Managing DEIMOS removable elements and instrument configuration

D. A. Clarke, S. L. Allen, A. C. Phillips, R. I. Kibrick, V. Wallace, and J. P. Lewis

UCO/Lick Observatory, University of California, Santa Cruz, CA 95064 USA

ABSTRACT

This paper describes tools and methods used to manage DEIMOS removable elements (filters, gratings, and slitmasks). The iterative process of adapting and refining our basic strategy to the working conditions and requirements of Keck Observatory staff is not yet complete; hence this paper should be read as a Work In Progress report.

Keywords: spectrograph, DEIMOS, multi-object, slitmask, grating, barcode, configuration, removable elements, filter

1. INTRODUCTION

1.1. Requirements

DEIMOS[*] is a large multi-object spectrograph using astrometric slitmasks to obtain spectra for many objects in one exposure.[1] The instrument contains a three-position grating transport mechanism, one of whose positions is a fixed-angle mirror; the other two positions are independently tiltable holders for removable grating elements. At present there are four gratings in the inventory. DEIMOS contains a seven-position filter wheel, and at present there are eleven removable filter elements in the inventory. Lastly it contains a movable cassette and insertion arm which comprise a “jukebox” for slitmasks. The cassette has 11 slots, and as many as 50 slitmasks might be manufactured for one observing run.

Setting up the instrument for a run might involve (in order of increasing probability) some number of filter exchanges, one or two grating exchanges, and (almost certainly) slitmask replacements.

During observing, these elements are selected (moved into the light path) by the observer using an instrument GUI which features both menus and matrices of pushbuttons. Obviously it is essential to ensure that the names which appear on the GUI (in menus, button labels, *etc.*) match the elements currently loaded in the instrument.

A software daemon which we call a *dispatcher* mediates between the many clients of our distributed instrument control system and the hardware controllers hard-wired to motors and pneumatic actuators. This dispatcher must be able to accept a command expressed in namespace (*e.g.* set `DWFILNAM`[†] to be *V*) and translate it into a move request in raw motor counts (e.g. set `DWFILRAW` to be *56789*).[2] It must be configured correctly, so that names map properly to currently installed elements in specific positions.

Finally, when an image is acquired, many keywords are captured in the FITS file header to record the state of the instrument. Among these are the names of the removable elements which were in the light path during the exposure. In addition, a set of FITS tables is appended to each image taken through a slitmask. These tables contain a complete description of the mask, including the catalog of objects to which it pertains, its ideal (sky) geometry, and its actual (metal) geometry as milled. (We usually refer to the sky geometry as the *design* and the metal geometry as the *blueprint.*)

Further author information: (Send correspondence to D.A.C.): D.A.C.: de@ucolick.org / S.L.A.: sla@ucolick.org / A.C.P.: phillips@ucolick.org / R.I.K.: kibrick@ucolick.org / V.W.: vernon@ucolick.org / J.P.L.: jeff@ucolick.org

[*]The **De**ep **I**maging **M**ulti-**O**bject **S**pectrograph, designed and built by UCO/Lick Observatory and commissioned at Keck Observatory in the summer of 2002

[†]The keyword nomenclature convention in use for recent Lick instruments reserves the first 5 characters for the motor or mechanism name, and the last 3 characters for the quantity (with implicit units) being expressed. Hence `DWFIL` is the dewar filter wheel; `DWFILNAM` refers to a named position, and `DWFILRAW` to a position in raw motor encoder counts.

The mask data attached to the image serve two important purposes: they enable semi-automated mask alignment during observing, and they are essential for later data reduction. Each image taken with DEIMOS contains self-descriptive information needed for its own reduction. These data in the FITS files (both header keywords and mask design data) must match the elements currently loaded into the instrument and used for the exposure.

Failure in any of these three configuration essentials (GUI nomenclature and labelling of controls, dispatcher configuration, and correct FITS header and mask design information in image files) will render the instrument difficult to operate, and/or sabotage downstream data reduction and corrupt the archival record. This paper describes our attempt to make instrument configuration both reliable and relatively easy.

1.2. Strategy

From a software standpoint, there is only one truly authoritative source of information about the instrument configuration, and that is the relational database in which we store most of the "metadata" required for instrument operation. A table in this database maps, per stage per instrument, the correspondence of raw (motor encoder counts), ordinal (numbered detentes for a rotating or linear stage), and named positions. In this table is recorded the fact that `DWFILRAW=56789` corresponds to `DWFILORD=4`, and that in this position we have loaded the filter whose name is *V*.

In our design, this type of information is actually found in two distinct tables. One contains what we might call the *factory* or *default* configuration of the stage, whose positions might be named *Pos1*, *Pos2*, and so forth. The other contains the *dynamic*, run-specific, current configuration data in which positions might be named *B, V, R, I* and so forth. The factory defaults table stores an archival copy of our design intention for the stage: (for example) it has seven ordinal positions or detentes, and these positions correspond to certain motor encoder readings. The dynamic table is a snapshot of the stages as configured for tonight with real removable elements.

Since these dynamic data (as the name implies) can change nightly, they are not compiled into any code. Instead, the utility called `CodeGen`[3] generates configuration files for the dispatcher to read at startup; the "dashboard" instrument control GUI[4] reads the database dynamic map table directly at startup in order to paint its menus correctly. The collection and correct use of mask design data are a little more complicated and will be discussed below; but all this configuration information is dynamic and is either read directly from the database, or from files which are generated from the database contents and replaced as a result of completing the instrument configuration activity successfully.

