

DECam SISPI

Functional Requirements Document (FRD)

Alarms	2
Database (DB)	8
Disk Cache (DC).....	10
Focal plane Control System (FCS).....	12
Focal plane Data (FD).....	15
Focus	17
Guide CCD Control System (GCS)	19
Graphical User Interface (GUI).....	22
Guide.....	23
Image Builder (IB).....	28
Instrument Control System (ICS).....	30
Observation Tactical Planner (ObsTac).....	32
Observation Control System (OCS).....	34
Quality Analysis (QA)	37
Sky Camera (SkyCam).....	39

DECam SISPI Component Alarms

Functional Requirements Document (FRD)

Document name: DECcam-instr-FCS-FRD-req-v0.3.doc

Authors:

K. Wyatt Merritt (wyatt@fnal.edu)

K. Honcheid (kh@mps.ohio-state.edu)

Date: 2005 09 5

Distribution: DES-DAQ group

Version History:

Version 0.1 Initial version

Version 0.3 kh suggestions

Purpose of this document

This document is the Functional Requirements Document for the Alarm System of the DECcam Survey Image System as defined in the Process Integration document.

Introduction

- The responsibilities of the Alarm System cover management, recording, and display of warnings and alarm conditions reported by any of the DECcam subsystem. The Alarm System is a set of software processes that include a centralized AlarmHandler to process and to evaluate alarm conditions, a library to be used by DECcam subsystems to interface to the Alarm Handler, as well as a graphical user interface and a database for display and record keeping. Every error and alarm detected by any of the DECcam subsystem will be processed and recorded by the Alarm System. However, given the latency inherent to a distributed alarm system dedicated hardware interlocks and devices will be used protect the camera hardware. These devices are not considered part of the Alarm System and will not be covered in this document.

Functionality

Software processes assigned to the DECcam subsystems continuously monitor the status of all hardware, software, observatory, and environmental conditions that are relevant to the DECcam Survey Image System. Alarms, errors and warnings – possibly including periodic status updates and auditory notification of important status changes as well – will be reported to the central AlarmHandler process. Subsystems create alarm messages using library functions provided as part of the Alarm System. An alarm message is a structured object that includes information such as the name of the component, date and time, a severity code, a string with the alarm message and the name or address of a call back function. From the alarm library messages are either retrieved by (pulled) or sent to (pushed) the AlarmHandler via an inter-process communication protocol (IPC). A third possibility is that subsystems store alarm messages in a database that is then queried by the AlarmHandler. The AlarmHandler records information about the alarm condition in the alarm database and updates the alarm GUI. As the development of the DECcam instrument control system continues we will discuss if the AlarmHandler should be able to take further action such as contacting OCS in case an alarm condition indicate that an exposure or observation sequence is compromised.

The alarm GUI presents the information in layered views to human observers, both as real time and time history displays. The term ‘layered views’ means that displays will be designed to present compact views of the overall status (compact compared to the available computer monitor real estate AND available human operator attention) that should provide visual cues whenever the human operator needs to react. The displays should also provide easy interfaces for the operator to access more detailed information, as well as aid in identifying and debugging problems.

Alarm System Requirements

This section describes the requirements for the Alarm System that are necessary to achieve the goals of the DES experiment.

General Requirements

Requirement 1.0.1: The Alarm System shall record, archive and display every alarm, warning, and status message reported by the DECam subsystems.

Justification: This is just the definition of what the Alarm System has to do. Requiring that every message has to be recorded will define the performance/throughput of the system.

Requirement 1.0.2: Each DECam component shall include its own monitoring. The Alarm System shall provide interface libraries that the DECam components will use to interact with the Alarm System.

Justification: This describes the distributed nature of the DECam control system and the Alarm System in particular. Each sub-component is expected to include some kind of monitoring process that reports alarms and warnings to the central Alarm System using standard system libraries (Standard in the sense that they are provided to every component by the Alarm System).

Requirement 1.0.3: The Alarm System shall not be responsible for the protection of the camera hardware. DECam components that can be damaged when operated incorrectly shall provide the own, independent protection mechanism.

Justification: The latency of the Alarm System will be too large to react quickly to certain alarm conditions that might damage the camera if not corrected in a short amount of time.

Requirement 1.0.4: The Alarm System shall be able to interact with the Observation Control System (OCS) if an alarm condition or combinations of alarm conditions indicate that an exposure or observation sequence is compromised.

Justification: (this needs to be discussed). The idea is that the alarm system can signal to OCS that an exposure should be aborted without having the operator intervene.

DECam Standards

In order to simplify development and maintenance the Alarm System will comply with all applicable DECam standards.

Requirement 1.1.1: The Alarm System shall be based on standard DECam control system components. In particular, the Alarm System shall use the same inter-process communication protocol, database management software, and standards for graphical user interfaces that will be chosen for the entire experiment.

Justification: Successful development and also maintenance of a distributed control system has to rely on standards.

Requirement 1.1.2: The Alarm System must comply with all DECam software, version control, and documentation standards.

Justification: Complex systems require standards that everybody – including the Alarm System – follows.

The Alarm Database

Archival storage of alarm message is provided by the Alarm database.

Requirement 1.2.1: The Alarm System shall provide a centralized repository for all subsystems (Alarm Database).

Justification: We want one (alarm) database for the entire system.

Requirement 1.2.2: Records of alarm, warning and status messages shall be accessible to the collaboration until all data analyses are completed.

Justification: Information about alarm conditions will be needed during image reconstruction and processing.

Requirement 1.2.3: The Alarm Database must support retrieval of archived information without affecting the performance of the Alarm System.

Justification: This requirement guarantees that we can dump the database information for example to ship it to a data processing site without affecting the performance of the rest of the system.

The Alarm GUI

The Alarm GUI or console will inform the operator about the status of the system. If necessary, the operator will acknowledge alarms using the Alarm GUI. The operator can also use the Alarm GUI to play back archived messages from the Alarm database.

Requirement 1.3.1: The Alarm System shall provide a centralized display for all subsystems (Alarm GUI).

Justification: We want one display that shows the status of the entire system.

Requirement 1.3.2: The Alarm System shall support viewing of alarm messages stored in the Alarm database.

Justification: This allows playback of archived information. It might also lead to a simple implementation of the Alarm GUI.

Requirement 1.3.3: The Alarm GUI shall provide support for both visual and audio notification of the operator.

Justification: For some alarm conditions it will be useful to have an audio alert as well. We could consider expanding this to include e-mail notification of experts.

The Alarm Handler

The Alarm Handler is the central element of the Alarm System. It monitors and/or receives alarms from any DECam subsystem and forwards alarm messages to the Alarm Database and the Alarm GUI.

Requirement 1.4.1: The Alarm System must provide a central Alarm Handler component that retrieves (pull) or receives (push) alarm messages from any DECam subsystem.

Justification: This is the central clearing house for alarm, warning and status messages. It can be decided later if the alarm system will be based on a push or pull architecture – or a combination of both.

Remote Accessibility

Requirement 1.5.1: The Alarm System shall provide remote access capability.

Justification: Remote access to alarm messages (as well as other status information) will be an important debugging and monitoring tool for the collaboration.

Appendix (left from original version)

Implementation

Details of the implementation are not yet determined. The Alarm System is a part of the DECam instrument control system and as such will use the inter-process communication protocol, database management software and standards for graphical user interfaces chosen for the entire experiment. Remote viewing of alarm and status messages will be supported.

Implications for other components

The Alarm System is connected to (almost) every component of the DECam instrument control system. Data in form of alarm messages is received (or retrieved) from other DECam subsystems or played back from the alarm database. Data is sent to the Alarm GUI and the alarm database. Control information is received from the operator through the GUI or possibly a startup/configuration script. Control information is sent to OCS should we decide to allow for the AlarmHandler process to directly interact with the ongoing observation.

The same information in tabular form:

Receives control from:

- GUI
- Startup/configuration script

Sends control to:

- None (Alarm system informational to human observer, by definition) unless we allow for a direct connection to OCS

Receives data from:

- Alarm (Input) Database
- Many subsystems (TBD)

Sends data to:

- GUI display
- Alarm Database

Things to do

There are many more things that need to be addressed. Here is a first list of related items that we should discuss (eventually)

- Acknowledgment of alarms. Should alarms be “one way” message only? Should an alarm state persist in a component until it’s acknowledged (at least for some alarms)?
- Should there be blocking alarms or should we support only unblocking?

