1987

PyGraph: a graphic front-end for the PAISLey executable specification language

Richard Willison

Rochester Institute of Technology
School of Computer Science and Technology

PyGraph: A Graphic Front-End for the PAISLey Executable Specification Language

by

Richard "Buzz" Willison

A thesis, submitted to
The Faculty of the School of Computer Science and Technology,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Approved by: Alan R. Kaminsky

Professor Alan Kaminsky, Chairman

Jeff A. Lasky

Professor Jeff Lasky

Peter G. Anderson

Professor Peter Anderson

October 9, 1987

Title of Thesis: PyGraph: A Graphic Front-End for the PAISLey Executable Specification Language

I, Richard Willison, prefer to be contacted each time a request for reproduction is made. I can be reached at the following address:

554 Beach Ave.
Rochester, N.Y.
14612

Date: November 3, 1987

1.2. Abstract.

This project is an application software system providing a graphical interface to PAISLey. PAISLey (Process Oriented. Applicative. Interpretable Specification Language) is an executable specification language developed at AT&T Bell Labs to model real-time systems. The interface provides the user with a tool to graphically depict a real-time system. The graphics package, called PyGraph, has been implemented on a Sun 3 in a multi-window environment. While developing a graphic, PyGraph generates a PAISLey specification template for the graphic image. The template is filled out by the user with assistance provided by PyGraph.

1.3. Computing Review Subject Codes.

The following are the classification codes for this paper as specified in the New (1982) Computing Reviews Classification System.

Primary: D.2.1. Specifications/Requirements.

Secondary: 1.3.2. Graphics Systems.

D.2.2. User Interfaces.

1.4. Table of Contents.

1. Preliminary Information

1.1 Title and Acceptance Page..... 1

1.2 Abstract..... 2

1.3 Computing Review Subject Codes..... 2

1.4 Table of Contents..... 3

2. Proposal (abbreviated due to original length)..... 5

3. Introduction..... 5

3.1. Executable Specification Languages..... 5

3.2. Graphic Interfaces..... 6

3.3. Previous Work in this Field..... 7

4. Background..... 9

5. Graphic Interface Definition - Functions Performed. 19

6. Architectural Design..... 23

6.1. Foundation - Object Oriented System..... 23

6.2. Interface Flow Diagram..... 24

6.3. Object Oriented Development..... 24

6.4. PyGraph File Structure..... 24

6.5. PyGraph Function Hierarchy..... 29

6.6. PyGraph Data Structures..... 44

7. Verification and Validation..... 45

7.1. Testing Process..... 45

7.2. Test Results..... 46

8. Conclusions..... 48

8.1. Problems Encountered and Solved..... 48

8.2. Limitations of the System..... 49

8.3. Lessons Learned..... 50

8.3.1. Alternative Approaches for Improved System..... 50
8.3.2. Future Extensions and Related Thesis Topics..... 51
9. Bibliography..... 53
10. Example PyGraph Output..... 55
11. PyGraph Program Listing..... 59
12. PyGraph Templates..... 184
13. PyGraph User's Guide..... 204

2. Proposal.

The PyGraph package includes very similar functionality and appearance to what was originally proposed. Minor changes were made in order to take advantage of hardware capabilities and simplify the user interface. Due to the length of the original proposal and the limited changes that were made, it is not included here as a separate package. However, some of the following sections are very similar to the proposal.

3. Introduction.

3.1. Executable Specification Languages.

An executable specification language (ESL) such as PAISLey enables a software developer or analyst to formally develop software requirements. PAISLey is an example of an ESL particularly well suited to real-time systems. Real-time systems are characterized by processing which must be accomplished within predefined time constraints to avoid failure of the system. Two significant software engineering benefits of using ESL's are verification of specification completeness and substantiation of specification accuracy. The process of developing the specification organizes the complexity inherent in real-time systems. The specification can then be executed using an interpreter to model the potential system. In some systems such as AdaGraph [1] and USE.IT [8], source code which implements the specification is automatically generated. Another major benefit of these languages is the ability to simulate the dynamics of the

potential system. Users can see the system operate in simulated time and decide if their functional and performance requirements are being satisfied. The formalism of PAISLey requires that many questions be answered early-on that often could be overlooked until later life-cycle phases. Elements that may have been left out or required change at significant added cost or schedule overrun are now part of the original specification. Thus, it provides a firm foundation for eventual development of the final application.

3.2. Graphic Interfaces.

It is well acclaimed that individuals have a preference for graphical representations of systems. Research supporting this positive outlook has just been developed in the last few years. In [6], a study was conducted that assessed programmer reaction to interactive, graphic based programming. It was found that a vast majority of the subjects preferred the graphic alternative over more traditional text based programming. One of the reasons cited was the presence of a "complete, static representation of their programs in graphical form". The educational world has been one of the key groups spearheading graphics research because of the multitude of advantages that exist. Research in [4] has shown that understanding the operation of algorithms and properties of programs can occur more efficiently and effectively when presented graphically (and animated with dynamic simulation).

Current advances in hardware technology have expanded the application of real time systems. In turn, the need to graphically depict them prior to implementation, particularly in light of their stringent performance requirements and layered complexity [16] is also amplified. In addition to the benefits mentioned, a graphic provides a communication document for user/analyst display and discussion of the system.

The physical process of writing PAISLey specifications can be greatly simplified and streamlined. This is primarily because of the syntactical formality and high degree of completeness necessary to get a specification to execute. A graphical interface tool can be of significant assistance to a user. This is accomplished by enabling semi-automatic development of the PAISLey specification directly from the graphic as developed on the terminal.

3.3. Previous Work in this Field.

There has been considerable interest in graphics-based computer applications. Several of these have been developed for simulation languages and a few have been developed for ESL's. AdaGraph [1] is one such graphical system developed for the PAMELA specification language. Together, ADAGraph and PAMELA enable the user to specify a real-time system as a set of connected icons. Icons are used to represent processes, dataflows and other entities. The internal representation of the icons can be interpreted and ADA code

automatically generated. Another example is STATEMATE1, a methodology and computerized working environment for specifying and designing real-time and interactive systems. The methodology utilizes three views of the system being developed, each with a supporting chart technique. The three views include the behavioral view using a state chart, the functional view using an activity chart and the structural view using a module chart. A software package is then available for assistance in developing these charts and simulating the operation of the system. Another masters thesis here at RIT is being developed that will provide a graphical front-end to GPSS [10]. One last example is USE.IT, a family of packages designed to automate software development. A graphic can be developed using a graphical language, then corresponding code can be created automatically from one of several available code generators (FORTRAN, PASCAL, C, or COBOL).

The process just traced is a top-down approach, often starting at the the specification, eventually resulting in a final hard-coded program. Another graphics-based application that has generated significant attention is in the area of graphics-based programming systems. Here the graphics capability is used to simplify the bottom-up development of software and its associated algorithms. "Programming in Pictures" [12] is one such programming system. The tool enables the user to draw pictures of data and algorithms along with participative execution of the program. "Pecan"

[12] is another such system that allows several simultaneous views of the program on the terminal. Among those included are the program listing, symbol table, flow graph, execution stack and input/output displays.

The focus of some systems is on the graphic depiction of the program in execution. "Program Visualization" [12] is a package that allows the user to monitor control sequences and changes in data. An animation kit has been developed for "Smalltalk" [11]. It enables a user to view graphic representations of "the essentials of the algorithm". That is, pictures of the central data structure are displayed as execution occurs.

4. Background.

PAISLey is a language used in the Operational Approach to software development [13]. There are four basic steps to this approach. First, the requirement is identified and the behavior of a system that models the requirement is specified. Next, this behavior is documented in a formal operational specification using PAISLey. The specification becomes a record of the design decisions that are made and can be interpreted to model the behavior of the system. In the third step the problem oriented operational specification is transformed into a solution oriented specification. The key objective in this step is to transform the specifications structure while retaining its desired behavior. Finally, in the last step, the comprehensive PAISLey specification can be

implemented in a programming language or hardware and run in real time.

PAISLey - Key Words, Phrases and Concepts

PAISLey is one of several systems providing the formalization necessary to apply the operational approach. The conceptual basis for PAISLey is in communicating sequential process theory. PAISLey advocates a description of the system as a group of processes interacting over channels. The processes are either in the environment (such as a sensor providing a measurement) or in the system itself (such as an entity accepting the sensor measurement). A channel is a directed vector identifying a flow of some form of data from one process to another. Processes consist of a state and a successor function. Thus, each process is essentially a finite state machine and PAISLey cycles them as follows: 1) initial state -> 2) successor function -> 3) new state -> 4) repeat steps two and three.

A PAISLey specification is constructed with four types of statements. These are 1) System Declaration, 2) Set Definition 3) Function Declaration and 4) Function Definition. Each statement type has its own form and syntax which will be briefly reviewed later in an example.

The system declaration statement is basically a tuple including each process in its initial state. The set definitions define names of sets and allowable members for the set name. Function declarations indicate the cyclic allowable states of the process in the form of a domain and a

range. The domain is the set of allowable inputs to a process and the range is the set of allowable outputs. They are always in terms of the declared sets. **Function definition** statements define unique function names in terms of function expressions. The expressions are ultimately evaluated by the PAISLey interpreter to iteratively move the processes from one state to another. Function expressions can be quite complicated with the many elements that can be involved. The expression can include function names, formal parameters, specific function evaluations, tuple constructions and conditional selections.

The key elements that enable processes to interact are exchange functions. These intrinsic functions enable processes to carry out "two-way point-to-point mutually synchronized communication" [16]. An exchange function consists of an exchange type descriptor (x, xr, or xm) followed by a channel name over which the exchange occurs and an argument that will be exchanged. Each exchange type descriptor provides a different kind of synchronization. The x exchanges will match and swap arguments with any other incidence of an exchange on the identified channel. The xm exchanges also wait, but will not exchange with any other xm on the same channel. The 'm' stands for many-to-one because any other xr or x on the channel would cause an exchange. The xr exchange only occurs with x or xm. However, xr's do not wait for another exchange incidence on the channel. If no exchange is waiting when xr is evaluated, it does not wait

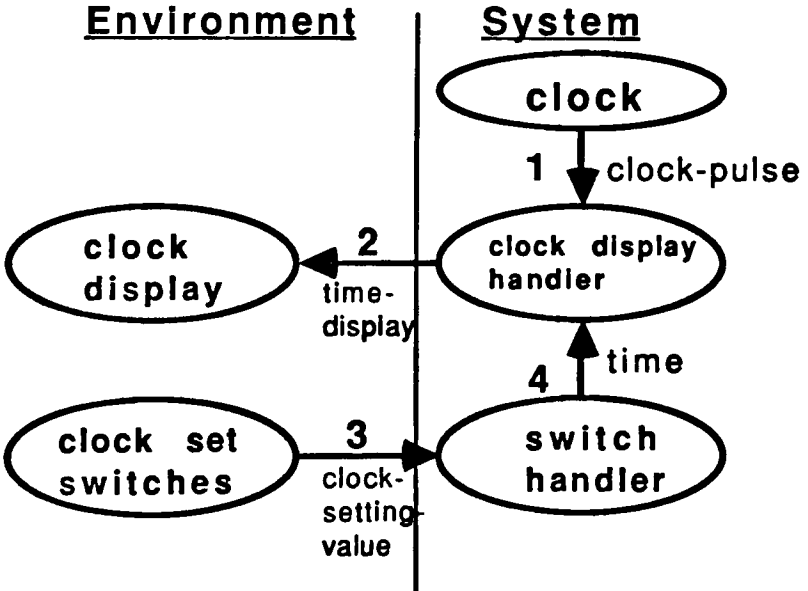
for a match and only returns its own argument to itself. The exchange functions are placed on the specification at the locations that synchronization is required. These locations are called interaction sites.

PAISLey Digital Clock Example

As an example of PAISLey specification development, a segment of the process for a digital clock is documented below.

I. Process Identification

The first step in developing a PAISLey specification is the identification of processes that are interacting in the system that it is desired to model. In this example there are five, two in the environment (the display and the clock setting switches) and three in the system (the clock, display handler and switch handler). A graphic illustration of the processes and channels is shown below.



DIGITAL CLOCK GRAPHIC

The arrows between the processes signify interactions between them. The interaction occurs over a channel, graphically represented with an arrow. It is here that the actual communication between the processes occurs. The clock affects the state of the display handler (via channel 1) which affects the state of the display (via channel 2). Additionally, the clock setting switches affect the state of the switch handler (via channel 3) which again affects the display (via channels 4 and 2).

All of the processes illustrated have a standard PAISLey template [14]. The template provides the four types of PAISLey statements previously mentioned: 1) System Declaration, 2) Set Definition, 3) Function Declaration and 4) Function Definition. Set Definition must be provided by the user because it is often too specific to an application to fully standardize in a template. Additionally, template modifications need to be made. These include the assignment of a unique name to the functions, identification of interaction sites and labeling the appropriate sites with exchange functions. Finally, initialization criteria must be placed in the specification.

A. Processes in the Environment:

The first process in this system is the clock display. This corresponds to a physical digital readout displaying the time of day. In order to describe the specification of a display device in PAISLey, one must realize that the display device is an example of a more general category called an

output device. The PAISLey template for this is shown below.

The Output Device Template:

```
1. ( . . . output-device-cycle[Null],. . . );
2.
3.
4. output-device-cycle:  FILLER --> FILLER;
5. output-device-cycle[Null] =
6.    output[get-output];

7.    get-output: --> ITEM;
7a.
8.    "This is intended as an interaction site, to be
      connected to the device handler."

9.    output:  ITEM --> FILLER;
      "This function represents output on the device."
```

The template above is modified (via a mapping process) in several ways to produce the following specification. The mapping is required because the template is not complete. There are blanks that must be filled in before running the specification on the PAISLey interpreter. It could be run through the interpreter in the processed configuration shown below, but the user may wish to further fill out the modified template to better match the system being modeled.

```
1. (. . . clock-display-device-cycle[Null], . . .);
2. TIME-DISPLAY-VALUE = STRING;
3. DISPLAY = TIME-DISPLAY-VALUE;

4. clock-display-device-cycle:  DISPLAY --> DISPLAY;
5. clock-display-device-cycle[d] =
6.         clock-display-output[get-clock-display];

7.         get-clock-display:  --> DISPLAY;
7a.        get-clock-display =
8.              x-time-display-chan[Null];

9.         clock-display-output:    DISPLAY --> DISPLAY;
```

The mapping that has taken place is from a generic

output device template to the customized clock display. Automatable parts of the mapping are of primary interest to this research because they provide the core functionality of PyGraph. The in-depth theory behind the template is considered to be definition that is beyond the scope of this research and a cursory definition will suffice.

Following is a description of the specification architecture. The line numbers in the example have been added for reference purposes.

System Declaration:

Line 1 would typically occur at the beginning of the specification with all of the other process initializations. It is grouped here with the rest of the statements from the template for completeness purposes. It consists of the successor function name with the initial state of the process as its argument.

Set Definition:

Lines 2 and 3 are set definitions. They describe all allowed values and members of sets that are referenced in the process. The statements are of the form SET NAME = SET EXPRESSION. Following PAISLey syntax, the first syllable of the set name must be all letters and they all must be capitals.

Function Declaration:

Every function in PAISLey must have a range and possibly a domain. The function declaration specifies them in the form successor-function-name: DOMAIN --> RANGE. Line 4 is an

example where the successor function clock-display-device-cycle has the display as input and output. Note that each syllable of the function name is either all lowercase or all digits.

Function Definition:

PAISLey functions have the form function-name[argument] = expression where the expression can be a constant or another function. If the expression is a function, then the direction of evaluation of the function is inner level to outer level. Lines 5-9 describe a multilayer function definition with lines 7 and 9 being further function declarations. Briefly, the clock-display-device-cycle consists of outputting the get-clock-display which is obtained from the time-display-chan channel. The time comes from the display handler and will not be obtained until the time is put on the channel by the handler (by definition of an x exchange). Once the get-clock-display is obtained, it is output according to the clock-display-output function. Finally, the evaluation of the function clock-display-device-cycle is complete.

The other process in the environment is the clock setting switches. They are necessary for the user of the clock to be able to set the time on the digital clock. This is an input device for which there is another PAISLey template similar to that for an output device.

The other standard processes in the environment are identified and briefly described below. All of the

corresponding PAISLey based PyGraph specification templates are presented in Section 12.

Controlled Object:

The controlled object models a unit in the environment that interacts with the system. It provides sensory information to the system or receives control output from the system (example: temperature sensor).

Input Device:

The input device models an object in the environment which accepts user input. An example is the clock setting switches in the digital clock.

B. Processes in the System:

Reader:

A unique reader process in the system must exist for each sensor out in the environment. It determines the frequency of read from the sensor and optionally, checks validity of sensor values, and converts the sensor reading to values that are internally useful to the system.

Monitor:

The monitor responds to some specific condition or combination of conditions in the environment. It may monitor several conditions via readers and then broadcast to selected controlled objects or output devices when required.

Informer:

The informer provides information to users or other

systems through output devices without regard to a condition. It provides data of interest that users have said they want to know about.

Modifier:

The modifier reconfigures the system by modifying process functions or sets of functions. It turns functions on and off and changes parameters in other processes.

I/O Handler:

There is one handler for each input or output device to provide conversion of values or buffering, as required. Two handlers exist in the digital clock example, one each for the input and output devices.

II. Process Connection

The second step in PAISLey specification development is the connection of processes that must interact with each other. This is accomplished by posting appropriate exchange functions at the desired location in the PAISLey specification. As previously mentioned, an exchange function consists of the type of exchange, a channel reference on which the exchange will occur, and an argument that will be exchanged. Thus, an example of a valid exchange from the previous example is; x-time-display-chan[Null]. This is an x exchange of the argument 'Null' on the channel time-display-chan.

