Contents

List of Figures

List of Tables

Part I

Overview and Summary

Chapter 1

Introduction and Overview

1.1 Highlights of CDR Document

This section provides an overview of the CDR document which consists of the following five parts:

- **I. Introduction and Overview**
- **II. Subsystem Designs**
- **III. DEIMOS Calibration**
- **IV. Flexure Compensation**
- **V. Budget and Schedule**

Since the document is very long, we list here the sections to which we would particularly like to direct the review board's attention. We hope that all board members will have a chance to read these essential sections prior to the CDR.

- **Part I, Appendix A:** Contains an updated and detailed response to the report of the DEIMOS Software PDR Review Board. It describes the specific actions we have taken in response to the board's recommendations and provides a comparison of projected versus actual progress over the interval between the PDR and CDR. **This appendix also highlights several critical personnel and management issues which urgently need to be resolved; see I.A.4.2.** A summary of this appendix is provided in section I.2.1.

- **Part I, Chapter 3:** Describes those areas of the DEIMOS system in which we have a solid proof of concept for the underlying software and/or hardware components. In many cases, these proofs are in the form of fully functional software tools (operating within a well-defined software infrastructure) that have been successfully used to complete and document the software designs presented in this volume.

- **Part I, Chapter 4:** Describes areas of risk for which neither a detailed design nor proof of concept yet exists. Contingency plans for the Image Display subsystem are discussed, and an abbreviated set of functional requirements for the Flexure Compensation and Calibration subsystems is provided. (A more detailed discussion of the Calibration and Flexure Compensation subsystems is provided in Parts III and IV, respectively.)

- **Part II, Chapter 1:** Describes the design of the software and operating procedures for the slitmask design and fabrication subsystem. This design has evolved significantly from what was presented at the PDR. Since the inventory and production control aspects of this subsystem are ones that we have not faced in previous instruments, these aspects should be afforded careful review.

- **Part II, Chapter 7:** Describes the information management (database) subsystem, which is integral both to the operation of the overall instrument and to the design and documentation of all of the DEIMOS subsystems (including itself). This chapter addresses various concerns about this subsystem that were raised at the PDR, including the concern that the database component *per se* will delay the project schedule or consume more than its share of resources. This chapter illustrates that the database subsystem now has a more limited and well-defined scope and that so far it appears to be ahead of the other major instrument subsystems: the design of this subsystem is nearly complete, the implementation of most of its components is quite advanced, and many of its components are already operational. The various documentation products that are automatically generated by this subsystem offer solid proof of the utility of its design and its cost effectiveness.

- **Part V:** Describes the DEIMOS software budget and schedule. This part was not ready in time for the pre-CDR distribution of documents, but will be presented at the CDR. A significant block of time is being allocated for its presentation and subsequent discussion at the CDR.

The remaining portions of this CDR document are less important. In particular, parts III and IV consist of supplementary material:

- **Part III** is intended to become a separate document on DEIMOS Calibration, but it is not yet complete. Its contents are summarized in section I.4.3 and it is included **in draft form for reference only**. It defines the procedures and mathematical relationships required for the calibration of DEIMOS data but does not specifically address the detailed design of the calibration software. Accordingly, Part III is a preliminary step in defining the functional requirements for that subsystem. (Since implementation of pipeline data reduction is a goal rather than a commitment, it will likely not occur until much closer to or even after instrument commissioning. Recall that the DEIMOS software effort is budgeted to continue for 2 years beyond first delivery.) A complete detailed software design of this subsystem will not commence until later in the project. However, the material in Part III indicates how well the requirements of this subsystem are understood, and the magnitude of the design and implementation task which remains. An understanding of these points is needed to assess the budget and schedule impacts of implementing pipeline data reduction.

- **Part IV** is intended to become a separate document on the Flexure Compensation (FC) Subsystem, but it also is not yet complete. Its contents are summarized in section I.4.2 and it is included **in draft form** as part of the CDR document to serve as a preliminary set of functional requirements for this subsystem. However, the FC subsystem is a required part of the instrument, so the detailed design of its software must proceed now. To date, the overall FC system (optical, mechanical, electronic, and software) has not been subject to any detailed independent review. This software CDR represented the first opportunity to review much of the material contained in Part IV, and while the completed portions of this FC document do not specifically focus on software, they do indicate the level to which the requirements of this subsystem are understood, and the magnitude of the design and implementation task which remains. We recommend that the overall FC system be reviewed at a later date.

In addition to the five parts of this CDR document, we have also produced a preliminary draft of *Lick Observatory Technical Report 80: The DEIMOS Keyword and Database Dictionary*. This report (commonly referred to as LOTR 80 or "The Dictionary") provides a dictionary defining all of the DEIMOS keywords and database tables. Example pages from this dictionary are included in section I.3.4 of this document. However, due to its large size, we are not including LOTR 80 as part of the pre-CDR distribution of documents to the review board. Copies of LOTR 80 will be available for inspection at the CDR. An on-line copy of LOTR 80 will be available on or before June 13 via the WWW at the following URL:

http://www.ucolick.org/ deimos/lotr80/lotr.html

For those reviewing this dictionary, remember that it is automatically generated from the data contained within the various tables of the DEIMOS database, using the document generation tools which are described in section I.7.1.1. Not all data are complete, but the dictionary currently represents approximately 80% of its finished equivalent.

1.2 Overall System Summary

This section provides a global overview of the DEIMOS software system. It includes two diagrams showing all of the major subsystems and their interconnectivity (see Figures 1.1 and 1.2).

The DEIMOS system consists of the following major software subsystems, which are described in the indicated sections of this report.

- Mask Design: Part II, Chapter 1
- Mast Fabrication: Part II, Chapter 1
- DataBase: Part II, Chapter 7
- Mask Management: Part II, Chapter 1, Section 5
- Spectrograph Control: Part II, Chapter 3
- Header Collection: Part II, Chapter 4 (handouts at CDR)
- Science Data Capture: Part II, Chapter 4 (handouts at CDR)
- Image Display: Part II, Chapter 6
- STB: Existing system at Keck
- Pipeline Reduction: Part III
- Mask Alignment: Part II, Chapter 2
- Detector Control: Part II, Chapter 4 (handouts at CDR)

- DCS and Guider: Existing system at Keck
- Rotator Control: follows model of HIRES rotator (existing system)
- Flexure Compensation: Part IV
- Documentation Facility: Part II, Chapter 7
- Calibrate Spectrograph: Part III
- Logging Facility (KTLwatch): Part II, Chapter 8
- Engineering Data Visualization: Part I, Chapter 3, Section 2
- KTL User Tools: Part I, Chapter 3, Section 2, plus existing KTL UIs

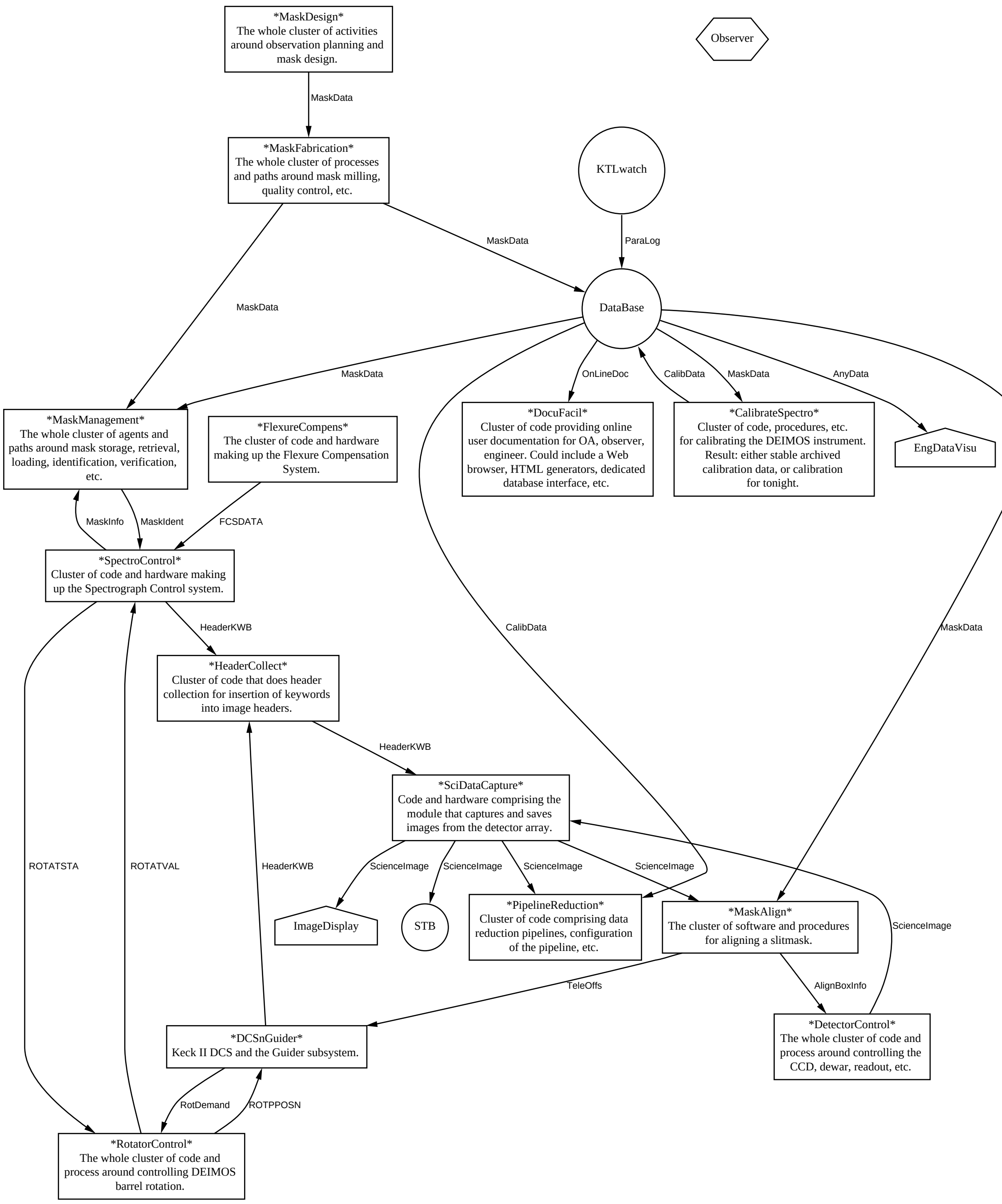


Figure 1.1: Overall system data flows

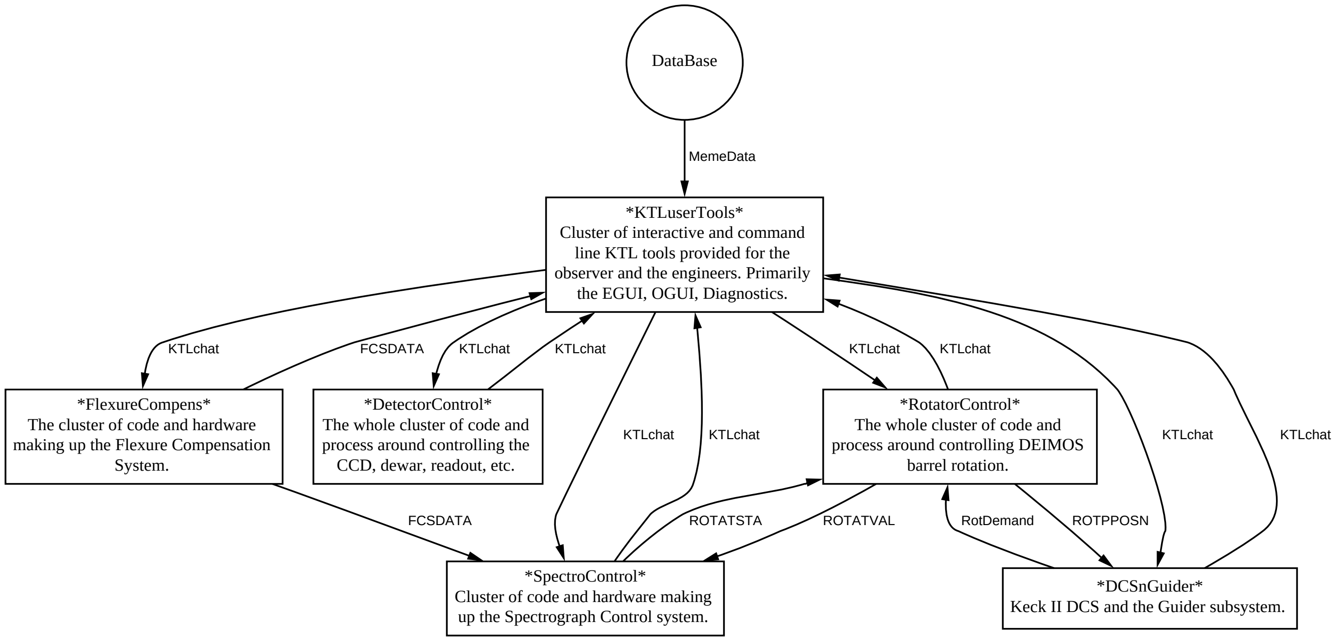


Figure 1.2: KTL User Tools data flows

1.3 Changes to the Instrument

The basic optical properties and capabilities of DEIMOS were described in Chapter 2 of the Software PDR Report presented on March 22, 1996. We list here the significant changes that have occurred since then that are relevant to software development.

- The final choice of CCDs has not been made, but the probable properties are better known, as discussed at the recent Detector/Controller CDR held on May 20, 1997. Copies of that report are available. A summary is given here in Part I, Chapter 2.7.

- The slitmasks are now flat sheets of aluminum as carried in the slitmask cassette holder and are deformed to the curvature of the focal plane when inserted. The curved rigid holders have been eliminated. As a result we can now carry 12 slitmasks plus one longslit, an increase of 3 slots. Mounting and handling will be greatly simplified.

- The TV CCD is a Photometrics 1K $\times$ 1K SITe chip with 24 μ pixels, a scale of $0\farcs2$ px^{-1}, and a total FOV of $3\farcm3$. The Photometrics camera looks at the telescope focal plane in staring mode. The visible focal plane is divided into two parts. One area is covered by a curved mirror permanently mounted in the telescope focal plane beside the slitmask area. This mirror is always visible regardless of whether the slitmask is in or out and provides offset guider capability when objects are not visible on the slits (e.g., during direct imaging). The dimensions of the offset guider area are $1\farcm3 \times 3\farcm3$. The remaining area of the TV FOV images either the cylindrical slitmask or a spherical mirror with a long slit, depending on which is inserted. The long slit is located $4\farcm5$ off axis from the center of the telescope focal plane. The dimensions of the imaged slit area are $2\farcm0\times3\farcm3$. The entire long slit area is unvignetted in the TV by virtue of its spherically curved mirror. Because the slitmask is cylindrical rather than spherical, objects will be visible only in a strip approximately $30''$ wide surrounding the longslit. However, the top surface of the masks is reflective, and it should be possible to place objects directly into the slitlets using the TV.

- The grating slide still has slots for the imaging flat mirror plus three gratings. However, there is now room for only one 8 $\times$ 12-in grating (in Position 1), as Position 3 has been trimmed to accept only 6 $\times$ 8-in gratings to simplfy its encoder design.

- The option of tilting all grating positions to the flat mirror imaging angle has been retained, as it appears likely that notch mirrors for narrow-band imaging can be fabricated, giving DEIMOS a narrow-band imaging capability that is otherwise not possible. Extra interlock hardware and software is needed to prevent collisions when grating slots are tilted to the mirror positions. ??True?

- The stated image stability goal of 0.25 px rms in each coordinate may not be achieved despite the Flexure Compensation system owing to thermal changes in the camera plate scale. These may cause image motions (contraction, expansion) of a few pixels at the edge of the detector if not successfully compensated passively in the thermal design of the camera lens barrel. Residual motions of this magnitude would have adverse implications for flat-field accuracy and wavelength stability, as described in Part I, Chapter 4.3 and Part III of this report.

- The slitmask milling machine has been received and debugged. It operates successfully.

- The SDSU-2 controller has been received and tested. It meets our needs, as described in the recent Detector/Controller CDR, held on May 20, 1997. Copies of that report are available.

- A barcode reader has been added inside the spectrograph to read the IDs of slitmasks after the masks are loaded. These IDs are checked against plan and then loaded into the instrument control program and database.

- Remote observing from Keck Headquarters at Waimea has been demonstrated using LRIS in multislit mode and is highly efficient, including slitmask alignment. Our default plan for DEIMOS after instrument commissioning is to observe remotely from Headquarters.

Chapter 2

Summary of Changes from the PDR

2.1 Highlights of PDR Response

- DEIMOS will use the second-generation SDSU CCD controller hardware.
- NOAO "PIE" server is planned for final mosaic image display server.
- Keck Figdisp will be used as image display server during initial engineering and as fallback for commissioning.
- The role of the database component has been more clearly defined.
- Lick is starting a collaboration with CARA on developing a new guider.
- LateX was used for preparing the CDR documents.
- An insufficient fraction of Kibrick's time has been available for the DEIMOS software effort.
- While excellent progress has been made on developing individual subsystems, the coordination of these development efforts has been insufficient.
- Current software staffing levels and management structure need reexamination.
- Office space for additional DEIMOS software staff is a serious problem.

- Investment in software infrastructure required more effort than originally planned, but has paid off.
- An overall subsystem diagram has been produced.
- CVS has been adopted both for code and document managament.
- Plans for configuration and release management are currently being negotiated with CARA.
- DEIMOS will use the music/traffic model and not switch to EPICS.
- FITS WCS proposals have been revised to conform to emerging standards.
- Initial CCD controller implementation will provide for single-amplifier readout but provide simple upgrade path to dual-amplifier readout.
- As recommended, significant effort was put into prototyping user interfaces.
- Prospective DEIMOS users were surveyed regarding IRAF and IDL, with no clear consensus for a switch to IDL
- The CCD VME crates will be located in the computer room and not at Nasmyth focus.

2.2 Highlights of Chapter Changes to PDR

- Slit masks are no longer mounted in frames.
- Added new module, "MillControl" to interface to NC mill.
- Slitlets are specified by four corners.
- Plans for both image storage and display have changed since PDR.
- The scope of the database effort has been significantly reduced and focused on observing and engineering support
- Our plans for document generation and the practical application of a central database of keyword information have been realized
- We have dropped consideration of the guider from the CDR

- There have been no changes to the slitmask alignment procedure
- Due to delays in delivery of science CCDS, we are considering fabrication of an interim mosaic.
- Production of the science CCD mosaic detector is now the pacing item on the critical path.

2.3 Slitmasks (PDR Chap. 3)

As noted in Chapter 1.1, hardware design has changed significantly since the PDR. Slitmasks are no longer mounted in frames, simplifying the required slitmask handling/holding facilities, and the number of masks that can be loaded in the instrument has increased from 10 to 13 per barrel.

There have been no major changes in software since the PDR. Much of the slitmask fabrication process has been tested (see I.3.1). Detailed procedures for slitmask fabrication and handling have been developed (see II.1). We have identified the need for an additional software module ("MillControl") to interface with the NC mill and the database during slitmask milling and check-in precedures.

Outstanding issues listed in the PDR have been settled. Slitlets will be specified by a center, length, width and angle, which defines a parallelogram. (More complex shapes, such as arcs, can be constructed by multiple slitlets.) The "mask design" and method of ingestion into the database has been decided. Mask design now consists of target objects ("Objects"), slits on the sky ("MaskDesign"), a physical description of the mask ("Blueprint"), and a mapping of target objects to slitlets ("SlitObjMap"). These four tables are packaged in a single FITS file (the "MaskFile") that is sent to Keck for database ingestion and mask fabrication.

2.4 Mosaic Image Handling

Plans for both storage and display of DEIMOS images have changed since the PDR. Differences in the manner of storage are described below and detailed in I.3.6 and I.3.7. Differences in the display are described here and detailed in I.4.1 and II.6.

2.4.1 Mosaic Image Storage

Chapter 7 of the DEIMOS software PDR contains detailed specifications of image storage formats. Some standard practices for handling mosaic images have been emerging from the astronomical community. These practices were not present at the time of the PDR, and some aspects of mosaic image storage have been modified to conform better.

In the original design we proposed that the sections of image from all CCDs in the mosaic could be placed into a single FITS image array. The principal advantage of this scheme is that the images can be displayed by existing astronomical image display clients. The disadvantage is that it interleaves sections of exposed pixels with calibration data.

Emerging consensus in the FITS community dictates that archival storage of raw image data is best done by placing the image sections from each CCD amplifier into a separate FITS HDU. This scheme is in use by the NOAO mosaic detector. (IRAF has already been modified to process FITS files containing multiple IMAGE extensions of such single-amplifier data.)

In the FITS community there is less consensus about the archival storage of "pipeline-calibrated" image data. If the DEIMOS CCD controllers are read using dual-amplifier mode then the calibrated data from a single CCD will be stored in a single FITS image array. This greatly facilitates astrometric and photometric analysis.

For the purposes of quick-look analysis the image frames may be written into a single data array which has approximately correct geometry. We do not intend to describe any form of archival storage for such images.

The PDR also specified that table extensions containing slitmask data would be appended to the image. The procedures and data requirements for slitmask fabrication and observation have been determined. The details of these tables can be found in Part I Section 3.6.

2.4.2 Mosaic Image Display

Chapter 8 of the DEIMOS software PDR contains detailed requirements for the quick-look image display. There has been no significant change in the requirements.

At the PDR De Clarke demonstrated a modified version of the ESO RTD program as a possible prototype for the DEIMOS guider and quick look image display. Subsequent testing of ESO RTD by Richard Stover revealed some problems that the ESO RTD development team was reluctant to fix.

The current DEIMOS display plan is a two-phase process. Initial laboratory testing of the DEIMOS mosaic will use a slightly modified version of the existing Keck Figdisp. As the NOAO Real Time Display becomes available for initial testing we intend to collaborate with them on the development of features needed for DEIMOS. Details are documented in section I.4.1.

2.5 Revised Database Scope

Since the PDR we have reduced the scope of our database effort to one tightly focused on observing and engineering support. We no longer plan to tackle the problem of public accessibility to, or publishing of, DEIMOS science data; that problem is beyond the scope of the instrument project.

We have more faith that STB as installed at K1 and K2 will be adequate for archiving DEIMOS images, and that the database of images headers at CARA will be accessible to us for engineering and observer support. No separate DEIMOS archiving system is currently proposed. Although STB has not been tested with images as large as those DEIMOS will produce, we have no reason to suspect that it will fail due to image size.

We have designed schema for 90 percent of the functionality we expect from the DEIMOS database component (see Part II Chapter 7 for details), and defined the operational role of the database in such a way as to provide a layer of separation from vendor-specific database server interfaces. The detailed schema design is documented in Lick Observatory Technical Report (LOTR) number 80, copies of which will be available at the CDR.

Our plans for highly-developed Web interfaces to DEIMOS data have been scaled back, since we no longer propose to support widespread public access. We will direct our efforts instead to engineering analysis tools and comprehensive instrument performance logging (by means of existing KTL services), friendly online documentation, and semi-automated data reduction.

Our plans for document generation and the practical application of a central database of keyword information have already been realized. Large chunks of the CDR documentation are generated rather than manually drawn or written, and can be regenerated as design changes occur. We have early prototype code for source code generation as well. The database of DEIMOS keywords is about 80 percent complete and is documented in LOTR 80, aka "The Dictionary".

The only areas which remain ill-defined at this point are the instrument's requirement for calibration data, and the exact functioning of the FCS. We have made headway on function definition here, but no concrete schema design has yet taken place because the hardware and procedures are still unsettled. Some preliminary schema design is documented, but should not be regarded as final or even firm.

2.6 Guider System & Object Acquisition (PDR Chap. 5)

Upon the recommendation of the PDR panel, we have dropped consideration of the Guider. However, we would like to list some desired features for the guider which would aid DEIMOS observations and efficiency:

- The guider image display should include flexible contrast and gain control, for image quality assessment;
- Automated image capture in selectible image formats;
- Ability to coadd guider frames for improved S/N;
- Ability to monitor focus;
- "Handpaddle" capability for positioning and focus;
- Differential guiding, that is, maintaing a guide star position with respect to a reference mark in the image;
- Monitoring image FWHM, integrated counts, pixel location, and background level, with alarms if any of these change drastically.
- Several options for determining image position (eg., centroid, edge detection, cross-correlation).
- Ability to coadd guider images into an integrated "science" frame, which can be used for calibration and intepretation purposes (eg., "what was the average seeing and transparency? Exactly where did the longslit fall on the object? What were the estimated slit losses?").

There have been no changes to the slitmask alignment procedure. This procedure has had an additional year of field-testing with LRIS, with no reported problems.

2.7 CCD Status

The specifications for the DEIMOS mosaic CCDs are shown in Table 2.1 The current status of CCD fabrication and availability was extensively discussed at the Detector/Controller CDR on May 20, 1997 (copies of those documents are available). The consensus at the review was that DEIMOS would ultimately be able to obtain CCDs that met or exceeded all the specifications in Table 2.1, but not on the originally planned timetable. We are currently discussing the advisability of fabricating an interim detector, and the results of that discussion will be presented at the Software CDR.

Production of the DEIMOS mosaic CCD detector is currently the pacing item on the critical path. Figure 2.1 shows a success-oriented schedule involving a single detector (i.e., no interim detector). Instrument delivery is already delayed four months beyond our original schedule despite the success-oriented approach. This schedule calls for delivery of the main mosaic CCDs by January 1, 1998. None of the development paths outlined below can be depended upon to deliver CCDs on such a tight schedule, hence our consideration of an interim detector.

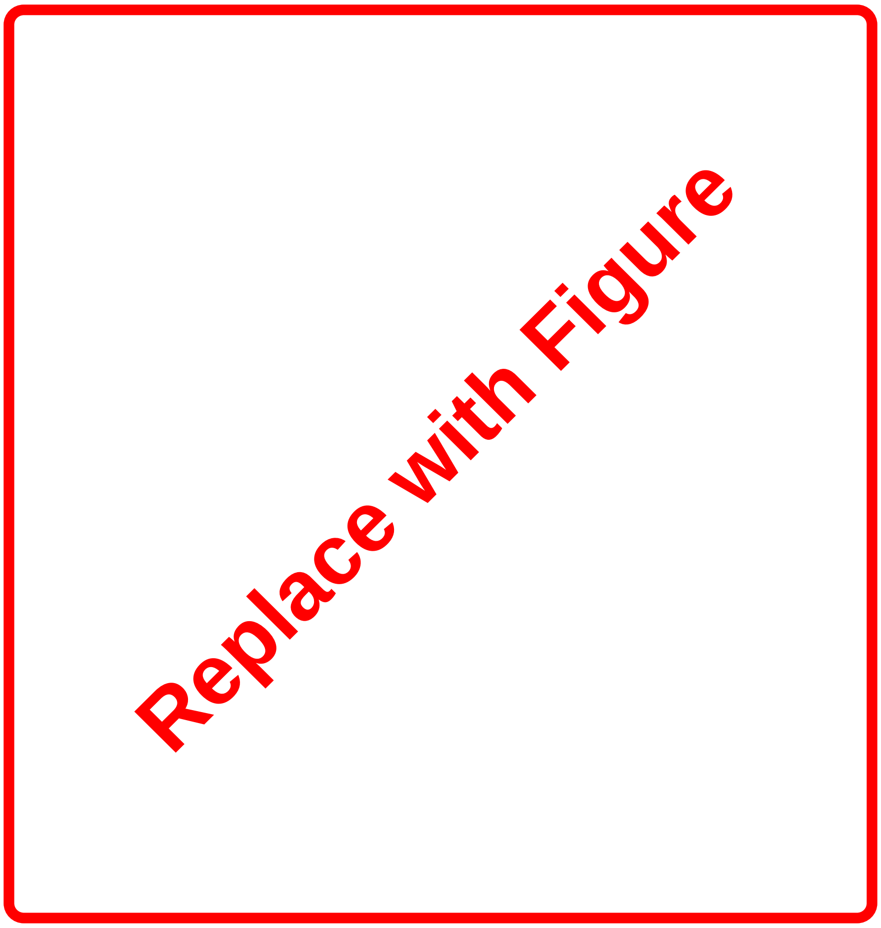
The following is a brief summary of the status of three potential delivery paths for the DEIMOS mosaic CCDs.

2.7.1 MIT Lincoln Laboratory

CARA entered a consortium with five other observatories to co-fund the development at MIT Lincoln Laboratories of a 2K $\times$ 4K 3-side buttable sensor meeting the specifications given in Table 2.1. Thinned and packaged CCDs from the first round are just now being tested. The sensors are of two types — a normal 20-μm thick epitaxial device and a thicker 40-μm thick "deep depletion" device built on high-resistivity silicon. The latter has the advantage of pushing the onset of fringing from 7000 Å out to 8000 Å. Red QE is also increased, being over 80% from 6000 Å to 9000 Å. The 20-μm epitaxial devices from the same round also appear to be excellent performers for their type.

DEIMOS expects to receive about six usable devices from this first development round. However, before sinking all or part of our CCD budget into a second round, several technical problems would need to be solved:

(1) The MITLL production line is switching from 4-in to 6-in wafers, with the result that the new mask design brings the corners of the CCDs to within 4 mm of the wafer edge. This necessitates a whole new method of


Replace with Figure

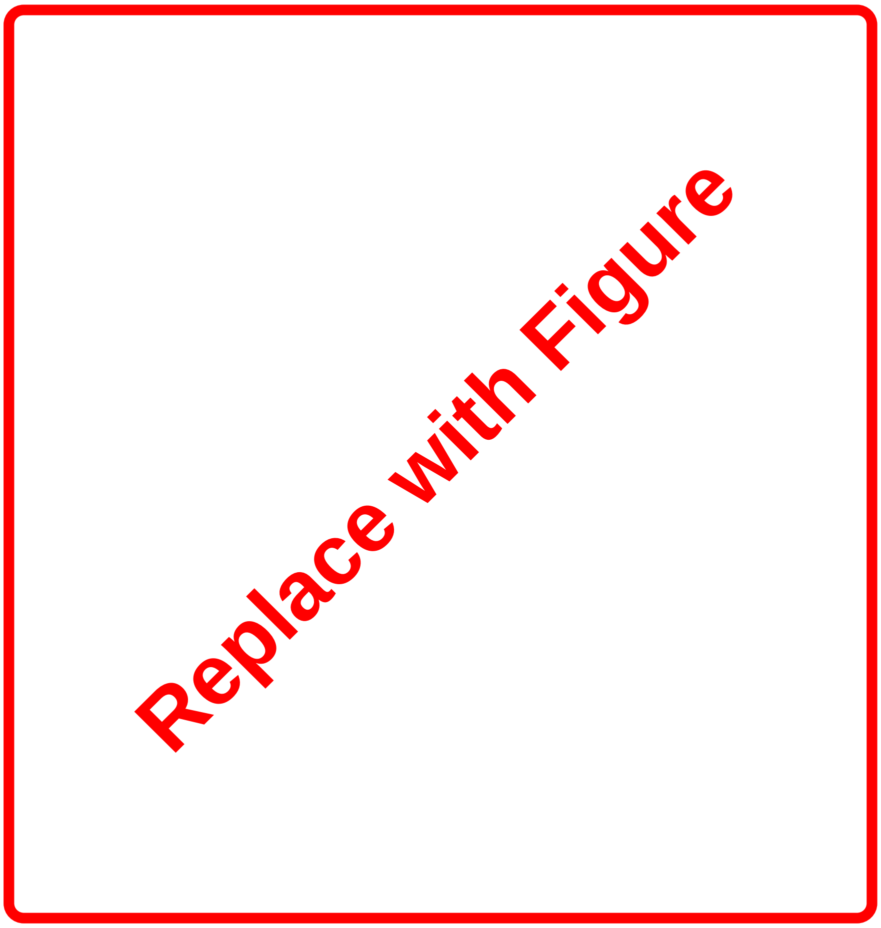


Figure 2.1: Current Schedule

thinning and may introduce defects into the portions of the CCDs near the edge.

(2) A new blue-sensitive two-layer anti-reflection coating needs to be developed to increase QE in the blue. This might require coating by an independent contractor such as Mike Lesser, complicating the production process.

(3) MITLL uses a laser-annealing process with boron implant to passivate the back surface. The laser heats the back surface unevenly, leaving a "brick-like" pattern of QE variations with peak-to-valley amplitude of 15% at 4000 Å. Since the amplitude of this pattern is likely to be steeply color-dependent, flat-fielding direct images with objects of widely varying color could prove impossible in B, and perhaps also in V. This is unacceptable and must be reduced in amplitude by at least a factor of three, which might require major alterations to the laser.

(4) High-resistivity 40-μm thick devices demand much higher-quality silicon wafers and clean-room conditions. More testing of high-resistivity wafers is needed before a large-scale run of these devices could be embarked on.

Testing of these items is underway, but results will not be known until mid-September at the earliest. Should tests be successful, another 8 months at minimum would be needed to deliver DEIMOS CCDs, pushing the delivery of CCDs to mid-May and the final delivery date 5 months beyond that shown in the success-oriented schedule in Figure 2.1.

2.7.2 SITe

We recently tested a very good SITe 2K $\times$ 4K CCD. It meets our requirements except that QE in the blue and visual is low (40% at 4000 Å, 60% at 6000 Å). SITe has accepted orders for several tens of astronomical CCDs but is having trouble meeting production schedules. It is not yet delivering devices routinely. We have so far been unable to get them to quote a firm delivery date on DEIMOS CCDs.

2.7.3 EEV

EEV has two 3-side buttable formats. One has 13.5-μm pixels in a 2048 $\times$ 4500 format. This size would provide the same spectral coverage as our 15-μm px DEIMOS specification but would cover 10% less slit length. EEV is taking orders for delivery of this device by the end of 1997 but none in the full size has as yet been tested by American astronomers. Smaller devices with the same amplifier have been tested by Geary at CfA and are said to meet the DEIMOS specifications in Table 2.1. Adoption of the 13.5-μm format would make it awkward to mix these devices with those from other vendors, locking us into one supplier.

EEV are also developing a standard 2K $\times$ 4K 15-μm-px device for ESO. Its development is being pursued by several European observatories.

The lead time for either kind of device is said to be 12 months, and thus is comparable to the most optimistic delivery time for MIT Lincoln devices.

2.7.4 Summary

The salient conclusions from the Detector/Controller CDR are that:

(1) we should continue to develop software to operate an 8K $\times$ 8K focal plane with specifications as originally planned,

(2) the delivery schedule of the final CCDs is highly uncertain and likely to be delayed at least 6 months beyond the success-oriented schedule in Figure 2.1, and

(3) it may be desirable to fabricate an interim detector to keep the project moving along. Such an interim detector could be constructed of thick CCDs from our in-house Orbit development program at Lick, perhaps including some engineering-grade devices from the first round of CCDs from MIT Lincoln.

Chapter 3

Prerequisites: Items with Proofs of Concept

3.1 Slitmasks Fabrication & Handling

There has been considerable progress in the area of slitmask fabrication and handling since the time of the PDR. First, the NC mill with 60,000 RPM head has been purchased by the project; this mill has been successfully used to fabricate LRIS slitmasks which were used for science observations. DEIMOS test masks have also been milled, held in place by the large-size magnetic chuck intended for this use. An identical second mill was purchased by CARA for immediate use in fabricating LRIS masks and later use for DEIMOS masks after DEIMOS commissioning (as of late July 1997, this mill is operational.)

Second, the expected method of slitlet milling has been tested: four corners are specified for each slitlet. The mill tool makes an initial pass to remove the bulk of material, and then cuts the actual slitlet edges. This appears to produce adequately clean and uniform edges.

Third, barcode scanners have succesfully read barcodes attached to mask stock. The scanners can reliably read the barcode labels at fairly steep angles, permitting the spacing of masks in the cassette to be reduced (and the number of masks in the cassette increased) accordingly.

Fourth, a detailed procedure for mask handling, particularly for avoiding misidentification between physical masks and mask designs, has been produced (see Part II, Chapter 1). A proposal to mill an identifying barcode into the mask stock was tested and rejected, both because the milling was highly inefficient and because the resulting barcode could not be read reliably by the barcode scanners.

Lastly, the overall feasibily of software for mask design through fabrication has been verified. This software was written for LRIS masks, but the software should be easily adaptable for DEIMOS masks. The mathematical mapping of sky positions onto the tilted cylinder of the slitmask (see Part III) has been applied to an LRIS mask and produced residuals no larger than $\sim$0.1 arcsec (the *rms* errors were much smaller); this means empirically-determined corrections to the sky-slitmask mapping should be small, of order 0.1 arcsec or less.

One unresolved item that arose is that the slitmask milling produces small chips of metal that must be cleaned out of the slitlets. The method for doing this has not been identified, but limited testing with ultrasonic cleaners have shown excellent results.

Another identified concern is that the positional accuracy of the mill is of order 0.0003 inch, corresponding to about 0.010 arcsec at the telescope focus or 0.1 pixel at the DEIMOS detector. Reduction software will need to account for slit nonuniformities at this level, if such accuracy is required.

3.2 GUI and GUI design : the "dashboard" application

During early planning for the DEIMOS instrument, the software team wanted some means of managing the large number of new (or slightly variant) keywords the instrument would need. We constructed a relational database schema for modelling FITS keywords, storing keyword attributes such as datatype, format, semantics, etc. The schema rapidly grew in complexity, incorporating new concepts such as interkeyword syntactic and semantic relationships, hierarchical keyword grouping, etc. See Part II, Chapter 7 ("Information Management") for more details on the database design and function.

Since any graphical user interface to KTL-based instruments can be seen as a visual representation of a collection of keywords, the keyword database became a valuable resource for developing our various instrument GUI.

3.2.1 The "Dashboard" application: a soft realtime GUI

A complete description of the "Dashboard" philosophy, design, and feature set will be found in Attachment I [1]. In brief, the Dashboard interface is a flexible, "soft" interface to any KTL service. It makes use of KTL keyword libraries and of the Ktcl extension written by W. F. Lupton (CARA). Any supported keyword can be represented on the dashboard surface by a variety of traditional "meters" (i.e. X11 representations of a gas gauge, dial gauge, strip chart, function light, push button, etc.).

The meters can be repositioned freely, and all their X11 attributes can be edited. The attributes can also be made to change in response to changing values or conditions of KTL keywords. Boolean expressions involving several keywords at once can be evaluated and used to trigger appearance changes, or to deliver alarms. The designer can also invent "pseudo-keywords" as expressions involving KTL keywords, constants, and global variables. There is no distinction between editing the application and running it; the running application can be edited and redesigned while connected to a live KTL source. This reduces the turnaround time for testing new features to approximately zero.

A variety of graphical constructs such as lines, arrows, rectangles, circles, text, bitmaps, etc. can be placed on the dashboard surface for decoration and visual organization. These also can be reconfigured, resized, and made

[1] "Dashboard: a Knowledge-Based Control System Interface" by De Clarke, to be presented at the 1997 TCL Conference.)

to respond to KTL conditions. Text is a valid graphical object and can be used to convey messages, warnings, status, etc. as well as to label areas or components of the UI.

The application is layered, that is, it is possible to pop up “sub dashboards” by double-clicking on any graphical object. This metaphor is intended to support levels of lesser and greater detail, but can be used arbitrarily to create any type of popup screen.

Though the Dashboard in designer mode is extremely fluid (the user having the option to reposition, reconfigure, delete and create) it also runs in a “production” mode in which the user has no such power. Our plan is to design the DEIMOS interface using the flexible “GUI builder” mode, then ship a “frozen,” stable, fully-tested UI with the instrument for observer use. CARA and Lick engineers, as well as observers, could run the “designer mode” Dashboard at any time to invent their own KTL control panels, but the canonical DEIMOS UI would remain unaltered except by official releases from the DEIMOS project engineers. We should note that the Dashboard also supports “No Writes” mode, in which the user can see KTL keyword values but is not permitted to do KTL writes, and a “No KTL” mode in which development can continue in the absence of a working instrument, telescope, etc.

3.2.2 Proof of Concept

A prototype interface has been developed using the HIRES instrument; this interface, while functionally not more than 50% complete, demonstrates and tests most features of the application. The “Dashboard” application will be used to develop interfaces for the Lick Prime Focus Camera (PFCAM) and Keck ESI instruments, which ship earlier than DEIMOS; DEIMOS will benefit from the experience gained during those commissionings, just as the other two instruments have benefitted from DEIMOS investment in the keyword database and GUI development project.

Astronomer response to the “Dashboard” product has been positive so far. We anticipate a good level of acceptance by users, but we also know that it is relatively trivial to alter the appearance and function of the finished product to resolve user complaints and new feature requests. We also anticipate that CARA and Lick engineers will find the “roll your own” aspects of the tool very useful when setting up tests and diagnostics. Dashboard’s logging and plotting features make it a handy “logic analyzer” for KTL-based systems.

We have inserted a colour page into this chapter showing two generations in the evolution of the DEIMOS UI sketch. The earlier generation (upper half of the page) shows a colour-coded version in which each major stage in the light path was represented by a pastel box. The later generation, redesigned by Faber, Phillips, and Clarke, shows a less "busy" colour scheme, but similar graphical design. The philosophy is to show an effective light path through the instrument as a simple progression from left to right. The UI should provide obbvious graphical indication of occultation, so that the observer can see easily whether light is reaching the detector, and from what source – but at the same time, the front end should remain very simple, so that the observer can interact (when all goes well) with a clean and uncluttered interface that does not occupy the entire workstation screen. The "long and skinny" look of the DEIMOS prototype is deliberate; at some point we may need to display UI for both beams on one screen.

The second image of each version shows an interface state in which the user has changed a desired value (bottom half of the graphical representation of each stage) from the current value. The UI has lit up that area in yellow to indicate the discrepancy between Actual and Desired. When the GO button is pushed, and stages move to their new positions, the yellow highlighting will become orange or red, and will not return to its normal colour until the stage motion is complete. Many other graphical and text display strategies can be used to indicate intent, motion, and completion, but after many discussions the DEIMOS PI and software team are leaning strongly towards the model shown here.

The dashboard can save a current KTL setup for future inload. We are debating the desirability of loading KTL setups from FITS image headers (i.e. "Set the instrument up the way it was when the image in this FITS file was taken.") It also has a scripting capability, offering the user an edit window for writing KTL control scripts.

Double clicking on any stage will pop up a more detailed interface for that stage, and those sub-dashboards may also contain popups for extreme levels of detail. However, once the instrument is calibrated, regular science datataking can be done entirely from the simple top level interface.

The Dashboard derives its knowledge about KTL keywords from the authoritative database; no keyword knowledge is hardwired into the application. It can cache the downloaded keyword data in a local ASCII file (in accordance with the DEIMOS project philosophy of imposing a layer of separation between database server and operational applications); but the cache can be refreshed at any time from the database server, and any changes of keyword definition and usage will then be known to the UI.

Context-sensitive help is built in automatically. By means of a Control-click the user will be presented with a complete description of the KTL keyword being monitored by any given meter. Also, with a slightly different mouse click (button 1 instead of 2) the meters will give the user hints on how to interact with them. Information about "how to" use the mouse to get to these help facilities will of course be available from a visible menu, for the first-time or casual user.

Our experience with prototyping, to date, has been encouraging. It takes very little time to develop a dashboard panel, and each panel can be revised rapidly and easily as our design discussions progress. The Dashboard is based entirely on free and portable code, so it runs with equal facility on Digital Unix, Solaris, Linux (Dashboard on a Linux laptop is an attractive portable engineering tool), etc. Our 50% complete HIRES interface, which has been tested extensively against the running instrument, took less than 2 days to construct. We are fairly confident that UI design will not be a major consumer of DEIMOS programming resources, and that we will be able to respond quickly to initial user feedback and instrument changes.

In summary, we feel that we have substantive proof of concept for our GUI design philosophy, approach and toolkit.

3.3 Software Design and Documentation Tools

3.3.1 Exploiting an Existing Resource

Every project of this complexity needs a way to document design and process. Besides the usual reams of textual documentation, drawings and diagrams are needed for easier comprehension of highly complex systems. In our case, these systems can be seen in terms of information passing through various handlers on its way to a final storage medium.

Having already established the value of the online authoritative database of keywords and made a commitment to creating and maintaining it, we are in possession of a valuable resource for software and system design. Since the keywords (or more generally, "memes") in the database represent information, we have already modelled half of the problem. The other half is the list of "agent" or processing nodes through which information passes.

A keyword like TEMPDET, for example, can be seen as a piece of information (the detector temperature) which originates with a hardware sensor, is read via an ADC on a board in a VME crate, then is held in the crate's memory pending broadcast or on-demand transmission of this parameter to KTL clients. The keyword eventually ends up in a FITS header, captured at end of exposure, and the FITS header ends up on an Exabyte tape. The astronomer takes the tape home, and (from our point of view) the story ends there.

We could vastly simplify this chain of events, or we could document it in excruciating detail; but a piece of information certainly does "move" through our hardware and software system on its way to an archival storage medium, and this motion can be modelled using the database.

We established a schema for describing information flow. One element of this schema is the table of Agents. An Agent is any hardware, software, or human element present in the observing or engineering process, which somehow handles (originates, transmits, stores, alters, or destroys) information. Since this code was written for the software group, naturally the Agents table describes software agents in more detail than any of the other types; it records module name, authorship, location of source in the CVS repository, language, brief functional description, etc. (and from this table, many pages of CDR documentation can be generated).

We then established a table Mpaths (meme paths) which describes the passage of Memes between Agents. We say that a meme travels from a Sender to a Receiver, via a Medium, in a Format, with a certain Timing, requiring a certain Elapsed Time, and on the instigation of a Controller of some kind. For example, the slit mask design file travels to the slit mask design collector via the medium of email, in the format of a text file, "on demand", and the controller is the Observation Planner (the author of the file). We have created tables of Formats, Media, and Timing to collect and document all the varieties of these attributes which can apply. (See Chapter 7 for more description of the information flow database, and the Lick Observatory Technical Report – "The Dictionary" for schema design details.)

3.3.2 A User Front End for Design Data

Obviously an endless list of paths in textual form would be unreadable and useless documentation; we require a graphical representation of our design data. The free tool "dot" (Koutsofios and North, Lucent Technologies) renders digraphs from a simple human-readable specification language which can easily be generated from our database. We generate dot-language, which dot then renders into PostScript. The result is a publication-quality drawing representing our software or hardware (or process) design. Several of these drawings appear in the CDR document, notably in Chapter 7.

That takes care of the output; but the input seems terribly tedious. Entering meme ID numbers and agent ID numbers for many hundreds of paths, even using GUI forms, menus, dialogue boxes and all the ergonomics we can afford, is not practical. We required a GUI design tool, something almost as easy to use as a pencil, but with the ability to dump data into the database (or extract and use data from the database).

We call the tool "Etcha" (for obvious reasons). It provides a crude but adequate drawing interface in which the user can place and size (and re-locate and re-size) nodes (Agents), then draw paths between them to indicate the passage of Memes. The Agents and the paths are editable; editor panels pop up displaying all the known attributes of an Agent or a Path. The editor panels have limited query ability into the database, and can offer menus of legitimate values for a given attribute, or can "look up" the attributes of an Agent by name, ID number, etc.

The Agent editor can even be used to add a new Agent to the database, so that the user does not need to leave "Etcha" in order to correct an oversight in this regard.

The tool does not attempt to do any layout or rendering; it does exactly what it is told. The first attempt at this application used the powerful "dged" Tcl suite (North and Ellson, Lucent Technologies), a realtime interactive version of dot. We found it disorienting and frustrating; the layout changed dynamically as new nodes were added and moved, or even across Save and Reload operations. Although "Etcha" draws crude and simple pictures, it does nothing unexpected.

Once the user is content with the layout sketched on the Etcha screen, it can be Saved as an Etcha layout file (which can later be reloaded for reference or further development) and/or Committed to the database. The Commit operation wipes out all paths in the user's design "context" and replaces them with the current drawing.

Etcha can also retrieve an existing context from the database. Having no layout or rendering ability, it cannot make a "pretty" drawing from the data, but it can retrieve and display all the nodes and paths, leaving the user to reposition them in a sensible or comprehensible layout.

Etcha is still a young product, but so far it has proven easy to learn and quick to use. The Overview drawing seen in I.1 was produced in less than 3 hours, and can easily be modified by direct edit of the data or by further Etcha sessions and Commits.

One convenient peculiarity of the MFLowData schema and its interpretation is that "contexts" (effectively, drawings) can be combined. If a flow diagram is too complex to be drawn comfortably with Etcha, it can be drawing in two, three, or more pieces given related drawing names (like DEIMOS.DB.out.doc and DEIMOS.DB.out.op). The drawing generator (traceFamily, documented in Chapter 7) is intelligent enough to recognize that a request for "DEIMOS.DB.out" should include both DEIMOS.DB.out.doc and DEIMOS.DB.out.op, and the rendering program is intelligent enough to suppress any duplication and turn the two drawings into one unified picture. This flexibility is very useful, compensating for the limitations of physical screen size and Etcha's simplicity.

Since database Agents can be created in advance of the actual existence of code modules, Etcha can be used to plan and propose software designs, as well as to document what already exists.

We plan to use Etcha, the database, and `dot` to document and plan our software suite as well as the interactions of the software suite with other (hardware and human) agents. We are encouraged by our success so far; much of the documentation for this review has been generated from our various database resources, and we feel this bodes well for both online and paper DEIMOS documentation to be delivered with the instrument.

A sample Etcha session will be found in Chapter 7, Figure 7.4.

3.4 Keywords in Database

As part of the information management subsystem, we have implemented a keyword database for managing the hundreds of keywords associated with DEIMOS. These keywords are used to operate the instrument and are also recorded within the headers of the FITS images that the instrument produces. The online database serves as a central repository of authoritative information about keywords. Extracts from it are used to build configuration files for operational software (e.g., "dashboard" user interfaces as described in section I.3.2) and to generate documentation.

The keyword database is fully operational and approximately 75% of the DEIMOS keywords (primarily those involved in spectrograph control) have been defined and entered into the database. Tools for generating a formatted and indexed keyword dictionary are also operational, and have been used to generate the major portions of *Lick Observatory Technical Report 80: The DEIMOS Keyword and Database Dictionary.* Copies of this dictionary will be available for review at the CDR, and as noted in section I.1.1, will be available via the WWW on or before June 13.

A sample page from this dictionary is supplied to provide an example of the information that is contained in the database (see Figure 3.1). For each keyword, the dictionary specifies:

- the context in which the keyword is defined
- whether the keyword is read-only (r), write-only (w), or read/write (rw)
- the keyword type (e.g., int, real, char)
- the format to use when formatting the keyword value for a FITS header
- the units in which the keyword is maintained (e.g., degrees C)
- the valid range of values
- a brief definition of the keyword
- references to where any corresponding hardware signals are shown in the electronic schematics for the instrument

8 *II.1: DEIMOS KEYWORD DOCUMENTATION*

failed_okay (map.helpers) r `char(1) L1` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.*

AIRSUPLY (DEIMOS2) r `varchar(68) A` (0.0 to 1.0) *Compressed air supply alarm. Place(s) where you can monitor this signal (air pressure is OK* I): At pin 15 on cable/J3 as shown on drawing el-3036-1d – corresponds to signal chan/16 on Galil controller 1.* Is an instance of failed_okay:

failed_okay (map.helpers) r `char(1) L1` (0.0 to 1.0) *Generic map meme for a system status boolean state (FAILED/OKAY) keywords.*

BARENFAN (DEIMOS2) rw `varchar(68) A` (0.0 to 1.0) *Instrument (barrel) electronics cooling fans. The barrel electronics cooling monitor keyword BARENMON must be set to MANUAL to prevent the monitor from controlling the fan state. Place(s) where you can monitor this signal (fans on* (barrel) O): At pin 33 on cable/J1 as shown on drawing el-3034-1d – corresponds to signal chan/7 on Galil controller 1. At pin 16 on panel/1 as shown on drawing el-3004-1l – corresponds to signal chan/7 on Galil controller 1.* Is an instance of off_on:

off_on (map.helpers) rw `char(1) L1 %s` (0.0 to 1.0) *Generic map meme for boolean state (OFF/ON) keywords.* Valid values of off_on are:

- off :
- on :

BARENMON (DEIMOS2) rw `varchar(68) A` (0.0 to 1.0) *Instrument (barrel) electronics enclosure temperature control monitor.* Is an instance of manual_auto:

manual_auto (map.helpers) rw `char(1) L1 %s` (0.0 to 1.0) *Generic map meme for a control mode boolean state (MANUAL/AUTOMATIC) keywords.* Valid values of manual_auto are:

- manual : System is in manual mode. Automated control suspended.
- auto : System is in automatic control mode.

BARENTRG (DEIMOS2) r `int I11 %d` *[(12 bit adc value)]* (0.0 to 4095.0) *Instrument (barrel) electronics enclosure target temperature in raw A/D converter counts.*

BARENVAL (DEIMOS2) rw `real E13.6` *[degC]* (-50.0 to 50.0) *Instrument (barrel) electronics enclosure temperature in degrees celsius. Place(s)*

De Clarke `<de@ucolick.org>` 1997 June 8

Figure 3.1: Sample Page of Keyword Dictionary

3.5 Software Infrastructure

The original platform for development of Keck instrument control code was Sun/Sparc running SunOS. The code management and development schemes for Keck software have been intimately dependent on features and tools specific to that one platform. Code management was done using the proprietary SCCS software from Sun. Makefiles for the code trees relied on features unique to Sun's version of make.

The inevitable evolution of computing hardware is now forcing a change from SunOS to Solaris. Furthermore, DEIMOS has been contemplating the use of Dec alpha systems for the observer's data acquisition and reduction workstations. The utility of Intel-based laptop computers for code development is another motivation for changing from the original scheme. During the past year CARA and UCO/Lick have collaborated on the design and implementation of significant changes to enhance portability.

3.5.1 Software Maintenance

It is imperative to maintain authoritative central repositories of the source code for the core system and for each instrument. However we also need to be easily able to propagate changes and bug fixes to and from each of the instrument development sites.

The Source Code Control System (SCCS) is not available on non-Sun platforms. Furthermore, the original scheme for its use had relied on the existence of NFS mounts over long distances. Evolving security issues on the ever-growing internet had forced the shutdown of this scheme some years ago. Since that time the separate code repositories at each site have been evolving without means of resynchronizing.

Both CARA and Lick have experience with a freely-available source code system known as CVS. CVS has already been ported to all interesting computing platforms. It provides an environment for code sharing and development which is both natural to use and robust against misuse by unfamiliar users.

CARA is the logical site for the repositories of telescope control code (ACS, DCS) and common instrument control code (KTL). In general, the instrument-building sites are the logical site for the repository of code for each instrument. The new versions of code described in this section have already been placed into CVS. Both CARA and Lick have already demonstrated that we can obtain source code from each other's repository.

3.5.2 Software Portability

CARA made an initial port of the Keck Tasking Library (KTL) code to Solaris during 1996. This port made minimal changes to the makefiles and source code. It relied on many of the compatibility features that Sun provided to ease the transition and was not suitable for non-Sun platforms.

At UCO/Lick De Clarke and Steve Allen adopted this Solaris port of KTL. The GNU autoconfigure program is a widely-used solution for ensuring portability across all platforms. The original KTL makefiles were replaced with GNU configure-style makefiles, many sections of BSD-specific code were replaced with POSIX-conformant code, and erroneous C code using pointers was made 64-bit clean. By the end of last year the same KTL code built and operated correctly on Sparc/SunOS, Sparc/Solaris, Dec alpha/Digital Unix, and PC Intel/Linux platforms.

Although they are completely portable there are several drawbacks of the GNU configure-style makefiles. They require each site to have the entire souce code tree, including portions only relevant to other instruments. They also require significant expertise on the part of each instrument development team in order to maintain them. CARA and Lick agreed that these aspects were not desirable.

William Lupton undertook the task of merging the centrally-managed CARA scheme with the GNU scheme from Lick. His result requires the use of GNU make, but it has been demonstrated to work on all 4 platforms. The makefiles run GNU configure once in the `kroot/etc` directory to convert a `config.mk.in` file into `config.mk`. The makefiles in all other branches of the code tree include the `config.mk` file to inherit default rules and platform dependencies. A little more testing is needed before this scheme can be considered final, but it addresses the needs of the code now and into the future. It also implements a new installation scheme.

3.5.3 Software Installation

The original CARA installation procedure placed all of the installed files into a single directory named `kroot/rel/default` regardless of their function. Many of the installed files were symbolic links into version-numbered directories resident within the source code tree. Following ideas from several discussions between CARA and Lick, Lupton has reworked the installation scheme.

The name of the new installation directory is configurable. Files which are executables, includes, libraries, and data are installed into a version-numbered directory in the install tree. Symbolic links to the current version are placed into separate bin, include, lib, etc. directories. All files installed from "release points" in the source code tree are placed into a particular version-numbered directory.

The discussions between Lick and CARA outlined one other desirable feature which is not currently included in the installation scheme. Certain branches of the source code must be built and installed before other branches may be built. We desire to be able to document exactly which versions of installed include files and libraries were used when each subsequent branch of code is built. Only with this knowledge is it possible to reconstruct an arbitrary previously-installed version of a branch. We expect to continue work on implementing mechanisms for recording this information.

3.5.4 Summary

During the past year the Keck source code has evolved from a state where it could only be built on a single outdated OS to a point where it can be run on a laptop or any other modern Unix variant. The shared portions of code can again be maintained by all interested sites. The installation procedures are much more conformant with standard existing practices. We believe that we will be able to allow all instrument development teams to continue porting this scheme to arbitrary new computing platforms with minimal effort by a single expert.

3.6 FITS File Interchange Formats

The software PDR contained descriptions of several different file structures for DEIMOS images. Not all of these structures have proved necessary, and there is now a clear distinction between archival FITS formats and interim formats suited for operational purposes. The specifications of procedures and dataflows for the design and manufacture of slitmasks are complete, and they use FITS tables to store and communicate information between observer, database, and mill control computer. The content of all these FITS files is detailed in this section.

3.6.1 Archival formats for FITS images

Subsequent to the PDR we organized a Birds of a Feather (BOF) session on mosaic images at the 1996 ADASS meeting in Charlottesville, VA. This allowed us to interact with other sites planning mosaic detectors. As a result it became clear that some standards are emerging for archival storage of multi-amplifier CCD images.

3.6.1.1 Raw images

Following the pattern of HIRES and LRIS the PDR suggested that image sections from several CCDs might be combined into a single FITS array. This is contrary to the consensus of the FITS mosaic community which indicates that raw data from each CCD amplifier will be stored in a separate FITS image file, even when the amplifiers are on the same CCD. This has the disadvantage of breaking spatially contiguous regions of image data into separate FITS arrays. However, it also permits existing software to handle differences in bias, gain, noise, and other electronic characteristics.

These files will be approximately 140 Mbytes in size, and in the case of dual-amplifier readout each image extension will exceed 8 Mbytes. IRAF version 2.11 contains support for multi-extension FITS files, and using beta releases NOAO has already demonstrated the processing of such images.

3.6.1.2 Pipeline-processed images

After pipeline processing, most of the electronic artifacts from each amplifier will be removed from the image sections. In the case of a single CCD where the pixels from several amplifiers are spatially contiguous, and the gain and noise characteristics of the amplifiers are sufficiently similar, it may be desirable to combine the image sections from the amplifiers into a single

FITS array. This may ease the task of existing applications which perform astrometric and photometric operations on extended objects. However, this image format is only relevant in an archival sense if at some time there might be an archive of pipeline-processed DEIMOS images. As this is not in the current plan, we have not specified details of this format.

3.6.2 Operational formats

For the initial testing of the CCD mosaic we will be using Figdisp to display the images. During this interval the CCD data capture software will descramble the amplifier data into a single large image array so that Figdisp can display it all. Contrary to the scheme detailed in chapter 7 of the software PDR, this single large array will not correspond to a FITS image array.

When the prototype version of the NOAO RTD becomes available we intend to modify the data capture agent such that it stores the image sections from each amplifier using the archival format for raw images. NOAO is already testing their RTD with images in this format. The FITS grouper/ungrouper task remains available to assist computing systems which have difficulty manipulating single files of this size, but the DEIMOS systems will be capable of handling these large files.

See Part I section 4.1 for more details on the display issues mentioned in this subsection.

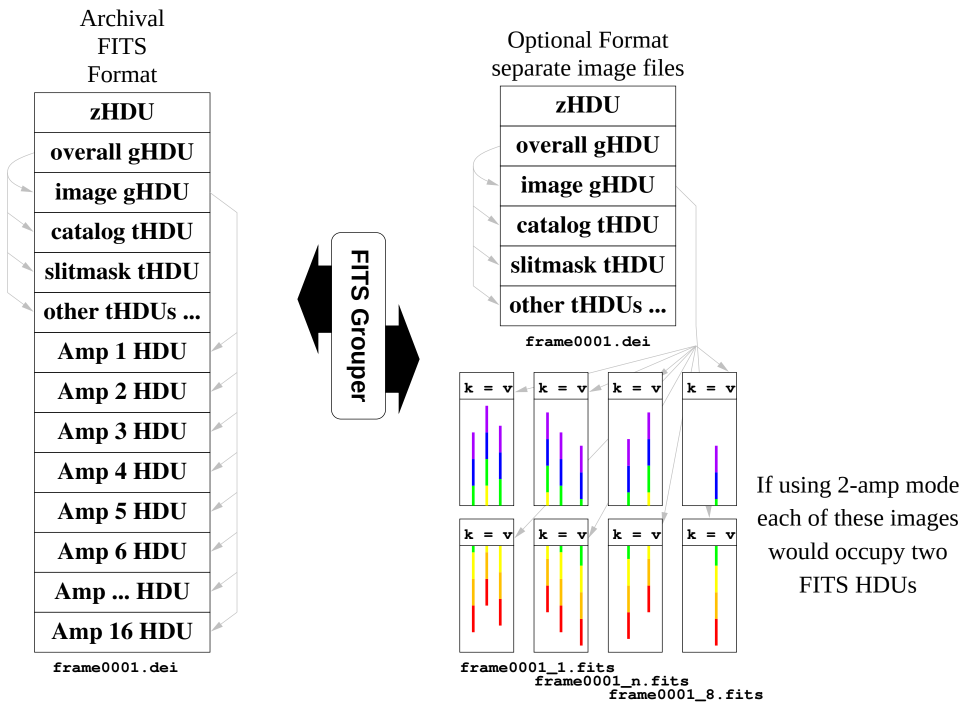


Figure 3.2: FITS File Format

3.6.3 Formats for FITS tables

During the course of the design of the DEIMOS dataflows we determined that FITS tables and RDB tables can be employed in an isomorphic fashion. We have created procedures which directly ingest FITS tables to the RDB, and we can dump any RDB table into FITS for storage or transmission. For details of this see the database schema in Part II Section 7.7. The detailed contents of all RDB tables are contained in Chapter II.2 of the Dictionary.

The design of slitmasks produces dozens of slitlets from catalogs with hundreds of objects. It is an interactive, and often an iterative, process. This requires saving of state during the design in a MaskFile, and subsequent transmission of that file to Keck. Upon arrival at Keck the MaskFile is stored as MaskData in the database. At milling time the MaskData must be sent to the mill for fabrication of slitmasks. Details of these procedures are in Part II Chapter 1.

On the night of the observation the quick look display needs the MaskData to identify each of the spectra. For archival purposes the MaskData must be appended to DEIMOS images taken with a slitmask. Examples of the FITS table headers for the MaskFile are shown below.

