RETROCAM Command Scripts

2006 December 18

General Commands:

Script commands for RETROCAM are shown in the table below. Optional arguments are enclosed in []'s, whereas required arguments are in *italics*. Commands are case-insensitive, but arguments like target names, filters, and filenames preserve their case.

OBJECT *target name*	Setup for an OBJECT image named "*target name*". Multi-word strings do not need to be enclosed in quotes
FLAT *target name*	Setup for an FLAT FIELD image named "*target name*"
DARK *target name*	Setup for a DARK image named "*target name*"
BIAS *target name*	Setup for a 0^s BIAS image named "*target name*"
EXPTIME *tsec*	Set the exposure time to "*tsec*" seconds (1..1048s)
IMGTYPE *type*	Set the image type, *type*=(OBJECT, FLAT, DARK, BIAS) without changing the target name
FILENAME *rootname*	Set the root filename for subsequent images Example: "FILENAME r040811.%4.4d". If the format string is omitted, one will be appended.
SEQNO *n*	Set the image sequence number for the next image Range: *n*=1..999 (depending on the format in FILENAME) **NOTE**: SEQNO must **follow** FILENAME in a script as the system automatically resets the SeqNo counter to 1 when the file rootname is changed.
FILTER *num* FILTER *name*	Select filter by number (*num*=1..5) or Select filter by *name* (g, r, i, z, etc.) An *EXACT* name match is required,
GO [n]	Acquire a sequence of "n" images. "GO" by itself is the same as "GO 1".
PAUSE *message text*	Pause script execution and display "*message text*". Pops up a window and waits for the user to press the OK button.
GETNAME *prompt text*	Pause script execution and pop up a dialog box to prompt the user to enter the target (object) name and image type.
GETEXP *prompt text*	Pause script execution and pop up a dialog box to prompt the user to enter the exposure time.
GETFILE *prompt text*	Pause script execution and pop up a dialog box to prompt the user to enter the filename and sequence number.
STOP	Insert a STOP into a script
END	End of the script
# *comment text*	Insert a comment into the script

Remote Telescope Commands:

MIRROR *in*|*out*	Insert/Retract the RETROCAM pickoff mirror The mirror must be **IN** to take images of the sky.
FOCUS *f*	Set the 2.4m telescope mirror focus to "*f*" encoder steps
FOCSTEP ±*df*	Step the 2.4m telescope focus by ±*df* encoder steps
GXY *x y*	Move the MIS guide probe to position (*x*,*y*) encoder steps
GDXY ±*dx* ±*dy*	Offset the MIS guide probe by (±*dx*,±*dy*) encoder steps

RETROCAM Scripts

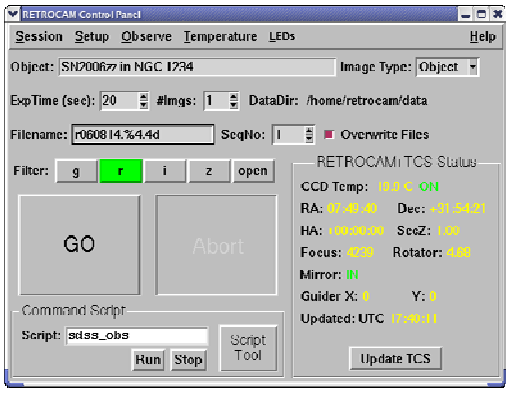
RETROCAM scripts are ASCII text files containing RETROCAM commands to be executed in order from the first to last line of the file. They provide a way to make multi-step or repetitive observations with RETROCAM efficiently. Running a script is equivalent to many separate commands (button presses, box entries, etc.) that would have to be done by hand.

The way to think about RETROCAM scripts is that they fill in boxes and "push buttons" on the main control panel for you. Each of the relevant entry boxes and buttons on the main control panel have corresponding script commands. This means that if a script has no "object" or "exptime" commands, the values that currently appear in the respective boxes on the main control panel will be used for the next image. Similarly, when a script command corresponding to one of the control panel settings is executed, you will see the control panel change in response. Thus a "object" command in a script changes the contents of the Object: entry box on the control panel, a "filter" command lights up the corresponding filter button on the control panel, and so forth. When a script is done executing, the various boxes and buttons on the control panel will correctly reflect the state of RETROCAM at the end of the script.

