Zwicky Transient Facility (ZTF)   
Software Architecture Document

CIN #xxx

Document Owner: Richard Dekany

Contributions by:

J. Cromer, R. Smith

**Caltech Optical Observatories**

Cahill Center for Astronomy and Astrophysics

Caltech M/C 11-17

1200 E. California Blvd.

Pasadena, CA 91125

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Revision Description** |
| 0.0 | 2014-01-29 | Cromer | Initial version. |
| 0.1 | 2014-03-18 | Cromer | More words and diagrams. |
| 0.2 | 2014-08-10 | Cromer | Added data flow diagrams, focal-plane diagram, more words. |
| 0.3 | 2014-08-11 | Cromer | Exposure walkthrough, FITS header and Frameworks sections added. |
| 0.4 | 2014-08-12 | Cromer | Development tools and testing sections added. |

Table of Contents

Revision History 2

2 Introduction 4

2.1 Scope 4

2.2 Acronyms and Abbreviations 4

2.3 Reference Documents 4

2.4 Point of Contact 5

3 Hardware Overview 5

4 Performance Overview 8

5 Software Overview 9

6 Observing System 9

7 CCD Control and Readout 10

7.1 Overview 10

7.2 Exposure Walkthrough. 10

7.3 FITS File Creation 11

8 Shutter Control 12

9 Power Control 13

10 Dewar Temperature Control and Monitoring 14

11 Dewar Pressure Monitor 14

12 Filter 15

13 Window Condensation 16

14 TCS Interface 16

15 Guiding 17

16 Focus 18

17 Frameworks 18

18 Development Tools 19

19 Testing 19

# Introduction

The purpose of this document is to discuss the architecture of the software to control and read images from the ZTF CCD controller and the additional software and hardware necessary for power, dewar and telescope environment monitoring, filter, focus and guide-camera control.

## Scope

ZTF OS control software covered in this document consists of the following functional modules:

* Observing System module
* CCD control module.
* Shutter control module.
* Dewar pressure monitor.
* Dewar temperature control and monitor.
* AC Power control module and monitor.
* Filter control module.
* Window condensation control and monitor.
* TCS module.
* Focus module.
* Guider module.

## Acronyms and Abbreviations

* AO – Adaptive Optics.
* Archon – Name of CCD controller built by STA.
* DRP – Data Reduction Pipeline.
* DS – Data System.
* FITS – Flexible Image Transport System.
* OS – Observing System.
* STA – Semiconductor Technology Associates.
* TBD – To Be Determined.
* TBC – To Be Confirmed.
* TBR – To Be Revised.
* TCS – Telescope Control System.
* WaSP – Wafer-Scale camera for Prime, a large-format CCD camera for the Palomar 200” prime focus.
* WMKO – William M. Keck Observatory.
* ZTFC – Zwicky Transit Facility Camera.

## Reference Documents

* ZTF Instrument Requirements Document (IRD) *[657 to be completed]*
* ZTF Software Requirements Document *[658 to be completed]*
* Smith and Kaye. ZTF CCD Controller: Architecture and Performance Requirements. August 14, 2013.
* Smith. ZTF Image Quality Requirements. February 1, 2014. (Actual title given at the top of the document is “ZTF Image Quality Assurance.”)
* Conrad, Albert R. *Software Systems in Astronomy*. Springer. 2014.

## Point of Contact

John Cromer, Software Engineer [cro@astro.caltech.edu](mailto:cro@astro.caltech.edu) (626) 395-4118

Roger Smith, Technical Lead [rsmith@astro.caltech.edu](mailto:rsmith@astro.caltech.edu) (626) 395-8780

Richard Dekany, Project Manager [rgd@astro.caltech.edu](mailto:rgd@astro.caltech.edu) (626) 395-6798

# Hardware Overview

Below is the context diagram, which shows the hardware and software environment in which the ZTF software must operate.

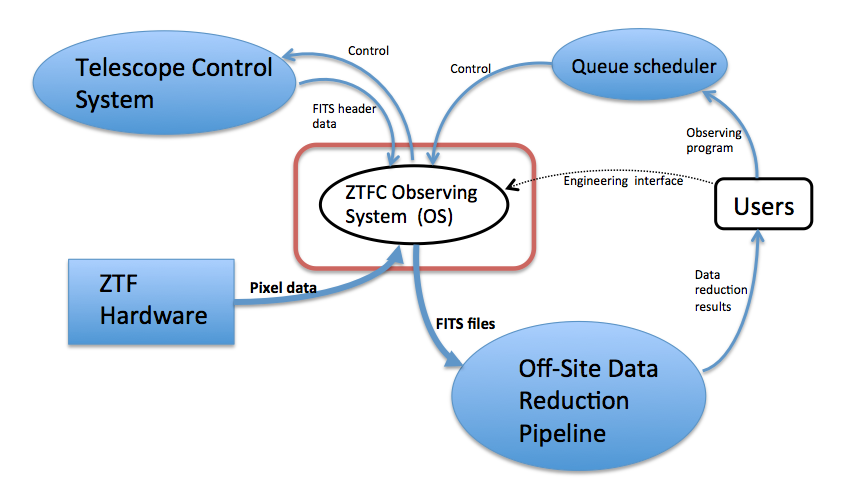


Figure 1 ZTF software context diagram.

The heart of the Zwicky Transit Facility camera is the CCD detector array – 16 6K x 6K CCDs in a 4 x 4 array:



Figure 2 Detector Layout in the ZTFC focal plane.