3.6.3.1 Object Catalog FITS Table

The list of objects which were contained in slitlets is part of the archival information for the observation, and it is also used by the GUI during quick look display.

```
XTENSION= 'TABLE   '           / ASCII table extension
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / always 2 for a FITS table
NAXIS1  =                  xxx / number of chars needed to describe members
NAXIS2  =                  xxx / number of objects in the catalogs
PCOUNT  =                    0 / required FITS extension keyword
GCOUNT  =                    1 / required FITS extension keyword
TFIELDS =                   19 / Number of columns in the table
----------------------------------------------------------------
EXTNAME = 'DEIMOS_Object_Catalog' / This table describes DEIMOS object catalogs
EXTVER  =                    1 / Unique index of this Catalog table in file

AUTHOR  = 'Rocket J. Squirrel' / Identity of person who constructed catalog
CATNAME = 'My Object Catalog'  / Observer's pet name for catalog
DATE    = 'yyyy-mm-ddThh:mm:ss'/ Date of construction of input catalog table
```

```
-----------------------------------------------------------------
NPRIKEY =                    1 / Number fields contributing to primary key
PKTYP1  = 'ObjPKey'            / native TTYPE of primary key component 1
-----------------------------------------------------------------
The design application may find useful to remember the source
catalogs whence the objects in this HDU originated.  (I.e.,
it's a feature to save state such that the user/designer can
more easily recreate a previous design session.)  If so, the
source catalogs can be stored in the manner described in these
next two bunches of FITS cards.
NFORHDU =                    1 / Number of foreign HDUs referenced by this HDU
HDLOC1  = 'path/to/catfile'    / URI of file containing foreign HDU
HDXTN1  = 'TABLE   '           / FITS XTENSION type of foreign HDU
HDNAM1  = 'Catalog_Files'      / FITS EXTNAME of foreign HDU
HDVER1  =                    1 / FITS EXTVER of foreign HDU
HDPOS1  =                      / Ordinal position of foreign HDU in URI
HDSUM1  = '0000000000000000'   / checksum of foreign HDU
HDDCS1  = '       0'           / checksum of foreign data

NFORKEY =                    1 / foreign key component references from this HDU
FKHDU1  =                    1 / refers to index counted by NFORHDU
FKCOL1  =                      / TBCOL of foreign key component in native HDU
FKTYP1  = 'CatFilePK'          / TTYPE of foreign key component in native HDU
FKFOL1  =                      / TBCOL of primary key component in foreign HDU
FKFYP1  = 'CatFilePK'          / TTYPE of primary key component in foreign HDU
-----------------------------------------------------------------
  Here are the FITS table column descriptions
TBCOL1  =                   xx / First character in column
TFORM1  = 'I6      '           / Fortran 77 format of column
TTYPE1  = 'ObjPKey '           / Primary key for table of objects

TBCOL2  =                    1 / First character in column
TFORM2  = 'A40     '           / Fortran 77 format of column
TTYPE2  = 'OBJECT  '           / Name of object

TBCOL3  =                   xx / First character in column
TFORM3  = 'F11.7   '           / Fortran 77 format of column
TTYPE3  = 'RA_OBJ  '           /
TUNIT3  = 'deg     '           / Unit of measure of column
```

```
TBCOL4  =                     xx / First character in column
TFORM4  = 'F11.7   '             / Fortran 77 format of column
TTYPE4  = 'DEC_OBJ '             /
TUNIT4  = 'deg     '             / Unit of measure of column

TBCOL5  =                     xx / First character in column
TFORM5  = 'A8      '             / Fortran 77 format of column
TTYPE5  = 'RADECSYS'             /
TNULL5  = 'NULL    '             / Indicates data not provided in input

TBCOL6  =                     xx / First character in column
TFORM6  = 'F13.6   '             / Fortran 77 format of column
TTYPE6  = 'EQUINOX '             /
TUNIT6  = 'annum   '             / Unit of measure of column
TNULL6  = '-99999.      '        / Indicates data not provides in input

TBCOL7  =                     xx / First character in column
TFORM7  = 'F17.9   '             / Fortran 77 format of column
TTYPE7  = 'MJD-OBS '             /
TUNIT7  = 'diem    '             / Unit of measure of column
TNULL7  = '-999999.          '   / Indicates data not provided in input

TBCOL8  =                     xx / First character in column
TFORM8  = 'F7.3    '             / Fortran 77 format of column
TTYPE8  = 'mag     '             /
TNULL8  = '-99.    '             / Indicates data not provides in input

TBCOL9  =                     xx / First character in column
TFORM9  = 'A6      '             / Fortran 77 format of column
TTYPE9  = 'pBand   '             /
TNULL9  = 'NULL    '             / Indicates data not provides in input

TBCOL10 =                     xx / First character in column
TFORM10 = 'F14.3   '             / Fortran 77 format of column
TTYPE10 = 'RadVel  '             /
TUNIT10 = 'm/s     '             / Unit of measure of column
TNULL10 = '-999999999.   '       / Indicates data not provided in input

TBCOL11 =                     xx / First character in column
TFORM11 = 'F10.7   '             / Fortran 77 format of column
```

```
TTYPE11 = 'MajAxis '           /
TUNIT11 = 'arcsec  '           / Unit of measure of column
TNULL11 = '-9.       '         / Indicates data not provided in input

TBCOL12 =                   xx / First character in column
TFORM12 = 'F11.7   '           / Fortran 77 format of column
TTYPE12 = 'MajAxPA '           / c
TUNIT12 = 'deg     '           / Unit of measure of column
TNULL12 = '999.       '        / Indicates data not provided in input

TBCOL13 =                   xx / First character in column
TFORM13 = 'F10.7   '           / Fortran 77 format of column
TTYPE13 = 'MinAxis '           /
TUNIT13 = 'arcsec  '           / Unit of measure of column
TNULL13 = '-9.       '         / Indicates data not provided in input

TBCOL14 =                   xx / First character in column
TFORM14 = 'F9.4    '           / Fortran 77 format of column
TTYPE14 = 'PM_RA   '           /
TUNIT14 = 'arcsec/(Julian annum)' / Unit of measure of column
TNULL14 = '-999.    '          / Indicates data not provided in input

TBCOL15 =                   xx / First character in column
TFORM15 = 'F9.4    '           / Fortran 77 format of column
TTYPE15 = 'PM_Dec  '           /
TUNIT15 = 'arcsec/(Julian annum)' / Unit of measure of column
TNULL15 = '-999.    '          / Indicates data not provided in input

TBCOL16 =                   xx / First character in column
TFORM16 = 'F7.4    '           / Fortran 77 format of column
TTYPE16 = 'Parallax'           /
TUNIT16 = 'arcsec  '           / Unit of measure of column
TNULL16 = '-9.     '           / Indicates data not provided in input

TBCOL17 =                   xx / First character in column
TFORM17 = 'A20     '           / Fortran 77 format of column
TTYPE17 = 'ObjClass'           /
TNULL17 = 'NULL    '           / Indicates data not provides in input

TBCOL18 =                   xx / First character in column
```

```
TFORM18 = 'I6      '           / Fortran 77 format of column
TTYPE18 = 'CatFilePK'          /
TNULL18 = '    -1  '           / Indicates data not provides in input

END
```

3.6.3.2 Catalog File FITS Table

This table is mentioned by the commentary in the catalog table. The SIMULATOR will provide a better user interface for iterative mask design if it saves state information about the catalog(s) whence the objects came. This FITS table is *not* part of the MaskFile transmitted to Keck; it is local to the observer.

```
XTENSION= 'TABLE   '           / ASCII table extension
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / always 2 for a FITS table
NAXIS1  =                  xxx / number of chars needed to describe members
NAXIS2  =                  xxx / number of objects in the catalogs
PCOUNT  =                    0 / required FITS extension keyword
GCOUNT  =                    1 / required FITS extension keyword
TFIELDS =                    2 / Number of columns in the table
-----------------------------------------------------------------
EXTNAME = 'Catalog_Files'      / This table describes catalog source files
EXTVER  =                    1 / Unique index of this Catalog table in file

DATE    = 'yyyy-mm-ddThh:mm:ss'/ Date of construction of catalog table
-----------------------------------------------------------------
  Here are the FITS table column descriptions
TBCOL1  =                    1 / First character in column
TFORM1  = 'I6     '            / Fortran 77 format of column
TTYPE1  = 'CatFilePKey'        / Primary key for table of objects

TBCOL2  =                    7 / First character in column
TFORM2  = 'A255   '            / Fortran 77 format of column
TTYPE2  = 'FileName'           / Name of catalog file
-----------------------------------------------------------------
NPRIKEY =                    1 / Number fields contributing to primary key
PKTYP1  = 'CatFilePKey'        / native TTYPE of primary key component 1
-----------------------------------------------------------------
The following information is for documentary purposes only.
```

```
In normal operation this HDU is referenced by other HDUs, but
it is not used to make references into any other HDU.
The sole reason for the existence of this block of information
is to assist a human to determine the purpose of this HDU
if it should become orphaned.
NFORHDU =                    1 / Number of foreign HDUs referenced by this HDU
HDLOC1  = 'path/to/catalog'    / URI of file containing foreign HDU
HDXTN1  = 'TABLE   '           / FITS XTENSION type of foreign HDU
HDNAM1  = 'DEIMOS_Input_Catalog'/FITS EXTNAME of foreign HDU
HDVER1  =                    1 / FITS EXTVER of foreign HDU
HDPOS1  =                      / Ordinal position of foreign HDU in URI
HDSUM1  = '0000000000000000'   / checksum of foreign HDU
HDDCS1  = '       0'           / checksum of foreign data

NFORKEY =                    0 / foreign key component references from this HDU
-----------------------------------------------------------------
END
```

3.6.3.3 Mask Design FITS Table

The output of SIMULATOR is a Mask Design.

```
XTENSION= 'TABLE   '           / ASCII table extension
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / always 2 for a FITS table
NAXIS1  =                  xxx / number of chars needed to describe members
NAXIS2  =                  xxx / number of objects in the catalogs
PCOUNT  =                    0 / required FITS extension keyword
GCOUNT  =                    1 / required FITS extension keyword
TFIELDS =                    8 / Number of columns in the table
-----------------------------------------------------------------
EXTNAME = 'DEIMOS_Mask_Design' / This table describes DEIMOS mask design
EXTVER  =                    1 / Unique index of this Design table in FITS file

AUTHOR  = 'Rocket J. Squirrel' / Identity of person who designed mask
DESPID  =                 nnnn / DEIMOS-assigned identity of designer
DESNAME = 'My Mask Design'     / Observer's pet name for mask design
DESTOOL = 'My Mask Design Tool'/ Name & version of tool used to make design
DATE    = 'yyyy-mm-ddThh:mm:ss'/ Date of construction of mask design

COMMENT   Various keywords specifying the telescope axis as designed
```

```
RA_PNT  =          xxx.xxxxxxx / Right Ascension of Keck II axis [deg]
DEC_PNT =          xxx.xxxxxxx / Declination of Keck II axis [deg]
RADECPNT= 'FKx     '           / Reference frame for RA_PNT and DEC_PNT

PA_PNT  =         xxxx.xxxxxxx / Position angle of DEIMOS mask on sky [deg]

EQUINPNT=        xxxxxx.xxxxxx / Equinox for RA_PNT and DEC_PNT [annum]
MJD_PNT =    24xxxxx.xxxxxxxxx / needed if RADECPNT is FK4 [diem]

LST_DES =          xxx.xxxxxxx / designed Local Sideral Time for mask use [deg]
DATEDES = 'yyyy-mm-ddThh:mm:ss'/ designed date for mask use
----------------------------------------------------------------
  Here are the FITS table column descriptions
TBCOL1  =                    1 / First character in column
TFORM1  = 'I6      '           / Fortran 77 format of column
TTYPE1  = 'slitPKey'           /

TBCOL2  =                   xx / First character in column
TFORM2  = 'F11.7   '           / Fortran 77 format of column
TTYPE2  = 'slitRA  '           /
TUNIT2  = 'deg     '           / Unit of measure of column

TBCOL3  =                   xx / First character in column
TFORM3  = 'F11.7   '           / Fortran 77 format of column
TTYPE3  = 'slitDec '           /
TUNIT3  = 'deg     '           / Unit of measure of column

TBCOL4  =                   xx / First character in column
TFORM4  = 'A1      '           / Fortran 77 format of column
TTYPE4  = 'slitTyp '           / Parallelogram, Alignment, Ellipse, Square

TBCOL5  =                   xx / First character in column
TFORM5  = 'F11.3   '           / Fortran 77 format of column
TTYPE5  = 'slitLen '           /
TUNIT5  = 'arcsec  '           / Unit of measure of column

TBCOL6  =                   xx / First character in column
TFORM6  = 'F11.3   '           / Fortran 77 format of column
TTYPE6  = 'slitWid '           /
TUNIT6  = 'arcsec  '           / Unit of measure of column
```

```
TBCOL7  =                    xx / First character in column
TFORM7  = 'F11.7   '            / Fortran 77 format of column
TTYPE7  = 'slitLPA '            / sky pa of length direction of slit
TUNIT7  = 'deg     '            / Unit of measure of column
TNULL7  = '999.       '         / Indicates data not provided in input

TBCOL8  =                    xx / First character in column
TFORM8  = 'F11.7   '            / Fortran 77 format of column
TTYPE8  = 'slitWPA '            / this may not be the best choice of name
TUNIT8  = 'deg     '            / Unit of measure of column
TNULL8  = '999.       '         / Indicates data not provided in input
----------------------------------------------------------------
NPRIKEY =                     1 / Number fields contributing to primary key
PKCOL1  =                     1 / native TBCOL of primary key component 1
PKTYP1  = 'slitPKey'            / native TTYPE of primary key component 1
----------------------------------------------------------------
The design application may find useful to remember the source
catalog which was used to create the design in this HDU.
(I.e., it's a feature to save state such that the
user/designer can more easily recreate a previous design
session.)  If so, the catalog can be stored in the
manner described in this next bunch of FITS cards.
NFORHDU =                     1 / Number of foreign HDUs referenced by this HDU
HDLOC1  = 'path/to/catalog'     / URI of file containing foreign HDU
HDXTN1  = 'TABLE   '            / FITS XTENSION type of foreign HDU
HDNAM1  = 'DEIMOS_Input_Catalog'/FITS EXTNAME of foreign HDU
HDVER1  =                     1 / FITS EXTVER of foreign HDU
HDPOS1  =                       / Ordinal position of foreign HDU in URI
HDSUM1  = '0000000000000000'    / checksum of foreign HDU
HDDCS1  = '       0'            / checksum of foreign data

NFORKEY =                     0 / foreign key component references from this HDU
----------------------------------------------------------------
This last chunk is not yet fully specified.

The FITS header explicitly represents a single row of a
RDB table containing the information common to all slitlets
in a single mask design.  Its primary key cannot be known
in advance of RDB ingest, but upon outgest it can be stored
```

```
in the FITS EXTVER keyword.

This FITS table explicitly represents a selected subset of a
RDB table containing the slitlet information on *all* mask
designs.  The RDB table will have an additional field which
represents the uniqueID of the particular mask design.

Inclusion of the foreign key formalism here permits us to
point to the values of several keywords which do not
belong in this FITS header but which do belong in the
RDB table row that will be created from the header.

In this case the next block of lines could equally well be
communicated simply by redundantly filling in the values
from the catalog FITS header.
NQRYKEY =                    3 / Number of keywords with queryable values
QKHDU1  =                    1 / refers to index counted by NFORHDU
QKCOL1  =                   -1 / TBCOL of foreign key component in native HDU
QKTYP1  = 'CatAuth '           / TTYPE of foreign key component in native HDU
QKFOL1  =                      / TBCOL of primary key component in foreign HDU
QKFYP1  = 'AUTHOR  '           / TTYPE of primary key component in foreign HDU
QKHDU2  =                    1 / refers to index counted by NFORHDU
QKCOL2  =                   -1 / TBCOL of foreign key component in native HDU
QKTYP2  = 'CatName '           / TTYPE of foreign key component in native HDU
QKFOL2  =                      / TBCOL of primary key component in foreign HDU
QKFYP2  = 'CATNAME '           / TTYPE of primary key component in foreign HDU
QKHDU3  =                    1 / refers to index counted by NFORHDU
QKCOL3  =                   -1 / TBCOL of foreign key component in native HDU
QKTYP3  = 'CatDate '           / TTYPE of foreign key component in native HDU
QKFOL3  =                      / TBCOL of primary key component in foreign HDU
QKFYP3  = 'CATDATE '           / TTYPE of primary key component in foreign HDU
----------------------------------------------------------------
END
```

3.6.3.4 Mask Blueprint FITS Table

The output of MAPMASK is a Mask Blueprint.

```
XTENSION= 'TABLE   '           / ASCII table extension
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / always 2 for a FITS table
```

```
NAXIS1  =                  xxx / number of chars needed to describe members
NAXIS2  =                  xxx / number of objects in the catalogs
PCOUNT  =                    0 / required FITS extension keyword
GCOUNT  =                    1 / required FITS extension keyword
TFIELDS =                    8 / Number of columns in the table
----------------------------------------------------------------
EXTNAME = 'DEIMOS_Mask_Blueprint' / This table describes DEIMOS mask blueprint
EXTVER  =                    1 / Unique index of this Blueprint table in file

AUTHOR  = 'Rocket J. Squirrel' / Identity of person who converted design to blue
BLUPID  =                 nnnn / DEIMOS-assigned identity of author/mask owner
BLUNAME = 'My Mask Blueprint'  / Observer's pet name for mask blueprint
BLUTOOL = 'My Blueprint Tool'  / Name & version of design->blueprint tool
DATE    = 'yyyy-mm-ddThh:mm:ss'/ Date of construction of mask blueprint

LST_USE =         xxx.xxxxxxx / intended Local Sideral Time for mask use [deg]
DATEUSE = 'yyyy-mm-ddThh:mm:ss'/ intended date for mask use

TELESCOP= 'Keck II '           / was used to look up geod & atmo info for refrac

REFRALG = 'SLALIB  '           / atmospheric refraction algorithm
ATMTEMPC=               xxx.x / air temperature used for refraction [deg C]
ATMPRES =              xxxx.x / air pressure used for refraction [mbar]
ATMHUMID=               0.xxx / relative humidity used for refraction
ATMTTLAP=              0.0065 / troposph. adiabatic temp. lapse rate [K/m]
REFWAVE =        0.xxxxxxxxxx / wavelength used for refraction [m]

DISTMETH= 'name of telescope distortion methodology'
----------------------------------------------------------------
  Here are the FITS table column descriptions
TBCOL1  =                    1 / First character in column
TFORM1  = 'I6      '           / Fortran 77 format of column
TTYPE1  = 'bSlitPK '           /

TBCOL2  =                   xx / First character in column
TFORM2  = 'I6      '           / Fortran 77 format of column
TTYPE2  = 'dSlitPK '           /

TBCOL3  =                   xx / First character in column
TFORM3  = 'A1      '           / Fortran 77 format of column
```

```
TTYPE3  = 'slitTyp '          / regurgitated from Mask Design

TBCOL4  =                  xx / First character in column
TFORM4  = 'F9.6    '          / Fortran 77 format of column
TTYPE4  = 'slitx1  '          /
TUNIT4  = 'm       '          / Unit of measure of column

TBCOL5  =                  xx / First character in column
TFORM5  = 'F9.6    '          / Fortran 77 format of column
TTYPE5  = 'slity1  '          /
TUNIT5  = 'm       '          / Unit of measure of column

TBCOL6  =                  xx / First character in column
TFORM6  = 'F9.6    '          / Fortran 77 format of column
TTYPE6  = 'slitx2  '          /
TUNIT6  = 'm       '          / Unit of measure of column

TBCOL7  =                  xx / First character in column
TFORM7  = 'F9.6    '          / Fortran 77 format of column
TTYPE7  = 'slity2  '          /
TUNIT7  = 'm       '          / Unit of measure of column

TBCOL8  =                  xx / First character in column
TFORM8  = 'F9.6    '          / Fortran 77 format of column
TTYPE8  = 'slitx3  '          /
TUNIT8  = 'm       '          / Unit of measure of column

TBCOL9  =                  xx / First character in column
TFORM9  = 'F9.6    '          / Fortran 77 format of column
TTYPE9  = 'slity3  '          /
TUNIT9  = 'm       '          / Unit of measure of column

TBCOL10 =                  xx / First character in column
TFORM10 = 'F9.6    '          / Fortran 77 format of column
TTYPE10 = 'slitx4  '          /
TUNIT10 = 'm       '          / Unit of measure of column

TBCOL11 =                  xx / First character in column
TFORM11 = 'F9.6    '          / Fortran 77 format of column
TTYPE11 = 'slity4  '          /
```

```
TUNIT11 = 'm       '           / Unit of measure of column
-----------------------------------------------------------------
NPRIKEY =                    1 / Number fields contributing to primary key
PKCOL1  =                    1 / native TBCOL of primary key component 1
PKTYP1  = 'bSlitPK '           / native TTYPE of primary key component 1
-----------------------------------------------------------------
The following information permits references to the mask design
NFORHDU =                    1 / Number of foreign HDUs referenced by this HDU
HDLOC1  = 'path/to/mask_design'/ URI of file containing foreign HDU
HDXTN1  = 'TABLE   '           / FITS XTENSION type of foreign HDU
HDNAM1  = 'DEIMOS_Mask_Design' / FITS EXTNAME of foreign HDU
HDVER1  =                    1 / FITS EXTVER of foreign HDU
HDPOS1  =                      / Ordinal position of foreign HDU in URI
HDSUM1  = '0000000000000000'   / checksum of foreign HDU
HDDCS1  = '       0'           / checksum of foreign data

NFORKEY =                    1 / foreign key component references from this HDU
FKHDU1  =                    1 / refers to index counted by NFORHDU
FKCOL1  =                      / TBCOL of foreign key component in native HDU
FKTYP1  = 'dSlitPK '           / TTYPE of foreign key component in native HDU
FKFOL1  =                      / TBCOL of primary key component in foreign HDU
FKFYP1  = 'dSlitPK '           / TTYPE of primary key component in foreign HDU
-----------------------------------------------------------------
END
```

3.6.3.5 Slit/Object Map FITS Table

The correspondence between objects and slitlets is not one-to-one, so a separate table is needed to map between the catalog and design. There may be more than one object within a given slitlet. (For the purposes of this database, regions or sub-components of an object which fall into more than one slitlet are deemed to be separately named objects.)

```
XTENSION= 'TABLE   '           / ASCII table extension
BITPIX  =                    8 / number of bits per data pixel
NAXIS   =                    2 / always 2 for a FITS table
NAXIS1  =                  xxx / number of chars needed to describe members
NAXIS2  =                  xxx / number of objects in the catalogs
PCOUNT  =                    0 / required FITS extension keyword
GCOUNT  =                    1 / required FITS extension keyword
TFIELDS =                    2 / Number of columns in the table
```

```
-----------------------------------------------------------------
EXTNAME = 'DEIMOS_Slit_Object_Map' / This table identifies objects in slits
EXTVER  =                    1 / Unique index of this Map table in file
-----------------------------------------------------------------
  Here are the FITS table column descriptions
TBCOL1  =                    1 / First character in column
TFORM1  = 'I6      '           / Fortran 77 format of column
TTYPE1  = 'dSlitPK '           /

TBCOL2  =                   xx / First character in column
TFORM2  = 'I6      '           / Fortran 77 format of column
TTYPE2  = 'ObjPKey '           / Primary key for table of objects
-----------------------------------------------------------------
  It is not clear that we need or want a primary key in here
  because every row should be unique.
NPRIKEY =                    2 / Number fields contributing to primary key
PKCOL1  =                    1 / native TBCOL of primary key component 1
PKTYP1  = 'slitPKey'           / native TTYPE of primary key component 1
PKCOL2  =                    2 / native TBCOL of primary key component 1
PKTYP2  = 'ObjPKey '           / native TTYPE of primary key component 1
-----------------------------------------------------------------
  We must specify foreign keys into two other HDUs
NFORHDU =                    2 / Number of foreign HDUs referenced by this HDU
  The following information permits references to the catalog
HDLOC1  = 'path/to/mask/design'/ URI of file containing foreign HDU
HDXTN1  = 'TABLE   '           / FITS XTENSION type of foreign HDU
HDNAM1  = 'DEIMOS_Mask_Design' / FITS EXTNAME of foreign HDU
HDVER1  =                    1 / FITS EXTVER of foreign HDU
HDPOS1  =                      / Ordinal position of foreign HDU in URI
HDSUM1  = '0000000000000000'   / checksum of foreign HDU
HDDCS1  = '       0'           / checksum of foreign data
  The following information permits references to the catalog
HDLOC2  = 'path/to/catalog'    / URI of file containing foreign HDU
HDXTN2  = 'TABLE   '           / FITS XTENSION type of foreign HDU
HDNAM2  = 'DEIMOS_Input_Catalog'/FITS EXTNAME of foreign HDU
HDVER2  =                    1 / FITS EXTVER of foreign HDU
HDPOS2  =                      / Ordinal position of foreign HDU in URI
HDSUM2  = '0000000000000000'   / checksum of foreign HDU
HDDCS2  = '       0'           / checksum of foreign data
```

```
NFORKEY =                    2 / foreign key component references from this HDU
FKHDU1  =                    1 / refers to index counted by NFORHDU
FKCOL1  =                      / TBCOL of foreign key component in native HDU
FKTYP1  = 'dSlitPK '           / TTYPE of foreign key component in native HDU
FKFOL1  =                      / TBCOL of primary key component in foreign HDU
FKFYP1  = 'dSlitPK '           / TTYPE of primary key component in foreign HDU
FKHDU2  =                    2 / refers to index counted by NFORHDU
FKCOL2  =                      / TBCOL of foreign key component in native HDU
FKTYP2  = 'ObjPKey '           / TTYPE of foreign key component in native HDU
FKFOL2  =                      / TBCOL of primary key component in foreign HDU
FKFYP2  = 'ObjPKey '           / TTYPE of primary key component in foreign HDU
----------------------------------------------------------------
END
```

3.7 Simulated DEIMOS Image

We have constructed several simulated DEIMOS images. The images have been successfully viewed using existing image display clients.

File transfers of 138Mbyte FITS files over the 100Mb/s ethernet have been clocked at 45 seconds, or about 3Mbytes/s.

We are awaiting the official releaes of IRAF 2.11 which will permit us to perform more timing tests. Of particular interest will be the procesor speed with which the Dec alpa with 1Gbyte of RAM runs typical pipeline reduction steps such as those in IRAF `ccdproc`.

3.8 Tests of SDSU 2nd Generation CCD H/W

We have received and tested our first set of 2nd generation controller boards, and the results of these tests were reported at the DEIMOS CCD review held May 20, 1997. Copies of the report from that review are available, and selected chapters (3: CCD System Overview, 5: Science CCD Controller, and 6: Data Capture) of that report are available via the WWW at the following URL: http://www.ucolick.org/~kibrick.

Our existing engineering software tools for operating the SDSU controllers have been updated and tested with the second-generation SDSU controllers boards. We can now successfully program and generate valid CCD clocking waveforms and bias voltages using the second-generation boards, and have used a high-speed oscilloscope to verify the proper operation of the video processing signal chain. Our VME crate (currently SunOS-based) is reliably communicating with the second-generation SDSU timing board and utility board.

We are sufficiently confident with this new hardware and with our ability to generate DSP software for it that we have sought (and obtained) from the DEIMOS CCD Review Board permission to place a large order for second-generation SDSU controller boards. That purchase order has cleared the campus purchasing office, and was issued on June 5. Delivery of second-generation SDSU boards should begin in July and be completed by the end of September.

3.9 Instrument Control and Terminal Server Tests

This section covers the tests of the motor controllers, their firmware, the terminal servers connected to the motor controllers, and the dispatch software interface communicating with the motor controllers via the terminal servers. It describes the extent of the platform independence of the software components, the level of completeness of the software, and the functional proof required for the system designs. It does not cover all the configurations of hardware and software used in the tests.

3.9.1 General hardware overview

The motor controller and terminal server tests were conducted in Santa Cruz and at Mt. Hamilton using Lick Prime Focus Camera (PFCAM) and ESI stages. Five stages were available for the tests. The two linear stages and two rotary filter wheels from PFCAM were used separately and together for the majority of the testing. An ESI filter wheel was also tested separately.

A variety of Galil DMC-1500 series controllers were used to control the stages during testing. They included a four axis controller with an auxillary I/O module added (DMC-1540-72), a five axis controller (DMC-1550), an eight axis controller with the auxillary I/O module added (DMC-1580-72), and an eight axis controller without the auxillary I/O module (DMC-1580).

Communication with the motor controllers was via two serial ports on the motor controllers. The controller serial ports were connected to I/O ports on a Lantronix terminal server. RTS/CTS was used for the handshaking protocol. In normal operations the main controller serial port accepts commands and the second auxillary serial port is used exclusively as a diagnostics message port. Serial communications tests of the terminal server used both the main serial port and the auxillary serial ports of the motor controllers extensively. Initial tests of the Lantronix ETS8/UF terminal servers revealed flaws in their communications protocol and scheduling algorithms. Lantronix sent an ETS8P beta unit to us for testing and evaluation. The ETS8P beta unit was used in all of the tests described in this document, unless otherwise noted.

The tests were performed with the terminal server connected to to either a private ethernet in Santa Cruz or to the Mt Hamilton network.

The software communicating to the motor controller via the terminal server ran on a variety of SPARC platforms running either SunOS or Solaris.

3.9.2 General software overview

During normal control of the stages, the motor controller was running two tasks continuously. One, called UHAUL, monitors the manual control panel. The other, called BEKINS, handles controlling the stages when requested, either by UHAUL or by external commands sent to the motor controller's main serial port. During tests of the Lantronix terminal server, special programs were loaded onto the motor controllers to flood the main and auxillary serial ports with data.

A prototype dispatch process was connected to the motor controller via the terminal server ports to test stage control from De Clarke's prototype dashboard GUI and the existing "show" and "modify" KTL shell commands.

A serial I/O test program was connected to the motor controller via the terminal server ports to test the terminal server handshaking, bandwidth and scheduling.

During tests in Santa Cruz, the dispatch process communicated to the dashboard and KTL shell commands via messages sent to a traffic process running on a local SPARC10 running SunOs. During tests on Mt. Hamiltom with PFCAM, the dispatch process communicated to dashboard and KTL shell commands via messages sent to a traffic running on the same machine in Santa Cruz, then via traffic running on a similar machine on at Mt. Hamilton.

The user interface processes were run from a SPARC5 running SunOS in Santa Cruz and the SPARC10 on Mt. Hamilton.

3.9.3 Lantronix testing

The Lantronix ETS8P beta version terminal server was tested with the main and auxillary serial ports of four Galil DMC-1500 series controllers connected to the eight serial ports on the terminal server. The baud rate for each port was set to 19200, with RTS/CTS handshaking enabled. A simple test program pumped a test pattern to the controllers and read the test pattern echo back on the main serial port and read the output of the test pattern on the auxillary serial port. The test ran for 48 hours. The average simultaneous throughput per port was 1400 characters per second compared to an expected maximum of 2000 characters per second. Counters in the terminal server indicated RTS/CTS handshaking was working without any errors. No data errors were seen in the test patterns returned by the controllers.

The terminal server also spent two nights riding on the prime focus ring of the Shane Telescope at Mt. Hamilton handling communication to the PFCAM motor controller. No problems were observed with the terminal server during use on the telescope.

We have negotiated an agreement with Lantronix to exchange the model ETS8/UF terminal servers for credit toward the purchase of the improved ETS8P devices. This exchange should be completed by the end of June.

3.9.4 Motor control tests

The goal of the motor controller firmware was to provide tasks that handled command requests for homing and moving stages from either a manual control box connected to the digital input ports of the motor controller, or external requests arriving in Galil command language on the controller's main serial port.

The UHAUL and BEKINS tasks were developed by Jim Burrous to satisfy these objectives. The UHAUL task handles the signals from the manual control box. It sets up request information for the BEKINS task and BEKINS handles the request. Incoming requests on the serial port set up the request information similar to UHAUL and BEKINS handles it just like the manual move requests from UHAUL.

Jim performed the validity tests of the firmware using the PFCAM instrument. All of the movement options available from the manual control box were exercised extensively during the construction and testing of the PFCAM instrument. Each stage was homed, jogged at fast and slow speeds, moved to ordinal positions defined in the position data arrays on the controller and stopped during motion via the stop button. The same tests were performed via the serial interface to the controller by typing the commands required by the controller to set up home/move/stop commands. The raw positioning movement mode needed by dispatch was tested via the serial port.

Linear stages with a single encoder were tested in raw position and jog mode out to the secondary limit switches at each end of travel. The firmware was tested for correct handling of all limit switch conditions. Both single and dual encoder rotary stages have been controlled successfully.

3.9.5 Dispatcher interface and keyword tests

Dispatch's MUSIC message handling interface was tested with a basic set of PFCAM stage keywords. The message handling routines used by the PFCAM keyword library and the PFCAM dispatch process are identical to those used for DEIMOS. Only the xxxxxRAW, motor encoder position keywords, were available. The xxxxxRAW keywords provided all necessary services to test reading and writing values to dispatch and handling of dispatch value change broadcasts. The show, modify, and xshow shell commands and De Clarke's dashboard user interface provided the access to the keyword library.

3.9.6 Changes since PFCAM engineering run

A minor problem was found in the dispatch service loop. It responded too quickly to show keyword commands with more than one keyword in the command line. The show command did not have time to send a request for the next keyword on the command line before dispatch went off to read status information from the motor controller. This resulted in a half second wait before the display of each keyword in the show command line. This has been fixed by adding a small wait inside the dispatch service loop.

The only major change to the dispatch was in the dispatch service routines handling stage I/O requests. The original xxxxxRAW routines were divided into a configuration device type used to generate the xxxxxRAW device data and the device data for the other types of stage keywords, i.e. xxxxxTRG, xxxxxORD, xxxxxNAM, etc. The new device code works fine for reading xxxxxRAW and xxxxxTRG keywords. It looks like it performs everything required to perform xxxxxRAW writes, but it hasn't been tested with a stage for verification.

3.9.7 Summary

After discovering problems with the ETS8/UF Lantronix terminal servers received for use in DEIMOS, a new terminal server model, the ETS8P, has been tested and found to perform acceptably.

The major components of the motor controller firmware, dispatch interface and KTL keyword library has been tested successfully. The tests included running the PFCAM instrument during an engineering run on the Shane Telescope at Mt. Hamilton. The dispatch process has been run from both SunOS and Solaris machines, including connecting a remotely run dispatch task with the terminal server via the network connections between Santa Cruz and Mt. Hamilton.

Chapter 4

Risk Items

4.1 Image Display Plans

The DEIMOS CCD mosaic will closely resemble operational mosaics at University of Hawaii and NOAO. These mosaics share many image display requirements. For the sake of the occasional users of DEIMOS we prefer not to create yet another image display peculiar to one instrument. Subsequent to the software PDR we have continued to investigate various existing image display options. We now intend to follow a two-phase plan.

4.1.1 Phase 1: Plan for Laboratory Testing

For the PDR we had investigated the behavior of several commonly-used astronomical image display clients with DEIMOS-sized images. We found that the existing clients had no limits inhibiting the display of 8k×8k images. When tested on machines with sufficient RAM to hold the entire image each of the clients demonstrated acceptable responsiveness to user interaction. The responsiveness is adequate for laboratory testing of the DEIMOS CCD mosaic. However, all existing clients lack numerous features specified for the operational system in the DEIMOS PDR.

During initial laboratory testing of the DEIMOS CCD mosaic we intend to use a modified version of the existing Keck image display client, Figdisp. Figdisp already handles real-time display of data from disjoint sections of a multi-amplifier CCD. Figdisp also provides access to the actual 16-bit pixel values. Both of these capabilities have proven invaluable during laboratory testing of CCD detectors for previous instruments.

John Cromer at Caltech recently ported Figdisp from SunOS to Solaris. We are continuing that port to be 64-bit clean for operation on DEC alpha (and other) processors. While porting we will modify Figdisp so that its pixel tracking can indicate that image sections may originate from separate CCDs with individual coordinate systems. This will provide us with an interface that can monitor and inspect image sections in real time during laboratory experiments on the DEIMOS CCDs and controllers. NOAO is already using a similarly modified version of the IRAF image interaction tools with their mosaic.

Existing image display clients have required that the image sections be stored in a single data array. For HIRES and LRIS at Keck this is done as part of the descrambling by the "lickserv" process. Early versions of the software for the NOAO mosaic relied on an external IRAF procedure When being used with Figdisp the DEIMOS "lickserv" process will assemble the image sections during the readout into a single array which approximates the true geometry.

4.1.2 Phase 2: Plan for Commissioning

Subsequent to the DEIMOS software PDR the NOAO mosaic became operational. The display for the NOAO mosaic was not specifically designed for mosaic images. Initially it used an existing image display client (ximtool) in a manner akin to the scheme presented for DEIMOS at the PDR. This display was recently enhanced to permit real-time viewing of the mosaic CCD readouts using data stored in the archival FITS format.

Last year SAO/CfA and NOAO received approval and funds to create an improved version of their existing image display clients (ximtool and SAOtng). This project is known as the Plug-in Image Extension (PIE) server. PIE will use a message-passing system to permit cooperation between tasks running on a heterogeneous network. Using a plug-in API it will interact with arbitrary data source and processing modules. PIE will not be ready in time for the testing phase of DEIMOS. Many of its features will, however, be prototyped in the NOAO Real Time Display (RTD), which is to be available in late 1997.

We have held several discussions with Doug Tody on the subject of the NOAO RTD. These include two full day, on site visits to gather details of its architecture and development. RTD is being built to serve the immediate needs of the NOAO mosaic detector. However, it is also serving as a prototype implementation of PIE. A delivery schedule for RTD was outlined at a meeting in 1996 December:

1997 May RTD design will be ready for outside review, including details of the plug-in API.

1997 August A working release of RTD will be available to UCO/Lick.

1997 October RTD will be in operation with the NOAO mosaic detector.

An early version of RTD is already in use for astronomical observations with the NOAO mosaic. The messaging system is already functional, and Doug Tody has supplied a copy of its documentation. We are continuing to correspond with Doug Tody. As of the beginning of 1997 June the RTD development is on schedule.

4.1.3 Contingencies

If by the time of DEIMOS commissioning the NOAO RTD were not available then we would continue to use Figdisp for observations. The capabilities of Figdisp are adequate for the purposes of scientific data gathering. However, several notable features specified at the PDR would be missing.

- conversions from pixel coordinates to various world coordinates
- automatic lookup of slitlet id and wavelength
- communication with KTL for indicating readout regions
- communication with non-Figaro image analysis systems
- graphics overlays from external processes

Without major changes Figdisp will not meet the DEIMOS requirements. However, we are quite certain that it will suffice for the laboratory phase.

4.2 Flexure Compensation System: Overview

4.2.1 Motivation

The Flexure Compensation (FC) system is a subsystem designed to maintain constant image position on the detector as the spectrograph rotates. The reasons for including this system are explained below. Here we give a brief overview of the system. Further details of the full system that are background to the software design are given in Part IV.

As the spectrograph rotates, deformation of the optical supports causes image motion on the detector. Finite element analyses of all elements and tests of prototytpe grating cells and mounts suggest that cumulative passive tilt and rotation errors through the optical path will cause variations in the beam entrance angle to the camera of about $10''$ to $20''$. This is after careful attention to these errors at every stage of the design. A $20''$ variation corresponds to 2.5 px of *bulk image motion* on the detector. This is about 10 times larger than the image motion error budget (see below) and must be corrected by the FC system.

In addition to bulk image motion, the structural deformations induce small amounts of image degradation and distortion. These latter are within the error budget, and the FC system does not need to address these lower-order effects. Image scale may also vary with the temperature of the camera, but the FC system contains no correction for that. Finally, a $10''$ in-plane rotation of the gratings or detector corresponds to 0.14 px (rms) of image *rotation* (0.20 px at the edge of the field). The passive design stiffness appears to be better than this, and no active correction will be made for image rotation.

Two requirements set the stringent error budget for image motion. The clearest requirement comes from **image quality**, which is tightest in spectroscopic mode at high dispersion in the red spectral region. The requirement that image motion degrade the FWHM by no more than 10% sets a limit of 0.25 px rms during an exposure. Image sharpness for direct imaging and other types of spectroscopy set an image motion tolerance of 0.5 px rms. Both of these are 5–10 times better than the projected passive stiffness as described above, requiring some form of active beam steering.

The second requirement for the FC system comes from consideration of fringing in CCD detectors. This is described in Part I Chapter 4.3. A long-term goal of DEIMOS is to provide pipeline data calibration using a long-lived calibration database. The ability to reset the spectrograph accurately to any previous setting will be a key ingredient in creating such a long-lived database. An important limitation on long-term stability is variations in the flatfielding accuracy caused by **fringing** in the CCD detectors. In spectroscopy mode, the principle cause of fringing is spectrograph **flexure**. Fringing variations limit the ultimate accuracy of spectroscopic CCD flat-fields in the red spectral region and thus set the limiting S/N that can be achieved for sky-limited observations. As noted in Chapter 4.3 the improvement in flatfielding that would result from image stability at the 0.25 px level is not known for certain because other effects may contribute to fringing variations (e.g., pupil rotation and accompanying changes in the pattern of pixel illumination). However, it appears likely that fringing variations will be significantly reduced if the above stability goal of 0.25 px rms is met.

4.2.2 System Overview

The performance requirements for the FC system are as follows:

- Operate in both direct-imaging and spectroscopic mode with all filters, gratings, and grating tilts.
- Provide rms accuracy in each coordinate of 0.25 px.
- Have minimal impact on normal operations.
- Provide actuator range to accommodate the expected range of image motion during normal operation, and the expected image displacements caused by removal/reinsertion of the detector and/or all optical elements (e.g., for collimator aluminization).
- Provide guide signal updates every 15 seconds to the actuators.
- Monitor plate scale, image rotation, and focus.
- Provide continuous feedback to the observer and signal various alarm conditions.
- Store relevant data in the DEIMOS database every 15 seconds.

A system has been designed to achieve these goals. Details of the design are given in Part IV, which is a draft report on the FC system. The system consists of light piped into the focal plane from arc lamps via optical fibers. The images of these fibers are captured by two separate CCDs (the FC CCDs) that reside in the dewar next to the main CCD mosaic. A view of this arrangement is shown in Part IV, Figure 1.1. The images from the FC CCDs are read out every 15 sec, and their centroids are used to compute the motions of two control actuators. One of these tilts the tent mirror, which moves the image along the spectral dispersion direction. The second actuator is inside the dewar and moves the detector (and FC CCDs) along the slit direction.

The basic feasibility of the system has been established from optical distortion studies, flexure estimates, computation of the necessary photon fluxes, light integration times, and computation times, and prototype construction of both actuators. The basic command scheme and astronomer GUI have been outlined. Details of these and other items may be found in Part IV.

The major unknown in achieving the desired image stability goal is sensitivity of the camera focal length to temperature variations. The FC system can steer out bulk image motions but cannot correct for image size variations. The camera thermal problem is summarized in Part I, Chapter 4.3.

4.3 Calibration Stability

In this section we discuss open issues with respect to the calibration stability of the instrument. A draft calibration plan and associated software for DEIMOS are described Part III of this CDR Report. This plan must be considered tentative pending resolution of several unresolved issues that cannot be dealt with until the final instrument is assembled and tested. Most of these issues are familiar from other optical CCD spectrographs, but a few are peculiar to DEIMOS. This section summarizes these items under the headings of broadband direct imaging and spectroscopy. Narrowband imaging using interference filters is currently not contemplated with DEIMOS because the filters are located in the converging beam close to the detector.

For background on the optical layout of DEIMOS needed to understand the following sections, please consult Figure 4.1, which shows the overall optical layout, Figure 4.2, which shows rays traced through the camera, Figure 4.3, which shows the footprints of beams on the tent mirror, Figure 4.4, which shows the percent vignetting by the tent mirror as a function of position in the imaging FOV, and Figure 4.5, which shows the amount of light in the outer "ears" of the irregular Keck pupil outline.

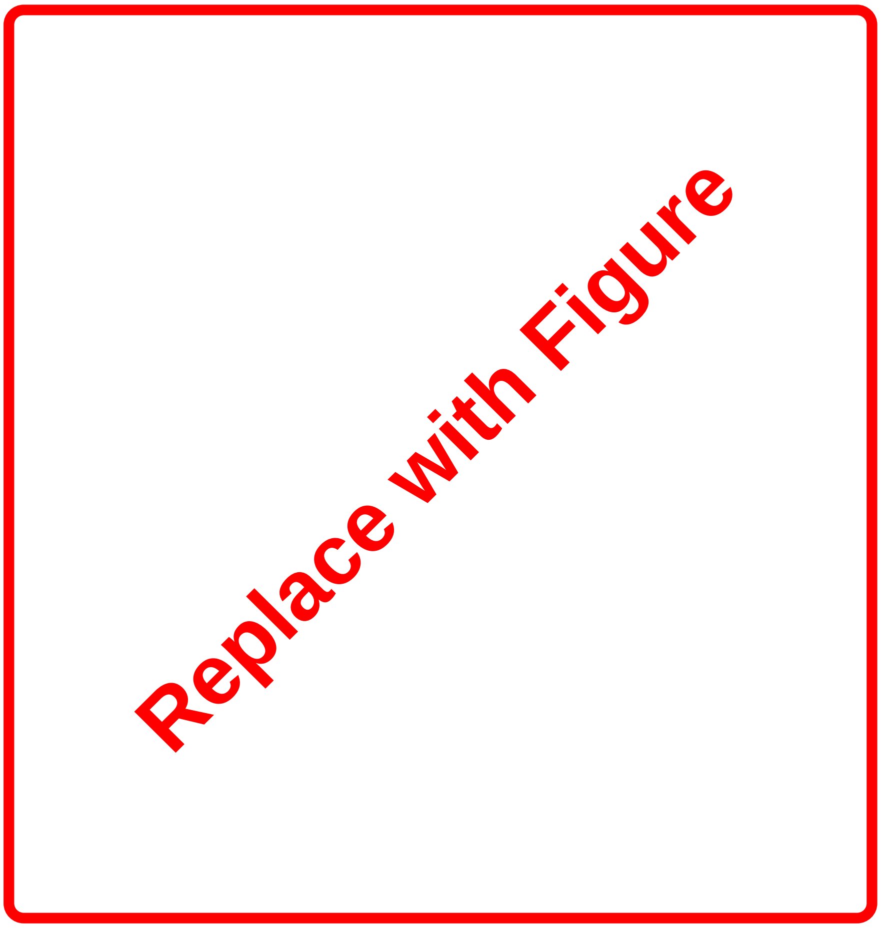


Figure 4.1: DEIMOS layout

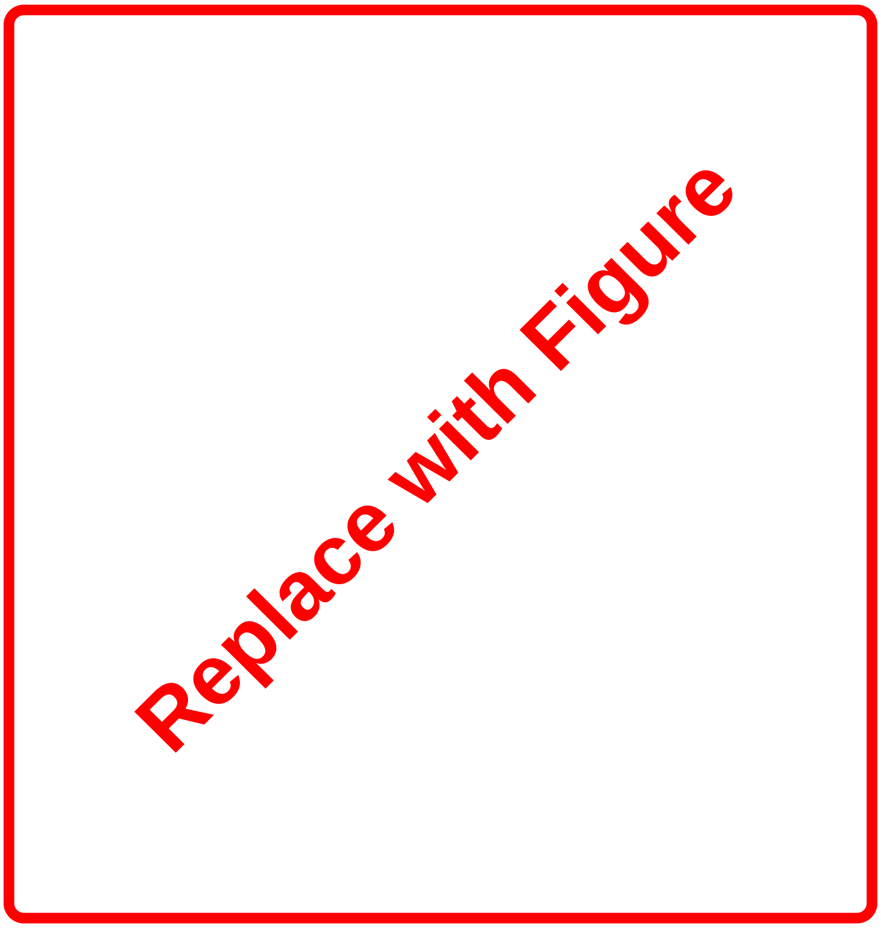


Figure 4.2: camera ray trace

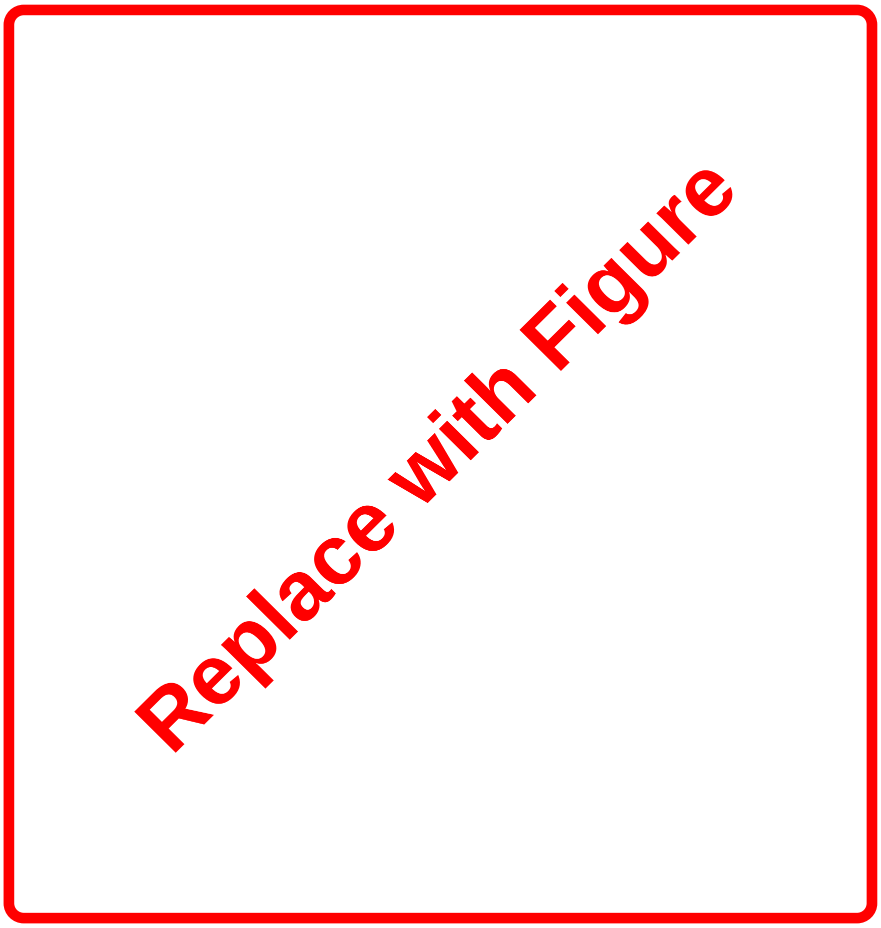


Figure 4.3: tent footprints

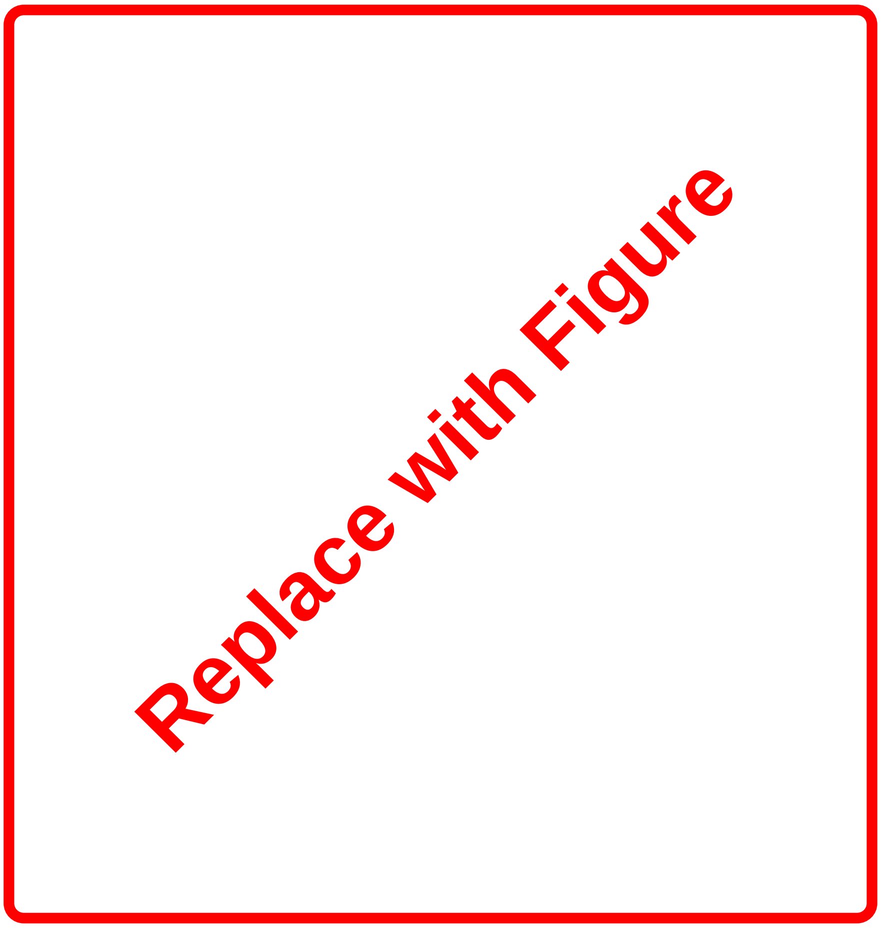


Figure 4.4: focal plane vignetting

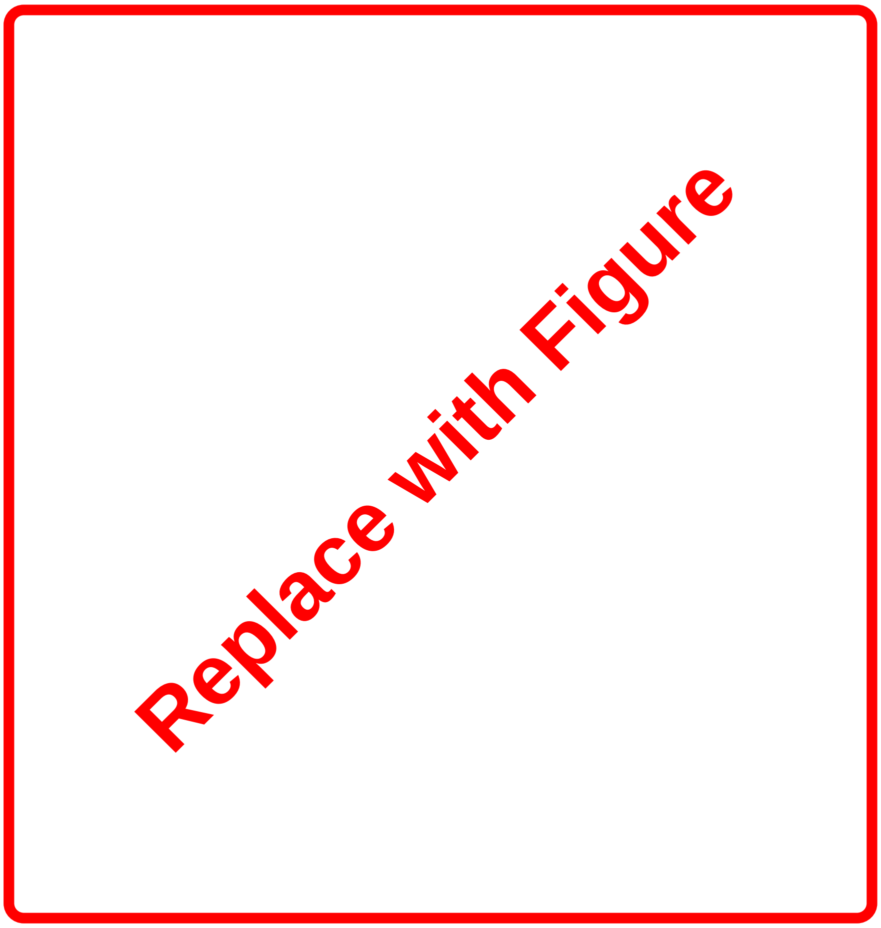


Figure 4.5: pupil vignetting

4.3.1 Broadband Direct Imaging

4.3.1.1 Fringing

The main obstacle to high-precision flat-fielding and flux calibration of CCD optical spectrographs is fringing in the thin CCD detector. The roughly 20-μm thick silicon membrane becomes transparent beyond 7000 Å, with the result that light that enters is trapped inside a resonant cavity. The internal reflection is quite efficient since the index of refraction of silicon $n \simeq 3.7$ near 1 μm; thus light tends to remain inside once it enters, unless the front surface of the silicon is treated with an efficient anti-reflection coating. Such a coating has the double benefit of increasing the amount of light entering the membrane in the first place and reducing the internal reflections that lead to fringing.

The interference condition is

$$N\lambda = 2nt\,\cos\theta, \tag{4.1}$$

where n is the index of refraction of Si (3.7 at 8300 Å), θ is the ray angle from normal within the silicon, t is the membrane thickness (about 20 μm for typical thinned devices), and N is an integer. For $\lambda = 8000$ Å, the above formula gives $N = 185$, so the interference order is high. That means that the interference phase angle rotates through a full 2π if λ changes by only $\Delta\lambda = \lambda/185 \sim 40$ Å at normal incidence. Fringe phase is therefore extremely sensitive to the precise wavelength of illumination.

Night-Sky Subtraction In broadband direct imaging, the bandpass is typically 1000 Å wide and contains about 25 2π interference cycles. The effective QE of the CCD varies depending on whether a given wavelength is at a peak or valley of the interference cycle. Beyond 7000 Å where fringing is a problem, the night-sky spectrum is dominated by the jagged emission-band spectrum of OH airglow. To remove this accurately requires using a background sky flat constructed from the *same* spectral-energy distribution. Typically this is done by taking the median at each pixel of many night-sky images. This median sky flat is scaled/offset and subtracted from each science frame before the remaining celestial sources are flat-fielded with a normal continuum dome or twilight flat.

At Keck on LRIS, fringes are non-existent in B and V, barely visible in R, and attain an amplitude of a few percent in I. With care, they can be removed to an accuracy of 0.5% in I and better at shorter wavelengths. However, this is well below the potential accuracy set by photon statistics, so we would like to do better in order to achieve the fundamental limit set by photon statistics for photometric accuracy.

Two barriers to improvement presently intervene. The ratios of sky emission lines across the I bandpass may vary with time, which changes the effective **fringe phase** (and fringe amplitude?) of each pixel. Second, the ratio of OH light to continuum light in the spectrum of the night sky changes as the sun and moon rise and set. The spectrum of the sky is continually changing, with the result that **fringe amplitude** is again continuously varying. For accurate work, it is necessary to obtain separate continuum and "OH" flats and scale them separately to match the observed fringe amplitude in the sky. This is routinely done at Keck and reduces the fringe amplitude error considerably (Guhathakurta, private communcation). We plan to accumulate such continuum and OH flats from twilight and dark-sky flats as part of the calibration plan (Part III).

Scattered Light Even after this treatment, sky residuals remain in LRIS flats at the sub-percent level. These are visible as large-scale undulations and/or tilts in the residual sky level. The origin of these errors is not understood. For red frames, they may come in part from fringe phase errors caused by variations in the relative strengths of OH or other strong emission features. In the blue, where spectral CCD response varies strongly with color, they may come from spectral mismatch between the night sky and the twilight flats used to get the large-scale response pattern. At all wavelengths, scattered light (from the wings of bright stars, stars beyond the field, and light scattered from the slit and dome) is also suspected to be a problem.

With respect to scattered light, DEIMOS will likely be at a *dis*advantage compared to LRIS because its optical design (Figure 4.1) puts the camera entrance aperture just behind the entrance aperture for direct imaging. Light can therefore scatter off structures located in front of the entrance aperture near the center of the focal plane and enter the camera mouth. This is the area occupied by the TV camera and its supporting structure. We are paying close attention to the blackening and baffling of the TV surfaces, but the ultimate flatness of the sky background due to scattered starlight cannot be predicted reliably.

Flux Calibration of Celestial Sources After sky subtraction, the celestial sources must be flat-fielded and flux calibrated. Flat-fielding will be done using broadband continuous dome or twilight spectra. Since most astronomical sources also have continuous spectra, fringing should not be a problem since there are ~25 interference cycles across the bandpass and phase variations should cancel out. For narrowband sources dominated by a single emission line, residual flat-field errors could amount to $\pm A$, where A is the fractional fringe amplitude at that wavelength, of order 10% in the red. We do not have plans to provide a general calibration for narrow-band sources.

Celestial sources vary in color and cannot all be flat-fielded by a single continuous broadband flat. However, the residual errors are a function of source color. Thus, global color errors can be removed by a standard photometric color term, and local errors that vary from pixel-to-pixel are expected to be small based on the sky flatness errors noted above.

4.3.1.2 Variable Vignetting

A special calibration problem arises at Keck owing to the **segmented nature** of the primary mirror, coupled with **pupil rotation** of the alt-az mount. Since different groups of segments are coated at different times, large-scale reflectivity variations of ~10% may exist across the optical pupil. Variations may be even larger on Keck II since the primary mirror is to be silver coated, and silver may degrade more rapidly and more unevenly than aluminum. Also, the outline of the segmented mirror is not circular.

These effects cause variations in system throughput as segments of differing reflectivity appear and disappear behind internal obscurations in the spectrograph. In direct imaging mode, the main source of internal vignetting is the straight edges of the tent mirror, which obscure up to 10% of the light at the inner and outer boundaries of the imaging FOV (see Figure 4.3). Throughput variations due to this vignetting are of two types: (1) variations due to the irregular boundary of the primary mirror, which causes the visible area to vary by approximately ±0.8% as the pupil rotates behind the tent mirror, and (2) variations due to non-uniform segment reflectivity. An upper limit to the latter may be estimated by considering the edge of the imaging FOV, where pupil vignetting by the tent mirror is 10%. If the reflectivity of a vignetted portion of the primary were 20% lower than average, throughput would increase by 2% when this part of the primary was obscured.

To summarize, throughput variations amounting to a few percent for direct imaging will exist at the edge of the imaging FOV. A fraction $< 1\%$ is due to the irregular edge of the Keck primary and could be calibrated out from the known rotation angle of the pupil. The remainder arises from reflectivity variations on the primary and will vary with time as the segments are re-coated. Regular maintenance of the primary mirror is desirable to minimize this latter class of variations.

4.3.2 Spectroscopy

4.3.2.1 Fringing

Fringing occurs in spectroscopic mode as well as direct imaging, with the difference that any one pixel is now illuminated by light of a single color. The response of a single pixel is given by

$$I = I_0(\lambda)\ (1 + A\cos\eta), \tag{4.2}$$

where A is the fractional amplitude of the fringe pattern and η is the fringe phase at that pixel, given by $\eta = 4\pi tn(\cos\theta)/\lambda$ (see equation 4.1). If CCD thickness and θ were both constant along a spectrum, I would trace out a perfect sine wave. In practice, θ varies smoothly with wavelength due to the optical design, but thickness varies randomly due to small-scale variations in the thinning process. The resultant contours of constant η have complex topologies resembling wood grain contours.

Flat-fielding problems arise when λ on a pixel differs between the flat-field image and the science image. Slight variations are serious owing to the very high order of interference ($N \sim 185$). The change in response is the derivative of equation 4.2 and thus is a maximum at the zeroes of the fringe pattern. If the fractional change in response is $\Delta I/I$, then the largest change is

$$|\frac{\Delta I}{I}| = 2\pi AN\frac{\Delta\lambda}{\lambda}, \tag{4.3}$$

where N is the interference order, $\Delta\lambda$ is the wavelength shift, and $\cos\theta$ has been set equal to 1 for normal incidence. (Note: for long wavelengths where the CCD is essentially transparent, AN is essentially constant, so that near 10000 Å, a thick CCD doesn't help at all.) A typical flexure shift for Cassegrain spectrographs is 5 Å, and fringe amplitude A is ~ 0.1. Inserting into equation 4.3 yields $|\Delta I/I| = \pm 0.01$, which is similar to the typical flat-fielding fringe errors seen in LRIS spectroscopic images at Keck. Since night sky OH lines produce more than 100,000 photons over regions of interest, they should subtract to better than 1% by photon statistics. Fringing prevents this, with the consequence that fringing is again the principal factor that prevents large telescopes from reaching their theoretical shot-noise limit beyond 7000 Å.