There are also special script functions that let you make remote telescope and MIS settings (secondary focus, pickoff mirror inert/retract, and guide probe position), and to provide very primitive script "flow control", namely pause and stop. Missing are such high-level functions as subroutines (scripts called from within other scripts), loops, if/else tests, user prompts etc. These functions more properly belong in interpreted programming languages not scripts, and are not implemented. Our goal is to help increase observing efficiency while keeping things simple.

Running Scripts

Command scripts may be executed either from the main RETROCAM Control Panel, or by using the ScriptTool window. Below is a screenshot of a RETROCAM Control Panel, showing the Command Script entry area at the lower-left corner below the exposure GO/Abort buttons.



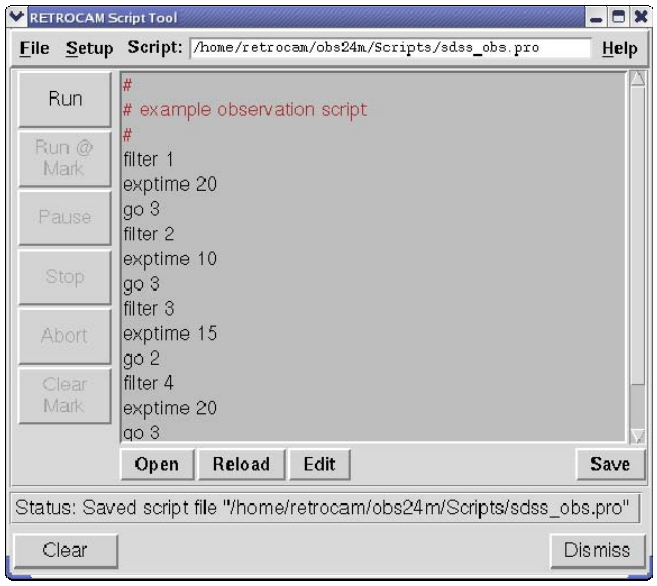
Typing the rootname of your script into the Script: box and **hitting the Enter key** will load the script from the default directory and activate the Run button. You don't need to explicitly type the default directory path or .pro file extension as those will be added as required.

Once a script has been loaded, the Run button will let you start execution of the script, and the Stop button will halt execution of the script after the currently running command is finished (most commands are not interruptable). You can run the same script many times without reloading it by hitting the Run button to re-run the script.

Even with a script loaded, you can always change any of the other settings (object name, exptime, etc.) and take single images (or multi-image sequences) directly using the GO button. Note, however, that "GO" **does not execute a script**: only the Run button does that.

An alternative way to select a script to load is to click the mouse on an empty Script: box and hit the Enter key, which will pop up a script selection dialog box and let you browse around for the script you wish to load. The file browser starts by looking in the default script directory. The default Script directory is defined at startup time in the Observing Session Setup window, and is usually the "`Scripts/`" subdirectory in the obs24m account on the retrocam machine. You an also change the default script directory during a session by using either the "Setup...Observing Session" menu on the main RETROCAM control panel, or the "Setup...Directory" menu in the ScriptTool window.

Pressing the "Script Tool" button on the main RETROCAM control panel will open up the ScriptTool window:



Notice that the full qualified name of the script (full path + .pro extension) is shown in the Script: box on the ScriptTool menu bar. The Run and Stop buttons on the ScriptTool are identical to those on the main control panel. A nice feature of the ScriptTool is the ability to monitor the progress of a script in real-time, as well as to restart a script from an arbitrary line by clicking on the desired line with the left mouse button and then hitting the Run@Mark button.

The ScriptTool window provides a number of script editing and run control features unavailable from the main RETROCAM control panel, but **you do not need to use the ScriptTool to run a script**. If you open the ScriptTool after a script has been loaded from the main RETROCAM control panel, you will see the script in the script viewing area, like shown above.

If you open the ScriptTool window while a script is running, it will show you the current command in progress and let you view the Status: line. If you close the ScriptTool by clicking on the Dismiss button, your script will keep running.

The ScriptTool can also be used to create new scripts, or to modify existing scripts using an editor (vi or Emacs are currently supported), reload a script from disk, and save modified scripts to disk. The Help menu at the upper right-hand corner of the ScriptTool includes a list of commands.