The three smaller focus CCDs and one guider CCD are also shown. Because of the large data files and high data-rate requirements, operating the CCDs in parallel is necessary, and having multiple computers interfaced to the CCD controllers reflects that necessity. Below is the proposed hardware layout for the ZTF software:

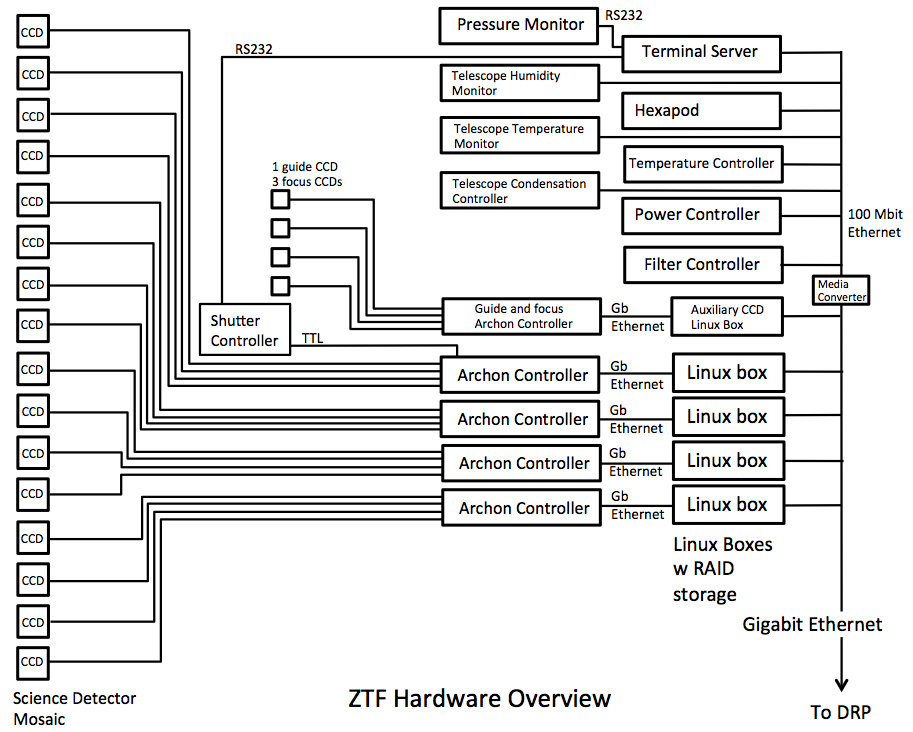


Figure 3 ZTF Control system hardware overview.

Use of the “Archon” controllers from Semiconductor Technology Associates is TBC. Controllers from Astronomical Research Cameras are also being considered. In either case, four CCDs per controller-computer combination can be handled.

The following table gives a list of hardware controllers required by the ZTF system.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Controller Name** | **Hardware to be Controlled** | **Number Needed** | **Embedded Software** | **High-Level Software** |
| CCD | STA Archon controller and E2V CCD231-C6 6K CCD | 5 | Vendor-Supplied | CCD Server |
| Dewar Temperature | Lake Shore 336 | 1 | Vendor-Supplied | Temperature Control Server |
| Dewar  Vacuum | Granville Phillips MicroIonPlus | 1 | Vendor-Supplied | Pressure Monitor |
| Power | TBD | 1 or 2? | Vendor-Supplied | Power Server |
| Filter Motor | TBD | 1 | Custom | Filter Server |
| Physik Instrumente Hexapod Controller | PI Hexapod | 1 | Vendor-supplied | Focus Server |
| Telescope Temperature Monitor | TBD | 1 | Vendor-Supplied | Telescope Temperature Monitor |
| Telescope Humidity Monitor | TBD | 1 | Vendor-Supplied | Telescope Humidity Monitor |
| Dry Air  Flow Controller | TBC, TBD | 1 | Vendor-Supplied | Window Condensation Mitigation Server |

# Performance Overview

Below is the same observing-cadence timing diagram that appears in the software requirements document. The observing rate drives a good bit of the software and hardware design:

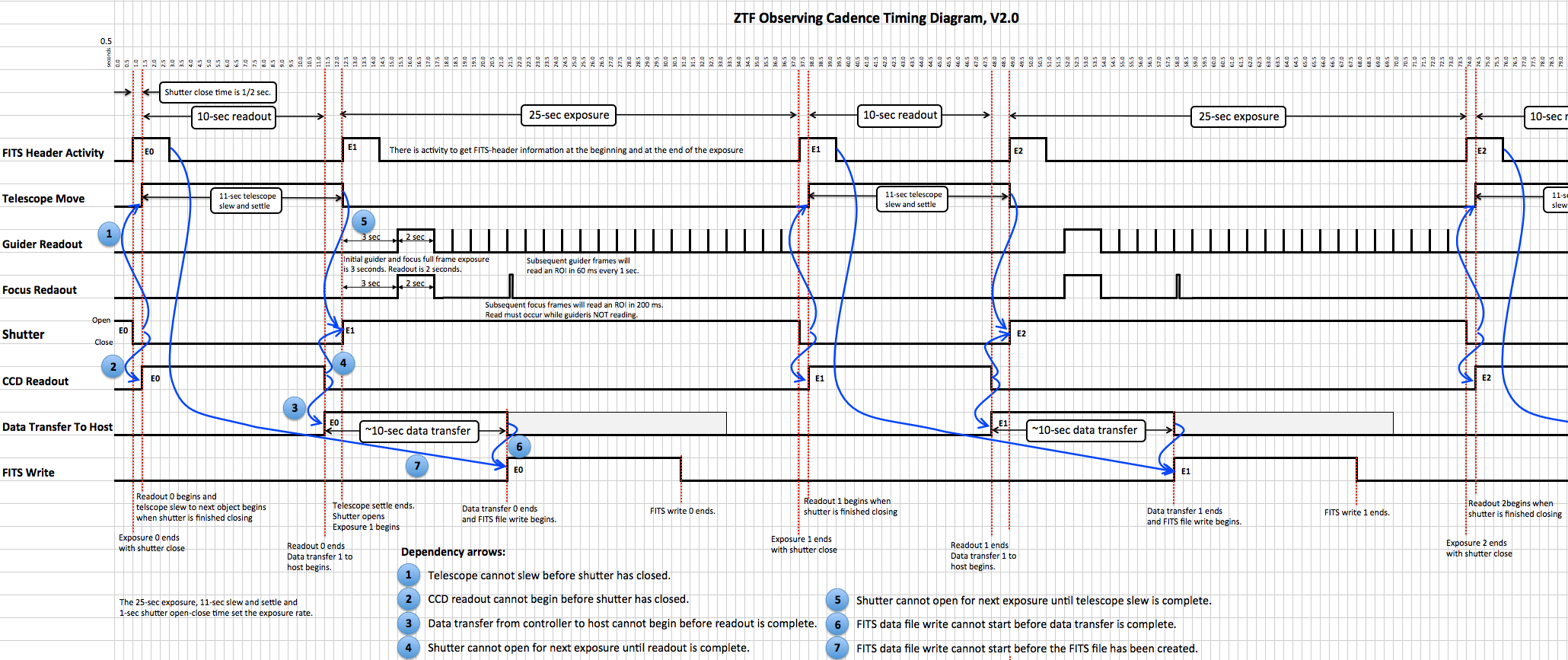


Figure 4 Observing Cadence Diagram, Version 2.0.

A larger, easier-to-read version of this diagram is available as an Excel spreadsheet, ZTF\_timing\_v2.xlsx, on the “ZTFC Instrument Control Software” twiki web page.

Note that this diagram represents an “operational assumption,” not a requirement, per se. It is a goal of the software to do everything it can to realize this assumption, which carries with it a number of implications:

* The software will not contain any arbitrary delays in the in-line code.
* The readout of four CCDs will occur in 9 to 10 seconds. This will likely be a matter of physics more than software.
* The writing of the data to four FITS files will not take more than 10 seconds. At a conservative estimate of 72 MB/s for writing to disk and reading out four CCDs per controller per host, this should be obtainable.
* The transfer of the pixel data from the CCD controller to the host computer and subsequent FITS file write will overlap the beginning of the next exposure.

# Software Overview

The ZTF software consists of 11 top-level modules, nine of which interface to hardware or external systems and one, the Observing System (OS), which interfaces to the Queue Scheduler (QS) and is the central organizing module for the entire system. The figure below shows a top-level data-flow diagram for the system.

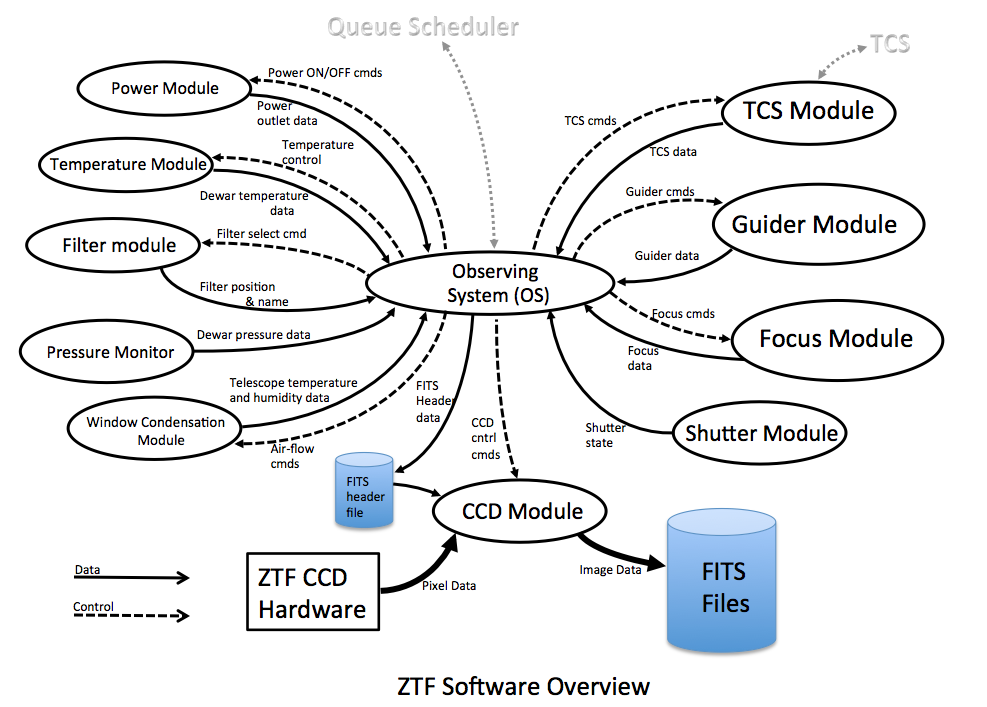


Figure 5 ZTF Software overview. Top-level data-flow diagram.

Each module acts as a server, able to accept commands and requests via network TCP connections. Each configures itself from configuration files. Each is multithreaded, with one thread that monitors the hardware and one that deals with commands and requests. Each module includes a configurable data and transaction logger.

# Observing System

The ZTF Observing System (OS) is the top level module of the system and serves as a single-point router for instructions from the Queue Scheduler as well as commands and requests from a user. It is also responsible for monitoring the health of the other subsystems and for monitoring disk space for FITS files and for the operating system.