In summary, the reconfiguration process always consists of four phases: (1) gather information about the new configuration; (2) rewrite the dynamic database table; (3) re-generate and install configuration files; (4) restart dispatchers and any instances of the GUI, so that they "see" the new information.

2. PREVIOUS INSTRUMENTS

Traditionally, we have provided instrument scientists/technicians at Keck Observatory with a GUI (`inconfig`) which enables them to define, for each removable-element stage on a spectrograph (such as ESI[*]) a set of raw positions (motor counts), ordinal positions, and names of the ordinal positions. In operational use these positions are named after the removable elements which occupy them. The process of configuration was somewhat tedious.

After elements had been physically inserted, removed, swapped, etc., the responsible party sat down at a GUI and selected each stage in turn, correcting the element names as needed – either from memory or by means of a written list. Each correct name had to be typed in verbatim, rather than selected from a menu. In some cases (the ESI filter and mask wheels) a stage might be so configurable that the number of ordinal positions and their corresponding raw positions might change. The responsible party then also had to type in integer numbers, some of them many digits in length.

After proofreading the information displayed in the GUI, the user would press a `COMMIT` button. This would have two results. First, the current configuration data as edited by the user would be committed to the relational

[*]The **E**chellette **S**pectrograph and **I**mager, designed and built by UCO/Lick Observatory and commissioned at Keck Observatory in the summer of 1999

database. Then the `CodeGen` application would be run, generating configuration files and installing them in the appropriate locations. The user would then restart the dispatcher daemon(s) and the observer GUI. This completed all the phases of the configuration process outlined above.

This first-generation instrument configuration GUI was not universally beloved. It was rather complicated and unwieldy to use. On more general principles, it was never very satisfactory to force the user to type in the names of filters, aperture masks, slitmasks and so forth. The possibility of scribal error was always present.

3. DEIMOS INNOVATIONS

For DEIMOS we went one step further and designed a new database schema describing all the removable elements. Each filter is described by data in RDBMS tables, as is each grating and every slitmask ever milled. Information such as dimensions, weight, glass type, and coating is archived for the filters and gratings. Complete design data as well as owner, intended use date, intended use location and so forth, are archived for every slitmask design ever submitted for milling.

We decided to affix permanent bar code labels to the holders for the glass elements, so that the physical element could be instantly and automatically associated with all the descriptive data in the database. Each element lives in a permanent holder. Since the holders are fully specified by AutoCAD drawings, they can be semi-automatically produced using a machine-controlled mill. We made more filter holders than would initially be needed.[*] The slitmasks did not have holders or frames, so their labels would be attached directly to the metal of the mask.

Originally[5] we had hoped to install three permanent fixed-mount bar code scanners inside the instrument, aimed at the bar code labels which would become visible as the elements moved into position. Thus we hoped that (once the technician or instrument scientist had finished installing removable elements) the instrument could "scan itself" and determine what was where. However, in an attempt to reduce hardware costs this scheme was descoped. As built, only the slit mask cassette mechanism retained a fixed-mount scanner. The grating transport and filter wheel do not have dedicated scanners.

We therefore provided a hand held (*bar gun*) scanner on the exterior of the instrument for identifying the gratings and filters. We wrote a new kind of daemon (named `dremel`, for **D**EIMOS **Rem**ovable **El**ements). At startup, this daemon would cache the last-known instrument configuration data. Being a KTL[†] keyword client, it could then listen for access hatch open/close events and hand-held bar gun events. The scanned values would be interpreted as element barcodes or as commands (see below). If the bar code for a known filter were to be scanned when the filter access hatch was open and the filter wheel was in a specific unload position (say, *Unload4*), then the daemon would conclude that someone had put or was putting the scanned filter into that position. It would update its cache accordingly.

In theory, the instrument scientist (hereafter IS) or technician would go to the instrument on the Nasmyth platform with a cart full of removable elements. After stuffing the filter wheel, he/she would rotate it into each ordinal unload position and scan the exposed bar code of each filter holder in turn. A similar process would identify the gratings. The `dremel` daemon would always be listening; no software setup or special tools would be required.

The requested slitmask inventory for the night could be loaded into the cassette in any order, with no need to keep track of what went where. The tech would then scan a "magic" barcode on a laminated sheet affixed to the instrument; this would trigger the automatic scanning of all barcodes on the installed slitmasks (a subroutine in the `dremel` code).

[*]Our intention was to set aside a couple of the spare filter holders to house filters brought to Keck by DEIMOS observers. These filters would only be known as *USER1* and *USER2*; it would be up to the observer to keep track of these "foreign" elements.

[†]See Ref. 2. The Keck Tasking Library is an API designed at Keck and used extensively by developers of instruments for Keck telescopes. It implements a keyword-based client/server hardware control system whose keywords conform to the FITS standard and hence can easily be used for documentation in FITS headers.

When this process was complete, `dremel`'s internal cache would be updated with the new configuration data. The tech would finally scan a magic `COMMIT` barcode on the laminated sheet; this would trigger the same type of commit operation which for previous instruments was triggered by pressing a button on the configuration GUI. Our intention was that the tech would never have to go to a computer screen and use a keyboard and mouse; instrument configuration could be done without taking one's gloves off.

As we all know to our sorrow, theory and practise are not always on speaking terms. We found that it was impossible to scan all the mask barcodes accurately in less than about four elapsed minutes. The techs did not like to stand about idly on the chilly Nasmyth platform while waiting for this lengthy procedure to complete. The techs preferred to go back to a more comfortable control room and complete the process from there.