4.3.2.2 Variable Vignetting and Slit Losses

Variable vignetting and pupil rotation will again cause throughput to vary in spectroscopic mode, with the extra fact that additional, significant vignetting occurs at the grating and camera mouth. The beam footprint on the gratings is ellipsoidal, and light at the ends can be lost at large tilt angles. The large 8×12-in grating used in the red at high tilts creates a long pupil whose ends miss the camera mouth, which is only 11.5-in wide. The product of these obscurations can amount to $\sim 25\%$ of the light.

By analogy with the tent mirror vignetting in Section 4.3.1.2 it is clear that segment reflectivity variations coupled with pupil rotation will again cause throughput variations. The effect is larger now because a larger fraction of the primary mirror is vignetted. If the maximal 25% vignetted area were 20% lower in reflectivity than the rest of the primary, the net throughput increase would be 7%. This is probably an upper limit, but variations could clearly amount to a few percent, especially at large grating tilts (red wavelengths).

The 6×8-in gratings are narrower (6.05-in) than the full "ear-to-ear" diameter of the beam (6.35-in). Pupil motion on the gratings will therefore cause variable light loss. Specifications state that the alignment of the beam on the grating should remain constant to within 0.05-in, which would cause a throughput variation of less than 1% at all wavelengths. However, alignment to this level is considerably better than has been achieved for LRIS or HIRES on Keck I. This effect will vary systematically with elevation angle and could be calibrated out.

The final source of error for flux calibration in spectroscopy mode is slit losses owing to atmospheric refraction. These depend on the detailed light distribution in the object and will vary with wavelength. We plan to provide information on this for the observer but do not expect to make any correction for this effect.

To summarize, total fluxes and spectral energy distributions in spectroscopy mode will be in error by a few to many percent, depending on the airmass. Accurate measurements will require the use of large apertures and careful observations of standard stars at similar elevation angles and pupil orientations.

4.3.3 Reducing Fringing Errors

Several aspects of DEIMOS' design are intended to reduce residual fringing errors:

(1) The **Flexure Compensation System** is designed to maintain wavelength on a given pixel constant to 0.25 px rms (see Part II, Chapter 5). This virtually eliminates fringe phase shift errors due to change in λ.

(2) Efficient **anti-reflection coatings** are being sought for the CCDs. A good coating not only promotes transmission of light into the silicon but also reduces trapped light and consequent fringing. Poor coatings have fringe amplitudes $A \sim \pm 15\%$. We are aiming for a good coating like that in LRIS, which has $A = \pm\ 5\%$.

(3) **Thicker devices** of **high-resistivity silicon** are being explored for DEIMOS' red CCDs. Thicker silicon pushes the onset of fringing (i.e., transparency) to longer wavelengths. A program to develop such detectors is underway at MIT Lincoln Laboratories (see DEIMOS Detector CDR, May 20, 1997). Fringing begins redward of 8000 Å in these detectors. Also, CCD thickness variations are much smoother (and smaller) in the MIT/LL thick CCD, compared to the usual thin CCD.

Note that items (2) and (3) will reduce fringe errors in direct images as well as for spectroscopy.

Despite these efforts, complete removal of fringe errors from flat-fields cannot be guaranteed:

(1) Because of the extremely high order of interference, fringe phase may be sensitive to unsuspected physical processes. For example, fringe phase varies strongly *across* the pupil at a given wavelength. This occurs because beams from different parts of the pupil impact the CCD at different angles, causing θ to vary. At the edge of the detector, θ varies (within the silicon layer) from about 0° to 8°, and at the center the variation is 0° to 4°. These ranges correspond to 1.7 and 0.4 interference orders, respectively. It is possible that fringe-phase pupil variations could combine with segment reflectivities, vignetting, and pupil rotation to create unpredictable variations in total response.

(2) Thermal effects on camera plate scale could alter the mean wavelength on a pixel by more than the stability goal of 0.25 px rms, thereby confounding the FC system. Such changes in plate scale cannot be corrected by the FC system, which merely steers the beam. Maintaining constant wavelength at the edge of the detector would require that the camera temperature be held stable to better than 0.5 Kelvin. We are attempting to design the camera lens barrel to adjust the lens spacings passively to compensate for temperature changes, but the success of this approach is not yet known.

(3) The plan in Section 4.3.1.1 to subtract sky from red broadband im-

ages requires a suite of both OH and continuum sky flats. It is unrealistic to expect an observer to obtain a complete suite of such flats on each run, let alone each night. This sky-removal technique will work only if the instrument is stable enough to permit assembly of a longterm calibration database. We are endeavoring to make the instrument quite stable but the success of this approach is not yet known.

Appendix A

Response to Preliminary Design Review

A.1 Introduction

We received our copy of the report of the DEIMOS Software PDR review board in late April 1996. Our initial response to that report was contained in the software section (2.4) of DEIMOS Quarterly Report No. 8, which was issued in mid-July. (That section is reproduced at the end of this Appendix.) That response described what we intended to do regarding the issues raised by the review board.

This appendix represents an updated response that describes what we have done during the 15 months since the PDR to address those same issues. For ease of reference, points will be addressed in the order in which they appear in Steve Kent's email of Apr 23, 1996 (a concatenation of the reviewers' notes).

A.2 List of Important Issues

The review board identified the following 5 items as most important:

- The choice of CCD controllers is the biggest unresolved issue. The panel recommends the second-generation SDSU controllers be used if at all possible.

- Quick Look display: Be prepared to simplify requirements.

- Databases: Need a better definition of requirements and specifications. Need to understand the interface with CARA better (both for construction and operation) since some proposed functions extend beyond DEIMOS.

- Management: Be prepared to prioritize requirements in case descoping of effort is required to stay within budget

- Guider: This is a CARA responsibility, yet DEIMOS relies on it for efficient operation and has a detailed "wish list" for functional requirements. DEIMOS should take the lead in getting CARA to commit to providing that functionality or else develop contingency backup plans.

We have decided to use the 2nd generation SDSU controllers, as advised. We have received and tested our first set of 2nd generation controller boards, and the results of these tests were reported at the DEIMOS CCD review held May 20, 1997. Copies of the report from that review are available, and selected chapters (3: CCD System Overview, 5: Science CCD Controller, and 6: Data Capture) of that report are available via the WWW at the following URL: http://www.ucolick.org/~kibrick.

The choice of quick-look display has remained a difficult one. While we hope to use the NOAO "PIE server" distributed real-time image display tool (currently under development) for DEIMOS, it is not yet available and may conceivably not be released in time for commissioning. Our interim plan is to use Keck Figdisp for engineering tests, and as a fall-back display for commissioning. Meanwhile, CARA personnel are investigating the latest release of ESO/RTD (which was used for demo purposes at our PDR) as a candidate for the new Keck Guider display. RTD may emerge as an alternative fall-back QL display if CARA's results are positive. The review board advised us to be prepared to simplify the requirements for this display. We are prepared to do so, and the contingency of commissioning with Figdisp will represent an implicit simplification of those requirements.

The review board asked for a better definition of the requirements and specifications for the database, and suggested a need to better understand the interface with CARA. The CDR documents include a detailed specification of database function and schema design as requested. Details of procedural and technical dependencies on CARA are still under discussion, so we cannot at this time state how much unification of telemetry, environmental and weather logging, etc. will take place at Keck by the time of DEIMOS commissioning.

With respect to management, the board recommended that we be prepared to prioritize requirements in case descoping of effort is required to stay within budget. While we are prepared to do so, a prioritized list of requirements has not yet been generated. As described later, obtaining a sufficient allocation of software staff time is currently a more serious management problem than budgetary constraints.

The board suggested that DEIMOS should take the lead in getting CARA to commit to providing the guider functionality that DEIMOS requires. Lick and CARA have recently begun a joint effort (headed by Jerry Nelson) to develop a new Guider system for Keck. DEIMOS project staff will be minimally impacted by this effort, but will have input into the new guider specification. Another member of the Lick scientific software staff (Deich) who is not currently assigned to the DEIMOS project will be assigned to this effort at approximately 15%-time.

A.3 CDR Suggestions

The review board made five suggestions for the CDR:

- Provide a top level diagram showing the major subsystems being developed by DEIMOS and the interconnectivity
- Provide a data flow diagram
- Provide detailed requirements for the computing hardware
- The CCD controller issues should be resolved by the CDR
- Bring back most (if not all) of the same reviewers.

High-level functional diagrams of the DEIMOS hardware/software system and certain procedural elements (like slitmask fabrication) are provided in the CDR documents, as requested. These diagrams show data flow as well as interdependence. While additional work remains to provide a complete detail-level interconnectivity diagram of all subsystems, that work is in progress.

Decisions about actual brands and models of computing hardware will be made as late in the project as possible, because we have taken considerable effort to ensure that the software we develop is and will continue to be portable to most likely platforms. However, we will provide at the CDR our performance and memory requirements for whichever vendor's product is eventually purchased.

The CCD controller issues have recently been resolved, and a second-generation SDSU system will be used. A detailed hardware design for the CCD controller was presented and accepted at the DEIMOS CCD Review held on May 20. The purchase order for this hardware has been issued, and the first shipment of boards should arrive in July. In the interim, software development will proceed using the initial set of evaluation boards that we received in March. We expect to receive the final shipment of controller boards in September.

We have re-invited all of the reviewers from the PDR panel, as requested.

A.4 Individual Reviewers' Comments

We have attempted to abstract the main points of each of the individual reviewer's comments. These are shown as italicized text. In cases where the same comment or concern was raised by more than one reviewer, usually only the first occurrence will be listed and answered.

A.4.1 George Jacoby

- *Take the lead in an attempt to unify several groups wrestling with nearly identical display client needs, and organize a community development project.*
- *Begin ASAP to enhance an existing display package*

The project has taken effectively both of George's suggestions with regard to image display software. We have identified an existing image display project involving a large subset of the astronomy community (NOAO RTD) and although DEIMOS cannot be said to "take the lead" in this effort, we are actively participating in and monitoring the progress of the project. Doug Tody came to Lick last December for initial discussions about this collaboration, and Steve Allen paid a return visit to NOAO in March. Results of these discussions are described in section I.4.1.

In the meantime, we are also enhancing an existing product (Keck Figdisp) for immediate engineering use and as our insurance policy should the NOAO software not ship in time for our needs. Keck Figdisp is the image display server currently used by three of the first-generation Keck instruments (HIRES, LRIS, NIRC). It was recently ported from SunOS to Solaris by John Cromer of CIT (at no cost to the DEIMOS project) and has subsequently been ported to Dec Alpha/Digital Unix and Intel-586/Linux by

Steve Allen. We do not intend to enhance Keck Figdisp to meet the full range of DEIMOS image display requirements at this time, and will only make the minimal enhancements necessary to allow its use during the engineering and test phase of the project. If, by October, it appears that delivery of the NOAO RTD will be significantly delayed, then further enhancements to Keck Figdisp will be considered at that time.

- *It is unclear which goals of the database project are needed to take science data.*
- *A subset of database specifications should be prepared which places a minimum of demands on the database development team.*

The scope and utility of the database component of DEIMOS have been more accurately defined. Our work assisting CARA in the automatic ingestion of STB (an image archiving system) headers into a relational database has proved useful in clarifying DEIMOS' needs and our practical capabilities in the realm of image location and retrieval. The database as repository of authoritative keyword information and instrument design information has proven practical (as the many pages of generated documentation in the CDR packet may show). The role of the database in slitmask fabrication and documentation, pipeline data reduction, and online information and help systems has been realistically defined.

There should be a supported Keck "approved" pathway to deriving slit mask coordinates.

This is a worthwhile goal, but requires coordination with CARA. No action has yet been taken on this suggestion.

The panel suggest a more convenient approach (than HTML) to printable documents be provided...Possible ideas are Frame, LateX, or other word processors with HTML converters.

The DEIMOS software team has adopted LaTeX for all its formal documentation. The keyword documentation generators produce both HTML and LaTeX directly; handwritten documents are written in LaTeX and converted to HTML using publicly-available tools. A standard format is imposed on all project documentation, and all handwritten documents are maintained under CVS for revision control and ease of access.

CCD controller hardware requirements (such as matching amplifier gains to 1%) should be relaxed if equivalent functionality can be provided via hardware.

Software equivalents will be implemented if the corresponding hardware functions prove too challenging. However, preliminary tests with MIT Lincoln Lab CCDs and the second-generation SDSU controllers indicate that amplifier gain matching to the 1% level may well be achievable.

A.4.2 Steve Kent

- *Doubts labour estimates; suspects they are low by a factor of 2.*
- *Wants prioritization of goals.*
- *"It is not clear to me who, if anyone, is responsible for managing the day to day activities of the programming group. No one works full time on DEIMOS - everyone has responsibilities to support previously developed systems or commission current systems. It would be advantageous to have a manager who will ensure that DEIMOS activities do not slip because other activities assume higher priorities by default"...*

Of all the suggestions made by the PDR review board, this last item is the one that should have been taken more seriously. Kibrick is the one responsible for managing the day to day activities of the Lick scientific programming group overall, and is also responsible for managing the DEIMOS software effort. In addition, Kibrick has been deeply involved in the selection and testing of the second-generation CCD controller boards and in the overall hardware design for the DEIMOS CCD controller. He is also responsible for the management of the software effort for the Keck ESI Spectrograph, as well as several recent upgrades to the Keck HIRES Spectrograph. One of these (the HIRES image rotator upgrade) took considerably more effort than was budgeted, partly because it required additional effort to adapt to the changes in the Keck Telescope control interface that resulted from retrofitting the Keck-2 system onto Keck-1 at the beginning of this year. Further, Kibrick has ongoing responsibilities to support both telescope and instrument control systems for both the Shane 3-m and Lick 1-m Telescopes at Mt. Hamilton, including completion of the commissioning of the MOS and PFCAM instruments for Mt. Hamilton.

While all of the members of the DEIMOS software team have obligations to other projects, Allen and Clarke have been able to devote the majority of their time to the DEIMOS software effort during the past year. Phillips has been able to devote 50% time, and Tucker 40%. (In addition, Tucker has had assistance with the Galil Motor Controller software from Jim Burrous of the Lick Electronics Shop.) For the period October 1996 through May 1997, the percentage of Kibrick's hours devoted to DEIMOS was 21.4%, with the bulk of that time expended in the months of April and May.

While excellent progress has been made by the other members of the DEIMOS software group (as evidenced by their respective chapters of this document), many of the overall software management functions (e.g., documentation of top-level software design, overall software planning, establishment of standards, budgeting, scheduling, and reporting) have not received adequate attention due to Kibrick's obligations to other projects. This lack of attention is readily apparent in those sections of this CDR document for which he is responsible. While UCO Director Miller has recently made efforts to temporarily disengage Kibrick from non-DEIMOS activities, this disengagement did not occur until April 1, and is currently slated to expire following the DEIMOS software CDR. In addition, between April 1 and late May, most of Kibrick's time (while directed towards DEIMOS) was spent on preparations for the DEIMOS CCD Review, at the expense of preparing for this CDR.

In hindsight, it is obvious that our expectations regarding the fraction of Kibrick's time that would be available for DEIMOS software project management were overly optimistic. As Kent so aptly noted, DEIMOS activities (in this case, management activities) did slip because other activities (typically those impacting scheduled telescope time) assumed higher priority. Clearly, the current arrangement is not adequate, and a greater allocation of effort towards this area is urgently needed. This, and the question of overall software staffing levels for the DEIMOS project, is one of the more urgent items that needs to be addressed by this CDR, and for which suggestions from the review board are particularly desired.

The availability of office space is a related concern which impacts our decisions regarding staffing. Both the overall DEIMOS project manager (Cowley) and Kibrick have felt for some time that the combined software demands of DEIMOS and ESI indicated the need for additional software staff. While that need has been partially offset by the transfer of Jim Burrous from Mt. Hamilton to Santa Cruz, Jim is attached to the Lick Electronics shop and his primary responsibilities involve electro-mechanical testing and integration. However, Jim has made very significant contributions to the Galil motor controller firmware used by DEIMOS, ESI, and PFCAM, and we should plan to make as much use of his software talents as possible.

Nonetheless, both DEIMOS and ESI could benefit from the addition of at least 1 more programmer, or even a summertime student worker. However, the scientific programming group (currently housed in temporary offices in Thimann Labs) has no additional office space in which to locate a new hire. In fact, our repeated requests for additional office space have been consistently denied due to a global lack of office space within the Observatory and the campus in general. Our group will be moving back into the Natural Sciences 2 building (the location of our offices prior to November 1995) now that the seismic repairs on that building are complete, and this might provide a window of opportunity for acquiring one small office.

However, even this is doubtful, since current plans call for cutting the square footage assigned to the Observatory by 1900 square feet. This severe shortage of office space needs to be considered when proposing any plans to augment the software staff available to work on the DEIMOS and/or ESI projects. It should also be noted that negotiations involving the allocation of office space have consumed an inordinate amount of staff time throughout the Observatory. While these negotiations have had negligible impact on most of the Lick Technical Facilities (since they are housed in buildings in which space is not being reallocated to other campus units), they have represented a significant drain of time on the scientific programming group (especially Kibrick) and on the staff of the CCD Lab (especially Stover).

An area that did not appear explicitly on any budget but which might absorb a lot of time is generic support for infrastructure

Kent was also correct on this point. The DEIMOS software group has made a significant investment in software infrastructure during this past year, as should be apparent from the relevant sections of this document. While this was an extremely worthwhile investment, it certainly involved more hours than we would have predicted at the PDR. While the most significant portions of this investment have been completed, we will be careful to budget for an on-going commitment to infrastructure-related activities.

The biggest single item that I found missing in the PDR documentation was a diagram showing all the subsystems with their interconnectivity... Such a diagram is a must for the CDR.

Considerable effort has been applied towards this task, especially towards development of effective tools which utilize information from the DEIMOS design database to automatically generate such diagrams. Major portions of such an overall diagram have been completed using these tools, and are included in this document. Additional diagrams will be presented at the CDR. However, it is clear that further effort is still required to provide a truly complete overall diagram, and the absence of such a diagram from the initial release of this document is entirely due to insufficient availability of Kibrick's time.

It was stated during the review that files would be used to exchange information if the database were not available...The use of two databases, one for "operational" needs and one for offline scientist queries is also the approach chosen by Sloan.

In most cases where access to database-resident information is required for processes such as slitmask fabrication or design, pipeline data reduction, etc., a level of isolation has been introduced; authoritative data are maintained and resident in an online database, but the applications run against human-readable extract files (some in standard formats such as FITS table extensions). In this way the applications are decoupled from dependence on particular database characteristics; only the relatively simple extract generators need be changed if the database engine or requirements change. These extract files can, at need, be generated by other means, or manually customized by users. We are considering this policy of isolation not as an interim measure, but as a permanent design feature. Remote access to database information can be arranged by access to extract generators via Web pages, with the generated extract downloadable to the remote user's site.

A very useful activity is to construct a data flow diagram for the whole software system.

We have tools in hand which create, maintain, and generate paper copy of information flow diagrams. (See diagrams included in CDR documents). We concur that all major subsystems should be sketched, examined, and refined before code is written. While considerable effort has been applied to this task, additional work remains.

It is not clear if the display software is intended only for use at the mountain or if it should also run on user's home workstations.

Our current plan is that it should be usable at both. Assuming the NOAO RTD effort is successful, that display server should ultimately become available as a supported part of IRAF, and thus should be available on user's home workstations.

Keck Figdisp should also run on user's home workstations, and it has been ported to several platforms including SPARC/SunOS, SPARC/Solaris, DEC Alpha/Digital Unix, and Intel-586/Linux. However, convenient GUI access to Keck Figdisp requires purchase of a DataViews runtime license which cost approximately $100 per machine, and which is available for all of the above platforms except for Linux. In the absence of DataViews, all Figdisp functions are still accessible via mappable function keys.

How much input will the user community have into the specification and requirements for this (Quick Look display) and other functions such as spectrograph control? (Is the opportunity passed?)

While the opportunity is not yet passed with respect to the Quick Look display, it is not clear how to obtain greater input from the user community. Many have been invited to the software CDR, but few have chosen to attend.

The same was true of the PDR. All of our PDR documents have been and will continue to be accessible via the WWW, as will the documents from the CDR. Kibrick has sent surveys to all Keck users at UC requesting suggestions for improvements/additions to Figdisp, but has received only a handful of responses. We consider such input extremely valuable, and would like to find a way to encourage further participation by the DEIMOS user community. Suggestions are solicited from the review board regarding how best to obtain additional input and review from the DEIMOS user community.

The control room layout plans assume that the observer is at the telescope, yet a desire has been expressed that remote observing be supported.

Once the instrument is fully commissioned, our long-term plan is that DEIMOS will be operated remotely from Keck HQ in Waimea. Upgrades to the network bandwidth between the Keck summit and Waimea are now in progress that should make this feasible.

Sloan has used CVS for code management for several years...

The DEIMOS software group thanks the review board for recommending CVS. We have converted our pre-existing SCCS-based source repositories to CVS, and are using CVS to maintain all DEIMOS software and documentation, including the chapters of this document.

Configuration management is a thornier problem...and means tracking not only versions of a particular software package, but also tracking the dependencies of one package on another, possibly for multiple platforms simultaneously.

Configuration management is indeed a thorny issue for instrument control systems. Lick and CARA personnel are collaborating on a revised build and release structure for the CARA KTL library code. We hope that this project will ensure easier porting, co-development, and configuration management of KTL-based instrument control systems. If there is sufficient interest, the current state of this collaboration can be discussed at the CDR.

The biggest potential problem in supporting multiple platforms may be the interface between the VME crate and the instrument computer; although this interface uses "standard" network protocols, problems peculiar to each platform may crop up.

This is one of several reasons why our default plan is to run Solaris on the CPU in the VME crate. In doing so, the VME crate becomes indistinguishable (from the point of the OS) from a standard Sparcstation 5. Only if serious problems develop with this plan will we switch to running VxWorks in the VME crate, in which case this issue of "peculiarity" may become relevant.

The hardware requirements were not discussed. They should be a sig-

nificant item for the CDR...One should buy a system with plenty of CPU, memory, and disk. One should avoid saving on hardware at the expense of making the software more complicated.

Hardware requirements will be presented at the CDR. Certainly we would like to have plenty of compute resources, but we have a fixed budget for the instrument computer hardware. By deferring this purchase as long as possible, we hope to acquire the most powerful machine we can afford.

If funding is found for the second beam, will that funding include upgrades to computing to accomodate the doubling of the data volume, or is the current hardware supposed to support two beam?

Depending on the CCD readout rate (which depends on the characteristics of the CCDs ultimately selected for the DEIMOS mosaic), the current system may be barely adequate to support both beams. In seeking funding for the second beam, we will attempt to acquire funds for additional computing hardware to more adequately support the second beam. However, since the level of funding that might ultimately be obtained for a second beam is unknown, we can't provide a firm answer to this question.

A.4.3 John Cromer

Of all the various processes that make up the HIRES and LRIS software suite, the MUSIC "traffic" process has been a rock. It is easily the most robust and stable piece of control software we have running now.

The DEIMOS software team has elected to stay with the music/traffic instrument control software rather than migrate to the newer EPICS system currently in use by CARA for telescope control. We concur with John's assessment of music and traffic.

WCS Issues and FITS keywords: Seems to imply an enormous number of keywords for the various file headers...

Steve Allen's latest revisions of FITS WCS representation use FITS table extensions rather than enormous numbers of indexed keywords. This reduces the WCS problem to the case for existing single-CCD images.

Spectrograph Control:

- *It is a good idea to have a complete log of the serial transactions which pass back and forth between the control computer and the motor controller.*

- *Care must be taken when swapping in spare motor controllers that may be of a different revision level than the unit which failed.*

- *The DEIMOS software that runs directly on the motor controller should also be kept in the files on the main development computer and subject to the same revision control and source code management procedures as the DEIMOS Unix software...The relationship between this controller code and the Unix control software control has the potential to raise maintenance difficulties after the instrument is handed off to CARA.*

Multiple levels of transaction logging are being provided for the commands between the control computer and motor controller. Keywords have been implemented to track the revision level of the software that runs in the computer. The DEIMOS motor control software will be maintained under CVS. Automated procedures have been implemented for downloading the firmware from the control computer to the motor control.

I do not believe the project should be delayed in order to meet the 1% gain-matching requirement or dual-amp readout.

It won't. The CCD controller design presented at the DEIMOS CCD review on May 20 provides for a phased-implementation approach by which we can initially support single-amplifier readout and then upgrade to dual-amplifier readout at a later date without the need for any rewiring of the CCD dewar or the CCD controller. Our initial purchase order to SDSU for CCD controller boards includes only enough video processing boards to support single-amplifier readout. A more detailed discussion of these issues can be found in Chapter 5 of the document from the DEIMOS CCD review, which can be obtained at http://www.ucolick.org/~kibrick/deimos-ccd/chapter5.html.

Data Storage Formats: Observers are going to spend more time trying to figure out the data storage formats than they did collecting the data. Adequate software tools should be supplied to address this complexity... The team should be prepared to back off some of the requirements to meet the delivery schedule.

During the ensuing year we have conferred with other teams constructing mosaic detectors (notably NOAO). It is now clear that the image section from each CCD amplifier should be stored in its own IMAGE extension. I.e., there should be no archival image format where the separate image sections are combined into a single FITS array. This removes the need for the extremely complex set of indexed keywords which describe the geometries of image sections within a combined array.

The display server should deal sensibly with conflicting client programs and be able to distinguish between a real-time client and a non real-time one...Does a new image begin to overwrite the screen while a quick-look *procedure is still processing the previously-displayed image?*

We have not yet explicitly addressed this issue in our design, and this is a question on which additional input from the DEIMOS user community would be highly desirable. Our instinct is to abort any such in-progress procedure and return an error condition from that procedure so as to not delay display of the newly-available image. However, this may not be what some observers desire. Further suggestions from the review board regarding alternative solutions to this problem would be appreciated. In particular, it would be useful to know the current plans for how the NOAO RTD server will address this problem.

- *The database management system should not significantly impact the performance of the real-time data acquisition system*
- *Operationally, I am concerned about its impact on the CARA staff after it is installed*
- *Will there be any QA/QC procedures to insure the quality of data in the database?*
- *Who maintains the database system, CARA or Lick?*
- *What are the manpower requirements for maintenance?*

The degree of isolation between database and applications described above should address John's concern about impacts on observing-time performance. Software has been implemented and tested to perform sanity/consistency checks across those portions of the database which maintain the system design description and the keyword database. Where feasible, we will attempt to implement similar consistency for the other portions of the database, and to provide proper documentation to insure the CARA staff are able to run these as necessary. Some of these consistency checks may be run automatically via cron jobs. We have tried in the CDR document to make more accurate estimates of the ongoing commitment required of both Lick and CARA staff to keep the information management system functional.

A.4.4 Jill Knapp

Our upfront, highlighted, can't-emphasize-it-enough recommendation is to manage the code development via CVS.

As mentioned above, the DEIMOS team has settled on CVS as our source code control standard and we are well pleased with it.

The presentation should have contained a data flow diagram...

As mentioned above, data flow diagrams are included in the CDR, although this effort is not yet complete. It should be noted that these diagrams are well integrated with database so that they can be properly maintained as the software evolves. The software to generate these diagrams from the design model in the database will be a part of the delivered system, so CARA should be able to regenerate these diagrams on demand.

Possible compute platforms should be specified and a proposed set of benchmarks presented at the CDR.

Benchmarks of typical image reduction operations are included in the CDR.

Quick Look: The proposed combination of a very heavily binned image with a panned/zoomed mouse-driven display of small regions at full pixel resolution might not be adequate for either engineering tests or commissioning. It might be useful to also produce instrument statistics...

We do not see an obvious alternative to this approach, and this combination remains our default plan.

Keck Figdisp, which we plan to use for initial engineering tests, already provides many of the statistics capabilities requested. We plan to implement a more generalized statistics process as part of the data capture pipeline.

Simulated images might be very useful for testing the software.

Allen has generated simulated images using the IRAF artdata package, and these are being used to the performance of Keck Figdisp on DEIMOS-sized images.

A.4.5 Al Conrad

CCD Readout Software: The biggest software risk area seems to be the DSP and the low level C programming required to readout the mosaics...The DEIMOS team should focus on developing the DSP and low level C code needed to operate the second generation SDSU controller.

For this reason, a significant fraction of Kibrick's time was devoted to tests of the second-generation SDSU controller boards following their delivery in late March, in order to have results in time for the May 20 DEIMOS CCD Review, and to ensure that the major purchase order for this equipment could be issued prior to a May 31 fiscal closing deadline for orders of this size.

Our existing engineering software tools for operating the SDSU controllers have been updated and tested with the second-generation controllers boards. We can now successfully program and generate valid CCD clocking waveforms and bias voltages using the second-generation boards, and have used a high-speed oscilloscope to verify the proper operation of the video processing signal chain. Our VME crate (currently SunOS-based) is reliably communicating with the second-generation SDSU timing board and utility board. We are sufficiently confident with this new hardware and with our ability to generate DSP software for it that we have sought (and obtained) from the DEIMOS CCD Review Board permission to place a large order for second-generation SDSU controller boards.

More Prototyping Encouraged

- *Put more effort into prototyping user interface tools and iterating the prototype with a selected set of future DEIMOS users.*
- *The image display tool should be tried out on simulated images of multislit spectra...Ask users to use the tool for specific investigations.*

The DEIMOS team has used the working HIRES instrument as a prototyping test platform for the GUI design tools. The DEIMOS GUI software has been adopted as the standard for ESI and PFCAM as well, so that there will be deployed interfaces for these projects prior to DEIMOS commissioning, and the DEIMOS UI will benefit from our experiences with the other two instruments. The interface software, now in its second major release, has been upgraded to run in "no-KTL-service" mode so that prototyping for DEIMOS can precede access to the working instrument. Colour pictures of some early DEIMOS interface designs were distributed at the May 12th Keck Software Coordinating Committee (SCC) meeting, and a more complete set are in the CDR. We concur that user interface prototyping and early user feedback are very important.

Initial tests of Keck Figdisp using simulated DEIMOS images are in progress by the DEIMOS software staff. However, we have not yet had potential DEIMOS users take this system on a test drive. This needs to be done soon.

Mask Generation: CARA and DEIMOS programmers should work together to achieve as much commonality as practical between CARA's "sky" planning tool and DEIMOS' mask planning tool, which have much in common. Both should use common formats for star lists.

This is a desirable goal, but will require some commitment of resources from CARA if it is to be achieved. In particular, it would be useful if Al Conrad (the author of CARA's "sky" program), could spend some time in Santa Cruz to work with Drew Phillips (the author of the DEIMOS mask planning tool) to help achieve such commonality. Methods for funding the costs of such collaboration need to be explored.

The DEIMOS team should review their commitment to IRAF (for the mask planning software) and consider a switch to IDL.

The DEIMOS team conducted a survey of the potential DEIMOS user community vis-a-vis their preferences between IRAF and IDL. The results of the survey were completely inconclusive; as a result, the team is maintaining its current preference for IRAF as the development environment for certain end-user tools such as pipeline data reduction and slitmask design.

Spectrograph Control:

- *DEIMOS plans to provide an X-terminal or laptop next to the instrument so that the instrument specialist can run the full blown user interface while looking at the instrument...DEIMOS engineers should coordinate with CARA regarding the details of the ethernet connection.*

  Although a recent telecon with CARA regarding cabling may have briefly touched on this issue, this should be flagged as an action item to confirm that the necessary cabling will be in place.

- *More analysis is need to ensure that the serial connection to the Galil will not impose a bottleneck*

Extensive testing with the Galil DMC-1500 series 8-axis controllers was conducted earlier this year in conjunction with tests of the Lantronix ETS8/P terminal servers. These tests were conducted both to establish the reliability of RTS/CTS hardware handshaking under conditions of maximum traffic and to measure overall throughput of the complete system when operating multiple serial channels simultaneously at the maximum controller baud rate of 19.2 Kbaud. Based on the results of these throughput tests under worst-case loading, we believe the serial connection to the Galil controllers will provide the necessary bandwidth for our application. Operational tests using this system during PFCAM commissioning in March confirm this at least for the case of operating 3 stages under actual observing conditions.

The biggest issue with remote observing is the need to duplicate the shared guider console. This is not possible with the existing xguide/cam system.

This problem should be flagged as an action item to be addressed by the Lick/CARA collaboration that is working on the new guider system.

DEIMOS slit mask alignment requires 0.1" offsetting accuracy, which is not available on Keck-1. For Keck-2, an engineering test to demonstrate 0.1" offsetting accuracy should be written up and run during Keck-2 engineering time.

We do not know if CARA conducted this engineering test, and if so, the result. However, subsequent analysis of the offsetting accuracy required for DEIMOS slitmask alignment indicates that 0.1" is overly ambitious. We will take what we can get.

The planned multiple subraster feature will aid slit mask alignment. The GUI to control it will be a challenge and prototyping should begin ASAP...In addition, perhaps CARA should extend the field overlay popup in "sky".

No prototyping of this GUI has yet occurred, nor has this activity been explicitly scheduled. This is a concern. However, the specification of the position of the alignment boxes should come directly from the slit mask design, and thus the observer should not need to interactively specify the location of the alignment boxes. This simplifies the GUI requirements considerably, since the main function needed is simply a toggle to switch between the normal readout mode and the alignment box readout mode. When a given mask is loaded into the spectrograph, its bar code will be scanned, which identifies the relevant slit mask design from which the alignment box positions can then be automatically obtained.

With regard to the extension of the field overlay popup in "sky", this deserves further exploration with CARA, since the display of the alignment box positions with respect to the guider field could be quite useful. For this to occur, CARA will need to make some of Conrad's time available (preferably in Santa Cruz), but his time is typically in very short supply.

The delta PA during slit mask alignment should be negligible, although for unknown reasons this was not the case for LRIS on Keck 1.

It would be useful to hear from the CARA representatives or from Cromer whether this mystery has been resolved with the move of LRIS to Keck 2.

CCD Control:

- *Focus on developing DSP and low-level C code for second-generation SDSU controller*
- *The single-amp configuration with eight amplifiers driven by a single timing board has the most going for it*
- *If there is any question about the fiber-through-the-wrap noise problems, DEIMOS should keep the CPU (for the CCD VME crate) at Nasmyth focus (rather than running fiber through the telescope wrap and locating the VME crate in the downstairs control room.*

As noted earlier, we have focussed on the DSP and low-level C code and are now successfully communicating with and operating the second-generation SDSU controller boards. This effort has proceeded better than expected, and reflects a significant improvement in robustness between the first and second-generation SDSU controller hardware.

Our current CCD controller design provides for a phased implementation. The first-phase yields the suggested configuration (single-amplifier readout from eight amplifiers via a single timing board) while allowing a future upgrade to dual-amplifier readout simply by adding addition controller boards and changing jumpers and moving interconnect cables between different connectors.

Based on latest recommendations we have received from Mark Sirota at CARA, it appears that the fiber-through-the-wrap noise problems that LRIS had experienced on Keck-1 have been resolved by the replacement of the 50-micron fiber in the cable wrap by 62.5-micron fiber. Both the first and second generation SDSU timing boards use 62.5-micron fiber, and the fiber in the Keck-2 cable wrap is 62.5-micron. Accordingly, we are still planning to locate the CCD VME crates in the control room, and to connect these to their corresponding SDSU timing boards within the DEIMOS instrument via the fiber through the Keck 2 cable wrap. In general, Sirota is recommending that we use fiber links through the cable wrap in place of other types of cabling wherever possible.

In addition, since we are currently planning on using 100-Mbit/sec twisted-pair fast Ethernet to connect the CCD VME crate to the instrument computer, and since the telescope cable wrap probably exceeds the practical length limit over which this type of fast ethernet can operate, this also argues in favor of locating the CCD VME crate in the control room. (However, if we were forced to move the CCD VME crate back to Nasmyth, we could switch from 100-Mbit/sec fast ethernet to a fiber-optic-based ethernet, although we would need to confirm that it would operate correctly over the 62.5 micron fiber in the telescope cable wrap.) Finally, since Sun has announced that they are withdrawing support for diskless clients after Solaris version 2.5, we are now planning to install a local SCSI disk drive within the CCD VME crate so that it can boot standalone rather than from the instrument computer. This provides yet one more reason for locating the CCD VME Crate in the control room rather than at Nasmyth focus.

Data Storage Formats:

- *Storing the entire mosaic as a single FITS image creates difficulties even for machines with 256+ MB is processing multiple images simultaneously. The multiple file options are probably justified.*

- *DEIMOS should stick with two dimension (i.e., NAXIS=2), and not use datacube techniques to represent the mosaic.*
- *Regarding overscan pixels, DEIMOS should follow the philosophy adopted for HIRES/LRIS, using extensions of the same keywords to describe the overscan geometry.*

DEIMOS does not plan to use datacube (i.e., NAXIS greater than 2) FITS representations for its mosaic images. The DEIMOS team has been participating in FITS standards discussions, mosaic image discussions, and the NOAO RTD effort in an attempt to ensure compatibility with other large detector array projects. We will attempt to extend the existing HIRES/LRIS mechanism for dealing with the overscan regions from each chip, but the detailed design for this is not yet complete.

Image Display and Quick Look Reduction:

- *The performance of SAOtng should be checked with a 140MB image.*
- *The choice of SAOtng should be reconsidered if there is no 16-bit access via imexam*
- *The decision not to use IDL is disappointing since that is the direction of newer Keck instruments.*

SAOtng is no longer a primary contender for the DEIMOS image display. Keck Figdisp is the display server we will be using for engineering tests, and our fallback for commissioning. While IDL-based display tools appear to be the choice of newer Keck IR instruments, there are currently no Keck optical instruments now in operation or planned for which IDL-based display tools are embedded within the data acquisition system. It is also unclear whether the Keck IR instruments will be using a common IDL-based tool, or several different tools which simply happen to have been built using IDL. As noted earlier, our survey of the DEIMOS user community did not elicit any obvious consensus that we should switch to IDL-based software either for slit mask design software or for Quick Look data analysis.

Information Management and Data Archiving:

- *Only DEIMOS database requirements are in the scope of the DEIMOS effort; Keck environmental and telemetry data are not.*
- *An isolated demonstration project on Keck 1 (getting HIRES temperature logs into sybase) would be a good stepping stone toward getting a DEIMOS/CARA/Sybase collaboration going that would benefit both.*

Some discussion has taken place between CARA and Lick about the feasibility of a joint telemetry and archiving venture in support of not only DEIMOS but other Keck instruments. This discussion is not complete at this time.

A pilot effort succeeded in importing HIRES log data into the Sybase database at Keck. However, more polished end-user tools (for analysis, query, visualization) are needed to ensure acceptance and common use of such engineering databases. Should a joint venture as described above actually take place, development or acquisition of such tools would be a high priority.

A.4.6 George Jacoby II

IRAF:

- *While some astronomers argue NOAO should focus resources on operating telescopes rather than supporting IRAF, current NOAO management is committed to keeping the project alive, but this committment could be reviewed in a few years if budget conditions remain bleak.*
- *Gemini, HST, and AXAF all depend on IRAF. If NOAO support evaporates, one of these programs will be forced to assume some responsibility*
- *Perhaps Keck should assume some community responsibility (1-2 FTE) for supporting IRAF if NOAO withdraws support*
- *The aggregrate annual costs of licensing IDL for all potential Keck users adds up to something like the 1-2 FTE's that could otherwise be used to assume IRAF responsibility.*
- *For IDL to replace IRAF, a significant body of existing code would need to be ported*
- *The DEIMOS programmers have significant IRAF experience but lack comparable experience with IDL. However, they could probably get up to speed quickly.*

George's comments on IRAF's present and future are noted, and were taken into account as the DEIMOS team pondered the inconclusive survey results mentioned above.

A.5 Addendum: Initial Response to PDR Review Board Report

This initial response to the DEIMOS Software PDR Review Board Report was published as section 2.4 of DEIMOS Quarterly Report Number 8, which was issued in mid-July 1996. Although a more formal, detailed report (for which Kibrick was responsible) was planned at that time, it was never prepared. The first detailed response to the PDR Review Board Report is section 2 of this Appendix.

I have annotated this initial response to show how our statements, estimates, and projections (made in July 1996) compare with today's reality. (My annotations are italicized.) I believe this comparison provides a useful reality-check which should be used to calibrate the reliability of similar statements, estimates, and projections made in this CDR document.

In general, these comparisons show that the software tasks planned for Allen, Clarke, Phillips, and Tucker were completed as planned, but those planned for Kibrick slipped significantly due to competing demands from other projects as well as delays in the scheduled delivery of CCD controller hardware.

While a similar comparison of software projections versus reality could be made for the software sections of subsequent DEIMOS quarterly reports (numbers 9-11), I think the overall conclusions would be the same.

Section 2.4 of DEIMOS Quarterly Report Number, July, 1996
At the start of this quarter we received the review board's report for the DEIMOS software PDR that was held on March 22. The review board was generally satisfied with the preliminary design and the completeness of most of the material presented. The review board had specific recommendations addressed to a number of areas, and our efforts during the last quarter involved several of these:

- budget and schedule
- CCD readout hardware and software
- early prototyping of user interface software
- software tools for quick look image display and reduction
- coordination with CARA, especially with respect to guider
- allocation of effort towards general software infrastructure

- generation of simulated DEIMOS data

Other review board recommendations will be addressed in our formal response to their report.

Budget and Schedule The scheduling of the DEIMOS software effort since the PDR has been primarily short-term, based on relatively close day-to-day coordination among the DEIMOS software staff. While we are very pleased with the software progress since the PDR, it is clear that longer-range planning of the DEIMOS schedule is still needed, and that close coordination with the ESI software scheduling is also required. Toward this end, Bigelow and Kibrick have met twice to iterate on the ESI schedule, and Kibrick and Faber have several meetings scheduled for late July to refine the DEIMOS software schedule. An updated long-term software schedule for DEIMOS will be released following those meetings. *Note: Such an updated long-term software schedule was never prepared.*

We are still planning on holding the software CDR near the end of 1996, and will try to have the actual date established by the end of the next quarter. *Note: Due to late delivery of the second-generation SDSU CCD controller hardware, the software CDR was first delayed to March 1997, and then delayed again until June 1997 to insure that it occurred following the DEIMOS CCD Review.*

CCD readout hardware and software This was identified as the highest risk area for the DEIMOS software effort, and the board recommended focusing on development of DSP and low-level code for the second generation SDSU CCD controller as soon as possible. We agree with this recommendation, and will commence work on the prototype DSP software as soon as receive the beta-test version of the second-generation CCD controller boards from SDSU. This hardware was successfully tested at SDSU during a Gemini-related review held on May 23, and it achieved a per-pixel readout time of 2.4 microseconds/pixel reading out a 1K x 1K CCD. Subsequently, some manufacturing problems were discovered involving the reliability of chip sockets and the soldering of surface mount components. SDSU is working with the board manufacturers and assemblers to resolve these problems. Expected delivery of this hardware has now slipped approximately one month, from mid-August to mid-September. Should this delivery slip further, we will likely begin preliminary tests of our fallback plan of using the first-generation SDSU CCD controller hardware. The test CCD dewar should be completed well before software testing begins, and several engineering grade, thick, 2K x 4K CCD Orbit CCD chips are available for testing in that dewar. Significant effort on design and prototyping of the DEIMOS CCD controller software will thus commence near the end of the next quarter.

Notes:

- *The delivery of the second-generation SDSU hardware slipped to March 1997.*
- *Although nearly completed, the test CCD dewar has still not been delivered to our CCD lab*
- *Although delivery of the second-generation SDSU hardware slipped beyond mid-September 1996, we did not begin preliminary tests of the fallback plan of using first-generation SDSU CCD controller hardware, nor have we expended any further effort on that fall-back plan.*
- *Significant effort on the DEIMOS CCD controller software did not commence until March 1997*

Early prototyping of user interface software The board recommended putting more effort into early prototyping of DEIMOS user interfaces. We agree, and have begun assembling software toolkits to be used for this effort. We plan to use Tcl/Tk-based tools for UI prototyping, and numerous Tcl/Tk-related tools have been acquired, (ported where necessary), installed, and tested under Solaris 2.4.5. To support tests of these UI prototypes using existing Keck-style hardware and to also support the eventual development of the actual DEIMOS control and UI tasks, Clarke succeeded in porting the Keck Task Libraries (KTL), the complete HIRES keyword libraries, and Lupton's KTL-Tcl to Solaris 2.4.5. (The latter was also ported to Tcl version 7.5). This porting effort proved to be much harder than anticipated, due to a number of serious difficulties with the existing Keck-1-style make/build procedures provided by CARA as part of the KTL distribution, as well as the use of non-portable programming constructs within the KTL sources. As a result, we intend to develop alternative make/build procedures that will be easier to maintain and to use, and which will be portable between different operating systems, compilers, and hardware architectures. We also plan to follow the recommendation of the review board and convert to using CVS rather than SCCS (as was used for KTL and other Keck-1 software) for source code management. *Note: In collaboration with CARA, we have developed these alternative make/build procedures.*

To confirm the integrity of this porting effort and to provide a proof-of-concept of using these Tcl/Tk-based tools to build GUIs for instruments, Clarke successfully used this ported software to develop a working, albeit rudimentary, GUI for the HIRES image rotator. This GUI was used to operate portions of the actual rotator hardware currently in Santa Cruz (awaiting shipment to Keck). The DEIMOS software team has also reviewed the existing HIRES rotator GUI (specified by Tytler and implemented at CARA using the commercial GUI-builder software "Dataviews") and has identified a number of aspects of the design that can be improved for the DEIMOS GUI that will be used to control instrument rotation.

Software tools for quick look image display and reduction The review board expressed some concerns regarding our plan to use IRAF-based packages as our primary software for DEIMOS quick-look reduction and display, and suggested that we at least investigate other alternatives such as IDL. We prepared a questionnaire to better gauge the needs and preferences of prospective DEIMOS users, as well as the anticipated availability and support for IRAF and IDL as those users' sites. This questionnaire was distributed via e-mail at the end of the quarter to approximately 150 DEIMOS users at UC, CIT, UH, and NASA. The final date for the return of these questionnaires is August 1. (As of July 14, we had received 35 responses, but these have not yet been tabulated.)

Note: The results from the IRAF/IDL questionnaire were briefly described in DEIMOS Quarterly Report Number 9.

We have also completed a series of image display and quick look reduction tests on large images using IRAF, and have nearly completed corresponding benchmarks with IDL. Preliminary results do not indicate significant performance differences that would commend one package over the other.

Coordination with CARA The review board identified the need for close coordination between the DEIMOS team and CARA software developers with respect to several key areas:

- A. Development of Keck 2 guide camera and autoguiding software
- B. Physical layout of the Keck 2 control room
- C. Coordination of the DEIMOS database software development with any related development of a central Keck 2 database

- D. Achieving commonality between CARA's "sky" observation planning tool and the DEIMOS slit mask design tool
- E. Coordination of DEIMOS instrument rotation and small telescope offsetting accuracy with CARA DCS and autoguider software

During this past quarter, we put considerable effort into item C. Clarke and Kibrick visited CARA for several days in late April to work with CARA software staff on various DEIMOS-related issues and to develop a proof-of-concept of the utility of using a database package (i.e., Sybase) for managing and archiving both science images as well as engineering/ environmental data from instruments. (Clarke also watched Faber's LRIS observing run, and submitted a report to Faber in May describing various problems with the existing GUIs and observing procedures.) Clarke developed and installed a working prototype for ingesting the FITS headers from existing Keck science (and guider) images into a operational database. This tool proved sufficiently successful that it has been adopted by CARA as operational software, and the FITS headers of all of the HIRES, LRIS, and NIRC images that have been saved to Exabyte tape over the last several years via the STB backup software have now been stored into this database; headers for newly acquired images are being added to the database as part of the STB backup procedure. Using this database, relatively complex queries regarding the FITS headers of Keck images taken to date can now be rapidly answered.

Note: Clarke made a follow-up visit to CARA in January 1997 to make refinements to the STB header ingestion software and to provide training to CARA staff in the operation and maintenance of this suite of software.

Clarke also implemented and installed a prototype database for managing the HIRES engineering/environmental data, and all of this data for the last few years has been ingested. In addition, Clarke installed several Tcl/Tk-based GUIs to allow non-expert users to construct database queries to the FITS header database, and a prototype web-based UI for querying the database of HIRES engineering/environmental data. These tools have been well-received at CARA, although the web-based UI and other querry tools need further refinement.

These database prototypes represent a subset of the database implementation that we plan for DEIMOS. Their successful installation, operation, and reception at CARA serve to validate both the feasibility and utility of using database packages (such as Sybase) for managing these kinds of data from astronomical instruments, and represent the first step in developing at CARA the necessary infrastructure for supporting such database applications as we intend to deliver as part of the DEIMOS software.

Allocation of effort towards general software infrastructure The review board recommended that we allocate sufficient software effort early on towards defining and constructing an adequate software infrastructure to support DEIMOS software development. We agree, and during the past quarter considerable effort was devoted to acquiring, building, installing, and testing not only the GUI development tools noted above, but general-purpose software development tools including compilers (e.g., gcc), debuggers (e.g., gdb, ddd), source code utilities (e.g., tkdiff), code maintenance software (e.g., RCS and CVS), and Sybase-related tools. In addition, in response to the review board's recommendation that we not develop our documentation directly in HTML (as was done for our PDR documents), several documentation tools (tex, latex, latex2html) were installed which will allow us to develop documents using LaTeX which can then be translated automatically to HTML for posting to the web. CARA will also be providing to DEIMOS a version of the Frame package, including an option for generating HTML from Frame documents.

Note: DEIMOS stayed with the stated plan to use LateX for document generation, and have not actively used the Frame package provided by CARA for generation of software documentation.

Many of these general-purpose infrastructure tools were subsequently used for the porting and building efforts described above in the section on "Early Prototyping". All of these tools are now operational under Solaris 2.4.5, and many are also operating under Digital Unix (formerly OSF) on the DEIMOS Dec Alpha test-bed machine, radec. Porting/installing the remainder of these tools to Digital Unix is planned for completion in the next quarter. *Note: the porting of these tools across all of our supported platforms was successfully completed.*

Generation of simulated DEIMOS data The review board recommended that we generate and experiment with simulated DEIMOS data early on. We agree, and plan to start generating such data towards the end of the next quarter. In preparation for this task, as well as to support the overall design and documentation effort, Clarke has designed and implemented a database for managing the large number of Keck keywords that will be used to control and document the operation of DEIMOS, and has developed an exhaustive specification of legitimate FITS keyword syntax.

As a proof-of-concept, this database was successfully used to model all of the existing HIRES keywords. This database will be used to automatically generate HTML documents of keywords, as well as code to generate sample DEIMOS FITS headers.

In parallel, Allen has designed and documented all of the coordinate transforms needed to map between sky coordinates, focal plane coordinates, and both spectral and imaging pixels of the detectors of the mosaic. The FITS keywords and associated FITS extension tables for representing these transforms have been defined, and the means for representing these within the keyword database have been designed. Allen has also worked to finalize the coordinate system labeling and detector/amplifier numbering conventions, and was worked with Osborne to insure that the consistent use of these conventions on all DEIMOS engineering and mechanical design drawings.

Abstracts for two papers, which describe the keyword database and the coordinate transform designs, are being submitted for the Astronomical Data Analysis and Software Systems (ADASS) conference scheduled for late September. *Note: These two papers were presented at the ADASS conference. On-line copies can be obtained from*

http://www.ucolick.org/~sla/deimos/index.html

Plans for the next quarter We anticipate that Clarke, Allen, and Phillips will continue working on DEIMOS at roughly the same level of effort during the next quarter. Clarke will be concentrating on overall data-flows and global design issues, definition of database schema, and GUI prototyping. An internal software review of the data-flows and database schema is now scheduled for July 30.

Allen will begin software prototyping of the coordinate transformations and their representation in FITS image headers, with the goal of generating simulated DEIMOS FITS images that include all of the relevant DEIMOS keywords and header extension tables. This simulated data will be used to evaluate various quick image display and data reduction alternatives. Allen and Clarke each plan to present a paper at ADASS.

Phillips will continue his efforts on slit mask design software, and will begin work on integrating into that software the coordinate transform and FITS header mechanisms that Allen has defined. Phillips and Allen will also spend some time evaluating the capabilities of IDL (and the user-contributed IDL astronomy libraries). We will try to arrange for Phillips to visit with at least one astronomy research group which conducts optical observations at Keck and which uses IDL as its primary quick look reduction tool. With assistance from the group, Phillips will test the ability of IDL and existing IDL astronomy packages to perform typical quick look image display and data reduction functions on existing LRIS multi-slit data.

Note: Allen, Phillips, and Clarke did work on some of these activities as planned, and the internal DEIMOS review was held on July 30. However, Phillips did not visit an astronomy research group that uses IDL for quick look reduction of Keck data, and did not conduct any tests of using IDL for quick look display or reduction of LRIS multi-slit data.

The availability of Kibrick and Tucker to the DEIMOS software effort will be significantly constrained during the next quarter, due to vacations, HIRES image rotator installation and commissioning, several Keck AO CDRs (for which Kibrick is serving on the review board), and commissioning of the Lick Prime Focus Camera (PFCAM) on the Shane 3-m in late September. DEIMOS CCD software design and prototyping is now anticipated to commence near the end of the the next quarter (9), coincident with the expected delivery of the beta-test second-generation SDSU controller hardware. DEIMOS-specific motor control software effort will also commence at that time, building on common motor controller software developed for PFCAM.

Note: The availability-constraints imposed by the HIRES image rotator upgrade and the subsequent retrofit of the Keck-2 Telescope Drive and Control System (DCS) onto Keck-1 in reality stretched out through early March 1997, i.e., well in quarter 11. CCD software prototyping did not commence until late March 1997.

During the next quarter, we anticipate purchasing (with DEIMOS funds) an additional 4 GB of disk to support Solaris-based software development for DEIMOS, as well as a laptop computer (running Linux) that will be split-funded with DEIMOS, ESI, and UCO/Lick funds. We are currently obtaining pricing for Dec Alpha memory, with the goal of adding an additional 256 MB to the DEIMOS test-bed machine, radec, this fall to support tests involving simulated DEIMOS images. We plan to complete our formal response to the PDR review board report during the next quarter, following the tabulation of results from our IRAF/IDL questionnaire, and Phillips' evaluation of IDL capabilities.

Note: the proposed hardware was acquired, except the RAM upgrade for radec was increased to bring the total memory to 1GB. The laptop has proved to be extremely valuable to our porting efforts, as well as to the productivity of our visits to Keck for collaborations with CARA staff. The formal response to the PDR review board report was not completed as planned, although the *results from the IRAF/IDL questionnaire were briefly described in DEIMOS Quarterly Report Number 9. Phillips did not perform an evaluation of IDL capabilities.*

Part II

Subsystem Designs

Chapter 1

Slitmask Design & Fabrication

1.1 Overview

This chapter deals with slitmask design and fabrication, both from the observer's and the observatory's perspective. Under normal procedures, the user will produce slitmask designs, and the observatory will fabricate the masks.

The user initially designs a mask on the sky (for an relatively arbitrary epoch) which includes a target list and optional "custom" information for each object, e.g., a particular position angle and/or size of slitlet. This stage of mask design must be highly interactive, and provide not only the positions of potential targets on the sky, but also provide the user with information on atmospheric dispersion, telescope limits, wavelength coverage at the detector, location of gaps in the detector mosaic, suitable guide and alignment stars, etc. — anything that influences the placement of targets on the mask. Once the target list, position angle and mask center are settled, the final design is committed to a physical slitmask description for a specific observing epoch. This physical description, or "blueprint," includes corrections for refraction at a specified hour angle, as well as aberration of starlight, proper motion updates, telescope distortions, etc.

The target list and mask blueprint, packaged as a single FITS file, are forwarded to Keck for ingestion in the database and for mask fabrication. Fabrication involves translating the blueprint into instructions for the NC mill, running the mill, and inspection of the result. Tracking of masks is provided by barcode labels on the physical masks, which also provide the association of mask design in the database with a physical mask. The mask fabrication procedures are designed for minimum impact on Keck staff, both in mill operation and in handling a large volume of slitmasks for many different observers.

This chapter describes (a) the mask design software, up to production of the mask blueprint; (b) the fabrication software; and (c) the overall handling of physical masks by Keck staff.

1.2 Subsystem Figures

See Figure 1.1, "Slitmask Design – Overview," for a flow diagram of the mask design procedure.

Figure 1.2, "Slitmask Layout," shows a schematic diagram of the physical slitmask.

1.3 Nomenclature

Fundamental Sky Coordinates: Celestial coordinates tied to a fundamental coordinate system at a specified epoch and equinox.

Refracted Sky Coordinates: Fundamental Sky coordinates adjusted for refraction, aberration of starlight and proper motion.

Slitmask Coordinates: Physical coordinates on the flat slitmask (which becomes a tilted cylinder when loaded in DEIMOS). Defined at the nominal operating temperature for DEIMOS.

Camera Focus Coordinates: Coordinates in pixel units at the nominal focal plane of the camera, easily transformed to pixel coordinates on a given CCD.

Mask Design: The mask center, position angle, and list of slitlets and objects to appear on a slitmask, in fundamental sky coordinates. (See Part I, Section 3.6.3; and Part II, Chapter 2 of the *Dictionary* for details.)

Mask Blueprint: The description of the physical slitmask design in the slitmask coordinate system. (See Part I, Section 3.6.3; and Part II, Chapter 2 of the *Dictionary* for details.)

MaskFile: A FITS file with tables describing the Mask Design and the Mask Blueprint.

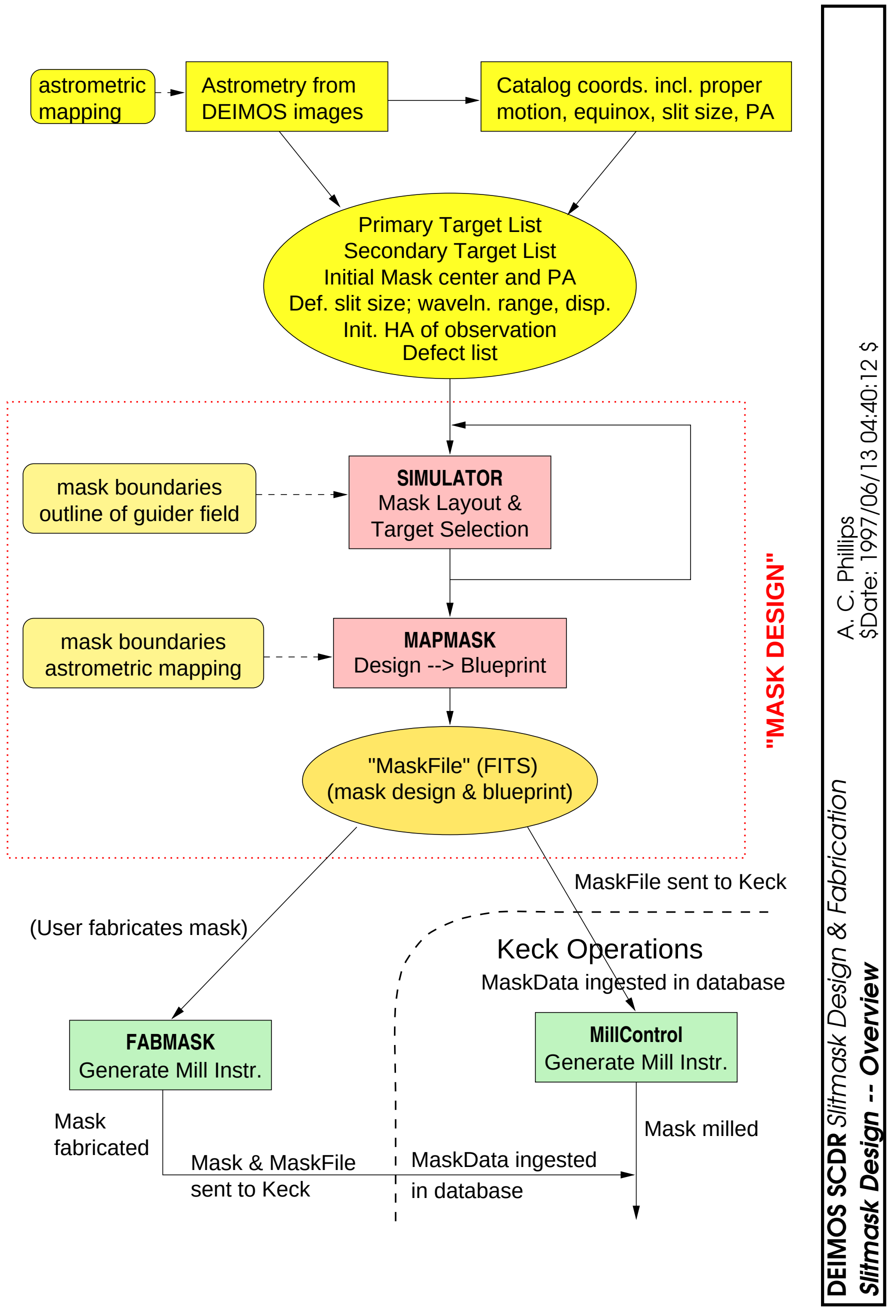


Figure 1.1: Slitmask Design – Overview

1.4 Component Modules

1.4.1 SIMULATOR

Function Interactive mask design program, producing a list of objects/slitlets in celestial coordinates.

Language IRAF/spp

Type tool

Author A. C. Phillips

Revision N/A

Essay A large, graphical, highly interactive stand-alone module which allows the user many options for target selection and overall mask specification. Mask specification options: translate mask center, rotate mask PA, change HA of observation. Target selection options: autoselect (select full set of targets to maximize the summed target priorities), add/delete target, bind targets to the same slitlet. Slitlet specification options: fix slitlet PA, set minimum length. Alignment options: select alignment and guide stars. View options: set field-of-view, show slitlets on mosaic, show spectra on mosaic, redisplay, label targets, print target information. Dispersion information: show dispersion vector in slitlet over the specified wavelength range. Miscellaneous: report telescope limit violations. The module will precess coordinates to a specified equinox, and update for proper motion. Lists of both selected and non-selected targets are output, both in the same format as the input lists, so that the module may be executed as many times as desired to refine the mask design.

Inputs primary, secondary lists of potential targets; initial mask center and PA; initial HA; wavelength range, dispersion; detector defect list

Outputs Mask Design: list of selected targets/slitlets, selected mask center and PA; also list of non-selected targets

Deployment Commissioning

Labor 240 hours

Keywords used by module (none)

1.4.2 MAPMASK

Function Map mask design (celestial coord.) into a mask blueprint (physical coord.)