- Alarm suppression – assuming an alarm condition persists for a while the monitor loop in a component’s control process will most likely submit multiple alarms. Who keeps track of whether an alarm is already active or not.
- We could include a web link in the alarm message that informs the operator where to find more information.
- Alarm Objects: what’s useful, what’s not?
- Since we have to allow remote viewing we will have to “teach” the GUIs how to get the current state (could be as simple as reading the latest entries of the alarm database) but how about the central process, the AlarmHandler, Do we envision a system that allows (software) components to be stopped and restarted or is that quickly getting to complicated and “unnecessary featureitis” for this project.

DECam SISPI component DB

Functional Requirements Document (FRD)

Document name: DECcam-instr-FCS-FRD-v0.1.doc

Authors:

K. Wyatt Merritt (wyatt@fnal.edu)

Version: 0.1

Date: 2005 08 2

Distribution: DES-DAQ group

Version History:

Version 0.1 Initial version

Introduction

Functionality

Provide organized local storage of control, conditions and status information from the Survey Instrument Systems, with efficient search and retrieval capability. Parameters are stored in a database if any of the following are true:

- a) The values need to be tracked vs time, either for online operational displays or for later correlation with image analysis. Examples: temperature of focal plane array.
- b) The values need to be communicated asynchronously to many other processes, but time performance is not critical. Examples: Guide star lists??
- c) The values need to be retrieved with flexible queries. Examples: Metadata regarding image collection conditions. May want to ask for integrated time in the past year with seeing better than some standard, or % completion of survey imaging goal with a particular filter.

Some information may be stored in the databases and also included in the FITS headers of the image data. Care should be taken to insure that information which can be obtained from two different sources is guaranteed to be the same: suggested procedure is that Subsystem A is the primary source of information (say, general sky conditions). SubA writes the data to a database. The Image Builder system gets the data from the database, NOT from SubA, and puts it in the header. So there is really only one primary recorded source, the database, and it is a clear error condition if the data in the header disagree.

Communications

Receives control from:

- GUI

Sends control to:

- None

Receives data from:

- Many subsystems (TBD)

Sends data to:

- Many subsystems (TBD)

Issues:

- What database(s) to use: ORACLE, PostGreS, MySQL, etc.
- How to organize schemas
- What technology to use for remote DB access: CORBA, etc. Three tiered or two tiered access model?

Implementation

Not determined.

DECam SISPI component DC

Functional Requirements Document (FRD)

Document name: DECcam-instr-DC-FRD-v0.2.doc

Authors:

Jon Thaler (jyt@uiuc.edu)

Joe Mohr (jmohr@uiuc.edu)

Jim Annis (annis@fnal.gov)

Version: 0.2

Date: 2005 09 22

Distribution: DES-DAQ group

Version History:

Version 0.1 First version

Version 0.2 JTA edits

Purpose of this Document

This document is the Functional Requirements Document for the Data Cache (DC) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

This document differentiates between the commands received by the DC and the instructions it issues.

This document refers to instrument components which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The DC is the repository for DES images. The Image Builder (IB) publishes data to it, and various other processes subscribe to the data. The DC reports its status and alarms. The primary purpose of the DC is to save the data taken during the night and subsequently make it available to the DM.

Functionality

The DC:

- Receives commands to initialize and update the subscriber list.
- Receives commands to initialize (*i.e.*, empty) the data buffers.
- Receives commands to clear sections (*e.g.*, certain images) of the data buffers.
- Receives commands to list the status and contents of the data buffers.
- Receives data from the IB.
- Sends instructions to subscribers (notification of data ready).
- Sends data to subscribers.

- Reports status to the DB.
- Sends information to Alarms, when appropriate.

The DC will have enough disk space to hold 3 nights worth of data from the DECam in normal operations. As a consequence, data requests must specify the image identifier.

Receipt of data from the IB will take less than 0.75 of the time required by the upstream data sources to collect (“read”) their data. This will ensure that upstream memory buffers will not overflow. If the DC finds that it is unable to write the image to disk before the next image is ready, an alarm must be generated and data acquisition stopped.

The DC will maintain a queue of pending commands. Commands have priorities, with data receipt being the highest, data transmission next, and configuration lowest. It may also be that subscribers have priorities (DM being the highest).

The DC will not assume that all subscribers will request all images. This may imply that receipt of data from the IB will never fail; that the oldest buffer will always be overwritten. It also implies that a request for nonexistent data is not necessarily an error; it is up to the subscriber to make that decision.

Implementation

The DC is entirely computational in nature and is therefore a software package running on a computer. It may be implemented as a part of the IB, in which case “receipt of data” functionality becomes irrelevant.

Implications for other components

DC functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
- That instrument components are not dependent for their operation on knowledge of the state of other components.

DECam SISPI component FCS

Functional Requirements Document (FRD)

Document name: DECcam-instr-FCS-FRD-v0.3.doc

Authors:

Mike Haney (m-haney@uiuc.edu)

Inga Karliner (karliner@uiuc.edu)

Mats Selen (mats@uiuc.edu)

Jon Thaler (jtt@uiuc.edu)

Date: 2005 09 05

Distribution: DES-DAQ group

Version History:

Version 0.1 Created by cloning and editing DECcam-instr-OCS_FRD-v0.2

Version 0.2 Edits by mjh and jtt.

Version 0.3 Edits by jta.

Purpose of this Document

This document is the Functional Requirements Document for the Focal Plane Control System (FCS) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

The term **Monsoon** refers to the hardware/software system developed primarily by NOAO to control and coordinate CCD readout. It consists of one or more Detector Head Electronics (DHE) crates, Pixel Acquisition Node (PAN) computers, each controlling and receiving data from one DHE crate, and a Monsoon Supervisory Layer (MSL) that oversees all the PANs. The FCS is an implementation of the MSL.

This document differentiates between the **commands** received by the FCS and the **instructions** it issues.

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The FCS is the interface to the CCD focal plane readout. Focal plane CCDs include the science array and the focus and wave front CCDs. The guide CCDs are read out separately (by the GCS). The FCS accepts commands, dispatches instructions to the PANs, and collects the resulting data from the PANs. The data is published in the FD repository for use by other SISPI components. The FCS must also report status information and alarms.

The FCS:

- Receives image capture commands from any processes wishing to obtain a focal plane exposure.
- Receives configuration commands designed to achieve specific focal plane configurations.
- Parses image capture and configuration commands into instructions for the PANs.
- Receives and monitors pixel data (images) and CCD temperatures from the PANs.
- Publishes image data to the FD.
- Receives and monitors status information from the PANs.
- Publishes status information in the FD or DB, or sends it to Alarms, as appropriate.
- On request, will collect a single focal plane image; the FCS will not accept the scheduling of multiple image collections in one command.
- On request, will collect a single-CCD image.
- Collects data from a fixed region of interest (ROI), controlled by the Focus process (possibly via DB). The initial ROI will be determined at system startup, and will not be updated.

Functional Components

The origin of a command should not matter to the FCS. However, commands should be tagged by origin, for more precise error reporting. This implies that issuers of instructions must identify themselves.

Proper FCS functionality during normal DES operation requires intricate coordination with the GCS and ICS (shutter). This coordination is the responsibility of the OCS.

The FCS will incorporate a mechanism for parsing incoming commands into instructions understandable by the PANs. The specific commands to be understood by the FCS, and their consequences, will be listed in the FCS Interface Control Document. This implies the existence of a command-instruction translation dictionary accessible to the FCS. Many of the commands will be identical to those in the GCS, and identity must be guaranteed.

When the FCS issues an instruction to the PANs, this implies a change of state or configuration of the focal plane. The FCS will incorporate a mechanism to identify allowable configuration changes and transmit, sequence, or block instructions accordingly. This implies that the FCS should maintain or have access to a description of the current state of the focal plane. This description, a state or configuration database, must be so designed as to ensure that it reflects reality at all times and does not suffer from the effects of communications latency between it, the FCS and instrument components.

The FCS will not maintain a queue of pending commands. Commands received that the FCS cannot process immediately will generate an error response.

The FCS assumes all responsibility for communication with the focal plane PANs. The FCS is also responsible for combining the focal plane data from the PANs to form one logical focal plane image for the FD in standard DES format.

The FCS will incorporate a mechanism for gracefully aborting any operation. If an abort is not possible, an error response will be generated. In this context, “operation” includes the entire focal plane exposure.

The FCS will not incorporate a GUI for use during integrated DECam operation, but will include one for stand alone operation (*e.g.*, in test stands) or debugging.

The FCS will communicate its results, status, and any error responses according to the DECam standards.