We therefore modified the `dremel` daemon to accept signals in lieu of scanned magic barcodes, so that the tech could trigger both the `SCAN MASKS` and `COMMIT` operations without using the bar gun on the Nasmyth platform. In the next revision of the code we will provide keyword control over `dremel`'s functions.

4. FALLBACK METHODS

We provided two levels of fallback in case one or more failures prevented the use of the semi-automatic configuration system. The most basic fallback (and the building block for all these configuration methods) was a very simple command-line utility called `confio`. This application could either extract the dynamic configuration data from the database table to a plain text file, or accept a plain text file as input and use its contents to rewrite the dynamic configuration table, then run `CodeGen`. The lowest-level fallback therefore is to extract this plain text file, edit it using any text editor, and recommit the changed version. Simple scripts were provided to make this operation as painless as possible.

It is not as painless as one would like, even so. Some of our DEIMOS stages (the slitmask mechanism for example) are compound or *meta-stages*, higher level constructs which mask a more complicated multiple-motor mechanism underneath. For these, the ordinal/named position map is often the same as for one of the substages being masked. This means that when editing the configuration file manually, the user has to enter the same information in two locations. This is tedious, and if overlooked has unpleasant consequences. As with all manual-edit methods, typos are always possible.

To make manual configuration slightly more user-friendly, we provided a simple GUI to simplify the editing of the plain text file. For the compound stages, this GUI (`tkremel`) is smart enough to "clone" name/position map information from one stage to the other, so that the user does not have to enter data twice. It also offers menus (derived from the database of removable element inventory) which speed the selection of elements and reduce the chance of scribal error: element names don't have to be typed in, as long as they are "known" in our inventory. A `Commit` button in `tkremel` runs the same code that would be run by scanning the magic `COMMIT` barcode (recall that all these commit methods eventually run `confio`, the base application).

We should note in passing that slitmasks do not fit neatly into the `tkremel` scheme. They are manufactured and used in such numbers that a menu of legitimate slitmasks (even for one run) is excessively long and unwieldy. Furthermore, at present neither `confio` nor `tkremel` is capable of generating the FITS chunks describing mask geometry. This would have to be done by extensive manual intervention involving expert knowledge. We will soon remedy this lacuna by providing a smooth command-line alternative for the generation of mask data.

5. EXTRA CONFIGURATION REQUIREMENT FOR GRATINGS

For engineering purposes, we were providing the grating tilt in raw motor counts (`G4TLTRAW, G3TLTRAW`), and in degrees (`G4TLTVAL, G3TLTVAL`). For real observing, DEIMOS users preferred to see grating tilts expressed in central wavelength (angstroms).

We had determined the parameters of the grating equations for the DEIMOS grating inventory, so the conversion of angle to wavelength was not difficult. But the grating in slider 4 (`G4TLTRAW, G4TLTVAL`, and now `G4TLTWAV` as well)[*] might be any of the gratings in our inventory, and this posed a problem. How does the

[*]UCO/Lick KTL keyword nomenclature convention: `G4TLT` is the tilt mechanism for slider 4, whose tilt angle can be expressed as `RAW` motor encoder counts, `VAL` (angle of tilt in degrees), or `WAV` (central wavelength).

dispatcher know the ruling density of the grating, which must be plugged into the grating equation to determine tilt in angstroms?

Because we had a fairly complete description of each grating in our database – linked to the bar code on the grating cell, and including ruling density – the answer was fairly simple. The `CodeGen` utility had to be slightly modified so that it would produce yet another configuration file. This file would associate grating ordinal positions with ruling density. The same old `COMMIT` operation would now include this configuration wrinkle. Restarting the dispatcher would not only refresh its knowledge of raw, ordinal, and named positions, but also of grating ruling constants: the dispatcher now knows the ruling density for position 3 *vs* position 4 and can compute `WAV` keyword values.

We should note that the dispatcher also publishes these ruling densities via three other keywords: `GRATERUL` (the ruling density of the currently selected grating), `GRATLIN3` and `GRATLIN4`. One client which relies on these keywords is the Flexure Compensation System[6] for DEIMOS. Flexure compensation will not work if the dispatcher is misconfigured.

Our basic strategy (single authoritative database, generated configuration files) has proven to be fairly flexible in practise, enabling us to adapt quickly to the additional requirements for DEIMOS grating configuration.

6. MASK COMPLICATIONS

6.1. Physical Differences from other Elements

Gratings and filters are relatively few and are mounted in permanent holders which in turn slide and lock into the appropriate transport mechanism (filter wheel or tiltable sliders). All that's necessary is to label the holder so that the bar code label remains visible when the element is installed and the stage is at the correct unload position. For filters this is trivial: the spine of the filter holder is easily visible when the holder is inserted into the wheel. For gratings it is a little trickier since the loaded grating is buried inside the instrument about 24 inches from the outer access hatch. However, at the extreme tilt necessary to install and remove the grating holder, the bar code on the holder is visible and can be "shot" through the hatch. This is a trifle awkward and we find that most technicians prefer to scan the grating in hand or in its case, just before inserting it into the (pre-selected) slider.

Slit masks pose quite a different problem. The slitmask cassette is a jukebox rather than a traditional multiposition linear or rotational stage. Hence the masks must be stored densely, in a tight stack. This makes it a bit more difficult to read their bar code labels.