Language IRAF/spp

Type tool

Author A. C. Phillips
Revision N/A
Essay This noninteractive module performs three functions: (1) transforming the slitmask design from fundamental sky coordinates into refracted sky coordinates; (2) transforming refracted sky coordinates into slitmask coordinates; and (3) adjusting final slitlet lengths to remove slitlet overlap and/or fill unused slit space. Mappings used in the two transformations are described in the Calibration Document.
Inputs Mask design from SIMULATOR; HA, atm (T,P), wavelength for refraction; date and (long,lat) for aberration; slitlet separation
Outputs FITS file of mask design + blueprint
Deployment Commissioning
Labor 80 hours
Module flow diagram Figure 1.3, "Flow Diagram for Module MAPMASK"

Keywords used by module (none)

1.4.3 FABMASK

Function Translate mask blueprint into NC mill instructions
Language IRAF/spp
Type tool
Author A. C. Phillips
Revision N/A
Essay This module is internal to MillControl, but is provided as a standalone for users wishing to fabricate their own slitmasks. This module produces NC mill instructions from a mask blueprint, and is normally executed (via MillControl) by Keck personnel immediately prior to milling the slitmask. The actual milling instructions are TBD, although an option for producing ASCII-format instructions will be included for stand-alone mode. The temperature of the milling environment is required as input because the mask blueprint is in slitmask coordinates (defined for the nominal operating temperature of DEIMOS), so a slight correction is required for thermal expansion of the aluminum mask.
Inputs Mask blueprint specification from database; Ambient Temperature
Outputs NC Mill instructions
Deployment Commissioning
Labor 30 hours

Keywords used by module (none)

1.4.4 PLOTMASK

Function Create a plot of the blueprint with targets superposed.
Language IRAF/spp
Type tool
Author A. C. Phillips
Revision N/A
Essay This module takes a mask design/blueprint and produces a PostScript plot of a mask.
Inputs Mask Design FITS file
Outputs PS Plot
Deployment Commissioning
Labor 30 hours
Keywords used by module (none)

1.4.5 SKYCOORDS

Function Transform pixel coordinates of a DEIMOS image into fundamental sky coordinates.
Language IRAF/spp
Type tool
Author A. C. Phillips
Revision N/A
Essay This module performs astrometric measurements of a DEIMOS direct image of sufficient accuracy that slitmasks may be designed. It applies a mapping from the CCD pixel coordinates to the sky. Detailed design is TBD.
Inputs Input pixel coordinates; othersTBD
Outputs List of sky coordinates
Deployment 1 yr Post-Commissioning
Labor 60 hours
Keywords used by module (none)

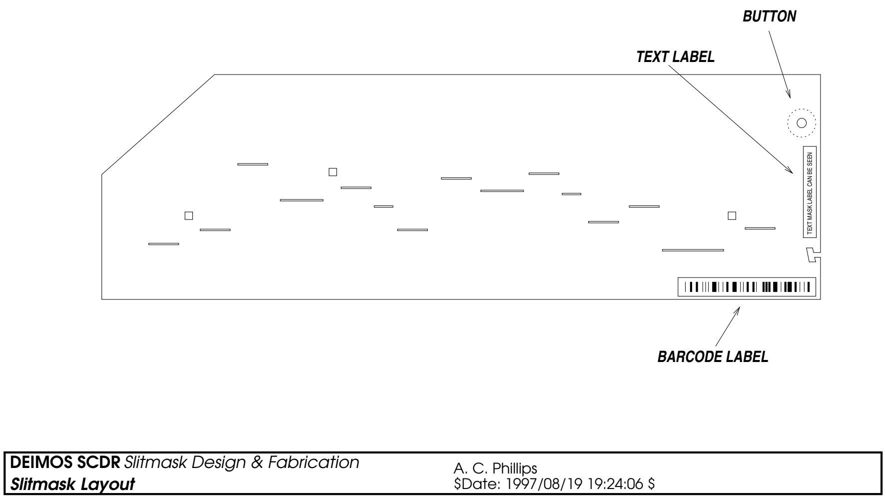


Figure 1.2: Slitmask Layout – this is a schematic of the physical slitmask. The barcode is affixed when the blanks arrive from the vendor; the button and text labels are added at fabrication.

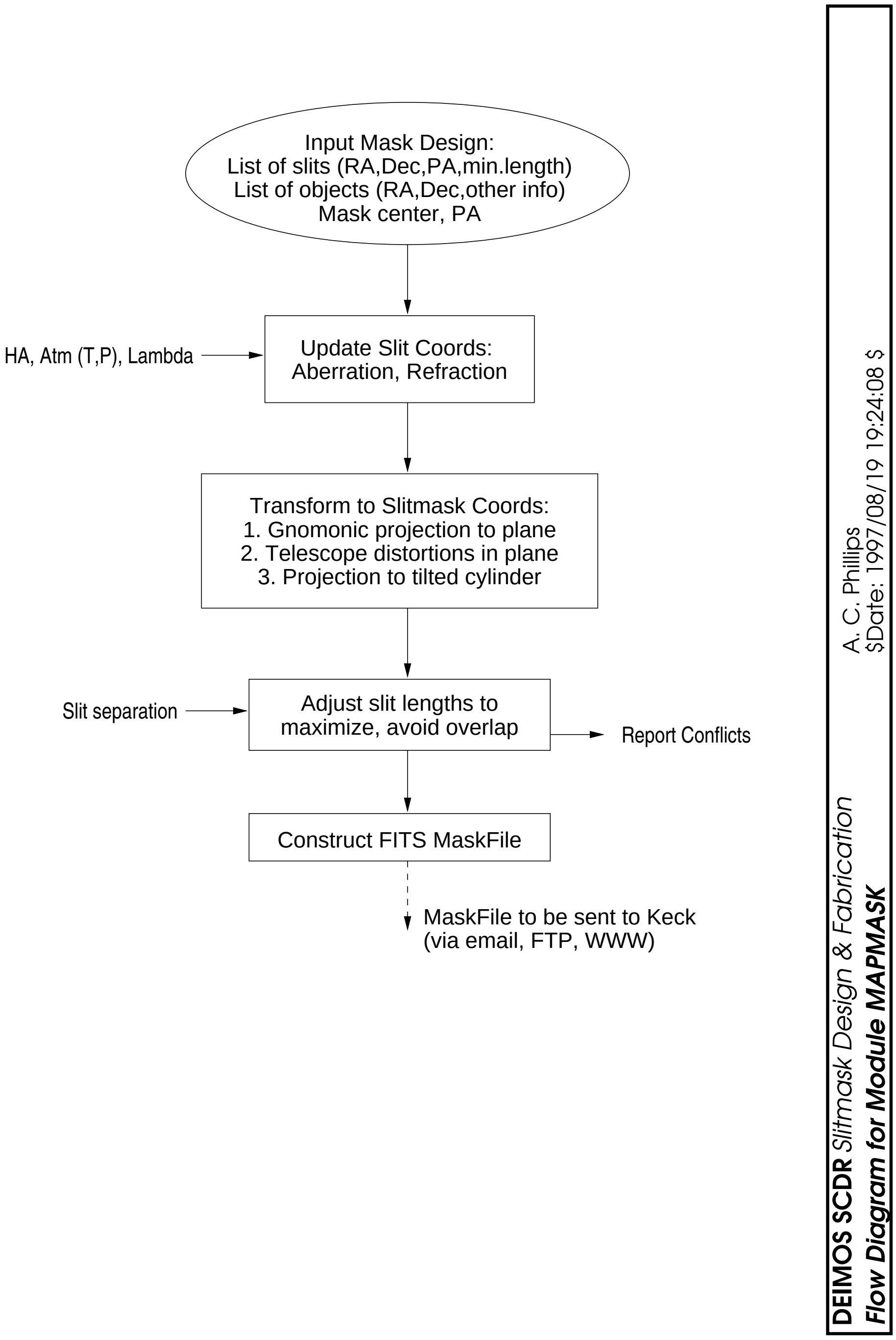


Figure 1.3: Flow Diagram for Module MAPMASK

1.4.6 MaskAcceptor

Function Accepts MaskData from Observers and fills mask database
Language Tcl/Tk
Type daemon
Author S.L. Allen/D.A. Clarke
Revision N/A
Essay The Slitmask design software produces a FITS file known as MaskFile containing catalog, design, blueprint, and slit/obj map tables. This file must be transmitted to Keck and stored into the slitmask database for later use. This module handles the transfer of the MaskFile using WWW and e-mail.
Inputs

- CGI queries from WWW server
- Mask Design, Blueprints, and status from database

(For details see the database schema in Part II Chapter 2 of the *Dictionary*, and/or the equivalent FITS tables in Part I Section 3.6.3).
Outputs

- SQL inserts to the slitmask database
- e-mail to blueprint owner verifying receipt

Deployment Commissioning
Labor 40 hours
Keywords used by module None. KTL is not involved with mask production and inventory control.

1.4.7 MaskQuery

Function Permits observers to verify the status of their slitmasks.
Language Tcl/Tk, HTML2
Type daemon
Author D.A. Clarke/S.L. Allen
Revision N/A
Essay Upon request of observers via a WWW interface, MaskQuery interrogates the database of slitmasks maintained by MillControl. Observers will be able to verify whether Keck has received their MaskFile, whether a mask has been fabricated, and whether the mask has been discarded. This interface will not divulge detailed slitlet information unless a suitable authentication scheme is implemented.

The observer interface will also permit the specification of the subset of masks for the observing run which should be loaded before each night. It will also permit requests for changes of filters and gratings.

Upon request of instrument specialist, MaskQuery interrogates the database of slitmasks to determine which masks need to be loaded tonight.

Inputs

- CGI queries from WWW server
- Mask Design, Blueprints, and status from database

(For details see the database schema in Part II Chapter 2 of the *Dictionary*, and/or the equivalent FITS tables in Part I Section 3.6.3).

Outputs HTML web pages indicating the status of the slitmasks

Deployment Commissioning

Labor 20 hours

Keywords used by module None. KTL is not involved with mask production and inventory control.

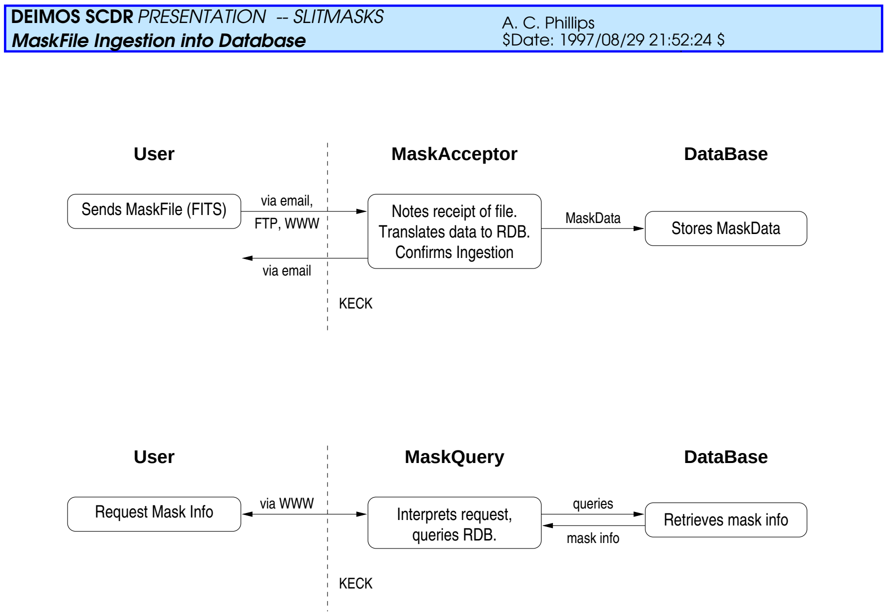


Figure 1.4: Slitmask Dataflow between User and Database

1.4.8 MillControl

Function Coordinates cutting and quality control during slitmask fabrication.
Ensures validity of mask database.
Language Tcl/Tk
Type tool
Author S.L. Allen/D.A. Clarke
Revision N/A
Essay MillControl is used by the mill operator for the routine cutting of slitlets into raw slitmask stock. It is the principal interface for inventory control over slitmasks and slitmask stock. It queries the database to determine which mask blueprints need to be cut and offers that list to the mill operator. It converts the selected mask blueprint to NC milling commands and downloads them to the mill. It requires the mill operator to scan the mask barcode after each download to ensure proper association between mask and blueprint. It permits the mill operator to enter quality control information about the fabricated masks.
Inputs

- Mask Design and Blueprints from database
- barcodes from handheld scanner
- ambient mill temperature

(For details see the database schema in Part II Chapter 2 of the *Dictionary*, and/or the equivalent FITS tables in Part I Section 3.6.3).
Outputs

- Fabricated slitmasks and associated database records
- e-mail to mask blueprint owner describing mask status

Deployment Commissioning
Labor 120 hours
Keywords used by module None. KTL is not involved with mask production and inventory control.

1.5 Subsystem Procedures

1.5.1 Pre-Commissioning

Verify/update mapping of slitmask–to–CCD coordinates (original mapping is theoretical; check in lab).

CCD defect list should be prepared.

1.5.2 Commissioning

Verify/update mapping of slitmask–to–CCD coordinates (original mapping is from lab; check following reassembly).

Verify/update mapping of refracted sky–to–slitmask coordinates ("slitmask astrometry"; see Part III).

1.5.3 Operational Procedure

Mask Design (performed by user for each slitmask)

1. Execute SIMULATOR to design mask

1a. Re-execute SIMULATOR, as desired to refine mask design

2. Execute MAPMASK

3. Send FITS MaskFile to Keck (email, FTP, WWW)

(4) Optional: user may follow own path to slitmask fabrication

1.5.3.1 Mask Fabrication

The purpose of this procedure is to ensure the correctness of the association between the barcode on the slitmask and the mask blueprint which was milled into the mask. When a dozen masks are milled during one session we want to avoid any problems of off-by-one or otherwise permuted identification between mask and blueprint.

We recognize that the mill operator may be involved in other tasks during the fabrication of DEIMOS slitmasks. We have minimized the activity required between the milling of one mask and the next. This procedure also permits slitmask Quality Control to be performed separately from slitmask milling.

Hardware Components These are used for the mask milling procedure.

- Numerically controlled mill (i.e., the Centurion V).
- Multi-tasking computer (preferably Unix, but minimally Windows NT or 95).
- Serial line connecting computer and NC mill for DNC operation.
- Handheld barcode scanner near the mill.
- Serial line connecting computer to handheld scanner.
- Status Barcode Chart of milling success/failure barcodes mounted near the mill. Barcodes fall into categories:

  status codes NOW_MILLED, DISCARDED

  quality codes MASK_GOOD, MASK_BAD

  request codes QUERY

  response codes YES, NO

  inventory codes FIRST_CHECKIN, LAST_CHECKIN

  foreign mill codes EXPORTED, IMPORTED
- Blank mask stock with pre-affixed barcodes readable by DEIMOS mask changer.
- Ambient mill temperature probe.
- Label printer.

Software Components The software required for this procedure is a collection of programs known as MillControl. MillControl can perform the following tasks.

- Run on the multi-tasking computer.
- Interrogate the database of yet-to-be cut mask blueprints.
- Permit mill operator to select next blueprint to be cut.
- Read the temperature probe and convert mask blueprints into NC mill programs (G-codes, M-codes).
- Send mill programs to the mill in DNC mode.
- Read the hand-held barcode scanner.
- Update the databases of mask blueprints (cut and yet-to-be cut).

The flow of information starting from the time of mask design through the time of mask production has been entered into the Agents database. This flow is depicted in Figure 1.5. See Chapter II.7 for more information about the Agents and meme flow.

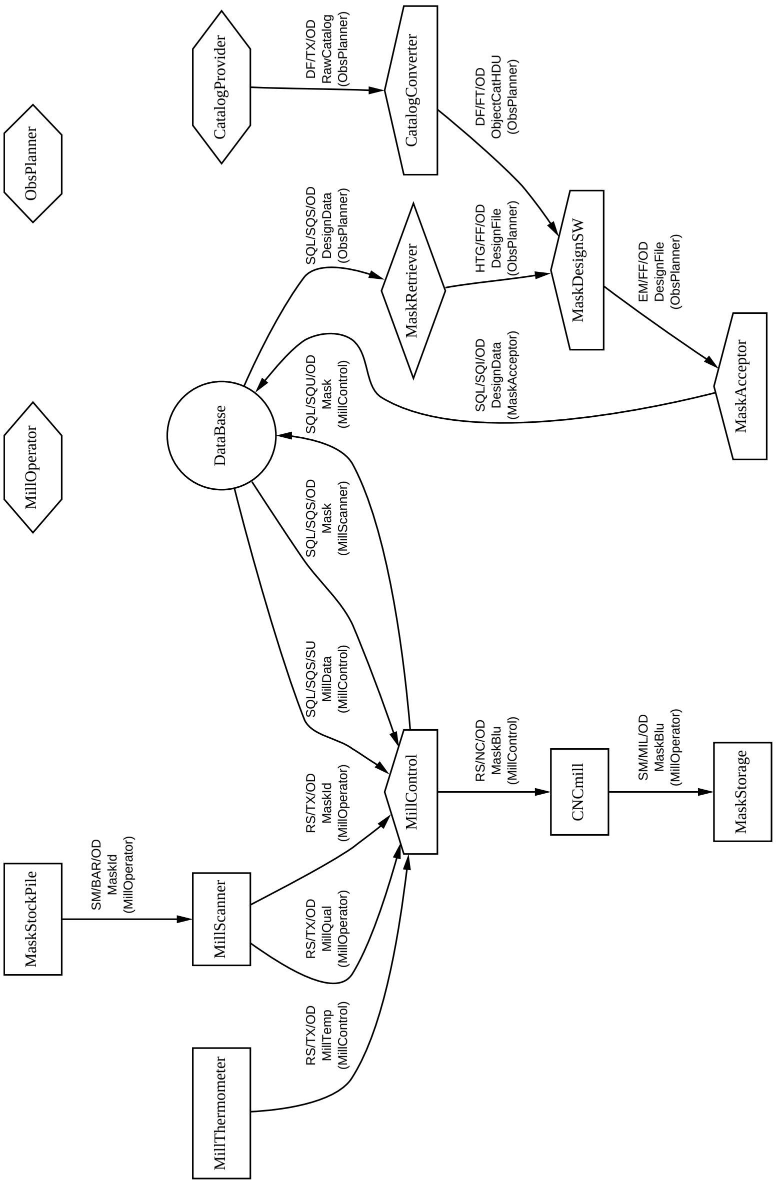


Figure 1.5: Data Flow for Slit Mask Production

Slitmask Stock Inventory Check-in The slitmasks loaded in DEIMOS are identified via barcode labels. Prior to milling, a barcode label must be affixed to each blank slitmask in a location suitable for reading by the barcode scanner within DEIMOS. This will be done whenever a new shipment of blank masks arrives. Then MillControl must be run to enter the entire shipment into the database. For each shipment we record the vendor, various physical characteristics of the stock, and the first and last barcodes in the sequence applied to that batch. (Note that we require the sequence of barcodes to be monotonically increasing.)

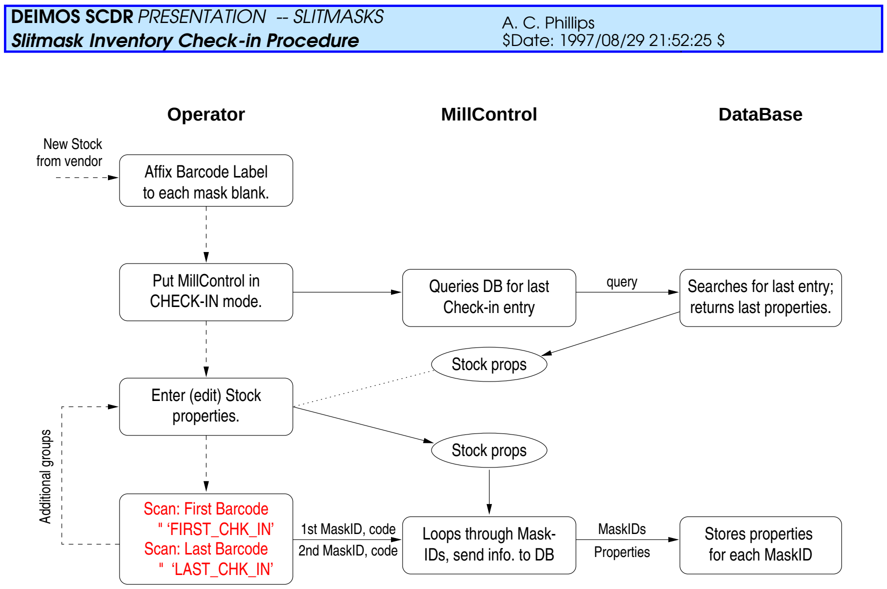


Figure 1.6: Slitmask Inventory Check-in

Milling Setup These steps are required at the beginning of a milling session.

- The mill operator asks the multi-tasking computer to start MillControl.
- MillControl queries the database and presents a list of DEIMOS masks which need to be cut in order to keep on schedule.
- The mill operator checks and inputs relevant parameters for milling. These include operator id, tool size, (tool shape), and ambient temperature.

- The mill operator places the NC mill into DNC mode awaiting instructions from MillControl.

Milling Procedure For each blueprint to be cut the following steps are performed.

- The mill operator places mask stock into the mill.
- The mill operator uses MillControl to select one of the not-yet-cut mask blueprints.
- MillControl will convert the selected blueprint into G-codes and M-codes for the mill, using ambient temperature to compensate for thermal expansion.
- The mill operator tells MillControl to begin the DNC download.
- MillControl sends the NC mill control codes over the serial line to the mill where they are executed. Upon beginning transmission MillControl will enter a mode where it demands the identity and status of the currently active mask before it will permit the milling of any other mask. Any other action, e.g. quality control, will be permitted
- If for some reason the download of NC mill instructions fails, the operator can command MillControl to begin the DNC again for the currently active slitmask.
- When the milling is done the mill operator removes the mask from the mill, scans its barcode, and scans NOW_MILLED on the status barcode chart.
- If there is a gross problem with the mask (e.g., the tool broke) then the mill operator may indicate it by scannning the mask barcode again, then scanning the quality code MASK_BAD.
- Repeat the above for the next blueprint.

Note that we do not permit a milling program to be restarted if it is interrupted in the middle. We expect that, for example, restarting a program after replacement of a broken tool would introduce unacceptably large systematic errors between the first and last milling sequences. Masks which are partly milled must be junked.

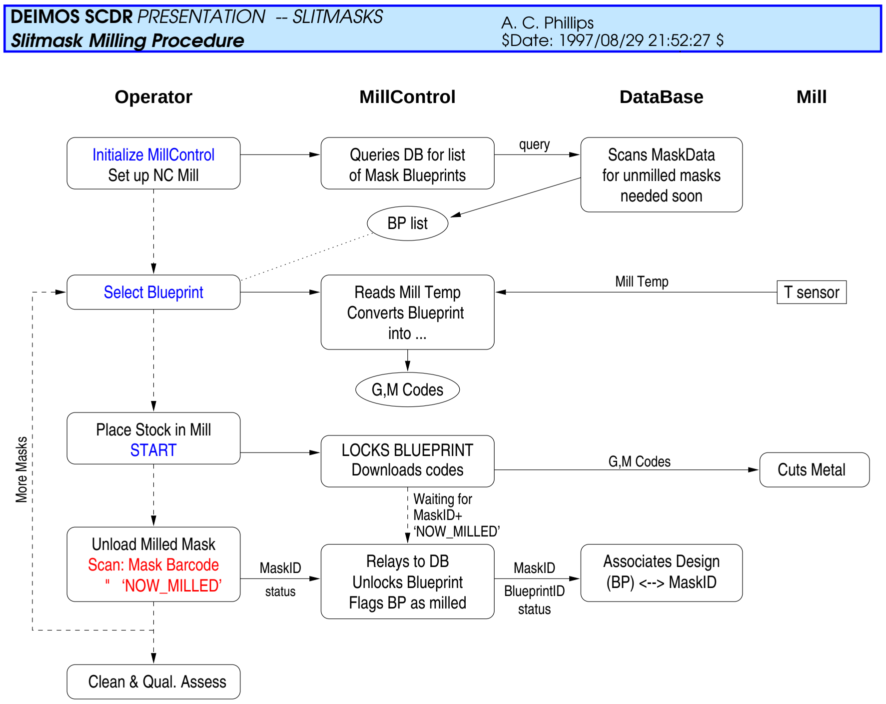


Figure 1.7: Slitmask Milling Procedure

Quality Control Procedure The quality control (QC) procedure could be done as an element of the milling procedure, or deferred until there is an entire batch of cut masks. If it is convenient, the mill operator may perform the Quality Control procedure on the previous mask while the next mask is being cut. If QC is done subsequent to the milling then all masks will already have been scanned as NOW_MILLED so that the association between blueprint and mask barcode is already in the database.

The following steps are performed for each slitmask.

- The mask is cleaned of any milling debris.
- The mask is inspected visually and classified good or bad.
- A two-step scanning operation is performed.
  - The hand-held scanner is used to scan the pre-affixed mask barcode.
  - The hand-held scanner is used to scan the appropriate quality barcode from the nearby chart, either MASK_GOOD or MASK_BAD.
- If the mask is classified GOOD then

  The mask is placed into the pre-observation holding bins. MillControl updates the database to mark the mask as good and removes the blueprint from the queue of things-to-do.

  Else if the mask is classified BAD then

  The mask is tossed aside for recycling/disposal. MillControl updates the database to mark the mask as junked and leaves the blueprint in the queue of things-to-do.
- Repeat the above for the next mask.

Responses of MillControl to a MaskId barcode Whenever the scanner scans a slitmask barcode, MillControl queries the database to see if a Mask record with that MaskId already exists. MillControl will then wait for a QC barcode, and if one does not arrive within a few seconds it will take action as if QUERY had been scanned. If MillControl receives two successive slitmask barcodes without an intervening QC barcode it will ignore the first slitmask.

Responses of MillControl to various QC barcodes If the scanner receives a QC barcode but has not recently scanned a slitmask barcode:

> MillControl displays an informational message indicating the QC barcode which was scanned and that no action was taken.

If the scanner reads QUERY then MillControl does the following:

> If the Mask is already in the database then present a detailed list of its known attributes. If the Mask is not already in the database then report that fact.

If the scanner reads NOW_MILLED then MillControl does the following:

> If MillControl is not awaiting an identity then this is an error. If the Mask is already in the database with a non-null MaskQual then MillControl must flag this as an error and require the mill operator to perform a detailed verification sequence. Otherwise, insert a new Mask record into the database containing MaskId, BluId, and all milling information except for MaskQual. MillControl retains the blueprint in the queue of things-to-do but marks it so that it cannot be selected for another DNC download.

If the scanner reads MASK_BAD then MillControl does the following:

> If the Mask is already in the database with a non-null MaskQual then MillControl must flag this as a likely error requiring a verification sequence. If it is in the database with a null MaskQual then update the Mask record setting MaskQual BAD. Otherwise, insert a new Mask record into the database containing MaskID, BluId, milling info, and a MaskQual of BAD. MillControl retains the blueprint in the queue of things-to-do and marks it so that it may be selected again.

If the scanner reads MASK_GOOD then MillControl does the following:

> If the Mask is already in the database with a non-null MaskQual then MillControl must flag this as a likely error requiring a verification sequence. If it is in the database with a null MaskQual then update the Mask record setting MaskQual GOOD. Otherwise, insert a new Mask record into the database containing MaskID, milling info, and a MaskQual of GOOD. MillControl

removes the blueprint from the queue of things-to-do, and sends e-mail to the owner of the blueprint indicating that the mask has been manufactured.

In case of failure of the barcode scanner, MillControl will accept the mask ID (barcode) and quality via keyboard entry rather than via scanner.

The important element of the Milling Procedure is that MillControl will not permit a new DNC download to be started until a mask barcode is associated with the blueprint for the previous DNC download. This forces the operator to scan each slitmask as it is removed from the mill. The mask status of NOW_MILLED permits complete flexibility in the scheduling of the quality control phase. The QC can be done subsequent to the milling of all masks, or it can be done while the mill is cutting another mask.

Slitmask Stock Check-out Some observers may wish to cut their own slitmasks. In that case the observatory will ship a number of blank slitmasks to the observer in advance of the run. MillControl will enter the identity of the observer. It will then permit masks to be scanned and tagged as EXPORTED. When the requisite number of masks has been checked out MillControl will print out the destination information for shipping.

Retirement of Slitmask Stock and Data It will eventually become necessary to dispose of the accumulated mass of previously used slitmasks. In order that the database of available slitmasks remain intact MillControl must be used to track the disposal. As each mask is discarded the mill operator must scan its barcode followed by scanning DISCARDED. The mask blueprint records for discarded masks may be deleted as part of the same operation.

1.6 Deliverable Documents

There are three main documents of relevance:

1. a User Manual for mask design (the "cookbook");
2. an Operator's Manual for slitmask fabrication, describing data ingestion, execution of MillControl, and procedures for slitmask fabrication.
3. The Calibration Document, which will describe the coordinate mappings used in SIMULATOR and MAPMASK.

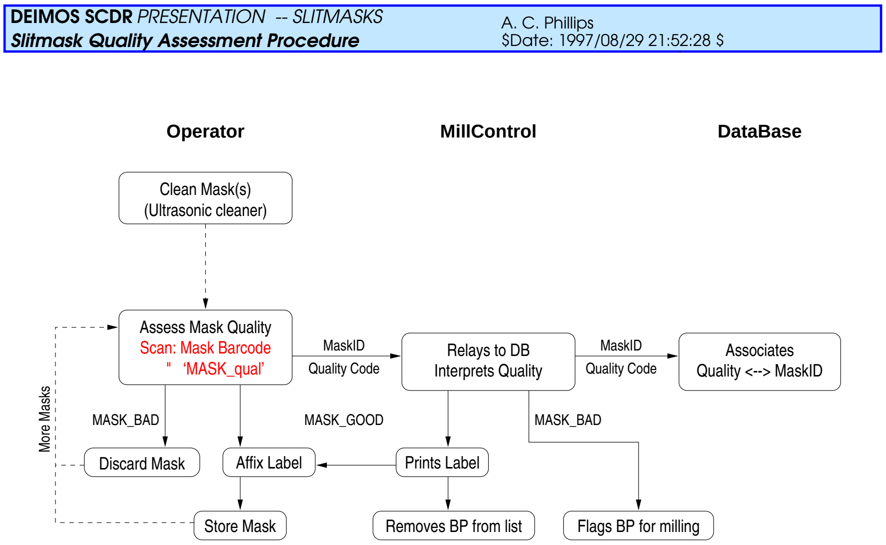


Figure 1.8: Slitmask Quality Assessment Procedure

Chapter 2

Acquisition & Slitmask Alignment

2.1 Overview

This subsystem is used to acquire a field and align a slitmask with respect to the sky. In practice, field acquisition is performed by the Observing Assistant, using standard DCS commands to place user-supplied coordinates at a desired pointing origin. Therefore, the subsystem description here will focus primarily on the mask-alignment process, although one of the modules is useful for precise relative positioning for any purpose.

The modules described are XLATE and SMALIGN.

There is a dependency on the mask-design (database), specifically with the slitmask initialization procedure (see 2.5).

2.2 Subsystem Figures

See Figure 2.1, "Mask Alignment Flow."

2.3 Nomenclature

Alignment Star: Preselected star or other object to be centered in an alignment box.

Alignment Box: Small rectangular aperture on slitmask, in which alignment star is centered when the mask is properly aligned.

Guide Star: Preselected star or other object to be placed at a particular location on the TV, either for object acquisition or guiding.

Database Extract: A file on the local disk containing slitmask designs, extracted from the database by the slitmask initialization process.

2.4 Component Modules

2.4.1 XLATE

Function Calculate and apply telescope offset to translate $(x, y)^{CCD}_{observed}$ to$(x, y)^{CCD}_{target}$.
Language IRAF/spp
Type tool
Author A. C. Phillips
Revision N/A
Essay This basic module takes an input (x,y) in CCD pixels ("observed" coordinates), and an instrumental position angle and target (x,y) in pixels, and calculates the necessary telescope offset in arcseconds to translate from the observed to target locations. The user is queried for permission to apply these offsets, and DCS commands are issued upon confirmation.
For mask alignment, the observed coordinates are those of an alignment star, and the target coordinates those of the corresponding alignment box. In practice, the positioning of the guide star at its predicted TV coordinates *should* be accurate enough that XLATE will not be required.
The source of the input parameters (input and target coords, PA) may vary. For example, the input (x,y) may be selected by mouse-click on the image display, the PA taken from the image header, and the target coordinates preselected from the database extract. Another source for input (x,y) might be a centroiding routine. Initially, we will provide only the simple interface – that is, the user enters all parameters by hand.
This module duplicates the functionality of modifying DCS keywords INSTXOFF and INSTYOFF, but is provided as an easier means for the observer to achieve the desired moves.
Inputs observed/target CCD coord pairs; PA of observation
Outputs printed telescope offsets; DCS commands
Deployment Commissioning

Labor 20 hours
Keywords used by module Modifies DCS keywords: RAOFF, DECOFF, REL2CURR
Waits on DCS keywords: AXESTAT

2.4.2 SMALIGN

Function Calculate and apply offsets to telescope/rotator to align masks
Language IRAF/spp
Type tool
Author A. C. Phillips
Revision N/A
Essay This interactive module analyzes an alignment image (that is, a direct image through the slitmask). It first calculates the relative position of each alignment star within its box, and then solves for rotational/translational offsets. The user is queried for permission to apply these offsets, and DCS commands are issued upon confirmation. The user initially specifies an image to be analyzed. The image header is checked for slitmask name, and the alignment box locations for that slitmask are read from the database extract. [Alternatively, the box locations could be calculated from the subraster specifications, also in the image header.] Instrumental PA also comes from the image header. During the first interactive phase, the module presents x- and y-profiles across each alignment box. The user interacts by setting a sky level, and a centering algorithm finds the alignment star's position within its box. Options exist for "forcing" a position if the algorithm fails. Upon exiting this phase, the differential offsets (star–box) are compared to the box locations across the mask, and a rotation and translation are solved. Residuals to this solution are displayed in the second interactive phase. The user may modify the sample of star-box pairs to provide a more credible solution, if necessary. Upon exiting, the actual telescope RA, Dec and PA offsets are calculated, and DCS commands executed upon user confirmation.
Inputs image name [FITS header provides needed PA, subraster coordinates]
Outputs printed telescope/rotator offsets; DCS commands
Deployment Commissioning
Labor 80 hrs

Module flow diagram Figure 2.2, "Module SMALIGN."
Keywords used by module Reads FITS header keyword: ROTPOSN
Modifies DCS keywords: ROTDEST, ROTMODE, RAOFF, DECOFF, REL2CURR
Waits on DCS keywords: ROTSTAT, AXESTAT

2.5 Subsystem Procedures

2.5.1 Commissioning

Define the offsets between the telescope axis and the pointing origins for the slitmask, longslit, imaging field and TV centers.

Calibrate the actual slitmask–to–TV coordinate system mapping.

Calibrate the Position Angle zeropoint of the instrument to within about 0.1 degree (preferably much better than this).

2.5.2 Mask Initialization

Instrument Specialist loads the slitmasks in the instrument. Then, either the observer or instrument specialist selects the "initialize" option on the instrument GUI (Part I, Chapter **??**). This causes the system to cycle through the loaded slitmasks, scanning barcodes and querying the database for the slitmask design associated with each mask. The slitmask info is placed on a local disk so that operation during the night does not depend on the availability of the database.

2.5.3 Mask Alignment Procedure

Expected Mode:

1. OA acquires field, sets instrument PA
2. Insert slitmask
3. Translate to place guide star at its expected TV location
4. Execute module SMALIGN

Fallback Mode (if placement of Guide star at expected TV coordinates fails to put the alignment stars within their boxes):

1. OA acquires field, sets instrument PA

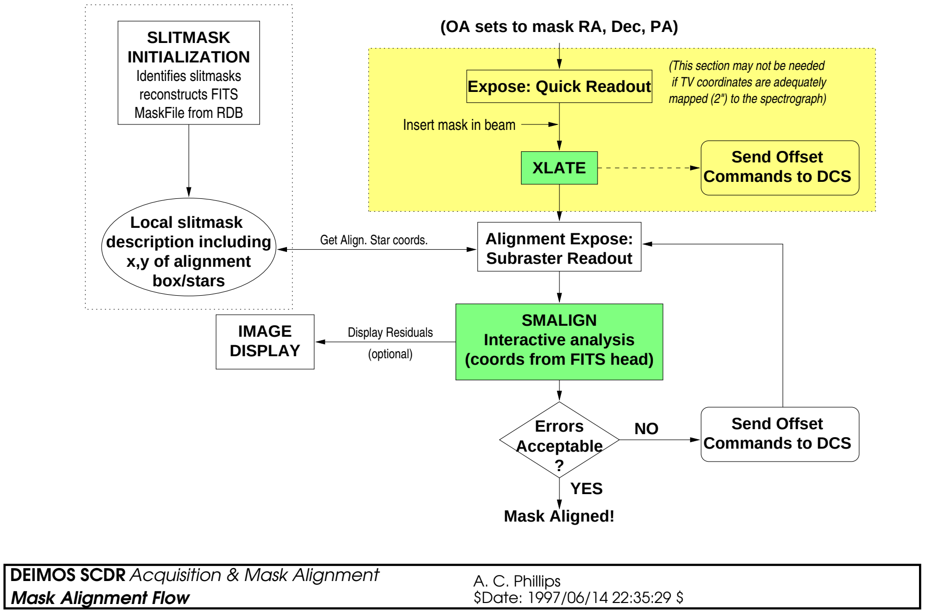


Figure 2.1: Mask Alignment Flow

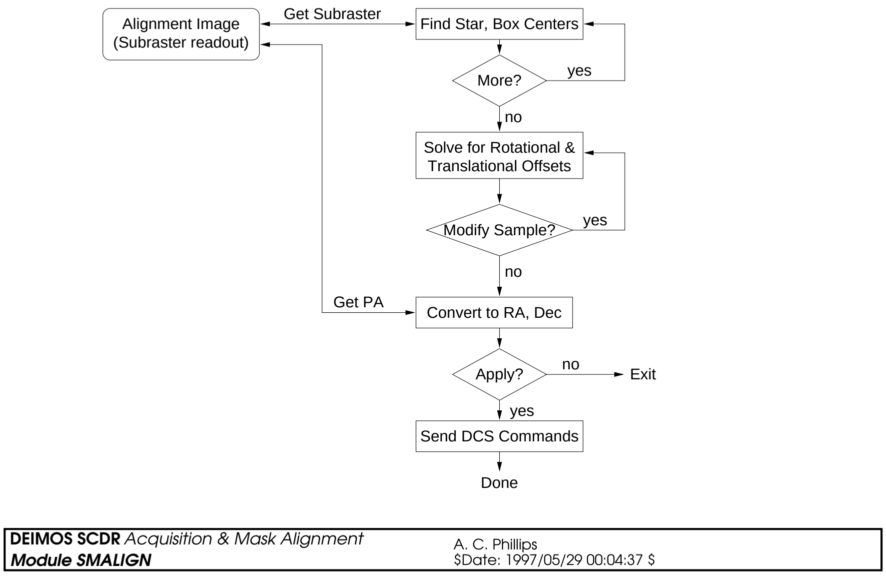


Figure 2.2: Module SMALIGN

2. Take direct image
3. Examine image to get alignment star coordinates
4. Execute module XLATE
5. Insert Slitmask
6. Execute module SMALIGN

2.6 Deliverable Documents

Instructions for the mask alignment procedure will be provided. These should be available in on-line form. Also, module SMALIGN will include built-in on-line help (eg., list of commands; sketch of normal operations; some troubleshooting).

Chapter 3

Spectrograph Control

3.1 Overview

The spectrograph control subsystem primarily connects commands and data requests from DEIMOS user interfaces and other processes such as watch_irot, the instrument rotation monitor, with the underlying mechanical hardware that moves optics, turns on lamps, controls temperature, and performs similar hardware control functions. See figure 3.1.

The DEIMOS spectrograph control subsystem consists of both pre-existing software modules already in use at Keck and new software modules developed at Lick for DEIMOS and other Keck II instrumentation. The pre-existing modules include:

- traffic - The communications hub process that passes keyword data in the form of MUSIC messages between processes.
- KTL support libraries - Libraries providing necessary support functions and data for KTL clients, such as user interfaces.
- show - A command line KTL keyword interface that displays keyword values.
- modify - A command line KTL keyword interface that changes keyword values, and possibly device states.
- waitfor - A command line KTL keyword interface that waits for a keyword to achieve a specified value.
- xshow - An X window KTL keyword interface that displays keyword values continuously.

These pre-existing modules will be included in the component module section of this document, but will not be covered in any great detail. References to existing documents describing them will be included.

New software modules unique to the DEIMOS spectrograph control subsystem include:

- UHAUL - a process running on the Galil motor controllers. It handles manual control requests from manual control boxes for each controller.
- BEKINS - a process running on the Galil motor controllers. It handles move requests from UHAUL and dispatch.
- dispatch - a process that receives keyword data requests and sends the requests to a controller or controller process like BEKINS, in the controllers native language.
- dispatch I/O libraries - seperate libraries, each containing the I/O protocol routines unique to a device type interfaced to the dispatch process.
- DEIMOS FIORD keyword library - a library required by KTL clients such as the show and modify processes to send and receive DEIMOS keyword data.
- watch_irot - the instrument rotation monitor, similar to the watch_irot task used with the HIRES image rotator.

New software modules related to the spectrograph control subsystem, described in detail in other chapters of this document include:

- dashboard[1] - the graphical user interface tool we will use to create a DEIMOS GUI.

3.2 Figures

[1] See Part I, chapter 3.3.

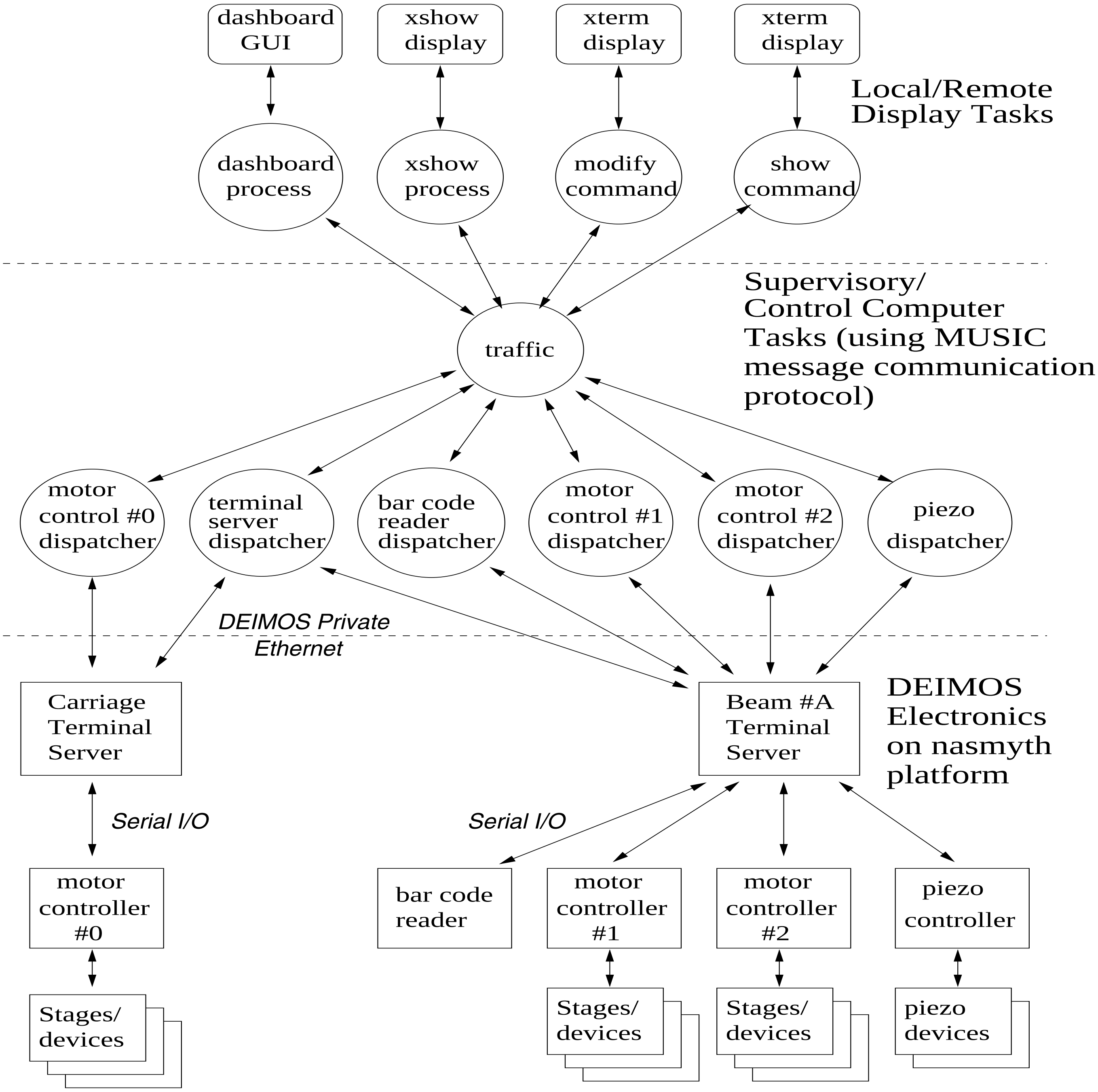


Figure 3.1: Spectrograph Control

3.3 Nomenclature

dashboard: The module using keyword information from the database to create flexible graphical user interfaces for DEIMOS.

dispatch: The hardware control interface module which communicates via traffic with MUSIC messages to KTL clients, such as dashboard, and other MUSIC message clients, such as watch_irot.

FIORD: FITS Input/Output Routine Database.

FIORD keyword library: A library of keyword data I/O routines and a table describing keyword I/O information used with KTL to provide access to an instrument or subsystem's set of keyword data.

KICS: Keck Instrument Control System

KTL: Keck Tasking Library, developed at Keck and used by Keck instruments and telescope control system to implement keyword-based multi-process applications for instrument control.

modify: A shell command used to change a KICS keyword value. "modify" uses KTL and a FIORD keyword library to accomplish this goal.

MUSIC: Multi-User System for Instrument Control. An interprocess communications protocol developed at Lick for transporting data between user interface processes and instruments. Currently in use at Lick and at Keck.

show: A shell command used to display a snapshot of a KICS keyword value. "show" uses KTL and a FIORD keyword library to accomplish this goal.

xshow: A shell command used to start a simple X window process continuously displaying keyword-value pairs, or the X window process itself. The xshow display uses KTL and a FIORD keyword library to access and update the display.

waitfor: A shell command used to wait for a KICS keyword to equal a value. "waitfor" uses KTL and a FIORD keyword library to accomplish this goal.

3.4 Component Modules

3.4.1 BEKINS

Function Service requests from UHAUL or dispatch.
Language Galil DMC-1500 motor controller command language.
Type daemon
Author J. Burrous
Essay The BEKINS process runs on Galil DMC-1500 series motor controllers. It is the daemon process that handles all stage motion. This includes calibrating the stages by finding home references, moving the stages to a physical position, and stopping moves in progress.

BEKINS communicates to other processes via variables maintained in motor controller memory. The UHAUL and dispatch processes send stage movement request data to BEKINS by setting these variables. They also check movement status by reading variables set by BEKINS. Every stage on a motor controller has an independent set of the following variables used by BEKINS to satisfy their interprocess communication needs for stage motion requests:

- InitAck - initialization acknowledgement. This variable is set to TRUE when BEKINS finishes handling an initialization (calibration) request. The stage hasn't completed the request at this time, BEKINS is just telling UHAUL or dispatch it is acting on the request.
- InitReq - initialization request. This variable is set TRUE by UHAUL or dispatch to request stage initialization (calibration).
- InitFaz - initialization phase. It takes several steps to calibrate a stage. This variable keeps track of what step is being executed. BEKINS can start a stage moving as part of the initialization of

the stage, increment the InitFaz variable to tell it what the stage is doing when it returns after servicing other stage's needs.

- Homed - A variable that tells UHAUL or dispatch, wheither or not a stage is initialized (calibrated). This flag is FALSE at motor controller power up. BEKINS sets this flag to FALSE at the start of an initialization request. It sets the flag to TRUE when initialization is completed.
- Wraps - not used by BEKINS for control. Tells how many revolutions a rotary stage has been moved from its home reference.
- CurrPosR - Current raw position.
- MoveReqJ - Jog move request flag.
- JogFwd - Jog forward flag.
- JogRev - Jog reverse flag.
- JogSpd - Jog speed.
- MaxSpd - Maximum speed allowed for stage.
- MoveReqR - Raw move request flag.
- MoveAckR - Raw move acknowledge flag.
- TargetR - The target raw position value.
- MoveFaz - A movement phase counter.
- StopReq - The stop request flag.
- MoveReqD - Discrete move request flag.
- MoveAckD - Discrete move acknowledge flag.
- TargetD - The target discrete position value.

An important feature of the request flags described above (MoveReqJ, MoveReqD, MoveReqR, StopReq and InitReq), is that they are interlocked by the UHAUL and dispatch processes. Only one move request flag can be set at a time. If a move is in progress, the stop flag must be set prior to starting a new move, and UHAUL or dispatch must wait for the stage to stop before setting the new move parameters, including the request flags for the new move. The same is true for initializing a moving stage, the stage must be stopped, then the new request flags are set.

Another important feature of BEKINS is how it handles long duration actions, such as waiting for a motion to complete. While waiting for an initialization, move or stop action to complete, BEKINS always continues to service new requests, and start new actions required by other stages to complete requests pending on them.

The module flow diagrams for BEKINS use the variables described above to show the actions of the BEKINS process as it watches for requests or acts on requests.

The BEKINS process consists of three basic parts executed in the order shown on the first flow diagram (figure 3.2). An initialization section handles incoming initialization requests and steps the stage through the sequence of actions required to initialize stages. A movement section handles incoming move requests of any of the three types of move options: jogging, moving to a raw encoder position, and move to a position found in a look-up-table mapping discrete values (such as 0, 1, 2, 3,...) to raw encoder positions (such as -35245, -1834, 23456,...). After handling a move request, BEKINS will execute any sequencial actions required by the move. The stop section is similar to the move sections and the initialization sections, it handles stop requests and the sequence of actions required to stop the stage.

Each stage has a unique set of functions to perform the actions identified on the first flow diagram. Each stage's initialization routine is called during the initialization phase shown in figure 3.2. Each stage's move routine is called during the move phase, and each stage's stop routine is called during the stop phase of the BEKINS service loop.

The second flow diagram (figure 3.3) shows the common elements in a stage initialization function executed during the initialization portion of the BEKINS service loop. UHAUL and dispatch set the initialize acknowledge flag (InitAck) to FALSE and the initialize request flag (InitReq) to TRUE when they want a stage initialized. The following is an outline of some of the elements of the flow diagram.

- The stage initialization function tests the InitAck and InitReq variables to see if a new initialization request is pending (InitAck = FALSE and InitReq = TRUE). If a request is pending it sets the InitReq, InitAck, InitFaz and Homed variables, turns on fiducials and starts servoing the stage in preparation for the first initialization move.
- If the InitAck flag is TRUE at the start of the procedure, the stage has acknowledged an initialization request, but it hasn't finished initializing the stage. The procedure tests to see if the stage is done with its last commanded move for its initialization

sequence. If it is not done with the move, BEKINS drops out of the initialization function for this stage and goes off to handle other services.

- If the stage hasn't finished initializing (InitAck = TRUE) and the stage has stopped moving, BEKINS checks to see if the stage is done initializing (InitFaz = 0). If it isn't done yet it executes the next initialization step in the procedure and then goes off to handle other services. If the stage is done initializing, it sets the InitAck, Homed, Wraps, and CurrPosR (current position) variables, turns off the fiducials, sets limit positions, and then exits the procedure to handle other services.

Figure 3.4 shows the BEKINS process control flow through the jog move procedure. The jog move request flag (MoveReqJ) is tested followed by the direction information. Limits are tested prior to starting the jog. Then the jog speed and direction is set and the jog is started.

Figure 3.5 shows the BEKINS process control flow through the raw move procedure. The raw move request flag (MoveReqR) is tested followed by a test determining if a move is in progress. If a move is in progress a stop request is set up and the next pass through the service loop the same procedure is followed until the stage has stopped. If the stage is not moving the move acknowledge and move request flags are reset and the target value and move phase vaiables are set up to cause a move when the move handler is reached. If the stage isn't homed the initialization request flag is set to cause the stage to home at the time it moves.

Figure 3.6 shows the BEKINS process control flow through the discrete move procedure. It is identical to the raw move procedure, with the addition of the shortest path algorithm. The signal flags are changed to MoveReqD, MoveAckD and TargetD to signal discrete movements.

The move handler flow diagram in figure 3.7 shows a simple three step move procedure.

Figure 3.8 shows the final section of the BEKINS service loop, the stop function.

Inputs Request variables set by UHAUL and dispatch.

Outputs Response/status variables read by UHAUL and dispatch.

Module flow diagram Figure 3.2 BEKINS Program Flow - Overview

Figure 3.3 BEKINS Program Flow - Initialization

Figure 3.4 BEKINS Program Flow - Jog Moves

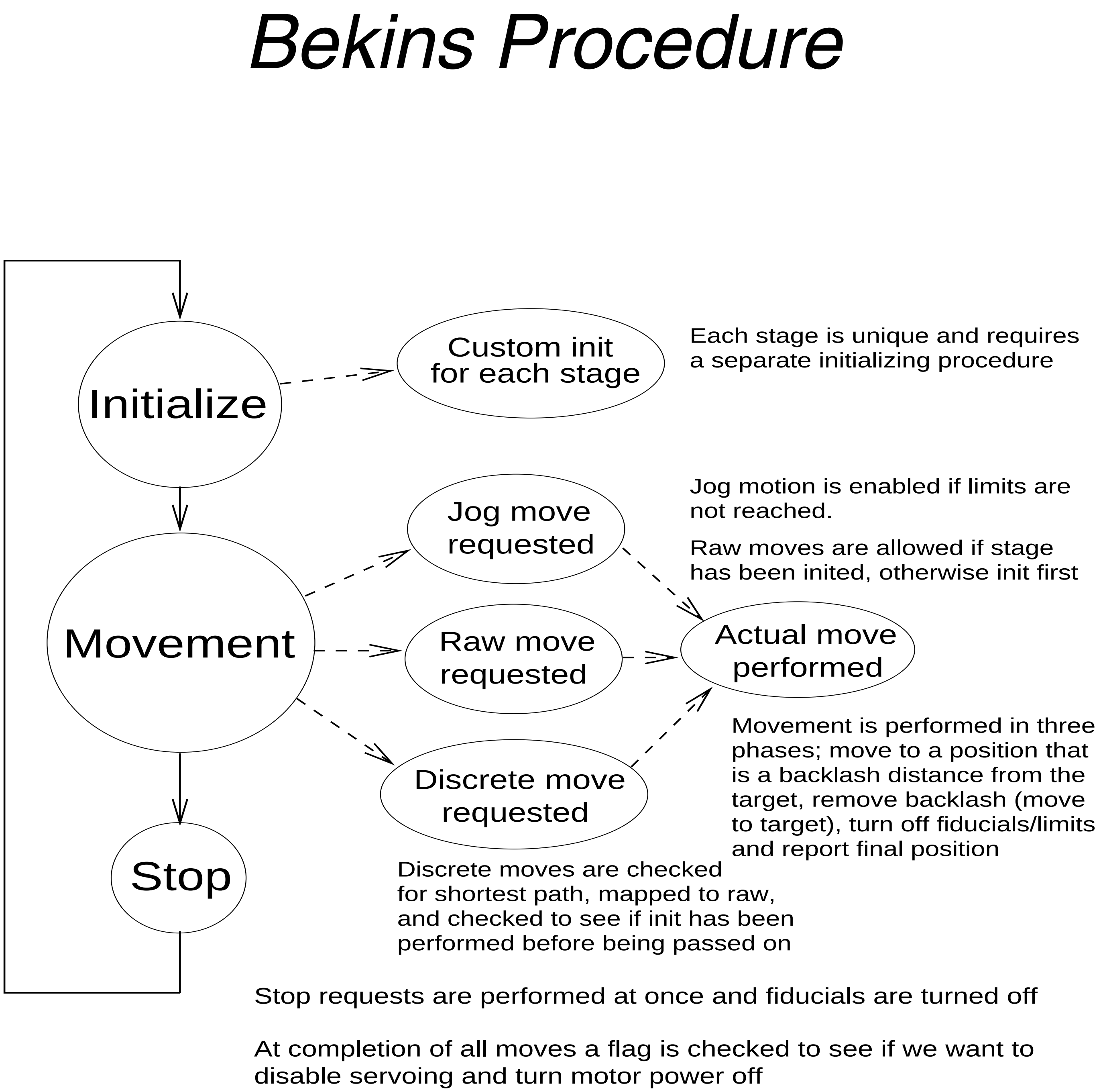


Figure 3.2: BEKINS Program Flow - Overview

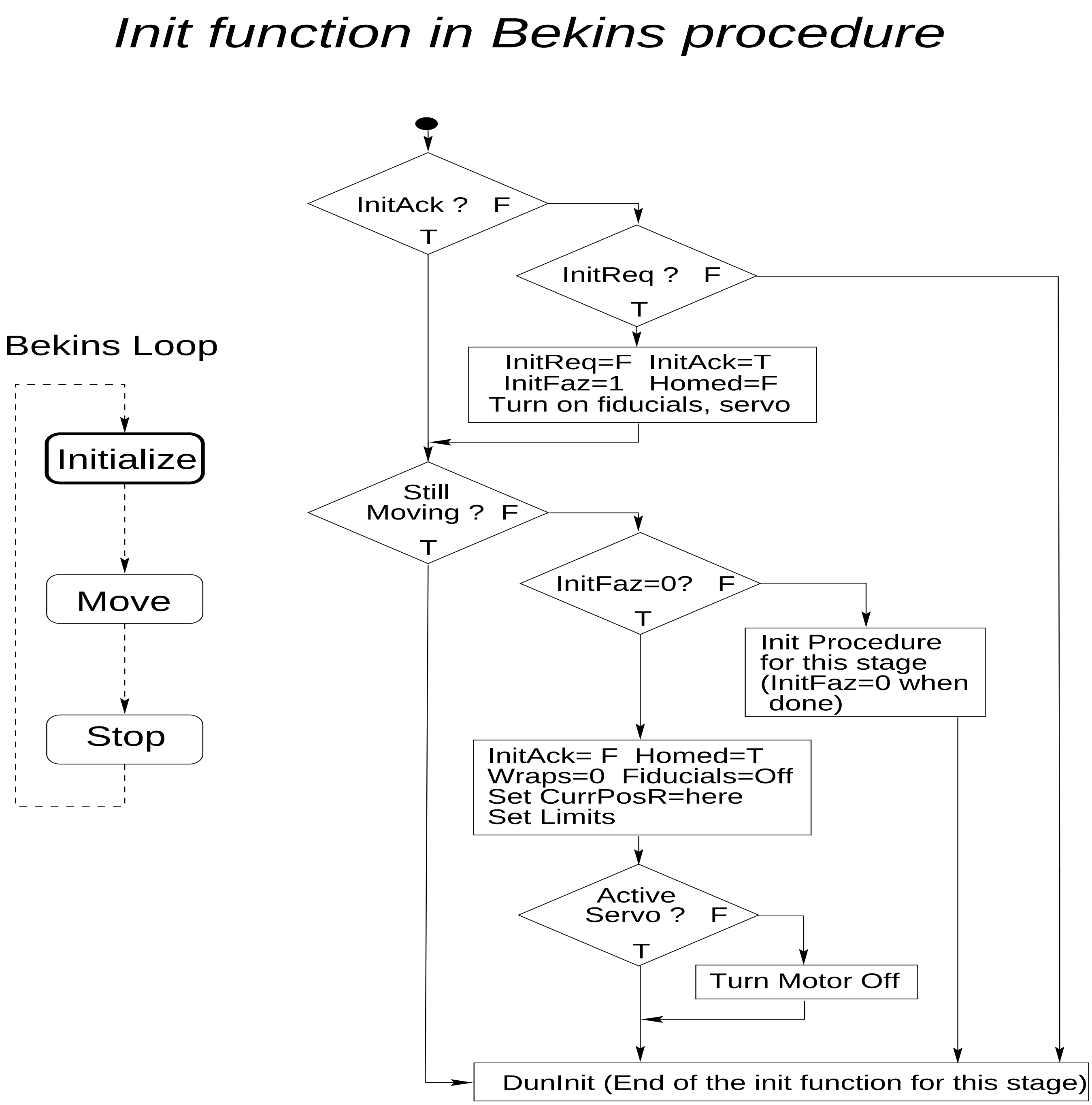


Figure 3.3: BEKINS Program Flow - Initialization

Jog move function in Bekins procedure

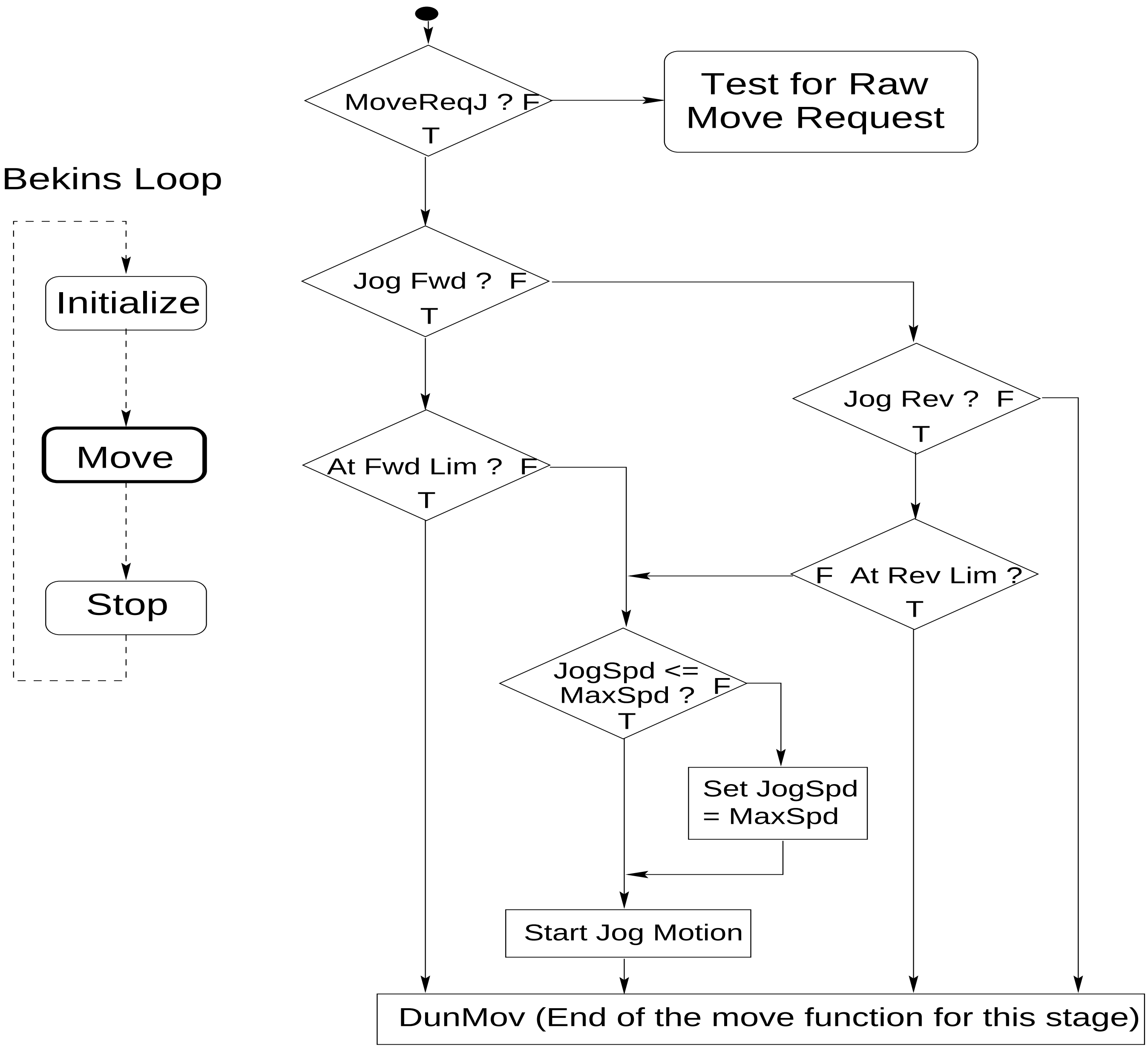


Figure 3.4: BEKINS Program Flow - Jog Moves

Figure 3.5 BEKINS Program Flow - Raw Moves
Figure 3.6 BEKINS Program Flow - Discrete Moves
Figure 3.7 BEKINS Program Flow - Move Handler
Figure 3.8 BEKINS Program Flow - Stop Procedure

Keywords used by module Modifies no keywords directly. Services no keywords directly. Keyword requests can cause responses by BEKINS. For example, setting the DWFILORD keyword can cause BEKINS to move the dewar filter wheel.

3.4.2 UHAUL

Function Monitors DEIMOS manual control box signals for manual spectrograph control requests. Passes manual control requests to BEKINS.

Language Galil DMC-1500 motor controller command language.

Type daemon

Author J. Burrous

Essay The UHAUL process runs on the Galil DMC-1500 series of motor controllers. It is the daemon process that catches digital input signals from the switches and buttons on the manual control box for a motor controller. UHAUL sets output signals to show control status information on the box, and sets interprocess communication variables shared with BEKINS and dispatch. The variables request actions by BEKINS to move stages or provide safe transfer of device control between dispatch and UHAUL.