Implementation

The FCS is entirely computational in nature and is therefore a software package running on a computer. That computer will need to be sufficiently powerful to achieve the aims of the FCS without significant latency and, likewise, should never be engaged in other tasks that might threaten the FCS' ability to respond to communications and commands. It remains to be determined whether the FCS can reside in the same computer as a PAN.

If the FCS resides on a PAN, then every PAN shall have the FCS executable available to it. The assignment of the FCS to a specific PAN shall be determined by a name held in the configuration database. PANS shall be differentiated only by what they are doing, and not by what they are (software) capable of doing. The use of a name held in a database will allow fault recovery (if the FCS PAN fails during a night, the system can be restarted with a different FCS PAN).

The FCS will gracefully support one or more PANs, as determined by a list of names held in the configuration database. While it may be necessary to restart the system to change the number of PANs, it will not be necessary to recompile any software in order to change the number of PANs. The use of names held in a database will allow incremental development and testing (when only a limited number of PANs may be available), as well as fault recovery (if a PAN fails during a night, the system can be restarted with fewer PANs).

Implications for other components

FCS functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
 - Handshaking to verify reception of an instruction by a component.
 - Responses from a component on completion of a task.
- That instrument components make no attempt to control, delay or obstruct the behavior of other components. All such control is in the hands of the OCS.
- That instrument components are not dependent for their operation on knowledge of the state of other components.

DECam SISPI component FD

Functional Requirements Document (FRD)

Document name: DECcam-instr-FDS-FRD-v0.2.doc

Authors:

Jon Thaler (jtt@uiuc.edu)

Jim Annis (annis@fnal.gov)

Date: 2005 09 26

Distribution: DES-DAQ group

Version History:

Version 0.1 First draft

Version 0.2 Edits by jta

Purpose of this Document

This document is the Functional Requirements Document for the Focal plane pixel Data (FD) repository of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

This document differentiates between the **commands** received by the FD and the **instructions** it issues.

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The FD is the repository for the CCD focal plane pixel data and metadata.

The FD:

- Receives commands to initialize and update the subscriber list.
- Receives commands to initialize, maintain (e.g., delete an image?), and clear the data buffers.
- Receives data from the FCS.
- Sends instructions to subscribers (notification of data ready).
- Receives requests (commands) for data and responds with data or an error.
- Reports status to the DB.
- Sends information to Alarms, when appropriate.

Functional Components

The FD might maintain a buffer of focal plane images. Data requests must specify the image identifier.

The FD will maintain a queue of pending commands. Commands have priorities, with data receipt being the highest, data transmission next, and configuration lowest. It may also be that subscribers have priorities (IB being the highest).

The FD will not assume that all subscribers will request all images. In order to ensure that receipt of data from the FCS will never fail, the oldest buffer will be overwritten if the buffer space is full. This implies that a request for nonexistent data is not necessarily an FD error; it is up to the subscriber to make that decision. Failure to accept data from the FCS is an FD error.

The FD will communicate its status and any error responses according to the DECam standards.

Implementation

The FD is entirely computational in nature and is therefore a software package running on a computer. It may be implemented as a part of the FCS, in which case “receipt of data” functionality becomes irrelevant..

Implications for other components

FD functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
- That instrument components are not dependent for their operation on knowledge of the state of other components.

DECam SISPI component Focus

Functional Requirements Document (FRD)

Document name: DECcam-instr-Focus-FRD-v0.1.doc

Authors:

Mike Gladders (gladders@ociw.edu)

Tim Abbott (tabbott@ctio.noao.edu)

Version: 0.3

Start Date: 2005 08 30

Distribution: DES-DAQ group

Version History:

Version 0.1 Created by cloning and editing DECcam-instr-FCS_FRD-v0.1

Version 0.2 contains comments from Abbott

Version 0.3 contains edits by Gladders

Purpose of this Document

This document is the Functional Requirements Document for the Focus system of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

The Focus system uses information from so-called focus chips placed in the CCD focal plane. These are currently envisioned as 2kx2k LBNL-style CCDs (apart from size, of precisely the same style as the science chips, but possibly of lower cosmetic quality) arranged at the outer (vignetted) edge of the science field and mounted with a tilt with a throw of several mm so as to sample a range of foci.

Introduction

The primary purpose of the Focus system is to monitor and correct for focus offsets in the imaging system. The relevant timescale for focus correction is on order of the science acquisition, and currently the focus chips are envisioned to be read out by the same electronics and at the same cadence as the science array. With this setup the Focus system has no separate image acquisition controls – the CCDs are read out as part of the science array.

Real-time analysis of the focus images will be used to monitor the drift of the telescope focus with time. Under typical (small slew) operations this analysis will feed inter-exposure corrections to the system. After large slews likely a “focus frame” may need to be acquired to re-establish the nominal telescope focus; experience with the system on-sky will dictate the details of this mode.

The most out-of-focus portions of these chips will also provide some information regarding active aberrations of the optical system (primarily translation of the optical axis of the primary wrt the corrector axis). It is unclear at this time whether this information will ultimately be useful, or if it will be used.

The Focus System:

- Accesses the focus chips images from readout of the science array
- Provides desired values of the instrument focus including relevant information such as a timestamp, the telescope position, and the filter used
- Possibly monitors changes in the alignment of the instrument wrt to the primary.

Implementation

The analysis of the focus images in real time will be a modestly demanding computation. The timescale required to analyse and produce a focus offset value for a given frame will be dependent on what the OCS intends to do with this information. It is possible to envision a situation in which an immediate response is required from the OCS, and so relatively rapid processing (<1 sec) is likely required. This may required a dedicated computer.

DECam SISPI component GCS

Functional Requirements Document (FRD)

Document name: DECcam-instr-GCS-FRD-v0.1.doc

Authors:

Mike Haney (m-haney@uiuc.edu)

Inga Karliner (karliner@uiuc.edu)

Mats Selen (mats@uiuc.edu)

Jon Thaler (jtt@uiuc.edu)

Version: 0.2

Date: 2005 09 5

Distribution: DES-DAQ group

Version History:

Version 0.1 First version.

Version 0.2 Edits by Mike Haney

Purpose of this Document

This document is the Functional Requirements Document for the Guider Control System (GCS) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

The term **Monsoon** refers to the hardware/software system developed primarily by NOAO to control and coordinate CCD readout. It consists of one or more Detector Head Electronics (DHE) crates, Pixel Acquisition Node (PAN) computers, each controlling and receiving data from one DHE crate, and a Monsoon Supervisory Layer (MSL) that oversees all the PANs. The GCS is an implementation of the MSL.

This document differentiates between the **commands** received by the GCS and the **instructions** it issues.

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The GCS is the interface to the CCD guider readout. Guide CCDs do not include the science array or the focus and wave front CCDs, which are read out separately (by the FCS). The GCS accepts commands, dispatches instructions to the PANs, and collects the resulting data from the PANs. The data is published in the GD repository for use by other SISPI components. The GCS must also report status information and alarms.

Similarities between the GCS and the FCS are entirely intentional. Both have significant commonalities.

The GCS:

- Receives image capture commands from any processes wishing to obtain telescope guiding information.
- Receives configuration commands designed to achieve specific guider configurations.
- Parses image capture and configuration commands into instructions for the PANs.
- Receives and monitors pixel data (images) and CCD temperatures from the PANs.
- Publishes image data to the GD.
- Receives and monitors status information from the PANs.
- Publishes status information in the FD or DB, or sends it to Alarms, as appropriate.
- On request, will collect a single guider image; the GCS will also accept the scheduling of multiple image collections in one command.
- Collects data from a variable region of interest (ROI), controlled by the Guide process (possibly via DB). The initial ROI will be determined at system startup, and may be updated at any time.

Functional Components

The origin of a command should not matter to the GCS. However, commands should be tagged by origin, for more precise error reporting. This implies that issuers of instructions must identify themselves.

Proper GCS functionality during normal DES operation requires intricate coordination with the FCS and ICS (shutter). This coordination is the responsibility of the OCS.

The GCS will incorporate a mechanism for parsing incoming commands into instructions understandable by the PANs. The specific commands to be understood by the GCS, and their consequences, will be listed in the GCS Interface Control Document. This implies the existence of a command-instruction translation dictionary accessible to the GCS. Many of the commands will be identical to those for the FCS, and identity must be guaranteed.

