DetCom Detector Control Agent
Revision 3.56

by Sidik Isani

2000 October 11th
2002 October 7th (add FAQ)
2004 June 30th (add install and session notes)
2004 October 15th (reserve and expose bg)
Last Revised: 2004 November 12 (TCS interfacing)

The latest version of this document is available on the Web at: http://software.cfht.hawaii.edu/detcom/

Abstract

DetCom is a command line interpreter that controls CCD hardware and generates FITS image files. Typically, DetCom is run under the custom command shell called Director, and therefore these two pieces of software are distributed together, under the GNU General Public License.

Contents

1 Downloading, Building, and Installing

There is separate documentation for Director. Following the instructions to download, build, and install the latest `director-and-detcom` package provides the pieces necessary to read an image. The documentation for Director links to somewhat old detcom versions. Until fresh documentation for the latest Director is complete (version 5), here are the links for the best version director-and-detcom-3.56.tgz. Note that this includes the old Director version 3. An exportable version of Director 5 can be created upon request.

In order to better integrate with a TCS and external instrument control, many sites will need to customize things to provide commands for these systems.

2 Invoking DetCom

Observing accounts should typically be set up by the site administrator so that DetCom is already running in a window. The observer should not invoke detcom themselves if it happens automatically when they log in.

To be fully functional, DetCom requires two other components:

1. The special command shell called Director.
2. A color-capable terminal window (`rxvt` and `xterm` work on most modern unix systems.)

The first step is to launch an `rxvt`, which can be done automatically from the user’s .xsession (in Sidious Linux, the file to look in is `.fvwm-initfunction` in the user’s home directory.) This file must be executable (run `chmod +x .fvwm-initfunction` to be sure) and contains a line such as this:

```
rxvt -e director &
```

Director, in turn, reads a startup file, `~/.director/startup` which should contain instructions to start detcom. A simple line such as:

```
# .director/startup - Startup commands for director shell.
agent start detcom
```

would cause DetCom to be started using the last known DSP code (which it saves in .detcomrc). This command can also be typed manually at the prompt in the director window.

2.1 DSP Code Selection

Rather than relying on .detcomrc to control which DSP code is loaded, most sites prefer to use the startup file to control which DSP code will be loaded. In this case, the DSP code filenames are listed on the “agent start” line which invokes detcom and any boot commands from .detcomrc are ignored. All other commands from .detcomrc are still processed normally. A startup file that controls which DSP code is loaded would look like this:

```
# .director/startup - Startup commands for director shell.
# Uncomment ONE of the following.  Restart director or type the
# command ‘‘source ˜/.director/startup’’ in the director window
# after changing this file.

#agent start detcom some_ltb.lod some_lub.lod
agent start detcom other_ltb.lod other_lub.lod
```

Finally, the “boot” command can also be used to quickly switch from one DSP code to another (but engineering mode must be active.)

2.2 Which DSP code is loaded?

Regardless of the method used to boot the DSPs, the currently active code can be seen on “info” lines 8 and 9. If they are not visible (below the director prompt), issue the command “infosize 9” to reveal them. The command “boot status” will also display the same information. For example:

```
> boot status
ltb: /home/ccd/DSP/psu4k/psu4kltb/psu4kltb_a.lod(#185) OK, 379 symbols.
lub: /home/ccd/DSP/detcom/detcomlub/detcomlub.lod(#57) OK, 276 symbols.
```

The numbers in parentheses are the serial numbers which are incremented by the Makefile each time a .lod file is rebuilt.

3 General Commands

This document describes commands which can be typed in the “director” shell window (identified by its white-on-blue “status bar.”) Some of these commands may be available in other contexts, but others may not. To be sure, enter these commands in the “director” window itself. From other shells (such as bash, or tcsh) send them to director with the help of the “clicmd” utility. For example:

```
bash$ clicmd go          # Look for the feedback in the director window.
> go                     # Typed at the director window itself.
```

Graphical user interface components may send these same commands, tied to button presses or menu selections. There are many possibilities, but typically they result in invoking the “clicmd” program, or making a call to the “clicmd(”command...”)” C-language interface.