Example Scripts:

Example 1: Take a series of images through 3 filters

This script takes a sequence of images through each of 3 filters (1-3), 3 images each, with different exposure times for each filter. We reference filters by number in this example.

```
#
# Image Sequence Script
#
filter 1
exptime 90
go 3
filter 2
exptime 60
go 3
filter 3
exptime 45
go 3
end
```

Example 2: Take a series of telescope focus images

This script sets the object name and root filename, resets the file counter to 1, and then takes a series of 5 images through the current filter at 5 different focus settings starting at 4250 and incrementing +10 steps per image.

```
#
# Focus Sequence Script
#
object Focus sequence
filename focus%3.3d
seqno 1
exptime 10
# starting focus is 4250, start 100 units below then forward
# to 4250 for the first, then +10 steps before the next 4
focus 4150
focus 4250
go
focstep +10
go
focstep +10
go
focstep +10
go
focstep +10
go
end
```

The focus values for each image will be (4250,4260,4270,4280,4290), and stored in the TELFOCUS keyword in the image FITS headers. The practice at MDM is to always approach focus from smaller to larger numbers. The initial "100 units below" step is more than enough to take up backlash in the focus mechanism. FOCUS sets the focus in absolute units, whereas FOCSTEP increments (steps) the focus relative to the current position.

Example 3: Target Setup and Image Script

Setup RETROCAM to observe a target with a pre-selected guide star in the gri filters. Here we reference the filters by *name*, and use a PAUSE to instruct the observer to setup the guider once the guide probe has been moved into place and hit OK before the observations will begin.

```
#
# Observe SN2006zz
#
object SN2006zz
gxy 1550 6570
PAUSE Setup the guider and continue when stable
# g filter 2x300s
filter g
exptime 300
go 2
# r filter 3x200s
filter r
exptime 200
go 3
# i filter 2x300s
filter i
go 2
end
```

Example 4: Generic Observation Script with User Prompts

This script takes a series of gri images of a field with the same basic exposure times, prompting the observer to enter the target name and the filename/seqno pattern to use.

```
#
# Quasar Lensing Field Exposure gri Sequence
#
getname Name of the Quasar Lens
getfile Filename and Sequence Number to use
# g band images – 3x120s
filter g
exptime 120
go 3
# r band images – 3x90s
filter r
exptime 90
go 3
# i band images – 3x120s
filter i
exptime 120
go 3
end
```

Prompts make it possible to make more generic scripts, avoiding a huge number of scripts for individual targets, or requiring users to remember to change key parameters before executing the script. The observing parameters with GET functions are:

`GETNAME` – prompt for the target name and image type
`GETEXP` – prompt for the exposure time
`GETFILE` – prompt for the filename and sequence number

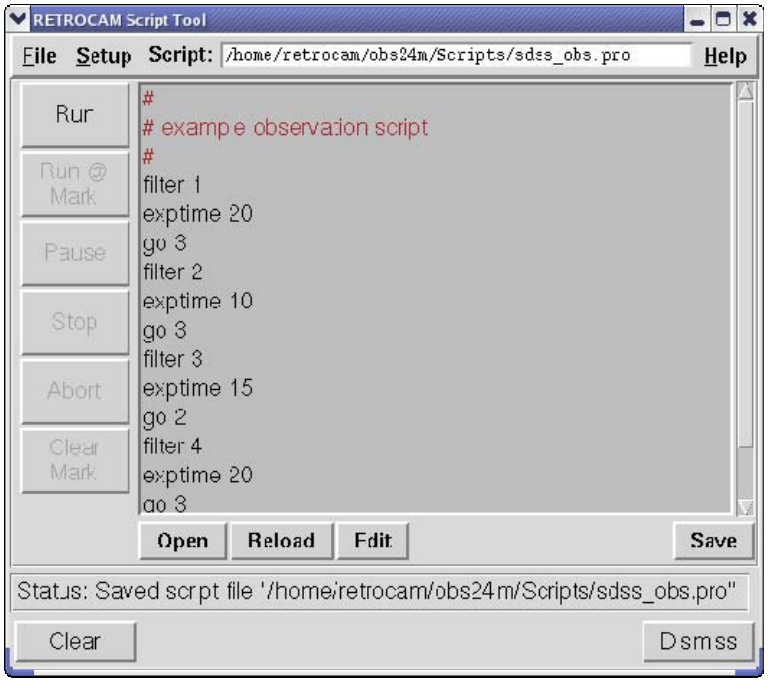
Example 5: Calibration Script with Pauses

