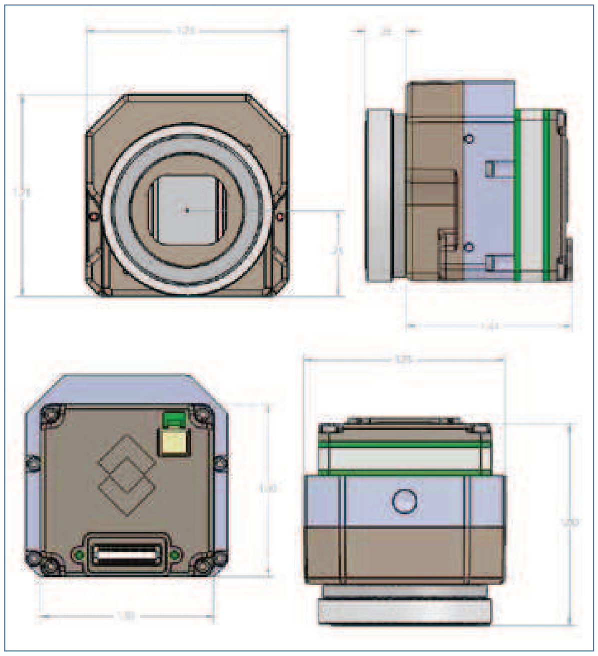
Contact FLIR at 719.598.6006 to speak with a low-light imaging expert about your requirements.

Figure 21: Tau SWIR 15XR Camera Data Sheet - Page 1

Tau SWIR 15XR Camera Specifications

Parameter	Value	Comments
Sensor Type	InGaAs	
Format	640 × 480 (analog), 640 × 512 (digital)	Analog VGA limits to 640 × 480
Pixel Size	15 µm	
Active Area (H × V)	Analog: 0.38″ × 0.28″ / 9.6 × 7.2 mm	Digital: 0.38″ × 0.3″ / 9.6 × 7.7 mm
Active Area (Diagonal)	Analog: 0.5″ / 12.0 mm	Digital: 0.5″ / 12.3 mm
Active Area (Area)	Analog: 0.11″^{2} / 69 mm^{2}	Digital: 0.11″^{2} / 74 mm^{2}
Fill Factor	100%	
Quantum Efficiency	0.6 to 1.7 µm	See QE plot to the right
Noise @ Sensor Temp = +10°C	Low Gain < 200 e rms; High Gain < 35 e rms	See Note 1 below
NEI	< 5.5 × 10^{8} photons/ cm^{2}/sec @1550 nm	See Note 1 below
Full Well	Low Gain = 400,000 e, High Gain = 10,000 e	Typical
Dynamic Range	Low Gain = 66 dB, High Gain = 50 dB	Typical
Operability	>99.5%	
Max Frame Rate	30 FPS (for 60 FPS contact factory)	
Exposure Time	500 ns to 500 ms	
Region of Interest (ROI)	Full resolution to 64 × 64 at 750 FPS	
Operability	>99.5%	
Image Correction	2-point (Offset / Gain) – user configurable	
Image Lag	<1% frame-to-frame	Assumes no over-exposure
Digital Data	14-bit low voltage CMOS or Camera Link*	
Analog Output	NTSC compliant	
Mechanical / Environmental		
Weight	130 g (M42 lens mount) 160 g (C-type lens mount)	See Note 2 below
Dimensions	38 × 38 × 48.25 mm	See Note 3 below
Lens Mount	C-Mount or M42 mount	
Operating Temperature (full performance)	-20°C to 55°C	Full performance
Operating Temperature (degraded performance)	Between -40°C to -20°C and 55°C to 71°C	See Note 4 below
Storage Temperature	-50°C to 85°C	
Humidity	<95%	Non-condensing
Power Requirements		
DC Input Voltage	12 VDC	
Power Dissipation	4 Watts typical	At 30°C case temp See Note 1 below





Note 1: Noise in high gain can be improved either by operating in a cooler environment or by increasing camera power dissipation. For example, noise can be reduced to less than 20 electrons (NEI of <3 × 10^{8} photons/cm^{2}/sec) at a sensor temperature of 0°C.

Note 2: Much of the camera weight is associated with the housing and lens mounting hardware. Custom core designs can be used to reduce this weight.