When the GCS issues an instruction to the PANs, this implies a change of state or configuration of the guider. The GCS will incorporate a mechanism to identify allowable configuration changes and transmit, sequence, or block instructions accordingly. This implies that the GCS should maintain or have access to a description of the current state of the guider. This description, a state or configuration database, must be so designed as to ensure that it reflects reality at all times and does not suffer from the effects of communications latency between it, the GCS and instrument components.

The GCS will not maintain a queue of pending commands. Commands received that the GCS cannot process immediately will generate an error response.

The GCS assumes all responsibility for communication with the guide PANs. The GCS is also responsible for combining the guider data from the PANs to form one logical guider image for the GD.

The GCS will incorporate a mechanism for gracefully aborting any operation. If an abort is not possible, an error response will be generated. In this context, “operation” includes an entire sequence of guide exposures that take place during a focal plane exposure.

The GCS will not incorporate a GUI for use during integrated DECam operation, but will include one for stand alone operation (*e.g.*, in test stands) or debugging.

The GCS will communicate its results, status, and any error responses according to the DECam standards.

Implementation

The GCS is entirely computational in nature and is therefore a software package running on a computer. That computer will need to be sufficiently powerful to achieve the aims of the GCS without significant latency and, likewise, should never be engaged in other tasks that might threaten the GCS' ability to respond to communications and commands. It remains to be determined whether the GCS can reside in the same computer as a PAN.

If the GCS resides on a PAN, then every PAN shall have the GCS executable available to it. The assignment of the GCS to a specific PAN shall be determined by a name held in the configuration database. PANS shall be differentiated only by what they are doing, and not by what they are (software) capable of doing. The use of a name held in a database will allow fault recovery (if the GCS PAN fails during a night, the system can be restarted with a different GCS PAN).

The GCS will gracefully support one or more PANS, as determined by a list of names held in the configuration database. While it may be necessary to restart the system to change the number of PANS, it will not be necessary to recompile any software in order to change the number of PANS. The use of names held in a database will allow incremental development and testing (when only a limited number of PANS may be available), as well as fault recovery (if a PAN fails during a night, the system can be restarted with fewer PANS).

Implications for other components

GCS functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
 - Handshaking to verify reception of an instruction by a component.
 - Responses from a component on completion of a task.
- That instrument components make no attempt to control, delay or obstruct the behavior of other components. All such control is in the hands of the OCS.
- That instrument components are not dependent for their operation on knowledge of the state of other components.

DECam SISPI component GUI

Functional Requirements Document (FRD)

Document name: DECcam-instr-GUI-FRD-v0.1.doc

Authors:

K. Wyatt Merritt (wyatt@fnal.edu)

Version: 0.1

Date: 2005 08 2

Distribution: DES-DAQ group

Version History:

Version 0.1 Initial version

Introduction

Provide display infrastructure and graphical command input for the other SISPI systems. The types of displays required are:

- Histograms
- Strip charts
- Color-coded arrays of status icons
- Scrolling text windows

Functional Components

Implementation

TBD

Implications for other components

Receives control from:

- Human observers

Sends control to:

- All subsystems (potentially)

Receives data from:

- Alarms

Sends data to:

- None

Issues:

- What to use as IPC .
- What GUI implementation to use

DECam SISPI component “Guide”

Functional Requirements Document (FRD)

Distributed to DES-DAQ@fnal.gov as example only

Document name: DECcam-instr-Guide-FRD-vX.Y.doc

Author: Tim Abbott, CTIO, tabbott@ctio.noao.edu

Version: 0.1

Date: 2005 08 01

Distribution: DES-DAQ group

Version History:

v0.1 distributed to DES-DAQ@fnal.gov as example only.

Purpose of this document

This document is the Functional Requirements Document for the Guide package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

The Guide package is a software process which acquires image data from the guide CCDs on the DECcam focal plane via the Monsoon B, Guider Control System (GCS) and Guider pixel Data store (GD). From this image data the Guide package derives telescope tracking error signals and makes them available to correct telescope tracking.

Terminology

Tracking: the telescope is tracking when it is being driven across the sky at the sidereal rate in Right Ascension. This is an open-loop process.

Guiding: the telescope is being guided when it is receiving error signals from a guide package which can be used to provide small adjustments to the tracking to keep the telescope on a specific region of sky. This is a closed-loop process.

Error signal: an error signal is an instantaneous measure in Right Ascension and Declination of the offset of the telescope from the required position on the sky.

Functional Overview

The Guide process should be able to accept three general directives:

1. Initialize – in which the Guide process is instructed to acquire guide star images, identify guide stars in those images and establish a zero point on which to guide the telescope during an exposure on the night sky.
2. Guide – in which the Guide process is instructed to acquire guide star images, identify guide stars in those images and generate error signals indicating the offset of the field from the zero point identified in the response to the Initialize directive. This process continues indefinitely.
3. Stop – in which the Guide process is instructed to stop acquiring guide star images, stop generating error signals and return to a static state awaiting further instruction.

Context

The DECam FPA is expected to incorporate multiple CCDs specifically dedicated to guiding. They will be located in the same plane as the science CCDs so as to have the same focus.

The Guide CCD specification may be found in: ??? (The Monsoon B FRD?)

The Guide process may be expected to handle image data from up to six (6) CCDs each of 2048×2048 physical pixels each of which is 15×15μm or XX×XX arcsec on the sky. The Guide process should be able to interpret overscan data in both dimensions.

The Guide process will require a user interface which is expected to be incorporated into the DECam GUI Package (see SISPI definition and GUI FRD). This interface will incorporate methods to deliver atomic commands to the Guide process, display status information and display images taken from the guider CCDs with appropriate overlay markings.

The Initialize Directive

The entire imaging area of the guide CCDs may be used to acquire the initial frame for generating the zero point. The Guide process must be able to acquire full-frame images from the guide CCDs.

On receiving a directive to initialize, the Guide process will acquire a set of images from the guider CCDs and use the stars imaged therein to establish a reference position or zero point for subsequent guiding of the telescope. As such, it will establish a map of stars within the fields of view of the Guide CCDs and an optimal location for that map.

The guide zero point may be established according to three potential operating modes:

1. From-catalog – in which the Guide process is expected to automatically locate a specific star or group of stars in the field and use them to establish the zero point for guiding.
2. Directed – in which the Guide process is expected to automatically locate a specific star or group of stars in the field, but use a previously defined zero point for guiding.
3. On-the-fly – in which the Guide process automatically identifies any stars that it can and uses their initial locations to define the zero point for subsequent guiding.

The stars to be located in the From-catalog and Directed modes above may be provided at run time or extracted from a catalog which is available to the Guide process.

In all three operating modes, provision should be made for the interactive intervention of a human operator, to override or specify guide star and zero point selections.

As part of the initialization process, the Guide process will also generate a partial readout map for minimizing the areas of the guide CCDs to be read out, reducing their readout time and maximizing their duty cycle. A mechanism must be established for transmitting this partial readout map to the GCS and/or Monsoon B.

Possible error conditions:

Insufficient guide stars located. (Respond by increasing exposure time?)

The stars specified in “From-catalog” or “Directed” modes cannot be located

The FPA is already integrating (it may not be advisable to start guiding after a science integration is begun)

The Guide Directive

On receiving the Guide directive, the Guide process will acquire whole or partial images from the guider CCDs. It will attempt to locate the guide stars identified in the initialization sequence in those images and it will calculate their positional offset from the zero point established by the initialization sequence. It will repeat this function indefinitely, until the guide stars cannot be located in the guider CCD images or until the Stop directive is received.

Generating the error signals for a given image should always be completed before the next image is available.

Possible error conditions:

The Guide directive is received when the telescope has not been guided for XXX seconds or the telescope has been moved from the location on the sky where the last initialization was performed.

The guide stars are lost due to tracking errors, clouds or bright sky.

The Stop Directive

On receiving a Stop directive, the Guide process will cease all processing at the first available opportunity. It will, however, maintain the zero point in use so that guiding may be restarted on the same field if required.

Processing outline

The itemized list below illustrates the Guide process as an outline.

1. Initialize

1.1. Poll telescope status

If telescope not tracking, there's no point in acquiring guide stars.

1.1.1. Record latest pointing instruction

Will need this to confirm validity of guide field. I.e. If a new pointing instruction is delivered, this invalidates the guide field.

1.2. Check shutter status

If the shutter is closed, can't acquire an image. It is assumed that the guide CCDs are frame transfer and do not require shutter control to acquire an image.

1.3. Check FPA status

If the FPA is integrating, it is a possible error condition to try to acquire a guide image.