This script takes a series of flat-field and bias exposures, using the PAUSE command to prompt the observer to turn the dome screen lights on/off. It also changes the file rootnames for each set of calibration images, and resets the seqno counter to 1.

```
#
# End of Night Calibrations
#
flat g dome flat
filter g
filename gdome.%3.3d
seqno 1
exptime 30
PAUSE Point scope at the dome screen and turn on the lamps
go 10
# r dome flats
flat r dome flat
filter r
filename rdome.%3.3d
seqno 1
exptime 20
go 10
# done
PAUSE Turn off the flat field lamp and stow the telescope
bias End Bias
filename ebias.%3.3d
seqno 1
go 10
end
```

The ScriptTool

The ScriptTool window is shown below with a script loaded and ready to run. The name and full path of the current script is shown in the top bar, along with File, Setup, and Help menus.



Clicking on the Run button at the upper left of the script viewer will start execution from the first line. While the script is running, the Pause, Stop, and Abort buttons at left are activated, the command executing is highlighted in green, and command progress is reported in the Status box.

The 4 buttons below the script viewer let you open, edit, and save scripts:

Open is used to load a script from the Scripts directory. It pops up a standard "open file" dialog box. Scripts are given the ".pro" extension by default. You can also type the name of a script into the "Script:" box in the menu bar at top.

Reload will reload the script file in the top bar from the original file into the ScriptTool, erasing any changes you might have made.

Edit opens either a vi or Emacs window with a copy of the script view window. It does not edit the script file proper, only the copy stored in ScriptTool. You can also press the Edit button with an empty script viewer to create a new script. You can choose the editor to be used from the Setup menu on the top bar.

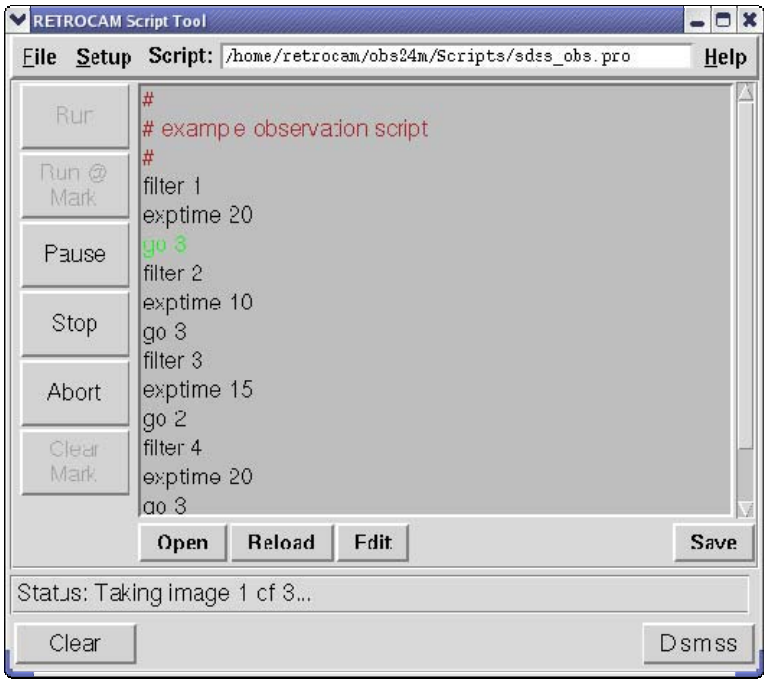
Save will save the contents of the script-view buffer to a file. It pops up a standard "save-as" dialog box. The default filename will be the name of the script in the top bar, but it will not overwrite the file without permission.

The Help button at the upper right on the top bar contains a list of commands and a message describing the ScriptTool.

The Clear button at the lower left erases the contents of the script viewer and resets ScriptTool.