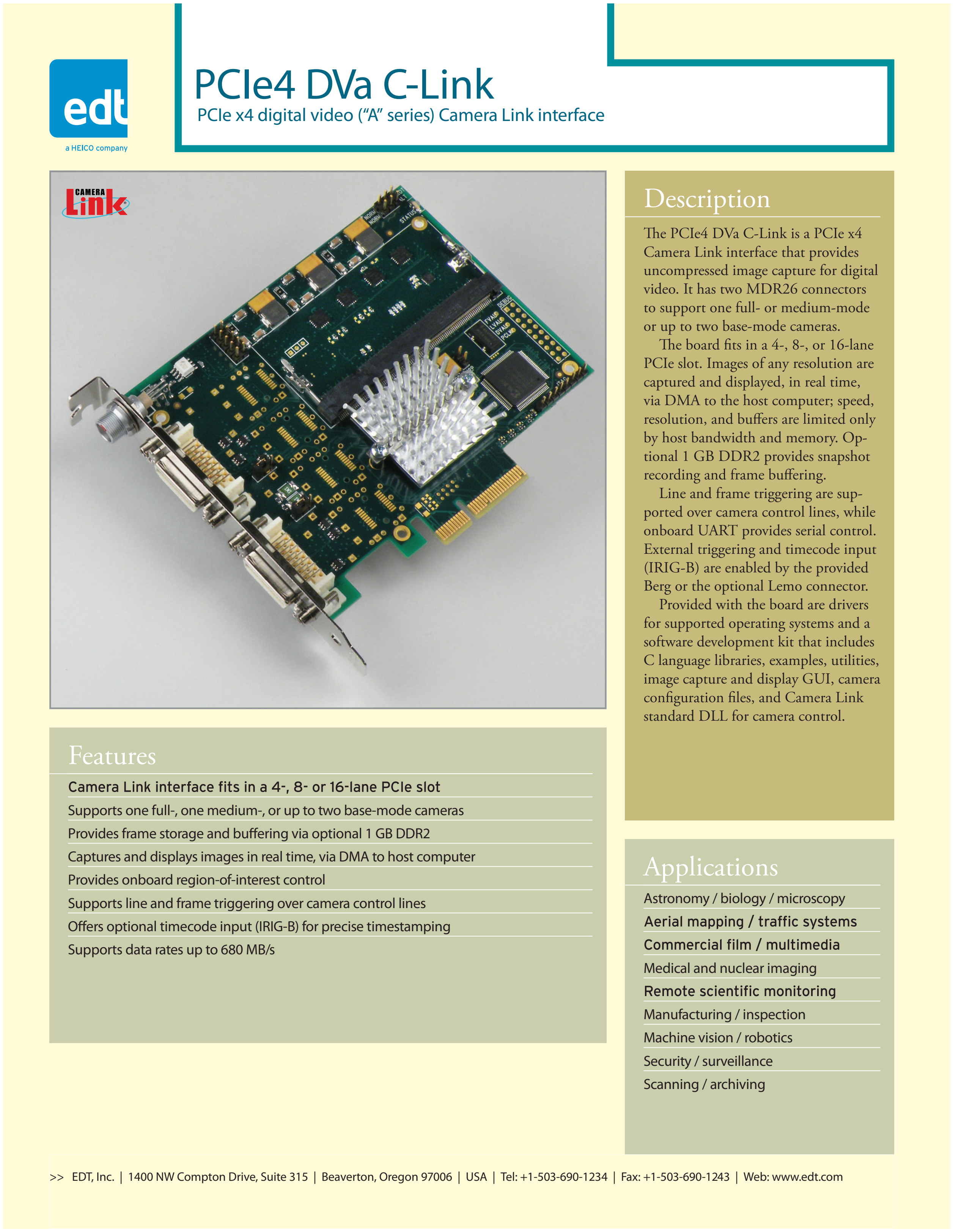
Note 3: Dimensions are typical and depend upon exact lens mount configuration chosen.

Note 4: Degraded performance results in higher random noise in high-gain mode.

FLIR Systems, Inc.
5061 N. 30th Street, Suite 103
Colorado Springs, CO 80919
PH: +1 719.598.6006
Sales Inquiries: AISsales@flir.com