When a graphical interface action results in a command being sent to the director shell window, the command and its feedback are echoed there, just as if the user had typed in the command.

3.1 `help`

This displays a summary of the commands available in the director shell window. If no specific “help” script has been installed (which may be done to document additional instrument-specific commands) then the “help” command will go directly to “detcom” and display only the detector control commands. Here is an example of the output:

```
*---[ Detector/Instrument Setup Commands ]-----------------------------------*
defaults           : Reset filename, etime, etype, and raster settings
mef on|off         : Save multi-extension (on) or separate FITS files (off)
header <c> <val>   : Set keyword ‘<c>’ (comment,observer,object) to ‘<val>’
nheader <args>     : Set any keyword; <args>=‘index CARDNAME value comment’
filter <name>      : Moves a new filter into beam
raster <rspec>     : <rspec>=‘xc yc xs ys [xb yb]’ (See ‘raster help’)
etype <type>       : Select <type>=OBJECT, BIAS, DARK, FLAT
etime <sec>        : Set exposure time; <sec> can be a floating point number
*---[ Starting and Stopping Exposures ]--------------------------------------*
go [n]             : Start (n) exposure(s) with current etime and etype
stop               : Shorten an exposure (close shutter and readout now)
break              : Complete current exposure, then break out of sequence
abort              : Use with caution, and only when all else fails!
*---[ Other Commands ]-------------------------------------------------------*
cd [<directory>]   : Change/Show current working directory for images
ls [<files>]       : List files in the current working directory
say <message>      : Broadcast a message to other users on-line
*---------------------------------------------------------------------------*
```

In a complete observing environment, the DetCom help command is typically overridden by a “help” script (installed as`~/.director/bin/help`). This script displays the DetCom command set, plus any other commands (to control an instrument, telescope, etc.) that are available. If such a script is in place and you want to see the actual detcom help screen, you must type `detcom.help` instead of just `help`. (Note that the help script can be set up to do this for users, by including the line `clicmd detcom.help` inside the script.)

3.2 `mode`

The mode command should not be used by observers. If DetCom is not in “OBSERVING” mode, then the current mode will be displayed at the left side of the blue status bar (above the prompt.) If this line begins with the current filename, then the current mode is “OBSERVING” mode.

3.2.1 OBSERVING mode

Observing mode is entered by typing `mode observing`. The `defaults` command also selects OBSERVING mode. In this mode, potentially dangerous commands are disabled and not recognized by the system. These include commands which:

- Write controller DSP memory locations directly.
- Change the DSP code filename loaded in the controller.
- Power on and off or change voltages on the detector.

Still other commands disappear from the help screen, but are still accepted because it is harmless for the user to enter them, and their functionality is sometimes required in scripts.

Observing mode also enables automatic retries and recovery from certain faults. (This behavior could make trying to debug the faults difficult in a lab, so that is why they are disabled in engineering mode.) Automatic state changes, such as enabling a background "sweeping mode" that cleans the detectors while idling, and a timer which relaxes spring tension on the shutter (if applicable to your shutter) are also enabled in observing mode.

3.2.2 ENGINEERING mode

Given the differences between observing and engineering mode described above, engineering mode should not blindly be used for all "engineering" work. If someone has their fingers in the instrument, it is certainly a good idea to enable engineering mode to prevent automatic actions from being taken, especially if your shutter has the "sleep mode". When trying to trap a specific fault, engineering mode is also useful otherwise DetCom may simply reset the fault and try to continue before you can trap it. But for many other tests, it is advisable to use observing mode in order to test the system in as much a way as it will be used on the sky as possible.

3.2.3 SAFE mode

DetCom will remember the previous mode across restarts. If DetCom is started for the first time, or its state file has been removed, it reverts to a special default mode. SAFE mode is DetCom's way of waiting for the user to choose a mode (either `mode engineering` or `mode observing`. The reason for having a SAFE mode is that ENGINEERING mode is not safe for an observer, for obvious reasons, but OBSERVING mode is also not always safe in an engineering environment, where the user may not want automatic initialization or recovery actions to be taken on a lab system as would be done for an installed observing system.