5. Graphic Interface Definition - Functions Performed.

PyGraph Mappings

Mappings from a standard PAISLey template to an application dependent specification provide the core functionality of PyGraph. Each type of symbol that is placed onto the graphic has a corresponding template. When the object is drawn and named, a copy of the template is made and a customized specification generated for the symbol. All of the templates are presented in Section 12 by symbol type. When a template is processed, it is customized for the application. Where a ~1 appears in the template, the symbol name is placed in lowercase. Similarly, where a ~6 appears, the symbol name is placed in uppercase. A ~2 initiates the placement of a note that it is a default value. A ~3 places a default value of REAL.

The following is a summary of PyGraph functionality that is fully explained along with graphic displays in the User's Guide of Section 13.

Access PyGraph

The user is equipped with the ability to enter PyGraph from the Suntools graphics environment. Upon typing a specific command or making a menu selection, an initial screen is displayed containing buttons for all of the PyGraph graphics in the user's directory. Each button has a title including a button for creating a new graphic.

Create a Graphic of a Real-Time System

A graphic is comprised of one or more processes chosen from nine standard process symbols. All of the standard process symbols have a corresponding PAISLey template. When a symbol is placed on the drawing surface, the template is first customized by PyGraph. Next, the textual PAISLey specification will usually need to be filled out by the user for the specific system to be modeled. The ninth process symbol for a blank template enables the user to create a custom process. PyGraph enables the user to choose from symbols on the screen in constructing a graphic of the system. The nine standard symbols available in PyGraph for constructing a drawing are; 1) Controlled Object, 2) Input Device, 3) Output Device, 4) Reader, 5) Monitor, 6) Informer, 7) Modifier, 8) I/O Handler and 9) Blank. Functionality and use for each of these is described above in the background of section four. One or more of each of these symbols can be pulled from a palette and placed onto the drawing window. The capability also exists to delete a symbol from the graphic once it is selected and no channels are connected to it.

The next step is to identify interaction sites between the processes on the drawing. This is accomplished by placing a channel between the interacting processes. The channel is drawn by PyGraph when the user makes appropriate entries. The channel can also be deleted by removing it from the specification and taking appropriate steps.

Save a PyGraph Graphic

Upon completion of the graphic creation process, the user is asked if it's desired to save the changes made since the last save.

Delete a PyGraph Graphic

At any time in the graphic creation process, the user can request PyGraph to delete the entire graphic. A popup comfirmer verifies that the user really wants to delete the graphic, then returns the user to the directory.

Access an Existing Template

It was stated earlier that upon entering PyGraph, titled buttons display corresponding to the existing graphics that are in the user's directory. Selecting one of these will bring an existing graphic into the Graphic Development Environment.

Modify PAISLey Template

The Graphic Development Environment is the focal point for PyGraph. Both creating a new graphic and modifying an existing graphic are accomplished in this multi-window environment.

Process Symbols:

Each symbol on the graphic must have a unique name. The user is prompted by a popup panel to input a name. PyGraph

takes the name, checks it for uniqueness, and uses it to provide unique names to the structures in the textual PAISLey specification. By selecting a specific process symbol and activating it, another window appears with the PAISLey specification. The user has complete word processor capabilities to add and delete characters and lines from the specification as desired. If it is desired to move directly to a function in the template there is a quick find capability utilizing simple mouse operations. Possible interaction sites are identified for the user on the specification. At each site as desired, the user can post channel interactions with or without PyGraph assistance. PyGraph allows the user to choose the desired type of interaction to be posted for each interaction, with the click of a mouse button.

Channel Symbol:

Channels (straight lines) are automatically drawn between symbols on the graphic when requested by the user.

Create a File Containing a PAISLey Specification

When finished with graphic development, a file can be created containing the PAISLey specification that was developed in PyGraph. A popup panel is used for input of a file name. PyGraph creates the file and posts the specification into the file.

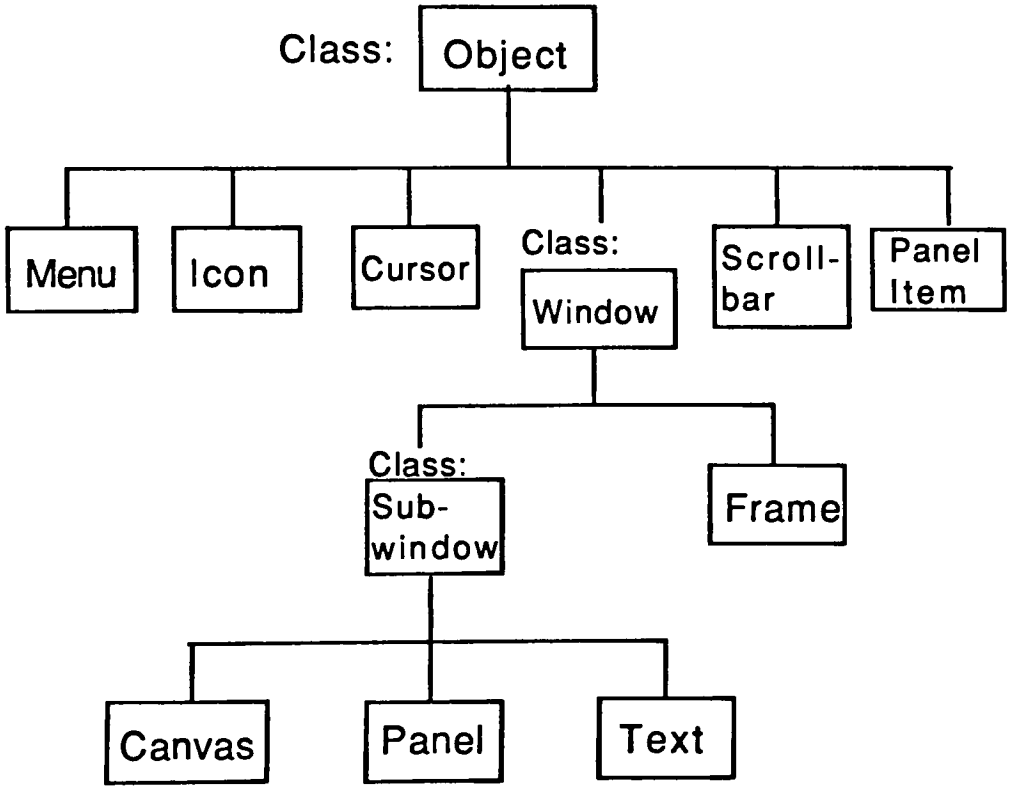
Screen Print

A capability exists for the user to request PyGraph to issue a screendump of the contents of the current terminal screen to a file for future printing.

6. Architectural Design.

6.1. Foundation - Object Oriented Structure.

The PyGraph program specification and development uses an object oriented approach. The reason for this is that Sunview, the graphic development environment for the Sun Workstation, is an object oriented system. Sunview has an array of objects that are visual building blocks for a graphic applications. PyGraph is a good match for this platform by taking advantage of all the objects. Figure 1 is a graphic depiction of the object classes and objects that Sunview offers and provide a basis for PyGraph.



6.2. Interface Flow Diagram.

Object oriented development advocates suggest the use of traditional requirements and analysis methods to provide a background of real world knowledge and intuitive understanding of the problem [3]. Figure 2 is an interface flow diagram to provide a depiction of reality. This exercise was helpful in providing understanding of the flow of user interaction and window management requirements.

6.3. Object Oriented Development

The first step in an object oriented development process [3] is the identification of objects. These are the major actors in the problem space. Given the interface flow diagram, it was a relatively straightforward process to pull out all of the objects. They are the nouns on the diagram and often are tangible, visible elements. Figure 3 is an object oriented decomposition of the problem space. It is clear from this drawing that Sunview is at the heart of the environment.

6.4. File Structure.

With the object oriented decomposition in place, an intermediate step was taken to categorize the objects into program source files. In this manner, each file could maintain a major segment of functionality. A diagram depicting this process is shown in Figure 4. The final file structure including the supporting elements is shown in Figure 5. The symbols are one of the supporting elements. They have been created as icons and are stored in their own

directory. Another element, the specification templates reside in the user directory, all having a standard file name format.

