NIRSPEC

UCLA Astrophysics Program **U.C. Berkeley** **W.M. Keck Observatory**

Jason Weiss **March 12, 1999**

NIRSPEC Software Programming Note 13.03
Quicklook

1. Introduction

Quicklook (henceforth referred to as QL) is the real-time data view program used with NIRSPEC. It gives the user the ability to examine a FITS file, whether it be retrieved from memory from a prior source, or from an exposure taken by NIRSPEC and reduced by the data reduction pipeline. In addition to the mere display of an image, it allows the user to perform further data extraction by means of built-in functions such as gaussian fitting and a photometry package. This document outlines the organization of QL and comments on the further enhancement of the package.

2. Organization

QL is comprised of many modular programs, each written in IDL, all of which contain one or more related procedures to be called during the execution of the main program. The program that begins QL is **`ns_quicklook.pro`** in the QL directory (currently **`/kroot/kss/nirspec/ui/qlook`**). It decides how QL is to be run and sets up variables to be called throughout its execution. It then launches the main user interface module in **`ns_display.pro`**. Here, the widgets are set up, and procedures are assigned to the various controls.

3. Important Modules

This section describes some of the more important modules in the QL package, and what they do and when they are called.

A. `ns_quicklook`

1. **Called by: startup scripts**
2. **Procedures called: `ns_display`, `ns_frame_ready_loop`**
3. **Description / Notes**

   This routine it the tool used to launch and maintain the running of the QL package. By default, QL runs with two displays, one for the spectrometer, and the other for the scam. Additionally, it can be run with the following parameters:

- NOSCAM: To prevent the scam window from being launched.
- NOSPEC: To prevent the spectrometer window from being launched.
- GENERIC= '*name*': To create a window with the title '*name*'.
- DRP: To run QL in DRP mode.
- KIDL: Implies NIRSPEC server program is running.
- SIM: To run QL with a simulated server.
- GDIR= '*directory*': Set generic default directory to '*directory*'.
- DBASE=*variable*: Return the display base widget id to *variable*.
- STATUS=*variable*: Return the status window text widget id to *variable*.

This routine also sets up the first of two important user value (uval) structures to be referred to throughout the execution of the program. The uval created here is associated with the base called `infobase`, a base that is never mapped and is created solely for the purpose of maintaining this uval. This uval is often referred to as the `info_uval` whereas the second uval, established in **`ns_display`** is referred to as `base_uval` or simply `uval`. The `info_uval` maintains the widget id's of display bases and the status of the parameters mentioned above. It also contains information about the camera obtained from the server.

B. `ns_display`

1. **Called by: `ns_quicklook`**
2. **Procedures called: `ns_show_current`, `ns_register_template`, `ns_graphwindow`**
3. **Description / Notes**

This routine is the main executing body of the program, and no doubt, the most important one. Most importantly, it creates the widgets used in the main user interface. It creates the buttons and the main menu toolbar used to perform the various tasks QL was designed for. The menu bar is defined in `plot_menu_desc` and its event and draw events are stored in **`ns_display_event`**. Button events and resize events are strored in **`ns_display_base_event`**. In addition to creating these user interface widgets, an invisible base is created to provide functionality for the arrow keys to move the cursor. This is done by establishing keyboard shortcut resources stored in the **`.Xdefaults`** file.

As mentioned above, this routine also sets up the `base_uval`. This structure contains all the information the various subroutines would need to perform their functions. Since this program does not use common blocks, all global variables are stored here, associated with the display base id. Therefore, to access this information, a widget control call to get the user value must be made in each routine. It is suggested that this be done at the beginning of each routine. Furthermore, if a new routine is created, it is important to pass the display base id in as a parameter so that the new procedure has access to the uval. The use of the uval instead of common blocks is very useful in that each display can contain its own user values directly

related to its own image. In this way, for example, the spectrometer display won't access a variable set for the scam display.

Once the widgets and uval are created, the routine calls **ns_show_current** to load an image. If there is no current image loaded, it loads a blank screen. Then it registers the base id's of the displays it created with the info_uval by calling **ns_register_template**. Finally, it calls **ns_graphwindow** to create a widget box that will display the various plots the user may wish to view. It remains unmapped until the user calls upon it.

