**PRIME Software Control System**

**Specification and Design Document**

by Eric Lyness [eric.i.lyness@nasa.gov](mailto:eric.i.lyness@nasa.gov)

April 2020

This document format is based on the Software Design template originally created by Bradford D. Appleton, Copyright © 1994-1997 by Bradford D. Appleton

**Table of Contents**

[1. Introduction 3](#_Toc37171232)

[2. System Overview 3](#_Toc37171233)

[2.1. System Overview: Hardware 4](#_Toc37171234)

[2.2. System Overview: Software 5](#_Toc37171235)

[3. Design Considerations 5](#_Toc37171236)

[3.1. Assumptions and Dependencies 6](#_Toc37171237)

[3.2. General Constraints 6](#_Toc37171238)

[3.3. Development Methods 6](#_Toc37171239)

[4. External Interface Specifications 6](#_Toc37171240)

[4.1. Detectors 7](#_Toc37171241)

[4.1.1. H2RG Detector Interface Specification 7](#_Toc37171242)

[4.2. PRIME Thermal Control System 8](#_Toc37171243)

[4.3. RIMAS Thermal Control System 8](#_Toc37171244)

[4.4. Cryo Cooler 8](#_Toc37171245)

[4.5. Pressure Monitor 8](#_Toc37171246)

[4.6. Telescope Control 8](#_Toc37171247)

[4.7. Filter Wheel Control 8](#_Toc37171248)

[4.8. RIMAS Liner Focus Camera 8](#_Toc37171249)

[4.9. RIMAS Slit Camera Viewer 8](#_Toc37171250)

[5. Master Controller Specifications 9](#_Toc37171251)

[5.1. Channel Definitions 9](#_Toc37171252)

[5.2. Local Storage and Archive 9](#_Toc37171253)

[5.3. Alarms 9](#_Toc37171254)

[5.4. Data Viewing User Interface 10](#_Toc37171255)

[6. Architectural Strategies 10](#_Toc37171256)

[7. System Architecture 10](#_Toc37171257)

[7.1. Subsystem Architecture 10](#_Toc37171258)

[8. Detailed System Design 10](#_Toc37171259)

[9. Glossary 10](#_Toc37171260)

[10. Bibliography 10](#_Toc37171261)

# Introduction

This document describes the software that will provide centralized control and data acquisition and storage for the near-infrared camera that will be used on the PRIME telescope at the South African Astronomical Observatory. The PRIME project is intended to measure microlensing events in the direction of the Galactic bulge.

The control software ties together subsystems that encapsulate the complexities of detector readout, temperature control and telescope control. It also communicates with simpler subsystems for pressure and temperature measurements and filter wheel control.

Section 4 will detail each sub system, including:

* Detector via the MACIE interface
* Custom thermal controller
* Pressure and Temperature gauges
* Telescope control system

The configures the detectors, telescope and other subsystems then triggers images. The images are then stored in FITS files with all the necessary metadata in the FITS file headers. All software control features can be operated manually using the software’s user interface. Additionally, the software can execute scripts to automate image acquisition.

This document contains two main sections: The Specifications section and the Software Design section. The specification section defines the hardware, hardware interfaces and software functionality. The design section describes the software implementation of the system.

The purpose of this document is twofold: First, it communicates the operation of the system to the larger team of operators, managers, technicians and designers. Second, it records the as-built implementation to guide future engineers in maintaining and updating the system.

This document is not intended to be a user manual.

# System Overview

The software for the PRIME telescope camera control is based on the software used by the Rapid Infrared Imager Spectrometer (RIMAS). The software will remain compatible with RIMAS or PRIME.

Figure 1, below, shows a schematic of the PRIME software.



Figure System Layout

## System Overview: Hardware

As shown in Figure 1, the hardware consists of:

* PRIME Detectors: Communicates with separate detector Python-based software application, running on the same computer. The software “asdetector.py” provides a TCP socket interface.
* RIMAS Detectors: The H2RG application runs on a separate computer. Interface is also through a command line interface over SSH connection. Files are copied to from the detector computer to the local computer using SCP.
* PRIME Thermal Control: (aka Housekeeping box, or Cryo Cooler) A custom-built thermal control receives commands for setpoints and other required control information and transmits status and temperatures. Interface is through a TCP socket interface.
* RIMAS Thermal Control: LakeShore commercial instruments, models 325 and 218, are used for temperature monitoring and control. The software communicates with the LakeShores via TBD link.
* PRIME and RIMAS: Cryo Cooler, TBD
* PRIME Pressure Monitor: A Lesker commercial pressure gauge. Model: TBD, Interface TBD
* RIMAS: Pressure Monitor: Pfeiffer Vacuum pump. Communications is via a serial link (TBD)
* PRIME and RIMAS: Telescope Control: Telescope control is through separate software running on the same computer. Interface is though the command line (TBD)
* PRIME and RIMAS: Filter Wheel control. Interface is through a serial link to Arduino controller (TBD)
* PRIME: Linear Focus motor: This interface is the same as the filer wheel interface, using an Arduino for motor control. Interface details TBD.
* RIMAS: Slit View camera: This camera is controlled by a National Instrument Compact RIO embedded controller running FPGA-based software to read out the camera. Interface is TBD.

## System Overview: Software

The LabVIEW software will act as a single point of control for all operations. It will provide visualization of all aspects of the system and a scripting engine to allow automated control. It generates standard FITS files for downstream processing. The software will be multi-threaded and modular such that each instrument or device the software receives data from and/or controls will run independently from other modules. A master software module will launch each module. A communications protocol will be used to feed data back to the master software module and receive commands from the master software module. This design allows the software to continue operating even if one or more devices is offline. It also allows for incremental development as new modules are added to the system.

The master control software will be initiated by the user. At start up, the software will dynamically launch subsystem control software based on the saved configuration of the system. RIMAS and PRIME system will differ based only on the configuration file.

Communication between each module and the master control software is via LabVIEW queues. Each subsystem has a unique queue for commanding. Each subsystem sends data back to the master controller via a single data queue. The master controller is responsible for logging all data.

The master controller is also responsible for alarms. In the event a pressure or temperature is out of safe ranges, the master controller alerts the user. Automated safing procedure is TBD.

If a subsystem requires a user interface, it will automatically display the interface when it is launched.

The master controller provides an interface to see the status of all subsystem modules. Users can enable or disable subsystems using the master controller software.

The master controller can execute scripts to perform automated image acquisition. As much as possible, all functions available on the user interface are also available in scripts.

# Design Considerations

Modularity is the key to the design. The detector subsystem and thermal subsystems are custom-built for PRIME and can be very complex. Therefore, the software to communicate with them should be easy to debug. The subsystem module for each interface should run independently of the master control software such that it can be tested and debugged without invoking the master controller.

Each subsystem code should use a standard interface to the master controller. This includes a standard initialization command, status commands, telemetry output and shutdown. The software will use variants where the parameters to a command will vary depending on the subsystem type. A template should be created from which all subsystem code is developed.

Each subsystem should be re-entrant such that multiple instance of a subsystem’s control software can be launched independently without interfering with each other.

Understanding and documenting the physical and logical interface to each subsystem is essential. Some of the interfaces are custom and require some design and cooperation with developers of the hardware for those subsystems.

The software should be responsive to telemetry received from the subsystem. If a temperature or pressure or other important telemetry is out of expected range, the user should be notified as soon as possible.

The FITS file output should be organized in a way such that files are easily located. Large numbers of files (more than 1000) in a single folder should be avoided.

## Assumptions and Dependencies

It is assumed that the archive of the data outside of the system will be performed externally. No Internet connection is required to operate the software.

## General Constraints

The software should be capable of running continuously. There should be no degradation in performance over time. Housekeeping data, such as temperatures and pressure should be recorded locally. A new telemetry file should be generated each calendar day to avoid excessive size.

A text log should be recorded each time a command is sent to a subsystem. It should also include any alarms or special events, such as errors, received from subsystems.

The user interface should be similar the user interface for the Large Monolithic Imager located at Lowell Observatory.

## Development Methods

Once developed, the software will run as an executable and not require a LabVIEW development license.

# External Interface Specifications

This section describes the detailed operation of the software focusing on the interfaces to the subsystems.