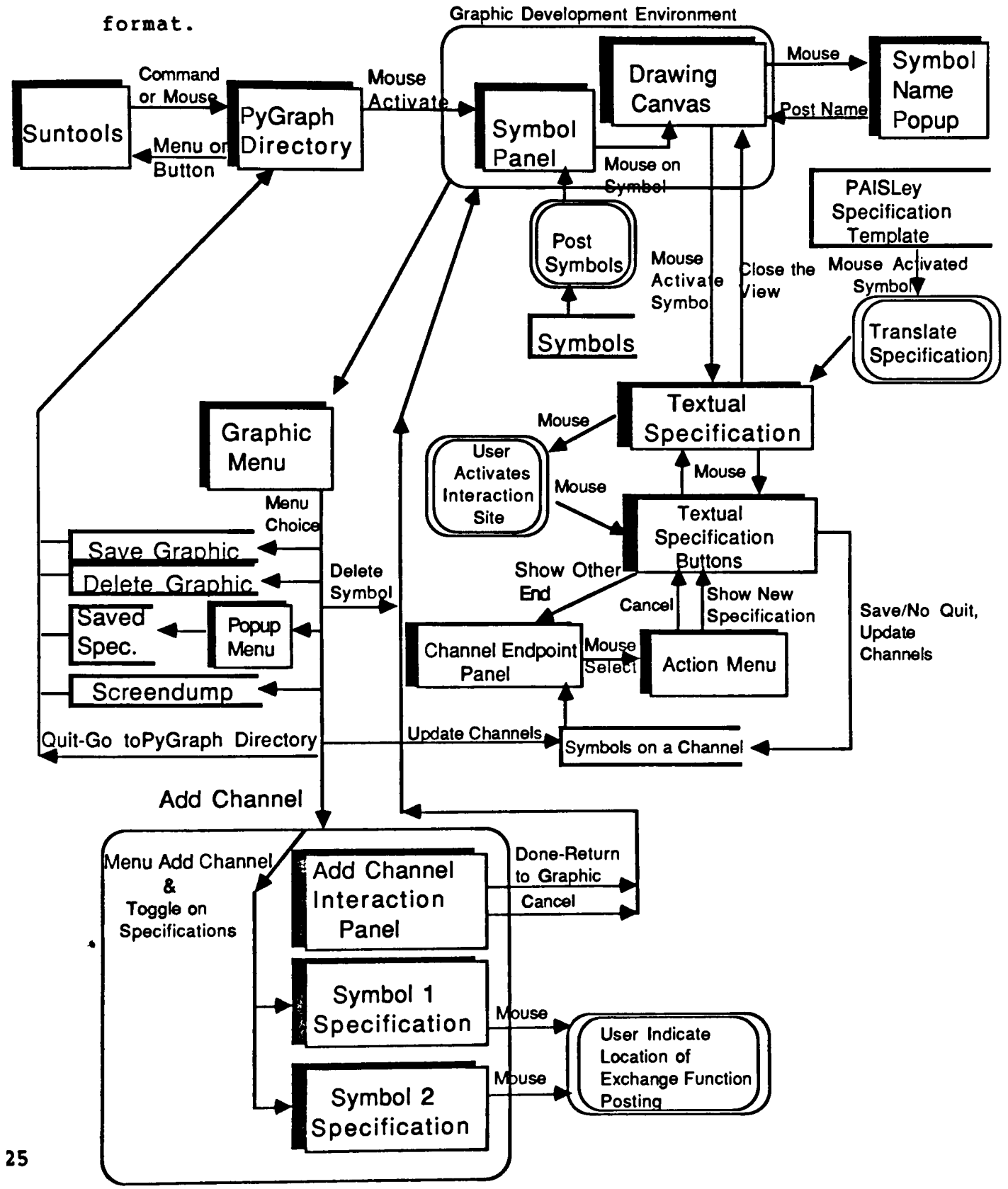


Fig. 2. Interface Flow Diagram

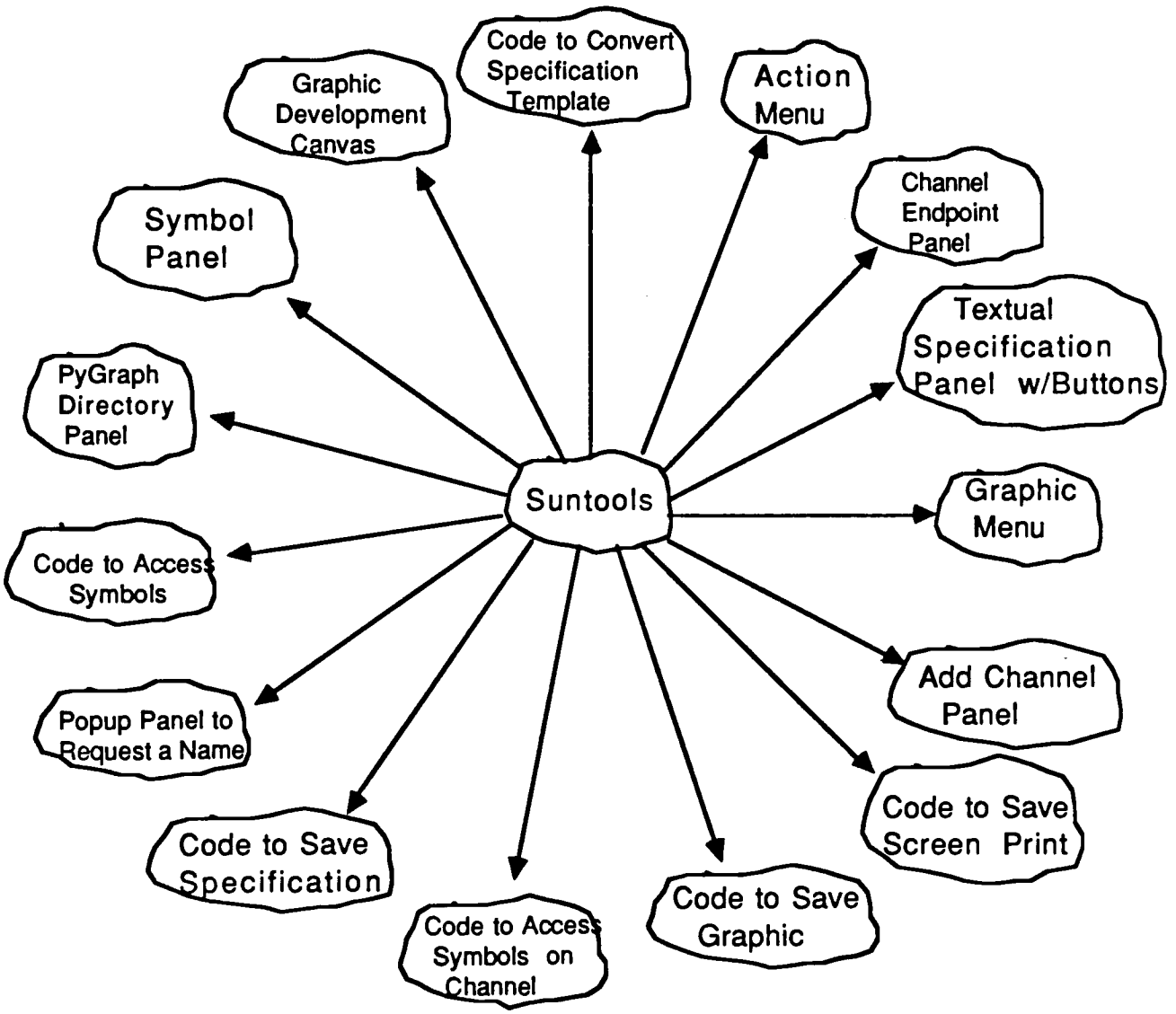


Fig. 3. Object Oriented Decomposition

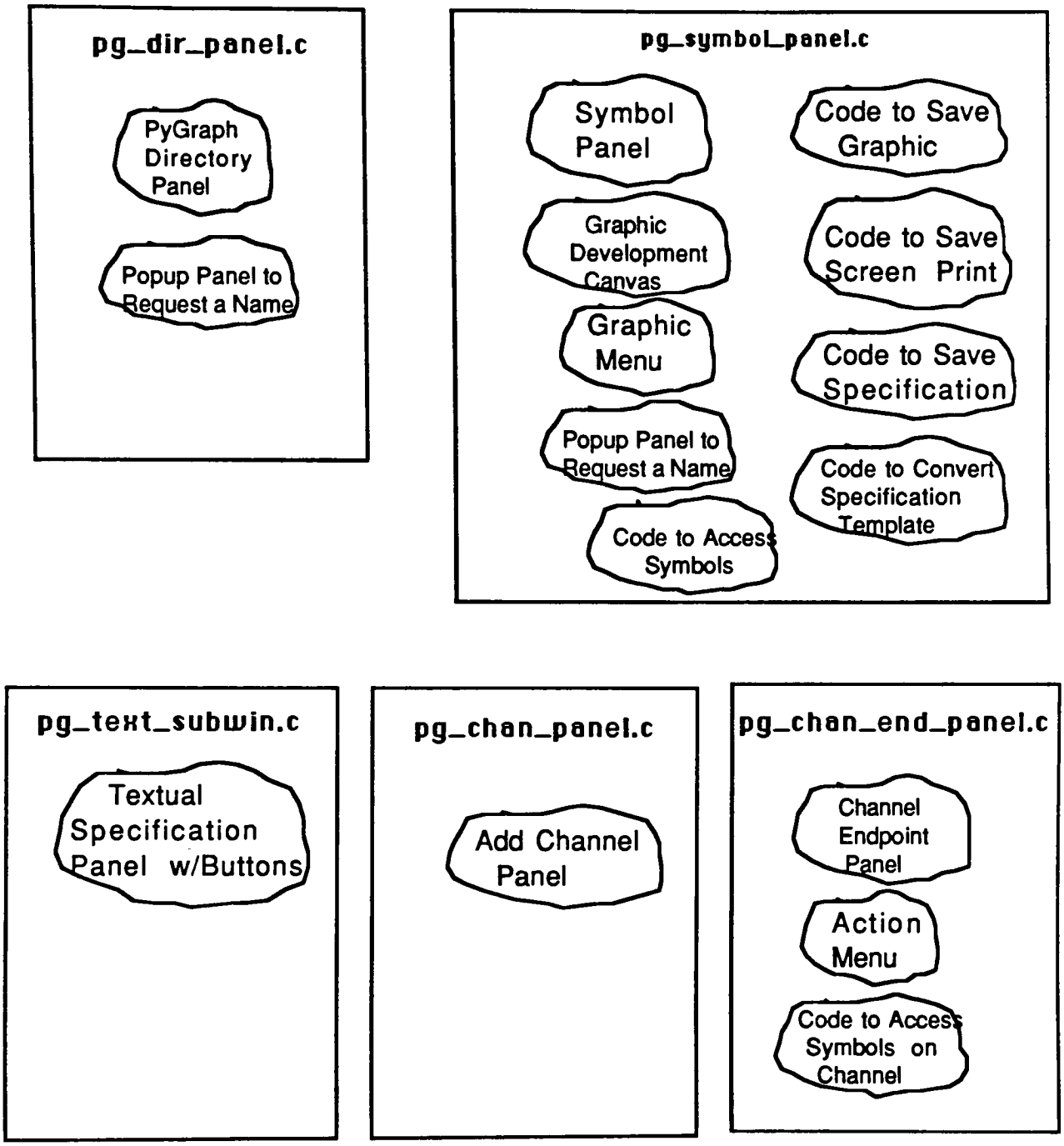


Fig. 4 - PyGraph Object File Assignment

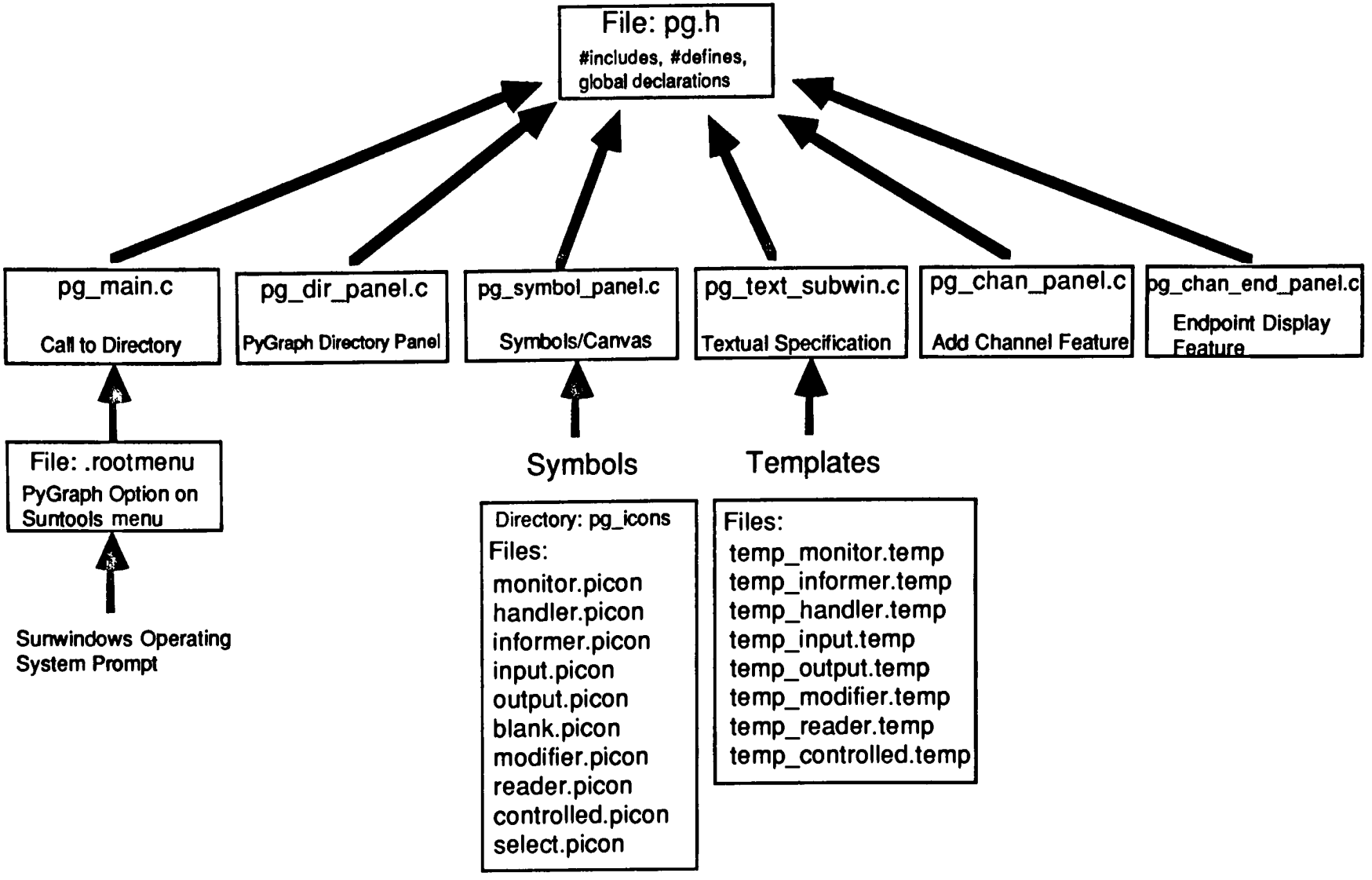


Fig. 5 - PyGraph Program/File Structure

6.5. PyGraph Function Hierarchy.

With the interface flow diagram, specification of functionality and object file structure complete, the next step involved a great deal of matching. The capability of the hardware and Sunview development software had to be matched with the desired capability, ultimately filling in the gaps with C language software. Each file contains a number of functions to create the desired appearance and capability. Figure 6 is a numbered graphic depiction of the PyGraph functional hierarchy followed by a key to the functions. Each function name has a corresponding one or two line description.