A description of the following motor controller variables can be found the module description of the BEKINS process: StopReq, InitReq, MoveReqD, MoveReqJ, JogSpd, TargetR, and TargetD. These variables inform BEKINS of the type of action requested by UHAUL and pass on additional data required to perform the UHAUL request. Variables unique to UHAUL include:

- AutoMan - A flag indicating who has access to device control, UHAUL or dispatch. UHAUL has control when AutoMan = MANUAL. dispatch has control when AutoMan = AUTO. UHAUL sets this variable after all stages have stopped due to a change in the Auto/Manual switch on the control box.
- DirLok - An internal status flag to indicate the FWD or REV buttons were pressed.

An outline of the basic UHAUL service loop is in the upper right of

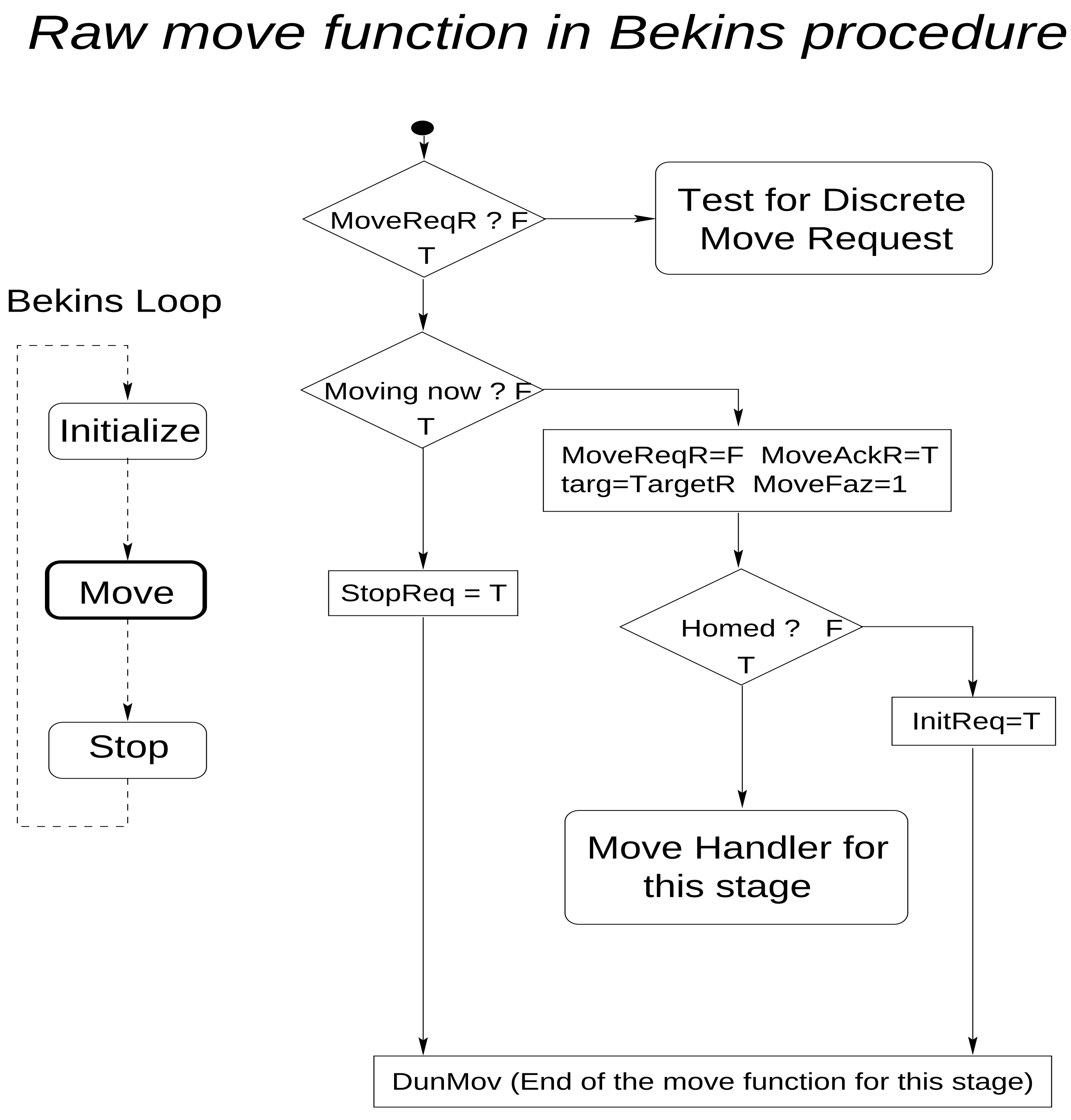


Figure 3.5: BEKINS Program Flow - Raw Moves

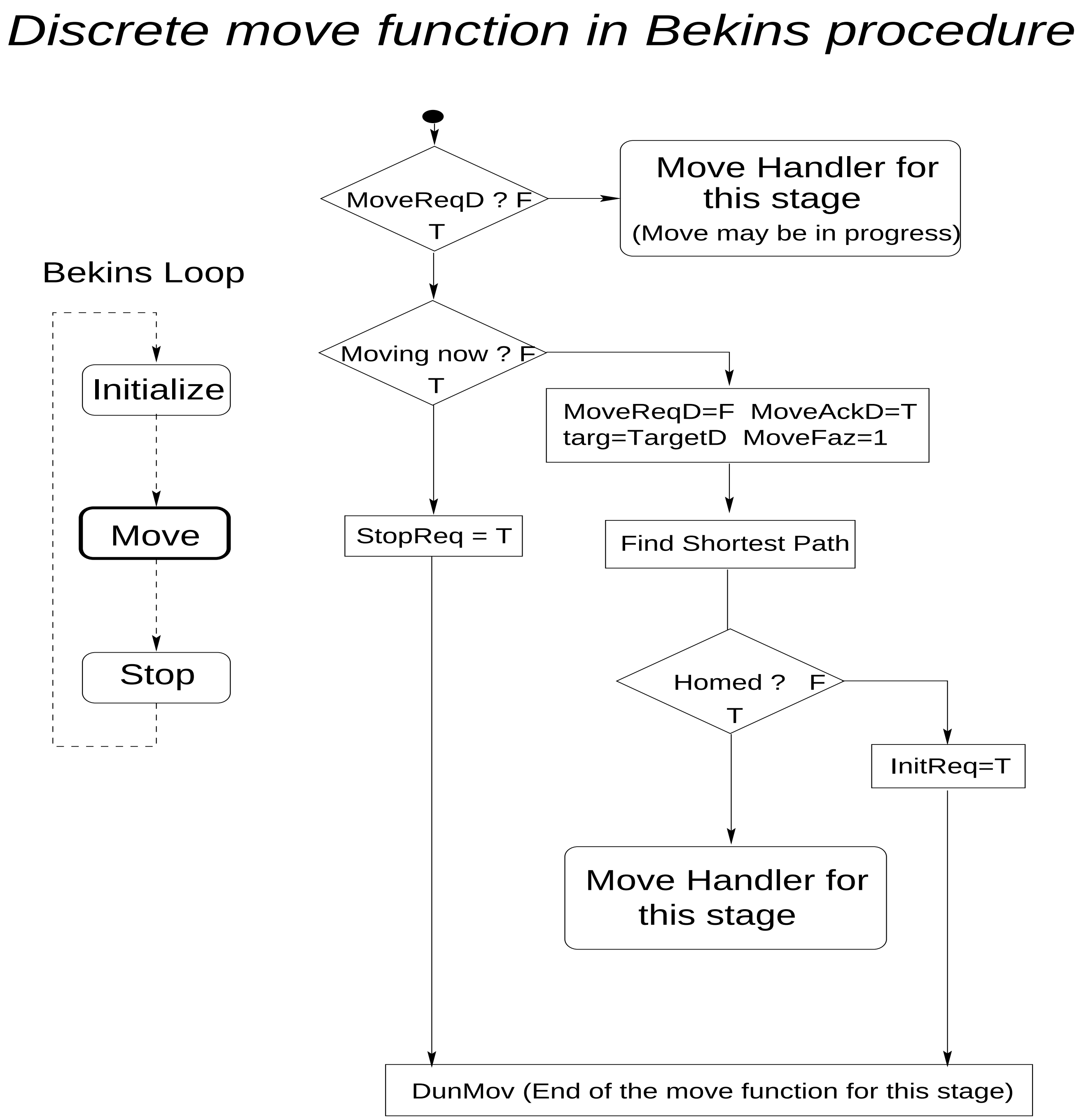


Figure 3.6: BEKINS Program Flow - Discrete Moves

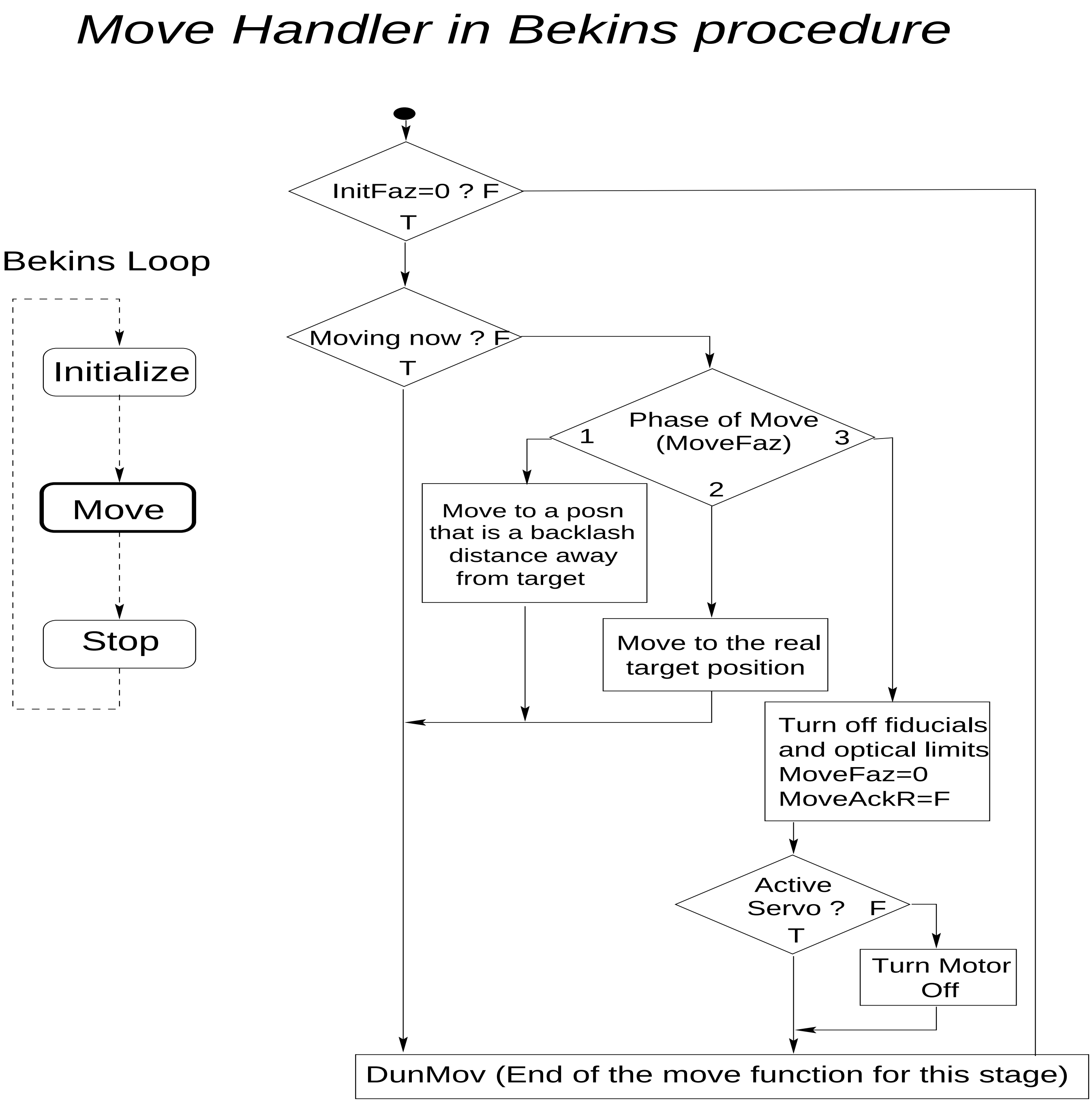


Figure 3.7: BEKINS Program Flow - Move Handler

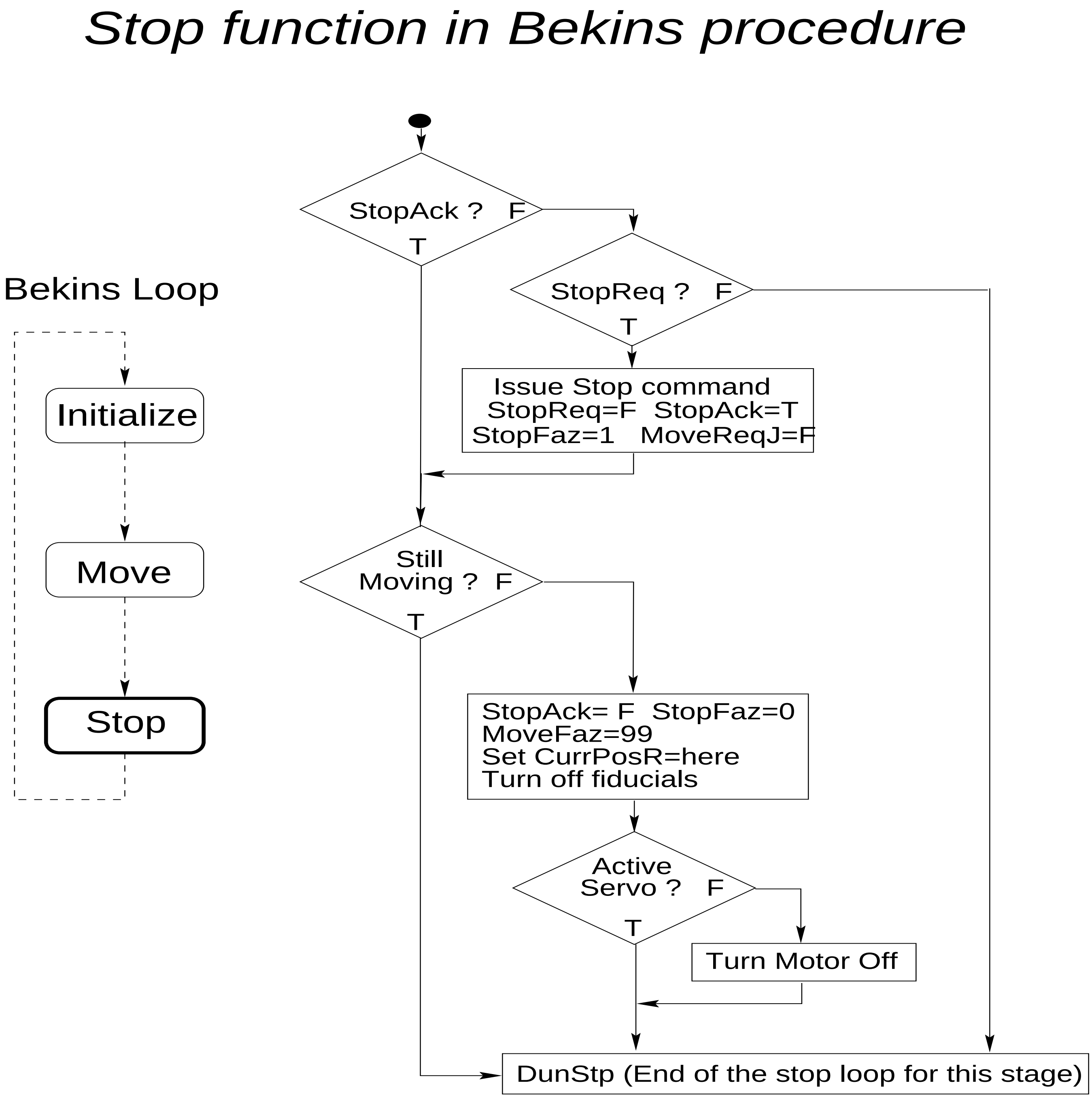


Figure 3.8: BEKINS Program Flow - Stop Procedure

figure 3.9 or 3.10. UHAUL checks the input signals from the switches and buttons on a manual control box, performs any actions required by the signals, and provides feedback to the operator at the control box via LEDs. The LEDs show movement and limit status data for motor stages and position data for air cylinder stages such as the hatch.

Figure 3.9 shows a key loop in the UHAUL task. When the Auto/manual switch changes state, UHAUL sets appropriate flags to cause BEKINS to stop of all motor controller motors and prevent either UHAUL or the dispatch process from moving any stages until all of the motors have stopped moving. Only after the AutoMan flag is set to Auto at the end of the loop can dispatch send stage move commands to BEKINS. Only after all motors have stopped can UHAUL exit the loop.

Figure 3.9 also shows the conditions that results in a reset (cold start) of the controller (hold all three motion control buttons (FWD, REV and STOP) down for three seconds), and a halt of all stage motion (hold the stop button down for three seconds). (Note: This last condition is likely to change such that the stop button always halts all stages immediately. It is also likely that UHAUL will be changed to allow only one stage to move at a time.)

Figure 3.10 selection of the move routine appropriate for the switch selected stage when any of the motion control buttons are pressed.

Figure 3.11 shows the conditions necessary to request one of the four following actions from BEKINS: stop a stage, initialize a stage, move the stage to a discrete position, or jog the stage. It also shows how each action, except the stop action is bypassed while the stage is still moving.

Inputs Signals from manual control box switches and buttons.

Outputs Sets BEKINS service request variables. Sets signals to control box LEDs.

Module flow diagram Figure 3.9 UHAUL Signal Handling Part 1
Figure 3.10 UHAUL Signal Handling Part 2
Figure 3.11 A Typical Manual Move Procedure for UHAUL

Keywords used by module Modifies no keywords directly. Its actions can change the values of keywords for devices controlled by the manual control box, such as HATCHPOS.
Services no keywords.

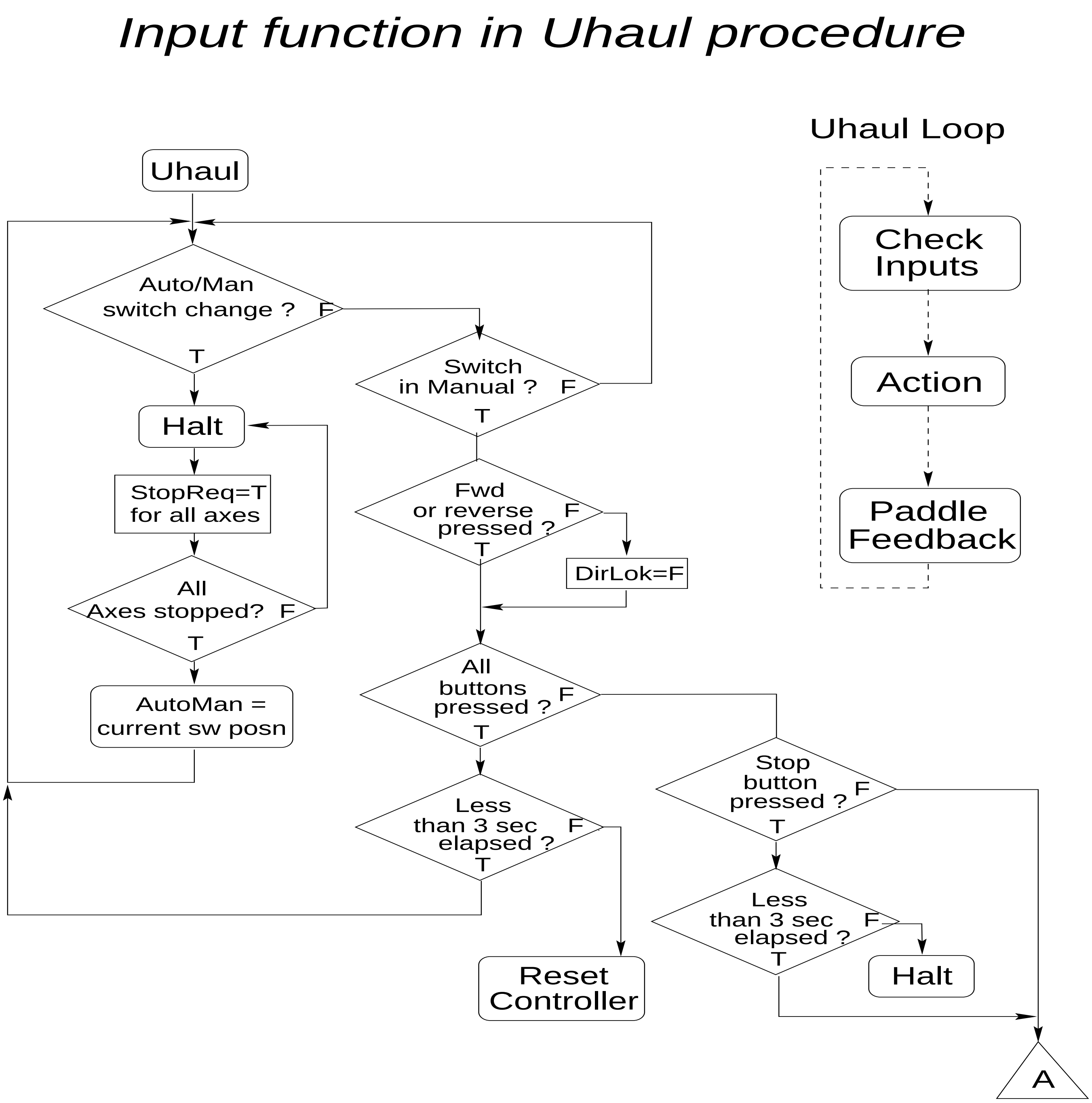


Figure 3.9: UHAUL Signal Handling Part 1

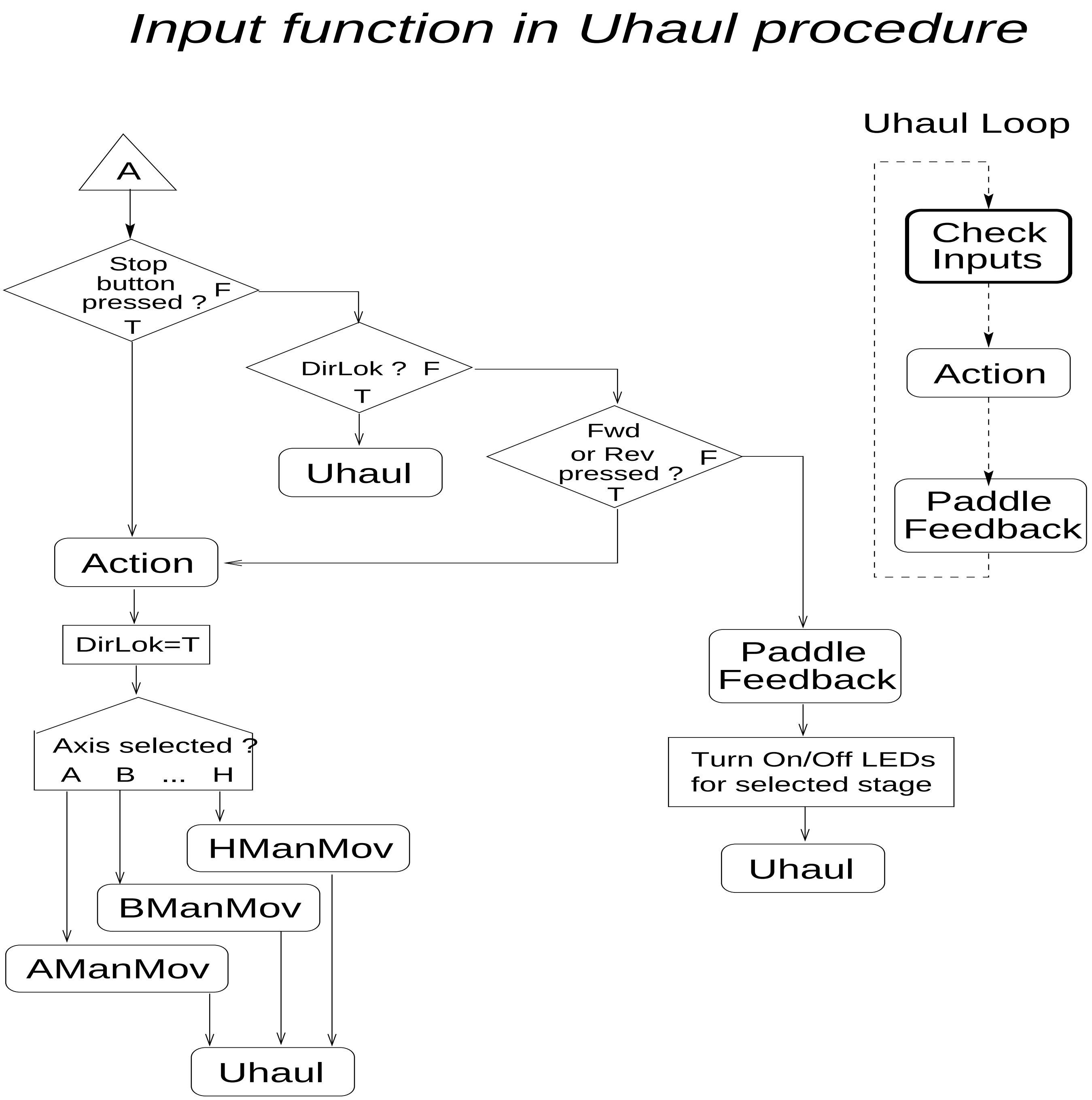


Figure 3.10: UHAUL Signal Handling Part 2

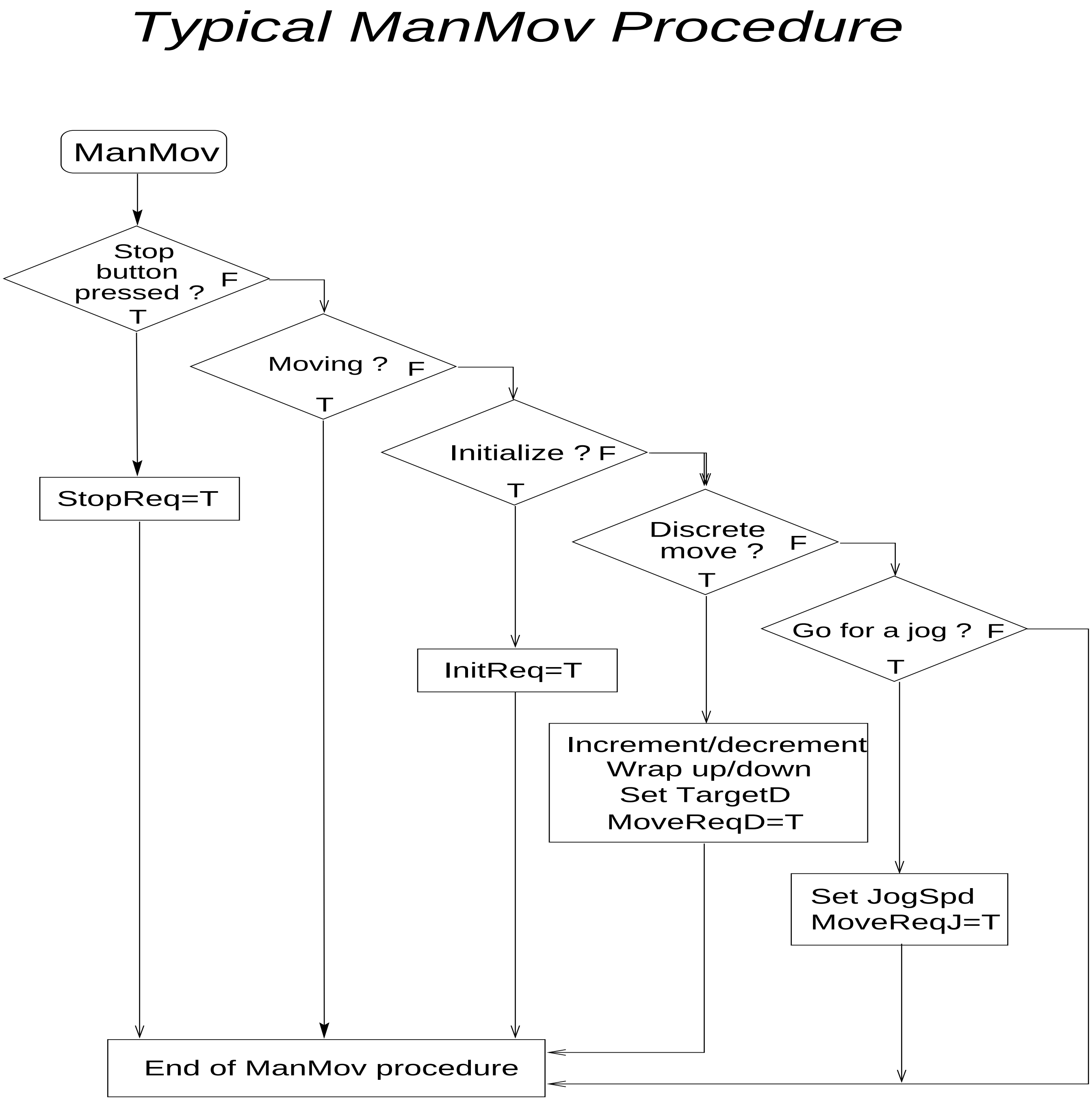


Figure 3.11: A Typical Manual Move Procedure for UHAUL

3.4.3 dispatch

Function dispatch has several important functions

- It services keyword requests for spectrograph control and data by passing requests to motor controllers, bar code readers and piezo controllers.
- It performs high level control and monitoring functions not appropriate for the controllers to perform. This includes setting keywords for other processes and itself.
- It maintains keyword data internally and in flat files for table/map driven device information such as name-ordinal-raw position mappings, and conversion constants for raw-cooked (xxxxxRAW to xxxxxVAL keyword) maps.
- It validates corresponding configuration data between controllers and the database.

Language ANSI standard C
Type daemon
Author D. Tucker
Essay The dispatch process should be able to run on any Unix system that it can be compiled on. It is the daemon process that services keyword requests directed to instrument control hardware, monitors and controls functions such as dewar filling, and maintains and validates mapping data relating instrument control keywords.

Dispatch communicates with clients requesting keyword data or keyword changes via MUSIC messages. The MUSIC messages arrive on a socket connected to the traffic process. Dispatch returns response messages via traffic sending them in the opposite direction. Dispatch passes change requests to hardware controllers, such as the bar code readers and Galil controllers, in their own language.

Dispatch uses the message service type and device id in the message to map data service requests arriving as music messages to internal devices and their information structures. Data in the device information structures identifies the controller with the hardware associated with the internal dispatch device and I/O routines required to access its data. Data in the controller information structure identifies the port data, information and I/O routines required to access the devices data on the hardware controller (e.g. a Galil controller).

Figures 3.12 through 3.14 describe the major elements of dispatcher control flow. The most important of these is figure 3.13 showing the event sequence of the dispatcher service loop. It consists of five autonomous sections. They could almost be seperate processes sharing the same memory. The first section handles reconnection to traffic and serial ports when connections are lost. The second section reads all status information in one gulp from a controller or simple device with a serial interface. The third section processes incoming MUSIC messages until no more are pending. The fourth section checks the status data read from section two and the message request data from section three to determine if any responses to request messages need to be sent due to a device completing a change request or having a change request cancelled. The last section of the loop checks for any changes to device state data. Broadcasts messages are sent with all of the new device states.

Inputs MUSIC messages, controller status data, configuration data files.

Outputs MUSIC messages, controller commands, configuration data files.

Module flow diagram Figure 3.12 Dispatch Overview

Figure 3.13 Dispatch service loop

Figure 3.14 Service MUSIC messages

Keywords used by module Can modify the value of all readable DEIMOS instrument control keywords.

Services requests for all writeable DEIMOS instrument control keywords.

3.4.4 Galil DMC-1500 Dispatch I/O Library

Function Handles I/O protocols between dispatch and Galil DMC-1500 series controllers.

Language ANSI standard C

Type library

Author D. Tucker

Inputs Galil controller and device I/O requests.

Outputs Galil controller and device data.

Keywords used by module The Galil DMC-1500 dispatch I/O library handles the keyword data and device I/O for all stages and other hardware controlled by Galil DMC-1500s. I/O for all DMC-1500 controller keywords (CTRLxxxx and MANxxxxx keywords) is also managed by the library.

Dispatch Daemon - Overview

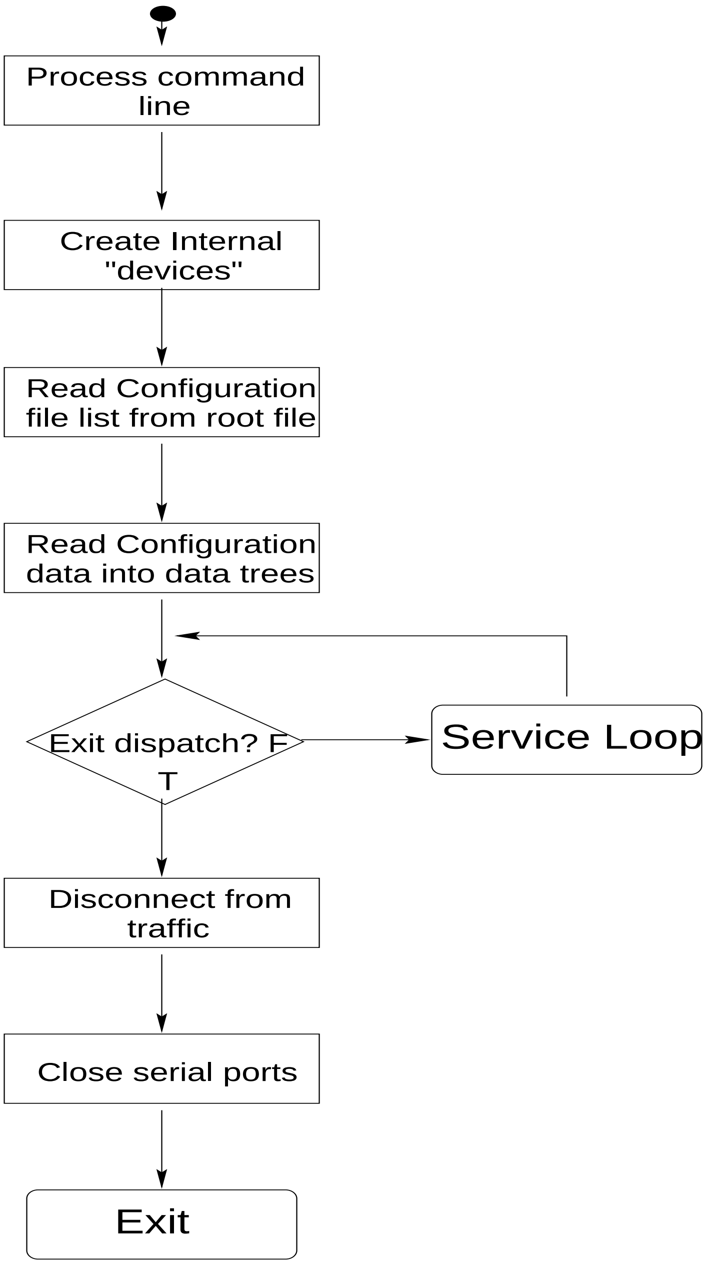


Figure 3.12: Dispatch Overview

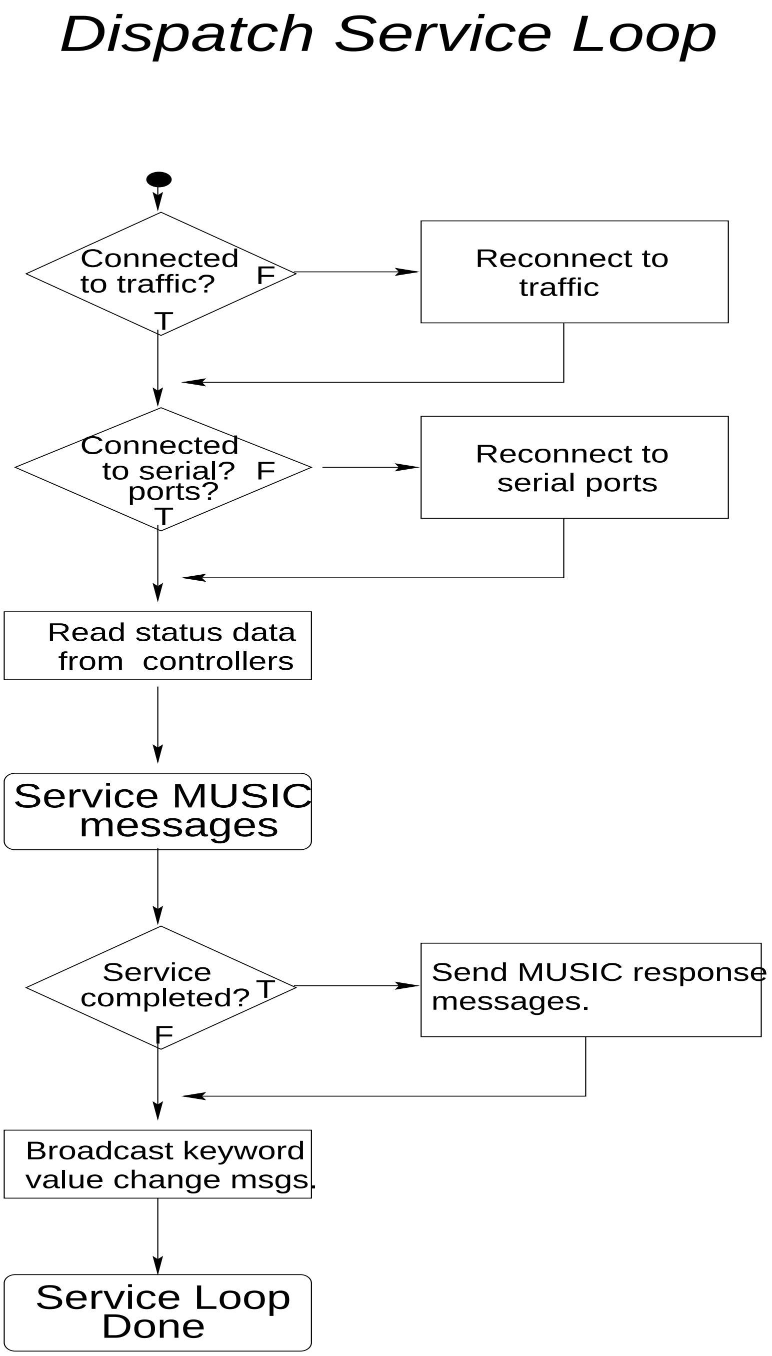


Figure 3.13: Dispatch service loop

Service MUSIC messages

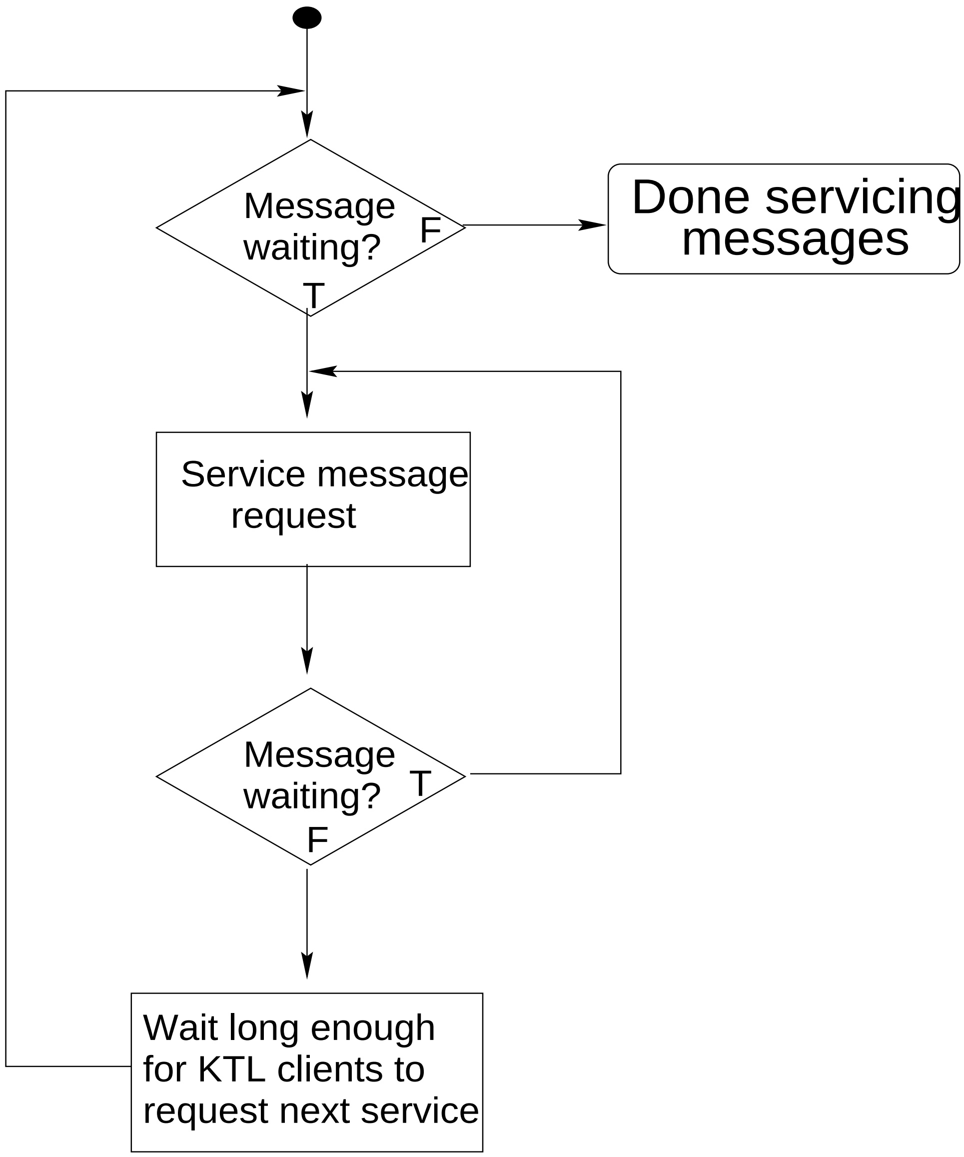


Figure 3.14: Service MUSIC messages

3.4.5 Bar Code Reader Dispatch I/O Library

Function Handles I/O protocols between dispatch and bar code readers.
Language ANSI standard C
Type library
Author D. Tucker
Inputs Bar code reader I/O requests.
Outputs Bar code reader data.
Keywords used by module The bar code reader dispatch I/O library handles the device I/O for all bar code reader key words (xxBARCFG, xxBARCMD, xxBARGET, xxBARRSP, xxBARSCN.

3.4.6 Piezo Controller Dispatch I/O Library

Function Handles I/O protocols between dispatch and piezo controllers.
Language ANSI standard C
Type library
Author D. Tucker
Inputs Piezo Controller I/O request data.
Outputs Piezo Controller data.
Keywords used by module The piezo controller dispatch I/O library handles the device I/O for all bar code reader key words (TMIRRCAL, TMIRRHLT, TMIRRMSG, TMIRRNAM, TMIRRORD, TMIRRRAW, TMIRRTRG, TMIRRVAL).

3.4.7 Lantronics Terminal Server dispatch I/O Library

Function Handles I/O protocols between dispatch and lantronix terminal servers.
Language ANSI standard C
Type library
Author D. Tucker
Inputs Lantronix terminal server I/O request data.
Outputs Lantronix terminal server data.
Keywords used by module The Lantronix terminal server dispatch I/O library handles the device I/O for all Lantronix terminal server key words (TBD).

3.4.8 watch_irot

Function Instrument rotation monitor task

Language ANSI standard C

Type daemon

Author B. Kibrick

Essay Watch_irot watches the telescope position via requests to DCS and, based on the desired rotation mode, sends position or velocity commands to the instrument rotator to keep the image positioned correctly.

Inputs MUSIC messages, configuration data files.

Outputs MUSIC messages.

Keywords used by module Reads ROTDETS1, ROTPDSTS and ROTDETS2 from DCS. Reads IROT2ANG from the instrument. Controls the instrument velocity (and position) via ROTATVEL. Sets the instrument control mode (position, velocity) via ROTATMOD. Sends messages to the instrument user interface via ROTATMSG. Sends status information to DCS via ROTSRVER, ROTPPOSN, ROTSTAT, ROTSTST and ROTERRS.

3.4.9 DEIMOS FIORD keyword library

Function Define DEIMOS keywords and functions for formatting keyword data and transmitting it between KTL clients such as show and modify, and server processes such as dispatch.

Language ANSI standard C

Type library

Author D. Tucker

Essay The DEIMOS keywords are defined in the database. See Lick Observatory Technical Report #80, part II, chapter 1. The DEIMOS FIORD keyword library will be automatically generated from the data in the database.

Inputs N/A, library used by other modules

Outputs N/A, library used by other modules

Keywords used by module Provides protocol for passing keyword data between processes, but does not modify or wait for specific keywords.

3.4.10 traffic

Function Passes MUSIC messages between processes.
Language C
Type daemon
Author R. Stover
Essay The traffic process receives MUSIC messages from client processes and pass on the messages to other client processes. The messages usually contain keyword data being transmitted between the client processes.

For a complete description of traffic, see Lick Technical Report #55.
Inputs MUSIC messages.
Outputs MUSIC messages.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.

3.4.11 KTL support libraries

Function Provide support functions needed by KTL clients, such as the show and modify commands or dashboard.
Language C
Type library
Author CARA
Essay The KTL support libraries include ktl, ktlker, kicsglue, fiord, music, info, d, and traffic.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.

3.4.12 modify

Function Change a keyword value from a shell command line.
Language C
Type command
Author A. Conrad
Revision
Essay A command line application written for Keck I. It is invoked with a list of keyword=value pairs. It can either wait for the keywords to equal the specified values, or exit before the change completes.

modify is a KTL client, it uses the KTL support libraries and a FIORD keyword library to access keyword data.

Inputs List of keyword names and values.
Outputs Result of request.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.
Keywords used by module Can change any writable keyword.

3.4.13 show

Function Display a keyword value from a shell command line.
Language C
Type command
Author A. Conrad
Revision
Essay A command line application written for Keck I. It is invoked with a list of keywords to display. The information is displayed as keyword=value pairs in the window.

show is a KTL client, it uses the KTL support libraries and a FIORD keyword library to access keyword data.

Inputs List of keyword names.
Outputs Keyword name and value pairs.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.
Keywords used by module Can display any readable keyword.

3.4.14 waitfor

Function Wait for a keyword value to equal a specified value.
Language C
Type command
Author A. Conrad
Revision
Essay A command line application written for Keck I. It is invoked with a list of keyword=value pairs. It waits for the keywords to equal the specified values.

Waitfor is a KTL client, it uses the KTL support libraries and a FIORD keyword library to access keyword data.

Inputs List of keyword names and values.

Outputs Result of request.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.
Keywords used by module Can wait for any writable keyword to change.

3.4.15 xshow

Function Continuous display of keyword values.
Language C
Type daemon
Author A. Conrad
Essay An X window application written for Keck I. It is invoked with a list of keywords to display in the window. The information is displayed as keyword-value pairs in the window.

Xshow is a KTL client, it uses the KTL support libraries and a FIORD keyword library to access keyword data.

Inputs Keyword names and values.
Outputs Keyword name and value pairs in an x-window display.
Deployment Currently in use at Lick and Keck.
Labor 0 hours. Completed product.

Can display any readable keyword.

3.5 Subsystem Procedures

3.5.1 Overall Instrument Control Data Flow

Figure 3.1 shows a diagram of the instrument control data flow. At the top KTL clients and other processes, such as watch_irot (not pictured), pass MUSIC messages downward to the traffic process. Traffic passes the messages to their destination, in the case of a request message, the message would go to the dispatch process responsible for handling the request. Once the dispatch process receives the message, it calls a service routine that parses data in the message and passes it on to a device I/O routine. The device I/O routine accesses local information for the device to provide response message data, if the request was for a keyword value. If the request was for a change to a keyword state, such as a stage position, the device I/O routine sets up the request information in the device's data such that the change request is sent to the device or device controller. Requests to serial devices are sent via a socket connection on a terminal server.

Data flow in the opposite direction starts at the controller level. The dispatch processes reads keyword and other status data changes from the controllers. The data is saved in device data structures in dispatch. The read takes place during the "read status data from controllers" step in the dispatch service loop (see figure 3.13). Status keyword data is returned to requesting processes during the next step in the service loop. If the data indicates that a change service request is complete, the response is set to the requesting task in the send MUSIC response messages step of the service loop. Finally at the end of the service loop, broadcasts of status keyword data changes are sent.

To minimize response time by the dispatch processes, a separate process is used for each device requiring a serial connection.

3.5.2 Loading New Data and Programs onto Controllers

For Galil controllers, new data describing device positions and downloadable program changes can be sent to the controller via the DISPxCFG keyword. Writing this keyword causes the old position and program information to be erased and the new program(s) and data loaded from the file specified by the keyword. A "BP" command must be sent via the DISPxCMD keyword to burn in a new program.

3.6 Deliverable Documents

Both LaTex and html versions of the manuals will be provided.

3.6.1 Programmer's Manual

The Programmer's Manual describes the software components, interfaces, data, and their relationships.

3.6.2 Operations Manuals

The operations manuals describe normal operating procedures and contain information important to the Observers, Night Assistants and Engineers. Each manual is oriented to simplify the different tasks these people encounter when working with DEIMOS.

- Night Assistant's Manual
- Observer's Manual

- Engineer's Manual

Chapter 4

CCD Control

Chapter 5

Flexure Compensation

Chapter 6

Image Display

6.1 Overview

In Part I Section 4.1 we described the two-phase plan for the DEIMOS image display. We focus here on the second phase.

There are long-term advantages to the choice of the NOAO RTD/PIE server.

- The image display will be deployed throughout the IRAF user community. Observers will already be familiar with its basic operation.
- The images will be directly available to IRAF without the need for intervening file writing.
- The message bus/plug-in architecture will permit interaction with tools that support individual observing programs. Any task will be able to connect to the DEIMOS image in real time.
- The field acquisition and mask alignment procedures described in Part II Chapter 2 can be written to use the message bus.
- It will facilitate automated focus and instrument diagnostic procedures.

The plug-in architecture means that each piece of functionality can be developed separately from the others.

Because the NOAO RTD server is at such an early stage of development it is difficult to be specific about the display modules and their mechanisms. We describe the first plug in that will be required for DEIMOS.

6.2 Figures

(none)

6.3 Nomenclature

Figdisp: The image client program at Keck which can incrementally display LRIS and HIRES images in real time as the CCD computer transmits them.

NOAO RTD: Real Time Display. The image display client program under development at NOAO for use with the NOAO Mosaic detector.

PIE: Plug-in Image Extension server. The next generation of astronomical image display being jointly developed by SAO/CfA and NOAO.

Message Bus: The underlying communications mechanism of the NOAO RTD and the PIE.

Distributed Shared Image: A form of data storage managed over the message bus which permits multiple processes on multiple hosts to interact with an image.

6.4 Component Modules

6.4.1 mFigdisp

Function display laboratory CCD data
Language C
Type daemon
Author Allen/Cromer
Essay This is a modified version of Keck `figdisp` which will display mosaic images from the DEIMOS dewar during the laboratory phase of assembly. It will be superseded by a version of the NOAO RTD for delivery.
Inputs pixel data from mlickserv
Outputs image on X server
Deployment Laboratory testing of mosaic
Labor 160 hours
Keywords used by module Reads CCD readout keywords in the image headers

6.4.2 mLickserv

Function receive data from the CCD crate, relay it to mFigdisp, and write a FITS file to disk

Language C

Type daemon

Author Kibrick/Allen

Essay This is a modified version of `lickserv` which will handle mosaic data during laboratory testing of the CCD dewar. It will combine image sections from all CCDs into a single large image for display by mFigdisp. It will write a multi-HDU FITS file containing images and tables.

Inputs interleaved pixel streams from the CCD crate

Outputs combined image in shared memory, FITS file

Deployment Laboratory testing of mosaic

Labor 160 hours

Keywords used by module Collects keywords from the mosaic version of watch_ccd and inserts them into FITS headers.

6.4.3 rtLickserv

Function receive data from the CCD crate and make it available to processes on the message bus

Language C

Type daemon

Author Allen/Kibrick

Essay This is a modified version of `lickserv` which will bridge the gap between existing Keck CCD systems and the PIE server.

Inputs interleaved pixel streams from the CCD crate

Outputs a distributed shared image

Deployment Commissioning

Labor 10 weeks

Keywords used by module Reads DCS and instrument keywords destined for the image headers

6.5 Subsystem Procedures

(none)

6.6 Deliverable Documents

A section of the lickserv manual describing the connection to the NOAO message bus.

Chapter 7

Information Management

7.1 Overview

The scope of the database component of the DEIMOS software complement has been reviewed and revised since the PDR (see part 1, chapter 2, section 5). We currently plan to use the online database for five major instrument support functions:

- Documentation and Design
- Instrument User Interface
- SlitMask Fabrication and Management
- Performance and Failure Analysis (Logging)
- Instrument Calibration / Pipeline Data Reduction

Implicit in our plans is the continued use of the STB image archiving software at Keck, and the continued ingestion of image headers into the online database there for later search, retrieval, and analysis. These procedures are already in place and do not require DEIMOS resources. We have abandoned any plan to implement a special image archiving facility for DEIMOS.

For full engineering support of the instrument, complete telemetry and logging of instrument and telescope/dome control systems is desirable. However, *truly* complete telemetry and logging should be provided by a CARA effort (or by a joint effort between CARA and Lick) for support of all instruments at K1 and K2, not by DEIMOS funds. Some discussion has already taken place between CARA and Lick as to a joint effort to improve logging and image archiving. This effort might involve new keywords for CARA systems, and is outside the scope of the DEIMOS project.

Absent the sudden realization of this joint effort, DEIMOS database logging and analysis will be limited to information readily available via current CARA KTL facilities, and its purpose and scope will be limited to instrument engineering and science support. No effort will be made to provide Web-based public access; the focus will be on tools and utilities for CARA and Lick engineers and DEIMOS observers only.

In reviewing our progress to date, it is clear that over 90% of the schema design for DEIMOS is already complete, and that the suite of documentation generators is at least 80% complete. The keyword dictionary is over 80% complete. A useful collection of Agents and high level system designs are already expressed in database form. Most remaining incomplete or uncommenced modules are relatively simple extract generators, with the exception of EngDataVisu (q.v.). and CodeGen.

In other words, a large percentage (definitely more than 75% overall) of the effort of DEIMOS database development has already been realized in the course of preparation for the CDR. The remaining problems are not schema problems, but extract problems; and only two of those are likely to be nontrivial. We feel that this is good news. We are also pleased with the benefits realized so far from the "authoritative database" approach, particularly in the generation of figures and reference sections for this CDR report and the LOTR. In sum, we find our results so far quite encouraging, and do not see grounds for concern that the database component *per se* will delay the project schedule or consume more than its share of resources. So far it appears to be ahead of other major instrument subsystems, and to be cost-effective.

Having limited and defined the scope of the database component to five areas, we now present an overview of each one.

7.1.1 Documentation and Design

At the core of the DEIMOS information management system is a database schema representing (among other things) KTL/FITS keywords. This schema represents not only the simple attributes of keywords (such as datatype, read/write-ability, and format) but complex relationships between keywords (such as that NAXIS controls the instantiation of CRVALn), and conditions of identity and variance among keywords with the same names. In addition to describing keywords as they appear in FITS headers, the schema also represents the mapping of internal instrument control system values into external representations (using simple lookup table maps or more complicated mathematical transforms).

From the tables comprising this schema, various forms of documentation can be generated. We can produce HTML or LaTeX formal documentation for keywords (a "dictionary" of DEIMOS keywords generated in this fashion is found in the Lick Observatory Technical Report (LOTR) accompanying the CDR document). It is not difficult to add other markup languages such as SGML or MIF (Maker Interchange Format) as long as they support constructs similar to those in LaTeX and HTML.

We can produce fake FITS headers (to test data reduction or other image-handling software). Some additional documentation products will be discussed below. We can also generate certain repetitive blocks of low-level source code for the instrument control system. Related tools can convert "bundles" of keywords into SQL code that creates appropriate database table definitions for the storage of those keywords (thus, a FITS header definition can be transformed automatically into a database table).

Not only FITS/KTL keywords (and hierarchical collections of those keywords) can be documented in this fashion. All the database tables required for the DEIMOS project are also defined in the "keyword" schema. Since the schema has taken on a more general function than the description of literal keywords, it's no longer called the Keywords database, but has acquired the nickname "Memes" (a database of units of meaning). The database table definitions in the Memes database can be documented formally using the same utilities that produce keyword documents, and we have tools to convert an existing Sybase table automatically into a bundle of Memes requiring only minimal further data entry for full documentation. Features such as primary and foreign keying are easily represented in the schema and expressed in the documentation, and SQL code for table creation is (as mentioned above) easily generated from the schema. See Figure 7.8 for a high-level picture of the Memes tables.

An additional schema representing DEIMOS system components (software, hardware, and human "agents") permits us to describe the migration of memes (information) through the instrument, or through the observing (or slitmask fabrication, or calibration...) process. Diagrams showing the collection, generation, modification, and movement of information by and between agents document the software/hardware design of the instrument and the observing process. This additional schema, the "MFlow" schema (when fully populated), should describe and document, at both a high level of generality and a low level of minute detail, the intent and design of our finished products.

The MFlow schema is a cluster of tables documenting Agents (software or otherwise) and the Formats and Media in which information travels among them. A table of Timings expresses the scheduling of information transit (On Demand, Cron Job, End of Exposure, etc.). A table of Mpaths (Meme Paths) describes the passage of information (Memes) between agents. Several figures in this chapter are generated from the data stored in this schema. See Figure 7.9 for a high-level picture of the MFlow tables.

A relatively simple sketching tool ("Etcha") is used to draw initial designs, which can then be committed to the database, recalled, revised, and so forth. A report generator (`tracepaths`) is used to produce "dot language" files for input to a digraph generation tool (`dot`), which produces camera-ready PostScript output. See Figure 7.4 for a typical Etcha session.

More detailed information about Etcha and related tools will be found in Part I, Chapter 3 (Proofs of Concept) Section 3 (System Design Tools). Exhaustive definitions of the Memes and MFlow schemata will be found in the accompanying Lick Observatory Technical Report.

In addition to such abstract entities as keywords, software modules, etc., the instrument also possesses simple, physical components like filters, gratings, and detectors. See Figures 7.10 and 7.11 for a high-level picture of the tables describing physical components of the instrument. These provide a portion of the "as-built" documentation describing DEIMOS in its finished (commissioned) state.

The entire "knowledge base" about DEIMOS that resides in the online database, most of which will be accessible to the DEIMOS user at all times during observing [1]. From overview to fine detail, we plan to make the complete information set available for the benefit of observers, OAs, and Lick/CARA engineers onsite. For example, one optional attribute of some KTL keywords is a signal ID (on the electronics drawings) and the test point or pinout at which the signal can be tested. In the online DEIMOS documentation system, this information can be looked up rapidly and interactively by keyword name. It also appears (where available) in the LOTR.

7.1.2 User Interface

The GUI Builder ("Dashboard") merits mention in this section because of its ability to use the Memes database to configure its behaviour. It can be

[1] Certain types of data will have to be protected, to preserve the privacy of observers' projects and observing plans. Observers will have access to all public data, and all private data pertaining to their own research.

made to cache the retrieved information in ASCII files, and can thereafter be started and run with no further online database access; however, its design is based on the notion of an available, authoritative database of keyword specifications. More detailed information about Dashboard and related issues will be found in Part I, Chapter 3 (Proofs of Concept) Section 2 (GUI Builder).

7.1.3 Slitmask Fabrication and Management

The fabrication and use of DEIMOS slitmasks requires a lot of record-keeping (like any industrial fabrication process) to ensure quality control and appropriate deployment of finished masks. There will be hundreds of DEIMOS slitmasks, with operational lifetimes of anywhere from part of one night to several weeks or months, all of which must be properly identifiable (by electronic means) throughout their existence. When a mask is damaged or destroyed, the termination of its existence should also be recorded for posterity.

We plan to use the database to store the object catalogs and slitmask blueprints of successfully machined masks. See Chapter 1 (Slit Masks) for a complete description of the mask fabrication process including database interactions.

Masks will be identified by barcode scanning when loaded in the cassette. The complete description of each identified mask, down to the object catalog and milling instructions, can be retrieved at any time. More obviously useful information – such as the observer's nickname for the mask, date of manufacture, and so forth – should help to prevent any confusion, misloading or misidentification of masks (such as has already cost several hours of LRIS time).

See Figure 7.12 and 7.13 for a high-level picture of the tables which describe slitmasks.

7.1.4 Performance and Failure Analysis

More detail on the use of database-resident parametric logging as an engineering resource will be found in Chapter 8, Section 3. If detailed parametric logs are available online in database form, analysis and visualization can be made relatively easy. Arbitrary queries across a large dataset are more easily achieved using SQL, or preferably a GUI SQL generator, than by the laborious construction of individual C code modules, FORTRAN programs, awk/sed scripts, or other "from scratch" approaches.

We propose to perform as complete parametric (environmental) logging as can be achieved at low cost and without undue impact on CARA systems and staff. Our belief is that both software and hardware instrument support will be improved and made easier thereby.

7.1.5 Instrument Calibration, Pipeline Reduction, FCS

DEIMOS is a complex instrument, and analysis of DEIMOS images will be a complex and time-consuming task if performed by traditional, manual means. We would like to provide some degree of automated or semi-automated (pipeline) data reduction support, or at the very least a useful online library of calibration information for use in manual reduction.

For pipeline data reduction, the database is useful as a quick method of retrieval for appropriate comparison images and spectra [2], and for the storage of authoritative information about DEIMOS lamps, gratings, filters, etc. A body of "standard calibration data" is required for pipeline reduction to work, and the retrieval of appropriate data will be faster and easier using a database. Obviously it would be impractical to acquire the mass of calibration data by any manual process. It should be ingested automatically as the final step in a successful recalibration procedure (the proposed stable calibration database and pipeline reduction would require recalibration to be performed on a regular schedule).

We have proposed to store, in addition, more abstract entities like coefficients for interpolation and correction algorithms. (see the Lick Observatory Technical Report number ?? documenting the database schema and DEIMOS keywords). See Part III for details of the calibration and pipeline data reduction proposal, associated procedures, and the role of the database in facilitating calibration and pipeline reduction.

Instrument stability is an unknown quantity at the time of writing, as are many details of flexure correction and calibration requirements. All schema and procedures related to these topics should be regarded as parts of an experimental model, almost certain to change significantly before commissioning. The data reduction pipeline, though useful, is not absolutely necessary at commissioning, and will not be delivered if time and budget constraints bring it into conflict with other more essential deliverables.

[2] Of course, we do not plan to store actual binary image data in the database, only header information and the URL of the actual image, in this case a FITS image file on local disk.

7.1.6 Summary

Having defined the role of the online database with greater precision than we could at the PDR, we are able to document existing (production) schema currently in use (Memes, MFlow), well-developed schema soon to be implemented for instrument assembly and testing (Optical components, Slitmasks), and speculative schema (FCS, Calibration) for processes and techniques not yet fully understood. We can also define fairly accurately the list of software modules needed to make use of the information stored in these schemata, and their functions (again, some are already in production and others are only in design or early prototype).

The remainder of this chapter consists of Figures, a Glossary, Software Module descriptions, Procedure Descriptions, a list of major deliverable documentation products, and Schema Documentation.

7.2 Figures

7.2.1 Helpful Keys to Flow Diagrams

In reading the information flow diagrams in this and other chapters you may find the following translations helpful.