In the slitmask cassette, masks are held in a series of slots, supported on one long side, and inserted by pushing on one short side. A pneumatic arm pushes the mask into a cylindrical-section form, where it is held firmly in place by the pressure of the air piston. The motorised cassette mechanism moves back and forth to align any given mask correctly with the pusher arm and the entry into the form.

Since the slitmask has no holder or frame, the bar code label must be applied directly to the mask itself. It must occupy a location that is astronomically useless, *i.e.* already occulted. For various reasons it was not possible to locate the scanner in the "nose cone" in front of the form; so we scan the masks in the cassette rather than in the inserted position.

For our label position we selected the very edge of the mask, at the outermost "upper" corner, an area which falls outside the form inner perimeter and hence outside the image. This "upper" corner is unsupported, and is clearly visible when the mask is installed in the cassette (see Figure 1).

The set of loaded masks in the cassette is spaced just far enough apart that the fixed bar code scanner, mounted "above" the cassette and looking "down" on the tops of the masks at an angle, can scan each label in turn through a certain range of motion of the scissor jack which drives the cassette (see Fig. 1). This scannable location bears little or no relation to the correct insert position for the mask. It's quite variable due to mask bending and curl (see below).

The labels are not large (2.0 inches by 0.20 inches) and the bar code is even narrower since a human-readable equivalent is printed beneath it. However, the scanner beam is also narrow due to the fairly short throw, and

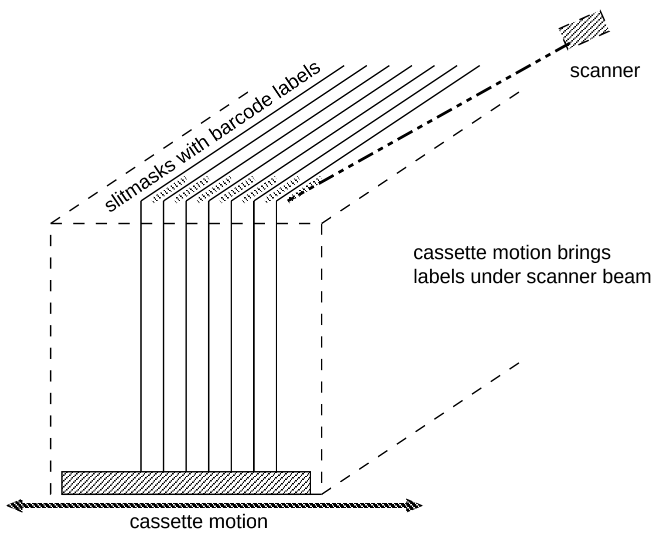

Figure 1. Simplified Drawing of Slit Mask Cassette Mechanism

the position window through which one label can be scanned varies between 7500 and 21000 cassette motor encoder counts. If the scanner scans repeatedly as fast as its software will drive it, and the cassette moves at 1000 motor counts per second (this is rather slow), then each label can be scanned an average of five times as it passes under the beam. Unfortunately the scissor jack drive mechanism produces a nonlinear physical motion for a fixed motor speed, so a speed that is reasonable for labels at the upper end of the motion range may be excessively slow at the lower end of the range. This peculiarity could be overcome by adjusting the cassette speed as scanning progresses, but we have not yet decided that the efficiency gain would be worth the effort.

6.2. Extra Configuration Requirements for Masks

Slitmasks are different from other removable elements in another way: our configuration code is required to do far more than tell us the name of the mask in slot 3. We also require the generation of *FITS chunks* (files containing valid FITS table extensions describing the mask design, as mentioned above) to be appended automatically to each acquired image[*].

Our method of doing this using `dremel` is as follows: after the automatic mask scanning operation, we know the barcode, hence the mask blueprint ID and name, of every mask in the cassette. A simple text file is written to `/tmp` which contains this minimal information. Here is one instance of this ephemeral interchange file:

```
7 7 4 LONG.A
26 26 5 DEEP2.1HS.2143
49 49 7 DEEP2.1HS.3155
27 27 8 DEEP2.1HS.2146
39 39 9 DEEP2.1HS.3140
40 40 10 DEEP2.1HS.3143
25 25 11 DEEP2.1HS.2140
44 44 12 DEEP2.1HS.1100
```

The fields in this file are the mask design ID, the mask blueprint ID, the ordinal position, and the mask design name (for human readability)[†].

[*]There are exceptions: some of our oldest masks predate the canonical manufacturing method and are “undocumented”. Eventually these will be retired and new versions milled which will be properly documented.

[†]The attentive reader will have noticed that the first two columns of this file are identical. If the design and blueprint

`dremel` then forks off a command-line application (`GenMaskData`) which accepts the name of this intermediate file as its argument. `GenMaskData` handles the somewhat lengthy process of looking up the mask design data and creating a FITS chunk for each mask. This process was forked because it takes several minutes and would have prolonged the already over-lengthy mask scanning activity, tying up `dremel` for many minutes. We wanted `dremel` to return to its interactive functionality as soon as possible. Therefore it forks a child process and then periodically checks for that process' termination.

`GenMaskData` produces the mask design files and then renames them according to a convention which identifies them by ordinal position (*Slot.2.fits*, *Slot.3.fits* and so forth). The renamed files are then installed in a known location on the host where image capture software runs.

At image write-to-disk time, the daemon in charge of image capture and disk writing has captured the slitmask cassette position information for inclusion into the main FITS header. This is an integer value, and using this value it constructs a filename (*Slot.N.fits*). If this file is not found in the canonical location then the mask design data cannot be appended to the image. If it is found, then the entire file is appended to the image and this completes the disk write phase of the exposure cycle.