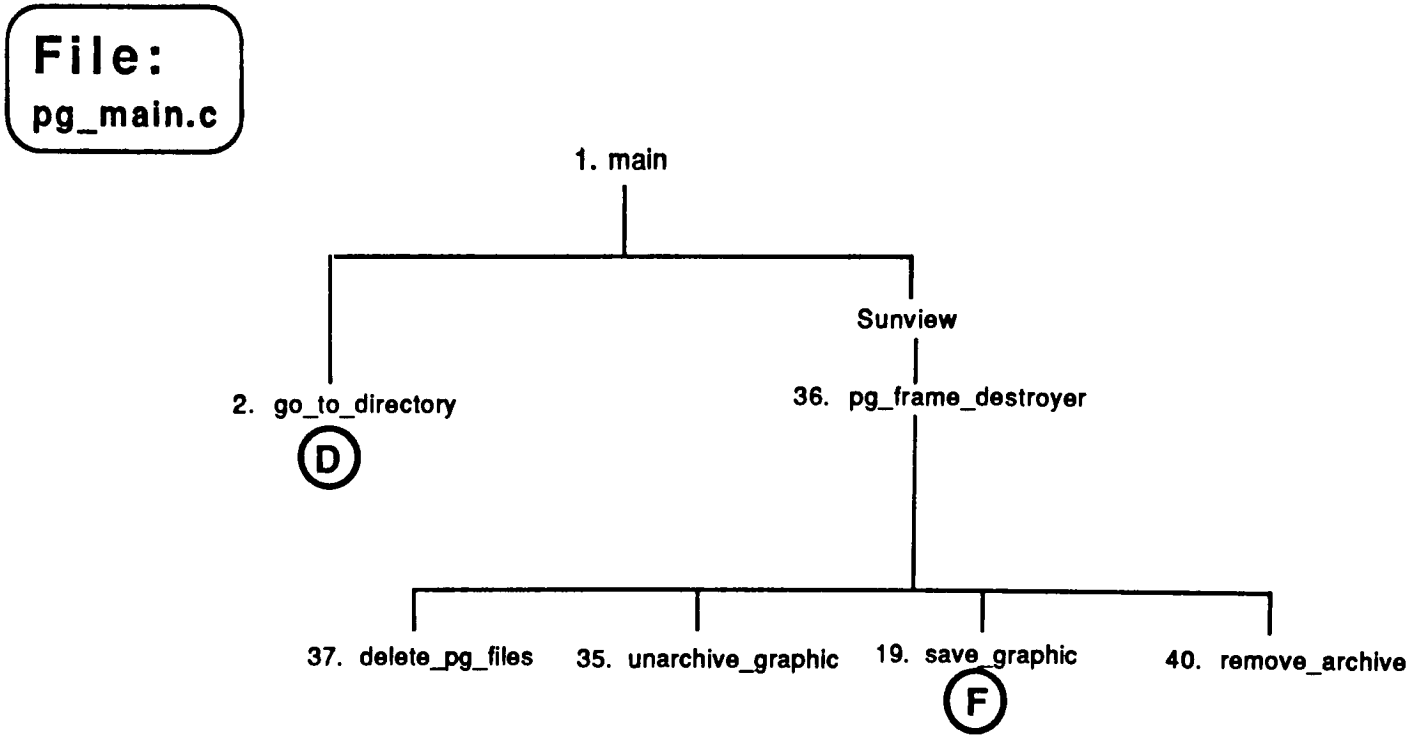


Fig. 6 - PyGraph Function Call Hierarchy

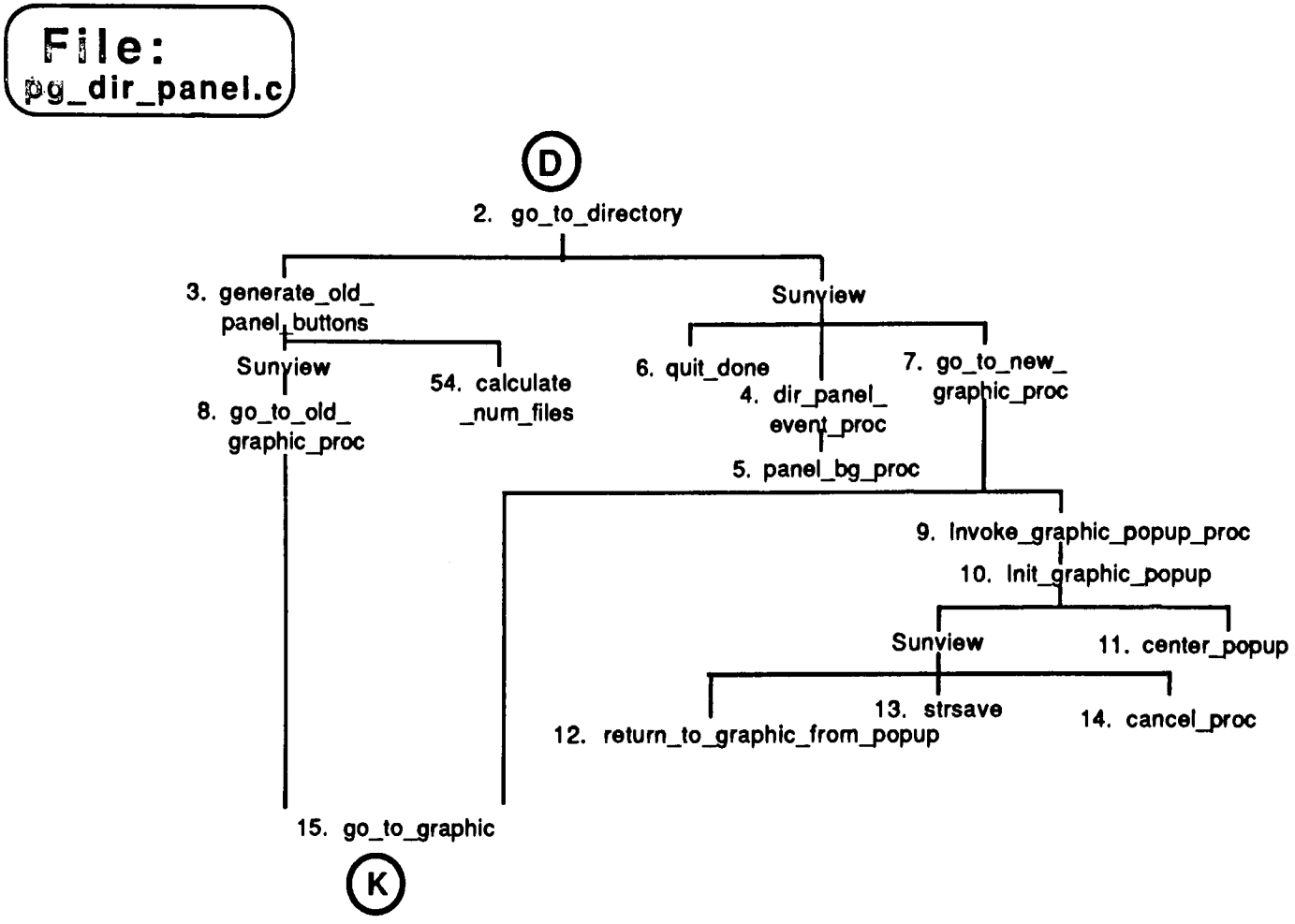


Fig. 6 (cont.) - PyGraph Function Call Hierarchy

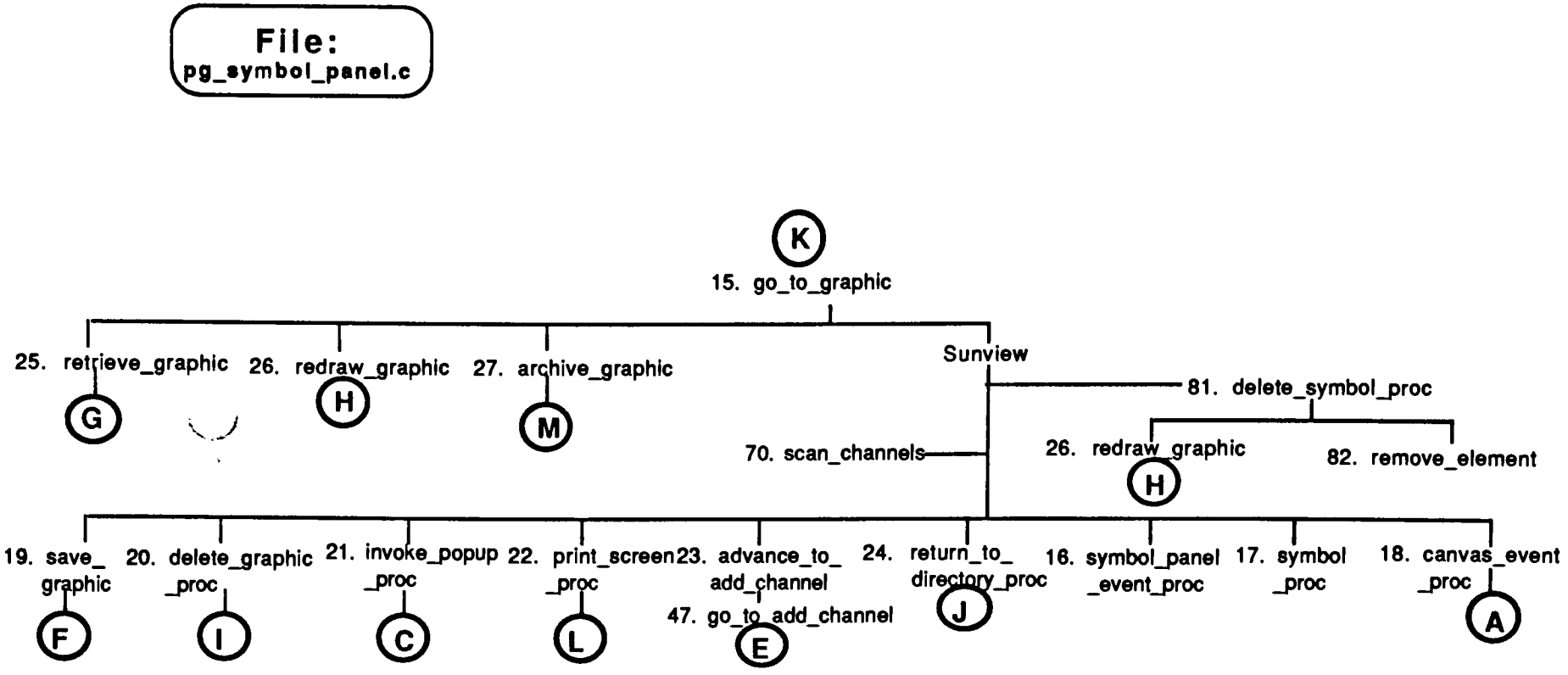


Fig. 6 (cont.) - PyGraph Function Call Hierarchy

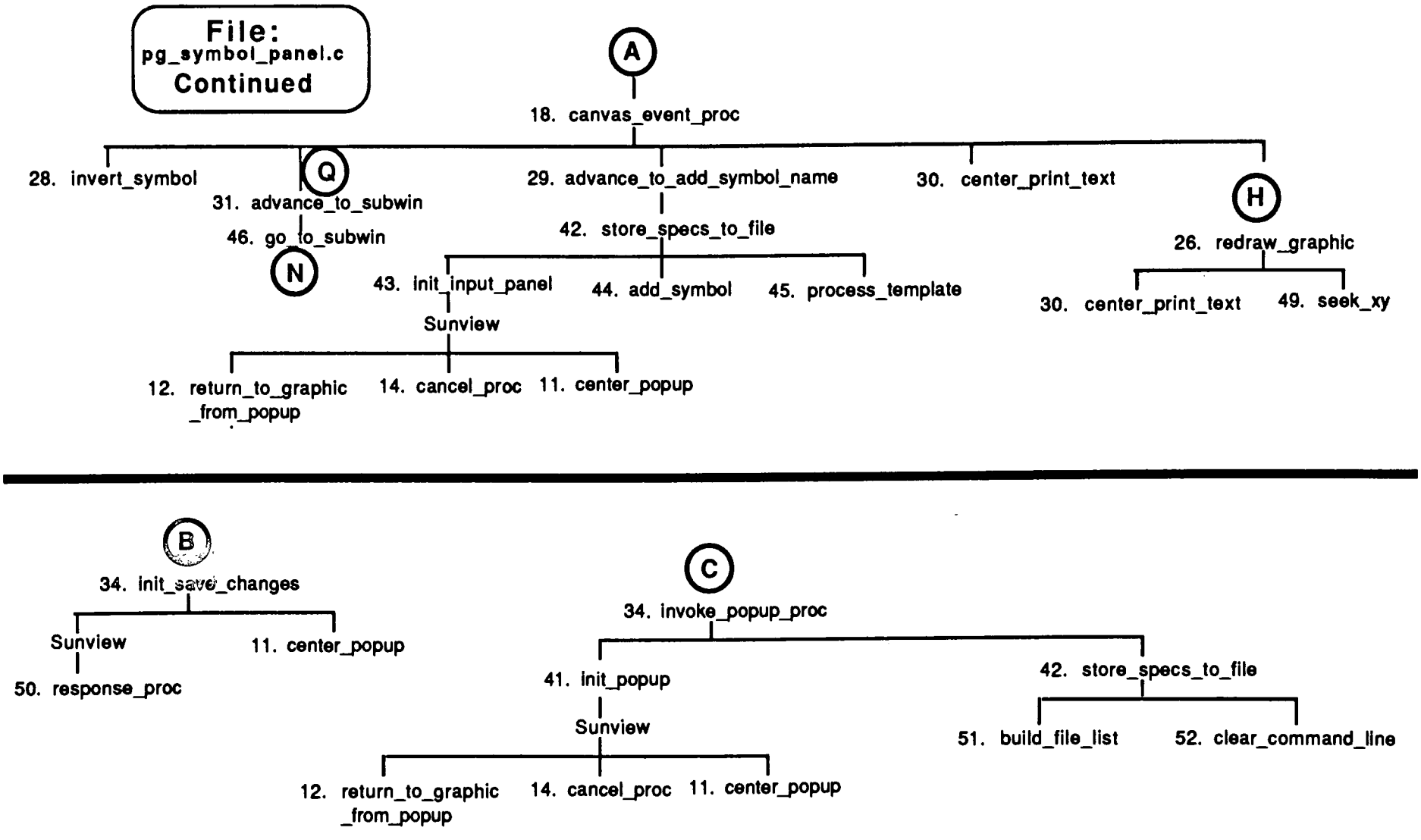


Fig. 6 (cont.) - PyGraph Function Call Hierarchy

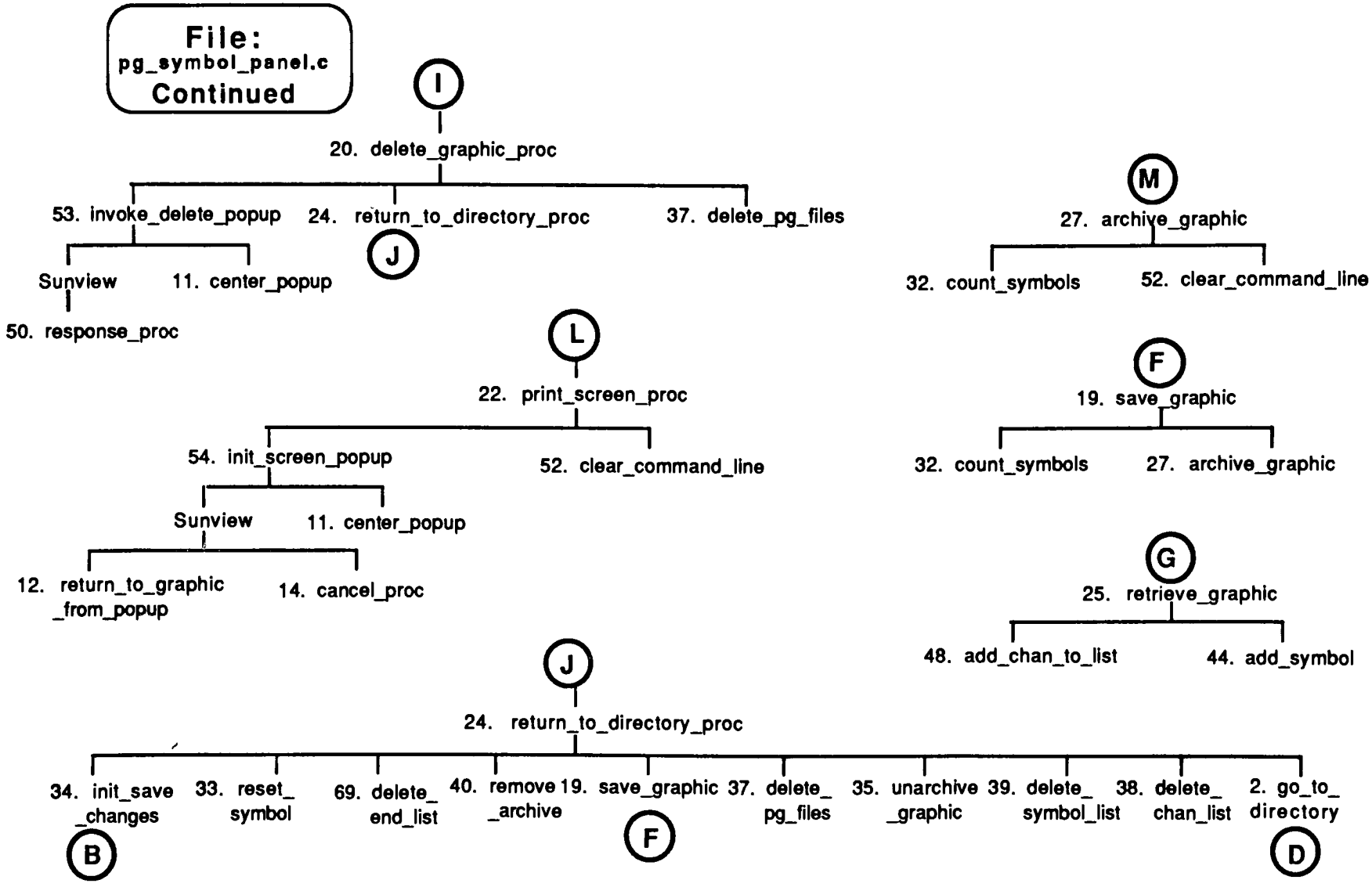


Fig. 6 (cont.) - PyGraph Function Call Hierarchy

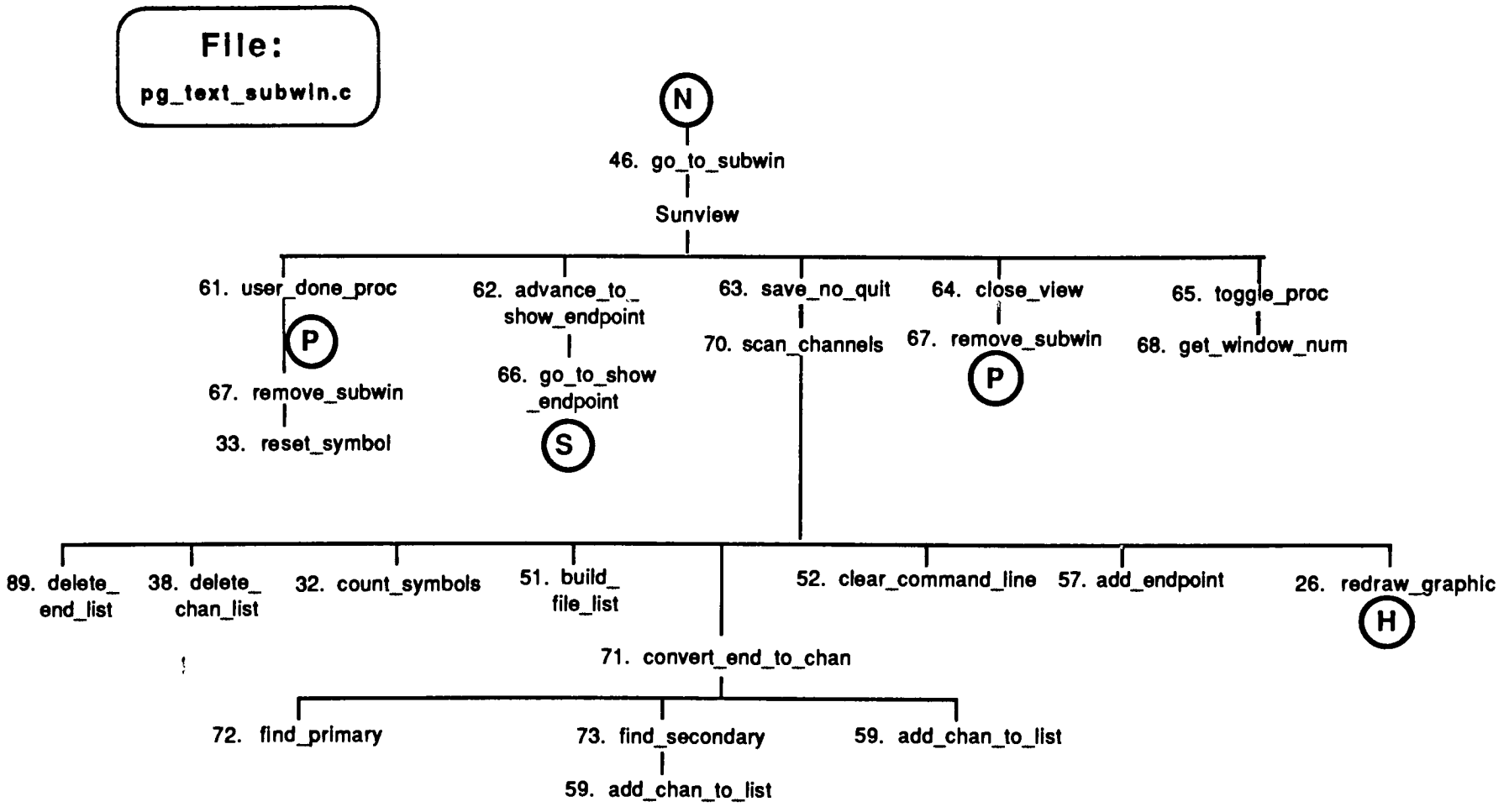
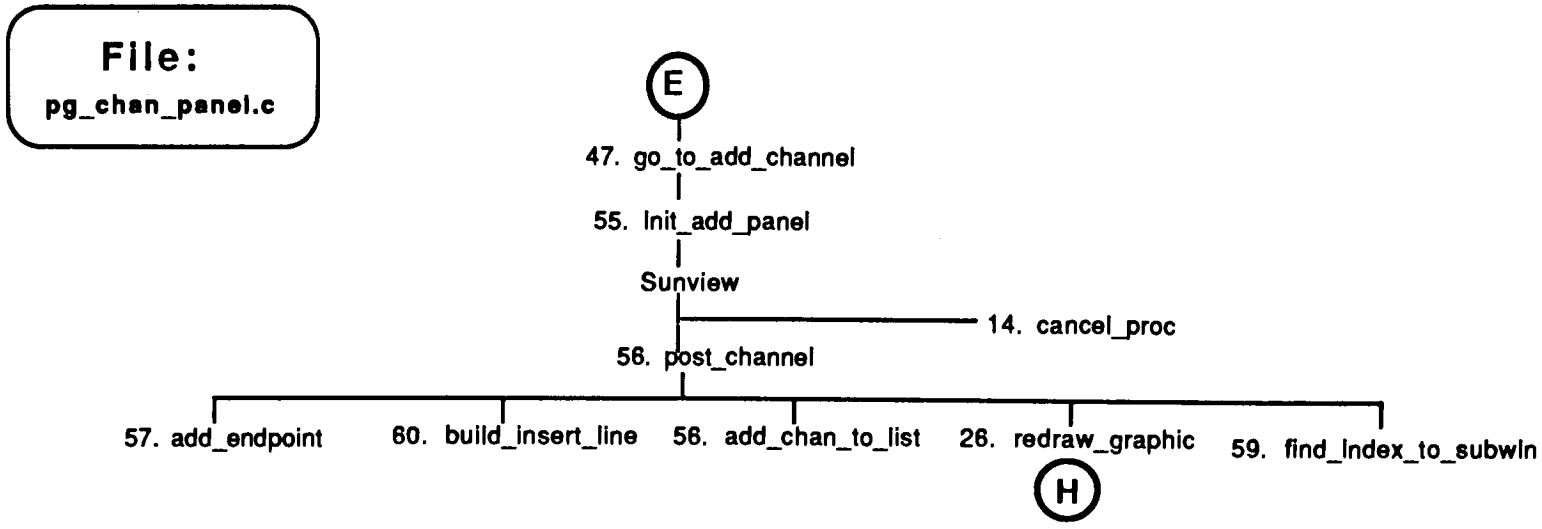


Fig. 6 (cont.) - PyGraph Function Call Hierarchy



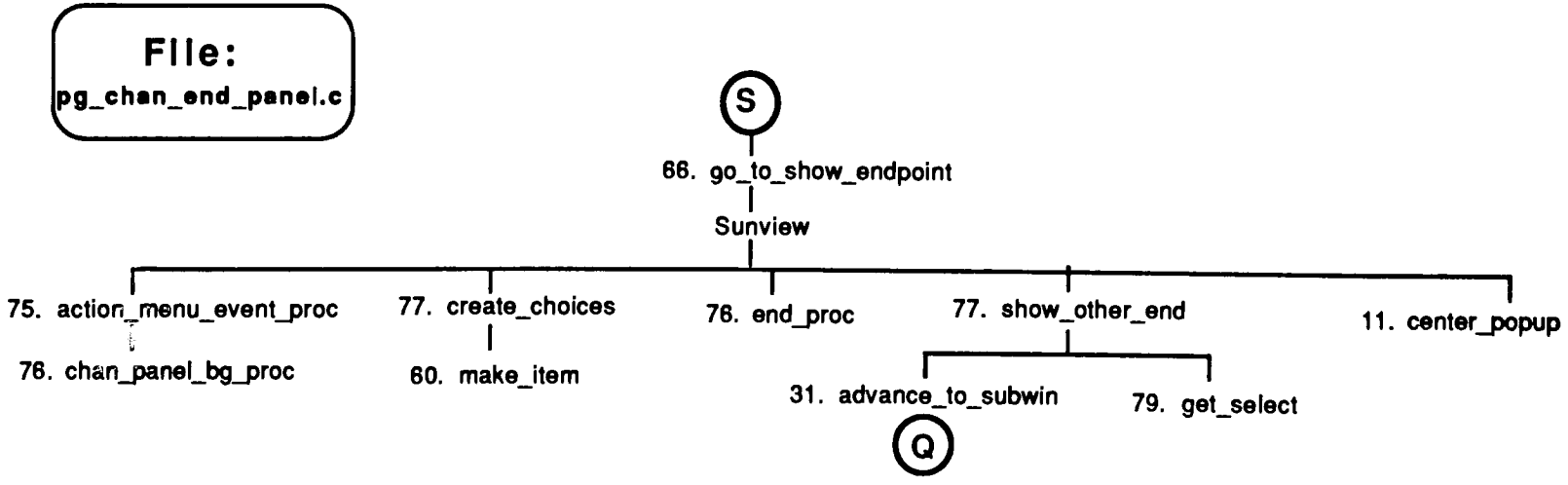


Fig. 6 (cont.) - PyGraph Function Call Hierarchy

PyGraph Function Descriptions

The following are short descriptions of the functions in PyGraph. The number preceding the definition corresponds to the numbered Hierarchy presented on the preceding pages.

1. main - creates the "base frame" for PyGraph. Includes initializations and calls out the directory panel.
2. go_to_directory - creates a panel with a message, quit button and quit menu.
3. generate_old_panel_buttons - creates the "New Graphic" button, plus the correct number of old graphic buttons.
4. dir_panel_event_proc - clarifies the event stream which could be a button select or menu request.
5. panel_bg_proc - shows the quit menu on mouse right.
6. quit_done - terminates the directory panel from button or from menu.
7. go_to_new_graphic_proc - initializations and advancement to the graphic development environment for new graphic.
8. go_to_old_graphic_proc - initializations and advancement to the graphic development environment environment for an existing graphic.
9. invoke_graphic_popup_proc - manages the popup panel that requests the user for a graphic name.
10. init_graphic_popup - creates the graphic popup.
11. center_popup - centers a popup on the display terminal.
12. return_to_graphic_from_popup - releases the terminal from

the i/o lock condition initiated by the popup.

13. strsave - creates a location for the directory buttons to associate a symbol name with them for when the button is selected.
14. cancel_proc - a special kind of return from a popup in which a cancel flag is set, eliminating any results from the popup.
15. go_to_graphic - creates the graphic development environment which consists of the symbol panel, drawing canvas and graphic menu.
16. symbol_panel_event_proc - handles the mouse right button which brings up the graphic menu over the symbol panel.
17. symbol_proc - changes the cursor upon choosing a symbol on the symbol panel.
18. canvas_event_proc - handles all the mouse events on the drawing canvas. This includes mouse left for selecting, mouse middle for dragging and mouse right for graphic menu.
19. save_graphic - stores the PyGraph drawing to file.
20. delete_graphic_proc - removes all of a graphics files from the directory.
21. invoke_popup_proc - manages the popup panel which obtains the input of a filename to store entire specification into.
22. print_screen_proc - manages the popup panel which obtains the input of a filename to store the screendump into.

23. advance_to_add_channel - verifies that certain criteria has been met before allowing user to add a channel.
24. return_to_directory_proc - manages the popup that asks the user if they want to save changes and executes per instruction, then returns to directory panel.
25. retrieve_graphic - for previously developed graphics, calls out the saved graphic image.
26. redraw_graphic - displays the most recent graphic image on the terminal.
27. archive_graphic - makes a duplicate copy of all the files associated with a graphic.
28. invert_symbol - inverts the current symbol being pointed to on the drawing canvas.
29. advance_to_add_symbol - manages the popup that asks user for the name of the symbol just placed on the canvas.
30. center_print_text - centers the symbol name under the symbol on the canvas.
31. advance_to_subwin - manages the number of open specification subwindows and advances the user to the specification.
32. count_symbols - provides a count of the number of current symbols on the canvas.
33. reset_symbol - upon closing of a specification subwindow, resets all of the variables that indicate that the window was open.
34. init_save_changes - creates the popup panel asking the user if they want to save the changes made to the

graphic and specifications.