1.4. Acquire guide image

If the guider is working "from catalog", it may already know where guide stars should be found within the image and then may request only windowed readouts from Monsoon B. This will require additional processing steps.

1.5. Process guide field

1.5.1. Locate stars in field

Algorithm TBD

1.5.2. Generate readout windows

1.6. Upload readout windows to GCS and/or Monsoon B

1.7. Signal readiness to guide.

1.8. Poll telescope status until guide field invalidated

1.9. If guide field invalidated, signal unreadiness to guide or loop to item 1.1, reacquire guide field.

2. Guide

2.1. Acquire guide image

Once guiding, guide images should be delivered to the guider process without its needing to request them. It is assumed that guide images are processed faster than they can be delivered. If the guide loop is not running, the guider process should start it at this point.

2.2. Poll telescope status

Is the telescope tracking? Has the telescope been repointed (invalidating the guide field)?

2.3. Process guide field

2.3.1. Locate stars in field.

Algorithm TBD

2.3.2. Generate error signal

2.3.3. Send error signal to TCS

2.4. Loop to point item 2.1

3. Stop guiding

Data Products

The guide process, besides generating the error signals necessary for guiding, will also produce measures of brightness and size of the guide stars, and the brightness of the sky all of which may be used for tracking observing conditions in real time.

Implementation

The guide process is entirely computational in nature and should run on a computer of sufficient power to fulfill the requirements of this document. It is recommended that this computer not perform any task not relating to guiding the telescope. As such, this computer could reasonably be responsible for the GD and GCS.

DECam SISPI component IB

Functional Requirements Document (FRD)

Document name: DECcam-instr-IB-FRD-v0.2.doc

Authors:

Jon Thaler (jtt@uiuc.edu)

Joe Mohr (jmohr@uiuc.edu)

Jim Annis (annis@fnal.gov)

Date: 2005 09 26

Distribution: DES-DAQ group

Version History:

Version 0.1 First version

Version 0.2 Suggestions by jta

Purpose of this Document

This document is the Functional Requirements Document for the Image Builder (IB) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

This document differentiates between the **commands** received by the IB and the **instructions** it issues.

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The IB assembles all DES image data that will be sent to data management (DM). This will include not only the primary image data from the focal plane CCDs, but also any other image specific data that might be important during later analysis (*e.g.*, guider error signals during the exposure). The data is published in the DC repository for use by other SISPI components. The IB must also report status information and alarms.

The IB:

- Receives commands from processes that control image acquisition.
- Parses commands into requests for image data.
- Receives image data and assembles it into DES images.
- Publishes image data to the DC.
- Publishes status information in the DB, or sends it to Alarms, as appropriate.

Functional Components

The origin of a command should not matter to the IB. However, commands should be tagged by origin, for more precise error reporting. This implies that issuers of instructions must identify themselves.

The IB will maintain a table of data sources and of their importance to the image. The IB will incorporate a mechanism for parsing incoming commands of two kinds: configuration and data collection. Configuration commands will cause the data source table to be modified. Data collection commands will be translated into instructions understandable by the image data sources. The specific commands to be understood by the IB, and their consequences, will be listed in the IB Interface Control Document. This implies the existence of a command-instruction translation dictionary accessible to the IB.

Data collection and image building will take less than 0.75 of the time required by the upstream data sources to collect (“read”) their data. This will ensure that upstream memory buffers will not overflow.

The IB will incorporate a mechanism for gracefully aborting any operation. If an abort is not possible, an error response will be generated.

Upon notification of data availability from a data source (*e.g.*, FD) the IB must retrieve it and write it to DC. The failure to finish this process before a new data notification is an IB error.

The IB will incorporate a mechanism for detecting nonresponsive data sources, and respond appropriately. If the data source (*e.g.*, FD) is noted as critical, it is an error if the data cannot be collected. If the data source is not critical (*e.g.*, some ICS telemetry), the image should be written to DC with a “data not available” flag.

The IB will communicate its results, status, and any error responses according to the DECam standards.

Implementation

The IB is entirely computational and is therefore a software package running on a computer. That computer will need to be sufficiently powerful to achieve the aims of the IB without significant latency and, likewise, should never be engaged in other tasks which might threaten the IB’s ability to respond to communications and commands.

Implications for other components

IB functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
 - Handshaking to verify reception of an instruction by a component.
 - Responses from a component on completion of a task.

DECam SISPI component ICS

Functional Requirements Document (FRD)

Document name: DECcam-instr-OCS-FRD-v0.1.doc

Authors:

Del Allspach (allspach@fnal.gov)

Mike Haney (m-haney@uiuc.edu)

Version: 0.1

Date: 2005 09 06

Distribution: DES-DAQ group

Version History:

Version 0.1 Initial Version

Purpose of this Document

This document is the Functional Requirements Document for the Instrument Control System (ICS) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

The document also refers to the **Instrument** which is the collective electrical, mechanical, and thermal system comprised of the Corrector Focus, Shutter, Filters, Focal Plane Cooling, Vacuum, FEE Cooling, etc.

Introduction

The ICS is the controls system which carries out commands from the OCS directed to the Instrument. The ICS will also receive commands from the TCS only when operating in the f/8 configuration. The ICS contains the control logic required to execute these commands.

The ICS:

- Provides control for Corrector Focus, Shutter, Filters, Focal Plane Cooling, Camera Vacuum, Cooling to FEE and other Cage electronics, etc.
- Contains process control loops plus other logic required to provide this control.
- Contains interlock logic required to protect Instrument components.
- Publishes ICS I/O real time values and device status so as to be available to the OCS, Status and Log DB, and ID in real time.
- Publishes interlock status and OCS command completion status so as to be available to the OCS, Status and Log DB, and ID.

Functional Components

The ICS will accept commands from the OCS to position the shutter, filters, and focus as instructed. The ICS will accept appropriate commands from the TCS when the Instrument is operating in the f/8 configuration. When the ICS has completed a given task, it will publish an action status indicating that the request was successfully completed. In case the instruction cannot be completed due to equipment failure or other detected failures, the failed action status will be published. The ICS will control the focal plane temperature profile per set points communicated via the OCS. The ICS will control camera vacuum, FEE and other electronic cooling, *etc.* per set points communicated via the OCS.

The ICS will maintain a nonvolatile, predefined specification for the default operating point (state) of the Instrument. In the absence of communications from the OCS, the ICS will use this predefined set for control purposes.

The ICS will make any and all ICS I/O real time values available to appropriate databases. This will allow for alarms and GUI display, *etc.* In the same way, interlock status and OCS command completion status may be made available.

The ICS will allow for an abort command from the OCS and respond by positioning all Instrument devices to a their predefined default state(s).

Implementation

This is an industrial controller with several I/O modules capable of sending and receiving standard industrial signals to/from the instrumentation and equipment required of the Instrument. The industrial controller will contain all required logic to provide controls, interlocks, and communications necessary to perform its tasks. In addition to the system instrumentation and equipment (sensors, motors, switches, valves, *etc.*) appropriate power supplies are required to complete the system.

The nonvolatile, predefined, default “state” for the Instrument may involve a number of remotely adjustable parameters (downloadable code and set points, *etc.*) as well as non-remotely adjustable parameters (*e.g.*, limit switches). In all cases, the default “state” will be nonvolatile (and therefore will not require remote “downloading” on system startup or restart), and safeguarded against accidental or otherwise unauthorized alteration.

Implications for other components

ICS functionality implies the existence of a common communications protocol with other instrument components allowing for:

- Receiving commands.
- Transmitting instructions.
- Handshaking to verify reception of an instruction by a component.
- Responses from a component on completion of a task.

DECam Observation Tactical Planner (ObsTac)

Functional Requirements Document

Document name: DECcam-instr-ObsTac-FRD-v1.doc

Author: Jim Annis, Fermilab, annis@fnal.gov

Version: 1

Date: 2005 08 29

Distribution: DES-DAQ group

Version History:

Version 1 is the initial version

Purpose of this Document

This document is the Functional Requirements Document for the observation tactical planner (ObsTac) package of the DECcam Survey Image System.

Terminology

ObsTac is a changed from the previous nomenclature of ObsSeq due to the unfortunate connotations of obsequiousness and the actual nature of its job, which is tactical planning of observations.

Introduction

The OCS is the interface to the instrument for any process wishing to obtain an exposure. Most basically, these may be an operator interface (GUI) and/or an observation planner, both of which are treated by the OCS as any other instrument component. In detail, the OCS contains all high level, inter-component control logic required to execute commands from these processes. The GUI allows astronomer on the site to run the system; it provides a user friendly interface to the OCS.