As mentioned earlier, A. C. Phillips has written automatic mask alignment software, using the mask design data found in the image file, which greatly improves observing efficiency. One DEIMOS observing team (Davis and Finkbeiner) has already developed software which uses the mask design data for semi-automated first-pass data reduction and spectral extraction. In fact, reduction of the complex multi-spectral DEIMOS images is virtually impossible without the mask design data. Therefore it is of the utmost importance that this information be permanently, archivally attached to the image, and especially important that it be the correct design information for the mask that was used!

7. MASK LABELLING AND MANUFACTURE

7.1. Labelling Issues

As noted above, slitmasks pose a number of special problems. Not the least of these are their ephemeral nature and the quantities in which they are made and used. Though DEIMOS has several filters and gratings, the total inventory is fairly small; it is not difficult to keep descriptive information in the database current and correct. Gratings and filters are expensive, and so are the machined metal cells that hold them. There will never be tens of them, and those which exist will be used for many years. The inventory is both limited and stable.

Slitmasks, however, are (relatively) cheap and (in practise) disposable. An observer might order 40 or 50 masks made for a single run, and most of these will never be used again. Keeping track of a large collection of ephemeral/disposable components is far more difficult than keeping track of a small collection of long-lived valuable components.

For spectrographs with a small mask inventory (of non-astrometric designs such as aperture masks and pseudo-deckers), technical staff can often identify slitmasks by eye, at a glance. With 30 × 9 inch DEIMOS masks, which may be milled with up to 200 slits of all sizes and angles, only the observer could hope to identify a mask just by looking at it – and even then only if the field was idiosyncratic in some way.

The barcode label is the only reliable ID. It is quite important, therefore, that this label remain securely attached to the mask throughout its working lifetime. A further incentive to select high-quality, highly-adhesive labels was provided by the troubling potential for contamination and interference in the slitmask cassette mechanism, should a label come adrift and lodge anywhere in the mechanism. Despite the ephemeral quality of the masks, they paradoxically require a permanent labelling technology. Happily, the same permanent labels are usable for the stable inventory of glass elements.

Since so much would depend on these labels, considerable time and effort went into their specification and selection. Eventually we settled on aluminium stock, .003 inches in thickness, with the barcode and the

IDs are the same thing, why do we bother to have two IDs in the first place? We have reserved the possibility of generating multiple blueprints from a single design – for use at different seasons of the year, for example. In practise, each design so far has had only one blueprint, but there is no guarantee that this will always be the case.

human-readable equivalent anodized into the surface. The adhesive is a 3M specialty product rated for orbital applications. Our supposition was that if it could survive vacuum and orbital temperatures, it would survive the low humidity and often freezing conditions on Mauna Kea.

So far these labels have held up well. They are vulnerable to gouging or abrasion on extremely rough surfaces; excessive pressure when passing over a sharp point repeatedly can even slice them in half. But they survive hundreds of normal passes through the roller-and-slot mechanism of the slitmask assembly. None of them has so far lost its grip on the metal mask, despite extreme temperature variations and some rough handling.

We did consider the option of engraving the barcode into the mask surface during milling, but abandoned this idea later due to unencouraging early trials. The engraved barcode was hard to read reliably unless a dark pigment was rubbed into the pattern; this was too labour-intensive. There was also some concern about putting extra wear on the fine-gauge mill bits.

Since the bar code label is the critical link in the chain of mask identification and configuration, we must be sure that the label is correctly associated with the mask design during manufacturing. The cost of mis-associating label and design is very high.

7.2. Manufacturing Issues

The process of mask manufacturing is approximately as follows (some details omitted for brevity!):

A *mask design file*, based on an object catalog and reference sky image, is produced in a canonical format by the official "DEIMOS Mask Design Tool" (`dsimulator`) written by A. C. Phillips. Any file produced by other means must be post-filtered into the canonical format: a single FITS file containing multiple FITS binary table extensions with a specified layout and nomenclature.[7] `dsimulator` is IRAF-based; source code will be made available to DEIMOS observers so they can install and use it at their home institutions.

The mask design file is sent to Lick Observatory for milling. The preferred method is to use a "friendly" web page to upload the file. The file is then *sanity checked* for internal consistency of FITS structures. After that, it is *ingested* into the relational database by a command-line application called `ingestMask`. This application knows almost nothing; all of its assumptions about how to interpret the FITS file and map its contents into database tables and fields derive from one of the FITS tables in the design file. The design file contains, in other words, the instructions for its own unpacking. This enables us to make minor changes to the file format (and the destination tables) without having to re-write hard-coded mappings in the ingester application.

Once the file has been ingested, the database is the authoritative locus of the mask design information. From now on, all useful versions of this information will be generated from this sole source. We are currently using the same Sybase RDBMS mentioned in our publications in the References to this present paper. Though this server is admittedly "old technology," we have found its performance adequate for our needs, and its reliability outstanding. Two Sybase servers support Lick instruments at Keck. The active or master server lives at the Lick computing facility on the mainland; the Keck summit server is a carbon copy of the master, refreshed on a regular schedule.