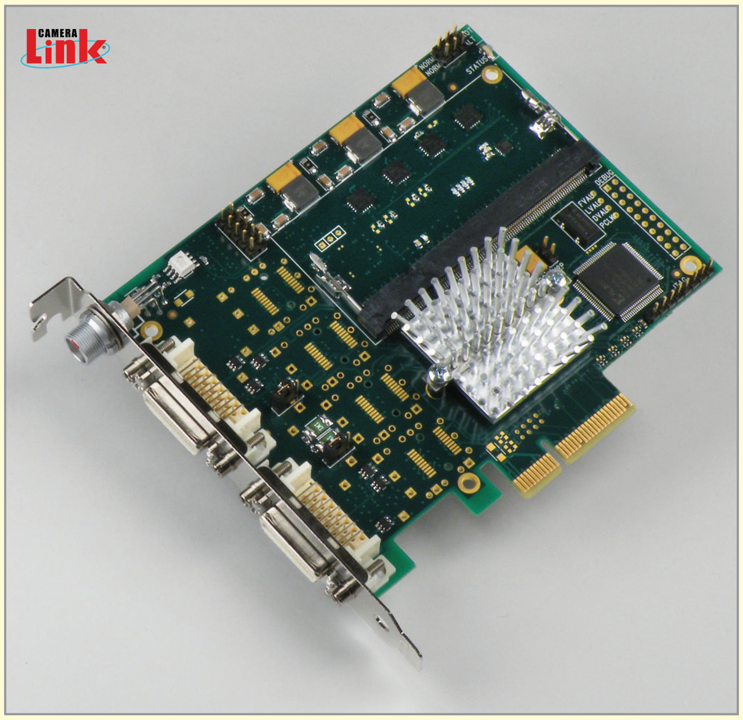
www.flir.com/cvs/cores

Figure 22: Tau SWIR 15XR Camera Data Sheet - Page 2



PCIe4 DVa C-Link

PCIe x4 digital video ("A" series) Camera Link interface



Description

The PCIe4 DVa C-Link is a PCIe x4 Camera Link interface that provides uncompressed image capture for digital video. It has two MDR26 connectors to support one full- or medium-mode or up to two base-mode cameras.

The board fits in a 4-, 8-, or 16-lane PCIe slot. Images of any resolution are captured and displayed, in real time, via DMA to the host computer; speed, resolution, and buffers are limited only by host bandwidth and memory. Optional 1 GB DDR2 provides snapshot recording and frame buffering.

Line and frame triggering are supported over camera control lines, while onboard UART provides serial control. External triggering and timecode input (IRIG-B) are enabled by the provided Berg or the optional Lemo connector.

Provided with the board are drivers for supported operating systems and a software development kit that includes C language libraries, examples, utilities, image capture and display GUI, camera configuration files, and Camera Link standard DLL for camera control.

Features

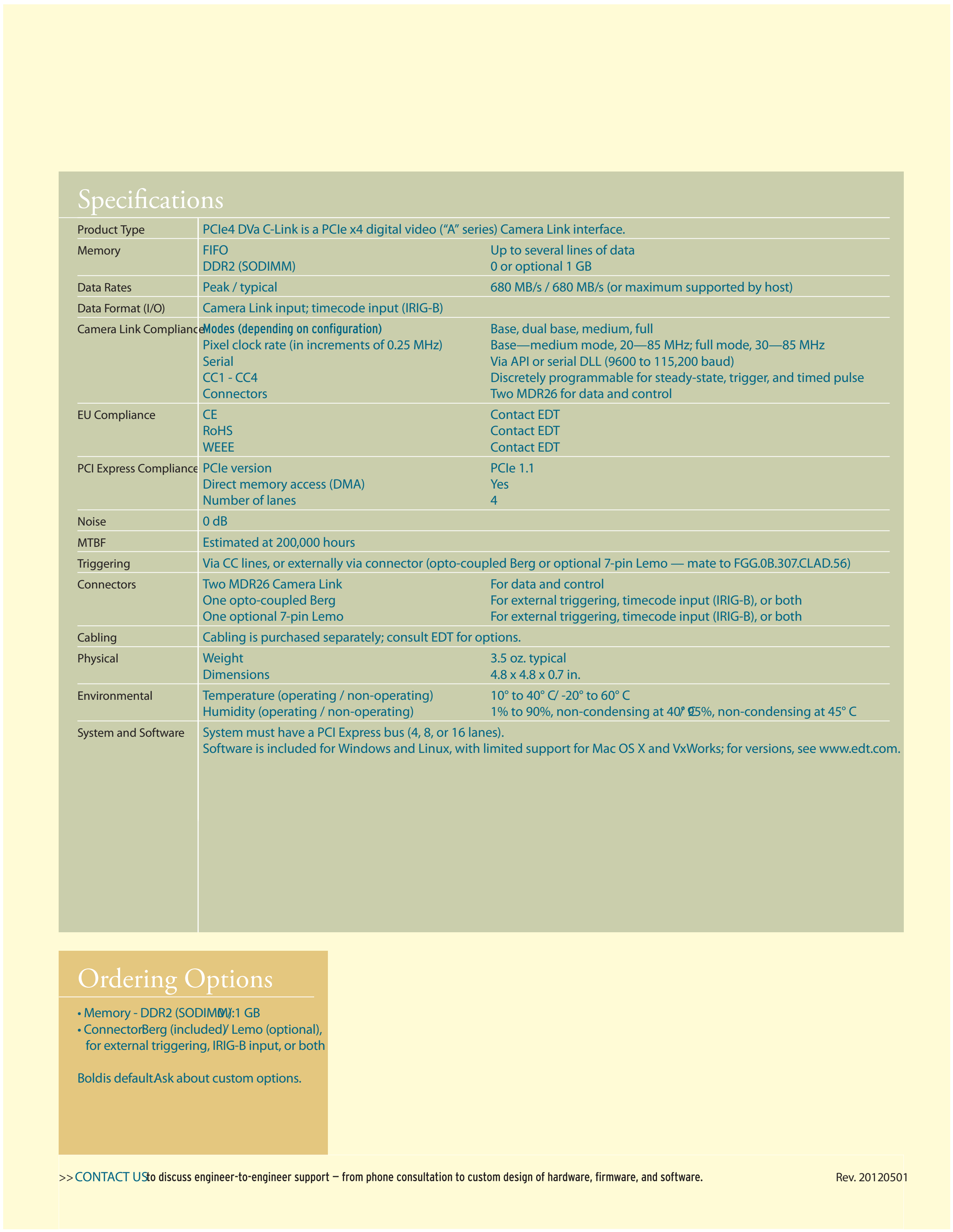
- Camera Link interface fits in a 4-, 8- or 16-lane PCIe slot
- Supports one full-, one medium-, or up to two base-mode cameras
- Provides frame storage and buffering via optional 1 GB DDR2
- Captures and displays images in real time, via DMA to host computer
- Provides onboard region-of-interest control
- Supports line and frame triggering over camera control lines
- Offers optional timecode input (IRIG-B) for precise timestamping
- Supports data rates up to 680 MB/s