ObsTac is a program that calculates the next observation given a series of tactical plans, input from the OCS status banks, and strategic control from astronomers on site. The plans are algorithms, for example, for choosing the next tile to observe given the last tile observed (in the case of the DES), or a dither pattern (in the case of a community observer).

Functional Requirements

ObsTac must be capable of being overridden by the on-site astronomer.

ObsTac must be aware of relevant observatory states such as weather, twilight, and moonrise. Likewise, ObsTac must be aware of instrumental states such as exposure complete and of survey status such as tiles completed.

ObsTac must allow for simple scripting of a given series of exposures. This the primary useage case for community astronomers and a good general purpose dither pattern should be provided.

ObsTac must allow for scripting of algorithms that allow for efficient choice of the next tile to observe and generally the next observation to take. A typical DES algorithm would take as inputs: 1) time, to compute moon position and twilight status 2) survey status, in the form of tiles observed. It would locate candidate tiles from the survey status information, and compute airmasses for the candidates. Given this information it would compute a metric for each candidate and then choose the best tile to observe.

ObsTac must send to the OCS all the information needed for the observation: ra, dec, exposure time, filter.

Depending on weather, there may be other tactical plans to follow. It will be up to the on-site astronomer to choose the appropriate plan.

Likewise, there may be calibration plans to follow. For example, we imagine pointing at a resolved local galaxy every night for photometric and astrometric calibrations. The whole sequence of observations would constitute a tactical plan, to be chosen by the on-site astronomer.

ObsTac, once started with a plan, will continue generating observations until there are no more valid observations to take (in the context of the plan, i.e., the sun has come up, or the calibration sequence is completed) or overridden by the on-site astronomer.

ObsTac will update its survey status information during the night automatically. The survey status information must be able to be reloaded given, for example, a list of tiles completed by data reduction.

Implementation

The ObsTac is entirely computational in nature and is therefore a software package running on a computer.

The ObsTac itself is probably best constructed from a scripting language given the requirement of easily constructed plans.

Implications for other components

The ObsTac requires information flow from the OCS, and it needs to be able to send information to the OCS. The OCS should consider the ObsTac stateless (and just keep broadcasting what it knows of the relevant observatory status) and should be prepared for asynchronus commands from the ObsTac.

DECam SISPI component OCS

Functional Requirements Document (FRD)

Document name: DECcam-instr-OCS-FRD-v0.2.doc

Author: Tim Abbott, CTIO, tabbott@ctio.noao.edu

Version: 0.2

Date: 2005 08 19

Distribution: DES-DAQ group

Version History:

Version 0.1 was too much about implementation

Purpose of this Document

This document is the Functional Requirements Document for the Observational Control System (OCS) package of the DECcam Survey Image System as defined in the Process Integration document (aka SISPI definition).

Terminology

The acronym OCS is potentially ambiguous as it can be taken for “Observatory Control System” which can imply a more extensive mandate over all observatory systems. The OCS as described here will contain aspects of this, but the intention is only to indicate control of observatory systems required to obtain an exposure (such as telescope pointing, but most likely not, for example, control of the dome shutters).

This document differentiates between the **commands** received by the OCS and the **instructions** it issues.

This document refers to instrument **components** which are those items identified in the SISPI document for which FRDs and ICDs will be written.

Introduction

The OCS is the interface to the instrument for any process wishing to obtain an exposure. Most basically, these may be an operator interface (GUI) and/or an observation sequencer (ObsSeq), but the OCS will treat these in the same way as any other instrument component. The OCS contains all high level, inter-component control logic required to execute commands from these processes. The OCS is the clearing house for all inter-process communication within DECcam. There may be exceptions in order to expedite time-critical operations (e.g. alarms, the guide control loop segment between the GCS and TCS).

The OCS:

- Acts as the interface to the instrument for any higher level process wishing to obtain an exposure.
- Acts as a central, single-point location for the overall control of DECam.
- Mediates inter-component communications within DECam.
- Receives commands designed to achieve specific instrument and telescope configurations and configuration changes.
- Parses those commands into instructions for the telescope and instrument components.
- Transmits instructions to telescope and instrument components thus assigning them tasks.
- Receives and monitors status information from telescope and instrument components.
- Monitors and coordinates instrument configuration, configuration changes, image acquisition, and data transfer.

The OCS is basically a monitor loop which monitors communications channels for incoming **commands**, parses these commands into **instructions** and transmits these instructions to the relevant instrument components.

Functional Components

The origin of a command should not matter to the OCS except that the OCS will incorporate a mechanism to ensure that any commands it receives originate from *authorized* sources.

The OCS will incorporate a mechanism for parsing incoming commands into instructions understandable by, and which can assign tasks to, the instrument components. The specific commands to be understood by the OCS, and their consequences, will be listed in the OCS Interface Control Document. This implies the existence of a command-instruction translation dictionary accessible to the OCS.

When the OCS issues an instruction to a component of the instrument, this implies a change of state or configuration of the instrument. The OCS will incorporate a mechanism to identify allowable configuration changes and transmit, sequence, or block instructions accordingly. This implies that the OCS should maintain or have access to a description of the current instrument state. This description, a state or configuration database, must be so designed as to ensure that it reflects reality at all times and does not suffer from the effects of communications latency between it, the OCS and instrument components.

The previous paragraph implies that the OCS will maintain a queue of pending commands and/or instructions while they wait for the appropriate instrument configuration (what happens if it never arrives...?). The OCS will incorporate a mechanism for sequencing instructions for tasks that *cannot* be executed concurrently, a mechanism for simultaneously issuing instructions for tasks that *can* be executed concurrently and a mechanism for distinguishing between the two. These mechanisms will allow for the possibility that one command or instruction may pass another in either queue (hence, for example, an **abort** command may override a pending **expose** command).

The OCS assumes all responsibility for handling wait states between instrument components. This implies the need for a suitable command/handshake/response or publish/subscribe mechanism for communications between the OCS and the instrument components.

The OCS will incorporate a mechanism for gracefully aborting any operation. This implies that all components of the instrument will also incorporate such a mechanism. When such an abort directive is issued, all instrument components should cleanly stop whatever they are doing and return to a well defined default state. (This is a perennially ignored issue in many astronomical instruments accounting for the loss of untold amounts of telescope time.)

The OCS will not incorporate a GUI, but should include some mechanism for direct communication via, e.g., an xterm for debugging purposes. Likewise there should be a mechanism for monitoring OCS functions directly (event signaling, error logging).

Implementation

The OCS is entirely computational in nature and is therefore a software package running on a computer. That computer will need to be sufficiently powerful to achieve the aims of the OCS without significant latency and, likewise, should never be engaged in other tasks which might threaten the OCS' ability to respond to communications and commands.

The OCS in itself is a general purpose module for the instrument, specific knowledge of instrument components will reside in the command/instruction library and configuration tables.

Implications for other components

The OCS' functionality implies:

- The existence of a common communications protocol with other instrument components allowing for:
 - Receiving commands.
 - Transmitting instructions.
 - Handshaking to verify reception of an instruction by a component.
 - Responses from a component on completion of a task.
- That instrument components do not attempt to directly communicate with each other without the mediation of the OCS.
- That instrument components make no attempt to control, delay or obstruct the behavior of other components. All such control is in the hands of the OCS.
- That instrument components are not dependent for their operation on knowledge of the state of other components.

Quality Analysis (QA)

Functional Requirements Document (FRD)

Document name: DECcam-instr-QA-FRD-v0.1.doc

Author: Huan Lin, Fermilab, hlin@fnal.gov

Version: 0.1

Date: 2005 08 29

Distribution: DES-DAQ group

Version History:

v0.1: Initial version

Purpose of this Document

This document describes the functional requirements for the Quality Analysis (QA) component of the DECcam Survey Image System.

Terminology

GUI: Graphical User Interface

DC: Data Cache

DECcam: Dark Energy Camera

DES: Dark Energy Survey

DM: Data Management

QA: Quality Analysis

Introduction

The Quality Analysis (QA) component of the DECcam Survey Image System will provide real-time QA for Dark Energy Survey (DES) observing operations. The QA component will perform basic tests to assess the quality of the incoming astronomical imaging data and will provide the test results as feedback to the observer and observing system to help in adjusting the nightly observing strategy. In addition, the QA component will also perform tests that will allow identification of instrumentation problems on the DECcam CCDs.