Various *extractors* can now retrieve the design data and format it for different purposes. Typical outputs include: a plain text chronological inventory of all mask designs; a set of human-proofreadable text files containing all the data describing one particular mask; formatted web pages listing an observer's mask inventory and mask milling status; and graphical overlay files for `ds9`[8] which draw the mask design on any direct image. A crucial output product of course is the FITS chunk version of the design data (binary FITS table extension format) to be appended to images. Interactive applications also extract data to produce screen plots of mask designs for human review.

One important extract is a very simple human-readable text file describing only a mask's slit geometry in millimetres (the blueprint). This file is fed to a command-line utility which converts it into CNC mill language[*]. The mill-language files are written to a DOS floppy and hand-carried to the CNC mill, which is capable of reading

[*]We plan to make the CNC mill-language generator capable of reading the FITS chunk files, thus reducing by one the number of file formats involved in the mask milling process.

floppies directly. The mill operator has only to install a mask blank on the vacuum chuck, select a file from the mill's own menu-based interface, and wait for automatic milling to complete.

After a set of masks is made, they are cleaned and inspected, and a couple of locating buttons are riveted into place. It is now time to apply the bar code labels to the clean, oil-free masks. After the bar code labels are applied, the mill operator has to associate the mask designs with the labels. This is the critical step.

Since the mask designs are not identifiable by human eye, leaving the association of label and design until the last minute (and relying on a human operator to "close the loop") obviously presents a risk. We had hoped to eliminate this risk of error by completely automating the mask selection and milling process, but unfortunately had to settle for a more manual, less seamless method.

Originally we had hoped to mill the masks synchronously – that is, to control the CNC mill by software which would feed the mill one mask design at a time. In this scheme, blank mask stock would have been labelled with a barcode; the label would have been associated with the piece of metal long before it was ever milled. The operator would have interacted with a GUI on an X terminal or PC, rather than with the mill's DOS computer.

The operator would select a mask design from a menu on the GUI, then scan the label which was already on the mask blank. The mask would then be milled and cleaned. The operator would then scan either a success or a failure code. Only then would the GUI permit him/her to move on to the next mask. The software would "know" throughout the process exactly which design had just been milled, and into which identifiable mask blank. Thus no confusion would ever arise in the labelling. If a mask were damaged in milling, scanning the `FAIL` barcode at the mill would mark the mask blank as failed and discarded.

This highly-controlled method would have eliminated human error, and so we pursued the idea with some enthusiasm. However, it proved impractical. There were difficulties in pre-labelling all the mask blanks, for a start. The real show-stopper, however, was discovering that we could not control the mill operation remotely. Furthermore, even had we been able to overcome these problems, the mill operators did not like the idea of making, cleaning, and labelling one mask at a time. It was far more efficient for them to make a stack of masks, clean them all, label them all, and so forth. The fully-automated scheme was not going to work.

Eventually we settled on a new scheme. The mask ingester was modified so that as the mask design file is accepted into the database, a two-letter "mill sequencing" code is attached to each blueprint. This simple code remains associated with the mask blueprint until it has been successfully milled and associated with a barcode. As the last step in milling the mask, this 2-letter alpha code is lightly engraved into one corner[†].

The operator who has just made and labelled a stack of 20 masks, can now interact with a very simple GUI which presents the list of mill sequence codes for uncompleted masks: *EG, EH, EI, EJ, EK* and so on. After typing in (or choosing from a menu) the two-letter code which appears on the mask being held in hand at the moment, he/she scans the barcode affixed to that mask. The mill sequence code uniquely identifies the blueprint and design; a new record is minted in a table which documents milled (physical, metal) masks, and in that record the mill sequence code, bar code, and design ID are permanently associated. The mill sequence code is then deleted from the blueprint, as an additional indicator that the mask has been successfuly milled and is no longer pending.

We are considering graycoding or other ways to induce pseudo-randomness in the mill sequence codes, so as to spare the operator the risk of confusing two adjacent codes (*EE* and *EF*, for example). If a typographical error is made (which is likely if codes are too similar), it may be impossible to detect if the codes comprise a gapless monotonic sequence; *EE* is in the current set and so is *EF*. If the sequence were randomized, so that

[†]The reader may wonder why we did not pre-assign bar code labels from our unique preprinted sequence, and engrave the proposed bar code number into the mask corner. There are two reasons. First, we feel that short letter codes like *BX* or *TR* are a little easier to read without error than small labels bearing tiny text like *000026* and *000028*. Second, if a label were damaged while peeling it off the backing sheet or applying it, or if the mask were not adequately cleaned and the label adhesive became contaminated, that label would have to be thrown away. It would then be confusing to have a mask engraved with one barcode but wearing a different label; and elaborate checks would have to be introduced into the software to verify the consumption of an extra label and the disposal of the expected one. We would still have to verify somehow, by scanning, that the label had really been applied; so there would be be no procedural advantage.

EF were legitimate and *EE* were not, then an operator input of *EE* could be instantly rejected. We may find it worthwhile to generate a series of codes whose values do not increase monotonically and whose letters are not adjacent on the keyboard, do not look similar, etc.

The final form of the operator interface, as well as the ergonomics of the workspace in which the masks are cleaned, labelled, and registered as complete, have yet to be determined. The physical mask fabrication process is an iterative problem, which requires a couple of runs' worth of masks to sort out. However, we think the current method of tracking the masks and minimising errors in the barcode assignments is the best we can achieve given the limitations of the mill, and with due regard for the patience and job satisfaction of the operators.

Masks will continue to be milled at Lick Observatory on the mainland until we have developed a smooth, reliable procedure and a suite of appropriate software tools. Only then will the process be exported to Keck Observatory, where a similar CNC mill will make DEIMOS masks in the future.