4 Engineering Commands

Many engineering commands take an argument describing a target "board" or endpoint inside the controller. For SDSU systems, this argument is typically either:

- `ltb` - for "Leach Timing Board" or
- `lub` - for "Leach Utility Board"

If multiple controllers are connected, the command will apply to all of them unless a `0` or `M` is applied for the first (or master) controller, a `1` or `S` for the slave or second controller, or `2` and so on if a system as more than two controllers. To apply a command to the slave utility board, the argument

```
lubS
```

can be used. To apply a command to all boards in the master controller, an argument of just

```
M
```

is also possible. This syntax is used where-ever the help screen specifies `''LB''`.

4.1 `tdl`

“Test Data Link” is the first test to use in determining if the controller is connected. While this is an engineering command, it is also available in observing mode because it is harmless. Many other commands include a quick “tdl” check at the beginning just to verify that things are connected. Since all known boot proms in existing timing boards and utility boards support TDL, for SDSU, this command is always available, even if specific DSP code has not yet been loaded and booted in the controller.

By default, `tdl` checks all boards with several data patterns. The first argument can be used to single out a specific board, and an optional third argument can be used to specify a data pattern to write.

4.2 `boot`

This command is also known as `download`. It loads the last known DSP code files to the controller and begins execution of the code. In engineering mode, this command also takes optional arguments to specify a board, and a filename of a “.lod” file to load. In observing mode, this is disabled for obvious reasons, and boot can only be used by itself to retry booting the already selected code. If `boot` is able to determine that the code is already loaded, it returns immediately. As with `tdl`, many higher level commands contain a quick call to `boot` to make sure all DSP code is loaded before proceeding, so the observer typically never uses this command.

4.3 `symbol`

This command can be used to read the current value of a DSP code symbol. The symbol names used here can also be used as address arguments to the commands below, instead of specifying the numeric value. This mode of addressing memory locations in the DSP is much safer than using the actual address, since the addresses could change or shift if the DSP code is modified.

4.4 `rdmem`

This takes two arguments. An `LB` specification as described at the top of this section, followed by an address argument. The address argument can either be a DSP memory location specified as MEMORYSPACE:HEXADECIMAL_ADDRESS, or the name of a symbol, like `OPTIONS`.

As an example, to read the OPTIONS bits of all endpoints in all controllers:

```
rdmem * OPTIONS
```

Since the rdmem command is harmless, it is enabled but not advertised in observing mode.

4.5 `wrmem`

Like `rdmem`, the first two arguments to this command are the endpoint and the address. A third argument must be specified giving the new value to be written to the specified memory location. This argument can either be a numeric value, or the name of an “N:” symbol in the DSP code. “N:” symbols in the DSP code can also be used as offsets, and when combined with a fourth argument to `wrmem`, this command can be used to turn on or off specific bits at a memory location. The special keywords `ON` and `OFF` have been defined as “all bits on” and “all bits off” to facilitate this. An example illustrates best:

```
wrmem ltb OPTIONS ON 1<<CUSTOMOPT
```

would turn on a CUSTOMOPT bit (CUSTOMOPT would be an “N:” symbol with a value of 0..23) in the OPTIONS of all timing boards.

DetCom uses this method internally to manipulate the options bits it knows about (test-pattern, disable shutter for darks, calibration LEDs) so that other bits are left as the user set them.

Because of potential risk to the detectors, use of this command is disabled in observing mode.

4.6 `save`

Normally, DetCom saves data at the same time a `go` is issued. If all data could not be saved successfully, another attempt to save the last-read data to disk can be made by typing `save`. DetCom does not currently check that there is enough disk space to save the next readout before starting the exposure, so the `save` command can be useful when a disk fills up. The exposure is not lost until a new “go” command is issued. This command is not advertised to users, but it is also available in observing mode.

4.7 `auto save on|off`

This engineering command can be used to disable writing of a FITS file during a readout. If this mode is enabled during testing, a single frame (the last one read from the camera) can still be saved to disk by typing `save` manually.

4.8 `auto clean on|off`

This engineering command enables/disables the “sweeping” mode which cleans the detector constantly during idle time.