The QA tests performed here are not intended to replace the in-depth data quality tests to be carried out during full data processing by the Data Management (DM) system. Rather, the QA tests here are intended to identify changes in observing conditions which may warrant changes in observing strategy, as well as to identify instrumentation problems with the DECcam CCDs which may require human intervention. Examples of such circumstances include high sky background counts on a science exposure which may indicate clouds in the field of view, or unexpected bias values in the overscan region of a particular CCD that may indicate a problem with that device. Also, as noted below, for a number of these QA tests, it is sufficient to carry out the tests for just a subset of the CCDs rather than for the full DECcam mosaic.

Functional Overview

The QA component should carry out the following functions:

- (1) Process nightly bias and flat field calibration data, and test for all CCDs whether bias values, read noise, and flat field patterns are within expectations (as defined by a set of master bias and flat field images).
- (2) Process, as available, standard star observations to check nightly values of photometric zeropoints and warn of large deviations from expected values.
- (3) Process subset of CCDs on each science exposure to measure sky values, and warn of large sky values and/or large variations from CCD to CCD, e.g., due to non-photometric conditions.
- (4) Process subset of CCDs on each science exposure to measure the PSF size of bright unsaturated stars, in order to measure seeing values and warn of poor seeing conditions or large seeing variations across the focal plane.
- (5) Process subset of CCDs on each science exposure to check for bright unsaturated USNO-B stars expected to be present and measure their positions, and warn if not found or if there are large astrometric offsets.
- (6) Check overscans of all CCDs on each science exposure to test whether bias and read noise values are within expectations.
- (7) The tests described in (3)-(6) will be carried out within the minimum DES science image exposure time (100 sec) plus readout (17 sec), in order to enable real-time QA during the night.
- (8) The QA test outputs will be presented in graphical and tabular format accessible using the Graphical User Interface (GUI) to facilitate human inspection of the QA results.
- (9) The QA tests are automatically initiated as new data appears in the Data Cache (DC).

SkyCam

Functional Requirements Document (FRD)

Document Name: DECam-instr-SkyCam-FRD-vX.Y.doc

Author: Douglas L. Tucker, FNAL, dtucker@fnal.gov

Version: 0.5

Date: 2005-08-10

Distribution: DES-DAQ group

Version History:

v0.5: Initial outline, text, and figures.

Purpose of this document

The purpose of this document is to provide functional requirements for a proposed 10 micron all-sky cloud camera to be placed at CTIO to support the Dark Energy Survey.

Terminology

APO: Apache Point Observatory¹

ARC: Astrophysical Research Corporation

DECam: Dark Energy Camera

DES: Dark Energy Survey

DIMM: Differential Image Motion Monitor

IRSC: Infra-Red Sky Camera²

SDSS: Sloan Digital Sky Survey³

TASCA: Tololo All-Sky CAmera⁴

Functional Overview

The SkyCam system should:

1. image the full sky at a wavelength of ~10 microns once every 30 seconds throughout the course of nightly operations of the Blanco 4.0m telescope,
2. process the images in real-time,
3. output in real-time a GIF version of the processed image to a webpage,
4. output in real-time a quantitative diagnostic indicating the cloudiness of the sky (e.g., the

¹ <http://www.apo.nmsu.edu/>

² <http://www.ctio.noao.edu/telescopes/dimm/dimm.html>

³ <http://www.sdss.org/>

⁴ <http://www.ctio.noao.edu/~david/tasca.htm>

- rms of the pixel values from the most recent processed image) to a web-accessible graph and to an archival database,
5. create an animation based upon the processed images from the past hour to detect cloud movement and output the animation to a webpage,
 6. create an animation based upon the full night's processed images at the end of each night, and
 7. archive the raw and processed FITS images, processed GIF files, the full-night animation to a web-accessible directory.

These functional requirements are based upon the actual functionality of the APO IRSC, which, in its current incarnation, has been operating successfully since 2001.⁵

Context

In order to make optimal use of the DECam for the Dark Energy Survey (DES), observations of standard star fields will be sparse in the time domain. Therefore, it will be difficult to ascertain how photometric a given observation is, especially in the first year of observations when tilings will still be few in number. Therefore, it is necessary to use an independent method of determining the sky conditions. At APO, a 10 micron all-sky camera is used to determine the photometric quality both for SDSS and for ARC3.5m observations, and has proven hugely successful for this purpose.

At CTIO, a Differential Image Motion Monitor (RoboDIMM), an optical all-sky camera (TASCA), a Multiple-Aperture Scintillation Sensor (MASS), and a weather station all help to monitor sky conditions. These all have their own individual strengths and weaknesses in determining the photometric conditions over the full sky. A 10 micron all-sky camera (SkyCam) will provide an essential addition to this suite of monitoring tools.

Processing Outline

TBD.

Data Products

TBD.

Implementation

We recommend implementing a duplicate of the current incarnation of the Apache Point Observatory Infra-Red Sky Camera (IRSC). This is a well-tested system, and relatively inexpensive (on the order of \$15,000 in total for parts, which includes the price of the mid-IR video camera.) The following discussion draws heavily upon the APO IRSC documentation website, which is at http://hoggpt.apo.nmsu.edu/irsc/irsc_doc/.

The APO IRSC uses a Raytheon Thermal-Eye 300A⁶ mid-IR detector suspended above a hyperbolic aluminum mirror (see Figs. 1 & 2). This camera, which works similarly to a low-

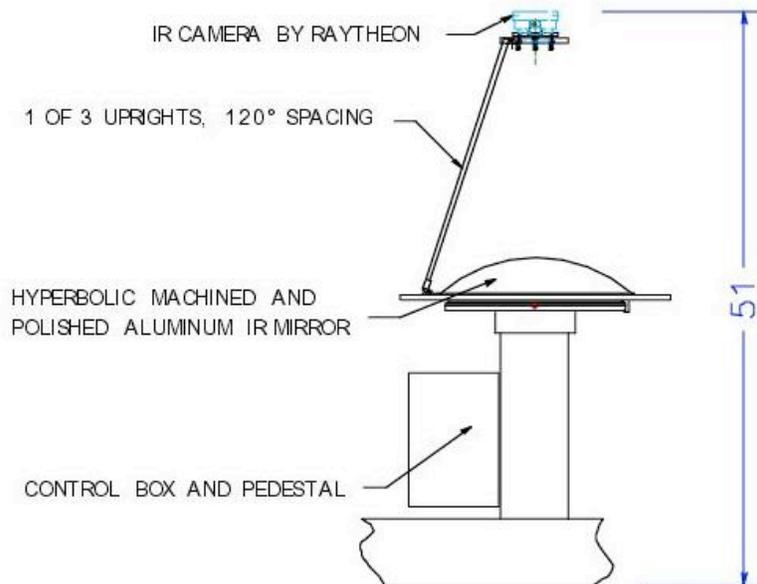
⁵ <http://hoggpt.apo.nmsu.edu/irsc/tonight/>

⁶ Note that Raytheon no longer manufactures the Thermal-Eye 300A. A similar product, however, is the Raytheon

resolution surveillance video camera, images the full sky down to the horizon at a rate of 30 Hz (i.e., 30 images per second). A standard, off-the-shelf video framegrabber digitizes this video output. The resultant images are co-added and then made available for display in 30 second intervals. These raw FITS images are then further processed to mask out the camera's support structures, and the brightness and standard deviation of the sky is then calculated. The images are converted to PGM/GIF image format for display on the web, and the positions of the SDSS 2.5m and the ARC 3.5m telescope pointings as well as an alt-az grid is overlaid on each image (Figs. 3 & 4). The webpage also displays running graphs of the sky brightness and the sky standard deviation.

The sky standard deviation is robust in determining structure in the sky images (i.e., in detecting clouds, even light cirrus.)

The greatest advantage of the Raytheon 300 series camera is that it outputs images at 10 micron wavelengths at video rates with a reasonably good response. A disadvantage to this camera is that it has built-in autonormalizing and autoscaling functions, which confound methods of properly calibrating the sky brightness. Nevertheless, the sky brightness can still be used for diagnostic purposes and therefore it is plotted along with the standard deviation to watch cloud cover trends throughout the night. Figure 5 shows an example of both plots; in the sky standard deviation plot, the horizontal green line indicates the empirically determined value below which conditions are photometric.



Thermal-Eye 300D, which is described at <http://www.raytheoninfrared.com/productcatalog/prodItem25.html> and at http://www.raytheoninfrared.com/admin/file/300D_elec_9.pdf.

Fig. 1: Design of the current APO IRSC, commissioned in 2001.



Fig. 2: A photograph of the APO IRSC.