35. unarchive_graphic - overlays the original version of the graphic onto the newest version.
36. pg_frame_destroyer - interposes on Sunview window menu selection of "Quit".
37. delete_pg_files - removes the newest version of the graphics files.
38. delete_chan_list - deallocates the linked list of existing channels on the canvas.
39. delete_symbol_list - deallocates the linked list of existing symbols on the canvas.
40. remove_archive - removes the archived files for a graphic.
41. init_popup - creates the popup panel that asks user to input file name to store entire specification into.
42. store_specs_to_file - concatenates all of the specifications for a graphic into a file.
43. init_input_panel - creates popup panel that asks user to input name of symbol just added to graphic.
44. add_symbol - allocates memory for linked list element that contains all information about a symbol on canvas.
45. process_template - upon adding a symbol, the appropriate template is customized for the symbol and placed into a file with the appropriate name.
46. go_to_subwin - creates the window that contains several buttons on a panel and the specification text

subwindow.

47. go_to_add_channel - creates the popup panel including interaction type choice switches, channel name input and management buttons.
48. add_chan_to_list - allocates memory to store the attributes of a channel that exists on the graphic.
49. seek_xy - provides the canvas coordinates of a given symbol.
50. response_proc - a special kind of return from a popup where the managing function needs to know whether the user answered yes or no.
51. build_file_list - creates a list of all the specification filenames that are included in a graphic.
52. clear_command_line - clears each element of the command_line character array.
53. invoke_delete_popup - creates the popup panel asking the user if they are sure they want to delete graphic.
54. init_screen_popup - creates the popup panel for the user to input the filename in which they want the screendump.
55. init_add_panel - creates the panel in which the user identifies the type of interaction that should occur over a channel and the channel name.
56. post_channel - posts new channel to storage and the channel name to the specifications in the text windows.
57. add_endpoint - makes linked list entries for each channel

endpoint.

58. add_chan_to_list - makes linked list entries for each channel on the graphic.
59. find_index_to_subwin - obtains the number of the text subwindow for a given symbol name.
60. build_insert_line - concatenates the channel interaction type and channel name into an array for eventual insert into the specification.
61. user_done_proc - handles the user selection of "done" on the Sunview window menu.
62. advance_to_show_endpoint - limits the number of open endpoint panels.
63. save_no_quit - causes the correct specifications to be scanned for channels without closing the text subwindow.
64. close_view - identifies which frame the request was called from.
65. toggle_proc - sets an array element which keeps track of the text subwindow that are currently eligible for the adding of a channel.
66. go_to_show_endpoint - creates the panel containing a list of the symbols and interaction types at the other end of a given symbol/channel combination.
67. remove_subwin - terminates a given textsubwindow after saving it.
68. get_window_num - determines the number of the textsubwindow that requested it.

69. delete_end_list - deallocates the memory that is held by the list of channel endpoints.
70. scan_channels - searches for channels in all specification files for a graphic and builds an endpoint list.
71. convert_end_to_chan - traverses the endpoint list to create a list of channels (two endpoints).
72. find_primary - searches the endpoint list to find the first occurrence of a given channel name and type.
73. find_secondary - searches the endpoint list to find the remaining occurrences of a given channel name and type.
74. calculate_num_files - determines the number of graphics that are in the user's directory.
75. action_menu_event_proc - manages the mouse interactions on the channel endpoint panel.
76. end_proc - terminates the channel endpoint panel.
77. create_choices - traverses the list of endpoints to find all occurrences of a channel and place them on the channel endpoint panel.
78. chan_panel_bg_proc - handles the display of action menu on the channel endpoint panel.
79. get_select - obtains the parameter of the user selection from the list of choices displayed.
80. make_item - sets the symbol name and interaction type on a channel endpoint panel item.
81. delete_symbol_proc - manages all the restriction, clean

up and rearranging of symbol linked list when a symbol is to be deleted.

82. remove_element - deletes everything associated with a symbol.

83. show_other_end - pulls up a new specification subwindow for the end of the channel selected to view.

6.6. PyGraph Data Structures.

Symbol Related:

The primary data structure in PyGraph is created when the user enters PyGraph and makes a request to create a new graphic. For each symbol that is entered, a linked list entry is made containing the symbol name, location on the canvas and whether it is inverted. The specification file for the symbol is also created at this time.

Channel Related:

There are two linked lists for channels. One is a list of endpoints, the other a list of channels which consist of two endpoints. It was decided to keep two lists to avoid having to frequently convert from one orientation to the other. Endpoints are needed when the user requests to see the other ends of a channel. Actual channels are needed when drawing channel vectors on the drawing canvas and when storing the graphic to file.

An array of other ends is utilized when the user requests to see the other ends of a channel. This array

minimizes the number of traversals made to the endpoint linked list.

Specification Subwindow Related:

An array of open specification subwindows is maintained to keep the user from entering a symbol specification more than once at a time and to control the number of such windows open.

Saving a Graphic:

When a graphic is saved by the user all of the contents of the symbol name list and the channel name list are saved into a file. The file name is "graphic name".gr. Each symbol also has a file for the storage of its specification. The file name for each of these is unique and formed as follows: "graphic name"_"symbol name".pg.

Archived Graphic:

Upon entering the Graphic Development Environment for a previously generated graphic, PyGraph automatically makes a duplicate copy of the entire graphic called the archive copy. All of the files in the graphic are copied into files of identical names except they have the phrase "saved_" in front of them. If the user wants to revert back to the original version, it is moved back to the current version.

7. Verification and Validation.

7.1. Testing Process.

Four types of testing needed to be done with PyGraph to verify that the system was working and that the output was

valid. The first type of testing is the most basic, that of confirming that the framework programmed using the Sunview macros provided the correct visual output.

Following the correct outward interpretation, it was necessary to verify the window interaction. A multi-window system such as PyGraph has an added complexity of dealing in multiple simultaneous windows. It is essential that the cycling from window to window is accomplished in a planned and controlled manner. This multi-window shell must operate as a firm platform before programming the in-depth algorithms. This step can drastically reduce confusion in later steps.

Once the algorithms providing the basic functionality of PyGraph are developed, they need to be tested for agreement with what is expected. This included a case by case analysis of feature and function.

Finally, one extra step needed to be accomplished with PyGraph. Because it is a graphic interface to PAISLey, the initial output from the PyGraph templates should provide a syntactically correct input to PAISLey. Thus, some basic PyGraph specification output needed to be run through the PAISLey interpreter to validate the PyGraph templates. A specification which included one of each of the templates needed to be run through PAISLey.

7.2. Test Results.

Attaining the correct visual output from PyGraph proved to be one of the simpler testing steps. One reason for this

was that Sunview and its inherent objects and capability lent itself well to the application at hand. It was a matter of choosing the Sunview capability to match what was desired or changing the design for improved user interface visual appearance.

The testing of the multi-windowing system was extremely time consuming and frustrating. It is suspected that the reason for this is that it is such an unfamiliar environment for most programmers. Until recently, windowing systems have not been accessible to most users. In particular, the Sunview system is new, user documentation is sparse and there is not a large or accessible group of advanced users available for assistance. Testing and development of the window management techniques thus was somewhat of a concurrent struggle. At the end of the process it clearly works well, but its due to discovery of the specific technique needed.

Testing of the algorithms and supporting data structures was often conducted with the assistance of dbxtool, a multiple window and visual version of dbx. This proved to be a very powerful and time saving package. Its hard to imagine ever going back to dbx.

Upon running the PAISLey parser and interpreter on some initial PyGraph produced specifications, it was found that the PAISLey syntax had changed since the PyGraph template source material was created. It was then necessary to update the templates, thus providing syntactically correct output.

8. Conclusions.

8.1. Problems Encountered and Solved.

Most of the problems developed around the Sunview windowing system. One specific situation involved an error message of "invalid instruction" whenever an entry was attempted into a popup text field. Dbxtool was not helpful in these situations because the error was occurring in Sunview routines which are essentially beyond programmer control. Particularly frustrating was the fact that the error was occurring in a region where no changes had been made. As it turned out, an integer was defined with the name "index" in a far downstream file/function. The compiler had not caught it as a reserved word or function. These are the kinds of problems that make a person question software development aides like Sunview. They are extremely helpful and make the application user friendly, but can be time consuming to learn and debug.

Several problems arose around the design of the user interface. At times it was unclear how much capability should be provided to user versus the inherent capability of the Sun 3 hardware/software versus the programming required in PyGraph to provide each of the features. One example was in the design of the channel creation/drawing/deletion process. Originally, all channels had to be entered via the "Add Channel Interaction" feature. Then it was decided that for a little more programming and use of the hardware, the "Scan Channels" capability could be added to keep the channel

lists/drawing updated at user request.

8.2. Limitations of the System.

The focus of this implementation is on creating a graphic implementation for specifying real-time systems with PAISLey. Therefore, the practical matter of outputting the full graphic specification was considered to be outside the focus of this research. This implementation does, however, include the ability to dump the screen contents to a file, but not the entire picture at once.

PyGraph is not intended to be an automatic PAISLey specification generator. The user has to be PAISLey literate. PyGraph requires the user to manipulate PAISLey specifications on the screen. However, the task is greatly simplified with provision of various features and built in capabilities.

The PyGraph Graphic Development Environment provides nine standard process symbols for placement on the drawing canvas. Ideally, a package would allow the user to define their own symbols for special applications and make free-form drawings on the template to further clarify the graphic. These enhancements to PyGraph have been left for future research.

Several internal limitations had to be set during the development of PyGraph. All are set with switches in the header file which can be easily changed. When accessing multiple specifications, only two specifications can be opened at once. This was set due to an existing UNIX

limitation on the Sun to approximately thirty-six devices. Since each window or open file is considered a device, with the large number of windows opened in PyGraph, the limit could be quickly reached causing a fatal error. Each specification window alone uses four "devices". Thirty graphics is the limit for a directory, the drawing canvas is 1000 X 1000 pixels and a channel can have at most fifteen other ends.

8.3. Lessons Learned.

One important lesson revolves around the concept of "feature creep". As the project moves along and it becomes clearer what the system looks like, additional features will not only be desired by the users, but will really make sense and be agreeable with the program writer. Consequently, everyone should plan for twenty percent or more time and cost to provide additional required capabilities and/or obvious enhancements.

Another important point is that working with a Sunview kind of development system, it appears inefficient to write only one application, then move on to something else. This is because there is a tremendous investment in time to learn how to use Sunview to put the pieces of a program together.

8.3.1. Alternative Approaches for Improved System.

PyGraph is well suited to the Sun system and at a macro level has a user friendly, well thought out appearance. At a more detailed level there may be some inefficiencies that could be modified to improve the system. One example exists

in the linked list, that contains the symbols as they are placed on the canvas. The name of the symbol is often used as the main key into the list. It may have been worthwhile to provide an alternate key, such as an identifying number for each symbol in a graphic. In this manner, some of the list traversing may occur quicker. For example, the list is searched every time the left mouse button goes down. If the cursor is over a symbol, PyGraph has to find out if the current inverted symbol is the one that is being pointed to. This involves string comparisons rather than integer comparisons. The manner in which it is programmed may be more straightforward, but also less efficient. Thus, there are tradeoffs, but an improved system may result if this were considered again.

8.3.2. Future Extensions and Related Thesis Topics.

Several areas exist for further enhancing PyGraph. A syntax directed editor could be incorporated that would check the grammatical input being made to a specification for accuracy. A related idea would involve a new data structure for the specifications. These could be held in a parse tree structure according to the grammar of the language. This would not necessarily have to check grammar on input as previously suggested.

A topic that appears to be getting growing attention, especially on Sun Workstations is the concept of animation. It would be advantageous for a designer or engineer to be able to view a graphic of their system in progress.

The drawing capabilities could be further enhanced beyond what is provided in this initial version. One can envision features that enable the creation of custom symbols and graphic and textual input directly onto the drawing canvas.

One last area is the matter of outputting the canvas to hard copy. A project could be undertaken that would enable a user to output the full graphic, no matter how large or small, to a laser printer.

9. Bibliography.

[1] AdaGraph - The Ada Reference Tool, "Reference Manual, Version 1.1", The Analytical Sciences Corporation, September 1986.

[2] Edward F. Berliner and Pamela Zave, "An Experiment in Technology Transfer: PAISLey Specification of Requirements for an Undersea Lightwave Cable System", Proceedings of the 9th International Conference on Software Engineering (to be published).

[3] Grady Booch, "Software Engineering with Ada - Second Edition", The Benjamin/Cummings Publishing Company, Inc., 1987.

[4] M.H. Brown and R. Sedgewick, "Techniques for Algorithm Animation", IEEE Software, Vol. 2, No. 1, Jan. 1985, pp. 28-39.

[5] Robert E. Filman and Daniel P. Friedman, "Coordinated Computing Tools and Techniques for Distributed Software", McGraw-Hill Book Co., 1984, pp.74-85.

[6] E.P. Glinert and S.L. Tanimoto, "PICT: An Interactive, Graphical Programming Environment", Computer, Vol. 17, No. 11, Nov. 1984, pp. 7-25.

[7] Robert B. Grafton and Tadao Ichikawa, "Visual Programming", Computer, Vol. 18, No. 8, Aug. 1985, pp. 6-9.

[8] M. Hamilton and S. Zeldin, "The Functional Life Cycle Model and its Automation: USE.IT", The Journal of Systems and Software, vol. 3, no. 1, March 1983.

[9] D. Harel, A. Pnueli, M. Politi, R. Sherman and A. Shtul-Trauring, "The AD CAD Methodology and STATEMATE1 Working Environment", CASE '87 - First International Workshop on Computer Aided Software Engineering c/o Index Technology Corporation, Advance Papers, Vol. 1, pp376-383.

[10] P. Hasenauer, "GPSSGraph - A Graphical Input Interface and Code Generator for the GPSS/H Simulation Language".

[11] Ralph L. London and Robert A. Duisberg, "Animating Programs Using Smalltalk", Computer, Vol. 18, No. 8, Aug. 1985, pp. 61-71.

[12] Georg Raeder, "A Survey of Current Graphical Techniques", Computer, Vol. 18, No. 8, Aug. 1985, pp. 11-25.

[13] P. Zave, "The Operational Versus the Conventional

Approach to Software Development", Communications of the ACM, Vol. 27, No. 2, Feb. 1984.

[14] P. Zave, "Anatomy of a Process Control System".

[15] P. Zave, "An Overview of the PAISLey Project-1984", ACM SIGSOFT SOFTWARE ENGINEERING NOTES, vol. 9, no. 4, pp. 12-19, July 1984.

[16] P. Zave, "An Operational Approach to Requirements Specification for Embedded Systems", IEEE Transaction on Software Engineering, vol. SE-8, no. 3, pp. 250-269, May 1982.

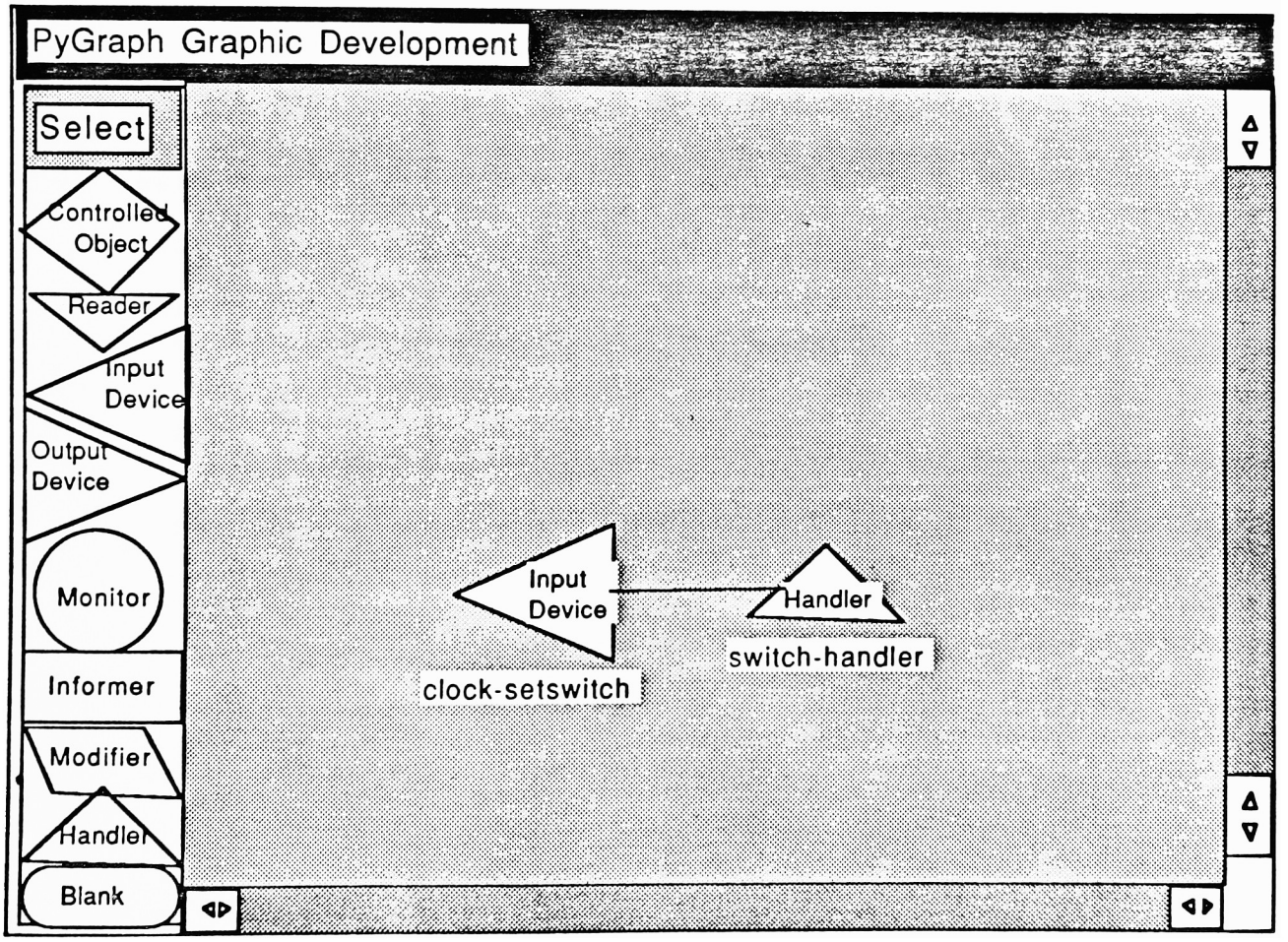
[17] P. Zave, "PAISLey User Documentation - Case Studies", Computer Technology Research Laboratory - AT&T Bell Laboratories, vol. 3, 1986.