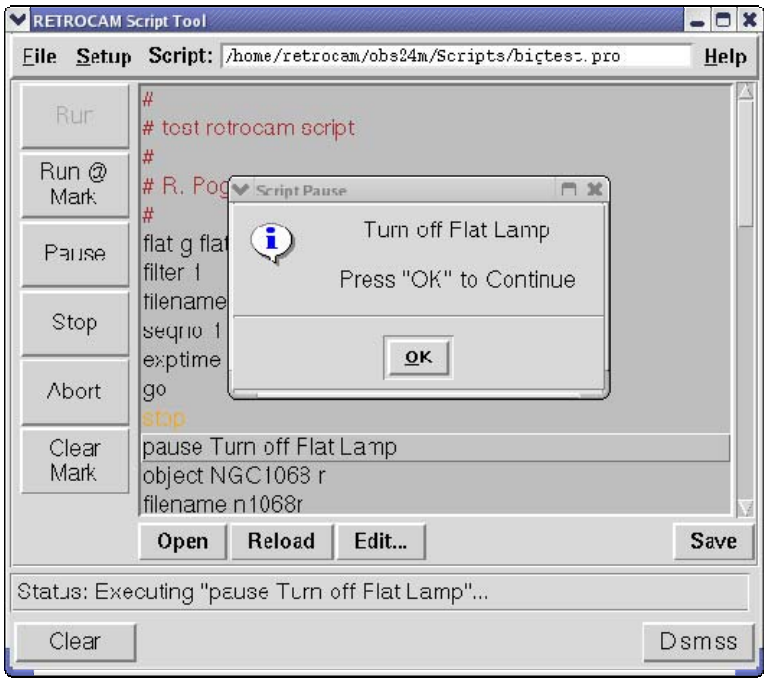
The Dismiss button at the lower right closes the ScriptTool window. When you re-open it, it will be restored as it was when dismissed, including any scripts from before.

The ScriptTool shown is running a script. The current command ("go 3") highlighted in green, and the status bar shows the progress of this command.



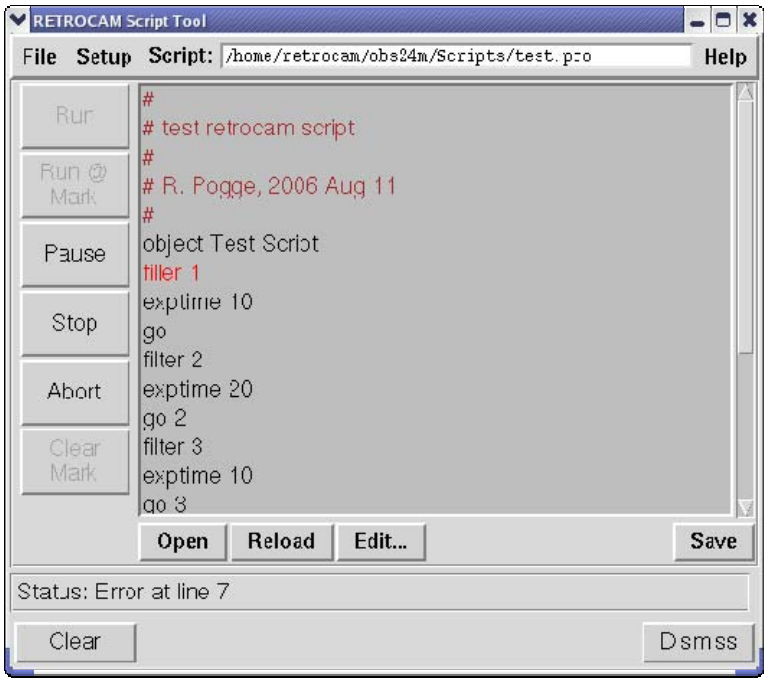
During script execution, the Pause, Stop and Abort buttons are active along the left-hand side. Pause suspends script execution *after* the current command is done and waits for an OK. Stop halts execution after the current command. Abort halts and rewinds the script to line 1.

The ScriptTool below shows the result of a PAUSE command in the script:



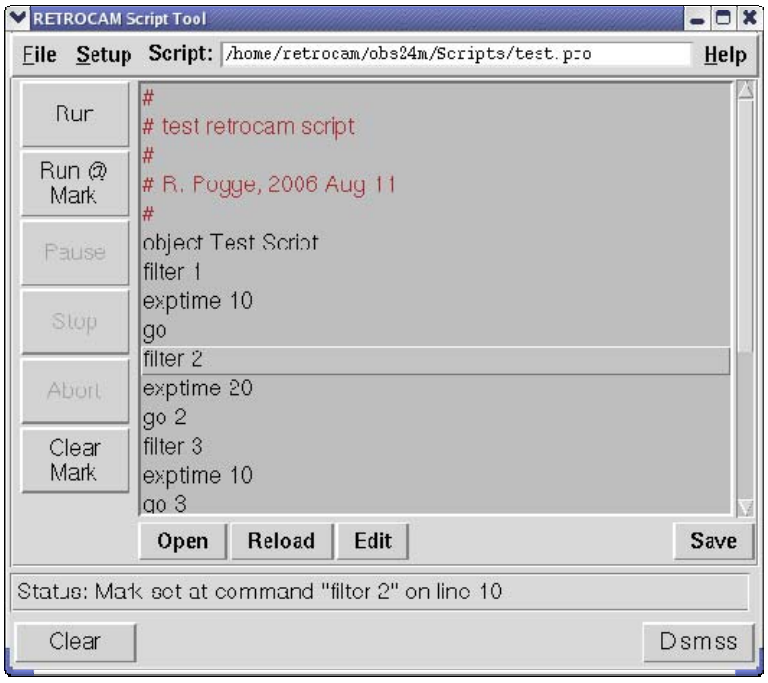
The script will resume when the OK button is pressed with the next command. PAUSE commands can be used to insert instructions for the observer to perform actions RETROCAM cannot do itself (e.g., point the telescope, turn on lights, setup the guider, etc.)

If a command syntax error occurs, ScriptTool will highlight the offending command in red and stop execution. Here the second command "filler 1" is "filter 1" mis-spelled.



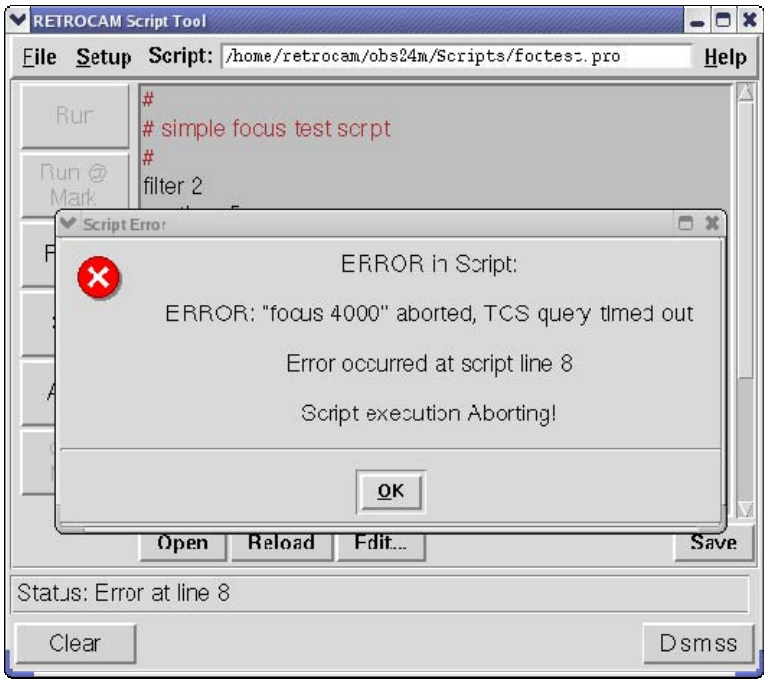
The "Edit" button launches vi or Emacs (select under the "Setup" menu at top left), and lets you edit the contents of the script buffer to correct the error. Note that the original script file will be **unchanged** unless you first "Save" the script buffer to a file.

ScriptTool allows you to start a script from any arbitrary line in the script by marking the desired command by clicking on it with the left mouse button. Once a command is highlighted, pressing the Run@Mark button will start script execution at that command, here "filter 2" in line 10.