4.9 `auto voltage on|off`

This engineering command enables/disables testing of the detector voltages during idle time. Voltages are displayed to the user in the “info” bars below director’s shell prompt. These same lines are used to display file-saving progress during a readout.

4.10 `cmd`

This engineering command (not available to observers) can be used to send any arbitrary three-letter SDSU command to any endpoint in the controller. The first argument is the board specification. The second argument is the three letter command, and an optional fourth argument specifies the expected response (use this if the command returns something other than “DON”.) An optional fifth argument specifies the maximum time in milliseconds to wait for a response. (The default is 3000 milliseconds.) For example, DetCom uses this command internally to clean the detectors:

```
cmd ltbS CLN
cmd ltbM CLR DON 8000
```

(Of course the user just types `clean`, but this is what happens inside.) To send a command which expects no response at all, use ” (the empty argument) for the their parameter:

```
cmd ltb RDC ’’
```

(This usage is very rarely needed.)

5 Exposure Configuration

5.1 `defaults`

This is a good command to include at the top of scripts, or when the system is in an unknown state. Note that even if DetCom has just been started from scratch, all previous states and modes are restored from the last known settings, so it is generally a good idea to start by sending `defaults` to reset things.

The command will reset the filename to a default pattern, select a full raster and a default object exposure time.

5.2 `mef on|off`

This command selects whether mosaic data should be saved in a sub-directory with one basic FITS file per amplifier, or whether a single Multi-Extension FITS file should be generated with IMAGE extensions for each amplifier.

If the filename currently displayed in the blue status bar contains an asterisk ("*") then mef mode is *off* and multiple files will be saved.

5.3 `reserve`

Versions of DetCom through 3.54 kept a minimum of 100 COMMENT keywords in each primary and each extension header unit of a FITS. These COMMENT lines are intended for downstream programs that wish to add (not modify existing) keywords. The reason for inserting these COMMENT lines is that the libfh FITS library is able to use them to add other keywords to the file after DetCom has closed it without reallocating the whole file. This is just a matter of efficiency, and a limitation of FITS.

As of version 3.56, this default has been raised from 100 to 200, but the `reserve` command has also been added. To switch back to the previous setting that was used by all DetCom versions through 3.54, use:

```
> reserve 100
```

If it is desirable to have DetCom produce FITS files with no space reserved for downstream use, this is also possible now with 3.56, by using the command `reserve 0`. Since it is only CFHT libfh FITS library, which uses these special reserve COMMENT lines, sites not using that library or adding more keywords downstream may want to use the 0 setting.

The current status of the `reserve` command is echoed when the command is typed with no parameters, and it is remembered in the `.detcomrc` file across restarts.

Ideas exist to make future versions of DetCom collect arbitrary keywords from the Status Server shared namespace which could eliminate the problem of needing to guess or calculate how much extra space to leave.

5.4 `header`

The valid choices for the first argument of this command are `comment`, `observer`, and `object` followed by arbitrary text to be inserted into the corresponding keyword in the FITS header.

The arguments `_runid`, and `_piname` (note leading underscore) are also accepted.

5.5 `nheader`

This command takes four arguments, and can be used to add or change arbitrary keywords that will appear in all FITS data. The first argument is a sorting number which controls where in the header the keyword will appear. These sorting numbers must be chosen relative to the ones DetCom uses internally for the keywords it generates. (See `det_data.c`.) A negative sorting number means the keyword will appear in the next FITS file saved (only) while positive sorting numbers cause the keyword to be added until an `nheader reset` command is issued.

The second, third, and fourth arguments specify the keyword name (up to eight letters), the contents of the keyword formatted as it should appear in the FITS header, and a comment to be placed after the slash ('/') character in the keyword line.

Many sites have strict requirements about FITS header formatting, and may not wish to have general users sending the `nheader` command.

5.6 `raster`

All systems accept `raster full` which selects the maximum number of pixels available. The other options to the raster command are system-specific, and the command `raster help` will list other options currently available for your system.

5.7 `etype`

A single parameter must be supplied selecting one of the following exposure types: “object”, “dark”, “bias”, “focus”, or “flat”. Note that some systems may implement other higher level exposure types, implemented as director scripts, which involve interaction with other subsystems such as telescope and/or instrument control. An example of how this is done for the CFH12K should be added here, in this manual.