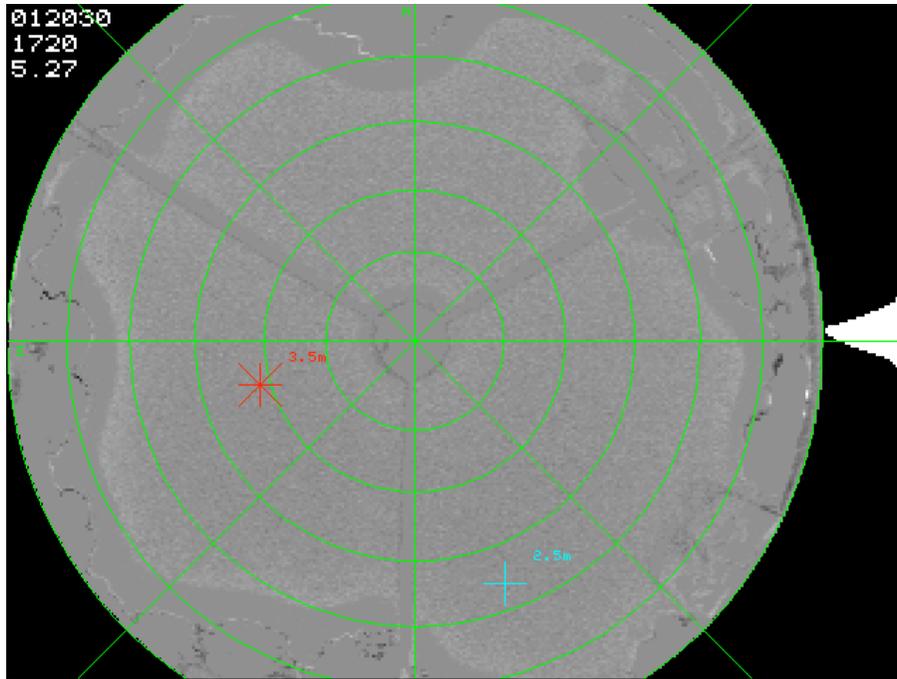


Fig. 3: A processed all-sky image from the APO IRSC under photometric conditions.

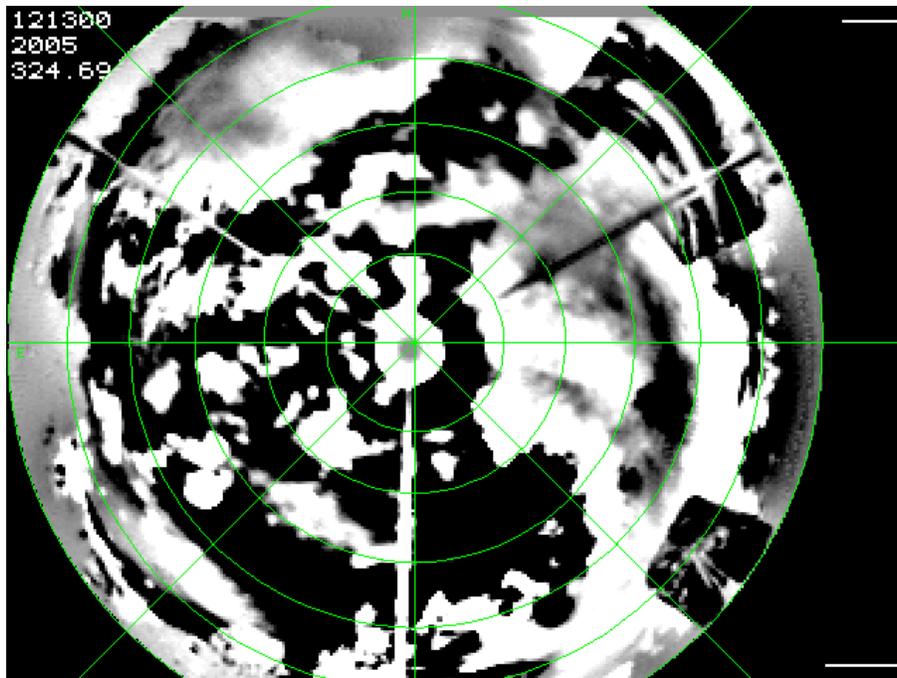


Fig. 4: A processed all-sky image from the APO IRSC under non-photometric conditions.

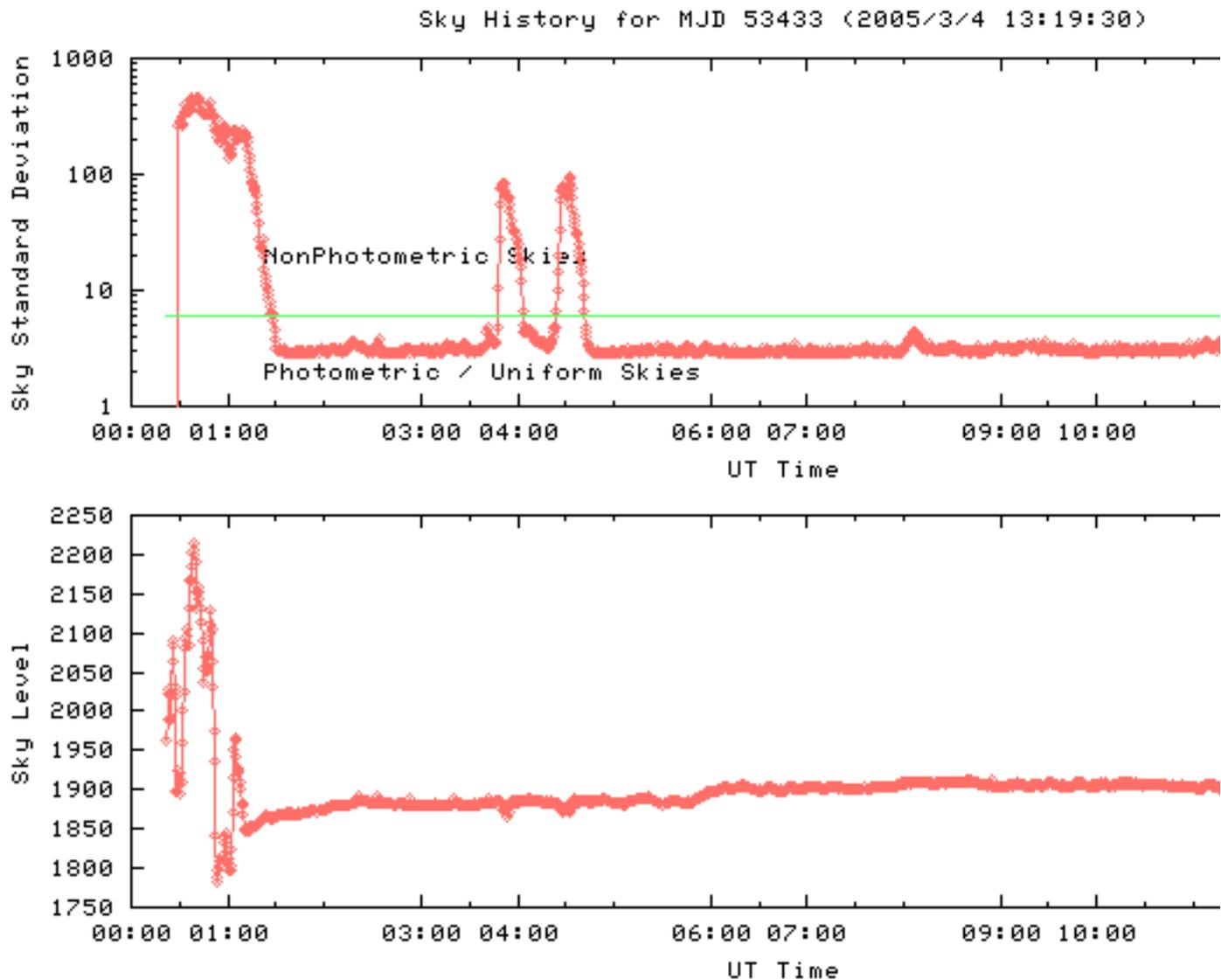


Fig. 5: The standard deviation of the pixel values of the all-sky image vs. time (top) and the sky level vs. time (bottom) for the night of MJD 53433. The horizontal green line in the upper plot indicates the empirically determined value separating photometric and non-photometric conditions. An mpeg movie of the cleaned all-sky images for this night can be found at <http://hoggpt.apo.nmsu.edu/irsc/movies/53411.mpg>.