C. ns_frame_ready_loop

1. **Called by: ns_quicklook**
2. **Procedures called: (in ns_frame_ready_loop_event) ktlClose, show, ns_open_fits**
3. **Description / Notes**

If the KIDL keyword is set, QL starts a loop contained here in this procedure. Essentially, this loop is run in the ns_frame_ready_loop_event as a recurring base event that constantly asks the server for information about the status of the camera. If the camera is done taking an exposure, this procedure asks the server for the type of exposure (test or real), filename, guiding status, and other information about the exposure, and opens the data as a FITS file in the respective display.

D. ns_open_fits

1. **Called by: ns_frame_ready_loop_event, ns_menu_event, ns_arrow_key_event**
2. **Procedures called: nir_pickfile, ns_readcheck, ns_get_pathname, ns_get_extension, ns_readfits, sxpar, ns_show_current**
3. **Description / Notes**

Adapted from the **readfits** function, this routine loads a FITS file from disk to a two-dimensional array to be displayed an manipulated by QL. If the pick keyword is set (e.g. by the menu option, Open...), a dialog box is opened for the user to browse to the file to be opened.

E. ns_show_current

1. **Called by: ns_crude_zoom, ns_display, ns_default_math, ns_menu_event, ns_display_base_event, ns_draw_event, ns_open_fits, ns_recenter, ns_set_display_event.pro, ns_reset_zoom**
2. **Procedures called: ns_readcheck, ns_readfits**
3. **Description / Notes**

This procedure displays the current image in the display window. If the bias and flat are toggled, then they are subtracted and/or divided from the current image.

F. `ns_register_template`

1. **Called by: `ns_display`**
2. **Procedures called: none**
3. **Description / Notes**

This procedure is called by **`ns_display`** when it creates a display for the user to use. It receives the base id of the newly created display and stores it in the `info_uval`.

G. `ns_display_base_event`

1. **Called by: resizing display base or hitting a button**
2. **Procedures called: `ns_center_setup`, `ns_recenter`, `ns_crude_zoom`, `ns_reset_zoom`, `ns_show_current`**
3. **Description / Notes**

This procedure handles button and resize events. If a button is added to the display widget, its handler must be added here. Button values are passed in by name.

H. `ns_display_event`

1. **Called by: selecting a display menu item or moving cursor into draw window**
2. **Procedures called: `ns_menu_event`, `ns_fits_event`**
3. **Description / Notes**

This procedure determines which kind of event occurred between a menu event and a FITS window event, such as mouse movement over the draw window. Menu events are sent to **`ns_menu_event`**, and window events are handled by **`ns_fits_event`**.

I. `ns_fits_event`

1. **Called by: `ns_display_event`**
2. **Procedures called: `ns_draw_event`**
3. **Description / Notes**

This procedure first determines if the event is a tracking event, that is, mouse movement in or out of the window. If it is, it marks the case in the uval. If it isn't, that is a clicking event or a motion event, it is sent to **`ns_draw_event`**.

J. `ns_draw_event`

1. **Called by: `ns_fits_event`**
2. **Procedures called: `ns_profile_plot`, `ns_show_current`, `ns_statistics`, `ns_horizontal`, `ns_vertical`, `ns_box_contour`, `ns_box_surface`, `ns_photometry`, `ns_gaussian`**
3. **Description / Notes**

   This procedure handles most of the events that occur in the image display window. It covers actions such as motions in the window, clicking in the window, and drawing boxes. It use event tag names such as `event.press` and `event.release` to determine what action was taken, and the uval flags `drawing_box` and others to determine what action to take. To determine which process needs the information extracted in this event, it calls the mode flag from the uval. For example, if the user wants to draw a box for a horizontal profile, to start the box, `event.press` is set to one, the horizontal profile sets `drawing_box` to one, and `box_mode` to `'Box Horizontal'`. The routine is divided into sections for each type of action, and they are identified by the comments.

K. `ns_arrow_key_event`

1. **Called by: hitting an arrow key, hitting a hotkey**
2. **Procedures called: same as `ns_menu_event`**
3. **Description / Notes**

   This procedure defines what action to take when a key accelerator defined in **`.Xdefaults`** is pressed. Currently, the supported keys are the arrow keys, which move the cursor one pixel in the corresponding direction, control + the arrow keys, which move the cursor four pixels in the corresponding direction, and the hotkeys defined for the menu actions.