The current etype will be stored in the FITS headers of subsequent exposures, but also sets everything needed for the current exposure type mode affecting the telescope, instrument, and detector control.

5.8 `etime`

Accepts a single parameter giving the “exposure” time in seconds. If `etype dark` is selected, this value is the integration time with the detector shutter closed, and if `etype bias` is selected, this value is ignored, but must still be left set to a value in the valid range of the detector system. Values can either be in MM:SS.ss format (minutes:seconds.fraction) or just in seconds. The range of legal values and the resolution depend on the detector system in use.

```
etime 600
etime 10:00
```

These commands are equivalent. Both select a 10 minute exposure.

6 Starting and Stopping Exposures

6.1 `go`

The commands in the previous section only set up DetCom for the next exposure. Except for filter motion (if applicable), which begins immediately, nothing else happens to the instrument until a `go` is issued in the director window. Go takes an optional argument giving the number of iterations.

6.2 `stop`

During the exposure phase of a `go`, it is possible to type `stop` in the director shell window. This is a special command which is processed immediately, as an interrupt to the current operation. (Other commands, such as another `go` will be queued up and processed after the current go has completed.)

As soon as the `stop` command is issued, the current integration will be ended (early) and the readout will begin. This is useful if clouds move in and the observer wishes to end an observation early, or if an incorrect exposure time was entered. It is not currently possible to modify the exposure time requested once an exposure has been started.

6.3 `break`

The `break` command is another special command, processed as an interrupt. For single exposures, it has no effect, but if multiple exposures are in progress, it causes the full current integration to complete but the subsequent one will not begin.

6.4 `abort`

This is the third special interrupt command, and should be used with *extreme caution*. It expresses the users desire to stop whatever is happening immediately (which is not always possible). Note that this could include aborting the saving of a previous exposure, which may still be in progress.

If a readout is in progress, the `abort` command is currently disabled because in many cases the DSP has been left in an unpredictable state if aborted at that time.

7 Other Commands

7.1 `cd <directory>`

When typed in the Director shell window, this command can be used to change the working directory where DetCom saves FITS files.

Without any arguments, the current working directory is displayed.

7.2 `ls`

When typed in the Director shell window, this command lists files and subdirectories in the current directory.

7.3 `say <message>`

The director shell window can be “cloned” by others with access to the local machine (and not necessarily having access to the observing account.) The “clone” window is launched by typing

```
director -k observer
```

where `observer` is the username of the observer who is running the “master” director. The user running `director -k` need not be the same, but they will have permission to only send `say` commands unless they know the `observer` password.

The `say` command can be used by engineers and observers to send messages back and forth. These messages can be sent at any time, even if DetCom is processing a command or in the middle of an exposure.

8 Special Exposure Sequencing Commands

8.1 `preexp`

This is an internal command. It communicates any necessary option bits and readout parameters to the controller. It is automatically run as the first step of a “go” command.

8.2 `clean`

This is an internal command. Normally in observing mode the detector is in a constant cleaning mode called “sweeping.” If this sweeping mode has not flushed the detector at least once, a “clean” runs automatically as the second step of a “go” command.

8.3 `expose`

This is an internal command. It performs the integration step and returns when integration is complete if typed with no parameters. The form “expose auto” is the third step of a “go” command. In this form, its behavior is dependent on whether “expose fg” or “expose bg” mode is active. The default mode is “expose fg” in which case “expose auto” is identical to “expose” and returns when integration is complete. If “expose bg” has been issued, however, then “expose auto” returns when integration starts so that other commands can be issued to Director which will overlap with the exposure step. Once such an exposure is started, “expose wait” can be used to perform a blocking wait for the end of the exposure, but even if this is not used, detcom will proceed automatically into the readout phase when the exposure completes. The following table summarizes `expose` options:

Table 1: Expose Command Options

expose	Integrate and return when complete
expose fg	(default) Causes subsequent `expose auto` to behave like `expose`.
expose bg	Causes subsequent `expose auto` to return immediately.
expose auto	Begin integration. When it returns depends on `fg|bg`.
expose wait	Block until current integration (if any) is complete.
expose poll	Return immediately. FAIL means integration is still in progress.