* The interface details between the OS and the QS are TBD.
* The interface details between the OS and other servers are TBD.
* Both interfaces will be based on network TCP/IP protocols.

# CCD Control and Readout

## Overview

The CCD detector mosaic is the heart of the system. As mentioned above, because of the CCD-controller constraints, the CCD control and readout system must be spread across four separate computers. We anticipate having four CCD server processes, one running on each computer, and one CCD master server that interfaces to the OS directly and coordinates exposures with the other four. [*Is this really necessary or could one of the four CCD servers be the master coordinator?]* Below is a data-flow diagram of the CCD software:

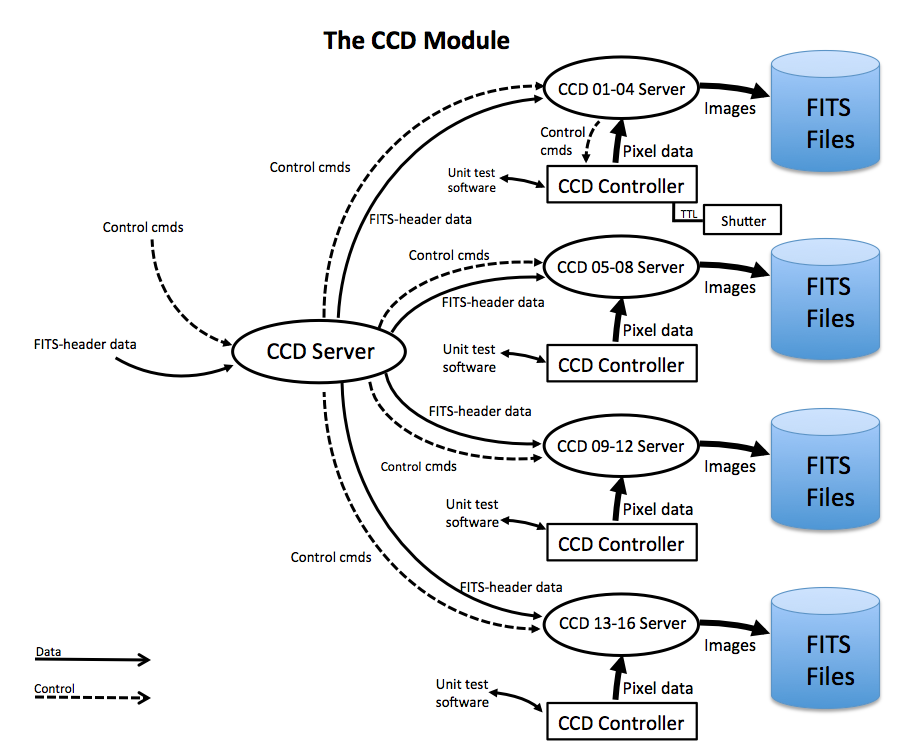


Figure 6 CCD Software Data Flow Diagram.

Each CCD sub-server will be multithreaded.

## Exposure Walkthrough.

A typical ZTF exposure should go something approximating the following (and note that a world of detail is missing from this list):

1. OS sets up the exposure first (configuration, exposure time, etc).
2. OS sends the Take-an-exposure-now command to the main CCD server.
3. CCD Server sends the Take-an-exposure-now command to the four CCD sub servers.
4. The four CCD sub-servers send commands to the CCD controllers to prepare for an exposure (stop idle clocking, wipe to the dump gate, etc.)
5. CCD controller #1 is special in that it has a TTL line to the shutter controller.
6. CCD controller #1 sends the TTL signal to open the shutter.
7. The time of the TTL signal is latched in the time card and read by the shutter monitor and stored for inclusion in the FITS header.
8. Each CCD sub-server now creates a FITS-header thread and a CCD-readout thread and the FITS-header thread creates the FITS files and begins collecting the data for the headers.
9. CCD controller #1 counts down the exposure time.
10. CCD controller #1 sends shutter close signal when exposure time counts down to 0.
11. Time of the TTL shutter-close signal is latched in the time card and read by the shutter monitor.
12. The FITS-header threads collect end-of-exposure header data.
13. Everyone waits for the shutter to close.
14. After the shutter is closed the OS tells the TCS to move to the next field.
15. CCD controller #1 signals through hardware to the other controllers that the exposure is over.
16. The four CCD controllers clock out the CCDs, synchronously, and in parallel, read the CCD data into the CCD controller memory, deinterlacing it as it is read. This should take about ten seconds.
17. After the readout is complete, the transfer of data from the CCD controllers to the host computer begins through the readout-threads and when all four of the CCD controllers have completed the readout, the status of the CCD system is sent to the OS telling it the CCD system is READY for the next exposure to begin.
18. When the telescope has settled on the next field and the READY signal is received from the CCD subsystem, the OS can begin another exposure.
19. Meanwhile the data transfer from controllers to computers completes after about 10 seconds.
20. The FITS-header threads close the FITS files and die after the writing of the FITS headers is complete.
21. The readout threads check to see that the FITS-header threads are done writing the header data to the FITS files and then they write the pixel data to the FITS files, close the files and die.
22. The OS is now signaled that the FITS file writing is complete and ready for the Data System to copy the files to HPWREN.

Many, many details need to be worked out and filled in in the above procedure, but, in concept, this is the way it should work.