[18] P. Zave, "PAISLey User Documentation - Reference Manual", Computer Technology Research Laboratory - AT&T Bell Laboratories, vol. 1, 1986.

[19] P. Zave, "PAISLey User Documentation - Tutorial", Computer Technology Research Laboratory - AT&T Bell Laboratories, vol. 2, 1986.

10. Example PyGraph Output

The following is a graphic depiction of a small example developed in PyGraph with two symbols. This graphic is not the actual screen print because of the absence of a printer to provide such a printout, but is identical with the graphic developed on PyGraph. The following pages are an actual specification output from PyGraph for the graphic.



```
(switch-handler-cycle[initial-switch-handler-state],
clock-setswitch-cycle[Null]
);

                                        "** clock-setswitch process "

"**" CLOCK-SETSWITCH-ITEM = REAL;

     clock-setswitch-cycle: FILLER --> FILLER;
     clock-setswitch-cycle[null] =
       clock-setswitch-give-input[clock-setswitch-input];

       clock-setswitch-input: --> CLOCK-SETSWITCH-ITEM;

       clock-setswitch-give-input: CLOCK-SETSWITCH-ITEM --> FILLER;

"**"           "Interaction site:"
                       clock-setswitch-give-input[s] =
                              x-clock-setting-value[s];


                                        "** switch-handler process "

"**" SWITCH-HANDLER-BUFFER = REAL;
"**" SWITCH-HANDLER-ITEM = REAL;

     switch-handler-empty-buffer: --> SWITCH-HANDLER-BUFFER;

     switch-handler-cycle : SWITCH-HANDLER-BUFFER -->
                                        SWITCH-HANDLER-BUFFER;
     switch-handler-cycle[b] =
       switch-handler-possible-consumption[
              switch-handler-possible-production[b]];

       switch-handler-possible-production:
              SWITCH-HANDLER-BUFFER --> SWITCH-HANDLER-BUFFER;

       switch-handler-possible-production[b] =

              /full[b]: b,
               True :
              switch-handler-append-item-if-any[(b,accept-item-if-any)
              ]
              /;

              switch-handler-full:
                       SWITCH-HANDLER-BUFFER --> BOOLEAN;


              switch-handler-accept-item-if-any:
                       --> SWITCH-HANDLER-ITEM | FILLER
"**"          "Interaction site:"
                       switch-handler-accept-item-if-any[s] =
                              x-clock-setting-value[s];
```

```
                switch-handler-append-item-if-any:

                        SWITCH-HANDLER-BUFFER *
                                (SWITCH-HANDLER-ITEM | FILLER)
                                        --> SWITCH-HANDLER-BUFFER;

switch-handler-possible-consumption:
                SWITCH-HANDLER-BUFFER --> SWITCH-HANDLER-BUFFER;
        switch-handler-possible-consumption[b] =

                /empty[b]: b,
                 True : switch-handler-possible-buffer-restore
                     [(b,switch-handler-offer-consumable-item
                          [switch-handler-highest-priority-item[b]])]
                /;

                switch-handler-empty:
                        SWITCH-HANDLER-BUFFER --> BOOLEAN;

                switch-handler-highest-priority-item:
                        SWITCH-HANDLER-BUFFER --> SWITCH-HANDLER-ITEM;

                switch-handler-offer-consumable-item:
                        SWITCH-HANDLER-ITEM --> FILLER |
                                        SWITCH-HANDLER-ITEM;
"**"            "Potential interaction site:
                        switch-handler-offer-consumable-item[s] =
                                        x?[s];"

                switch-handler-possible-buffer-restore:

                        SWITCH-HANDLER-BUFFER *
                                (FILLER | SWITCH-HANDLER-ITEM)
                                        --> SWITCH-HANDLER-BUFFER;

                switch-handler-possible-buffer-restore[(b,i)] =
                        /null[i]: b,
                        True : switch-handler-restore-to-buffer[(b,i)]
                        /;

                switch-handler-restore-to-buffer:
                        SWITCH-HANDLER-BUFFER *
                                SWITCH-HANDLER-ITEM
                                    --> SWITCH-HANDLER-BUFFER;
```

11. PyGraph Program Listing

File: pg.h

```
#include        <stdio.h>
#include        <suntool/sunview.h>
#include        <suntool/scrollbar.h>
#include        <suntool/panel.h>
#include        <suntool/canvas.h>
#include        <suntool/textsw.h>
#include        <suntool/walkmenu.h>
#include        <suntool/seln.h>

#define         MAX_SUBWINS             2
#define         MAX_GRAPHIC_NAME        20
#define         MAX_SYMBOL_NAME         15
#define MAX_FILENAME            40
#define MAX_INPUT_FILENAME      15
#define         MAX_CHANNEL_NAME        32
#define         MAX_COMMAND             60
#define         MAX_GRAPHICS            30
#define         MAX_CANVAS_HEIGHT       1000
#define         MAX_CANVAS_WIDTH        1000
#define MAX_CHANNELS            15
#define         MAX_LINE                80
#define MAX_INT                         2
#define MAX_CHOICES             15

static short    icon_image[] = {
#include        "./pg_icons/sy_PyGraph.picon"
};
DEFINE_ICON_FROM_IMAGE(PyGraph_icon, icon_image);


extern Frame    pg_base_frame;
extern Canvas   pg_canvas;
extern Panel    pg_symbol_panel;
extern Panel    pg_channel_panel;

extern void pg_init_symbol_panel();
extern void pg_init_canvas();
extern void go_to_directory();
extern void go_to_graphic();
extern void go_to_subwin();
extern void go_to_add_channel();
extern void go_to_show_endpoint();
extern void return_to_graphic_from_popup();
extern void add_chan_to_list();
extern void delete_chan_list();
extern void cancel_proc();
extern void reset_symbol();

extern center_popup();
extern int times_into_graphic;
extern int times_back_to_directory;
extern int left, top;
int     new_graphic,
        in_graphic,     /* indicates that user currently in graphic, not
                        ** dir */
```

```
        showing_other_end,    /* indicates that currently showing other
                              ** end */
        entry_made,
        count_of_symbols,
        chan_end_displayed;

                        /* used in system calls */
char    command_line[MAX_COMMAND];


extern char     *cur_symbol1,   /* Current first symbol activated */
                *cur_symbol2;

                        /* Channel to remember for subwindow search */
char    save_this_channel[MAX_CHANNEL_NAME];

                /* structure for managing open spec subwindows */
struct subwin  {
                        char    name[MAX_SYMBOL_NAME];
                        int     toggle_value,
                                in_use;
} sub_list[MAX_SUBWINS];

Frame  pg_subwin_frame[MAX_SUBWINS];
Textsw textsw[MAX_SUBWINS];
Panel  select_panel[MAX_SUBWINS];
Menu_item       m_item;
Menu            t_menu;

/*****************************************************************/
/*  used to reference linked list of channel names */

struct chan_rec {
                        char    sym_name[MAX_SYMBOL_NAME];
                        int     interact_type_1;
                        char    sym_name_other_end[MAX_SYMBOL_NAME];
                        int     interact_type_2;
                        char    chan_name[MAX_CHANNEL_NAME];
                                struct  chan_rec        *next;
                };

struct chan_rec         *chan_new,
                        *chan_head,
                        *chan_tail,
                        *chan_old,
                        *ptr_chan_ahead,
                        *ptr_chan_behind,
                        *save_chan;
/*****************************************************************/
/* declaration of channel endpoint list */
struct end_rec {
                        char    sy_name[MAX_SYMBOL_NAME];
                        int     inter;
                        char    ch_name[MAX_CHANNEL_NAME];
                        int     u_num;
```

```
                        struct  end_rec *next;
                };

struct end_rec          *end_head,
                        *end_tail,
                        *end_old,
                        *end_new,
                        *ptr_walk_chan,
                        *end1,
                        *end2,
                        *primary,
                        *starter;

/*****************************************************************/
/* Things used to maintain linked list for symbol names */
struct symbol_rec {
                        char    symbol_name[MAX_SYMBOL_NAME];
                        int     symbol_type,
                                symbol_x,
                                symbol_y,
                                inverted,
                                open;
                                struct symbol_rec *next,
                                                  *prev;
                };

struct symbol_rec       *head,  /* ptr to start of symbol name list */
                        *tail,  /* pointer to end of print list */
                        *new,   /* used to add new symbols onto list */
                        *old,   /* used in saving/deleting symbol_name
                                ** list */
                        *ptr_to_previous,/* saves current inverted */
                        *relative_ptr,
                        *rel_ptr,
                        /* current symbol activated for spec entry */
                        *cur_symbol,

                                /* for walking the symbol list */
                        *ptr_before, *ptr_after,
                        *traverse;

FILE    *temp_fp;
```

```
#include "./pg.h"

                        /* base frame */
Frame  pg_base_frame;

Notify_value   pg_frame_destroyer();

                        /* for managing the multiple window environment
                        */
int    times_back_into_directory, times_into_graphic,
times_into_subwin,
       delete_graphic, /* for cancelling a popup panel */
       cancel;

                        /* specification subwindow declarations */
Frame  pg_subwin_frame[MAX_SUBWINS];
Textsw textsw[MAX_SUBWINS];
Panel  select_panel[MAX_SUBWINS];

                        /* vars used to pass channel info between files
                        */
int    int_1, int_2, saved_yet;
char   sym_end_1, sym_end_2, chan_text;

                        /* references for interposing */
extern void    delete_pg_files();
extern void    unarchive_graphic();
extern void    save_graphic();
extern void    remove_archive();
```

```
/*****************************************************************/
/* creates the base frame and calls out directory panel ********/
main()
{
int     i, j;

        pg_base_frame = window_create(NULL, FRAME,
                        FRAME_LABEL,            "   PyGraph Directory",
                        FRAME_ICON,             &PyGraph_icon,
                        WIN_WIDTH,              900,
                        WIN_HEIGHT,             720,
                        WIN_ERROR_MSG,          "Can't create window.",
                         0);

        cancel = 0;
        new_graphic = 1;
        times_back_into_directory = 0;
        times_into_graphic = 0;
        times_into_subwin = 0;
        chan_end_displayed = 0;
        showing_other_end = 0;

        chan_head = NULL;
        end_head = NULL;

                        /* intialize array to manage open subwindows */
        for(i = 0; i < MAX_SUBWINS; i++)
        {
                sub_list[i].toggle_value = 0;
                sub_list[i].in_use = -1;
                for(j = 0; j < MAX_SYMBOL_NAME; j++)
                        sub_list[i].name[j] = '\0';
        }

        go_to_directory();

        notify_interpose_destroy_func(pg_base_frame,
                pg_frame_destroyer);
        window_main_loop(pg_base_frame);
}
```

```
/*****************************************************************/
/* interposes on the selection of "Quit" off the Sunview window menu*/
Notify_value
pg_frame_destroyer(frame, status)
Frame  frame;
Destroy_status status;
{
        if((status == DESTROY_CHECKING) && (new_graphic == 0) &&
                        (in_graphic == 1))
        {
                        /* clean up and restore original version */
                delete_pg_files();
                unarchive_graphic();
        }
        else if((status == DESTROY_CHECKING) && (new_graphic == 1) &&
                        (in_graphic == 1))
                save_graphic();
                remove_archive();

        window_set(pg_base_frame, FRAME_NO_CONFIRM, TRUE, 0);
        return(notify_next_destroy_func(frame, status));
}
```

```
#include "./pg.h"

                        /* objects included with the directory display
                        */
Frame           pg_base_frame;
Panel           panel;
Panel_item      graphic_name_item, new_item, old_item[MAX_GRAPHICS];
Menu            close_session_menu;
Frame  pop_graphic_frame;
Panel  pop_graphic_entry_panel;

                        /* directory_panel functions */
void            generate_old_panel_buttons();
void            panel_bg_proc();
void            dir_panel_event_proc();
void            go_to_new_graphic_proc(), go_to_old_graphic_proc();
void            quit_done();
void            cancel_proc();
void            calculate_num_files();

/*********************************************************************/
/* Popup graphic name request functions ******************/

void   invoke_graphic_popup_proc();
void   init_graphic_popup();
/*********************************************************************/

int             times_back_into_directory,      /* whether to create dir
                                                   panel*/
                cancel,                     /*whether operation has been
                                            cancelled*/
                a_symbol_inverted,          /* if a symbol is inverted */
                files_found,    /* number spec files for graphic */
                delete_graphic; /* if the graphic is being deleted*/

                        /* the name of graphic */
char            *save_name, graphic_name_save[MAX_FILENAME],
                        /* the name of symbol */
                *strsave();
```

```
/*********************************************************************/
/* creates and manages the directory panel */
void
go_to_directory()
{
int k;

    in_graphic = 0;

            /* do these things the first time into directory */
    if (times_back_into_directory == 0)
    {

       times_back_into_directory = 1;

       window_set(pg_base_frame,
                  FRAME_LABEL,            "   PyGraph",
                  0);

       panel = window_create(pg_base_frame, PANEL,
                  WIN_WIDTH,      WIN_EXTEND_TO_EDGE,
                  WIN_VERTICAL_SCROLLBAR, scrollbar_create(
                           SCROLL_PLACEMENT,       SCROLL_EAST,
                            0),
                  PANEL_BACKGROUND_PROC,  panel_bg_proc,
                  PANEL_EVENT_PROC, dir_panel_event_proc,
                  0);

       panel_create_item(panel, PANEL_MESSAGE,
              PANEL_LABEL_STRING, "Hit left mouse button to make
                     selection.",
              PANEL_LABEL_X,  20,
              PANEL_LABEL_Y,  10,
              0);

       panel_create_item(panel, PANEL_BUTTON,
              PANEL_LABEL_IMAGE, panel_button_image(panel, "QUIT",
                     5, 0),
              PANEL_ITEM_X,   520,
              PANEL_ITEM_Y,   10,
              PANEL_NOTIFY_PROC, quit_done,
              0);

       new_item = panel_create_item(panel, PANEL_BUTTON,
              PANEL_LABEL_IMAGE, panel_button_image
                     (panel, "CREATE NEW GRAPHIC", 50, 0),
              PANEL_ITEM_X,   170,
              PANEL_ITEM_Y,   60,
              PANEL_NOTIFY_PROC, go_to_new_graphic_proc,
              0);

                     /* generate panel buttons via function */
       calculate_num_files();
       generate_old_panel_buttons();
```

```
        close_session_menu = menu_create (MENU_TITLE_ITEM, "Quit?",
                                  MENU_ACTION_ITEM, "Yes", quit_done,
                                  0);

    }
                /* do these things every time but first into dir */
    else
    {
        window_set(pg_base_frame,
                FRAME_LABEL,              "   PyGraph",
                0);

        if (delete_graphic == 1)
        {
                delete_graphic = 0;
        }

                        /* make the right buttons on dir panel */
        for(k = 0; k < files_found; k++)
                panel_destroy_item(old_item[k]);
        generate_old_panel_buttons();

        window_set(panel, WIN_SHOW, TRUE, 0);
    }


}
```

```
/*************************************************************************/
/* creates the "New Graphic" button plus the correct number of old
       graphic buttons */
void
generate_old_panel_buttons()
{

int     f, j, l, distance_down;
                        /* Operating system command line variables */
char    *ptr_to_command, command_line[MAX_COMMAND];
FILE    *graphic_name_pipe, *fopen();

        calculate_num_files();
        if(files_found > MAX_GRAPHICS)
                files_found = MAX_GRAPHICS;
        if(files_found != 0)
        {
                graphic_name_pipe = popen("ls -c *.gr", "r");

                        /* Create panel button for each file */
                for (f = 0; f < files_found; f++)
                {
                        for (j = 0; j < MAX_FILENAME; j++)
                                graphic_name_save[j] = '\0';
                                        /* read through file name */
                        fscanf(graphic_name_pipe, "%s",
                                graphic_name_save);
                        j = MAX_FILENAME;
                        while((graphic_name_save[j] == '\0') ||
                                (graphic_name_save[j] == '\n') ||
                                        (graphic_name_save[j] == EOF))
                                --j;
                        for(l = 0; l < 3; l++)
                        {
                                graphic_name_save[j] = '\0';
                                --j;
                        }

                                /* how far down to place the button on
                                        panel */
                        distance_down = (40 * f) + 100;
                        save_name = graphic_name_save;

                                /* initialize selection buttons */
                        old_item[f] = panel_create_item(panel,
                                PANEL_BUTTON,
                                PANEL_LABEL_IMAGE, panel_button_image
                                    (panel, graphic_name_save, 50, 0),
                                PANEL_CLIENT_DATA,  strsave(save_name),
                                PANEL_ITEM_X,           170,
                                PANEL_ITEM_Y,           distance_down,
                                PANEL_NOTIFY_PROC,
                                        go_to_old_graphic_proc,
                                0);
```

```
                }
        }  /* end if .gr files found */
}
```

```
/*****************************************************************/
/* event procs to handle mouse buttons on panel ******************/
void
panel_bg_proc(panel, event)
Panel  panel;
Event  *event;
{

        if(event_id(event) == MS_RIGHT)
                menu_show(close_session_menu, panel, event, 0);
}

void
dir_panel_event_proc(item, event)
Panel_item      item;
Event           *event;
{
        panel_bg_proc();
        if (event_id(event) == MS_RIGHT)
        {
                Event   *adjusted_event;
                        adjusted_event = panel_event(panel, event);
                        panel_bg_proc(panel, adjusted_event);
        }
        else
                panel_default_handle_event(item, event);
}
```

```
/*****************************************************************/
/* terminates the directory panel from button or menu */
void
quit_done()
{

        window_set(pg_base_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(pg_base_frame);
}

/*****************************************************************/
/* inits and movement to the graphic development environment for new
        graphic */
void
go_to_new_graphic_proc()
{
                        /* Request user to input name of graphic */
        invoke_graphic_popup_proc();
        if(cancel == 0)
        {
                window_set(panel, WIN_SHOW, FALSE, 0);
                new_graphic = 1;
                a_symbol_inverted = 0;
                go_to_graphic();
        }
        else
                cancel = 0;
}
```

```
/*****************************************************************/
/* inits and movement to the graphic development environment for old
        graphic */
void
go_to_old_graphic_proc(item, event)
Panel_item item;
Event  *event;
{
        if(cancel == 0)
        {
                window_set(panel, WIN_SHOW, FALSE, 0);
                new_graphic = 0;
                save_name = panel_get(item, PANEL_CLIENT_DATA);
                strcpy(graphic_name_save, save_name);
                go_to_graphic();
        }
        else
                cancel = 0;
}
```

```
/*************************************************************************/
/* Popup frame to acquire graphic name ***************************/
void
invoke_graphic_popup_proc()
{
int     z, unique_name, jump_out;
char    *check_name;

                init_graphic_popup();

                /* check that user enters a graphic name */
        entry_made = 0;
        unique_name = 0;
        while((entry_made == 0) || (unique_name == 0))
        {
                /* assume an entry will be made and it's unique */
                entry_made = 1;
                unique_name = 1;

                        /* force user to provide a filename or cancel */
                window_loop(pop_graphic_frame);

                        /* eliminate some steps for a cancelled popup */
                if(cancel == 0)
                {
                        /* Assign name of graphic to pointer variable */
                        save_name = panel_get_value(graphic_name_item);
                        if(strlen(save_name) != 0)
                        {
                                        /* check for unique entry */
                                z = 0;
                                jump_out = 0;
                                while((z < files_found) &&
                                                (unique_name != 0) &&
                                                        (jump_out == 0))
                                {
                                        check_name =
                                                panel_get(old_item[z],
                                                PANEL_CLIENT_DATA);
                                        unique_name = strcmp(check_name,
                                                save_name);
                                        if(unique_name == 0)
                                                jump_out = 1;
                                        z++;
                                }
                                if(unique_name != 0)
                                   strcpy(graphic_name_save, save_name);
                        }
                        else
                                entry_made = 0;
                }
        }  /* end while no entry */

                                /* On the return, get rid of popup */
        window_set(pop_graphic_frame, FRAME_NO_CONFIRM, TRUE, 0);
```

```
        window_destroy(pop_graphic_frame);
}
```

```
/*****************************************************************/
/* creates the graphic name request popup */
void
init_graphic_popup()
{
        pop_graphic_frame = window_create(NULL, FRAME,
                                FRAME_SHOW_LABEL,       FALSE,
                                0);


        pop_graphic_entry_panel = window_create(pop_graphic_frame,
                                        PANEL,
                                0);

        panel_create_item(pop_graphic_entry_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING, "Enter Name for Graphic: ",
                0);

        graphic_name_item = panel_create_item(
                                pop_graphic_entry_panel, PANEL_TEXT,
                PANEL_VALUE_DISPLAY_LENGTH,     MAX_GRAPHIC_NAME,
                PANEL_VALUE_STORED_LENGTH,      MAX_GRAPHIC_NAME,
                0);

        panel_create_item(pop_graphic_entry_panel,      PANEL_BUTTON,
                PANEL_LABEL_IMAGE,
                        panel_button_image(pop_graphic_entry_panel,
                         "Done", 5, 0),
                PANEL_ITEM_X,   240,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      return_to_graphic_from_popup,
                0);

        panel_create_item(pop_graphic_entry_panel,      PANEL_BUTTON,
                PANEL_LABEL_IMAGE,
                        panel_button_image(pop_graphic_entry_panel,
                         "Cancel", 5, 0),
                PANEL_ITEM_X,   140,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      cancel_proc,
                0);

        window_fit(pop_graphic_entry_panel);
        window_fit(pop_graphic_frame);

        center_popup(pop_graphic_frame);

        window_set(pop_graphic_frame, WIN_X,    left,
                                WIN_Y,          top,
                                0);
}
```

```
/*****************************************************************/
/* Save the name of each graphic with the button */

static char *
strsave(source)
char *source;
{
   char *dest;

   dest = (char *) malloc(strlen(source) + 1);
   if (!dest)
      return NULL;

   strcpy(dest, source);
   return dest;
}

/*****************************************************************/
/* routine used to cancel a popup ***********/

void
cancel_proc()
{
       cancel = 1;
       window_return();
}

/*****************************************************************/
/* determines the number of graphics that are out in the directory */
void
calculate_num_files()
{
FILE   *gn_fp, *fopen();

                       /* count .gr files in the directory */
       system("ls *.gr 2> /dev/null | wc -w > pg_graphics.pg");
       gn_fp = fopen("pg_graphics.pg", "r");
       fscanf(gn_fp, "%d", &files_found);
       fclose(gn_fp);
                       /* clean up */
       system("rm pg_graphics.pg");
}
```