L. `ns_point_setup`

1. **Called by: `ns_arrow_key_event, ns_menu_event`**
2. **Procedures called: `ns_instruct_setup`, `ns_photometry`, `ns_gaussian`**
3. **Description / Notes**

   This procedure tells the program that a box is not being drawn and that a point is going to be picked. It then changes the cursor and begins the respective procedure that the picked point is for. Currently, this is only called for photometry and gaussian fitting. Profile at cursor plotting is started by **`ns_profile_setup`** which essentially does the same thing, but uses **`ns_draw_event`** to launch **`ns_rowcol_profile`**.

M. `ns_graphwindow`

1. **Called by: `ns_display`**
2. **Procedures called: `ns_box_contour`, `ns_box_surface`, `ns_rowcol_profile`**
3. **Description / Notes**

   This procedure creates the window in which the profile at point, contour and surface plots are plotted. When not in use, it is unmapped. If the user hits the 'Done' button, the base becomes unmapped. If the user kills the window, the window redraws itself unmapped. In this way, the base always exists.

N. `ns_menu_event`

1. **Called by: `ns_display_event`**
2. **Procedures called: `ns_show_current`, `ns_display_quit`, `ns_quit_all`, `ns_open_fits`, `ns_save_fits`, `ns_ps_current`, `ns_toggle_keep_zoom_center`, `ns_center_setup`, `ns_recenter`, `ns_change_color`, `ns_reset_display`, `ns_set_display`, `ns_set_autoscale`, `ns_point_setup`, `ns_default_math`, `ns_arithmatic`, `ns_sdiff`, `ns_statistics`, `ns_horizontal`, `ns_vertical`, `ns_profile_setup`, `ns_contour_toggle`, `ns_box_contour`, `ns_draw_box`, `ns_surface_toggle`, `ns_xsurface_toggle`, `ns_box_surface`, `ns_crude_zoom`, `ns_header`**
3. **Description / Notes**

   This procedure maps the user choice on the main menu bar to the corresponding procedure that performs the actions suggested by the name of the option. Since the menu is a `cw_pdmenu`, each option on the menu is treated as a button. The value of the menu is passed as the name of the option, so that if the order of `plot_menu_desc` is changed, this routine does not have to be changed. However, if a button title is changes, this routine must reflect that.

O. Brief Description of Remaining procedures (procedures called by)

1. **`ns_aper (ns_photometry):`** computes concentric aperture photometry
2. **`ns_arithmatic (ns_menu_event, ns_arrow_key_event):`** provides interface for performing mathematical operations on images
3. **`ns_box_contour (ns_draw_event, ns_menu_event, ns_graphwindow):`** displays contour plot of image area in the graph window
4. **`ns_box_surface (ns_draw_event, ns_menu_event, ns_graphwindow):`** displays surface plot of image area in the graph window
5. **`ns_calc_display (ns_show_current):`** determines scale and min and max value of pixels to find the best way to display the image

6. **ns_center_setup (ns_display_base_event, ns_menu_event, ns_arrow_key_event):** sets up new center picking
7. **ns_change_color (ns_menu_event):** allows user to change color table
8. **ns_contour_toggle (ns_menu_event):** sets filling of contour plots
9. **ns_crude_zoom (ns_menu_event, ns_arrow_key_event, ns_display_base_event) :** handles zooming in and out of the image
10. **ns_default_math (ns_menu_event, ns_arrow_key_event):** allows users to set BIAS/FLAT files and application.
11. **ns_gaussian (ns_point_setup, ns_draw_event):** displays information regarding the gaussian fitting of a point of the image
12. **ns_gaussian_fit (ns_gaussian):** performs actual gaussian fitting and sends results to **ns_gaussian**
13. **ns_get_extension (ns_open_fits, ns_arithmatic):** returns the extension of a filename
14. **ns_get_pathname (ns_open_fits, ns_arithmatic):** returns the pathname of a filename
15. **ns_horizontal (ns_draw_event, ns_menu_event, ns_arrow_key_event):** plots the average value of a horizontal cut of the image
16. **ns_instruct_setup (ns_center_setup, ns_draw_box, ns_point_setup, ns_profile_setup):** sets the value of the status bar on the display
17. **ns_moment (ns_calc_display, ns_statistics):** computes the mean and standard deviation.
18. **ns_movetel (ns_menu_event, ns_arrow_key_event):** send a command to the server to move the telescope an amount determined by user clicks.
19. **ns_header (ns_menu_event, ns_arrow_key_event):** displays fits header information about the current file.
20. **ns_pan (ns_arrow_key_event):** centers the image around a point where the user clicks.
21. **ns_photometry (ns_draw_event, ns_point_setup):** displays photometric information about a point where the user clicks
22. **ns_profile_plot (ns_draw_event):** sets up plotting of profile at point where user clicked
23. **ns_profile_setup (ns_menu_event):** sets up draw window so that point where user clicked is used to plot a profile at that point
24. **ns_ps_current (ns_menu_event, ns_arrow_key_event):** creates a postscript file of the current image.
25. **ns_print (ns_ps_current):** creates widget that prompts for printer name.
26. **ns_print_plot.pro (ns_horizontal, ns_vertical, ns_rowcol_profile, ns_box_surface, ns_box_contour):** creates widget that allows user to specify whether to print plot to a file or to a printer, and to specify a filename or printer name.