8. UNFORESEEN DIFFICULTIES AND POINTS OF FAILURE

8.1. Mask Scanning

During testing on the mainland it had become evident that our slitmasks, over several insertion and removal cycles, took on a noticeable curl along their long axis. This curl was always in the same direction: toward lower motor encoder counts when the mask was loaded in the cassette. It displaced the top corner of the afflicted mask, moving the barcode away from the scanning head. This meant that the cassette raw position associated with the successful scanning of a barcode did not map reliably to the ordinal position of the mask in question. The mask scanning code was rewritten to avoid any dependence on foreknowledge of ordinal positions.

The algorithm eventually adopted was crude but fairly effective: the slitmask assembly is instructed to attempt a mask insertion at each position. If the mask position sensors detect that the mask went into place, that slot is remembered as "occupied." If the insertion fails (an error is received), it is remembered as "occupied, but jammed." If the insertion arm travels freely to its limit but the sensors don't detect a mask, then we know the slot is empty.

Having counted the masks and established their positions, the mask-scanning routine now positions the cassette at the upper limit of its travel and starts a continuous smooth motion toward the lower limit, at a constant speed. While the stage is moving, the bar code scanner is instructed to scan repeatedly, as rapidly as possible. The result is a list of scanned bar code values.

Discarding duplicates from this list (since we know that our bar code labels are prefabricated and uniquely numbered), we end up with a simple list of bar codes. Provided that no labels failed to scan, we then need only match the list of occupied slots with the list of scanned labels, and we know the identity of each installed mask. This algorithm worked reliably during mainland testing and during early commissioning.

A few weeks after commissioning, however, a new problem arose. The auto-scanning code failed to identify the masks correctly. It was as if a label had failed to scan; but visual inspection showed that the labels were in good condition, as they had been when checked prior to mask loading.

We found eventually that some metal stock had been used which curled on the short axis of the mask, moving the upper corner in the reverse direction to that of "natural" curl induced by repeated use. One mask in particular had curled so far towards its higher-numbered neighbour that its barcode was wholly occulted, and thus was missed in the scanning pass. The list of scanned labels was shorter by one element than the list of occupied slots.

In seeking an algorithm which would disambiguate mask scanning results properly, we found it was not uncommon for a mask to be successfully scanned at a position more than halfway between it and its neighbour, due to the "use curl" in one direction or the "bad stock" curl in the other. Thus if, of two neighbouring masks, one barcode failed to scan, it could be impossible for software to assign the successful scan to the right slot. We are still working on this problem; the immediate solution is to avoid using metal stock with particularly severe curl in the "wrong" axis. As long as all masks curl in the same (negative) direction, the occulting problem doesn't arise – and if a scan fails for other reasons such as a missing or damaged label, it is possible to infer which mask is the problem.

8.2. Human Factors

Several "human factors" made themselves felt as soon as the instrument was in Hawai'i. It became apparent that we had not adequately rehearsed actual usage of the hand-held bar gun before locating the connector panel, deciding the "right-way-upness" of the filter wheel and its labels, etc. Due to the relative positions of instrument and telescope when DEIMOS is in the observing position (in the El bearing), the Keck staff find it very difficult to perform element exchanges from the right-hand (clockwise) side of the instrument. This work must be done from the left-hand (anticlock) side[*]. However, we had wired the bar gun to the right-hand side, and even with a custom-made, elongated cable it was uncomfortable and inconvenient to use the bar gun on the far side. The connector for the bar gun must be relocated.

We had also been overly optimistic about fully automated mask scanning, based on testing on the mainland. Mask scanning on the mainland always worked perfectly, probably because the set of masks in our possession at that early date were all milled from a consistent batch of sheet metal stock. Later failures to scan bar codes due to a new kind of mask curl were quite puzzling at first, and the code was not engineered to handle such failures gracefully. We made premature assumptions of robustness based on an inadequate range of variation in the masks.

Because the automatic scanning system had worked so well on the mainland, not enough effort was devoted to workaround methods in case of failure. We are still expanding and improving the suite of fallback tools and applications which will permit complete instrument configuration even without a working bar gun, fixed-mount-scanner, etc. This must of course include some way of compensating for an individual unreadable label (whether due to damage, adhesive failure, or pathological mask curl).

We are considering semi-interactive methods which would permit the observer or IS to interact with `dremel` using the main instrument GUI, and assist it to recover from damaged, missing, or occulted labels. This would require adding some more keywords both to indicate the state of the `dremel` software and to "drive" `dremel` using keywords; but fortunately, due to recent developments in our KTL/keyword toolkit, such as Tcl dispatchers, adding new keywords has become much easier.

Lastly we underestimated the strain put on a busy, already-understaffed organisation by the introduction of a new "labour-saving" technology such as semi-automated configuration. While we feel the DEIMOS configuration method will save a great deal of time and effort in the long run, the natural response of oversubscribed Keck staff was skeptical and less than enthusiastic. New features made DEIMOS different from other instruments and therefore implied re-training, learning new procedures, *etc.* New procedures specified by external instrument developers would also be "raw," not yet adapted and tailored to real working conditions – hence inconvenient or inefficient, and possibly unreliable as well.