File: pg_symbol_panel.c

```
#include        "./pg.h"

/*****************************************************************/
/* identify where all the icons are stored ***********************/

static short    pg_image[] = {
#include        "./pg_icons/pg.picon"
};
mpr_static(pg_pixrect, 16, 16, 1, pg_image);

static short    controlled_image[] = {
#include        "./pg_icons/sy_controlled.picon"
};
mpr_static(controlled_pixrect, 64, 64, 1, controlled_image);

static short    reader_image[] = {
#include        "./pg_icons/sy_reader.picon"
};
mpr_static(reader_pixrect, 64, 64, 1, reader_image);

static short    input_image[] = {
#include        "./pg_icons/sy_input.picon"
};
mpr_static(input_pixrect, 64, 64, 1, input_image);

static short    output_image[] = {
#include        "./pg_icons/sy_output.picon"
};
mpr_static(output_pixrect, 64, 64, 1, output_image);

static short    monitor_image[] = {
#include        "./pg_icons/sy_monitor.picon"
};
mpr_static(monitor_pixrect, 64, 64, 1, monitor_image);

static short    informer_image[] = {
#include        "./pg_icons/sy_informer.picon"
};
mpr_static(informer_pixrect, 64, 64, 1, informer_image);

static short    modifier_image[] = {
#include        "./pg_icons/sy_modifier.picon"
};
mpr_static(modifier_pixrect, 64, 64, 1, modifier_image);

static short    handler_image[] = {
#include        "./pg_icons/sy_handler.picon"
};
mpr_static(handler_pixrect, 64, 64, 1, handler_image);

static short    blank_image[] = {
#include        "./pg_icons/sy_blank.picon"
};
mpr_static(blank_pixrect, 64, 64, 1, blank_image);
```

```
static short    select_image[] = {
#include        "./pg_icons/sy_select.picon"
};
mpr_static(select_pixrect, 64, 64, 1, select_image);

Pixrect         *icons[] = {
                        &select_pixrect,
                        &controlled_pixrect,
                        &reader_pixrect,
                        &input_pixrect,
                        &output_pixrect,
                        &monitor_pixrect,
                        &informer_pixrect,
                        &modifier_pixrect,
                        &handler_pixrect,
                        &blank_pixrect,
};

/*****************************************************************/
/* Pointer to all of the specification template files **********/

char    *controlled_temp = "./temp_controlled.temp",
        *handler_temp = "./temp_handler.temp",
        *informer_temp = "./temp_informer.temp",
        *input_temp = "./temp_input.temp",
        *modifier_temp = "./temp_modifier.temp",
        *monitor_temp = "./temp_monitor.temp",
        *output_temp = "./temp_output.temp",
        *reader_temp = "./temp_reader.temp",
        *blank_temp = "./temp_blank.temp";

/*****************************************************************/
/* Pointers to user assigned names ***********************************/

char                                    /* name of current symbol */
        *input_symbol_name, name_of_symbol_in[MAX_SYMBOL_NAME];
extern char     graphic_name_save[];
char    out_file[MAX_FILENAME];


/*****************************************************************/
/* objects included with the graphic development environment */
Frame  pg_base_frame;
Panel  pg_symbol_panel;
Panel_item symbol_item;
Menu   graphic_menu;
Cursor cursor;

                        /* misc functions */
void    quit_proc();
void    return_to_directory_proc();
void    symbol_panel_event_proc();
void    advance_to_add_channel();
symbol_proc();
center_popup();
```

```
process_template();
void    save_graphic();
void    clear_command_line();

int     times_into_graphic,     /* whether to create the wins or just
                                        show */
        left, top,              /* used to center popups on screen */
        symbol_choice,          /* selected symbol number off panel */
        times_for_symbol,       /* Only allow paste of one symbol per
                                        select */
        cancel;

/*****************************************************************/
/* Canvas related declarations ***********************************/
Canvas pg_canvas;

void    canvas_event_proc();
void    center_print_text();
void    advance_to_subwin();
void    retrieve_graphic();
void    redraw_graphic();
void    delete_symbol_list();
void    invert_symbol();
void    delete_symbol_proc();
void    remove_element();
void    count_symbols();


Cursor normal_cursor;
Cursor symbol_cursor;

int     cur_loc_x, cur_loc_y,   /* Current cursor location on screen */
        text_loc_x, text_loc_y, /* Location of text of symbol */
                /* save location variables for inverted images */
        save_inverted_x, save_inverted_y, save_inverted_type,
                /* traverse list of symbols until symbol found */
        found,
                        /* a status of symbol on canvas */
        inverted,
                        /* a symbol is/is not inverted on the canvas */
        a_symbol_inverted,
        set_to_symbol_cursor,
        count_of_channels;

/*****************************************************************/
/* vars used to pass channel stuff between files */
int     int_1, int_2;
char    sym_end_1[MAX_SYMBOL_NAME], sym_end_2[MAX_SYMBOL_NAME],
        chan_text[MAX_CHANNEL_NAME];
char    *cur_symbol_1, *cur_symbol_2;

/*****************************************************************/
/* Popup specification save filename request functions ***********/
void    invoke_popup_proc();
void    init_popup();
```

```
void   store_specs_to_file();
void   build_file_list();

Frame  pop_file_frame;
Panel  pop_entry_panel;
Panel_item spec_save_file_item;

/*********************************************************************/
/* Popup symbol name request functions ******************************/
void   advance_to_add_symbol_name();
void   init_input_panel();
void   return_to_graphic_from_popup();

Frame  input_panel_frame;
Panel  input_panel;
Panel_item symbol_name_item;

/*********************************************************************/
/* Popup delete graphic stuff ***************************************/
void   delete_graphic_proc();
void   invoke_delete_popup();
void   delete_pg_files();
void   response_proc();
extern void    delete_end_list();

Frame  delete_graphic_frame;
Panel  delete_panel;

int    delete_graphic;

/*********************************************************************/
/* Popup screendump file stuff */
void   print_screen_proc();
void   init_screen_popup();
void   remove_screen_popup();

Frame  screen_file_frame;
Panel  screen_panel;
Panel_item     screendump_item;

/*********************************************************************/
/* Popup to make decisions on archiving  */
Frame  save_changes_frame;
Panel  save_changes_panel;

void   init_save_changes();
void   archive_graphic();
void   unarchive_graphic();
void   remove_archive();

/*********************************************************************/
/* channel related declarations  ************************************/
extern void    seek_xy();
extern void    scan_channels();
                    /* locational values returned by function */
```

```
int     sym_x, sym_y;
```

```
/*****************************************************************/
/* creates the graphic development environment */
void
go_to_graphic()
{
                        /* indicate that user is in the graphic */
        in_graphic = 1;
                        /* display graphic window header */
        window_set(pg_base_frame,
                        FRAME_LABEL,    "   PYGRAPH GRAPHIC
                                        DEVELOPMENT",
                        0);

                        /* Only create the Graphic Development
                        ** Environment once per session */
if (times_into_graphic == 0)
{

        times_into_graphic = 1;

        pg_symbol_panel = window_create (pg_base_frame, PANEL,
                        PANEL_LAYOUT,           PANEL_VERTICAL,
                        WIN_ROW_HEIGHT,         64,
                        WIN_COLUMN_WIDTH,       64,
                        WIN_LEFT_MARGIN,        2,
                        WIN_RIGHT_MARGIN,       2,
                        WIN_ROWS,               10,
                        WIN_COLUMNS,            1,
                        WIN_EVENT_PROC, symbol_panel_event_proc,
                        0);

        symbol_item = panel_create_item(pg_symbol_panel, PANEL_CHOICE,
                        PANEL_CHOICE_IMAGES,    icons[0], icons[1],
                                icons[2], icons[3], icons[4], icons[5],
                                icons[6], icons[7], icons[8], icons[9],
                                0,
                        PANEL_FEEDBACK,         PANEL_INVERTED,
                        PANEL_NOTIFY_PROC,      symbol_proc,
                        0);

        pg_canvas = window_create(pg_base_frame, CANVAS,
                WIN_RIGHT_OF,   pg_symbol_panel,
                CANVAS_AUTO_SHRINK,     FALSE,
                WIN_VERTICAL_SCROLLBAR, scrollbar_create(
                        SCROLL_PLACEMENT,       SCROLL_EAST, 0),
                WIN_HORIZONTAL_SCROLLBAR, scrollbar_create(
                        SCROLL_PLACEMENT,       SCROLL_SOUTH, 0),
                CANVAS_HEIGHT,  MAX_CANVAS_HEIGHT,
                CANVAS_WIDTH,   MAX_CANVAS_WIDTH,
                WIN_CONSUME_PICK_EVENT,         LOC_DRAG,
                WIN_EVENT_PROC, canvas_event_proc,
                0);

        graphic_menu = menu_create(MENU_TITLE_ITEM,     "Graphic Menu",
                MENU_ACTION_ITEM,       "Save Graphic",
```

```
                        save_graphic,
                MENU_ACTION_ITEM,       "Delete Graphic",
                        delete_graphic_proc,
                MENU_ACTION_ITEM,       "Put Specification to File",
                        invoke_popup_proc,
                MENU_ACTION_ITEM,       "Screen Dump",
                        print_screen_proc,
                MENU_ACTION_ITEM,       "Add Channel",
                        advance_to_add_channel,
                MENU_ACTION_ITEM,       "Update Channels",
                        scan_channels,
                MENU_ACTION_ITEM,       "Delete Symbol",
                        delete_symbol_proc,
                MENU_ACTION_ITEM,       "Quit - Go to PyGraph
                        Directory",
                        return_to_directory_proc,
                0);

}
 else
{
                /* In a given session, all but pg_canvas will only
                ** be initialized once.  From there, it will simply
                ** be redisplayed with the current graphic */

        window_set(pg_symbol_panel, WIN_SHOW, TRUE, 0);
        window_set(pg_canvas, WIN_SHOW, TRUE, 0);

}

                        /* initialize linked list of symbols */
        head = NULL;
        tail = NULL;
        old = NULL;

        if (new_graphic == 0)   /* old graphic initializations only */
        {
                   /* load symbols from the files of chosen graphic */
                retrieve_graphic();

                        /* fill canvas with the chosen image */
                redraw_graphic();

                        /* make an archival copy of the original spec */
                archive_graphic();
                found = 0;

        }  /* end initializations for old graphics */
}  /* end go_to_graphic */
```

```
/*************************************************************************/
/* Event procedures to provide mouse interactions in the */
/* Graphic Development Environment */
void
symbol_panel_event_proc(pg_symbol_panel, event)
Event  *event;
{

        if (event_id(event) == MS_RIGHT)
                menu_show(graphic_menu, pg_symbol_panel, event, 0);
        else
                panel_default_handle_event(symbol_item, event);
}

void
canvas_event_proc(pg_canvas, event)
Canvas pg_canvas;
Event  *event;
{
Pixwin *pw;
                        /* corrects the event space for scrollable
                                window */
Event  *cor_event;
int    ms_mid_down;     /* senses the current setting of middle mouse
                                button */
char   *name_of_symbol;

        found = 0;
                        /* location of cursor on the terminal */
        cur_loc_x = event_x(event);
        cur_loc_y = event_y(event);
                        /* get canvas pixwin */
        pw = canvas_pixwin(pg_canvas);

                        /* handle mouse right to give menu */
        if (event_id(event) == MS_RIGHT)
                menu_show(graphic_menu, pg_canvas,
                        canvas_window_event (pg_canvas, event), 0);

                                /* handle mouse left for
                                        selecting/inverting */
        if ((event_id(event) == MS_LEFT) && (event_is_down(event)))
        {
                                /* determine if user is pointing to
                                        symbol */
                if (symbol_choice == 0)  /* select mode */
                {
                                /* traverse list backwards until find
                                        symbol */
                        old = tail;
                        while ((old != NULL) && (found == 0))
                        {
                                if ((old->symbol_x < cur_loc_x) &&
                                        ((old->symbol_x + 64) >
                                                cur_loc_x) &&
```

```
        (old->symbol_y < cur_loc_y) &&
        ((old->symbol_y + 64) >
                cur_loc_y))
{
   found = 1;

        /* determine if symbol inverted
        */
   if(old->inverted == 1)
   {
        /* if that symbol inverted,
                uninvert */
      invert_symbol(old->symbol_x,
         old->symbol_y,
                old->symbol_type);
      old->inverted = 0;
      a_symbol_inverted = 0;

        if(event_shift_is_down(event))
        /* if shift down, go to text
                subwin */
        {
             /* open text_subwindow */
             cur_symbol = old;
             name_of_symbol =
                cur_symbol->symbol_name;
                advance_to_subwin(
                        name_of_symbol);

        }
   }  /* end if already inverted */
   else  /* may be pointing to another
         ** symbol than the one that is
         ** inverted */
   {
        /* for graphic with symbol
                inverted */
      if(a_symbol_inverted == 1)
      {
                /* uninvert previous
                        symbol */
        invert_symbol(save_inverted_x,
                save_inverted_y,
                 save_inverted_type);
        ptr_to_previous->inverted = 0;
        a_symbol_inverted = 0;
      }

        /* save current symbol pointed
        ** to's attributes */
      save_inverted_x = old->symbol_x;
      save_inverted_y = old->symbol_y;
      save_inverted_type =
                old->symbol_type;
      ptr_to_previous = old;
```

```
                        /* invert symbol being pointed
                                to */
                        invert_symbol(save_inverted_x,
                                save_inverted_y,
                                 save_inverted_type);
                        old->inverted = 1;
                        a_symbol_inverted = 1;
                    }  /* end else */
                }   /* end if pointing at a symbol */
                old = old->prev;
            }  /* end while */
            found = 0;
    }  /* end if symbol_choice is 0 */

                /* place an object on the canvas */
    if ((symbol_choice != 0) && (times_for_symbol == 0))
    {
                /* destroy symbol_cursor */
        set_to_symbol_cursor = 0;
        window_set(pg_canvas, WIN_CURSOR, normal_cursor, 0);
        cursor_destroy(symbol_cursor);

                /* prepare "inverted" for the add_symbol
                */
        inverted = 1;

                /* ask user for symbol name */
        advance_to_add_symbol_name();
        if(cancel == 0)
        {
                /* Limit user to one of that symbol */
            times_for_symbol = 1;

                /* uninvert previously inverted symbol
                ** if there is one */
            if(a_symbol_inverted == 1)
            {
                invert_symbol(save_inverted_x,
                        save_inverted_y,
                        save_inverted_type);
                ptr_to_previous->inverted = 0;
                a_symbol_inverted = 0;
            }

                /* write symbol on canvas */
            pw_rop(pw, cur_loc_x, cur_loc_y, 64, 64,
                PIX_SRC,
                icons[symbol_choice], 0, 0);

                /* invert that symbol for the user */
            invert_symbol(cur_loc_x, cur_loc_y,
                symbol_choice);
            save_inverted_x = cur_loc_x;
            save_inverted_y = cur_loc_y;
```

```
                save_inverted_type = symbol_choice;
                a_symbol_inverted = 1;

                        /* center and print the name under
                                symbol */
                center_print_text();
                        /* keep pointer to new list element to
                        ** be able to do an uninvert */
                ptr_to_previous = new;

            }  /* end if not cancelled */
            else
                /* cancel add of symbol occurred */
                cancel = 0;

                /* set back to select mode */
            panel_set_value(symbol_item, 0);
            symbol_choice = 0;

        }  /* end if not select mode */

}  /* end if MS_LEFT */

                                /* Handle drag of symbol */
if (((event_id(event) == MS_MIDDLE) && (event_is_down(event)))
                && (symbol_choice == 0))
{
        old = tail;
        while ((old != NULL) && (found == 0))
        {
                if ((old->symbol_x < cur_loc_x) &&
                        ((old->symbol_x + 64) > cur_loc_x) &&
                        (old->symbol_y < cur_loc_y) &&
                        ((old->symbol_y + 64) > cur_loc_y) &&
                        (old->inverted == 1))
                {
                        found = 1;
                                /* erase symbol at old location
                                ** then uninvert */
                        invert_symbol(old->symbol_x,
                                old->symbol_y,
                                old->symbol_type);
                                /* XOR */
                        pw_rop(pw, old->symbol_x, old->symbol_y,

                                64, 64, PIX_SRC ^ PIX_DST,
                                icons[old->symbol_type], 0, 0);

                                /* get next event */
                        window_read_event(pg_canvas, event);
                        cor_event = canvas_event(pg_canvas,
                                        event);
                        while (1)