27. **ns_quit_all (ns_menu_event, ns_arrow_key_event):** calls **ns_quit_quicklook**
28. **ns_quit_quicklook (ns_quit_all):** exits program completely
29. **ns_readcheck (ns_arithmatic, ns_default_math, ns_open_fits, ns_sdiff, ns_show_current, ns_statistics):** checks to see if a file exists
30. **ns_readfits (ns_arithmatic, ns_open_fits, ns_sdiff, ns_show_current, ns_statistics):** reads the data and header from an opened FITS file and stores them in arrays
31. **ns_recenter (ns_menu_event, ns_arrow_key_event, ns_display_base_event):** recalculates the center of an image
32. **ns_reset_display (ns_menu_event):** calls **ns_show_current**
33. **ns_reset_zoom (ns_display_base_event, ns_arrow_key_event):** resets the zoom to 1:1
34. **ns_rowcol_profile (ns_profile_plot, ns_graphwindow):** creates a historgram in the graph window in the x and y directions from a point that the user clicks on
35. **ns_save_fits (ns_menu_event, ns_arrow_key_event, ns_default_math, ns_display_base_event):** saves image to a FITS file
36. **ns_sdiff (ns_menu_event, ns_arrow_key_event):** subtracts the previous image from the current image
37. **ns_set_autoscale (ns_menu_event):** set the autoscaling of the image
38. **ns_set_display (ns_menu_event):** allows user to manually set the minimum and maximum pixel value for display scaling
39. **ns_show_original (ns_arithmatic, ns_sdiff):** calls **ns_show_current**
40. **ns_statistics (ns_draw_event, ns_menu_event, ns_arrow_key_event):** calculates statistical information about a range of pixels within a user drawn box
41. **ns_string_or_num (ns_arithmatic):** determines if input is a string or a number
42. **ns_surface_toggle (ns_menu_event):** toggles whether the surface plot is shaded
43. **ns_toggle_keep_zoom_center (ns_menu_event):** toggles whether zoom and center get reset when a new image is opened.
44. **ns_vertical (ns_draw_event, ns_menu_event, ns_arrow_key_event):** plots the average value of a vertical cut of the image
45. **ns_writecheck (ns_ps_current, ns_save_fits):** checks to see if a file exists and can be written to
46. **ns_writefits (ns_save_fits):** writes image data to a FITS file
47. **ns_xsurface_toggle (ns_menu_event):** toggles whether to use `xsurface` widget for displaying a surface plot.

P. IDLLIB files

Some routines are stored and accessed from the **idllib** directory (**/kroot/kss/nirspec/ui/idllib**). This directory can hold commonly accessed routines such as cursors and routines that communicate with the server. The following is a brief description of the files called by QL.

1. **kidl.pro (ex. ktlClose, show):** routines that make connection with server.
2. **nir_pickfile.pro:** user dialog box for picking files to be opened or written to.
3. **nir_tvbox.pro:** box shaped cursor drawn on the image where a user clicks, used in gaussian fitting.
4. **nir_tvcircle.pro:** circle shaped cursor drawn on the image where a user clicks, used in photometry.
5. **ns_xdisplayfile:** modified version of **xdisplayfile** that also contains a button for opening a webpage in Netscape.