We anticipated this to a certain degree, and deliberately left the final, surface appearance of the software and the finalising of procedure as post-commissioning tasks. We felt it would be impossible to pre-design and pre-write a successful instrument configuration procedure and software suite without several iterations on-site, with the people who must actually do the work. This too poses its difficulties, as an iterative process requires more involvement (and hence, staff time) from Keck. It appears perhaps impossible to introduce an innovative method of instrument reconfiguration without a substantial initial investment of staff time from the accepting institution: if it is delivered "turn key" without their participation, it will almost certainly be wrong, and will waste their staff time and require redesign (or be abandoned); but if it is delivered in beta release and refined by iteration and consultation, then staff time must be invested in that process.

What was clear after commissioning was that we should have invested more time on fallbacks and workarounds for the automated configuration system, so that these could be demonstrated immediately to the Keck staff. These, being simple GUIs and command line applications, do not require much iteration or tweaking, and

[*]Lick personnel generally refer to the right and left sides of the instrument from the "photon's view", *i.e.* looking into the main hatch while standing in front of the nose cone.

should have been completed and tested before shipping. This would, we think, have raised Keck personnel's confidence level and hence their enthusiasm for the new-fangled labour-saving procedures. We are now trying to address both issues: improvements to the semi-automated method, and improvements and repackaging of the workaround methods.

Although it would be difficult now to reconstruct exact costs and settle the question for sure, there is some reason to wonder whether we actually saved project money by descoping the fixed internal barcode scanners for the other two stages (filters and gratings). It is clear from the description of `dremel` that the software and procedures we developed to support the bar gun (*i.e.* to compensate for the missing scanners) were neither simple nor cheap. Particularly given our need to know the ruling density of all gratings, and our commitment to maintaining the option of automatic *filter/focus discipline*[*], it would not have been attractive to resort to wholly manual, scribal configuration procedures; bar coding was our best bet for reliable configuration. We could not descope the bar coding of these elements altogether.

Descoping the two internal scanners simplified the mechanical design and eliminated some data and power wiring. However, it also resulted in hours of meetings (both internal and with Keck staff) attempting to understand and define the hybrid bar-gun-based configuration procedure; the bar gun did not speak quite the same protocol as the original fixed-mount scanner, so new code was required to support it; and so forth. Whether the project ended up ahead by a few nickels or behind by a few nickels, the general rule seems to have been confirmed: apparent economies in hardware are usually paid for later in software.

9. CONCLUSION

For the last few instrument projects (ESI, PFCAM[†], HIRES Exposure Meter[‡], and DEIMOS) we have pursued a strategy of caching essential information in a single authoritative database, and generating various products (source code, configuration files, documentation) from this sole source. In general we have found this approach productive and have now, for DEIMOS, extended it to the inventory and documentation of removable elements. The strategy to which we committed ourselves over six years ago (*i.e.* the use of a relational database as sole authoritative source of critical instrument and software configuration data) has served us well so far, permitting fairly cheap incremental development of new schemata to support fairly radical new features such as barcode identification of removable elements.

10. ACKNOWLEDGMENTS

DEIMOS was started in 1992 with a Facilities and Instrumentation grant from the National Science Foundation (ARI92-14621) in the amount of $1.79 million. The California Association for Research in Astronomy has so far contributed approximately $7.2 million directly to the project, not including the additional costs of Keck liason and alterations that were needed for the Keck 2 telescope. UCO/Lick Observatory has contributed over $1 million so far in labour. The authors would like to thank Greg Wirth (DEIMOS instrument scientist at Keck) as well as Bill Mason and the summit technical crew, for their active participation in DEIMOS removable element discussions and demos, their many helpful suggestions, *etc.*

[*]This is what the Lick instrument group calls the capability of adjusting instrument focus automatically to compensate for filters of different thicknesses and glass types.

[†]The Lick **P**rime **F**ocus **Cam**era commissioned at the Shane 120-inch telescope on Mt. Hamilton in the Spring of 1997.

[‡]An upgrade to the **Hi**gh **Res**olution Spectrograph designed and built by UCO/Lick Observatory and commissioned at the Keck-I telescope in 1993, the Exposure Meter was installed in the summer of 2000.

REFERENCES

1. S. Faber *et al.*, “DEIMOS spectrograph for the Keck 2 Telescope: integration and testing,” in *Proceedings*, SPIE 2002, Kona, HI
2. W. Lupton, *Keck Software Document 28: KTL Programming Manual*, available in Postscript format by request from present author or may be found online at `http://www.ucolick.org/cgi-bin/Tcl/documents.cgi`
3. D. Clarke and S. Allen, “Using a Data-Driven Model for Instrument Software Development,” in *Proceedings, ADASS 1999*, Kona, HI, 1999.
4. D. Clarke, “Dashboard: a Knowledge-Based, Real-Time Control Panel,” in *Proceedings: 1997 Usenix Tcl/Tk Conference*, Boston, Mass., 1997.
5. Various Authors, *DEIMOS Preliminary Software Design Review* and *DEIMOS Critical Software Design Review*, available by request from present author or may be found online at `http://www.ucolick.org/cgi-bin/Tcl/documents.cgi`
6. R. Kibrick *et al.*, “A comparison of open versus closed loop flexure compensation systems for two Keck optical imaging spectrographs: ESI and DEIMOS,” in *Proceedings*, SPIE 2002, Kona, HI
7. various authors, DEIMOS Web site `http://deimos.ucolick.org` – DEIMOS documentation will be found here, including guides for observers who wish to design and use DEIMOS slitmasks.
8. W. Joye, primary author, `ds9` image viewing and analysis tool. See project home page `http://hea-www.harvard.edu/RD/ds9/` for more information.