Table External Interface List

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Subsystem | Project | Device | Physical Interface | Logical Interface | Commanding | Telemetry |
| Detectors | R | H2RG Software | None | API provided by H2RG Software | Detector Configuration | Status and Image Files |
| Detectors | P | Python software | None | API provided by Python sfotware | Detector Configuration | Status and Image Files |
| Thermal | P | PRIME Themal Control | TBD | TBD | Temperature setpoints? Other config? | Current temperatures, status |
| Thermal | R | LakeShore | USB-SERIAL | Lake Shore Commands | Temperature setpoints | Current Temperatures, status |
| Cryo cooler | P/R | ? | USB-SERIAL | ? | Temperature setpoints | Current temperatures, status |
| Pressure Monitor | P | Lesker | USB-SERIAL | Lesker commands | ? | Current presure(s) |
| Pressure Monitor | R | Pfeiffer | USB-SERIAL | Pfeiffer commands | ? | Current presure(s) |
| Telescope Control | P/R | Telescope Software | None | API from telescope software | Telescope Pointing | Telescope position and status |
| Filter Wheel | P/R | Arduino | USB-SERIAL | Arduino software interface | Position | Position, Status |
| Linear Focus | R | Arduino | USB-SERIAL | Arduino software interface | Position | Position, Status |
| Slit View Camera | R | cRIO | Ethernet | cRIO command interface | Image acq. Settings | Images, status |

## Detectors

The Detector object has five methods. PRIME and RIMAS implement each method differently. These is also a SIMULATION detector that provides pre-defined FITS files in order to test the software.

1. Init – Connect to hardware
2. Configure – Set image acquisition parameters
3. Get Images – Trigger the detectors to acquire images and return images
4. Make FITS files – Update the FITS file headers
5. Close – Release the hardware

PRIME has four detectors that are read out using the ACADIA-MACIE detector assembly. Communication with this hardware is through a Python application. The application, running on the same computer as the controller software, configures the detector readout configuration and stores the detector output as files. The control software interfaces with the software via a command line interface and TCP Socket connection. Parameters are provided to the Python software and detector acquisition is initiated via the TCP socket using the API. The Python software records image files locally as FITS files. The PRIME software updates the headers of the FITS files with appropriate settings.

For the RIMAS system, the H2RG software runs on a separate computer. Control is over Ethernet using a secure shell (SSH) connection. The commands are the same but the SSH connection must be established first on RIMAS. Also, files are stored locally on the machine running H2RG so on RIMAS the files must be copied to the local machine via secure FTP (SFTP) or other means.

### H2RG Detector Interface Specification

This section contains the specific commands and responses to and from the H2RG software. It should contain everything necessary to write the software to communicate with the detectors.

This should include SSH commands for RIMAS interface, IP address, and process for connecting, testing, etc.

## PRIME Thermal Control System (Housekeeping)

PRIME and RIMAS use entirely different thermal control systems so they are included in separate sections of this document.

PRIME uses custom thermal control hardware using a custom interface. Communication method is via TCP. Software written in Java will run on the Master Control computer. This Java software performs the actual communication with the Thermal Control System. The Master control software communicates with the Java software.

Temperature readings from the thermal control system should be synchronized with image acquisition (TBD) such that the temperatures can be correlated with each frame.



Figure 2 TCP connections to Thermal Control System

### Commands

Control Software can command the thermal control system to set any of n heaters setpoint, PID values, enable/disable, and to read telemetry.

Examples commands:

SetHeaterPID 1, 0.3, 0.1, 0.01

This command sets heater 1 PID values to P=0.3, I=0.1, D=0.01

HeaterSetpoint 1, 100

This sets heater 1 to 100K.

HeaterRead

Requests the thermal control system return all temperature values.

### Telemetry

The thermal control system returns data on command from Control Software. The temperature returned should be the temperature within one second of the request.

Values returned from thermal control system consist of internal and external temperatures, in degrees K.

Format of responses from thermal control system TBD.

### Connecting to Thermal Control System

This section defines the protocol and connection parameters (i.e. Ethernet or USB, command formats, etc.) TBD.

## RIMAS Thermal Control System (Lakeshore)

RIMAS use commercial Lake Shore temperature controllers. The interface is documented in TBD document.

## Cryo Cooler

Serial connection to Cryotel AVC commercial controller. This interface protocol is defined and available online. A sample program has been written to communicate with this device.

The instrument is polled at a regular rate for telemetry value.

http://loke.as.arizona.edu/~ckulesa/binaries/STO/Sunpower%20CT%20-%20AVC%20Manual%20Rev%208.pdf

### Commands

Control Software needs the ability to set the setpoint and PID values and set the control mode. Another command requests the status.

### Telemetry

The status command returns the current temperatures, mode, etc.

## PRIME Pressure Monitor (Lesker)

Lesker 354 Series Vacuum Gauge. This is a commercial pressure gauge. It can return up to three pressures. Communications is over a RS-485 connection.

<https://www.lesker.com/newweb/gauges/pdf/manuals/354usermanual.pdf>

### Commands

The Lesker 354 received commands to configure and enable the gauges.