## FITS File Creation

The FITS files are created by the FITS-header thread in each CCD sub-server, this thread is also responsible for gathering the information that goes into the header of each file. The process should proceed as follows: All of the sources of FITS-header information will log that data to “FITS-header stub” files that are located on disk partition that is cross-mounted across all five ZTF computers. See diagram below:

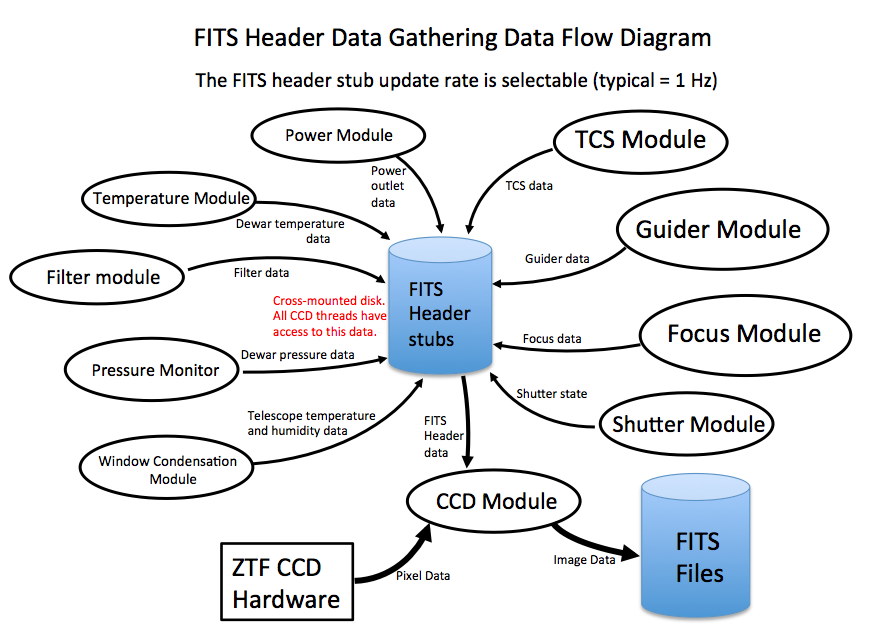


Figure 7 ZTF subsystems log FITS-header data regularly.

At the start of an exposure, a FITS-header thread and a CCD-readout thread are created specifically for that exposure. Both of these threads dies when they have finished doing their jobs. This one-to-one matching of threads to exposures should insure that the data for that exposure ends up exclusively in that exposure’s FITS files

The FITS-header thread gathers information at the start of an exposure and at the end of an exposure. It then creates the FITS files (remember, there will be four CCDs read out out per exposure per computer), writes the header, closes the file and dies. The CCD-readout thread will write the data to the FITS files, signal the FITS files are ready and then die. Mutual-exclusion semaphores will insure that the two threads do not stomp on each other. (Alternatively the FITS-header thread could pass all of the header information to the readout thread and the readout thread could do all of the writing.)

Since readout and FITS-header threads are specific to each exposure, there could be more than one set of these threads alive at a time, therefore data-transfer from the CCD controller and FITS file writing could theoretically overlap the beginning of the next exposure.

After all CCD-servers signal the OS that FITS file writing is complete the DS will whisk the files away to the ZTF DRP.

# Shutter Control

There are two ways to open and close the shutter.

1. Via serial interface to the controller processor.
2. Via TTL signal to the controller.

Below is a data-flow diagram for the shutter control system, showing some hardware elements also.

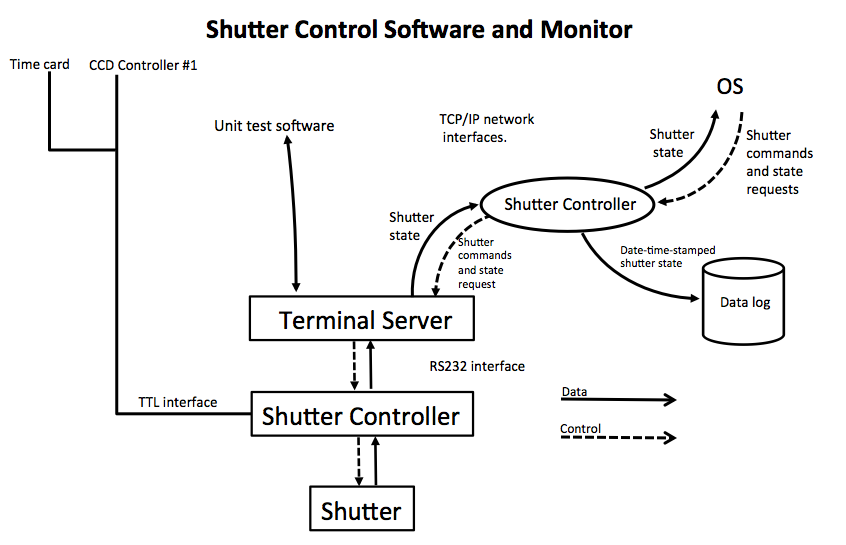


Figure 8 Shutter control and monitoring system, data-flow diagram.

# Power Control

The power-control software handles switching of AC outlet power to the connected ZTF subsystems. The list of subsystems to be controlled is TBD. Below is a simple data-flow diagram for the software:

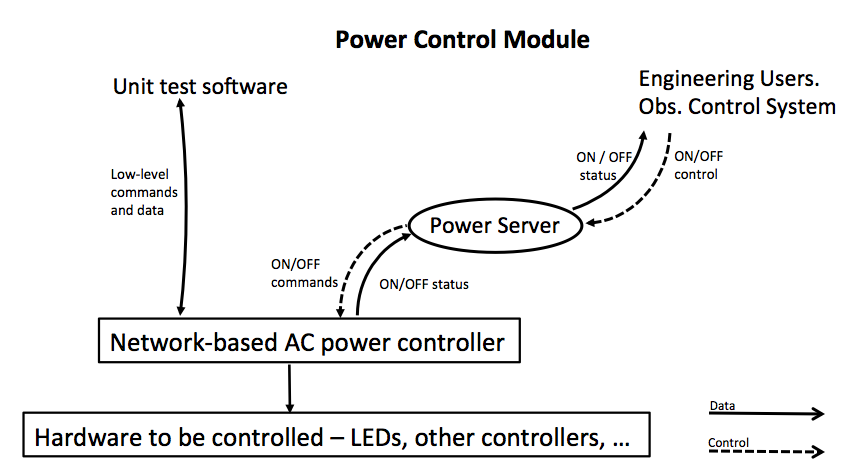


Figure 9 Power control and monitoring data-flow diagram.

