NIRSPEC

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

Jason Weiss **March 5, 1999**

NIRSPEC Software Programming Note 14.01
Echelle Format Simulator

1. Introduction

The Echelle Format Simulator (EFS) is a program that can be used to setup scripts to be run by the NIRSPEC server. Scripts can either be saved to disk in the form of a csh file, or can be executed by the EFS is already connected to the server. By modifying the filter setup, the slit, etc. the user can get a simulated echellogram to get a good idea of what the resulting image will look like.

2. Organization

The EFS is for the most part one self-contained program. The main program file is **`nirspec_efs.pro`** in the **`/kroot/kss/nirspec/ui/efsgui`** directory. It consists of the main program, plus the other modules and procedures called during the execution of the program. The few modules that are not within **`nirspec_efs.pro`** are:

- **`compare_list.pro`**
- **`ns_nod.pro`**
- **`plot_filter.pro`**
- **`read_lines.pro`**
- **`read_data.pro`**
- **`cw_toggle.pro`**

These modules perform specific operations called by modules from within the EFS.

Other important files are the data files used in various parts of the EFS, such as the bad pixel mask, the filter transmission curves, and the arclines lists.

Variables are passed among modules via common shared blocks. These blocks are:

- `shared_1`
- `shared_2`
- `shared_3`
- `shared_4`
- `shared_5`

- shared_6
- plot_shared

The first six blocks could be condensed into one, as they are all called by all procedures. The last block, plot_shared, is only called by main procedure, nirspec_efs, and plot_filter. It only contains the variables, plotted, a flag to let each other know if a window for the plotting of a filter transmission currently exists, efs_base, the widget id of the main EFS base, and efs_draw_index, the index of the main window of the EFS.

3. Important Modules

A. nirspec_efs

1. **Called by: none**
2. **Procedures Called: choose_standard_filter, setup_ech_filter, find_corners, draw_echellogram, draw_boxes, calc_pixel, fill_curbox_arrays**
3. **Description / Notes**

   This module is the core of the program. It does what most main procedures of any program do. It first gives the initial values to most of the variables. Then is creates the widgets and menu bar. Then, it assigns the widgets their initial status. Finally, it calls xmanager to manage the events.

B. formatmenu_event

1. **Called by: selecting a menu item**
2. **Procedures Called: open_config, save_button_event, format_quit, clear_overlay, user_lines_setup, oh_lines_setup, neon_lines_setup, argon_lines_setup, xenon_lines_setup, krypton_lines_setup, fill_curbox_arrays**
3. **Description / Notes**

   This procedure defines the action taken when a menu item is selected. Note for the lines procedures there are two cases each: one for when the option is on, and one for when the option is off.

C. draw_echellogram

1. **Called by: keyboard_event, format_base_event, filtermenu_event, full_event, slitmenu_event, specmode_event, new_box_event, del_box_event, next_box_event, pre_box_event, boxnum_event, open_config, clear_overlay, argon_lines_setup, krypton_lines_setup,**

xenon_lines_setup, neon_lines_setup, user_lines_setup, oh_lines_setup, nirspec_efs

2. **Procedures Called: draw_lines, fill_curbox_arrays**
3. **Description / Notes**

This routine draws the simulated echellogram in the draw window using such parameters the filter and slit used, as well as spectral imaging mode. It then calls draw lines to draw the appropriate arc lamp, user, or OH lines, if necessary. This routine is called every time a parameter is updated, or when the echellogram needs to be redrawn for any other reason.

D. draw_boxes

1. **Called by: keyboard_event, formatbase_event, filtermenu_event, full_event, slitmenu_event, specmode_event, new_box_event, del_box_event, next_box_event, pre_box_event, boxnum_event, open_config, clear_overlay, user_lines_setup, neon_lines_setup, argon_lines_setup, krypton_lines_setup, xenon_lines_setup, oh_lines_setup, nirspec_efs**
2. **Procedures Called: none**
3. **Description / Notes**

This routine is called whenever **draw_echellogram** is called. It redraws the box(es) for the current setup whenever the echellogram is redrawn.

E. draw_lines

1. **Called by: draw_echellogram**
2. **Procedures Called: read_lines**
3. **Description / Notes**

This routine draws the arc lamp, OH, and user lines on the echellogram. To know which lines to draw, a flag (e.g. neon_stat) is set in the setup procedure for each line (e.g. **neon_lines_setup**). This routine is called whenever the echellogram is redrawn by **draw_echellogram**.

F. draw_event

1. **Called by: moving or clicking in the draw window**
2. **Procedures Called: erase_curbox, find_corners, calc_pixel**
3. **Description / Notes**

This routines handles events performed in the main draw window. When the cursor moves across the echellogram, **calc_pixel** is called. Also tracking events occur so that these values are updated when an arrow key is hit. When the cursor is clicked

outside of a box, the current box is moved so that it is centered around the pixel clicked. Otherwise, a box can be dragged by grabbing it somewhere within the box and moving the mouse.