4. Adding routines

A. Tips and Comments

In this section, we will give tips on adding routines to the QL package, and important aspects of the program to take note of. One important aspect of QL, as mentioned above, is the accessing of global variables. If a new routine requires information that is accessed outside of the scope of the routine itself, it must take the display base id as a parameter so that it has access to the base uval. Also, if the information created for the new routine needs to be accessed elsewhere, the uval in **ns_display** must include these new variables. Furthermore, if your routine updates a variable in the uval, the routine must incorporate this change by using a `widget_control` call. Another aspect of concern is incorporation into the user interface so that the user has access to the new routine. Most new routines will want to be accessed by the main menu bar. Therefore, the title of the new routine will need to be added to `plot_menu_desc` in **ns_display**, and what to do when the option is chosen from the menu needs to be added to **ns_menu_event**. If you want to assign a hotkey to your routine, a few step must be taken: 1) create a button widget in the invisible base in `i_menu` of the invisible base in **ns_display**, giving it a resource name; 2) register the button with xmanager; 3) assign the hotkey in the **~/.Xdefaults** file; 4) add the handling routine in **ns_arrow_key_event**. If the new routine requires action for any cursor motion over the image or clicking on a pixel in the image, the event handler must be included in **ns_draw_event**. Also, if the user clicks on a point in the new routine, a setup file may be needed such as **ns_profile_setup**, or **ns_point_setup** can be modified. If the image needs to be redrawn, a call should be made to **ns_show_current** to do this. Any new windows drawn should be resizable, and remember to include `kill_notify` events where necessary. Also, any communications with the server can access the functions defined in **kidl.pro** in the **idllib** directory, such as `show`, which asks the server for the value stored in a keyword and copies it into a local variable. These procedures are touched upon in the following example.

B. Example

Suppose you want to add the module call **ns_my_math_routine**, which performs some math operation on a pixel the user clicks on. The header of the routine must take the display base id as a parameter, e.g.

```
pro ns_my_math_routine, display_base_id
```

Then, the first line should call to the global uval.

```
widget_control, display_base_id, get_uvalue=uval
```

Next, include your code and such, performing any operations desired. If any values in the uval are changed by your routine, remember to update this at the end:

```
widget_control, display_base_id, set_uvalue=uval
```

If you want to add a new variable to the uval, add it in both the `junk2` structure, which defines the uval structure, and the `base_uval` structure, which gives the variables their initial values, both in **ns_display**.

Now, you want to give the user access to your routine, so you want to put it in the `Math` heading of the menu bar. So add a line to `plot_menu_desc`, somewhere under `Math`, but before `Plot:`

```
{ CW_PDMENU_S, 0, 'My Math Routine' }, $
```

Then, you need to add a line to the menu event handler in **ns_menu_event** in the case loop:

```
'My Math Routine': ns_point_setup, Event.top, value
```

We call **ns_point_setup** since the user will click on a point. This will get QL ready to act on a click. Add the following lines:

```
if ( mode eq 'My Math Routine' ) then $
     ns_my_math_routine, ns_display_base_id
```

ns_point_setup redefines `uval.point_mode` as `'My Math Routine'`. Now we want your routine called whenever the user clicks on a pixel, so we add the following to the case loop in the last if block in **ns_draw_event**:

```
'My Math Routine': BEGIN
     ; your_code… (anything needed for your
          routine to work)
```

```
ns_my_math_routine, event.top
END
```

Now suppose you want to assign the key ‘Control+m’ to the routine. First, add a button to **ns_display** where the other keys are assigned:

```
mymathname = resource_root + ‘mymath’
i_mymath = widget_button(i_menu, value = ‘mymath’, $
     resource_name = mymathname
```

Then, register it with xmanager:

```
xmanager, ‘ns_arrow_key_event’, i_mymath, /just_reg, $
     /no_block
```

Then assign the hotkey in **~/.Xdefaults**:

```
Idl*scambase*scam_mymath*accelerator: Ctrl<Key>m
```

Note: scam is used regardless of display type. Then add the handler in **ns_arrow_key_event** just as done in **ns_menu_event**.

These actions are not comprehensive, but they outline the basic steps to be taken upon adding a module.