# Dewar Temperature Control and Monitoring

The temperature control and monitoring software will interface to the temperature controller used to control the temperature of the dewar. Data-flow diagram below:

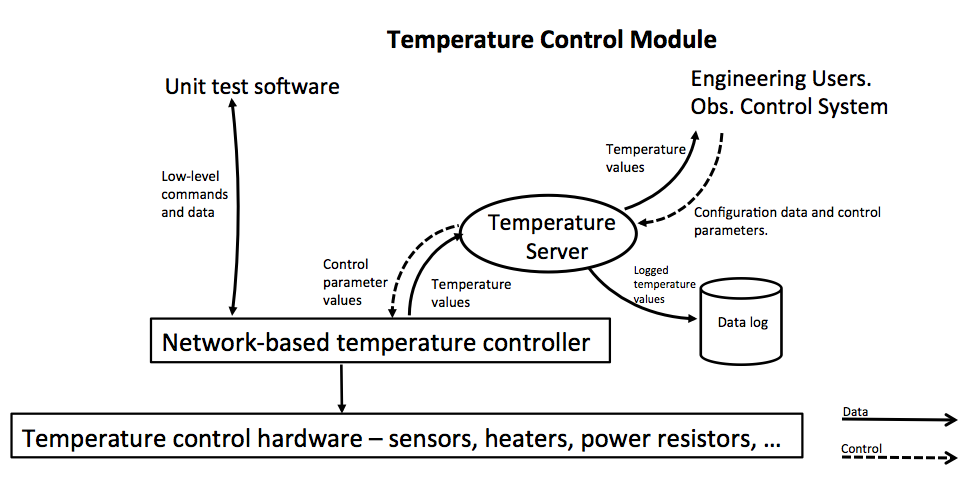


Figure 10 Dewar temperature control and monitoring system data-flow diagram.

Cool-down and warm-up scripts, which set up the target temperature, the slew rate and other control parameters, based largely on those developed for TRICK and MOSFIRE, will be developed during the dewar integration phase, when the thermal characteristics of the dewar are determined.

# Dewar Pressure Monitor

The pressure-monitoring software interfaces to the pressure gauge and its controller and reads its values. Below is the data-flow diagram:

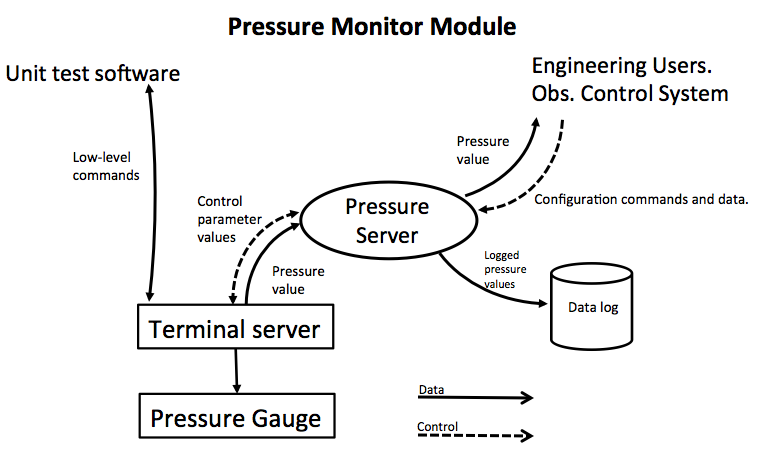


Figure 11 Pressure monitor data-flow diagram.

# Filter

The filter-control software will deploy and remove filters from the ZTF optical path and monitor which filter is deployed. Below is a data-flow diagram for this subsystem:

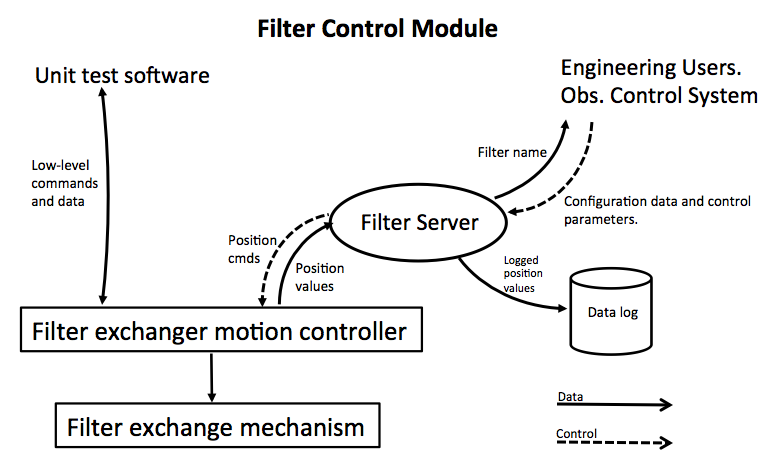


Figure 12 Filter exchanger software data-flow diagram.