Applications

- Astronomy / biology / microscopy
- Aerial mapping / traffic systems
- Commercial film / multimedia
- Medical and nuclear imaging
- Remote scientific monitoring
- Manufacturing / inspection
- Machine vision / robotics
- Security / surveillance
- Scanning / archiving

>> EDT, Inc. | 1400 NW Compton Drive, Suite 315 | Beaverton, Oregon 97006 | USA | Tel: +1-503-690-1234 | Fax: +1-503-690-1243 | Web: www.edt.com

Figure 23: EDT PCIe4 DVa C-Link Camera Link PCI Express Card Data Sheet - Page 1



Specifications

Product Type	PCIe4 DVa C-Link is a PCIe x4 digital video ("A" series) Camera Link interface.	
Memory	FIFO DDR2 (SODIMM)	Up to several lines of data 0 or optional 1 GB
Data Rates	Peak / typical	680 MB/s / 680 MB/s (or maximum supported by host)
Data Format (I/O)	Camera Link input; timecode input (IRIG-B)	
Camera Link Compliance	Modes (depending on configuration) Pixel clock rate (in increments of 0.25 MHz) Serial CC1 - CC4 Connectors	Base, dual base, medium, full Base—medium mode, 20—85 MHz; full mode, 30—85 MHz Via API or serial DLL (9600 to 115,200 baud) Discretely programmable for steady-state, trigger, and timed pulse Two MDR26 for data and control
EU Compliance	CE RoHS WEEE	Contact EDT Contact EDT Contact EDT
PCI Express Compliance	PCIe version Direct memory access (DMA) Number of lanes	PCIe 1.1 Yes 4
Noise	0 dB	
MTBF	Estimated at 200,000 hours	
Triggering	Via CC lines, or externally via connector (opto-coupled Berg or optional 7-pin Lemo — mate to FGG.0B.307.CLAD.56)	
Connectors	Two MDR26 Camera Link One opto-coupled Berg One optional 7-pin Lemo	For data and control For external triggering, timecode input (IRIG-B), or both For external triggering, timecode input (IRIG-B), or both
Cabling	Cabling is purchased separately; consult EDT for options.	
Physical	Weight Dimensions	3.5 oz. typical 4.8 x 4.8 x 0.7 in.
Environmental	Temperature (operating / non-operating) Humidity (operating / non-operating)	10° to 40° C/ -20° to 60° C 1% to 90%, non-condensing at 40° C/ 95%, non-condensing at 45° C
System and Software	System must have a PCI Express bus (4, 8, or 16 lanes). Software is included for Windows and Linux, with limited support for Mac OS X and VxWorks; for versions, see www.edt.com.	

Ordering Options

- Memory - DDR2 (SODIMM): 1 GB
- Connector: Berg (included) / Lemo (optional), for external triggering, IRIG-B input, or both

Bold is default. Ask about custom options.

>> CONTACT US to discuss engineer-to-engineer support – from phone consultation to custom design of hardware, firmware, and software.

Rev. 20120501

Figure 24: EDT PCIe4 DVa C-Link Camera Link PCI Express Card Data Sheet - Page 2