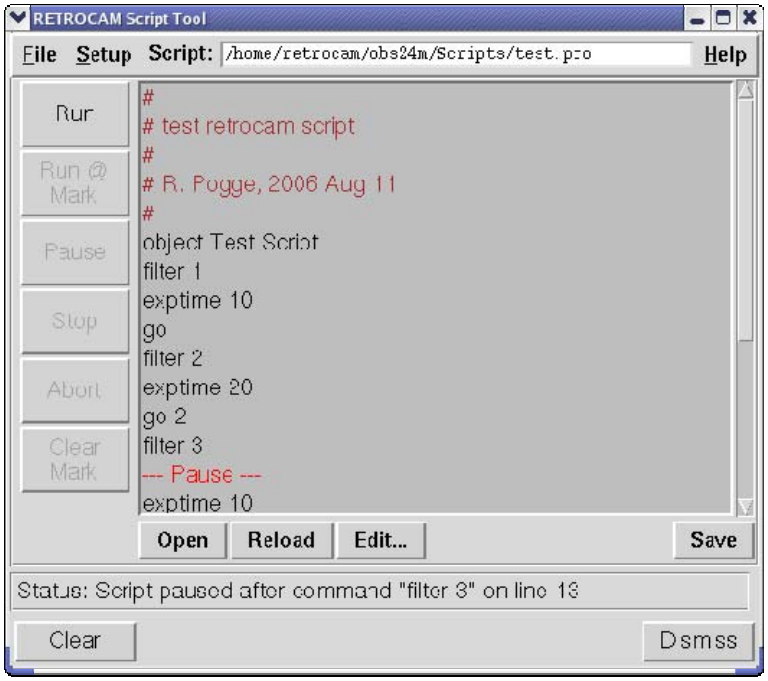
If you hit Run, the mark is ignored and execution will start with the first line in the file. To clear a marked line of a script, press the "Clear Mark" button. This deactivates the Run@Mark button.

If an error occurs during a script, ScriptTool will halt execution and pop up an error window, like shown below:



This particular error (TCS query timeout) occurs when the telescope has failed to respond to a "focus" command because the telescope drives are not active.

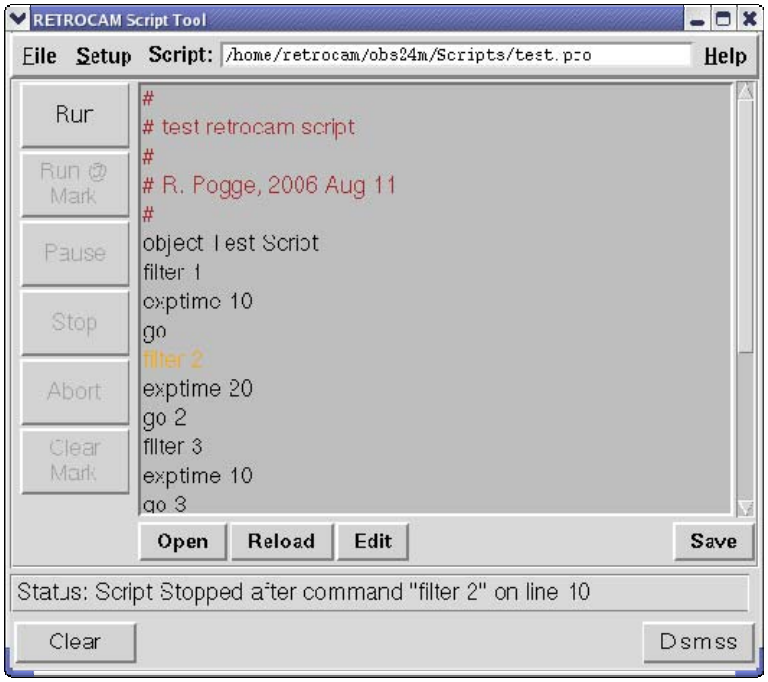
If the observer pauses a running script by hitting the Pause button, ScriptTool waits until the current command has finished executing, and then pops up a dialog box giving you the option of Resuming execution or Aborting.



The location where the script was paused is marked with the red "--- Pause ---" text as shown above. If you elect to Resume, the next command (exptime 10) will be executed, otherwise the script will Abort, the same as if you hit the Abort button.

If a Pause request comes during an multi-image exposure sequence (like "go 3"), the pause will take effect *after* the current image in the sequence is completed. Thus, if doing a sequence of 3 exposures and the Pause button is hit during exposure 1, after exposure 1 reads out and is stored to disk the script will pause and wait until you press the OK button to resume. It will then pickup with exposure 2 of the sequence.

The screenshot below shows the appearance of the ScriptTool window after the Stop button has been pressed and the script has halted.



The command that was completed before the Stop took effect is colored orange, so you know where to set the mark to restart the script from the next command, if desired.