7.2.1.1 Media Abbreviations

DF : Disk File

EM : E-Mail

FD : Floppy Diskette

HTG: HTTP Get

HTP: HTTP Post

KE : User Key Entry

MP : Manual Portage

MT : MagTape

PAP: Paper Output

RD : Raw Disk

RS : Serial Connection

SD : Screen Display

SM : SlitMask

SQL: Database Query

TCP: TCP Connection

UDP: UDP Packets

X11: X11 Display Server

7.2.1.2 Format Abbreviations

BAR: Barcode

BD : binary data

C : C source

DB : database table

DX : DXF Mill File

FF : FITS file

FI : FITS Image

FOR: FORTRAN source

FT : FITS table

HT : HTML source

IM : raw image data

KV : keyword/value data

LT : LaTeX

MIL: Milled Surface

NC : NC Mill G/M codes

PS : PostScript

SC : source code

SQI: SQL Insert

SQL: SQL source

SQS: SQL Select

SQU: SQL Update

SQW: SQL Write

TX : ASCII plain text

Tcl: Tcl source

dot: dot source

7.2.1.3 Timing Abbreviations

CJ : Cron Job

EE : End of Exposure

FI : Fixed Interval

OD : On Demand

SD : Shutdown

SE : Start of Exposure

SU : Startup

7.2.1.4 Agent Types

C: Command

D: Daemon

H: Hardware

P: Person

T: User Tool

7.2.1.5 Symbols used for Agent Types

box: hardware

circle: Unix Daemon

diamond: command line single command

hexagon: human being

house: interactive user tool

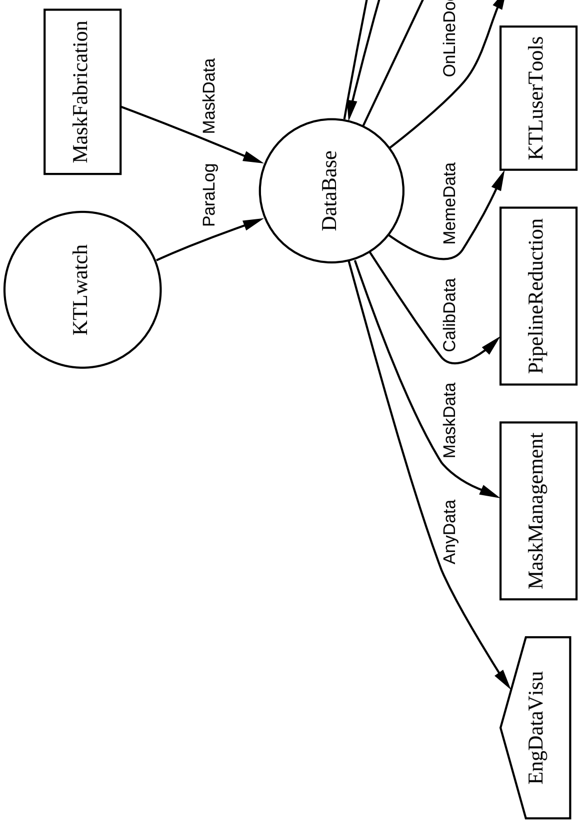


Figure 7.1: Operational Role of the Database

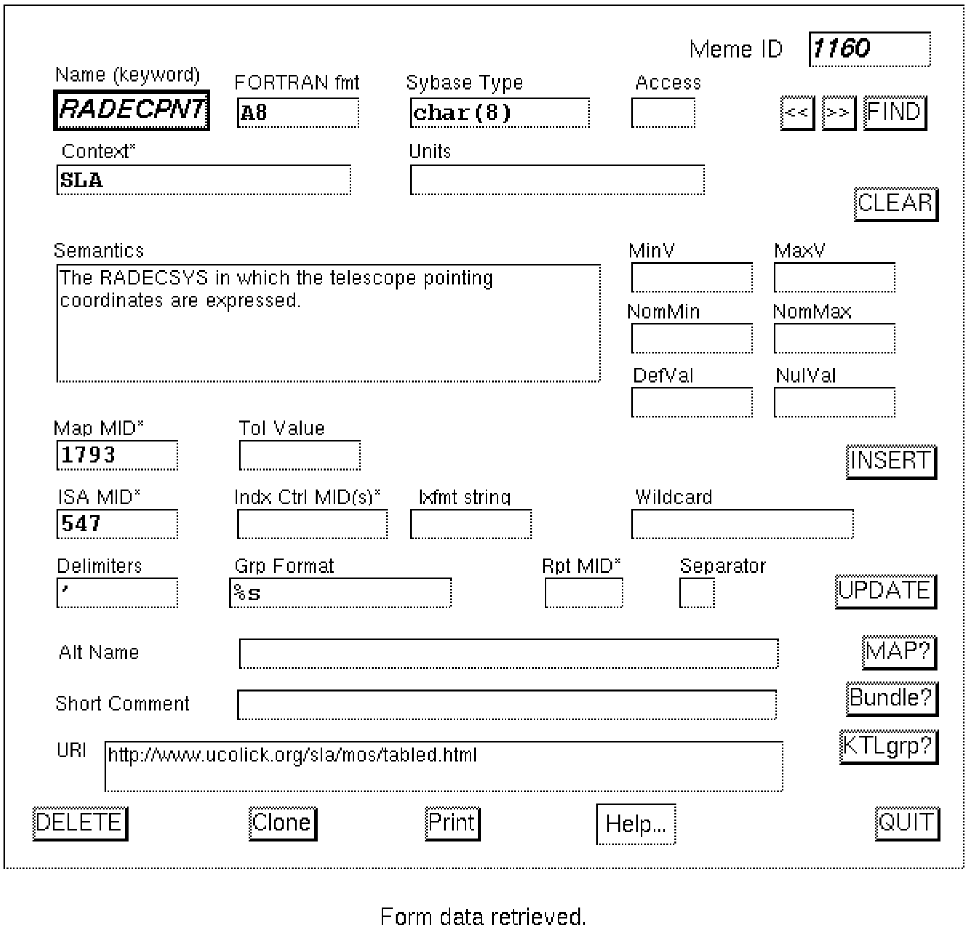


Figure 7.2: Typical Data Entry Form: Memes

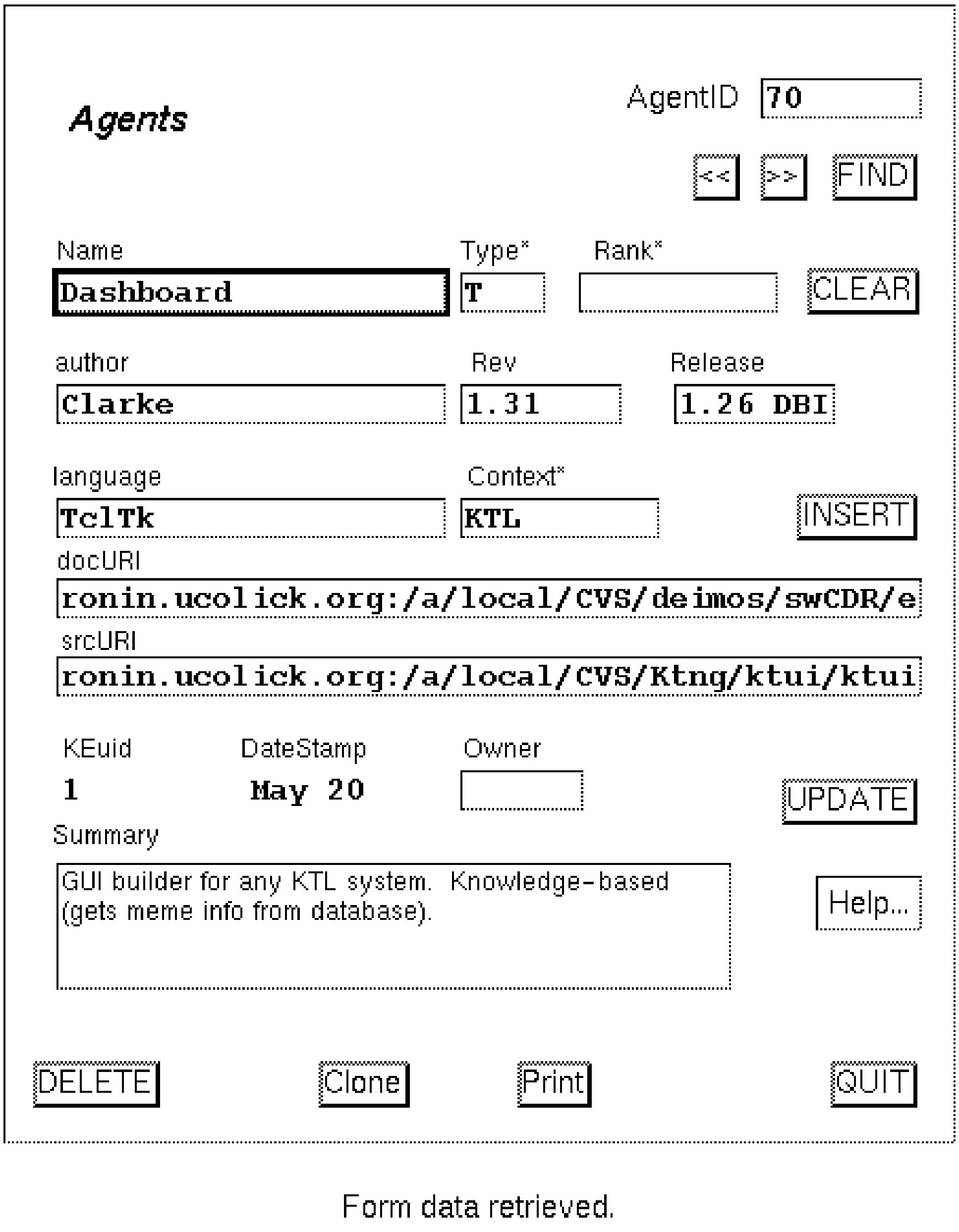


Figure 7.3: Typical Data Entry Form: Agents

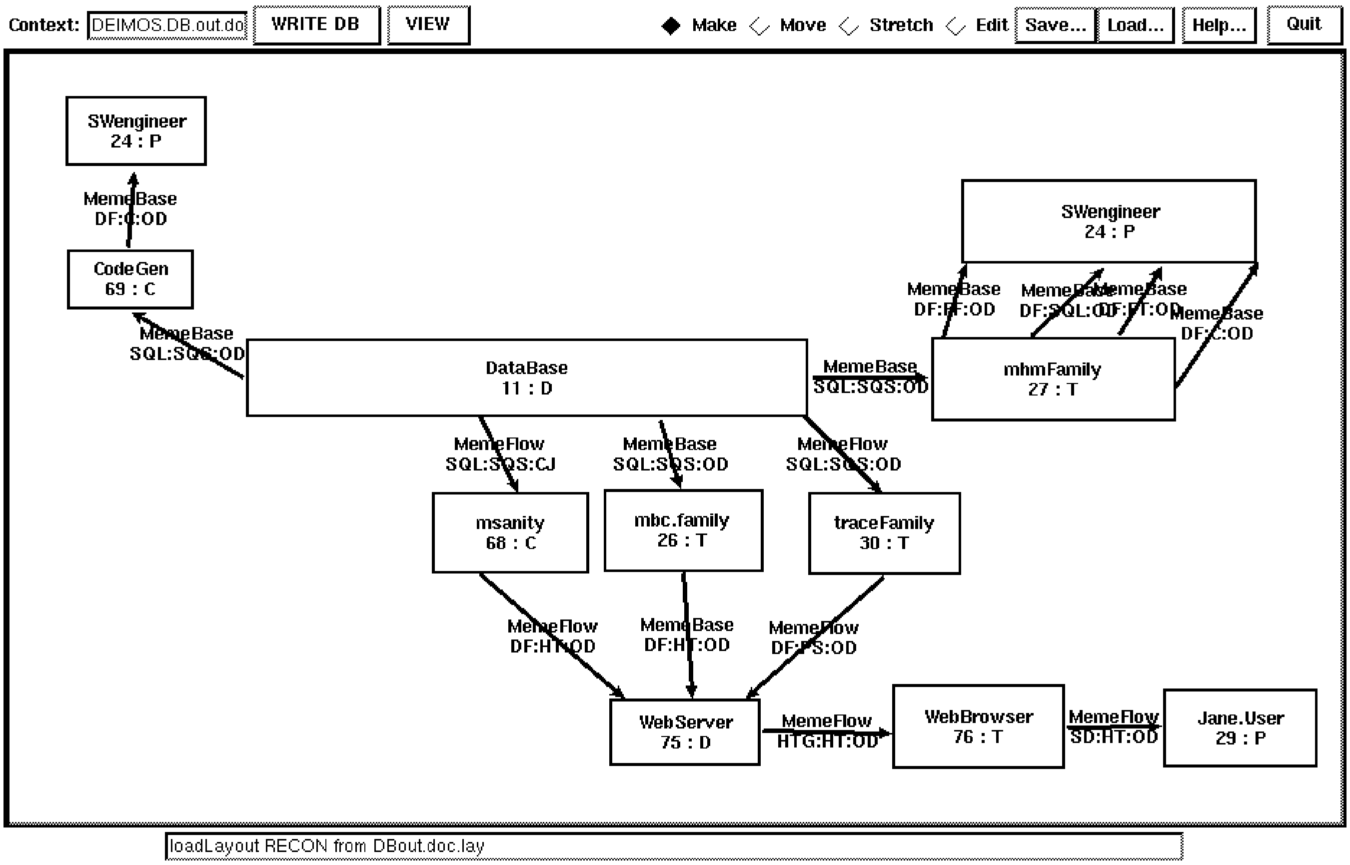


Figure 7.4: Typical Etcha Screen: DEIMOS.DB.out.doc

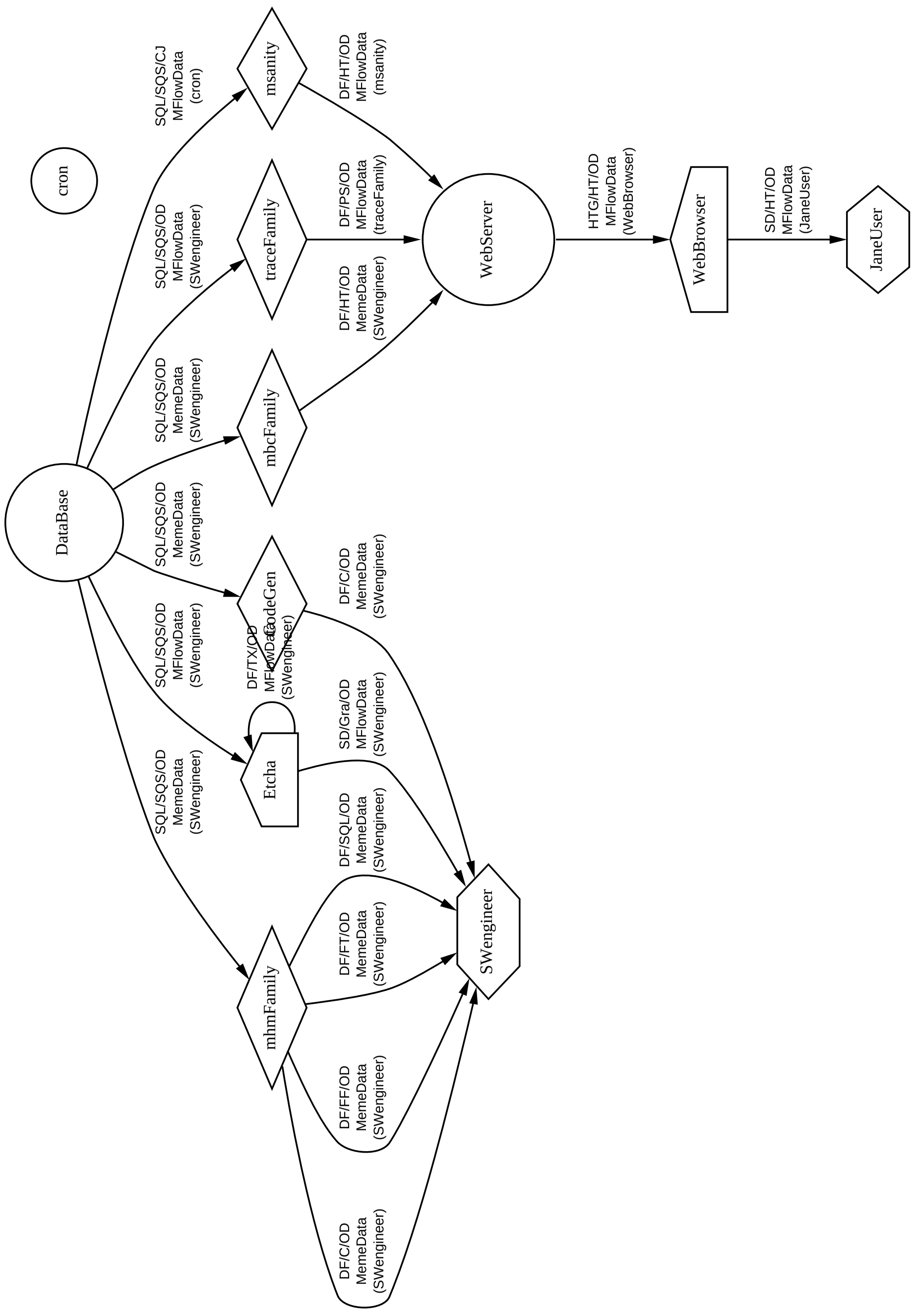


Figure 7.5: Finished Drawing: Database Documentary Outputs

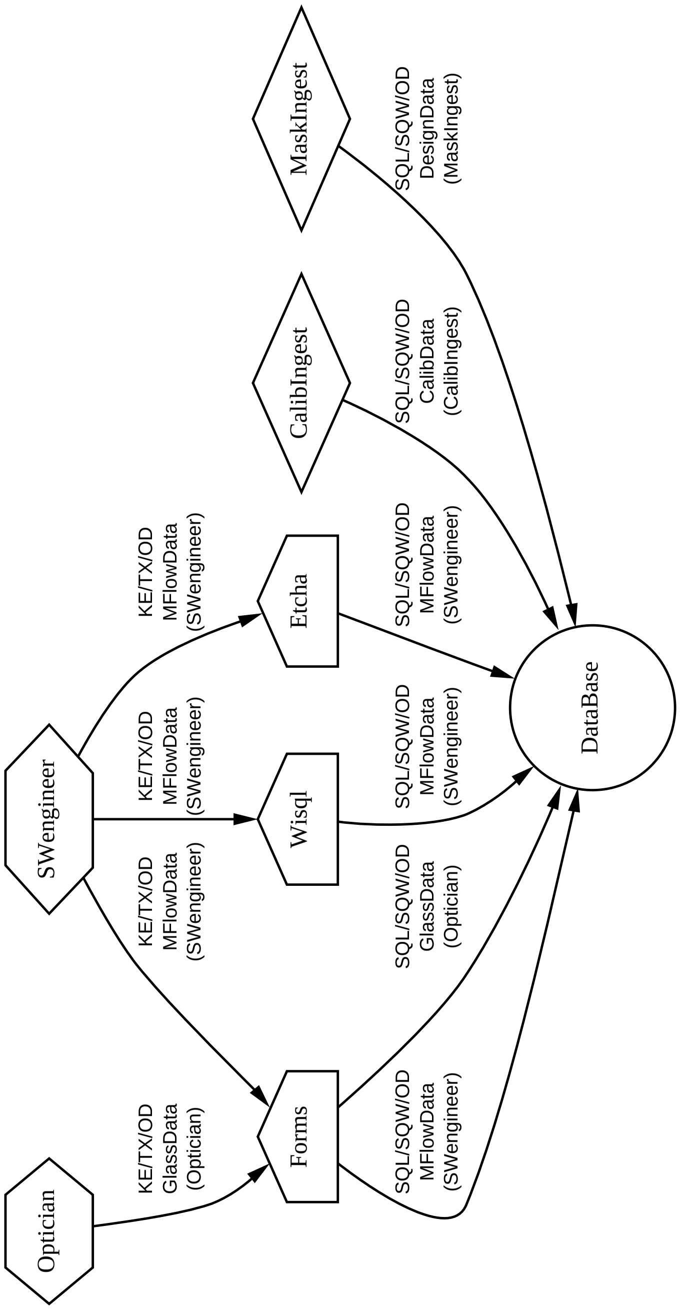


Figure 7.6: Finished Drawing: Database Information Inputs

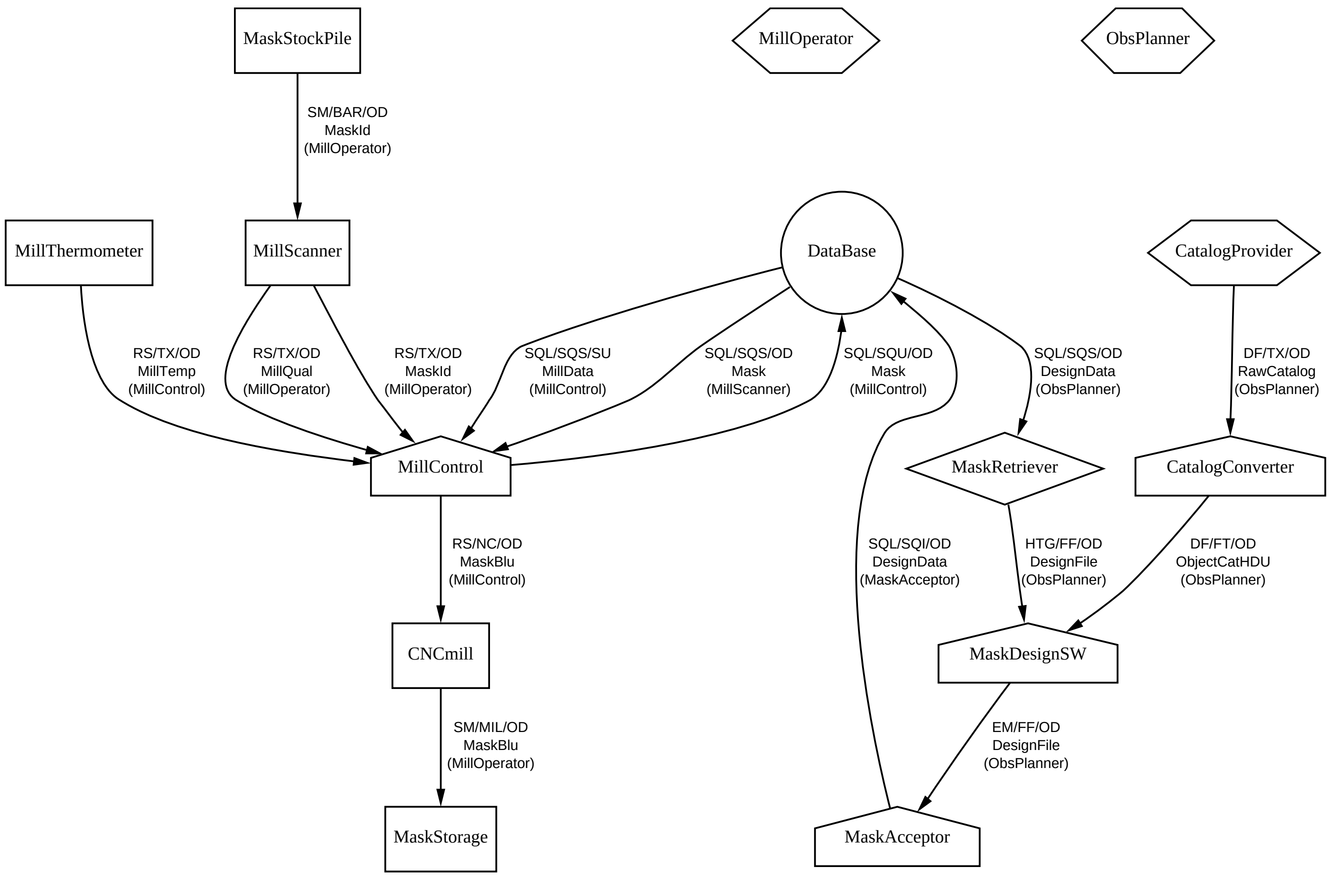


Figure 7.7: Finished Drawing: Slit Mask Production

7.3 Database Nomenclature

Access control: all RDBMS provide some mechanism for internal access control; each connecting process has a "user id" and not all user id's are able to see all tables or all fields of a given table. Privileges are similar to file protection bits in a file system but more elaborate: read, write, update, delete are the basic set, and there are many more in some RDBMS.

Agent: DEIMOS project jargon, derived from object oriented programming terminology: a software or hardware component, or a human, participating in processes related to the DEIMOS instrument and its data reduction

Backend: another name for the server process or "back end" of the database, the front end being the user interface

Cartesian Cross-Product: Ill-formed queries can result in the generation of enormous return data sets which are the cartesian cross product of two or more large tables. This type of query can exhaust the resources of the server.

Database: Literally, any collection of data; in this context, a related collection of tables of data

Digraph: "directed graph", a layout of nodes and edges with directionality along the edges; can be used to diagram data flow, function, *EER*, and so forth

EER diagram: Entity-Entity Relationship Diagram, a specialized schematic diagram used by relational DB designers to document a schema

Entity: In relational algebra, a meaningful unit of information in the real world which should be modelled as a table; something which has attributes

FSE: Field Selection Expression, that part of a SQL query which determines which fields will be returned

Field: A column of a table of data, also called an Attribute

Foreign key: A field in a table which links it to another related table (Customer ID might link a customer with the customer's invoice). More accurately, if field X is the primary key of table A and also appears in table B, it is a foreign key in table B.

ISA: One of several possible relationships between entities, as recognized by relational algebra. In the Memes database we use this term ISA to indicate a relationship where X "is a" Y, meaning that X and Y are fundamentally the same thing (same meaning) but expressed differently. The difference could be format or units, but the underlying meaning is the same. For example, the same date can be expressed in several formats, or the same stage position can be expressed in several units.

Index: To facilitate rapid retrieval of data, all RDBMS provide some facility for building indices on tables. Different indices can be built for each field, or an index can be built on some combination of fields. The sort algorithms used for the index will almost always be variants on B-tree and heap sort.

Ingres: Another major commercial RDBMS

Meme: DEIMOS project jargon, derived from "sememe": a unit of meaning, e.g. a FITS keyword

One-to-many: A possible relation between tables in a schema, in which one entity can be associated with multiple instances of another entity (one Customer, for example, can have many Invoices); this is sometimes called a parent-child relationship, by analogy with tree structures.

Oracle: A major commercial RDBMS

Postgres95: A major non-commercial RDBMS, descendant of Ingres

Primary key: Every table should have one field (attribute) which is unique to each record, or at least some combination of fields which taken together are unique to each record. There is no other way to distinguish a single record reliably. This field or combination is called the primary key.

Query: A SQL statement resulting in the return of data from an RDBMS

RDBMS: Relational Database Management System, a software package which manages a relational database, optimized for rapid and flexible retrieval of data; also called a *database engine*

RSE: Record Selection Expression, that part of a SQL query which determines which records will be returned

Record: A row of a table, also called an Instance of a Class

Relational: Describes the design of tables of data such that key fields within each table relate it to other tables

SQL: Structured Query Language, an ANSI standard language designed for manipulation of relational databases; some superset of ANSI SQL is built into every RDBMS

Schema: The design for a relational database, including the complete definition of tables and the relationships between them

Server: A host on which an RDBMS capable of accepting IP connections is running; a host which can answer SQL queries across a network. Also the *server process* which is running on that host.

Sybase: Another major commercial RDBMS

Table: A collection of information which has been organized into multiple rows, each row consisting of a fixed number of columns, each column being consistent in semantics and type across the entire table; also an Entity or Class

Tcl: "Tool Control Language," a flexible interpreter designed primarily by John Ousterhout at UC Berkeley. Extensible and powerful, it is a viable competitor of the perl language. Tk is the X11 toolkit extension to Tcl.

WWW: World Wide Web, common usage for the "web" of sites running HTTP (hyper text transfer protocol) servers offering HTML (hyper text markup language) documents across IP connections.

app: short for "application", an application program

business rules: Much of database design tradition stems from large financial applications. The "business rules" of a database are those dependencies and interactions not expressible in simple relational algebra. As, for example, the fact that a Meme which appears as a Ctrl_MID for other Memes must absolutely be an integer, or that Software Agents have to have a CVS reference, or that a field can only take on one of three known values. It is theoretically possible to break up a schema so that business rules can be reduced to relational algebra, but in most cases this would fragment the schema to the point of absurdity.

defined term: verbose definition of term. As many lines as is necessary.

file-system, filesystem: the structure of files on a disk medium which is visible via the operating system, i.e. the structure of files which a Unix user can see using "ls" and other tools

freeware: common parlance for software which is made available free of charge by its author(s)

inload: database managers' term for the process of loading data from "flat files" on disk into an online database

7.4 Component Modules

7.4.1 CodeGen

Function Tool to generate some portion of KTL realtime instrument control code from Memes database
Language Tcl
Type Command
Author Clarke/Allen
Revision
Source CVS Ref
Essay This module exists only in early prototype at CDR time, in the form of features of `mhm` and `mbc` that produce C data structure source and some simple FITSIO function calls. Our focus has been mostly on keyword definitions, documentation, and UI development. However, KTL controller code is very formulaic and we believe that from an adequate keyword database we can generate certain sections of repetitive boilerplate source code which otherwise would require tedious manual coding. These formulaic source modules are often a locus of cut/paste error. If we can generate them successfully, we may reduce error and improve productivity for the realtime component of the DEIMOS code.
Inputs MemeData
Outputs C source code
Deployment next on the list! Sep 97
Labor 80 hours

7.4.2 Dashboard

Function GUI builder for any KTL system. Knowledge-based (gets meme info from database).
Language TclTk
Type User Tool
Author Clarke
Revision devel 1.31 release 1.26 DBII-1-4a
Source CVS Ref ronin.ucolick.org:/a/local/CVS/Ktng/ktui/ktui
Essay The "Dashboard" agent is an interactive user tool. It derives information about keywords from the Memes database, then infers from

that information how to construct and use various graphical widgets to display readable and writable keyword values. The GUI builder is described more fully in Part 1, chapter 3 (Proofs of Concept), section 2 (GUI builder), where you will find some prototype DEIMOS UI layouts.

"Dashboard" is a general purpose interface/builder for KTL instruments, not intended solely for use with DEIMOS.

Inputs keyword data

Outputs user interface for KTL

Deployment complete: Release 1 May 1997

Labor 2 months

Inputs KTL keyword information from database; layout file; user interaction

Outputs KTL writes; log files; postscript plots; screen display

Flow Diagram(s) See Figure **??**.

Keywords Used Any and all KTL keywords.

7.4.3 Etcha

Function Sketching tool for creating meme flow designs; can write into the MemesWorld schema.

Language TclTk

Type User Tool

Author Clarke

Revision devel 1.5 release 1.5 ETCHA-1-2-2

Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/etcha/etcha

Essay "Etcha" is an interactive user tool. The user can create nodes and connect them with directional edges, assigning attributes to both nodes and edges interactively. The nodes correspond to Agents as represented in the database, and the edges correspond to Paths. The user can recall information about existing Agents and Paths to fill in the attributes of newly created nodes or edges.

Once the user has made a sketch, however simple or complex, in which all nodes are legitimate Agents and edges are legitimate Paths, a single button-press transfers the design to the database tables. The transfer replaces existing data, and can be repeated whenever desired. Etcha can also extract a complete path "context" from the database as the foundation for a new sketch. [3]

For convenience, another single button press will run the "tracepaths" tool (q.v.) and preview the PostScript output using Ghostview. In this way the user can review the camera-ready version of the chart and iteratively alter the design without leaving Etcha.

Inputs MemeData from DB; user interaction

Outputs datafile describing information flow; SQL insert into Mpaths table; SQL insert into Agents table; postscript plots; screen display

Deployment complete: Release 1 April 1997

Labor 1 week

Flow Diagram(s) See Figure **??**.

7.4.4 Forms

Function Interactive Sybase forms GUI

Language TclTk

Type User Tool

Author Clarke

Revision devel 1.1 release 1.1

Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/fosql/forms

Essay "Forms" is an interactive user tool. It provides an X11 GUI for database query, update, and insert using the visual metaphor of a paper form. Each field of a table is represented as a blank on the form. Blanks are labelled appropriately for their field name or meaning.

The "Forms" application has been in service at Lick Observatory for over 4 years. It was initially developed in support of a financial/administrative online database; however, it is entirely general-purpose and partially inferential (like Dashboard) so that it can be used with any Sybase schema.

Inputs database schema and data

Outputs GUI; user browses/maintains data

[3] The attribute "context" is available for each major entity in the DEIMOS database, and it means whatever the software engineers want it to mean; it is used merely to "label" or associate entities for practical purposes. During prototyping and schema development, this generic attribute provides a certain flexibility in coding, and latitude for invention and improvisation. As the schema and applications settle into their final form, the "context" attribute usually turns out to mean something more concrete. In the case of Paths, this attribute became useful for grouping Paths into "drawings" of related agents and their function. Each Path context now has an author, title, and so forth. When the great nomenclature cleanup arrives, this attribute of Paths will probably be renamed "drawing".

Deployment Complete early 1993
Labor 4 months
Flow Diagram(s) Appears in Figure 7.6

7.4.5 KTLwatch

Function Incorporates dashboard condition handling code; configures self from database of alarms and responses. Watches selected KTL keywords and takes action if conditions are met.
Language Tcl
Type Daemon
Author Clarke
Revision
Source CVS Ref
Essay See Chapter 8 for a discussion of the "watcher" daemon. In brief, this module would configure itself from the database, using both Memes information (for KTL keyword definitions) and Alarm/Condition information from database tables. Having autoconfigured, it would run indefinitely watching selected KTL keywords and taking action on the fulfilment of any of its list of conditions. The action could be delivery of warnings or alarms, or any other arbitrary Unix command(s).
Inputs readable KTL keyword values
Outputs records in ParaLog table

7.4.6 mbcFamily

Function mbc is also mbn/mbh/mbb/sbh. Produces LaTeX or HTML documentation of memes by context, name, header, bundle. Also generates C struct source.
Language Tcl
Type Command
Author Clarke
Revision devel 1.18 release 1.18 Memes-1-0
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/mbc
Essay The "mbc" family is actually one command-line utility to which various symlinks are made so as to use argv0 as a function control. The variants are `mbn, mbh, mbb, sbb`. The names stand for Memes By (Context, Name, Header, Bundle) and Storage By (Header).
The function control exercised by argv0 is in most cases to choose the

SQL select "where" clause when grabbing the memes data. In the case of `sbb` it also determines the output form.

The output of the "mbc" family is either LaTeX or HTML. A fixed format is used to document one meme after another until all the target memes have been documented. The only variant which does not work like this is `sbh`, which produces C source code defining appropriate data elements for the storage of a FITS header as defined in the Memes schema.

The "mbc" family exists only to produce documentation (or source code). They can be run rarely to produce paper or static online documents, or invoked interactively from a Web page to produce truly up to date online documents.

Inputs Data from Memes schema
Outputs HTML, LaTeX, source code, other markup languages possible
Deployment in production April 1997
Inputs data from MemeData
Outputs HTML, LaTeX, other markup languages, C source code
Labor 1 month
Flow Diagram(s) See Figure 7.5.

7.4.7 memestruct

Function generates dot-language and PS maps of the structure of a meme. Used to document complex (mapped) memes.
Language Tcl
Type Command
Author Clarke
Revision devel 1.7 release 1.7 Memes-1-0
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/memestruct
Essay `memestruct` is a close relative of the traceX family of modules. It draws a digraph representation of the internal structure of a complex meme. A complex meme is one which has ISA and mapping relationships to other memes or is a tuple value.

`memestruct` will follow ISA and hierarchy relations and draw a visual representation of the meme. For example, the meme DASECn has a complex internal structure which memestruct will reveal.

`memestruct` is not intended for end-users, but is a tool the software engineers can use to verify complicated meme definitions.

Inputs information from MemeData schema

Outputs PostScript plot
Deployment production May 1 1997
Labor 1 day

7.4.8 mhmFamily

Function `mhm` is also `mtm/stf/stff`. produces FITS headers and Sybase table defs from keyword bundles. also produces FITS extensions from sybase tables.
Language Tcl
Type Command
Author Clarke/Allen
Revision devel 1.25 release 1.25 Memes-1-0
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/mhm
Essay The "mhm" family of command line utilities is really one source called `mhm`, with symlinks to other names (`mtm, mdm, mcm, stf, stff`) so that argv0 can be used to control its function.

The "mhm" family makes structured representations of bundles (mostly tables) of memes. `mhm` makes a fake FITS header, generating a random value within the legitimate range and in the correct format for each meme. `mtm` makes a sybase table definition (simple or complex, as required) for a bundle. `mcm` (in development) makes KTL source code in C. `stf` and `stff` are related, but different: they produce a FITS table extension file from a Sybase table whose definition is found in the Memes database. `stff` produces a newline-terminated version, whereas `stf` produces the "real" 80-column card image version with no newlines.

`mtm` and `stf(f)` between them constitute a mechanism for transporting data between Sybase table representation and FITS table extension representation. Thus any of the data kept online in the database can be exported as a standard FITS table extension, for use by non-database-capable applications at remote sites.

Inputs information from MemeData
Outputs FITS header, SQL table definition, FITS table extension, C source code
Deployment production as of April 1997, code generation only in prototype
Labor 120 hours
Flow Diagram(s) See Figure 7.5.

7.4.9 msanity

Function sanity checker to detect rule violations in Memes/Agents world
Language Tcl
Type Command
Author Clarke
Revision devel 1.4 release 1.4 Memes-1-0
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/msanity
Essay The Memes/Agents schema is complex. Rather than implementing business rules in Sybase triggers (vendor specific and time consuming to code and test) we opted for portability and external rule checking. The `msanity` command line utility checks the schema for compliance with our design intentions, flagging incorrect and incomplete data entry.

It generates an HTML report. The intention is for `msanity` to run as a cron job, generating a known URL which can be checked at any time by developers. It can also be run manually, however, for instant feedback.
Inputs MemeData and MFlowData
Outputs HTML file
Deployment production in early May 1997
Labor 1 week
Flow Diagram(s) See Figure 7.5.
Inputs MemeData, MFlowData
Outputs log file calling out inconsistencies and errors in data entry

7.4.10 PipeConfig

Function Tool that generates the pipeline config files based on the date and type of the target image.
Language Tcl
Type User Tool
Author Allen/Clarke
Revision
Source CVS Ref
Essay The PipeConfig agent (not yet written) will accept arguments (or a configuration file) derived from a DEIMOS image header. Using primarily the OBSDATE of the header, but also other information about the image type, it will retrieve from the calibration database a set of appropriate values – such as parameters, the location of flat field and

calibration spectra, etc. Having retrieved this information it will write a configuration file for the pipeline. Its end product is this configuration file.

Inputs image header, calibration database
Outputs configuration file
Deployment Engineering Tests
Labor 1 week (not counting schema development for calibration)
Flow Diagram(s) See Figure **??**
Inputs CalibData
Outputs Pipeline configuration file (FITS)

7.4.11 stm

Function a close relative of mbc, mhm: creates MemeData from existing Sybase tables
Language Tcl
Type Command
Author Clarke
Revision release 1
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/stm
Essay When given a fully qualified Sybase table name and server name as its targets, `stm` reads metadata from the Sybase server to characterize the table. It then translates that information into SQL inserts into MemeData tables, documenting the Sybase table in the MemeData schema. It leaves certain fields blank (semantics, for example) for the user to fill in later.
Inputs Sybase table
Outputs MemeData
Deployment production April 1997
Labor 1 day

7.4.12 traceFamily

Function also traceagent/tracepath. produces dot-language and PostScript maps of Mpaths selected by meme, by agent, or by path context.
Language Tcl
Type Command
Author Clarke

Revision devel 1.9 release 1.9 Memes-1-0
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/memes/apps/tracememe
Essay The "trace" family of command line utilities are really one source code `tracememe`, to which symlinks are made so as to use argv0 for function control. `traceagent` and `tracepaths` are links to `tracememe`. Each of these modes produces as its end result a PostScript file. Data from the Memes, Agents, and Paths tables (and auxiliary tables in the schema) are used to construct a "dot-language" file. This file is a simple ASCII representation of a digraph (directed graph) consisting of direction edges connecting nodes. The nodes are agents, the edges represent the movement of information between agents. The PostScript file is generated from the dot-language file by an exec of the `dot` command line utility.
`tracememe` traces all the relationships of a particular meme (keyword): which agents handle it and how. `traceagent` traces all the relationships of one agent, and `tracepaths` traces all the paths in a particular context ("drawing") or group of contexts.
Inputs MemeData, MFlowData
Outputs Postscript plot
Deployment production April 1997
Labor 1 month
Flow Diagram(s) See Figure 7.5.

7.4.13 Wisql

Function Interactive RDBMS GUI for advanced users. Versions available for Oracle, Postgres, Sybase.
Language TclTk
Type User Tool
Author Clarke
Revision devel 4.1 release NoRel NoTag
Source CVS Ref ronin.ucolick.org:/a/local/CVS/tcldb/wisql/wisql
Essay "Wisql" is an interactive user tool. It provides a general-purpose interface to SQL RDBMS servers. Wisql is partly the work of Tom Poindexter (1992 version). It was reorganized and many features were added by De Clarke at Lick Observatory. The Lick version is in use at several Sybase sites internationally.
Wisql permits the direct entry, editing, and execution of SQL language statements of arbitrary complexity. It displays results and error mes-

sages, and permits saving and reloading of queries. The Lick version includes an "EZ" user mode in which GUI conventions are used to select tables and construct queries, as well as a plotting facility, detachable result and query windows, context-sensitive help, etc.

Wisql is intended for use by database administrators, software engineers, and others whose needs are not met by the Forms tool.

Inputs database schema and data

Outputs GUI; user browses, reports, maintains data

Deployment Complete early 1993; still maintained and developed

Labor 3 months

Flow Diagram(s) See Figure 7.6.

7.5 Subsystem Procedures

7.5.1 Software Design/Documentation

The heart of a KTL-based instrument is keywords, so the central fact of software design for such an instrument is inventing keywords and deciding (collectively) on their nomenclature, meaning, and usage. These decisions are formalized by data entry in the Memes schema. The software engineers enter data directly into tables using the Forms interface (see Figure 7.2) to create and correct information about keywords. They use utilities like `mbc` and `mhm` (directly or via a Web page) to check their data entry, or inspect the results of `msanity` online using a Web browser. The feedback loop is rapid; corrections and changes are easily made.

When the keyword database is correct, formal documentation is generated by means of Makefiles which produce LaTeX (stage 1) then produce HTML from the LaTeX source.

Software engineers use Etcha to sketch functional diagrams and to create new Agents (they can also use Forms to create Agents). They subsequently commit those diagrams to the database, then view the resulting PostScript drawing and correct the sketch as needed. The feedback loop, again, is rapid. Utilities like `tracepaths` and `traceagent` can also be used manually to verify the data, detect errors, and answer such questions as "What are the implications of changing Agent X?" Designs can be revised (and should be revised) at any time, before or after commissioning, since 90% of the documentation can be regenerated automatically.

The nature of these procedures is interactive. People browse and edit data interactively, then run "batch" jobs to process the data into useful output. A cronned script could regenerate online user documentation every

N hours after commissioning, to ensure that online information was fresh and correct.

We anticipate that user interaction with the documentary information will also be interactive, via Web pages and dedicated small GUI integrated with the rest of the DEIMOS software.

7.5.2 Instrument Calibration, Pipeline Reduction

We speculate that instrument recalibration should be done on a regular schedule by CARA staff after commissioning. We speculate further that this procedure should be semi-automated by means of scripts. The scripts should produce some datafiles which are then automatically imported into the database when the instrument specialist is happy with the calibration. These calibration data will be used later for observing and data reduction.

The nature of this procedure is semi-interactive. The instrument specialist interacts with scripts and possibly a simple GUI, but the data capture, assembly, and import is done invisibly and automatically.

The nature of pipeline reduction is semi-interactive. The observer interacts with a simple GUI to establish general parameters; the GUI retrieves appropriate calibration data for the image to be processed, and generates a configuration file to be read by the actual pipeline reduction program. The exact format and content of these configuration files is still under discussion, but they will almost certainly be FITS table extensions such as we already know how to produce automatically from the database.

7.5.3 Slit Mask Fabrication and Management

This topic is covered in detail in Chapter 1. In brief: slit mask designs are requested by mailing (or otherwise conveying) a standard Design/Blueprint File to the Observatory. This file is ingested into the database. The Mill Operator's GUI constructs a list of milling tasks from the database of pending Design Files, assigning priority according the "need dates" supplied by the observer. It then assists the operator through the process of selecting masks to be milled, generating mill control language, and milling. Acceptable masks are "scanned in" during quality control inspection, which causes the mill date, mask blueprint, the fact of successful milling, etc. to be recorded in the database. The barcode label on the milled blank will from now on be associated with the complete design and specification for that mask. This procedure is semi-automated, requiring no manual user interaction to load or retrieve data.

Recognition of masks at observing time is fully automated. Once the cassette is loaded, it can be scanned under software control and all data for all loaded masks can be automatically retrieved.

7.5.4 Performance and Failure Analysis

Chapter 8 (Alarms and Logging) discusses DEIMOS needs for both event and parametric logging. The procedures for capturing log data should be fully automated. Event logs are usually the record of alarms and other conditions of which the user is notified in real time; they may also be statistically analyzed for frequency of event types, etc. and therefore may be captured in the form of database tables.

However, the data more likely to be intensively analyzed and visualized by engineers and support staff are the mass of parametric log records. Procedures for capturing these data are trivial (monitoring KTL keyword values).

Procedures for analyzing captured parametric data, however, are not so trivial. They should be both interactive (exploratory visualization and analysis) and automatic (routine scans for known phenomena). Routine scans should generate email and/or update Web pages to inform observers and engineers of their results.

The UI should facilitate statistical queries by intuitive time periods and provide quick and useful graphical representations of results. Here the rapid retrieval and flexible query capability of the online database are essential in crafting a user interface fast and easy enough to encourage effective use of the data.

7.5.5 General Data Maintenance

Besides the need for occasional updates or corrections to keyword and design information, there will be need for certain basic maintenance activities on growing data stores (like the parametric log table(s)). Reindexing of rapidly-growing tables is necessary to preserve performance, and eventually (over a period of two or more years) log tables may grow beyond the bounds of real utility. Re-indexing can almost certainly performed by a cron job, but data pruning is a more critical operation and should probably be performed (at long intervals) by a CARA or Lick staff person. All automated data maintenance tasks must send confirming mail and log their activities, to ensure that staff are made aware of their success or failure.

Procedures which must be performed only at long intervals are easily forgotten; some automatic reminder mechanism will be needed to alert staff when such procedures are needed. Warnings and reminders should be issued well in advance of any actual failure date, so that staff have adequate time for scheduling these tasks.

Another procedure that cannot effectively be automated is the update of the Memes database to reflect the appearance of new keywords or changes to the usage of existing keywords. One or two instrument specialists, programmers, or engineers will need, over a period of months or years, to update the Memes database occasionally when keywords or reference tables change. Synchronization of the Santa Cruz and Waimea copies of the database is a serious concern, but one which I feel can be solved using existing resources.

The data maintenance procedure which represents real costs in staff time is the proposed Instrument Recalibration, requiring several hours of instrument time and of an instrument scientists' time as well. The database ingestion phase of this procedure is not the cause of its lengthiness (the length is driven by the minimum useful ELAPTIME and the number of exposures needed to calibrate); and the ingestion itself should be automatic once the actual calibration procedure has succeeded. The capture and maintenance of these calibration data may be the most complicated of all database interfaces for DEIMOS, but we reserve the option not to deliver a data reduction pipeline and therefore to postpone all modules associated with it, including the tools for instrument recalibration.

7.6 Deliverable Documents

Most of the documentation to be delivered *with* the database component of DEIMOS is in the form of data resident *in* the database. Documentation products that should be delivered at commissioning are:

- A fully populated set of database tables containing all the base information about DEIMOS (keywords, optics, detectors). These tables in combination with certain tools (below) constitute the bulk of the deliverable documentation.

- A stable release of all document generation tools, and a Makefile or script which generates both LaTex and HTML versions of all the instrument documentation including this chapter.

- A handcrafted cookbook explaining data maintenance, the relationship of tables to one another, the use of Forms, Wisql, and how to read `msanity` output, etc., with diagrams and friendly text for the naive reader.

- PostScript copies of any publications (ADASS, TCL97) relevant to the information management subsystem, for local reference.

7.7 Database Schema

The complete database schema documentation (detailed information on every table) is to be found in the LOTR. These figures, in conjunction with section 7.1, provides an overview of the schema and its function which may be helpful in understanding the minutely detailed material in the LOTR.

The schema is divided conceptually into subschemata, for ease of comprehension and reference. Here we present EER drawings of the subschemata, to show the number of tables and their general function and relation to one another. The semantics and other attributes of each field will be found in the LOTR, indexed by field name and by table name.

In these drawings, only key fields are explicitly referenced. The notation PK indicates the primary keying of the table, while FK indicates the foreign keying. Arrows are drawn in the traditional EER "one-to-many" orientation, with the arrowhead on the "many" side of the relationship. Nuances such as one-to-one and many-to-many are not expressed at this revision of the EER generator.

All of these tables function both as documentary sources (for the generation of online or paper documentation product) and as online references for automated, semi-automated, or user-driven procedures. The information flow diagrams earlier in this chapter show output from the database being used by various agents. Where the general meme names "mumbleData" have been used, this indicates use of various information from the corresponding schema.

The drawings which follow are another product of the Memes database, which is self-referential and self-documenting.

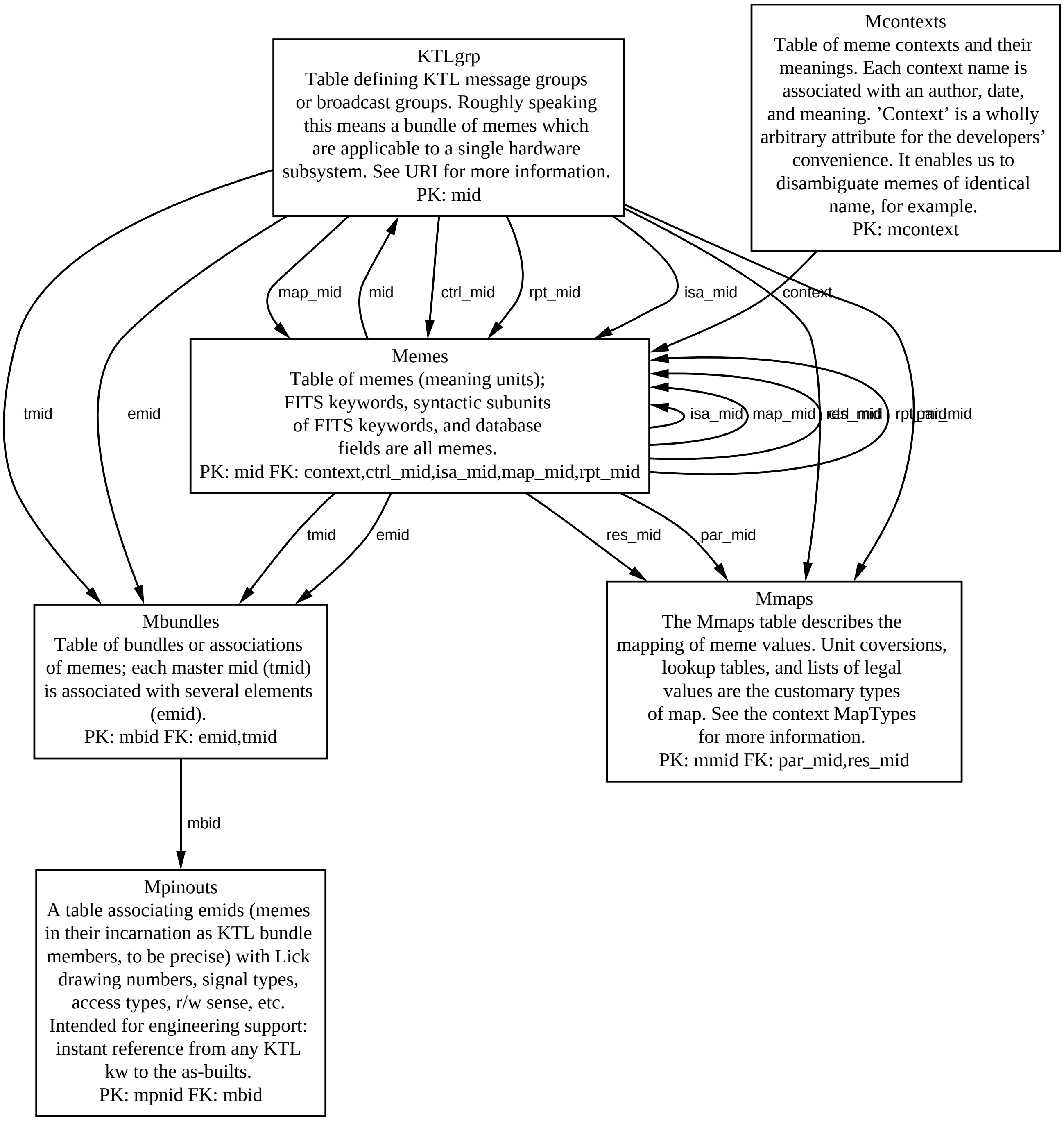


Figure 7.8: Memes schema EER

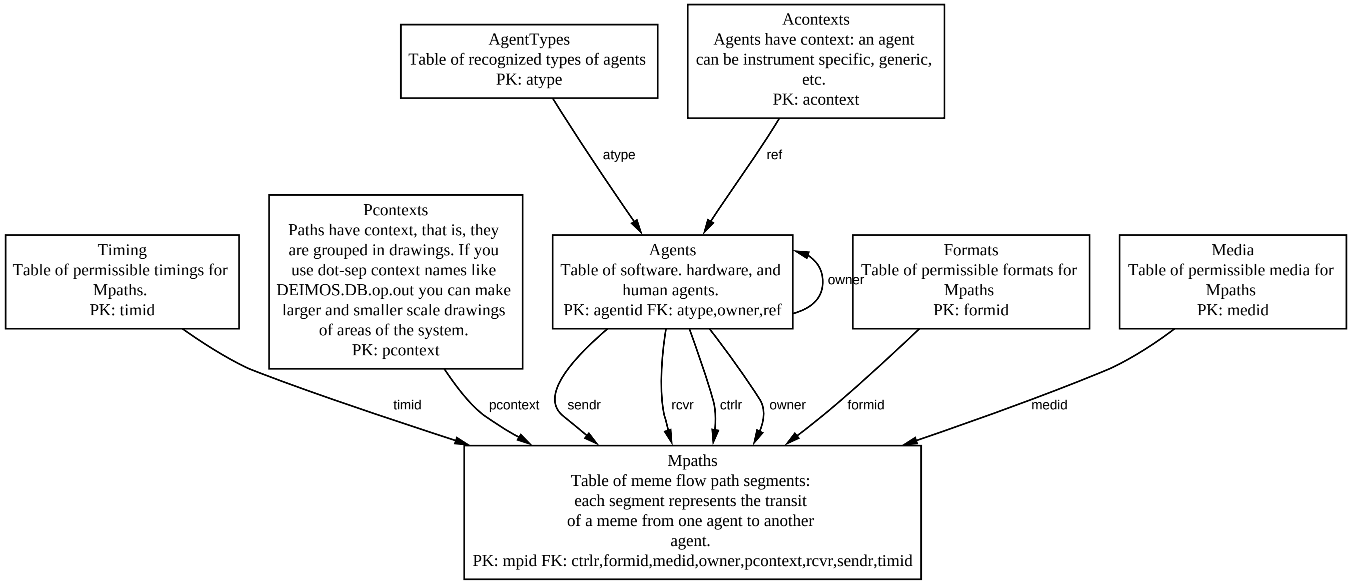


Figure 7.9: Information Flow EER

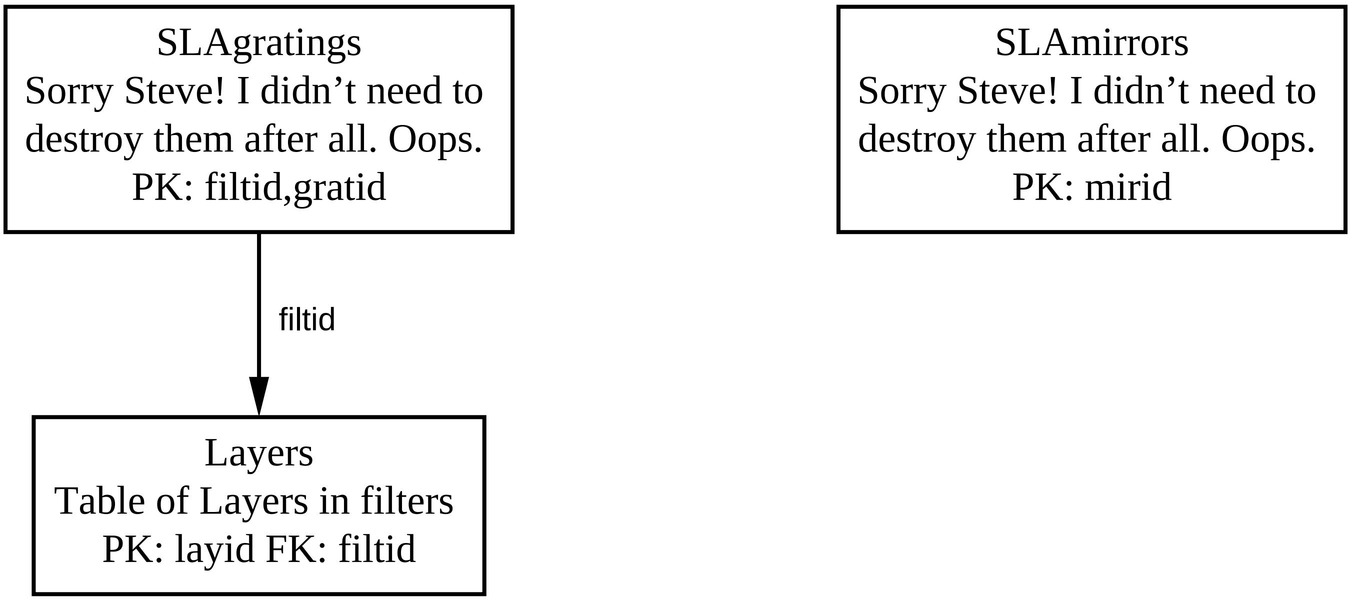


Figure 7.10: Glass schema EER

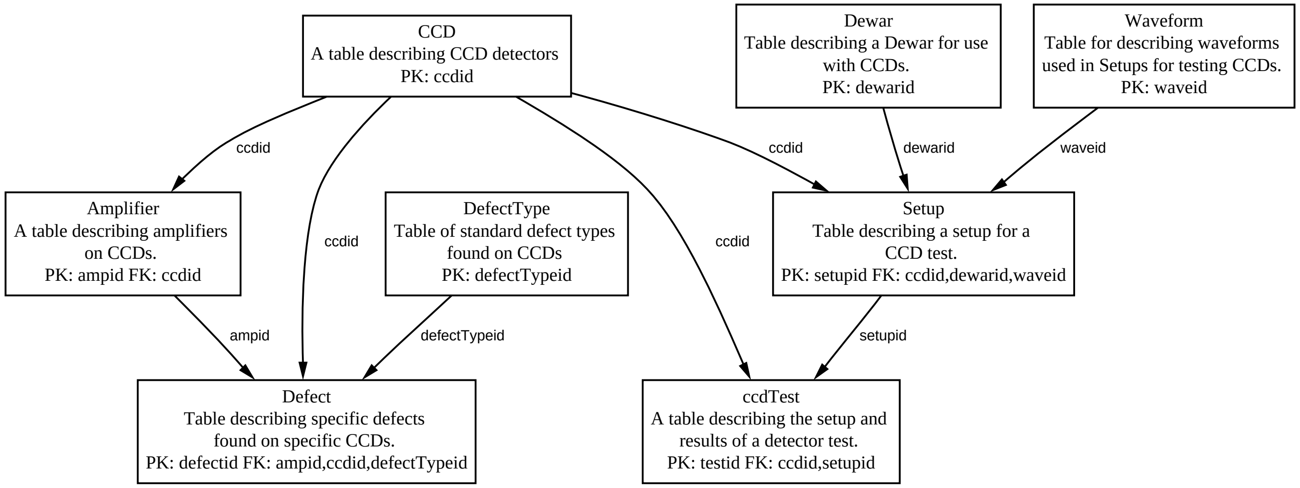


Figure 7.11: CCD schema EER

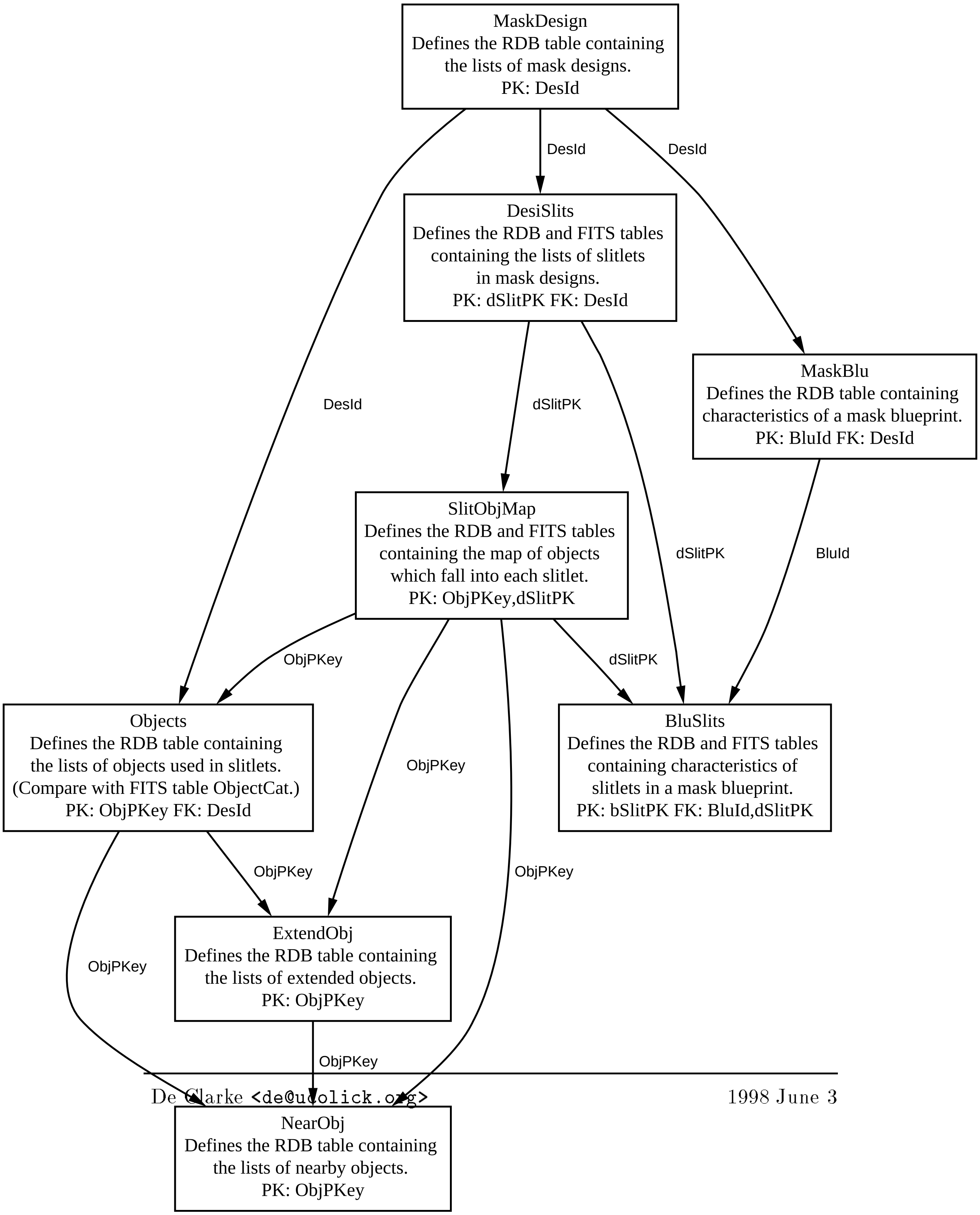


Figure 7.12: Mask Design EER

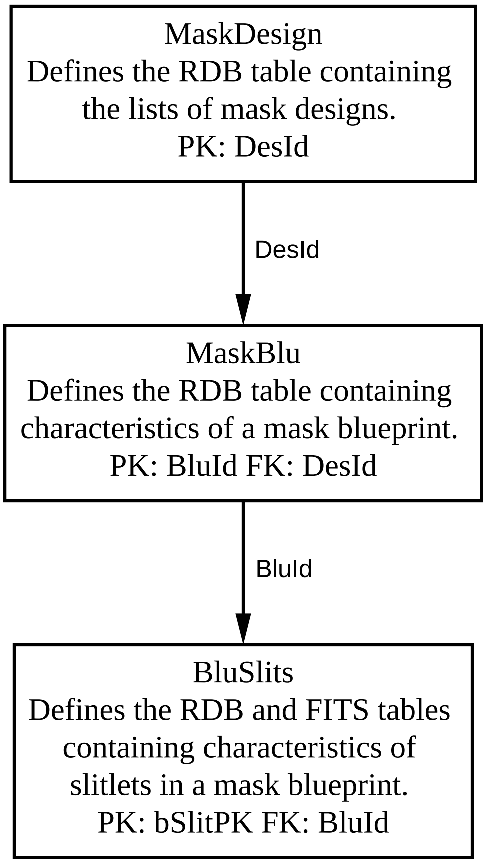


Figure 7.13: Mask Fabrication EER

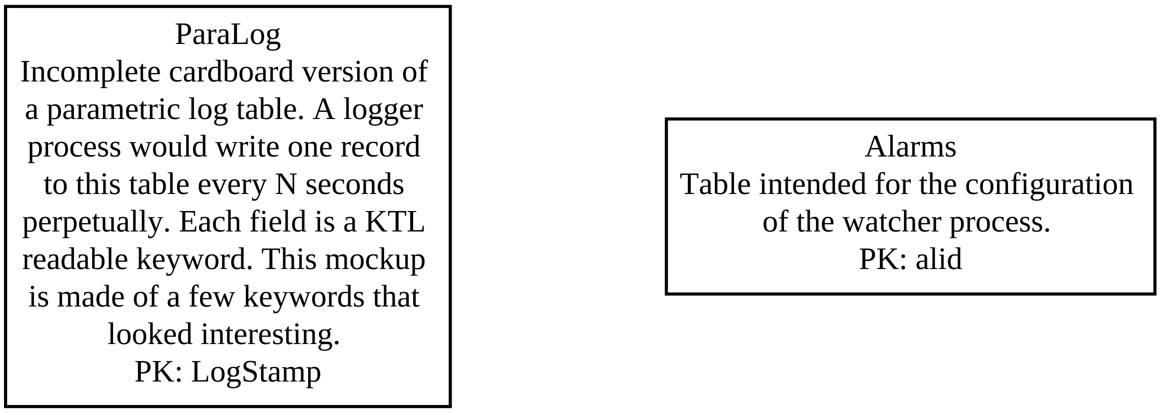


Figure 7.14: Operational Schema EER

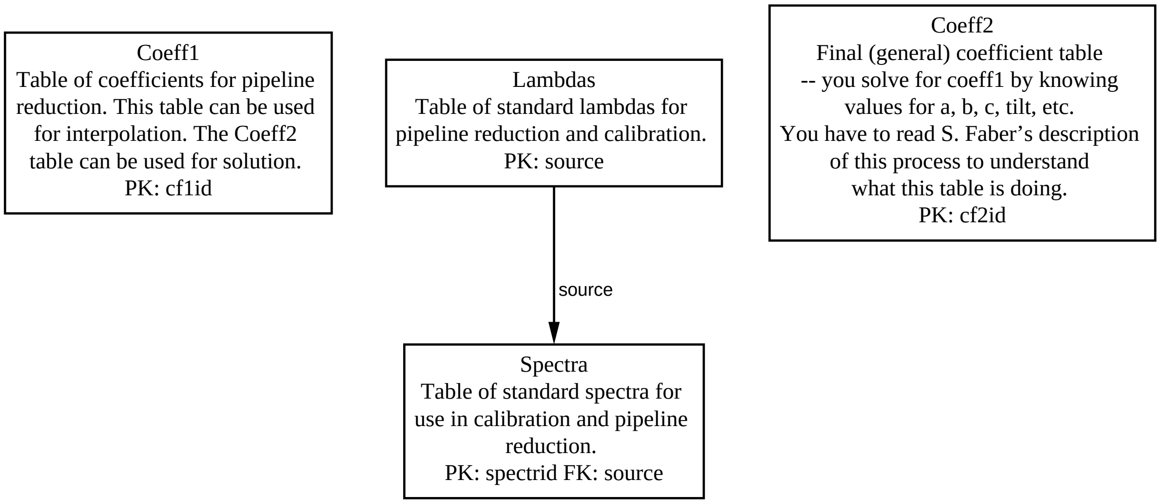


Figure 7.15: Calibration Schema EER

Chapter 8

Environmental Monitoring and Logging

8.1 Overview

Our goal for the DEIMOS instrument is to learn from, and improve on, our experiences with error logging, alarm delivery, and condition handling for previous Lick-built instruments. Specifically, we wish to provide a logging and monitoring facility that will:

- detect and respond to dangerous conditions, preventing damage to the instrument or physical injury to staff, and leave a reliable and accessible record of these conditions and responses
- detect and respond to conditions threatening observing efficiency or success, and alert observers and OAs to these conditions reliably (and promptly), leaving an accessible record of these alerts
- respond where possible in pre-emptive ways (preventing failure rather than merely warning of imminent failure)
- provide a reliable and accessible log of every KTL command sent to the instrument by the user(s)
- (at least during the first year of operation) routinely monitor and log a wide spectrum of instrument and telescope operating parameters, to facilitate post facto investigation of instrument faults and failures

Our efforts fall into three major categories:

- Robustness (responses that prevent any interruption of observing, preserve data, avoid error)
- Event logging (logging particular discrete events as they occur)
- Parametric logging (logging operating parameters at regular intervals for later analysis)

8.1.1 Robustness

We have established a DEIMOS software design principle: all process which log or preserve information, from the image capture facility to the parametric monitoring software, should have a "fallback chain" of strategies and locations for storing data. In other words, information should not be lost if, for example, database server performance degrades beyond acceptable limits, database server crashes, or disk space fills up.