### Telemetry

The Lesker 354 returns up to 3 pressure values. Units return in Torr (TBD).

## RIMAS Pressure Monitor (Pfeiffer)

Already implemented. TBD. Read two pressures. The Pfeiffer requires some manual setup.

## Telescope Control

External executable custom software running on the Control Software computer communicates with the telescope. This software sends and receives commands to the telescope via XML messages. This tells the telescope where to point. Document on this interface is TBD.

Interface is through the commandline. The Control Software generate XML commands and pass to the Telescope executable via the commandline.

The Control Software has to wait for moves to complete. It should wait for 3 things: Dome status, AOS (active optics system), and the move complete.

## Filter Wheel Control

The filter wheel(s) – Four for RIMAS, two for PRIME – consists of motor controlled by Arduino computers. The control software communicates with the Arduino over a USB-SERIAL interface.

This interface is well known.

## RIMAS Liner Focus Camera

Like the Filter Wheel control, the Linear Camera control uses the same kind of Adruino devices controlling the same kind of motors. The difference is in the application that controls the motor.

## RIMAS Slit Camera Viewer

RIMAS also include a slit camera that is readout using a National Instrument Compact RIO system running FPGA code. The interface is via Ethernet connection using a protocol defined by the software running on the cRIO.

This interface is known.

# Master Controller Specifications

## User Interface

The user interface should be based on the Lowell Monolithic Imager control software from Lowell Observatory. It does not need to duplicate the LMI software but should have a similar look and feel. See “The Large Monolithic Imager User Manual” by Philip Massey. A screen shot of this software is included below:

A screenshot of a social media post

Description automatically generated with medium confidence

Figure 3 Lowell Monolithic Imager User Interface

## Channel Definitions

Each temperature, pressure or other physical signal is referred to as a channel regardless of which physical subsystem it originates from.

Each channel has many attributes that can be explicitly defined by operators via the user interface.

Table Channel Attributes

|  |  |
| --- | --- |
| **Attribute** | **Purpose** |
| Name | User selected name for the channel to be used in plots and reports |
| Limits | The yellow and red alarm limits for both high a low. Limits can also be disabled on channel. |
| Scaling | Parameters for scaling an input signal into engineering units. Scaling can be linear or polynomial. Default is no scaling. |
| Units | The text that will be displayed on plots and reports of this channel. |
|  |  |
|  |  |

## Local Storage and Archive

Image data will be stored in FITS files with headers contain the information defined by the configuration commanded by the detector subsystem. FITS files will be stored locally in file structure that is TBD.

In addition, a message log of text messages will be stored locally with each message time stamped.

Other telemetry will be recorded locally in LabVIEW native TDMS format. A new TDMS file will be created each calendar day, based on local time.

## Alarms

Every channel, physical or virtual, has four limits:

Table Alarm Levels

|  |  |
| --- | --- |
| Low-Red | If channel value falls below this, set red alarm state |
| Low-Yellow | If channel value falls below this, set yellow alarm state |
| High-Yellow | If channel value exceeds this, set yellow alarm state |
| High-Red | If channel value exceeds this, set the red alarm state |

Each new data point for each channel is evaluated against the limits defined for the channel. If a channel changes to alarm state, that alarm is logged and indicated on the screen.

## Data Viewing User Interface

For data that is stored in TDMS files, a simple viewer will be provided that allows viewing of data by time for any given day.

## Scripting

The system can be operated via the user interface or via scripted commands. The script allows access to all subsystem commands. In addition it provides the following system commands:

* Wait N seconds
* No-op
* Ask (use dialog)

Scripts are stored in text format so they can be edited, saved, etc. outside of the software. Scripts are implemented as a separate loop.

# System Architecture

## Architectural Strategies

The software should be written in LabVIEW as it is based on the RIMAS software written in LabVIEW. The RIMAS software is the starting point of the control software.

As much as possible, the software should be multi-threaded such that a communications error or other blocking event does not stop monitoring and recording of other data.

LabVIEW also provides a built-in database format, TDMS, that is ideal for logging and archiving the relatively slow speed housekeeping data.

## Subsystem Architecture

All subsystem commands are sent through LabVIEW queues.

# Detailed System Design

# Glossary

An ordered list of defined terms and concepts used throughout the document.

# Bibliography

A list of referenced and/or related publications.

Brad Appleton <brad@bradapp.net>

http://www.bradapp.net