8.4 `readout`

This is an internal command. It is normally triggered immediately after an “expose auto” by the “go” command (even in “expose bg” mode.) Depending on the setting of “auto save”, a FITS file is usually saved to disk in parallel with the readout operation. Similar to “expose bg—fg”, a “readout fg” and “readout bg” mode exists. If “expose bg” is active, “readout bg” is implied and the setting is irrelevant. For the normal case of “expose fg”, however, “readout bg” causes a “go” command to return successfully at the start of the readout instead of the end. A “readout wait” can force a blocking wait for the end of the readout. Built-in DetCom commands that would conflict with a readout in progress already include an internal call to “readout wait” so it is only necessary to call “readout wait” explicitly before any non-detcom operation (such as slewing) if you do not wish for it to overlap with readout. Readout options are summarized in the following table:

Note that there are currently slight differences in the options between `readout` and `expose`, although the concept is the same.

Table 2: Readout Command Options

readout fg	(default) Subsequent `readout` returns when complete.
readout bg	Causes subsequent `readout` to return immediately.
readout auto	Begin readout. When it returns depends on `fg|bg`.
readout wait	Block until current readout (if any) is complete.

8.5 `waitdata`

This command does a blocking wait for any save-to-disk operation that might be in progress. If none is in progress, it returns immediately.

9 Frequently Answered Questions

9.1 DetCom complains that ”Readouts may fail because DetCom is not setuid root.”

Log in as root and run the following commands on your detcom binary:

```
chown root detcom
chmod u+s detcom
```

Also, detcom should preferably be a on a local disk. If it exists on an NFS server, make sure the directory is exported with ”no_root_squash” so it is possible to have setuid permissions.

For developers, if you want to rebuild DetCom and have it automatically installed with setuid-root permissions, make sure you have properly installed the ”setuidinst” tool from the source tree. If this program has setuid-root permissions, it will automatically make detcom setuid-root each time you do a ”make install” in the detcom source directory.

9.2 open(”/tmp/detcom_memory_mapped_readout_buffer”) fails

If you have problem with this file, it usually means the size or permissions of this file are wrong. Log in as root and remove it, then restart DetCom. DetCom will always create this file if it doesn’t exist. (But first, make sure you are running DetCom with setuid root permissions. See question 1.)

9.3 DetCom isn’t reporting filter position.

This will happen whenever there is no DSP code downloaded, no second (slave) controller, or if the slave utility DSP code does not define the symbol ACE_WVS. See also http://software.cfht.hawaii.edu/dspsymbol.html

9.4 DetCom isn’t reporting the CCD temperature.

This will happen whenever there is no DSP code downloaded or the utility board DSP code does not define the symbol A_CCDT. To properly support detector temperature feedback in DetCom, DSP code should also define DETTEM_C1, DETTEM_C2, DETTEM_MIN, DETTEM_MAX. See also http://software.cfht.hawaii.edu/dspsymbol.html

9.5 DetCom isn't reporting DSP voltages.

This will happen whenever there is no DSP code downloaded or the utility board DSP code doesn't define one of: K_HV I_HV C_HV T_P15 K_P15 I_P15 C_P15 T_M15 K_M15 I_M15 C_M15 C_P5. See http://software.cfht.hawaii.edu/dspsymbol.html for details.

9.6 How do I view messages that have scrolled off the screen?

When running Director, the terminal's scrollbar will not function correctly (and should, in fact, be disabled.) Use the PageUp and PageDown keys to view previous messages. You can also resize your window to view more lines/columns at once.

9.7 Why are FITS files larger after upgrading to 3.56?

DetCom has always reserved at least 100 keyword slots using COMMENT lines at the end of each header unit. As of 3.56, this number has been changed to 200, but a new command "reserve" has also been added. See the section on `reserve`.