Every process which logs directly to the database server must switch to a fallback mode if the database server does not respond within a suitable timeout period. The fallback mode (writing to a disk file) itself must have further fallback options (different disk partitions to use if the default partition fills).

Furthermore, no logging or monitoring process may, even in the event of complete failure, interrupt or impact the observing process. Even if no logging is possible, observing should continue as long as the instrument and infrastructure will support it. Observing should only be impacted in the case of instrument or personnel safety hazards.

The fallback rule should apply to image capture and storage as well; the write-to-disk operation should continue smoothly even if the default OUTDIR partition fills. The chain of alternate disk write areas, in all cases (logging and image capture) must be readily visible to the user. The value of OUTDIR must change when a new fallback disk is selected, so that the user can readily see where current images are being stored. Furthermore, a new keyword (DISKFREE) should be supported which displays in megabytes the amount of *total* disk space remaining in the entire fallback chain for image storage; and two new keywords DIREIMGS and SPECIMGS should display the number of direct or spectral (respectively) DEIMOS images which will fit in DISKFREE megabytes.

These principles and plans are intended to ensure that science data have the highest priority and will not be lost (or even mislaid) in the course of the night.

8.1.2 Event Logging and Response

We must consider two levels of "event". One is an KTL event (a write to a KTL keyword), the other is the detection of a condition (involving one or more keywords) which we have defined as noteworthy. All KTL writes should be logged, along with their source if it is identifiable. This is fairly simple to do (see below). Conditions, are somewhat more complicated, as they can exist at levels below KTL's consciousness. The hardware control crates can perceive and respond to certain conditions which are not visible at the KTL layer. Therefore, condition detection and response has to take place independently at both levels: in the crates, and at the control computer.

The crate-level condition detection should be restricted to safety hazards, stage collisions, hardware failures, and so on: conditions which may cause damage to the instrument or to staff, or which will disable observing altogether. The crate should log these conditions and its responses in a local (*not* NFS) log file. This log file should be accessible in some way from the control computer, but the live copy must reside on local disk. The crate condition handling code will have to be hand-crafted. However, since it must detect a limited number of severe conditions related to hardware, limits, interlocks, etc., it is unlikely that the condition handling code will be volatile.

The "Dashboard" tool already contains a fairly sophisticated KTL keyword condition processor. This facility can evaluate Boolean and arithmetic expressions in which KTL keyword names are terms; the KTL keywords need not be from the same KTL service, so complex conditions can be evaluated involving keywords from both DCS/ACS and the instrument. Any codable action can be taken as a result, e.g. popping up alert windows, altering the visual appearance of the GUI, sending mail, or executing other arbitary Unix commands.

The "Dashboard" configuration delivered for use as a DEIMOS GUI will doubtless include many "alarms and alerts" as part of the GUI. However, we feel the need for a watchdog process not tied to the user GUI, one which runs as a daemon (similar in spirit to the `watch_hirot`) process used now to monitor the HIRES rotator. This process could be effectively a subset of the Dashboard code (just the condition handling); once configured from an authoritative database table of conditions, severities, and responses, it would run continuously, monitoring as many services as necessary (and keeping track of available disk space, updating DISKFREE and the *IMGS keywords).

Another design principle that has emerged from our discussions is that alarm and alert delivery must be multimodal. We have found that it's difficult to get the observer's attention, in practice, when an alarm condition exists. Audio alarms via the /dev/audio port are ineffectual if the speaker is turned off or relocated out of earshot. Console window alarms are almost always ineffectual because the console window is usually hidden or iconified. Our conclusion is that important alarms should be delivered by multiple media (audio, popup alarm boxes, and mail, for example) and that drastic changes in the appearance of the screen are necessary to get the user's attention. We will design our DEIMOS alarms and alerts with these conclusions in mind.

8.1.3 Parametric Logging

Why bother to log parametric information, when a lot of it ends up in image headers anyway?

Particularly in the first year of operation (but arguably, throughout the instrument lifetime) a great deal of utility may be derived from a fairly comprehensive parametric (environmental) log. Such a log would include all raw telemetry values from the instrument (there may be a need for converted values, but raw values are the minimum requirement), and selected values from the dome and telescope (and possibly ambient condition monitoring hardware like thermometers, wind gauges, hygrometers, etc.)

This seems like a lot of data. One may well ask what is wrong with the existing procedure: when engineers want a log of certain parameters, they write a script, put it in their crontab file, and let it run for a few hours or nights; then they look through the log file it generated. The existing procedure has several disadvantages offsetting its one advantage (parsimony of disk space).

When an event takes place (instrument failure or mysterious instrument behaviour) it is often difficult to reproduce. If a comprehensive parametric log were always available, one could investigate and analyze an event the following day, rather than "set the traps" and wait hopefully for it to happen again.

In specifying a narrow filter of parameters to log, one makes implicit assumptions about the cause of the problem which may not be true. Often, the real question is not "Does X correlate with this event," but "Does *anything* correlate with this event." An accessible and comprehensive parametric log gives the engineer a larger model of the instrument to analyze when searching for the cause of a problem, and the opportunity to make discoveries and connections outside the bounds of his/her initial assumptions.

When individual engineers make individual tailored log files, those files and the data they contain are semi-private; they are not easily cross- correlated, and their location and nomenclature may be somewhat obscure to anyone other than the owner. A centralized online parametric log is available to everyone at all times, from astronomers to OAs to engineers.

Invidual tailored log files created and deleted at whim do not form a continuous historic record of instrument behaviour and performance. It is unlikely that an astronomer will be able to make arbitrary inquiries about instrument performance with regard to a particular image taken at a particular date and time. We feel that the astronomer may find it very useful to make such queries, and that these queries should not pose a burden on the observatory staff by requiring them manually to integrate disjoint private logfiles into a dataset for analysis.

Obviously it is not possible to capture every last operating parameter for the instrument. We can only capture what our telemetry gathers and presents in the form of keywords. However, we can try to ensure that keywords exist for as broad a set of parameters as is practical to implement.

Why should such a log be written directly to the database?

As a text file, it would be fairly unreadable. If written as keyword=value on separate lines, it would be unreadable due to the sheer number of lines and the laboriousness of parsing it into a horizontal format for plotting, etc. If written as records, the association betwen column and value will be very hard to maintain; and `awk` and `grep`, while powerful and handy tools, become somewhat unwieldy when used as a programming language for dealing with multi-hundred-column, multi-hundred-Krecord datasets. A relational database is designed to perform retrieval and statistical analysis on exactly this type of data set. To save the log to disk, in a relatively unusable form, then implement a scheduled ingestion of the data into the RDBMS, seems like unnecessary overhead.

Various tools, both commercial and free, are available for browsing, analyzing, and visualizing large RDBMS datasets. Once the information is in this form, there is considerable flexibility of choice in tools and methods for access.

How manageable would the data volume be?

The record length is more of an issue than the record count. The record count, assuming a 5-minute sample interval 24 hours per day, is about 105K records per year. Three years' worth of online record is by no means an excessive record count for rapid retrieval. The field count (number of columns in the table) could be high (between 100 and 200?). If raw values are being preserved rather than multiple value conversions and string representations, we could assume about 4 bytes per field, so a record length of between 400 and 800 bytes. Averaging that to a Kbyte, we could estimate a total dataset size of 55MB per year. This is not considered, by today's standards, an exceptionally large dataset. How much of this information would reside online, and how soon it would be flushed to secondary storage, are open questions.

Is this ambitious?

Actually, it's rather trivial to implement. The more challenging aspect is making intelligent use of the accumulated data. Capturing and storing the data are very simple problems, representing a very tiny fraction of the overall DEIMOS software effort. To make good use of the data, DEIMOS engineers will need to acquire or build tools; our feeling is that many good tools exist already for data visualization and analysis, so that retrieval is the primary issue. We already have in hand several flavours of "friendly" interfaces to database information retrieval, and by enhancing these, or acquiring similar products from other institutions, users other than database experts should be able to perform exploratory and analytical queries with relative ease.

8.2 Component Modules

8.2.1 EngDataVisu

Function Tool for retrieving and visualizing database contents.
Language Tcl/Tk?
Type User Tool
Author Clarke
Revision
Source CVS Ref
Essay This tool (not yet written) would present an easy and attractive interface to engineers and astronomers wanting to make exploratory queries into any database information. The express purpose is to grant analytical access to the instrument's operational history (parametric logs), but statistical analysis of image headers or other datasets could also be performed. The precise nature of the user interface is TBD. There are strong reasons for preferring a web-based approach, but some limitations of Web browser function might encourage a standalone, dedicated GUI instead. Whatever the nature of the GUI, it must offer basic plotting capability, and must handle the specific kinds of time-based statistics needed for understanding log data: statistics by month, week, day, fortnight, hour of the day, etc. Again, there are reasons for preferring to graft onto an existing data visualization suite,

and other reasons for preferring a standalone application with built-in graphics.

Some version of this tool, however constructed, should be available at engineering test time; a fairly complete version should ship with, or before, the instrument.

8.2.2 KTLwatch

Function Watches selected KTL keywords and takes action if conditions are met.
Language Tcl
Type Daemon
Author Clarke
Revision
Source CVS Ref
Essay A stripped down version of Dashboard (not yet written), running as a daemon; like Dashboard, it can configure itself from the database directly or from a cached local file of information originally retrieved from the database. It relies upon the keyword (Memes) tables and an additional table of Alarms (boolean pseudo-tcl expressions, severities, handlers etc.)

It can take any arbitrary action on fulfilment of a condition, or a default action depending on the severity of the condition.

8.2.3 ElectroLog

Function Online observer logbook
Language TclTk
Type Tool
Author Deich
Revision
Source CVS Ref
Essay ElectroLog (currently in early beta) is an X11 GUI observer logbook which accept both arbitrary interactive input from the user, and data input on socket connections from processes. It can easily be configured into any KTL system involving "Dashboard" in such a way that any KTL event can write a formatted record of KTL keyword values to the GUI logbook. A simple example might be to write, at end of exposure,

a user-selected list of KTL keywords as an observing log record of the exposure. The user can then annotate the logbook at will.

ElectroLog has various handy features including log print, log save, mail log to recipient, and so forth. It can retroactively ingest keywords from an entire directory full of FITS images, making a *ex post facto* observing log.

ElectroLog will be tested on Mount Hamilton with Lick instruments long before DEIMOS commissioning, and if it has won popularity by pre-ship review, will probably be included in the DEIMOS software suite as a "value added" feature.

8.3 Subsystem Procedures

8.3.1 Data Capture

Data capture for logs should be fully automatic. The only exception is the online observer logbook (ElectroLog), in which the observer can enter arbitrary personal comments; and these comments should probably not be ingested into any public record.

8.3.2 Data Retrieval

As we mentioned above, retrieval and visualization of data are the real challenges here. The data retrieval and review process is a combination of manual (user specifies the "slice" of data to be retrieved) and automatic (SQL is automatically generated from a GUI interface, or filters are automatically swept across the data on a regular interval). The automatic generation of complex SQL statements involving datetime functions is the only feature which does not already exist in one deployed application or another; a UI for datetime queries, and reliable code generation for such queries, will be the features distinguishing EngDataVisu from (e.g.) Wisql.

8.4 Deliverable Documents

Complete schema documentation should be provided with the delivered instrument (see Chapter 7). A complete user guide and cookbook should be provided with any data analysis or visualization tools that ship with DEIMOS. As mentioned in Chapter 7, similar guides and cookbooks are required for all the other database utilities.

Chapter 9

Acceptance Suite

Part III

DEIMOS Calibration: Procedures and Database Requirements — DRAFT

DEIMOS Calibration: Procedures and Database Requirements

A. C. Phillips, S. M. Faber

DRAFT: 1998 June 3

Overview

The term "calibrations" encompasses a large range of issues and procedures, and the current document is not exhaustive. We focus on photometric and astrometric calibration of *direct images* through the standard DEIMOS filter set (BVRI) and flux and wavelength calibration of *multislit spectra*, including longslit spectra of single objects as a special case. These categories comprise the great bulk of DEIMOS data. We further assume that the instrument consists of Side A only, although the coordinate systems for Side B have also been defined.

Two data calibration modes are considered:

- **Pipeline Mode:** This is analogous to the current mode for reducing Hubble Space Telesccope data, in which a standard suite of reduction programs removes the instrumental signature using a stored library of calibration data. The necessary software would be supplied by the DEIMOS project and would run both at Keck and at the observer's home institution (provided the appropriate hardware and operating systems are available). The ultimate aim is that pipeline data reduction will occur on-line as the data are taken, with observers returning home with data from which the instrumental signature has been removed. A further longterm aim is to minimize taking calibration data during observing and to use instead the stored library of calibration data to the fullest extent.

  Pipeline Mode at this point is only a *goal.* Its implementation is not supported by the current DEIMOS software budget, and it further presupposes stability of the (Keck II Telescope + DEIMOS) hardware system, which is currently unproven. Nevertheless, we are hopeful of implementing Pipeline Mode eventually and are designing the DEIMOS database to be able to accomodate it.

- **Baseline Mode:** Baseline mode is the traditional mode of data reduction at ground-based observatories, in which each observer is responsible

> for gathering the necessary calibration data and reducing the data. We recommend recipes for doing this with DEIMOS and outline a partial long-term calibration data base that will be useful for certain aspects of Baseline Mode calibrations ??. Software and database entires needed to implement Baseline Mode should be available at instrument commissioning or soon thereafter.

We begin with a discussion of *coordinate systems*, which are needed to understand the relationships between sky, focal plane and detector. This is followed by a description of the major calibration elements, the major dependencies for each of these elements, and general procedures for addressing each. We next discuss the detector calibrations, imaging data and spectroscopic data in turn, from the perspective of specific procedures and database products necessary for calibrating each stream of data. The final section covers routine diagnostic tests to monitor system stability, e.g., optical alignments.

This document presumes a working knowledge of the DEIMOS instrument, to be found in the Preliminary and Critical Design Reports, the Software Preliminary and Critical Design Reports, the Flexure Compensation System Report (TBD), and ??.

Chapter 1

Glossary of Terms, Acronyms, and Symbols

PA: Position angle of the DEIMOS x-axis with respect to the sky, measured north through east.

RPA: Rotator position angle of the DEIMOS spectrograph x-axis with respect to the Nasmyth platform. At $0°$, the slit direction is vertical with Side A on the right when looking at the focal plane from the outside. From this view, RPA increases as the instrument rotates counter-clockwise. This also corresponds to increasing PA of the slit on the sky.

FCS: Flexure Compensation System.

T: Temperature inside DEIMOS, in °C. (Nominal T = 2°C)

P: Atmospheric pressure, in mm of Hg. (Nominal P = 615 mm)

FKCS: Fundamental Sky Coordinate System.

RSCS: Refracted Sky Coordinate System.

MFCS: Mask Form Coordinate System.

SMCS: Slit Mask Coordinate System.

ICS: Image Coordinate System.

CCDCSi: Pixel coordinate system of CCDi.

FCSj: Pixel coordinate system of Flexure Compensation CCDj.

TVCS: Pixel coordinate system of the TV Guider CCD.

GOH: Grid of Holes mask, for direct imaging.

LOS: Line of Slitlets mask, for spectroscopy.

LOH: Line of Holes mask, for spectroscopy.

SGOH: Staircase Grid of Holes mask, for spectroscopy.

SGOTH: Staircase Grid of Tiny Holes mask, for spectroscopy.

FPFs: Focal plane fiducial marks. Two marks within the slitmask area near the focal surface and projected onto the fiducial CCD during direct imaging.

Fiducial CCD: The fiducial CCD is No. 3 in the CCD array. It is the first CCD installed in the focal plane and the one to which the others in the array are aligned. Images of the focal plane fiducials fall onto the fiducial CCD during direct imaging.

Slitmask Form: The cylindrically curved form into which slitmasks are pressed. Approximates the spherically curved telescope Nasmyth focal surface.

Astrometric Mask: A slitmask for direct imaging with boxes at the expected positions of astrometric standard stars. The displacement of the stars relative to the boxes provides corrections terms to the mapping RFCS $\leftrightarrow$ MF/SMCS.

Offset Guide Mirror: A permanently fixed mirror mounted in the focal plane next to the slitmask area. Its image of the sky is visible in the TV Guider at all times regardless of whether a slitmask or longslit is inserted.

TVFs: TV fiducial marks. Two marks etched onto the offset guide mirror, parallel to the focal plane fiducials within the slitmask area, and permanently visible in the TV guider.

Chapter 2

Coordinate Systems

2.1 Overview of Coordinate Systems and Mappings

To discuss calibrations, we first need to define several coordinate systems, enumerate potential variables and dependencies for each one, and describe how we will map from one to the next (see Figure 2.1). A *mapping* from one (two-dimensional) coordinate system to another consists of the following elements[1]:

- zeropoint offset in x and y;
- rotation;
- scale; and
- distortion terms.

This section briefly sketches how each of these elements will be determined, with more details to follow in Section 2.2 and Chapters 5, 7 and 8.

Fundamental Sky Coordinate System, FKCS: The fundamental celestial coordinates of targets. These depend on *epoch* (to account for proper motion, aberration of starlight) and include an *equinox* in their description.

[1] Except for the analytical mappings in Section 2.2. However, the analytical mappings could be used to generate mappings in the same form described here

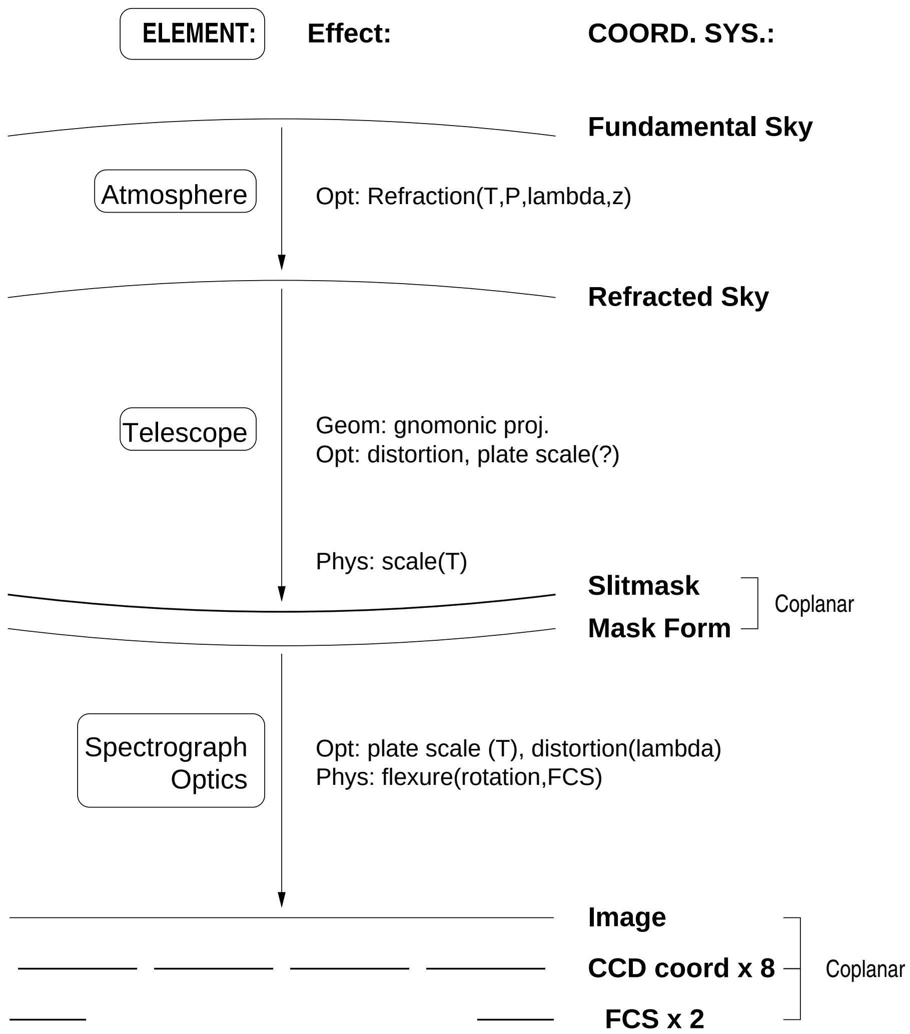


Figure 2.1: Coordinate Systems – Overview

Refracted Sky Coordinate System, RSCS: The apparent celestial coordinates as seen by the telescope. Required for slitmask design (forward mapping) and astrometry from direct images (backwards mapping). The mapping FKCS $\leftrightarrow$ RSCS depends on *atmospheric pressure* and *temperature, observed wavelength,* and *zenith distance* (neglected dependencies include humidity) and will be provided by standard models of refraction.

Mask Form Coordinate System, MFCS: Closely related to the Slitmask Coordinate System, SMCS (next), MFCS is a coordinate system close to the telescope image surface that is rigidly tied to the spectrograph. It is a rectilinear coordinate system defined on the cylindrically curved surface of the slitmask form with the x direction following the curved surface of the form, and the y direction perpendicular to that (see Figure 2.2). Its zeropoint and orientation are defined by two *focal plane fiducials* (FPFs) that are attached to the slitmask form near the focal surface. These project onto the *fiducial CCD* during direct imaging (see below) and are also visible in the TV Guide Camera, tying all these systems together. The MFCS coordinate system is used to align downstream coordinate systems such as the fiducial CCD and to derive the mapping from DEIMOS' front image surface to the camera focal plane. There is a slight dependence on spectrograph *temperature*, so the MFCS is defined at DEIMOS nominal operating temperature).

Slitmask Coordinate System, SMCS: The coordinate system tied to the physical slitmask itself. Ideally, this system is identical to the MFCS (above), but in practice the two may differ by small translations and rotations owing to slitmask insertion errors (typically $< 10\mu$m). For most purposes, these two systems can be considered identical and will be referred to as MF/SMCS (Figure 2.2). The SMCS is a rectilinear x, y system defined on the flat slitmask stock. Its fiducials are the edges of the mask that rest against mounting stops on the mill and on the slitmask form. There is a dependency on slitmask *temperature*, so the SMCS is defined for the nominal operating temperature. The mapping RSCS $\leftrightarrow$ MF/SMCS is determined to first order by the known mapping of the refracted sky onto the telescope Nasmyth focal plane, and thence by projection of the focal plane onto the tilted cylindrical slitmask. This first-order mapping is tuned by observations of astrometric standard star fields through *astrometric masks*.

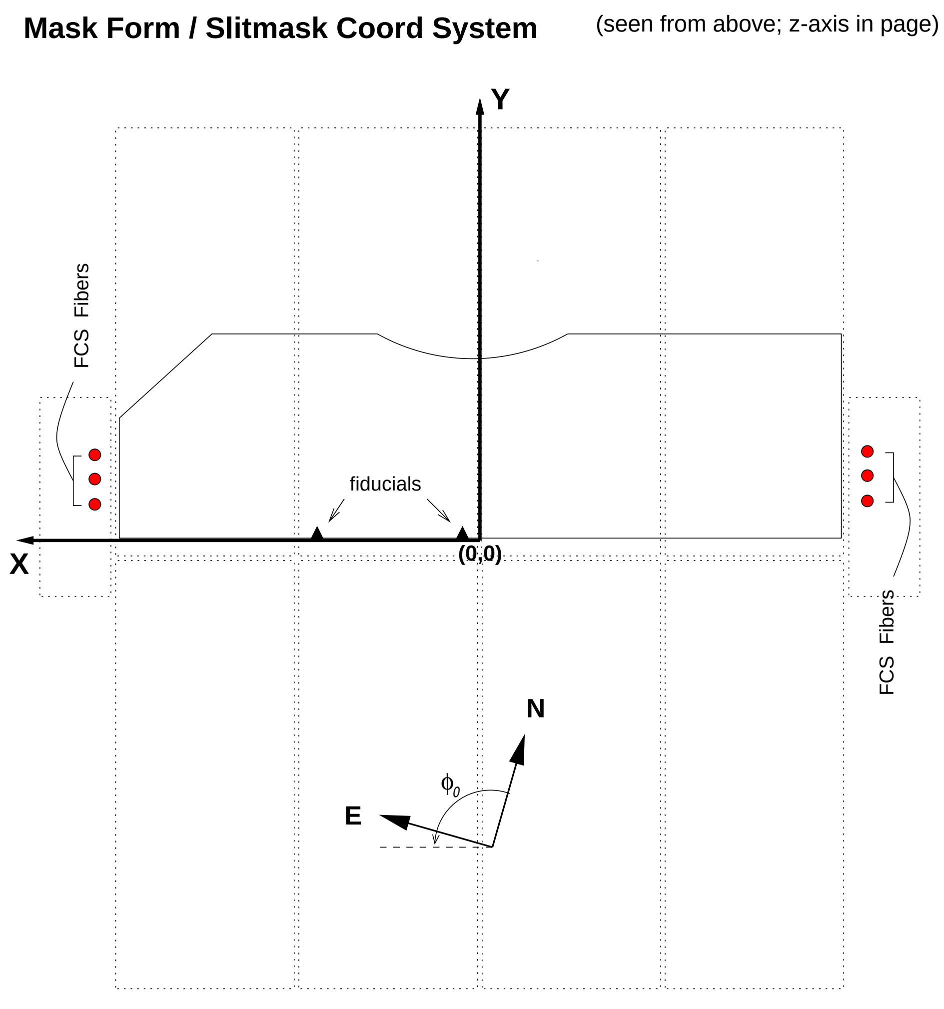


Figure 2.2: Mask Form / Slitmask Coordinate System, (MF/SMCS)

Image Coordinate System, ICS: This system lies in the focal plane of the spectrograph camera, coplanar with the adopted plane of best focus. Its axes are parallel to the imaged axes of the MFCS, as set by images of the focal plane fiducials (FPFs) (see Figure 2.3). The ICS may be thought of as the pixel system of a single, idealized 8K $\times$ 8K CCD with a square pixel grid having the same scale as the real CCDs. *[Zeropoint ?? — probably the projected zeropoint of the mask form, or the (nearby) optical axis of the camera. The later is not clearly measurable, but doesn't need to be, and it conceptually simplifies discussion of mappings.]* In practice, the mapping MF/SMCS $\leftrightarrow$ ICS is determined by measuring the individual mappings MF/SMCS $\leftrightarrow$ CCDi (see below), corrected back to the ICS using the mappings CCDi $\leftrightarrow$ ICS (see below).

There are two types of MF/SMCS $\leftrightarrow$ ICS mappings, one for direct imaging and one for spectroscopy. Each filter and (grating+tilt) has its own mapping. The direct imaging mappings consist of scale and distortion terms only (since the orientation and zeropoint of the ICS were defined above). These are determined by direct images of a special slitmask with a finely and accurately milled grid of small holes (*grid-of-holes mask*, GOH). The direct imaging mappings involve only CCDs 1-4. The spectroscopy mappings involve a wavelength function that maps pixel coordinates onto λ and a spatial function that maps pixel coordinates onto position along the slit. These functions are determined by arc spectra taken through a mask with a line of small holes spaced along the x (slit) direction (*line-of-holes mask*, LOH). These mappings may depend on *camera temperature*, *camera focus*, *flexure*, *order-separating filter* (for spectroscopy mappings), and the current *flexure compensation system (FCS) zeropoint.* A nominal mapping is therefore defined for standard T, rotator angle (RPA), focus, order-separating filter, and FCS zeropoint, with families of differential mappings as functions of T, PA, other order-separating filters, and FCS operation.

CCD Coordinate Systems, CCDCSi: Pixel coordinates on the individual CCDs. The *fiducial CCD* is No. 3 in the array — images of the focal plane fiducials (FPFs) are projected onto it during direct imaging (see Figure 2.3), and it will be the first CCD installed into the array if only one is available. The mappings ICS $\leftrightarrow$ CCDCSi consist of the rotation and displacement of each CCD from the idealized

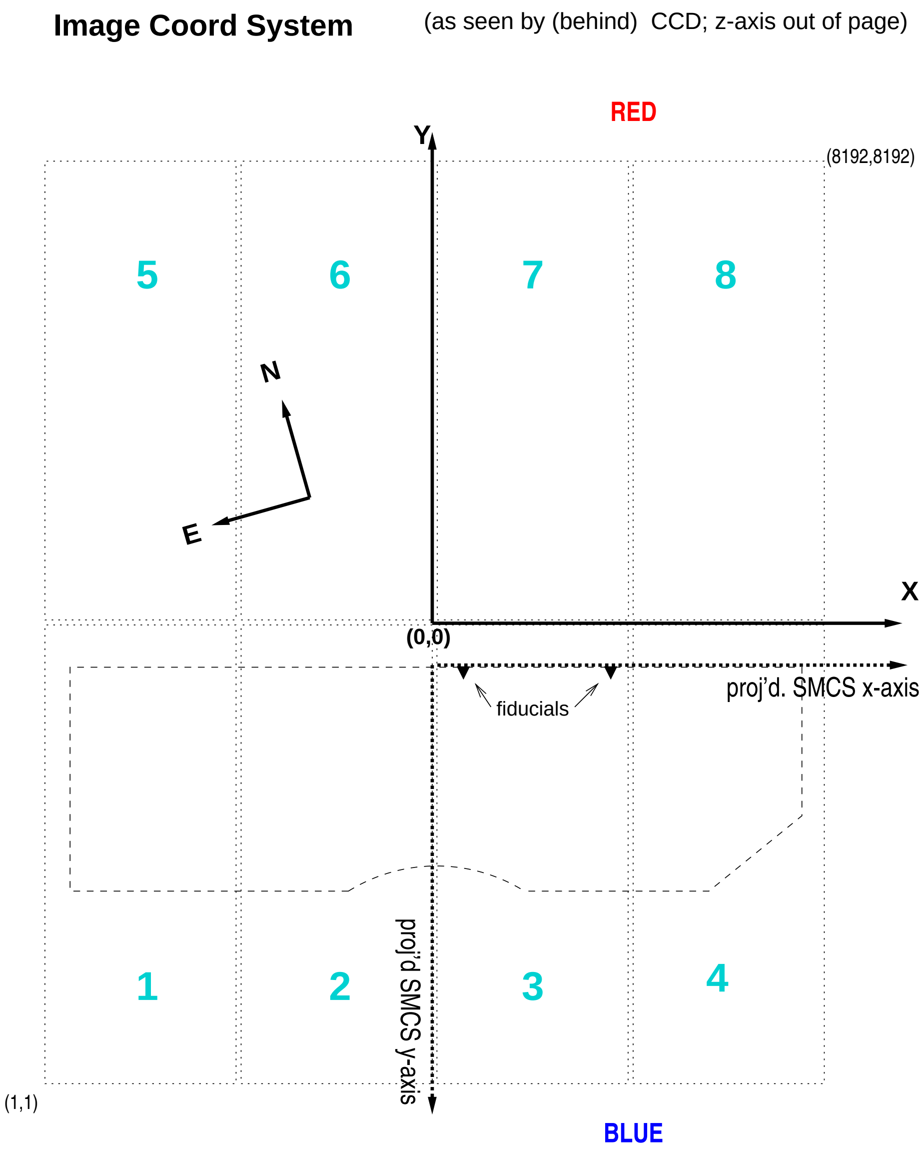


Figure 2.3: Image Coordinate System, (ICS)

focal plane coordinate system, ICS. These quantities are determined for CCDs 1-4 from direct images of the above-mentioned grid-of-holes mask. To extend this information to CCDs 5-8, an arc spectrum is taken through the line of holes (LOH) mask. The known rotations and displacements of CCDs 1-4 from part A are corrected to create the lower half of the idealized, seamless ICS focal plane. The spectral traces (x-heights) and λ solutions from this half are then continued onto CCDs 5-8. Ensuring continuity in both x-height and wavelength serves to determine the offsets and rotations of CCDs 5-8 relative to the ICS system determined by CCDs 1-4.

Flexure Compensation CCD Coordinate Systems, FCSj: Pixel coordinate systems on the two flexure compensation (FC) CCDs, at opposite ends of the slitmasks (see Figures 2.2 and ??). These are used *internally* to hold the image position steady and/or repeat a previous spectrograph setting, so absolute mappings to the other coordinate systems are not needed. However, calibrations are needed to relate *image motion* on the FC CCDs to *image motion* on the main detector (the two are not the same due to optical distortions). Separate mappings ΔFCSj $\leftrightarrow$ ΔICS are needed for each direct filter and grating+tilt combination and will be determined from tests of the flexure compensation system.

TV Guider Coordinate System, TVCS: Pixel coordinates on the TV Guider CCD. The guider has two regions, (i) the offset guiding area, fed by a fixed *offset guide mirror* in the focal plane that is always visible, and (ii) the slitmask area, which is empty for direct imaging but contains a reflective slitmask or longslit for spectroscopy, which images the sky (see Figure ??). For direct imaging, the slitmask is out and the FPFs are visible. Two additional fiducial marks, called *TV fiducials*, or TVFs, are etched onto the surface of the offset guiding mirror. The FPFs and the TVFs are all rigidly connected to the DEIMOS front focal surface and form parallel lines with a known offset. At least one pair of these fiducials is visible in the TV at all times.

Two TV mappings are needed, TVCS $\leftrightarrow$ MF/SMCS and TVCS $\leftrightarrow$ RSCS, but, since RSCS $\leftrightarrow$ MF/SMCS is already known, the single mapping MF/SMCS $\leftrightarrow$ TVCS suffices. The zeropoint and orientation of this mapping are provided by the visible pair of fiducials, whose measured pixel locations, continuously updated, allow constant correction for TV flexure. Guiding thus maintains constant guidestar position relative to DEIMOS' focal plane rather than relative to the TV pixels. Scale and higher-order distortion terms in MF/SMCS $\leftrightarrow$ TVCS are provided by calibration images of astrometric standard star fields. *[...if needed.]* Separate mappings MF/SMCS $\leftrightarrow$ TVCS are needed for each *TV filter*. They may also depend on *temperature*.

[a lot changed above – must be reviewed for consistency later on. Also, TV, FCS figures must be created. And two more figures below.]

2.2 Mathematical Expressions for Mappings

For most of these coordinate systems, the definition of the cardinal axes corresponds to the global coordinate axes of the DEIMOS design (see Figures ?? and ?? for the global and individual DEIMOS' coordinate systems). All x-axes point in a set direction ("up" in Figure ??), normal to the light path, the local z-axis is always in the direction of photon travel, and the y-axes generally point "outboard". However, this last convention is inverted for the ICS and CCDCSi coordinate systems, whose y-axes point *in*board (see Figure 2.3). The ICS is the coordinate system that the observer sees on the quick-look display, where it has (0,0) in the lower left corner with x increasing to the right and y upwards. It is also defined so that positive y points towards *increasing* λ along the spectrum and the display shows a direct view of the sky for direct imaging (see Figure 2.3). These constraints require that the y-axis reverse direction in the ICS and CCDCSi coordinate systems.

2.2.1 FKCS $\rightarrow$ RSCS:

The fundamental sky coordinates can be described in any equinox and epoch. To update and refract to *observed* coordinates on the sky, four transforms are needed, to account for (1) proper motion; (2) precession; (3) aberration of starlight[2]; and (4) atmospheric refraction:

$$(\alpha_\mu, \delta_\mu) = R_1(\alpha_0, \delta_0; \mu_\alpha, \mu_\delta, \Delta Epoch);$$

[2] Aberration of starlight is maximum in directions tangential to motion. However, the differential aberration is greatest in direction of motion, reaching a maximum of $20.5 sin\ \delta$, where δ is the edge-to-edge angle of the field. For DEIMOS, considering both beams and the radius of the FOV, $\delta = \pm 10'$ and the differential aberration can reach ± 0.06 arcsec in virtually any orientation. On the other hand, differential mask alignment between beams A and B can remove a large portion of the effect in the important cross-slit direction.

$$(\alpha_\nu, \delta_\nu) = R_2(\alpha_\mu, \delta_\mu; \Delta Equinox);$$

$$(\alpha, \delta) = R_3(\alpha_\nu, \delta_\nu; LST, Date);$$

$$(\alpha', \delta') = R_4(\alpha, \delta; z, T_{atm}, P_{atm}, \lambda),$$

where (α', δ') are the updated, refracted coordinates of the fundamental sky (α_0, δ_0) at the zenith distance z, wavelength λ, and atmospheric temperature and pressure, T_{atm}, P_{atm}.

These are all standard transformations, and will be taken from the Starlink SLAlib library.

2.2.2 RSCS → MF/SMCS:

This mapping is performed in three stages: first, a gnomonic projection is made of the [refracted] sky; second, an adjustment is made in the radial direction for distortion; and lastly a projection onto the tilted, curved surface of the mask[3]. Due to the format of the distortion mapping from ray-tracing programs, these take the form:

$$(\xi', \eta') = G(\alpha, \delta, \phi_0),$$
$$(x_p, y_p) = D(\xi', \eta'),$$
$$(x_m, y_m) = M(x_p, y_p; \theta_m, R_m),$$

where (x_m, y_m) are the physical coords projected onto the tilted, cylindrical surface of the mask (described by a tilt, θ_m, and a radius of curvature, R_m); (ξ', η') are the standard coordinates produced by a gnomonic projection of (α, δ) at position angle ϕ_0, converted to off-axis angles on the sky. The position angle of the mask, ϕ_0, is defined as the position angle of the x-axis on the sky.

The gnomonic projection is:

$$\xi = \frac{cos\delta\ sin(\alpha - \alpha_0)}{sin\delta\ sin\delta_0 + cos\delta\ cos\delta_0\ cos(\alpha - \alpha_0)}$$

[3] This procedure only works if the mask surface closely approximates the spherical image surface of the telesope. For differences h between the two surfaces, translational errors can reach $h\ sin(1.1°)$. At the extreme ends of the mask, the curvature reaches a difference of 1.3 mm ($\Delta = 0.034$ arcsec); the amplitude of the deviation between the cylindrical and spherical surfaces is ± 1.4 mm ($\Delta = 0.037$ arcsec) along the "flat" axis. This latter is the more critical, as it is in the "cross-slit" direction.

$$\eta = \frac{sin\delta\ cos\delta_0 - cos\delta\ sin\delta_0\ cos(\alpha - \alpha_0)}{sin\delta\ sin\delta_0 + cos\delta\ cos\delta_0\ cos(\alpha - \alpha_0)}$$

These are tangents; rotation so that η' is parallel to the y-axis and converting to angles:

$$\xi' = atan(\ sin\phi_0\ \xi + cos\phi_0\ \eta),$$

$$\eta' = atan(-cos\phi_0\ \xi + sin\phi_0\ \eta),$$

where ϕ_0 is the position angle of the x-axis on the sky.

The mapping from the off-axis angles to the projected physical coordinates, $(x_p, y_p) = D(\xi', \eta')$, is determined via ray-tracing[4]. It is best determined using image centroids rather than principle rays.

The mapping from the projected physical coordinates to the tilted, cylindrical surface is:

$$x_m = R_m\ \mu$$

$$y_m = \frac{(y_p - 135\text{mm})}{cos\theta_m} + R_m\ tan\theta_m\ [1 - cos\mu]$$

$$\mu = \frac{x_m}{R_m} = arcsin\frac{x_p}{R_m}$$

The value 135mm above is a zeropoint offset, that is, the value of y_p where $y_m = 0$ and $x_p = x_m = 0$.

In addition, the tilted and curved surface produces a modification to position angle ϕ with respect to the x-axis ($\Delta\phi = \phi_{slit} - \phi_0$), given by

$$tan\Delta\phi_m = tan\Delta\phi\ (\frac{cos\mu}{cos\theta_m}) + tan\theta_m\ (\frac{x_p}{R_m})$$

[4]This ray-tracing depends on the telescope focal length, f_{tel}, and thus is only valid for one value of the focus. However, f_{tel} appears to be quite stable — the primary's focal length cannot be changed simply by tilting the segments, and the segments and secondary are all in "zerodur" and so should not change shape (T. Mast, priv. comm.)

2.2.3 MF/SMCS → ICS:

This mapping is performed in three stages: first, a projection from the tilted, curved slitmask to a plane; secondly, conversion into input angles at the grating/mirror; then mapping with the camera distortions to the camera focus.

$$(x_p, y_p) = P(x_m, y_m; \theta_m, R_m),$$
$$(\alpha, \gamma) = A(x_p, y_p, \mu_{grat}; d_1, d_2, d_3, R_s, R_c)$$
$$(x', y') = C_{grat}(\alpha, \gamma; f_{cam}(T)^5, cam.distortion),$$

where d_1, d_2, d_3, R_s and R_c are various fixed pupil distances and radii (see Figure 2.4).

The first mapping is a reverse of the earlier projection:

$$x_p = R_m\ sin\mu$$

$$y_p = cos\theta_m\ (y_m - R_m\ tan\theta_m\ [1 - cos\mu]) + 135\text{mm}$$

$$\mu = \frac{x_m}{R_m}$$

The second mapping to input angles depends on the axial radii and heights, both at the image surface (*i.e.*, slitmask) and the collimator.

$$\rho_s = \sqrt{x_p^2 + y_p^2}$$

$$\zeta_x = arctan(\frac{x_p\ (d_2 - h_c)/(d_1 - h_s)}{(d_3 - h_c)})$$

$$\zeta_y = arctan(\frac{y_p\ (d_2 - h_c)/(d_1 - h_s)}{(d_3 - h_c)})$$

where h_s and h_c are the heights of the image and collimator surfaces:

$$h_s = R_s\ (1 - cos[arcsin\frac{\rho_s}{R_s}]) = R_s - \sqrt{R_s^2 - \rho_s^2}$$

[5] NB: indications are that the camera focus, f_{cam}, is very sensitive to the camera temperature, producing a scale change of ??? / C — hence the explicit presence of T as a variable in the mapping.

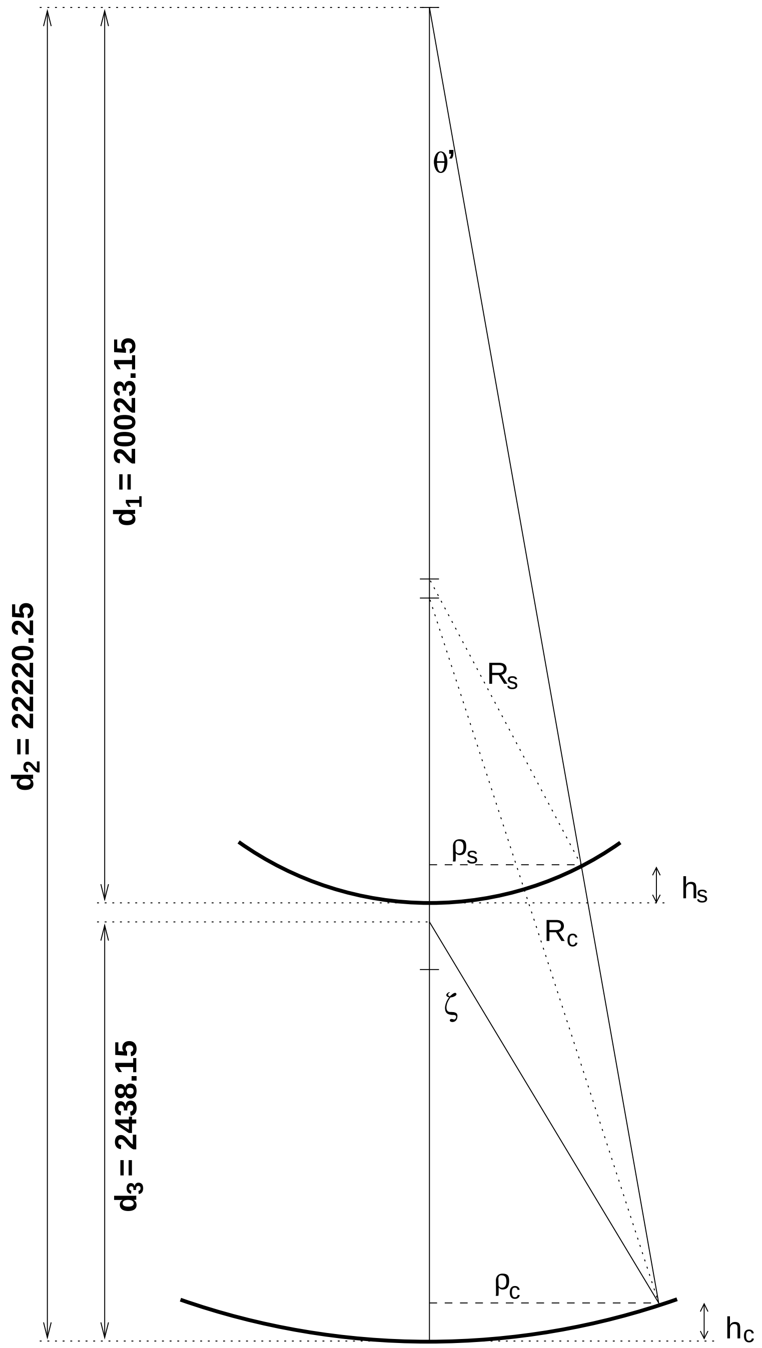


Figure 2.4: Schematic of rays, telescope through collimator

$$h_c \simeq R_c\ (1 - cos[arcsin\frac{\rho_s\ d_2/d_1}{R_c}]) = R_c - \sqrt{R_c^2 - (\rho_s\ d_2/d_1)^2}$$

For rough estimates, h_s and h_c can be ignored. In practice, the approximation for h_c is probably sufficiently adequate, although for highest precision *and* computational speed, it is probably best to provide a table of precise $h_c(\rho_s)$ values and interpolate for specific ρ_s.

The axial angles ζ_x, ζ_y are converted into incident angles at the grating by

$$\alpha = \zeta_y + 2\tau - \mu_{grat}$$

$$\gamma = \zeta_x$$

where τ is the tent mirror angle and μ_{grat} the angle of the grating normal, both with respect to the optical axis.

The third mapping is straightforward, although it includes factors for camera distortion which are available from ray-tracing (but may need to be determined empirically?). In the following, the angles and their sense are given in Figure 2.5.

$$x_i = f_{cam}\ tan(\gamma)\ f(cam.dist : \zeta)$$

$$y_i = f_{cam}\ tan(\beta - \beta_0)\ f(cam.dist : \zeta)$$

$$\beta_0 = \chi - \mu_{grat}$$

where χ is the camera angle with respect to the collimator optical axis. Note that x_i, y_i are measured with respect to the optical axis of the camera.

With the mirror in place, incident and reflected angles are equal in magnitude but opposite in sense:

$$\beta = -\alpha$$

With a grating in place:

$$\lambda = \frac{n\sigma}{m}\ cos\gamma\ (sin\beta + sin\alpha)$$

where m is the order, usually first, n is the index of refraction of air, and σ is the spacing of rulings on the grating.

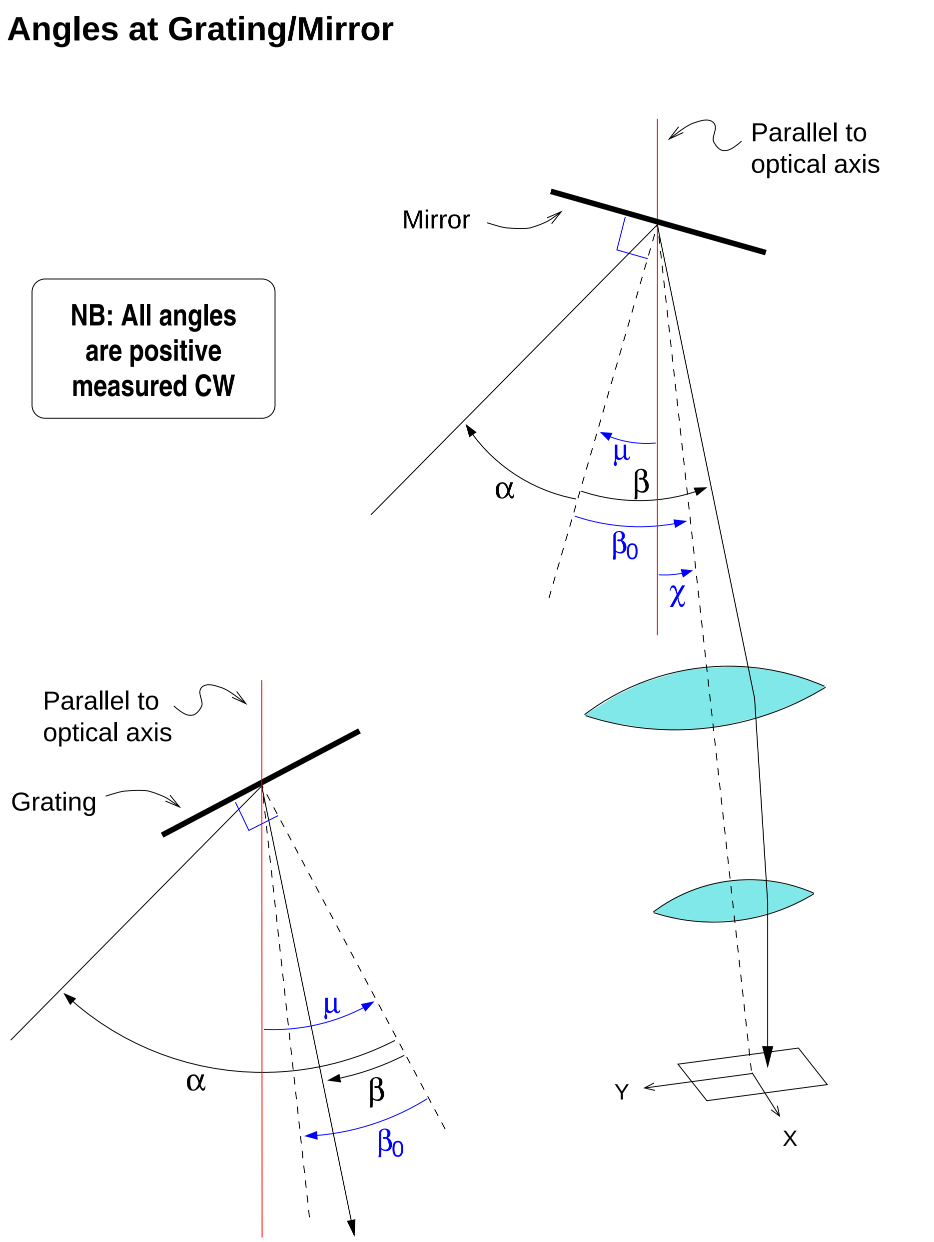


Figure 2.5: Schematic of rays, grating to detector

Note that the third mapping is usually needed in reverse, in order to estimate a λ for a given pixel location:

$$\beta = arctan[\frac{y_i}{f_{cam}}\ (distort.factor)] + \beta_0$$

$$\gamma = arctan[\frac{x_i}{f_{cam}}\ (distort.factor)]$$

2.2.4 ICS $\rightarrow$ CCDCSi:

This mapping involves rotation and translation constants for each of the eight CCDs in the mosaic. CCDCSi will generally be expressed in pixels.

$$x(\mathrm{CCDCS}i) = cos\phi_i\ x_c - sin\phi_i\ y_c + \Delta x_i$$

$$y(\mathrm{CCDCS}i) = sin\phi_i\ x_c + cos\phi_i\ y_c + \Delta y_i$$

where ϕ_i is the rotation and $\Delta x_i, \Delta y_i$ are the translations for each CCD with respect to the ICS. Note that even the fiducial CCD can have a rotation and offset with respect to the ICS, which by definition has axes parallel to those of the MFCS.

2.2.5 MF/SMCS $\rightarrow$ TVCS:

[Not yet described.]

2.2.6 Constants:

Engineering drawings currently give the following values:

R_m	2071.88 mm	Sutin (CDR)
θ_m	6.0°	Sutin (CDR)
d_1	20023.15 mm	Drawing D0023.D
d_2	22220.25 mm	Drawing D0023.D
d_3	2438.15 mm	Drawing D0023.D
R_s	2123.57 mm	Drawing D0106.F
R_c	4394.20 mm	Drawing D0106.F
τ	18.5°	Drawing D0106.F
χ	−2.33°	Drawing D0106.F
f_{cam}	(381) mm	??

The following grating parameters will also be of interest when discussing vignetting:

size	ruled-x	ruled-y	y-center
6 × 8	6.102 in	8.268 in	4.134 in
	155 mm	210 mm	105 mm
8 × 12	8.267 in	12.206 in	4.818 in
	210 mm	310 mm	122.4 mm

2.3 Empirical Mappings

[We need to make some direct mappings between non-adjacent coordinate systems, as in astrometry off direct images, and the definitions from the theoretical mappings above do not describe these well. Furthermore, these mappings must be made empirically, although the analytic mappings may be used in many cases to provide estimates of their form and scales. These are the mappings that will be discussed:

ICS ↔ CCDCDi. Although discussed above, we must measure the actual $\Delta x_i, \Delta y_i, \phi_i$ empircally.

ICS → FKCS. This is astrometry, going directly from the image to the fundamental celestial coordinates, via astrometric references in the field.

ICS → RSCS. Same as above, but two or less astrometric references so only refracted coordinates are possible.

RSCS → SMCS. The analytical mapping should work well, but can likely be improved through a differential mapping — this should be discussed here.]

Chapter 3

Calibration Elements

Note: This chapter is probably of limited interest to the reader, as it deals with the theory rather than practice of calibrations. It is intended as a "proof-of-concept-in-theory." The next chapters will deal with specific procedures for calibrating different types of data under normal operation.

"Calibration elements" refers to separable units in the overall calibration of the instrument. There are three categories of these: (1) detector calibration; (2) spatial calibration, including wavelength; and (3) flux calibration. These may be thought of as breaking the instrument into a detector and a spectrograph/camera. For each, we wish to provide mappings between input information and observed information, where the information is either spatial or intensities. The spatial mapping of the CCD itself is assumed to be known.

The goal of the current chapter is to identify the separable elements in each of these categories, understand their dependencies, and identify how they can be characterized. The need for this is driven by several questions:

- feasibility — can we calibrate DEIMOS?
- what level of accuracy can we expect?
- how can we calibrate DEIMOS data under baseline operation?
- for pipeline reduction, what sort of information needs to be available in the database?

We wish to identify two parallel paths for deriving and applying calibrations for data reduction. The first is baseline mode — this is the traditional method, where the user acquires *all* necessary supporting calibration data and which assumes neither information about nor understanding of the instrument. The other path is a completely automated reduction pipeline, where *no* information is needed for reduction other than what is contained in the FITS header for a given image. In practice, neither of these paths will be entirely applicable. A *completely* automated pipeline will probably not provide the necessary accuracy for wavelength calibration and distortion correction. On the other hand, the amount of calibration data needed for slitmask spectroscopy is prohibitive for a single user to acquire (not to mention that instrumentation failure and weather sometimes prevent its acquisition). Therefore, in practice certain elements will be borrowed from the database and others will come from the science images themselves. It will be a user option to decide how many supporting calibrations will be acquired for the user's particular data set.

3.1 Detector Calibration

These elements refer to individual CCDs.

3.1.1 Bias-Level:

This is easily determined from the overscan region of an image.

3.1.2 Bias-Structure:

This is easily determined by averaging enough bias frames to overcome the readout noise.

3.1.3 Dark Rate:

This is easily determined by averaging enough long-exposure dark frames to overcome the readout noise.

3.1.4 Pixel-to-Pixel Flats:

Pixel-to-pixel variations in the CCDs can be estimated easily by smoothing a flat-field frame by a robust smoothing filter of fairly large pixel scale, and dividing the original image by the smoothed image. *[TBD: decide on smoothing filters – probably a combination of spatial medians and gaussians – and on the smoothing length.]*

It is possible that pixel-to-pixel variations are not highly λ-dependent. Although it is well-known that there are spatial variations in DQE with color, these are believed due to variations in the thinning, and the spatial scale for these variations should be larger than several pixels. If this is the case, only *one* high S/N flat need be obtained for the pixel-to-pixel flat. Otherwise, high S/N flats will need to be acquired in several filters and/or grating tilts and interpolated to specific observed wavelengths during pipeline reductions.

[Recommended action item: explore color (in)dependence of pixel-to-pixel variations, using LRIS and/or KAST. Eventually can also be tested in DEIMOS in lab.]

3.1.5 Large-scale Flats:

These remove vignetting, and are discussed below in flux-calibration.

3.1.6 Fringing:

Fringing is caused by internal reflections in the CCD, and depends on variations of the thickness of the CCD, its transparency at a given wavelength, and the angles of illumination. As fringing appears to be the result of relatively high order ($n \sim 100$) interference, it is not feasible to calibrate the variations in thickness directly. Also, the changing illumination may cause a phase shift or changes in the fringe amplitude. For these reasons, the most direct method to remove fringes is to take a flat-field through the instrument with the *identical setup* as the science observations, that is, same slitmask, grating tilt, rotator angle, and divide by this flatfield.

Since a slitlet's position affects the wavelength scale, for automated removal of fringing we must treat each slitlet individually. If we have a series of spectral flats at various grating tilts *[TDB: determine needed sampling interval]* we should be able to choose a flat at similar wavelength to the science observation, and solve for a best-fit *phase offset* using bright night-sky lines. Note that this requires a special long-slit mask that covers the entire mask length. Note also that this scheme will fail if variations in vignetting produce a significant change a fringe amplitude.

[Recommended action item: explore behavior of fringing with LRIS and/or KAST, particularly with changes in rotator angle in LRIS.]

3.1.7 Cosmic Rays:

This is not technically a calibration, nor are there any calibrating database products, but we list CR-removal here as it is specifically a detector issue. CR-removal is a problem without easy solution, although the relatively small sampling scales of DEIMOS will help in discriminating between CRs and real features. [This is currently an unresolved issue.]

3.2 Spatial Calibrations

3.2.1 CCD Spatial Calibration

We assume the CCD spatial mapping is known (square grid, 15 μm square pixels).

3.2.2 Instrument Spatial Calibration

This refers to astrometric mappings. These are available in approximate form as CCDCSi $\rightarrow$ ICS $\rightarrow$ FKCS mappings (see Chapter 2). In automated mode, the limiting accuracy will be set by the telescope pointing [$\pm$10 arcsec], although the internal relative accuracy will be much higher [$\pm$1 arcsec?]. User identification of a single astrometric reference star on an image should bring absolute and relative accuracy to the same level. This should be adequate for quick-look reductions.

For astrometry adequate for slitmask design, three or more astrometric reference stars distributed across an image will be required. For spectroscopic reductions (*i.e.*, where to locate objects in slits), two or more objects with known coordinates will probably be required to set the spatial scale.

3.2.3 Wavelength Calibration

Approximate wavelength calibration is available from the mapping of the MF/SMCS to ICS. In practice, precise adjustments will be needed to provide calibrations of sufficient quality for astronomical work. This is necessitated by inaccuracies and unknowns in the system — for example, measurement errors in mapping camera distortions, flexure, irregularities in the slit structure within the mill tolerances, etc.

Strong night sky lines can usually be used for wavelength calibration adjustments after the wavelength mapping is performed. These adjustments may take the form of a simple zeropoint correction (one nightsky line needed), or full wavelength solution (several lines). The modeled wavelength mapping should be sufficient that strong night-sky lines can be identified without visual inspection by the user.