G. calc_pixel

1. **Called by: keyboard_event, filtermenu_event, full_event, specmode_event, new_box_event, del_box_event, pre_box_event, next_box_event, open_config, draw_event, nirspec_efs**
2. **Procedures Called: none**
3. **Description / Notes**

   This routine calculated the x and y pixel the cursor is on, and the echelle order and corresponding wavelength. If the cursor is not in the draw window, it either shows the values for the center of the box, or where the cursor left the draw window.

H. open_config

1. **Called by: keyboard_event, formatmenu_event**
2. **Procedures Called: choose_standard_filter, find_corners, update_buttons, setup_ech_filter, draw_echellogram, draw_boxes, fill_curbox_arrays, calc_pixel**
3. **Description / Notes**

   This routine reads a configuration from a file and sets up all of the boxes, buttons, menus, and labels to reflect the settings saved in the file. It also redraws the echellogram and the lines that correspond to the new settings.

I. save_script

1. **Called by: go_button_event, save_button_event**
2. **Procedures Called: none**
3. **Description / Notes**

   This procedure takes all the variables set by the EFS and prints them to a file in the form of a script that can be run by the nirspec server. The variables written to the script depends on which calibration mode is set.

J. user_lines

1. **Called by: user_lines_setup**
2. **Procedures Called: none**
3. **Description / Notes**

This procedure is an embedded main program for the user lines routines. It sets up a widget dialog that prompts for information about which lines to include, and which color to display them as on the echellogram.

K. plot_filter

1. **Called by: `plot_filter_button_event`**
2. **Procedures Called: `draw_profile`**
3. **Description / Notes**

   This routine handles the plotting of the transmission of the currently selected filter. It sets up the widgets to control the plotting, and then calls **`draw_profile`** to actually draw the plot.

L. keyboard_event

1. **Called by: Hitting a key on the keyboard**
2. **Procedures Called: `open_config`, `save_button_event`, `format_quit`, `clear_overlay`, `user_lines_setup`, `oh_lines_setup`, `neon_lines_setup`, `argon_lines_setup`, `krypton_lines_setup`, `xenon_lines_setup`, `erase_curbox`, `find_corners`, `erase_bad`, `find_bad`, `go_button_event`, `abort_button_event`, `new_box_event`, `pre_box_event`, `next_box_event`, `del_box_event`, `calc_pixel`**
3. **Description / Notes**

   This routine handles events that occur when the user hits a hotkey on the keyboard. Hotkeys are defined by giving the button (located in an unmapped base) a resource name. The resources are then defined in the **`.Xdefaults`** file in the home directory. Basically, this routine assigns a key to a menu event.

M. Other Modules (procedures called by)

1. **`go_button_event` (`keyboard_event`, clicking on the go button):** starts the current script
2. **`abort_button_event` (`keyboard_event`, clicking on the abort button):** stops the current script from running.
3. **`argon_lines_setup`, `krypton_lines_setup`, `neon_lines_setup`, `xenon_lines_setup`, `oh_lines_setup` (`keyboard_event`, `formatmenu_event`):** sets the flag so that lines are drawn on the echellogram.
4. **`user_lines_setup` (`keyboard_event`, `formatmenu_event`, `control_button_event`):** sets the flag so that the user line list widget is called.
5. **`clear_overlay` (`keyboard_event`, `formatmenu_event`):** removes all lines from echellogram by setting the flags to 0 and redrawing.
6. **`new_box_event`, `del_box_event`, `next_box_event`, `pre_box_event` (clicking on buttons):** creates, destroys, or switches between setup boxes.