# Window Condensation

The dewar window condensation software is responsible for keeping the window from becoming fogged from humidity in the telescope. Below is a data-flow diagram for the software:

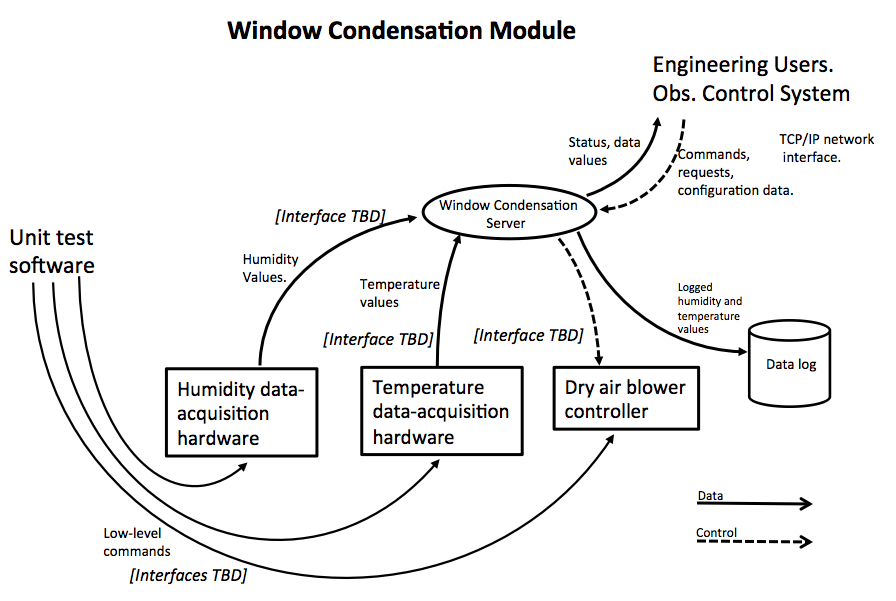


Figure 13 Window condensation software data-flow diagram.

It is possible that the telescope will simply be filled with dry air that is circulated constantly, independent of computer control. In that case, this software will only monitor the temperature and humidity of the inside of the telescope and send alerts if their values go out of bounds.

# TCS Interface

The TCS module is responsible for connecting and acting as a single conduit to the telescope control system. It is the primary conduit for sending new position commands to the telescope and dome. Below is the TCS software data-flow diagram:

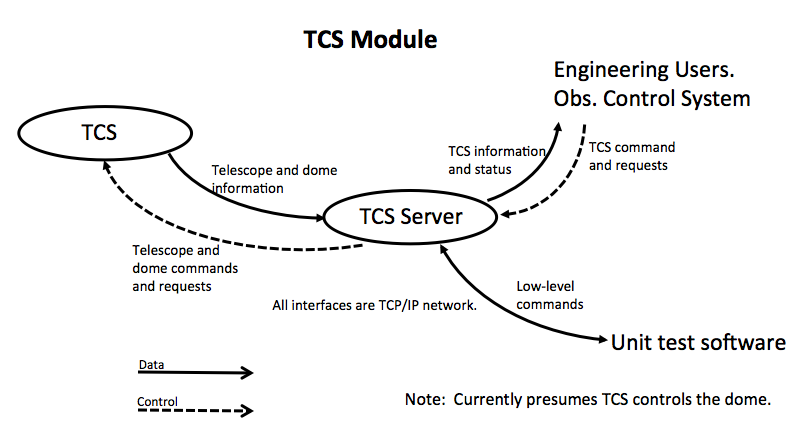
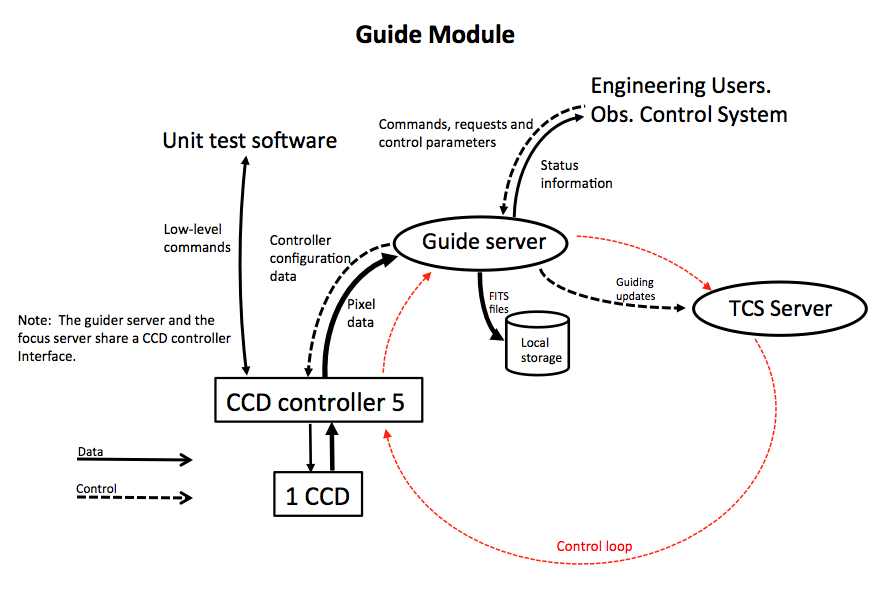


Figure 14 Telescope Control System software data-flow diagram.