9.8 How can Instrument Configuration be scripted?

This is more of a Director question. Here is a summary of one way to do it. Let's assume that you have created a script or program that moves a filter. Let's assume the command is called **filter**, and it does two things. First, it can be used to begin moving a new filter. The syntax to select the V filter, for example, might be:

```
filter V
```

Second, if you can make your filter command non-blocking, motion of the filter can be overlapped with other operations (such as reading the detector or slewing the telescope) and a command to wait should exist. For example:

```
filter wait
```

Again, this **filter** command can be a C program, a Perl script, etc. To make **filter** a command recognized by the director shell, simply copy it into this directory:

```
$HOME/.director/bin/
```

Now you'll want to do one final thing to ensure that the filter is not moving when you tell DetCom to **go**. The way to do that is to make **go** itself a script, instead of a command sent directly to DetCom. The **go** script will look no different to the user than the previous DetCom go, if it contains something like this:

```
#!/bin/sh
#
# go - replacement for DetCom's internal go.
#
clicmd filter wait     # Make sure filter is not moving.
clicmd detcom.go "$@"  # Continue by executing detcom's go.
```

clicmd is a utility that gets installed with Director. Install this script as:

```
$HOME/.director/bin/go
```

and make sure it is executable. Next time the user types **go**, it will run this script instead of sending **go** to detcom directly.

9.9 How can Instrument Keywords be added?

At CFHT, keywords are collected in a Status Server shared namespace. A slightly slower but simpler way to get your own keywords into DetCom's FITS headers involves DetCom's **nheader** command. This command should be used before a **go**. Suppose that the **filter** command in the examples above was able to report the current filter position if invoked with no arguments. In that case, the **go** replacement could be modified to look like this:

```
#!/bin/sh
#
# go - replacement for DetCom’s internal go.
#
clicmd filter wait     # Make sure filter is not moving.
FILTER=‘clicap filter‘ # Read current filter position.
clicmd nheader -1010.1 FILTER \"$FILTER\" \"Comment for header\"
clicmd detcom.go "$@"  # Continue by executing detcom’s go.
```

The choice of the floating point number 1010.1 is arbitrary (but controls the order in which keywords appear in the final FITS header.)

The negative sign on 1010.1 means this keyword will only go into the next FITS file. You may wish to drop the '-' and the keyword will be added until **nheader reset** is used to clear them all.

The argument after the number is the FITS keyword name (up to 8 characters).

The third argument is the filter position, as reported by your filter command, captured by **clicap** in the line above.

The fourth argument is an optional comment string for the FITS card.

The backslashes before the quotes are necessary because the /bin/sh shell eats one level of quotes.

9.10 How can Telescope Keywords be added?

You will first require an implementation-specific way to get current telescope information into script variables. Once you have that, exactly the same method used above to get a FILTER keyword into the header can be used, by adding to the **go** script:

```
#!/bin/sh
#
# go - replacement for DetCom’s internal go.
#
clicmd filter wait     # Make sure filter is not moving.
FILTER=‘clicap filter‘ # Read current filter position.
TCS_RA=‘get_tcs ra‘    # Query TCS for RA
TCS_DEC=‘get_tcs dec‘  # Query TCS for DEC
clicmd nheader -1010.1 FILTER \"$FILTER\" \"Comment for header\"
clicmd nheader -522.1  RA     \"$TCS_RA\" \"Object right ascension\"
clicmd nheader -522.2  DEC    \"$TCS_DEC\" \"Object declination\"
clicmd detcom.go "$@"  # Continue by executing detcom’s go.
```

See comments in the filter question, and also the section on nheader itself for more hints.

9.11 What mechanisms exist for post processing steps of FITS?

Using the **go** script above, it is trivial to add post processing steps to the end of this script. Those steps, however, can take up value observing time since Director is single threaded and will not allow the user to start the next **go**.

To get around this, director looks in `$HOME/bin/` for a different script, called **endgo**. If this script exists, it will be executed once per FITS file created. The name of the FITS file will be passed as an argument, and the current working directory will be the one where the FITS file was saved by DetCom. This **endgo** script runs asynchronously to observations, and can take as long as it wants to process the file.

At CFHT, endgo is used to display the image, pass the file on to the Elixir processing pipeline, and start the process to make a tape backup of the image.