7. **echbox_event, grabox_event, obj_name_event, star_name_event, itime_event, stime_event, coadds_event, scoadds_event, box_num_event (entering text into fields):** takes values user types in and assigns them to their corresponding variables.
8. **full_event, nods_event, reps_event, specmode_event, slitmenu_event, filtermenu_event (clicking on menu items):** takes new option clicked on in menu and assigns it to the corresponding variable
9. **thinmenu_event (clicking on button):** gets currently selected button of group and assigns it to `boxes(curbox).blocker`.
10. **mask_toggle_event (clicking the toggle switch):** turns on and off the showing of a bad pixel mask.
11. **formatbase_event (when user changes size of widget):** handles resize events.
12. **save_button_event (keyboard_event, formatmenu_event):** opens a file and starts `save_script`.
13. **format_quit (keyboard_event, formatmenu_event):** closes EFS
14. **plot_filter_button_event (clicking on the plot filter button):** gets current settings and calls the **plot_filter** routine.
15. **erase_curbox, find_corners (keyboard_event, formatbase_event, grabox_event, echbox_event, draw_event, filtermenu_event, slitmenu_event, specmode_event, new_box_event, del_box_event, next_box_event, pre_box_event, boxnum_event, open_config, nirspec_efs):** when a box is to be drawn, **find_corners** gives the location of the box. Using **erase_curbox** will draw a box, using it on top of a box that is already drawn will erase it.
16. **erase_bad, find_bad (keyboard_event, mask_toggle_event, draw_event, draw_boxes):** same as above, but draws the bad pixel map. Only drawn when `mask_flag` is set to 1.
17. **choose_standard_filter (filtermenu_event, specmode_event, del_box_event, next_box_event, pre_box_event, box_num_event, open_config, nirspec_efs) :** assigns filter information variables values based on the chosen filter.
18. **setup_ech_filter (filtermenu_event, slitmenu_event, specmode_event, del_box_event, next_box_event, pre_box_event, box_num_event, open_config, nirspec_efs):** calculates information about the echellogram and assigns that information to variables which are called by **draw_echellogram**.
19. **update_butttons (filtermenu_event, fullmenu_event, slitmenu_event, specmode_event, new_box_event, del_box_event, pre_box_event, next_box_event, open_config):** Sets all of the buttons, menu, boxes, and fields to the appropriate values.

20. **`fill_curbox_arrays` (`format_base_event`, `filtermenu_event`, `open_config`, `draw_echellogram`, `nirspec_efs`):** gives values to elements in arrays based on each setup box.
21. **`read_lines` (`draw_lines`, `open_list`):** reads lines from a file and fills an array with data.
22. **`input_proff`, `input_wavelength`, `input_wavelength_event`, `save_config` (none):** these routines are currently no longer in use.
23. **Additional User Lines Modules**
    - **A. `add_line_button_event` (user clicks on button):** adds line and comment to current list.
    - **B. `browse_button_event` (user clicks on button):** brings up dialog for selecting a file.
    - **C. `control_buttons_event` (user clicks on a button):** describes what is done when the user wants to close dialog.
    - **D. `ok_button_event` (user clicks on button):** closes color selection and applies change.
    - **E. `open_list` (`browse_button_event`):** opens a new list from a file.
    - **F. `remove_buttons_event` (user clicks on a button):** actions taken when user wants to remove a line from the list.
    - **G. `rgb_event` (user moves color sliders):** gets color and updates box with color.
    - **H. `save_as_button_event` (user clicks on button):** asks for filename and call **`save_list`**.
    - **I. `save_list` (`save_as_button_event`):** saves list to a file.
    - **J. `compare_list` (`browse_button_event`, `control_buttons_event`):** compares two lists to see if they are the same.
24. **Additional functions**
    - **A. `round2` (input: `num`):** returns `num` rounded to nearest 100^{th}.
    - **B. `roundphi` (input: `num`):** converts `num` to degrees, rounds to nearest 100^{th}, and the converts back to radians.
    - **C. `neflw` (input: `x`, `y`):** returns distortion width at point (`x`, `y`)
    - **D. `neflh` (input: `x`, `y`):** returns distortion height at point (`x`, `y`)
    - **E. `check_in_box` (input: `in_x`, `in_y`, `xs`, `ys`, `sptype`):** given a box with corners defined by `xs` and `ys`, checks to see if point `in_x` and `in_y` is with the box.

4. Modifying EFS -- Tips and Comments

Customizing EFS is not a very difficult thing to do. The important thing to do is to keep track of all of the variables. Before adding or changing a routine, it is worth the time to figure out what each variable represents. Since there is no difference in the six numbered common blocks, a new variable could be added to each one. Usually, variables are initialized in the main procedure, **`nirspec_efs`**. When adding a new routine, it doesn't hurt to include all six common blocks to the routine (as long as no variables in the new routine share the same names).

If a new parameter to the setup is added, make sure it is implemented in the opening of a configuration. When a saved configuration is opened, all of the settings need to be updated, including the new parameter. If the new parameter affects the way the echellogram is displayed, make sure to update the **setup_ech_filter** and **draw_echellogram** routines as necessary, and to call **draw_echellogram**, **draw_boxes**, and if necessary, **find_corners**, **erase_curbox**, **choose_standard_fitler**, and **update_buttons** to update the echellogram displayed. Note how the use of **erase_box** and **find_corners** (as well as **erase_bad** and **find_bad**, when necessary) to move a box. First, the current box is erased by calling **erase_box**. Then, **find_corners** is used to find location of new box. Then, **erase_box** is called again to redraw the box in the new location.

To add a hotkey, first add a button in the `i_menu` in the `i_base`. Give it a resource name and a value. In **keyboard_event**, add a case corresponding to the value giving the action to carry out when the button is pressed. In the **~/.Xdefaults** file, add a line giving the resource name with the appropriate lines to define the key. See existing lines for a model.