# Guiding

The guiding software will be responsible for reading images from the guider CCD and sending commands to the telescope to correct any motion of the guide object in the images. Below is a data-flow diagram for this subsystem:



# Focus

The focus software is responsible for reading images from the three focus CCDs and determining corrections to send to the hexapod, which controls six-axes of orientation of the dewar and focal plane. Below is a data-flow diagram of this subsystem:

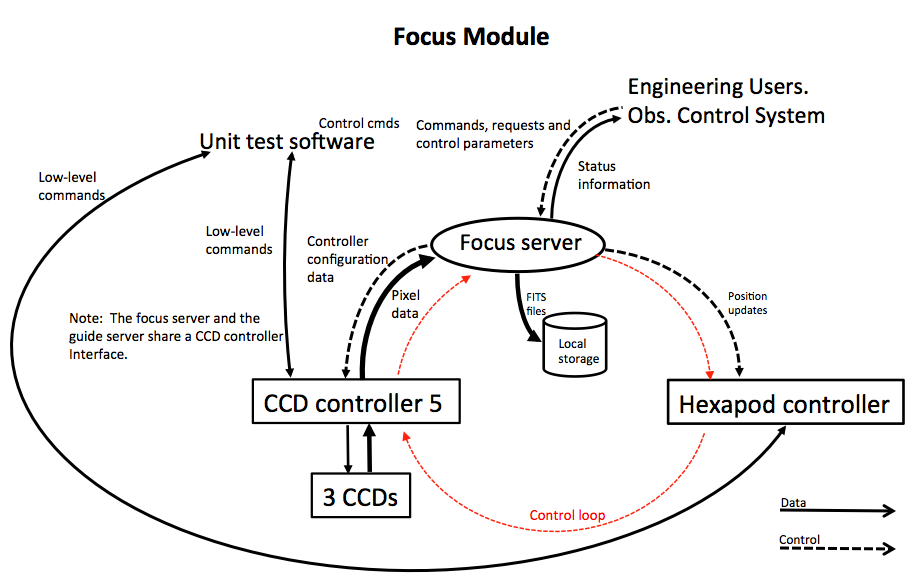


Figure 15 Focus-control software data-flow diagram.

# Frameworks

In his book, *Software Systems for Astronomy*, former WMKO software guru and mathematician, Al Conrad, states that for instrument-control software systems that are larger than 1000 lines of code, one should “use a framework,” a framework being an existing software system – tools, libraries, applications – which supplies so-called “common services,” logging, inter-process communications, network communications, error handling and watch-dog-timer functionality, for example.

For the ZTF software, we believe a framework should be used and could save the project much development effort and expense. It would also contribute to the reliability of the final product and allow programmers to concentrate on solving specific ZTF problems rather than spending time solving general problems that have already been solved.

There are a number of choices for instrument-control frameworks:

* KTL/RPC – The Keck Tasking Library and associated RPC server applications, currently used by a number of WMKO instruments, among them, MOSFIRE, OSIRIS, NIRC2, and NIRSPEC.
* EPICS – Experimental Physics and Industrial Control System, currently used by the Keck I and II drive and control systems, originally developed for instrument control in particle accelerators.
* MUSIC – Multiple User System for Instrument Control. Originally developed at UCSC/Lick Observatory this system is currently used also for a number of Keck instruments – LRIS, HIRES, ESI, DEIMOS.
* ARCVIEW—A LabVIEW-based control and data acquisition system currently used at Palomar on a number of instruments – DBSP, WIRC, TSPEC.
* Robo-AO – The Robotics Adaptive Optics system, currently in use at the P60 at Palomar for UV laser guide star AO programs and being developed for use on Mauna Kea for UH telescopes and for use at Kitt Peak.

Barring some as-yet-undiscovered disaster, ZTF will use Robo-AO as the framework for software operations. Reasons for choosing Robo-AO:

* Compatibility with existing Palomar system.
* Not wedded to a specific (and out of date) commercial product and obsolete hardware.
* Demonstrated performance.
* In-house expertise.

The ZTF software developers are just now getting into discussions with the Robo-AO software developers to facilitate this collaboration. (Reed and I have been talking…)

# Development Tools

The ZTF software team will use the GNU suite of tools (gcc, gmake, ld, ar, etc) for compiling and linking the software. Git will be used for source code control. We will also use any tools associated with the Robo-AO framework.

# Testing

The following points are guidelines for a conceptual test plan.

* All code will compile with no warnings or errors with the –Wall option.
* Unit tests will be built into the software development. All functions and methods will be written with an associated wrapper, used to test the function.
* Low-level test programs will be used to test hardware interfaces and to inform the development of the operational system.
* Firmware-level tests with delivered hardware will be tested with vendor-supplied software.
* All hardware interfaces will be tested for long-term reliability, accuracy and efficiency, e.g. repeated writes and reads show no dropped characters, etc.
* All defined commands to the individual software components will be tested for accuracy.
* Error recovery will be tested in the following situations (not exhaustive) with the following questions in mind:
  + Does the software detect the error?
  + Does the software recover from the error? If not, why not?
  + Does the software alert the responsible parties?
* Power glitches.
  + Cycle power to the entire system, observe and tune recovery.
  + Cycle power to individual subsystems, observ and tune recovery.
* Simulate connector problems, cable problems and cold-solder joint problems.
  + Pull cables during operations (ONLY SELECTED CABLES, where the hardware will not be harmed!)
* Process death or hang.
  + Use “kill -9” to terminate selected processes, observe and tune results.
  + Use infinite loops to simulate race conditions, observe and tune results.
* Disk space limitiations.
  + Simulate. Does the software report correctly.
* Memory allocation failures.
  + Simulate. Does the software report correctly.

Additional tests (also not exhaustive):

* Make sure FITS-header keywords do not “jump exposures.”

As this is being written, we are in the process of testing one of the STA Archon CCD controllers for long-term reliability by using the vendor-supplied software and some custom utility programs, by reading out the frame-buffer contents repeatedly, using loop-back electronics to feed the ADC board and subtracting successive frames to look for glitches.

All test programs, scripts, procedures, logs, results and other test items will be part of the development tree and will be (if desired) delivered with the final system.