3.3 System Throughput & Instrumental Signature

We identify the *efficiencies* of various components, and their functional dependence:

telescope: $\epsilon_{tel}(\theta, \lambda)$ describes both the radially-symmetric vignetting (θ) and reflectance[1] (λ) of the telescope. For DEIMOS, there is no telescope vignetting over the FOV ($10'$ radius), so we may simplify to $\epsilon_{tel}(\lambda)$.

collimator: $\epsilon_{coll}(\theta, \lambda)$ describes both the radially-symmetric vignetting (θ) and reflectance (λ) of the collimator. For DEIMOS, there is no collimator vignetting over the FOV, so we may simplify to $\epsilon_{coll}(\lambda)$.

tent mirror: $\epsilon_{tent}(\theta, \phi, \lambda)$ describes both the vignetting (which does not have radial symmetry, and so depends on θ, ϕ) and reflectance (λ) of the tent mirror. We can separate these into

$$\epsilon_{tent}(\theta, \phi, \lambda) = \epsilon^V_{tent}(\theta, \phi)\ \epsilon^R_{tent}(\lambda),$$

where the superscripts V and R denote vignetting and reflectance, respectively.

grating/mirror: $\epsilon_{grat}(\alpha, \beta, \delta, \lambda, \Phi)$ describes the grating efficiency and vignetting. Vignetting depends on the pupil image orientation, here denoted Φ, and potentially on the grating input angle (α). Grating efficiency depends on the blaze angle, denoted δ, input and output angles (α, β), and grating reflectance (λ). The theoretical efficiency for 100% reflectance, or "blaze function", is given by:

$$BF = \frac{sin^2\gamma}{\gamma^2},$$

where

$$\gamma = \frac{\pi\sigma cos\delta}{\lambda}[sin(\beta - \delta) + sin(\alpha - \delta)]$$

or

$$\gamma = m\pi cos\delta\frac{[sin(\beta - \delta) + sin(\alpha - \delta)]}{[sin\beta + sin\alpha]}.$$

[1]Reflectance for a given material depends not only on the wavelength of light, but also the angle of incidence. However, for small variations in incidence angle there is little variation in reflectance, and we ignore this slight dependency throughout this discussion.

Note that the theoretical blaze function can be written without an implicit λ-dependence. We retain λ in the blaze function, as there are sometimes wavelength-dependent deviations from the theoretical relation. We separate the grating efficiency into:

$$\epsilon_{grat} = BF(\alpha, \beta, \delta; \lambda)\ \epsilon^{R}_{grat}(\lambda)\ \epsilon^{V}_{grat}(\alpha, \Phi)$$

Note that "grating" can also refer to the imaging mirror, in which case there is only the reflectance λ-dependence, $\epsilon^{R}_{mirr}(\lambda)$.

camera: $\epsilon_{cam}(\alpha, \mu, \beta, \gamma, \lambda)$ describes the camera vignetting as a function of camera input angles (β, γ) pupil axial ratio (here characterized by incidence angle α) *and* its projection seen by the camera (μ), and possibly wavelength (λ), and the transmission of the optics (λ). The possible λ-dependence of vignetting is based on the idea that rays of different color can have different paths through the camera. In DEIMOS, there is no vignetting beyond the camera mouth, so there should be no λ-dependent vignetting. Also, the vignetting should be circularly symmetric about the optical axis of the camera, so that a single radius based on angles $\beta - \beta_0, \gamma$ should describe it, but for discussion it is better to leave it as shown. We can separate the camera efficiency:

$$\epsilon_{cam} = \epsilon^{V}_{cam}(\alpha, \beta, \gamma)\ \epsilon^{T}_{cam}(\lambda)$$

where the superscripts V and T denote vignetting and transmission, respectively. Note that there is no camera vignetting in imaging mode.

filter: $\epsilon_{filt}(\beta, \gamma, \lambda)$ describes vignetting by the filter (β, γ) and the filter transmission curve (λ). Similarly to the camera, we write

$$\epsilon_{filt} = \epsilon^{V}_{filt}(\beta, \gamma)\ \epsilon^{T}_{filt}(\lambda).$$

detector: $\epsilon_{CCD}(x_i, y_i, \lambda_i)$ is the color-dependent (λ) detector quantum efficiency (DQE) at pixel i. We assume no fringing; this is described elsewhere.

The simplification and separation of the various efficiencies above allow us to rewrite all the efficiencies in a more simplified way. Notice that the position on the tent mirror (θ, ϕ) maps uniquely into input angles at the grating (α, γ) and so we substitute those angles. Similarly, x, y at the deltector map onto β, γ. We leave out the detector efficiency for the time being. Furthermore, we leave the grating and filters separate, as they are changeable. This gives us:

$\epsilon_{sys}(\lambda) = \epsilon_{tel}(\lambda)\ \epsilon_{coll}(\lambda)\ \epsilon^{R}_{tent}(\lambda)\ \epsilon^{T}_{cam}(\lambda)$

$\epsilon^{T}_{filt}(\lambda)$

$\epsilon^{R}_{grat}(\lambda)$

$BF(\alpha, \beta, \delta; \lambda)$

$\epsilon^{V}_{tent}(\alpha, \gamma)$

$\epsilon^{V}_{cam}(\alpha, \mu, \beta, \gamma)$

$\epsilon^{V}_{filt}(\beta, \gamma)$

$\epsilon^{V}_{grat}(\alpha, \Phi)$

$\epsilon_{CCD}(\gamma_i, \beta_i, \lambda_i)$

The product of the first four factors is the **system throughput** (excluding the detector). There is a λ-dependence, and a weak positional dependence in the blaze function (ignoring this dependence introduces tilts in the spectra of usually less than 10%, end-to-end).

The product of the next four factors describe the vignetting, and are wavelength independent *but do depend on grating and tilt.*

The list above describes the general situation, as found in spectral mode. It is not immediately obvious that we can calibrate this data without observing flux standards in every slit, as there are many interdependencies that must be sampled. It is instructive to consider how things look for imaging mode, where $BF = \epsilon^{V}_{cam} = \epsilon^{V}_{grat} = 1$:

$\epsilon_{sys}(\lambda) = \epsilon_{tel}(\lambda)\ \epsilon_{coll}(\lambda)\ \epsilon^{R}_{tent}(\lambda)\ \epsilon^{T}_{cam}(\lambda)$

$\epsilon^{T}_{filt}(\lambda)$

$\epsilon^{R}_{mirr}(\lambda)$

$\epsilon^{V}_{tent}(\alpha, \gamma)$

$\epsilon^{V}_{filt}(\beta, \gamma)$

$\epsilon_{CCD}(\gamma_i, \beta_i, \lambda_i)$

Usually, the term ϵ^V_{filt} will also be unity — it is left here explicitly for the possibility of future, undersized filters. Given filters without vignetting, it is possible to solve for ϵ^V_{tent} (and also ϵ^V_{filt} for any undersized filters). This considerably simplifies the general case, leaving only $\epsilon^V_{grat}(\alpha, \Phi)$ as an unknown vignetting factor. Note that the Φ dependence is at most a $\pm 1\%$ effect. Furthermore, there is a dependence on α only for the 1200-line grating, and we can remove this by normalizing at a particular wavelength. With this normalization, and ignoring the variations with Φ, the total grating vignetting can be reduced to a constant factor that can be removed during system throughput observations with a standard star. This removes instrumental signature to approximately $\pm 1\%$.

In spectral mode, however, we must consider that $\epsilon^V_{cam} \neq 1$, but has an $\mu, \alpha, \beta, \gamma$ dependence. Note, however, that this vignetting is independent of wavelength — this means that it can be mapped out easily, for a given α, μ, by continuum spectra observed through a line-of-holes mask. A series of such exposures for each grating and at different tilts would be sufficient to map out this vignetting. Note that there will be some interplay between the tent mirror vignetting, grating vignetting and camera vignetting — that is, some shadowed regions are in common, which results in less overall vignetting in those regions.

We now consider the system throughput, which is measured by comparing spectral counts in standard star spectra with known fluxes. We assume observations of standard stars for a particular grating tilt but positioned at the extreme slit positions, so that the entire spectral range for all slits is encompassed. Ignoring the positional dependence of the blaze function leads to an error of up to $\pm 5\%$ under usual circumstances. Removing a theoretical blaze function should reduce this by a factor of several.

In summary, it appears that DEIMOS slitmask spectral data can be relative-flux-calibrated in a pipline mode, at least to levels of $\pm 5\%$ and probably to ± 1–2%. The most difficult problem is vignetting at the grating and camera mouth, especially as the grating and tent mirror vignetting has some overlapping shadowed regions. The most reliable way to handle this is empirically — that is, through a series of calibration observations, such as a series of dome flats through a set of "line-of-holes" masks, observed on a grid of grating tilts.

However, such calibration observations raise a difficulty of their own. Variations in the light source can be removed to first order by normalizing to a central slit. However, variations in the *spectral shape* of the illuminating source cannot be internally calibrated. We therefore require a lamp that has a constant spectral shape over the course of the calibration sequence for each grating. Certain high-pressure discharge tubes may be able to provide this stability. Without such a source, we would need to observe standard stars at each normalizing slit location and grating tilt, which is probably prohibitive in terms of observing time. *[If we adopt eight LOH masks, N grating tilts, 5 gratings, this requires 40N standard star observations.]*

Chapter 4

Overview of Data Procedures & Database Products

The following four chapters provide detailed information about calibrations required to characterize DEIMOS at commissioning, procedures needed to calibrate DEIMOS science data, products that will be maintained in the DEIMOS database, and instructions for creating those products. These chapters cover:

Engineering Calibrations: Those calibrations needed at commissioning, which must be performed prior to science observations: slitmask astrometry, TV astrometry, determining relative alignment of CCDs.

Detector Calibrations: Those steps needed to characterize and remove the signature of the detector and electronics, independent of the light falling on the detector: bias level, bias structure, dark counts, CR-cleaning.

Imaging Calibrations: Those steps needed to characterize and remove the signature of the total system in direct imaging mode: flat-fielding, flux calibration, geometric distortions.

Spectral Calibrations: Those steps needed to characterize and remove the signature of the total system in spectroscopic mode: flat-fielding, geometric distortions, wavelength calibration, flux calibration.

Engineering calibrations are of little direct interest to the user, but are of considerable importance to the acquisition of high-quality data. The remaining three types of calibrations are referred to as **science calibrations**, and must be carried out by the user or a data-reduction pipeline. A flowchart of these operations is shown in Figure ??, and a summary of the scince calibration steps/elements is given in the Table. *[Figure to be finished...]*

Each of the following chapters is organized to provide a description of terms, background information and relevant concerns regarding certain processes, and a brief description of each calibration step. This is followed by a summary of necessary database entries and instructions for how each will be created (the instructions will eventually be turned into coded instrument-control scripts which can be run with little input from the instrument specialist responsible for maintaining the calibration database). Finally, outstanding issues are listed at the end of each chapter.

All calibration procedures and products are required for **pipeline mode**. We identify two levels of pipeline mode, **Phase 1** and **Phase 2**, as well as a pre-pipeline, **Baseline** mode. Baseline mode will be delivered at commissioning. We hope to deliver Phase 1 pipeline within one year of commissioning. Phase 2 is somewhat dependent upon the performance of the Phase 1 pipeline and the instrument, and its implementation is uncertain. Since Baseline mode requirements are a subset of Phase 1 pipeline mode requirements, we describe the calibration procedures and database products from the perspective of pipeline mode; variations for baseline mode will be noted as needed. The instructions for creating database entities can be used by observers to obtain necessary calibration images in baseline mode.

4.1 Calibration Slitmasks

The calibration of certain properties require special slitmasks. These are described below, and their use is detailed in the following chapters.

Grid-of-holes (GOH) mask: a rectangular grid of very small holes at spacings of 1 cm (??) along grid patterns at 5 cm (??) for a total of (??) holes *[TBD]*.

Line-of-holes (LOH) mask: a line of rectangular apertures cut on a regular grid at 10 arcsec spacing. The apertures should be wide (3 arcsec in y) and fairly short (1.5 arcsec in x) so that small variations in slit area will not have a significant effect on the flux ($<< 1\%$). (The x-length will be adjusted to account for the mask curvature and thickness of the mask stock, so the effective aperture remains constant across the mask.)

Feature	Type	Operation	Source	Other input	Notes
		Detector Calibrations			
Bias Level	const	subtract	int		or 1-D fn
Bias Structure	image	subtract	DB		bias-level subtr. must match
Dark Structure	image	subtract	DB	t_{exp}	dark scaled by exposure
CR removal	proc	—	–		params set by CR stats
		Imaging Calibrations			
Flat field	image	divide	DB		per filter.
Fringing	image	subtract	DB/int		per filter. Amplitude scaled to fit fringes in sky
Sky subtract	const	subtract	int		
Distortion flat	image	multiply	DB		compensate for diff. pixel scale
Atm. extinction	const	multiply	DB	airmass	per filter. Average for MK
System efficiency	const	multiply	DB	t_{exp}	per filter. Average for sys
Astrometry	proc	—	DB		apply distortion mapping
		Spectroscopic Calibrations			
Px-px flat field	image	divide	DB		interpolate DB images?
Rectify-spatial	proc	—	int/DB	blueprint	
Wavelength cal.	proc	—	int/DB	blueprint	per mask. Use NS lines
Slit illum.	1-D	—	–	blueprint	per mask. User flat
Sky subtract	proc	—	int/DB	blueprint	per mask. May use fringing data from DB (TBD)
rel. flux cal. 1	TDB	—	TBD	TDB	between slits
rel. flux cal. 2	1-D	—	TBD	TDB	along spectra
Atm. extinction	1-D	multiply	DB	airmass	Average MK extn. spectrum
abs. flux cal.	const	—	DB?	exposure	Avg. for system

NOTES — Source may be internal, that is, from a particular science image, or database (DB). Other inputs refer to additional information needed in a particular operation (often a description of the mask via a mask "blueprint").

Line-of-slitlets (LOS) masks: Set of masks, with line of slitlets at y-spacings 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 7.0 and 8.0 arcmin from the telescope axis (8 in total). Each LOS consists of a line of 100 rectangular slitlets each 1×7 arcsec, separated by 3 arcsec, across the x-range of ± 8.5 arcmin.

In addition, there will be a number of slitmasks of astrometric fields used in slitmask astrometry.

Chapter 5

Calibrations at Commissioning

Prior to and during commissioning, various calibrations must be performed before the instrument is ready to produce science data that can be readily interpreted. These calibrations concern the geometrical mappings between coordinate systems described in Chapter 2. They are:

Slitmask Astrometry, or mapping the sky (RSCS) onto the slitmask (SMCS);

TV Astrometry, or mapping the sky (RSCS) onto the TV system (TVCS); and

CCD alignment, or measuring the rotations and translations of the individual CCDs in the mosaic (CCDCSi) relative to the Image Coordinate System (ICS).

This does not describe all mappings. Certain mappings (FKCS to RSCS) are handled purely through modelling. Others will be handled during observing (SMCS to MFCS) or determined empirically during science calibrations (SM/MFCS to ICS; ICS to RSCS).

5.1 Background and Issues of Concern

(nothing major)

5.2 Database Products

The slitmask astrometry mapping will be maintained in the database.

The TV astrometry mapping will be maintained in the database.
The CCDCSi $\leftrightarrow$ ICS mapping will be maintained in the database.

Entry	Type	Depend.	Descrip.	Deliv.	Maintenance	Notes
Slitmask Astrometry	2D func	–		0	Tel. mods	difference w/ model
TV Astrometry	2D func	–		0	TV mods	
CCDCSi mappings	8×3 const	per CCD	x_i, y_i, θ_i	0	Mosaic mods	relative to ICS

NOTES — Delivery: 0=Commissioning, 1=Pipeline.

Required Hardware:

– Mask form fiducial marks

– Slitmask of astrometric field, based on expected projections.

– Grid-of-holes (GOH) mask

– Line-of-holes (LOH) mask

5.2.1 To create a slitmask astrometry map:

1. Select astrometric field; prepare a slitmask using existing slitmask mapping (pure model initially). This slitmask will have small pinholes at the expected position of selected stars.

2. Align mask with the selected field; take one or more images of the mask.

3. Remove mask; take one or more images of the field; process all images through relative flux calibration.

4. Measure the pixel positions of each selected star and its corresponding pinhole.

5. Convert the differences between each coordinate pair into mm on the slitmask. (Detailed mapping not necessary in this conversion.) Fit the mm offsets as a function of slitmask location. This becomes the differential update to the slitmask astrometry model.

5.2.2 To determine SM/MFCS to TVCS mapping:

1. Take TV exposure showing mask form fiducials and fixed mirror.
2. Analyse location of maskform fiducials to solve for scale, rotation and translation of TVCS wrt MFCS.
3. Use slitmask astrometry mapping to relate this to the refracted sky.
4. Mapping can be verified against an astrometric field.

Note: the above assumes negligible distortion in the TV camera. If this is not valid, observations of astrometric fields must be used to map out the distortion. The level of distortion can be tested (pre-commissioning) in the lab, using the image of a grid-of-holes mask. The level of telescope distortion (also a potential factor) is ?? (negligible) over the FOV of the TV.

5.2.3 To determine relative alignment of CCDs:

1. Take a direct image (no slitmask) of flat field. Measure the location of the mask form fiducials. Using the known location of the fiducials wrt the telescope/collimator optical axis, solve for the rotation, scale and approximate zeropoint offsets for the fiducial CCD.
2. Take a flat field image through the grid-of-hole (GOH) mask. Measure the locations of the holes on each mask and solve for a rotation and translation relative to the fiducial CCD. Convert to ICS using the transformation above.
3. Take a spectral flat-field exposure through the line-of-holes (LOH) mask. Measure the spectral traces across the CCDs. The *change* in slope[1] characterizes the red vs blue CCD rotation. This rotation can be related to the fiducial CCD and thus the ICS.
4. Take a spectral arc exposure through the line-of-holes (LOH) mask. Using the traces above, locate and measure the emission line (spots) in the spectrum from each hole. These may be used to determine a δx,δy across each CCD boundary. In conjunction with the angle, these deltas provide the translations for the red CCD relative to the blue CCDs, which in turn are known relative to the ICS.

[1] The slope itself will contain a pitch error for each grating.

5.3 Other Products

5.4 Outstanding Issues

We have not addressed the FCCSj yet.

Chapter 6

CCDs — Procedures & Database Products

We define three areas of a CCD image: the **prescan columns**, actual CCD columns which are masked from light; the **"active area"** that represents measurement of the charge in each pixel of the CCD, and the **overscan** columns that are produced by continuing to read the serial register after the last "active" pixel in each column is measured. While the storage location of prescan and overscan columns may vary and may depend on one-or-two amplifier readout, their conceptual layout is shown in Figure 6.1.

This chapter describes these elements:

Bias level: An additive value representing the "zero level" in the counts as a function of row number. Sometimes called the DC offset. In general a one-dimensional function determined in the overscan region (sometimes takes the form of a constant); it is *subtracted* from all pixels in a given row read out through a particular amplifier.

Trim: Rewrite the image to include only the region of interest, usually the optically active area as opposed to (a) the prescan/overscan columns, or (b) regions of the physical CCD that are in shadow. Not a calibration, but usually performed while subtracting the bias level.

Bias structure: Image of structure left in a bias image (ie., a zero-second dark exposure) following the bias-level subtraction. This structure image is subtracted from other images. Note that the bias structure image must be produced using the same bias-level determination/subtraction as for other images. Usually neglected if there are no strong bias-structure featues.

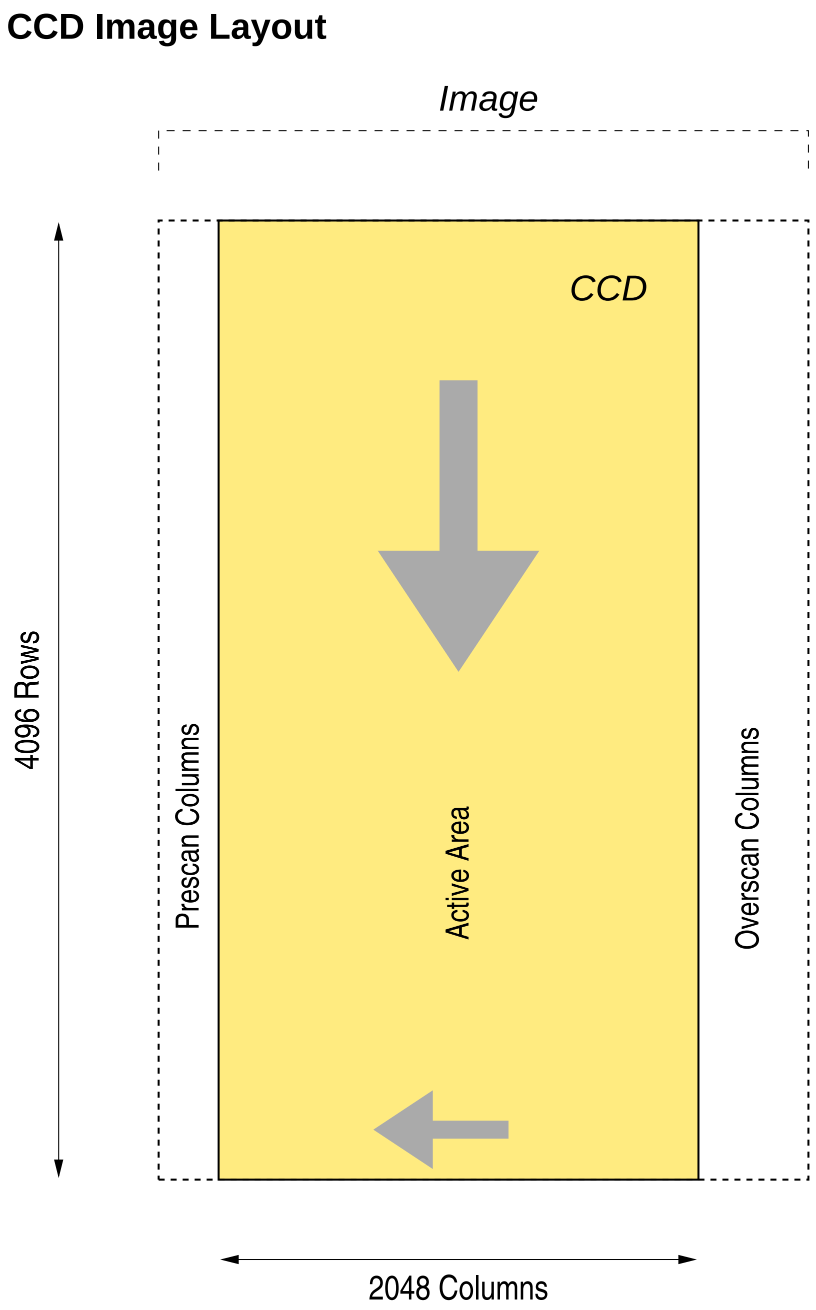


Figure 6.1: Image Layout (one-amp mode)

Dark structure: Rate of dark count accumulation, per pixel. This image must be scaled to other images by exposure time. Usually neglected, as the dark rate is low for most pixels.

CR-cleaning: Procedures for identifying and interpolating over radiation events or "cosmic rays." In practice, something of a black art, and the exact procedure depends on imaging vs. spectral data and single vs. multiple images of the same targets. As CRs are usually identified by their appearance, a set of statistics characterizing typical CR images is helpful.

6.1 Background and Issues

With the exception of CR-cleaning, the procedures described below are all well understood and familiar to observers. The properties of the DEIMOS science CCDs are not yet known, and may potentially require some modifications to standard practice.

[Expected CCD properties can be found in Detector and Software CDR docs.]

[Eventually this section may include long discussion of CR-cleaning.]

6.2 Procedures

Bias level correction and trimming: In individual images, median filter N columns of the overscan region. Fit resultant 1-D image with a low-order polynomial function of row number. Subtract function from each row of the image. Discard prescan and overscan regions (trim).

Bias structure correction: Retrieve bias structure image from database; subtract.

Dark structure correction: Retrieve dark structure image from database; scale it to match the exposure (or "darktime"); subtract.

CR cleaning: (TBD)

6.3 Database Products

A library of standard bias-structure and dark frames will be maintained for each CCD.

Entry	Type	Depend.	Descrip.	Deliv.	Maintenance	Notes
Bias Structure	8 images	per CCD		1	Elec. mods	
Dark Structure	8 images	per CCD, T		1	Elec.,dewar mods	
CR statistics	??	??		1	??	

NOTES — Delivery: 0=Commissioning, 1=Pipeline.

Required Hardware: (none)

6.3.1 To create a bias-structure image:

1. Take N bias frames (zero-exposure dark images).
2. Combine by taking the median level at each pixel (or use some other robust level determinator that rejects cosmic rays). Note that CRs are rare in these images.
3. Bias-level correct and trim the combined image.

(4.) Depending on resultant structure, various smoothing and/or modeling operations can be used to improve S/N. For example, an odd/even pattern in the bias structure can be improved by replacing individual values in each column by the average over the entire column.

The value of N should be large enough that the noise in the combined image is insignificant compared to the read-out noise in a single image. $N \sim 25$–100 is generally appropriate. Time to acquire: $N \times t_{overhead}$; for $N = 50$ and 2 min erase/read time overhead, this is about 1.7 hours for a sequence.

6.3.2 To create a dark-structure image:

1. Take N long-exposure dark frames (e.g., 1800 s each).
2. Combine by taking the median level at each pixel (or use some other robust level determinator that rejects cosmic rays). Note that CRs are common in these images.
3. Bias-level correct and trim the combined image.
4. Subtract the bias-structure image.
5. Scale to represent a one-second exposure.

(6.) Depending on resultant structure, various smoothing and/or modelling operations can be used to improve S/N. For example, if only a few "hot" pixels or columns are present, the vast bulk of the pixels may be set to an average dark-rate value rather than the individual values which suffer small number fluctuations.

(7.) The combined dark image may be subtracted from individual dark images to produce a "pure CRs" image. CRs can be analyzed in such images to build up statistics, e.g., what is the energy distribution? what is the size distribution? what do typical profiles look like? These statistics are useful for designing the CR-cleaning procedures.

The value of N should be large enough that the noise in the combined image (following scaling to typical exposure time) is insignificant compared to the read-out noise in a single image. $N \sim 25$ is generally a good compromise between noise and acquisition time. Time to acquire: $N \times (t_{expo} + t_{overhead})$; for $N = 25$, 0.5 hr exposures and 2 min overheads, this is 13–14 hours for a sequence.

6.4 Other Products

Instrument-control scripts and instructions to create bias structure and dark structure frames for the database will be required.

6.5 Outstanding issues

CR-cleaning procedures are still TDB.

There is no discussion of cosmetic defects. These are best removed at the time of CR-cleaning, both for efficiency and because defects often have sharp edges which can be confused with CRs.

There is no discussion of linearity.

We must determine the simplest suitable functional form for the bias level correction during pre-commissioning tests.

We must determine the necessity of bias structure and dark structure corrections during pre-commissioning tests.

It is probable that we can include a pixel-to-pixel response variations (measured through a "pixel-to-pixel flat") as a detector calibration as it is probably independent of color. Other response variations, such as color-dependent variations and fringing, depend both on the detector and its illumination and so cannot be dealt with here.

[A discussion of gain is missing in this section; gain measurements may need to be described.]

Chapter 7

Imaging — Procedures & Database Products

This chapter deals with these elements:

Flat field: An image of a uniformly illuminated ("flat") field. (In this section, "flat field" refers specifically to imaging mode.) Flat field images will be assumed to be normalized to unity at a specified "normalization point" (which in practice will be a cosmetically-clean region on the fiducial CCD). Flat fields are used to remove the signature of the CCDs ("pixel-to-pixel variations") and the system throughput variations ("vignetting"; others?) across the field of view. Flat fields are identified by their source of illumination — **twilight flats** of the twilight sky, **dome flats** of the dome illuminated by special lamps, and **internal flats** of the hatch illuminated by lamps within DEIMOS. In imaging mode, a flat field is alwyas associated with a particular filter and is useful only with that filter.

Fringe frame: A flat-fielded, normalized image of the dark sky (with the continuum component subtracted), showing a summation of interference fringes caused by strong night-sky emission lines. The stability and/or utility of fringe frames is yet to be established.

Relative flux calibration: Adjusting counts across an image to the same count-vs-flux ratio. This is traditionally performed by division by a flat field image, which has been normalized to unity at a specified "normalization point."

Absolute flux calibration: Conversion of count rates into physical flux units. This consists of two constants for each filter, a zeropoint (C_{filt}) and an extinction coefficient (E_{filt}). The defining relation is:

$$mag_{inst} = -2.5\ log(\frac{counts \times e^{-}/DN}{exposure}) + C_{filt} + E_{filt}\ x,$$

where x is the airmass at the time of observation. C_{filt}, sometimes refered to as **total throughput**[1], contains all the information about system efficiency or throughput for the normalization point of the flat-field image.

Standard magnitudes: Magnitudes on a widely accepted system such as Johnson UBV or the SDSS sytem. Conversion from instrumental to standard magnitudes traditionally takes the form

$$mag_{Std} = C_0\ mag_{inst} + C_1\ color + C_2\ color^2.$$

The coefficients C_0, C_1 and C_2 can be deduced from observations of standard stars, or "synthesized" using measured transmission curves and published spectral energy distributions for a wide range of spectral types.

Relative Astrometry: Determination of the relative positions, in arcseconds on the sky, of positions in an image. This can be expressed as a **distortion mapping** between the detector and the gnomonic projection of the refracted sky (corresponding to reverse transforms of D, A, C in the RSCS↔ICS mapping of Chapter 2). We assume independfent polynomial fits in x_{ICS} and y_{ICS}:

$$\xi = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4y + a_5y^2 + a_6y^3 + a_7xy + ...$$

$$\eta = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4y + b_5y^2 + b_6y^3 + b_7xy + ...$$

Absolute Astrometry: Celestial coordinates tied to a fundamental system, and the procedures for determining such coordinates.

[1] from **above** the atmosphere, as defined.

7.1 Background and Issues

Flux Calibration:

[To be written]

Defringing:

[To be written]

Astrometry: There are three levels of astrometry, all using astrometric mapping (CCDCSi $\rightarrow$ ICS $\rightarrow$ FKCS) that will be available from the database. The first level uses telescope telescope pointing information from the image header — expected relative accuracy should be good to [$\pm$1 arcsec] ignoring differential refraction and aberration; absolute accuracy is poor [$\pm$10 arcsec]. The second level adds the measurement of one star (or more) of known coordinates — this improves the absolute accuracy to the level of relative accuracy [$\pm$1 arcsec]. The third level requires three or more astrometric reference stars in the image, and provides a correction to the database astrometric mapping. The precision of this level is the minimum required for slitmask design. It is expected that three reference stars will be adequate to obtain $\pm$0.1 arcsec accuracy. *[Review all the accuracy estimates. The biggest unknown here is the exact pixel scale, and whether it will vary with T_{cam} and/or focus.]*

7.2 Procedures

In the following, "relevant" refers to images and constants that are associated with the filter used in the observation.

Relative flux calibration (Flat-fielding): Retrieve the relevant flat-field image from the database; divide.

Defringing: Retrieve relevant fringe frame from database; fit to sky fringes by scaling; subtract.

Absolute flux calibration: Retrieve the relevant constants (zeropoint and extinction coefficient) from the database; calculate the constant of proportionality ($C_{filt} + E_{filt}\ x$); place constant in the header for use by analysis applications. *[What about gain – should this be included??*

7.3 Database Products

A library of standard flat-field frames through each filter will be maintained for each CCD. [In practice, this may be a product of pixel-to-pixel flats and large-scale flats.]

Nominal system throughputs for each filter will be maintained in the database.

Standard extinction values for Mauna Kea will be maintained in database.

Transforms from instrumental to standard photometric systems will be maintained in database.

If useful/stable, a library of fringe frames for the night sky will be maintained for each filter.

Distortion mappings will be maintained in the database.

Entry	Type	Depend.	Descrip.	Deliv.	Maintenance	Notes
Flat field	$8N$ images	per CCD, filt		1	Optics mods	
Sys. throughput	$(8)N$ const	per filt(CCD)		1	Optics mods	
Extinct. coeff.	N const	per filt		1	Atm. change	
Std. transfroms	$3N$ const	per filt		1	??	
Fringe Frame	$8N$ images	per CCD, filt		1	Optics mods	
Distortion	2D func	??		1	Optics mods.	

NOTES — Delivery: 0=Commissioning, 1=Pipeline.

Required Hardware: – grid-of-holes (GOH) mask for differential distortion map.

7.3.1 To create a flat field image (method 1):

For each filter:

1. Take 3 or more twilight flat frames with adequate exposure. The telescope should be tracking during this sequence, but offset between exposures.

2. Reduce each through bias-level correction.

3. Combine by taking the median level at each pixel (or use some other robust level determinator that rejects cosmic rays). The individual flats should be scaled by their mean/median levels in a cosmetically clean region before combining, to remove differences in sky intensity.

4. Normalize to unity at the normalization point.

Advantages: Straight-forward; illumination by sky; better "color balance." **Disadvantages:** Difficult to obtain images with proper exposures (a model of sky counts per filter per sun altitude would help!); narrow time window; stars in field often leave residual features.

In the above, *adequate exposure* depends on the accuracy desired. Usually $< 1\%$ accuracy is the goal, which means 10^4 electrons level in each image is adequate.

7.3.2 To create a flat field image (method 2):

For each filter:

1. Take N dome flat frames with adequate exposure.
2. Reduce each through bias-level correction.
3. Combine by taking the median level at each pixel (or use some other robust level determinator that rejects cosmic rays). The individual flats should be scaled by their mean/median levels in a cosmetically clean region before combining, to remove fluctuations in lamp intensity.
4. Remove large-scale variations by dividing the combined image by a heavily smoothed version of itself — this leaves a normalized image of pixel-to-pixel variations ("pixel-to-pixel flat").
5. Obtain one (or more) twilight flat(s). (Combine multiple images by first scaling to same level, then taking median value at each pixel.) Smooth and/or spatially-median filter or fit by a surface to retain only large-scale variations ("large scale flat").
6. Multiply the large scale flat by the pixel-to-pixel flat.
7. Normalize to unity at the normalization point.

Advantages: Dome flats easy to acquire; pixel-to-pixel variations can be monitored and combined with large scale flats to update the database more often. **Disadvantages:** Color balance not as good as with sky.

As above, a goal of $< 1\%$ accuracy requires $N\times$ (electron level) $> 10^4$.

7.3.3 To create a fringe frame:

For each filter:

1. Take 3 or more deep frames in darktime with significant sky counts. The telescope should be tracking during this sequence, but offset between exposures. The field should contain as few stars as possible.

2. Reduce each through relative flux calibration.

3. Combine by taking the median level at each pixel (or use some other robust level determinator that rejects stars and cosmic rays). The individual frames should be scaled (or offset?) by their mean/median sky levels.

4. Smooth on scales of a quarter the fringe wavelength.

5. Normalize to unity.

7.3.4 To measure system throughput:

For each filter:

1. Take exposures of N photometric standard stars of varying spectral type; process through relative flux calibration.

2. Determine the count rate for each standard star. Adjust for expected atmospheric extinction. (Adjust for gain.)

3. *[?? The method of defining the instrumental system has not been investigated yet.]*

7.3.5 To create a distortion map:

1. Take one or more images of an astrometric field; process through relative flux calibration.

1a. Take one or more images of the grid-of-holes mask before/after to allow for future differential updates (dome or twilight sky illumination).

2. Measure the pixel positions of stars with known celestial coordinates.

3. Fit the pixel coordinates to the gnomonic projections of *refracted sky* coordinates; fit the same sky coordinates to the pixel coordinates. *[Functional form not yet known, but probably low-order polynomial will be fine.]*

7.3.6 To create a differential update to distortion map:

1. Take one or more images of the grid-of-holes mask (dome or twilight sky illumination).

2. Measure the pixel positions of holes in mask; compare with previous grid-of-holes measurements.

3. Fit the differences in pixel location; this becomes the differential update to be used in conjuction with the distortion map.

7.4 Other Products

Instrument-control scripts and instructions to create the database products listed above will be required.

7.5 Outstanding Issues

What are the standard DEIMOS filters? Do instrumental–standard conversions exist for filters of these specific design? Should the zeropoints follow Johnson, AB, or some other magnitude system?

What are the extinction values for MK? (See Keck Web pages)

Do we want a more elaborate scheme for characterizing the color dependence of pixel-to-pixel variations? This could be done spectroscopically.

There is no discussion of fringing here. This awaits analysis of fringing data.

Is distortion mapping the same for all filters, modulo scale change? This can be checked with grid-of-holes comparisons in different filters.

[The role of gain is not fully developed in absolute flux cal discussions.]

We need to discuss the effect of spatial distortion (different pixel scales throughout the image) on flux calibration. If the distortion is mapable (it should be!), there is no problem correcting the data for this.

How wide is the time window for twilight flats? How many filters can we expect to do in each window?

Chapter 8

Spectroscopy — Procedures & Database Products

Spectroscopic calibrations are complex, with added complexity from the multislit nature of DEIMOS. We assume multislit throughout; longslit is a special case of multislit.

This chapter deals with these elements:

X-rectification: This operation removes **x-distortion** — distortion in the x-direction — and is commonly referred to as "straightening the spectra." (Note that this only applies to the spectrum of a slitlet; tilts in object spectra due to atmospheric dispersion are not removed by this process.) For simplicity, the operation is applied to *all* spectral data before any other spectral calibration. We adopt a form:

$$x' = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4y + c_5y^2 + c_6y^3 + c_7xy + ...$$

to describe the x-distortion pattern.

Wavelength calibration: Assigning accurate wavelengths to each pixel in a spectrum. Accuracy of assignment is TBD, but is expected to be ± 0.25 pixel or better. We assume the form:

$$\lambda = f(x,y) = p_o(x) + q_o(x)y + r_a(x)y^2 + s_a(x)y^3 + t_a(x)y^4,$$

where the subscripts o and a denote derivation from the individual observation or from the database archive, respectively. We assume the spectra have already been **x-rectified** or straightened. This functional form describes the wavelength dependence for across an image;

note that between slitlets there will be a discontinuity in the coefficient functions $p(x), q(x), r(x), s(x), t(x),$. The general form requires knowledge of p, q, r, s, t as functions of $(X, Y)_{SMCS}$ as well.

Relative flux calibration: Adjusting counts along a spectrum *and* between slits so that the same count-vs-flux ratio applies at a particular wavelength.

Absolute flux calibration: Conversion of count rates into physical flux units. This consists of a "sensitivity curve" (spectrum) that converts a count-rate at a given wavelength into units of erg s^{-1} cm^{-2} $\mathrm{\AA}^{-1}$.

Flat field: *[To be written.]*

8.1 Background and Issues

Wavelength Calibration:

Internal arcs lamps cannot be used to precisely wavelength calibrate spectra on the sky for these reasons:

- Arc lamps don't illuminate optics the same as celestial objects;
- Temporal instabilies change lambda scale;
- Flexure effects;
- Scale change with T; and
- Scale change with focus

In addition, we don't want to take internal arc spectra during the night, both because of loss of observing time and because the hatch must be closed, which interrupts guiding.

However, spectra on the sky (particularly in the blue) will not always include enough night sky lines of sufficient S/N to provide a satisfactory wavelength solution. A hybrid calibration — using internal lamps for higher order terms and night sky lines for lower-order terms — is probably the most practical method to overcome these difficulties. In practice, we propose that the coefficients r, s, t can be obtained from internal arc spectra (and maintained in the database for pipeline mode), and only the zeropoint and scale (p, q) come directly from the science images using night sky lines. Our goal by this method is $\Delta\lambda = 0 \pm 0.25$ px.

The coefficients are measured for each grating using a set of "line-of-slitlets" (LOS) masks to provide a grid of positions in the MF/SMCS. Spectra are obtained and measured for each slit through a sequence of grating tilts. For each grating, this gives a set of coefficients on a representative grid in grating tilt (μ) and slitmask coordinates (X, Y), that is,
$p_g(\mu, X, Y)$
$q_g(\mu, X, Y)$
$r_g(\mu, X, Y)$
$s_g(\mu, X, Y)$
$t_g(\mu, X, Y)$
Note that, once x-rectified, there is a one-to-one correspondence between X and x[1].

There are two options for database storage. First, the actual 3-D data cubes (one per grating) can be stored. For reasonable values of gratings (5), LOS masks (8), slits per mask (100) and grating tilts (12), this means storing five data cubes of 9600 numbers each, or 48000 numbers total. To retrieve coeeficients would involve interpolating in the appropriate data cube. The second method is to fit each data cube with an interpolating function of the form:
$p_g(\mu, X, Y) = a + b_\mu \mu + c_\mu \mu^2 + d_\mu \mu^3 + e_\mu \mu^4 + \ldots + b_X X + c_X X^2 + d_X X^3 + e_X X^4 + \ldots + b_Y Y + c_Y Y^2 + d_Y Y^3 + e_Y Y^4 + \ldots$
For five interpolating functions with 13 coefficients each, we need to store only 65 numbers per grating, or 325 numbers total for five gratings. The actual number of coefficients needed is TBD.

Flux Calibration: The grating introduces an additional degree of complexity, partially by introducing an occasional vignetting that depends on grating tilt, but most importantly because the camera will now vignette *some* of the incoming beams. The vignetting at the grating itself, however, will affect absolute but not relative flux calibration, as the light is not yet dispersed. Furthermore, the camera vignetting is a function of the grating reflection angle only, and it does not matter if that angle is a result of dispersion, slitlet location, or both. For this reason, we can use continuous spectra of a line-of-holes (LOH) mask to map out the camera vignetting sufficient for relative flux calibration.

Here is a summary of the various components to spectral flux calibration:

collimator/tent vignetting: this affects the overall flux level, independent

[1] If the mapping of $X \leftrightarrow x$ changes (for example, by resetting the grating in its cell or adjusting pitch angle) then either a new calibration set must be acquired for that grating or the existing one must be corrected.

of wavelength, as a function of slitmask X, Y. Use a direct flat to determine its effect, and apply to the entire spectrum.

grating vignetting: only a problem for the large (8×12) grating at large tilts. Measurement TBD.

grating efficiency: can be measured through a series of observations over various tilts for each grating. TBD: standard light source appropriate for these measurements.

camera vignetting: can be measured via LOH spectra.

8.2 Procedures

8.2.1 Baseline Mode

x-rectification: The user may use either a LOH mask or even spaces between slits on a science slitmask, illuminated by internal lamp, dome lamp or (preferably) the twilight sky to trace out slitlet edges and solve for the x-distortion in (x, y).

Wavelength Calibration: In baseline mode, the user employs either internal arc spectra or night sky lines to solve for the values of the coefficient functions $p(x), q(x), r(x), s(x), t(x)$. In practice, this will probably involve internal arcs for r, s, t and night sky lines for the zeropoint and scale p, q. Again, note that these coefficients will be discontinuous in x between slitlets. A specific "recipe" follows:

1. Obtain internal arc spectra through each mask with same grating, tilt and order-separating filter as used for science exposures;
2. X-rectify in same manner as science image;
3. Y-rectify in same manner as science image, to straighten emission lines; *[Not described yet]*
4. Extract calibration spectra in same manner as object spectra;
5. Measure line peaks to obtain y position for each arc line;
6. Identify line wavelengths (arc and nightsky spectra and wavelengths will be kept in the database);
7. Solve for coefficients p_a, q_a, r_a, s_a, t_a (here, "a" refers to "arc spectrum");

8. Extract night sky spectrum (or local arc) taken at same position/time as science image (the S/N in the lines can be relatively low);
9. Measure and identify emission lines;
10. Solve for unknown coefficients p_o, q_o for each object spectrum, assuming r_a, s_a, t_a from previous fits.
11. Extracted object spectrum is remapped onto a specified (x, λ) grid.

Flux Calibration:

8.2.2 Pipeline Mode

x-rectification: The x-distortion map is retreived from the database; image is remapped. *[What about scale changes?].*

Wavelength Calibration: Coefficients p, q, r, s, t are interpolated for each x and $(X, Y)_{SMCS}$ from information in the database . Night sky lines (or internal arc at same position/time) are used to solve for specific p_o, q_o, assuming r, s, t from above. Image is remapped onto a specified (x, λ) grid.

Flux Calibration:

8.3 Database Products

The database will contain a current x-distortion map; for each grating, a 3-D table of coefficients p_a, q_a, r_a, s_a, t_a that can be interpolated to specific values at a given (x, y) and $(X, Y)_{SMCS}$ *or* interpolating functions for each grating to provide the same values; *[flux cal info, TDB].*

The database will also contain certain supplemental information:

wavelength- and flux-calibrated spectra of each arc lamp and the night sky, obtained with each grating;

list of wavelengths for each arc lamp and the night sky;

grating and order-separating filter information.

Required Hardware:

- Line-of-slitlets (LOS) mask
- Line-of-holes (LOH) mask

8.3.1 To create the database wavelength calibration:

1. Obtain spectral images through LOS masks illuminated with an appropriate arc source (appropriate means sufficient density of identified spectral lines throughout the spectral region of interest). This requires, for each grating, exposing through a set of LOS masks and a sequence of grating tilts. (Assuming 5 gratings, 8 LOS masks, 12 grating tilts, and 1 minute per exposure, this means 480 exposures for a total time of 8 hr.)

2. X-rectify each image.

3. Extract the spectrum for each aperture in the LOS mask (48000 spectra total).

4. Measure and identify lines in each spectrum and solve for the coefficients p, q, r, s, t as functions of μ, X, Y. Place in a 3-D table for each grating and tilt (240000 coefficients total).

[5.] Fit interpolating functions to the values of p, q, r, s, t as functions of μ, X, Y for each grating.

8.3.2 To create the database relative flux calibration:

8.3.3 To create the database throughput calibration:

8.4 Other Products

Instrument-control scripts and instructions to create the database products listed above will be required.

8.5 Outstanding Issues

As stated earlier, for flux calibration there is a need for a continuum light source of constant spectral shape.

We need to discuss the effect of spatial distortion (different scales throughout the image) on flux calibration. If the distortion is mapable, there is no problem correcting the data for this.

We have not described slit functions (variations in throughput due to irregularities in the slit shape). These must be mapped by the user for each mask, and will be sensitive to plate scale changes, but their measurement and removal is not expected to be a problem. Current milling practices appear to reduce variations along the slit to less than about 1%.

Chapter 9

Health & Stability of Instrument — Procedures

[To be written — below is outline only]

This chapter describes these elements:

1. Astrometric Stability
2. Flatfield Stability
3. Throughput
4. Grating Angles???

9.0.1 Astrometric Stability

9.0.2 Flatfield Stability

9.0.3 Throughput

9.0.4 Grating Angles???

9.0.5 Database Products

9.0.6 Other Products

We should provide scripts and instructions for putting the instrument through a standard diagnostic sequence that can be analysed for temporal changes.

9.1 Outstanding Issues

Part IV

Flexure Compensation

Chapter 1

Introduction

1.1 Motivation

The Flexure Compensation (FC) system is a subsystem designed to maintain constant image position on the detector over short timescales as the spectrograph rotates, and over longer timescales as the spectrograph optical components are manipulated and maintained. As position angle changes, deformation of the optical supports causes image motion on the detector. Finite element analyses and tests of the grating cells and mounts suggest that cumulative passive tilt and rotation errors through the optical path will create a variation in the beam entrance angle to the camera of from 10 to 20 arcseconds. This is after careful attention to these errors at every stage of the design. A 20 arcsecond variation corresponds to 2.5 px of bulk image motion on the detector. This is about 10 times larger than the error budget and must be corrected by the FC system.

Additional image motion is caused by errors in the re-positioning of optical elements. Most important are gratings, which are continually being removed and re-inserted, both on the grating mount as gratings are changed during an observing run, and within the grating holders when the grating complement is altered prior to a run.

All these sources of bulk image motion also induce distortion in the image. These distortions are not correctable by beam steering and thus leave residual position errors. In addition, there are two other kinds of image motion that are not correctable by the FC system: image scale changes and image rotation. The image scale is very sensitive to camera temperature in the basic optical design, but we are attempting to mitigate this by passive thermal compensation of the camera cell. A 10 arcsecond in-plane rotation of the gratings or detector corresponds to 0.14 px (rms) of image rotation (0.20 px at the edge of the field). The passive design stiffness appears to be better than this, and no active correction is made for image rotation.

An error budget for these residual uncorrected position errors is given below in chapter 2.

Two image quality requirements set the stringent error allowance for image motion:

Image Quality The most stringent requirement in image stability comes from line width (resolution) in spectroscopic mode. The tightest case occurs with the 1200-line grating used in the red region of the spectrum (where anamorphic slit demagnification can approach 2.0) and with narrow 0.5 arcsecond slits. The expected optical fabrication errors are low enough that rms image diameters will approach the theoretical design image size over much of the field. To avoid degradation of 23 μm rms diameter images (1-D rms radius = 0.54 px) by more than 10%, the image motion during the exposure should be limited to 0.25 px rms in the dispersion direction. This corresponds to a variation in the beam entrance angle to the camera of 2 arcseconds rms, or a cumulative tilt of the grating, tent mirror, and collimator mirror of roughly 1 arc second. In direct-imaging mode and at lower dispersions, this tolerance can be relaxed to 0.5 px rms, corresponding to 4 arcseconds rms of beam tilt. As mentioned above, the expected passive structural beam tilts are 10 to 20 arcseconds, between 2.5 and 10 times larger than the range of tolerances.

Calibration Accuracy A long-term goal of DEIMOS is to provide pipeline data calibration using a calibration database. The ability to reset the spectrograph accurately to any previous setting will be a key ingredient in creating such a database. A major limitation is caused by variations in the flat fields. The most important source of flat-field errors is fringing, which in turn is caused mainly by spectrograph flexure. Fringing variation is the limiting factor in the accuracy of spectroscopic CCD flat fields in the red spectral region and prevents sky-limited observations from reaching the theoretical photon-limited S/N.

The improvement in flat-fielding that would result from image stability at the 0.25 px level is not known because other effects may contribute to fringing variations (e.g., pupil rotation and accompanying changes in the pattern of pixel illumination). However, it is likely that fringing variations will be significantly reduced if the above stability goal of 0.25 px rms is met.

In particular, it was shown in Part 1, Chapter 4.3 that this level of stability would completely eliminate all contributions to fringing from wavelength variations. For further discussion of the physics of fringing, see the above section.

1.2 Requirements

The performance requirements for the FC system are as follows:

- Operate in both direct-imaging and spectroscopic mode with all filters, gratings, and grating tilts.
- Provide rms accuracy in each coordinate of 0.25 px.
- Have minimal impact on normal operations.
- Provide actuator range to accommodate the expected range of image motion during normal operation, and the expected image displacements caused by removal/reinsertion of the detector and/or all optical elements (e.g., for collimator aluminization).
- Provide guide signal updates every 15 seconds to the actuators.
- Monitor plate scale, image rotation, and focus.
- Provide continuous feedback to the observer and signal various alarm conditions.
- Store relevant data in the DEIMOS database every 15 seconds.

1.3 System Overview

A system designed to meet these requirements is shown in Figure 1.1. It consists of:

- arc lamps
- optical fibers
- FC CCDs and signal chain
- control algorithm

- actuators.

Emission lines from Ne, Ar, and Hg arc lamps are used to track the effect of structural deformations. Four optical fibers feed light from the arc lamps into the telescope focal plane adjacent to the active optical slit area. Arc lamps are used in preference to a Fabry-Perot etalon because their wavelengths are stable at all temperatures. All three lamps feed each of the four fibers. The light from each fiber is shaped into an f/15 beam so that it illuminates the spectrograph in a manner analogous to celestial objects.

The image of each fiber is a spot in direct-imaging mode and is a line of spots in spectroscopic mode. These images fall on two flexure compensation CCDs, which are read out during the regular science exposure at 15 second intervals using a separate signal chain.

The goal of the FC system is to steer the image in translation in X and Y (Y is in the dispersion direction; X is along the slit). This requires two pieces of information, which can be obtained in principle from the coordinate locations of a single spot. In imaging mode, the CCDs will see a total of four spots, one from each fiber. Thus there is redundant information. Part of this redundancy will be used to compare information from separate spots to ensure that the image is not corrupted by a cosmic ray or cosmetic flaw on the CCD. However, still more information is available from the fact that spots are on opposite sides of the mosaic. We will make use of this to monitor both image rotation and scale. These variables cannot be corrected by the FC system, but the information is valuable as a check on proper instrument operation. Finally, we also plan to measure spot size (and perhaps ellipticity and other higher-order moments) to monitor focus and other aspects of image quality.

In spectroscopic mode, there will usually be more than 4 spots that are visible; however, some tilts of the high-dispersion 1200-line grating will provide only one spot per CCD. The analysis program will have to adapt to the number of spots that are visible at any one time.

The FC CCDs operate in frame transfer mode, obviating the need for a mechanical shutter. Image centroid algorithm(s) find the image position(s) on the two CCD images and feed correction signals to two mechanical actuators in the spectrograph. One of these, a piezoelectric actuator, tilts the tent mirror and moves the image along the dispersion direction (Y). The other is a moving stage in the dewar that translates the detector in the direction perpendicular to the dispersion (X). Together, the two actuators stabilize the image in translation.

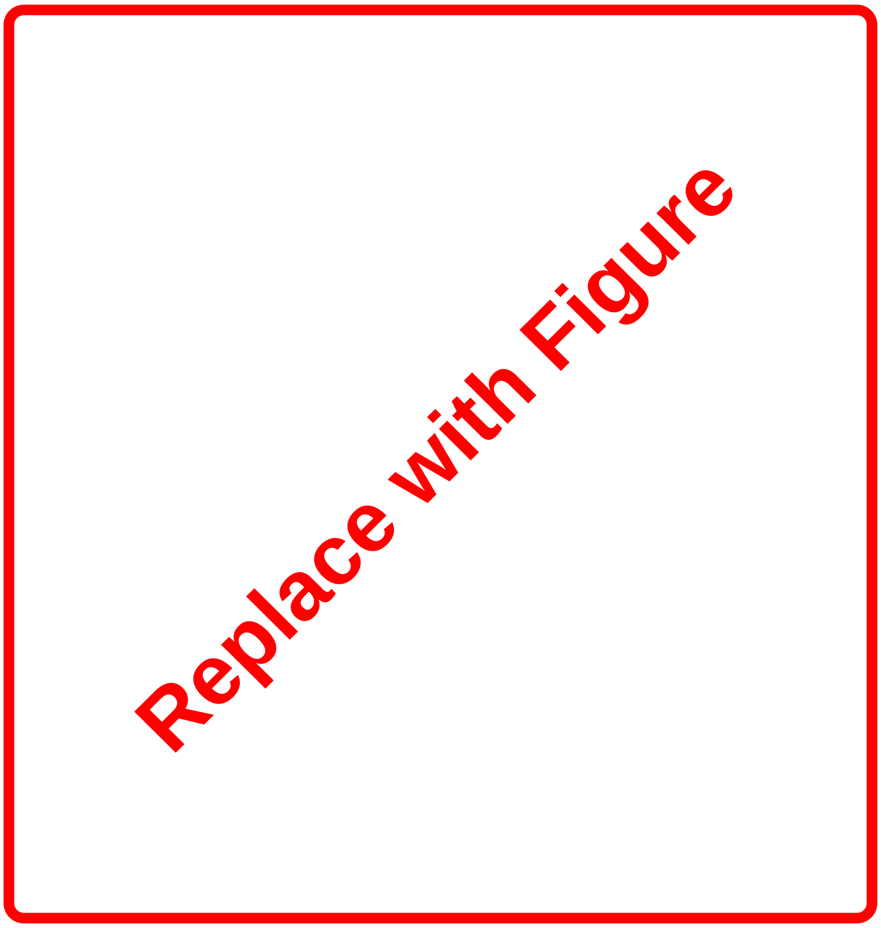


Figure 1.1: fc overview

The goal of achieving an accuracy of 0.25 px rms every 15 seconds determines most aspects of the FC system. Two hundred micron diameter fibers will produce 0.5 arcsecond (4.2 px) FWHM images at the edge of the FOV (Gaussian $\sigma = 1.8$ px). Roughly 50 photons are therefore sufficient in principle to yield 0.25 px rms precision per integration. To essentially eliminate photon noise from the error budget, we have set a threshold of 1000 photons per exposure (see Chapter 2). The faintest desired arc line contains about 1% of the total flux of the three lamps. Therefore, the light intensity can be adjusted to yield 1000 photons from this line while at the same time not exceeding 100,000 photons in total, which is less than full well. Broadband filters in direct imaging mode will reduce this integrated signal by a factor of several but will still leave sufficient photons to provide the required precision. Order-separating filters in spectroscopic mode will have no effect since the FC CCDs are near the center of the desired wavelength range. We have adopted a fixed exposure time of 10 seconds, which allows 5 seconds to read the image, centroid, compute the corrections, and move the actuators.

The FC CCDs have a 600 x 1200 format with 15 micron pixels. In frame-transfer mode the active area is 300 x 1200 px, with the long dimension along the spectrum. At the highest dispersion (1200-line grating, 0.32 A per px), each CCD captures 400 A of spectrum. By positioning the FC CCDs optimally along the dispersion direction, at least one usable arc line can be captured for all filters, gratings, and grating tilts.

From the expected flexure estimates in Section 1.1, the required range of the X- and Y-actuators in normal use will be a few pixels. The current design provides 12 pixels of motion for the X-actuator (detector) and 22 px of motion for the Y-actuator (tent mirror). A procedure has been developed to reset the mean position of all optical elements after servicing (such as the collimator and tent mirror) to within 5 to 10 arc seconds, corresponding to 1.3 to 2.5 pixels on the detector. The detector can also be removed and reset to similar accuracy. Thus both actuators have enough range to reset the image to its previous location after servicing of the optical train. Despite this, it is anticipated that changes to the optical train will likely require a complete rebuild of the calibration database, and this is allowed for in our operational plan.

Chapter 2

Error Budget

2.1 Photon Statistics

The ultimate limit on positional accuracy is the photon noise per FC spot. We assume that there is only one spot visible in each FC CCD. The expected FWHM of a spot image is about $0.5''$ (4.2 px), set by optical aberrations at the FC CCDs near the edge of the camera FOV. For a Gaussian PSF, the effective 1-d σ is 1.79 px. For a rough calculation of the expected S/N in a spot, we assume that the spot area is $4.2^2 = 18$ px^2, that the readout noise of the FC CCDs is 5 e$^-$ px^{-1} (no binning), and that the total number of detected photons $N = 1000$. The effective photon noise under the image in 18 px due to readout noise is 450 e$^-$. If the readout noise is ignored, the 1-d rms positional error, δ, is 1.79 px/$\sqrt{1000}$= 0.06 px. Subtracting the background from each pixel increases the (photon noise)^2 by 450/1000 = 45%, which increases the positional error by 22% to 0.07 px. Both of these errors are small compared to the final error budget of 0.25 rms in each coordinate.

We conclude that a detected photon flux of 1000 counts and a CCD RO noise of 5 e$^-$ px^{-1} are reasonable targets.

2.2 Optical Distortions *(partial draft)*

2.2.1 Displacement of the Optical Elements and the Detector

In spectroscopy mode, any translation of the image is accompanied by small optical distortions due to the reflection of the beam off a grating rather than a mirror. Each displacement of an optical element (e.g., the collimator mirror, tent mirror, grating, camera, or detector) has its own distortion pattern and magnitude. Residual errors in image position are introduced by the fact that distortions of the compensating elements (the tent mirror and detector) do not perfectly match the distortions of the other elements being corrected.

A rough rule of thumb is that an image motion of a given size is accompanied by an uncorrectable distortion that is 0.08 times as large. These residuals limit the largest bulk image correction that can be achieved by the FC system. For example, passive motions due to flexure may amount to as much as 2.5 px rms (see Section 1.1), inducing distortion errors that are 8% as large, or 0.20 px. This would take up essentially the whole error budget. We are striving to keep passive flexure a factor of two below this level, to leave room for other terms.

Note: This section is incomplete. It will be expanded in future to describe in detail the various kinds of optical distortions and compare them quantitatively with the expected motions of optical components.

2.2.2 Motions of Camera Lens Elements

Displacements of individual lens elements within the camera also cause position distortions. Flexure analysis of the camera indicates that these motions should be 10 times smaller than the effects noted above and can be ignored.

2.3 Error of the Applied Correction

Because the corrections are computed only every 15 s, the correction will not be perfect even if the centroids are perfectly known. The maximum error occurs when the PA is changing rapidly, which occurs near zenith. Here the spectrograph rotates 180°in approximately 5 minutes, the width of the alt-az dead zone. If we assume that the maximum peak-to-valley flexure of 5 px takes place in 5 minutes, a time lag of 15 s corresponds to an error of 0.25 px. This is large compared to the goal but exists over a short time only and could probably be reduced with a predictor-corrector scheme. We allot 0.05 px to this term.

Other correction errors should be small, as the system operates at low bandwidth compared to the bandwidth of the expected errors, and the gains of the active components should be highly stable over time after initial calibration.

2.4 Uncorrected Errors

The FC system is capable of steering the beam only – rotations and scale changes of the image are not corrected. We estimate here the errors introduced by these uncorrected terms.

2.4.1 Thermal Effects on Camera Image Scale

A major lien against image stability is changes in the focal length of the camera due to temperature changes. This is discussed in the memo "Change in DEIMOS Plate Scale due to Thermal Effects," by S. M Faber (copies available). The camera scale is exceedingly sensitive to temperature owing to the use of positive-power thermally-sensitive CaF_2 elements in conjunction with negative-power glass elements that are much less sensitive to temperature.

The fractional sensitivity of image scale to temperature is -141 $\times 10^{-6}$ per °C. If scale changes contribute no more than 0.1 px rms to the image motion budget, the camera temperature must be kept constant to 0.14 °Crms. The design operating temperature range is ± 5 °C, so this temperature constancy represents only 3% of the expected temperature range.

The optical design is non-telecentric, so that slight changes of focus alter the scale. If advantage is taken of this effect, the requirement on temperature constancy might be relaxed somewhat, to 0.6 °Crms, though pushing this approach is not attractive owing to probable degradation of image quality. This relaxed tolerance represents 12% of the range.

We are attempting to compensate for the thermal scale change by altering the separation of two multiplets in the camera as a function of temperature. This separation would be controlled by a passive spacer whose length varies with temperature in the desired way. The allowed error of the correction is the same as the above percentage fraction of the temperature range. The main source of error in the passive approach is ignorance of the true physical constants of materials – the indices of refraction and coefficients of thermal expansion. A plausible error due to this ignorance is 10%. This is in between the most stringent specification of 3% but less than the maximally relaxed tolerance of 12%.

We adopt a final error of 10% for the passive correction and assume that we will use some degree of focus correction as an additional aid. Together they imply a final error of about 0.15 px rms from this term. If this scheme proves inadequate, we still have the option of adding some form of active thermal control to the camera optics.

2.4.2 Image Rotation

The image may rotate on the detector owing to rotations of the detector itself, rotations of the camera holding the detector, and rotations of the grating. We allot 0.1–0.2 px to these terms, as suggested by results achieved from finite element analyses and prototypes of mounts that have been built.

2.5 Summary

The final error budget is given below:

Table 1.1. FC Error Budget

Error Source	1-d rms px
Photon statistics	0.07
Optical distortions	0.13-0.25
Correction error	0.05
Scale error	0.15
Image rotation	0.1-0.2
Total	0.24-36
Goal	0.25

These results suggest that, with care, the desired stability goal of 0.25 px rms may be met but that attention to every detail will be required. We note that the original goal was developed with the most demanding optical requirement in mind. Most optical images would be well preserved if the goal were relaxed by a factor of two. Fringing also degrades gracefully over this range. For a typical 600-line mm^{-1} grating yielding 0.66 Å/ mm^{-1}, the peak fringe flat-fielding error caused by an rms px shift of 0.5 px is still less than 1% (SEE Part I, Section 4.3).

We conclude that meeting the image stability goal of 0.25 px rms will be difficult but that the help offered by the FC system is essential to approaching this goal.

The following sections are still in preparation:

Chapter 3

Arc Lamps

3.1 Spectral Selection

3.2 Expected Fluxes

3.3 Scattered Light

Chapter 4

Optical Fibers

4.1 Fiber Selection

4.2 Fiber Illumination

4.3 Fiber Focal Plane Mount

4.4 Emitted Beam

Chapter 5

Signal Chain

5.1 FC CCDs

5.2 Preamplifiers, CCD Controller, VME Crate

Chapter 6

Duty Cycle

Chapter 7

Operating Modes

7.1 Overview of Modes

7.2 User GUI

7.3 Monitor Mode

7.4 Control Mode

Part V

Budget and Schedule

Chapter 1

To Be Written

Index