                        {
                             if((event_id(cor_event) ==
```

```
                        MS_MIDDLE) &&
                    (event_is_up(cor_event)))
                        break;

                    /* erase symbol at old location
                    */
                pw_rop(pw, cur_loc_x, cur_loc_y,
                    64, 64, PIX_SRC ^ PIX_DST,
                    icons[old->symbol_type], 0, 0);

                cur_loc_x = event_x(cor_event);
                cur_loc_y = event_y(cor_event);

                    /* place symbol at new location
                    */
                pw_rop(pw, cur_loc_x, cur_loc_y,
                    64, 64, PIX_SRC ^ PIX_DST,
                    icons[old->symbol_type], 0, 0);

                    /* get next event */
                window_read_event(pg_canvas,
                        event);
                cor_event = canvas_event(pg_canvas,
                                event);
        }  /* end while middle button down */

                /* final XOR */
        pw_rop(pw, cur_loc_x, cur_loc_y,
                64, 64, PIX_SRC ^ PIX_DST,
                icons[old->symbol_type], 0, 0);
        old->symbol_x = cur_loc_x;
        old->symbol_y = cur_loc_y;

                /* move symbol record to end of
                ** list to signify it as most
                ** recent */

                /* eliminate case of single
                ** symbol or its at end of list
                ** already */
        if(((old->next != NULL) && (old->prev ==
                NULL)) || (old->next != NULL))
        {
                /* handle beginning & middle
                        cases */

                ptr_before = old->prev;
                if(ptr_before == NULL)
                {
                        /* first in list */
                        head = old->next;
                        tail->next = old;
                        old->next = NULL;
                        old->prev = tail;
                        tail = old;
```

```
                                        head->prev = NULL;
                                    }
                                    else   /* middle of list */
                                    {
                                        ptr_after = old->next;
                                        ptr_before->next =
                                                ptr_after;
                                        ptr_after->prev =
                                                ptr_before;
                                        tail->next = old;
                                        old->next = NULL;
                                        old->prev = tail;
                                        tail = old;
                                    }
                                } /* end if not already at end of list
                                   */

                                   /* now redraw the graphic in order */
                                redraw_graphic();

                            } /* end if correct symbol */
                            if(found == 0)
                                    old = old->prev;
                        } /* end while */
                } /* end if MS_MIDDLE */
}
```

```
/*****************************************************************/
/* function to invert a symbol on the canvas *********************/
void
invert_symbol(x, y, t)
int    x, y, t;
{
Pixwin *pw;

        pw = canvas_pixwin(pg_canvas);
        pw_rop(pw, x, y, 64, 64, PIX_NOT(PIX_DST), icons[t], 0, 0);
}

/*****************************************************************/
/* function to center the symbol name underneath the symbol on a canvas
*/
void
center_print_text()
{
Pixwin *pw;
                        /* variables used in centering symbol name text
                        */
int    extra_char, pixels, num_char;

        pw = canvas_pixwin(pg_canvas);
                        /* first center symbol name under symbol */
        num_char = strlen(input_symbol_name);
        extra_char = num_char - 6;
        if (extra_char == 0)
                text_loc_x = cur_loc_x;
        else
                if (extra_char < 0)
                {
                        pixels = ((- extra_char) / 2) * 11;
                        text_loc_x = cur_loc_x + pixels;
                }
                else
                {
                        pixels = ((extra_char) / 2) * 11;
                        text_loc_x = cur_loc_x - pixels;
                }

        text_loc_y = cur_loc_y + 84;
                        /* print the text on the canvas */
        pw_text(pw, text_loc_x, text_loc_y, PIX_SRC,
                0, input_symbol_name);
}
```

```
/*****************************************************************/
/* manages the popup that asks user if they want to save changes and
** executes per instruction, then returns to directory panel */
void
return_to_directory_proc()
{
Pixwin *pw;
int     canvas_width, canvas_height,
        save_changes;

        count_symbols();
        if(delete_graphic == 0)
        {

                init_save_changes();

                save_changes = (int) window_loop(save_changes_frame);

                        /* do the following if not cancelled */
                if(save_changes != 2)
                {

                                /* check if any subwins open and close
                                        them */
                        relative_ptr = head;
                        while(relative_ptr != NULL)
                        {
                                if(relative_ptr->open != -1)
                                {
                                        textsw_save(textsw
                                                [relative_ptr->open],
                                                100, 100);
                                        window_set(pg_subwin_frame
                                                [relative_ptr->open],
                                                FRAME_NO_CONFIRM, TRUE,
                                                0);
                                        window_destroy(pg_subwin_frame
                                                [relative_ptr->open]);
                                        reset_symbol(
                                                relative_ptr->open);
                                }
                                relative_ptr = relative_ptr->next;
                        }

                        if(save_changes == 0)
                                        /* restore old files to new */
                        {
                                delete_pg_files();
                                if(new_graphic == 0)
                                        unarchive_graphic();
                        }
                        else
                        {
                                /* save all changes since last save */
                                save_graphic();
```

```
                        remove_archive();
                    }
                }

                    /* get rid of save changes popup */
                window_set(save_changes_frame, FRAME_NO_CONFIRM, TRUE,
                        0);
                window_destroy(save_changes_frame);
        }
        else
                    /* on delete, remove the archive copy also */
                remove_archive();

        if(save_changes != 2)
        {
                        /* eliminate lists of channels */
                delete_chan_list();
                delete_end_list();
                        /* clear canvas */
                delete_symbol_list();
                canvas_width = (int) window_get(pg_canvas,
                        CANVAS_WIDTH);
                canvas_height = (int) window_get(pg_canvas,
                        CANVAS_HEIGHT);
                pw = canvas_pixwin(pg_canvas);
                pw_writebackground(pw, 0, 0, canvas_width,
                        canvas_height, PIX_SRC);

                window_set(pg_symbol_panel, WIN_SHOW, FALSE, 0);
                window_set(pg_canvas, WIN_SHOW, FALSE, 0);

                go_to_directory();
        }
}
```

```
/*****************************************************************/
/* creates popup panel asking the user if they want to save changes
** prior to closing */
void
init_save_changes()
{
        save_changes_frame = window_create(NULL, FRAME,
                        FRAME_SHOW_LABEL,       FALSE,
                        0);

        save_changes_panel = window_create(save_changes_frame, PANEL,
                        0);

        panel_create_item(save_changes_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING,
                        "Save changes before closing?",
                0);

        panel_create_item(save_changes_panel,   PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        save_changes_panel, "Yes", 5, 0),
                PANEL_ITEM_X,   20,
                PANEL_ITEM_Y,   35,
                PANEL_CLIENT_DATA,      1,
                PANEL_NOTIFY_PROC,      response_proc,
                0);

        panel_create_item(save_changes_panel,   PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        save_changes_panel, "No", 5, 0),
                PANEL_ITEM_X,   110,
                PANEL_ITEM_Y,   35,
                PANEL_CLIENT_DATA,      0,
                PANEL_NOTIFY_PROC,      response_proc,
                0);

        panel_create_item(save_changes_panel,   PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        save_changes_panel, "Cancel", 5, 0),
                PANEL_ITEM_X,   200,
                PANEL_ITEM_Y,   35,
                PANEL_CLIENT_DATA,      2,
                PANEL_NOTIFY_PROC,      response_proc,
                0);

        window_fit(save_changes_panel);
        window_fit(save_changes_frame);

        center_popup(save_changes_frame);

        window_set(save_changes_frame, WIN_X,   left,
                                WIN_Y,          top,
                                0);
}
```

```
/*****************************************************************/
/** Procedure to execute upon the choosing of a symbol on the symbol
** panel and associated changing of the cursor shape */
symbol_proc(item, value)
Panel_item item;
int    value;
{
       symbol_choice = value;

                              /* if previously created cursor without
                              ** destroying it, do it here */
       if (set_to_symbol_cursor == 1)
       {
              window_set(pg_canvas, WIN_CURSOR, normal_cursor, 0);
              cursor_destroy(symbol_cursor);
              set_to_symbol_cursor = 0;
       }

                     /* change the cursor */
       switch(symbol_choice) {

       case 0 :  break;
       default : times_for_symbol = 0;
                 set_to_symbol_cursor = 1;
                 normal_cursor = window_get(pg_canvas, WIN_CURSOR);
                 symbol_cursor = window_get(pg_canvas, WIN_CURSOR);
                 symbol_cursor =
                       cursor_create(CURSOR_IMAGE, &pg_pixrect, 0);
                 cursor_set(symbol_cursor, CURSOR_OP, PIX_SRC, 0);
                 window_set(pg_canvas, WIN_CURSOR, symbol_cursor, 0);
       }

}
```

```
/*****************************************************************/
/** Popup panel to obtain input of file to store entire spec into */
void
invoke_popup_proc()
{
char   *save_file;

                        /* check that user enters a file name */
   entry_made = 0;
   init_popup();

   while(entry_made == 0)
   {
                        /* assume that an entry will be made */
       entry_made = 1;

       /* force user to provide a filename or cancel */
       window_loop(pop_file_frame);

       if(cancel == 0)
       {
               save_file = panel_get_value(spec_save_file_item);
               if(strlen(save_file) != 0)
                                /* make and fill the spec save file */
                       store_specs_to_file(save_file);
               else
                       entry_made = 0;
       }
       else
               cancel = 0;

   }  /* end while no entry */

       /* On the return, get rid of popup */
       window_set(pop_file_frame, FRAME_NO_CONFIRM, TRUE, 0);
       window_destroy(pop_file_frame);
}
```

```
/*****************************************************************/
/* create the popup for obtaining filename to save spec into */
void
init_popup()
{
        pop_file_frame = window_create(NULL, FRAME,
                                FRAME_SHOW_LABEL,       FALSE,
                                0);


        pop_entry_panel = window_create(pop_file_frame, PANEL,
                                0);

        panel_create_item(pop_entry_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING, "Enter Store Filename for
                        Specification: ",
                0);

        spec_save_file_item = panel_create_item(pop_entry_panel,
                PANEL_TEXT,
                PANEL_VALUE_DISPLAY_LENGTH,     MAX_INPUT_FILENAME,
                PANEL_VALUE_STORED_LENGTH,      MAX_INPUT_FILENAME,
                0);

        panel_create_item(pop_entry_panel,      PANEL_BUTTON,
                PANEL_LABEL_IMAGE, panel_button_image(pop_entry_panel,
                        "Done", 5, 0),
                PANEL_ITEM_X,   240,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      return_to_graphic_from_popup,
                0);

        panel_create_item(pop_entry_panel,      PANEL_BUTTON,
                PANEL_LABEL_IMAGE, panel_button_image(pop_entry_panel,
                        "Cancel", 5, 0),
                PANEL_ITEM_X,   140,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      cancel_proc,
                0);

        window_fit(pop_entry_panel);
        window_fit(pop_file_frame);

        center_popup(pop_file_frame);

        window_set(pop_file_frame, WIN_X,       left,
                                WIN_Y,          top,
                                0);
}
```

```
/*****************************************************************/
/* Popup to obtain the symbol name for a symbol placed on the canvas */
void
advance_to_add_symbol_name()
{
int     unique_entry;
char    *ptr_to_symbol_name;

        unique_entry = 0;
        entry_made = 0;

                        /* bring up popup */
        init_input_panel();

        while((unique_entry == 0) || (entry_made == 0))
        {
                        /* assume that a unique entry will be made */
                unique_entry = 1;
                entry_made = 1;

                                /* limit user to that window */
                window_loop(input_panel_frame);

                if(cancel == 0)
                {
                                /* Assign symbol name of symbol */
                        input_symbol_name =
                                panel_get_value(symbol_name_item);

                                /* check for an entry */
                        if(strlen(input_symbol_name) != 0)
                        {
                                /* check for duplicate symbol_name */
                           relative_ptr = head;
                           while((relative_ptr != NULL) && (unique_entry
                                        != 0))
                           {
                                ptr_to_symbol_name =
                                        relative_ptr->symbol_name;
                                unique_entry =
                                  strcmp(ptr_to_symbol_name,
                                   input_symbol_name);
                                   relative_ptr = relative_ptr->next;
                           }

                           if(unique_entry != 0)
                           {
                                /* initialize symbol to linked list */
                                inverted = 1;
                                add_symbol();
                                        /* build spec for that symbol */
                                process_template();

                           } /* end if unique */
                        } /* end if entry  made */
```

```
                    else
                            /* force user to make entry */
                        entry_made = 0;

                } /* end if popup not cancelled */

        } /*  end while not a unique symbol_name and name entered */

                    /* get rid of add_chan_frame */
        window_set(input_panel_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(input_panel_frame);
}
```

```
/*************************************************************************/
/* create the popup requesting the input of a symbol name */
void
init_input_panel()
{
        input_panel_frame = window_create(NULL, FRAME,
                        FRAME_SHOW_LABEL,       FALSE,
                        0);

        input_panel = window_create(input_panel_frame, PANEL,
                        0);

        panel_create_item(input_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING, "Enter name for symbol:",
                0);

        symbol_name_item = panel_create_item(input_panel, PANEL_TEXT,
                PANEL_VALUE_DISPLAY_LENGTH,     MAX_SYMBOL_NAME,
                PANEL_VALUE_STORED_LENGTH,      MAX_SYMBOL_NAME,
                0);

        panel_create_item(input_panel,  PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        input_panel, "Done", 5, 0),
                PANEL_ITEM_X,   240,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      return_to_graphic_from_popup,
                0);

        panel_create_item(input_panel,  PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(input_panel,
                        "Cancel", 5, 0),
                PANEL_ITEM_X,   140,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      cancel_proc,
                0);

        window_fit(input_panel);
        window_fit(input_panel_frame);

        center_popup(input_panel_frame);

        window_set(input_panel_frame, WIN_X,    left,
                                WIN_Y,          top,
                                0);
}
```

```
/*****************************************************************/
/* Function to add a symbol to the linked list of symbols in a graphic
*/
add_symbol()
{
int j;

        new = (struct symbol_rec *) malloc
                  (sizeof (struct symbol_rec)) ;

        for (j = 0; j < MAX_SYMBOL_NAME; j++)
                 new->symbol_name[j] = '\0';
        strcpy(new->symbol_name, input_symbol_name);
        new->symbol_type = symbol_choice;
        new->symbol_x = cur_loc_x;
        new->symbol_y = cur_loc_y;
        new->inverted = inverted;
        new->open = -1;
        new->next = NULL;

                        /* first in list */
        if (head == NULL)
        {
                head = new ;
                tail = new ;
                new->prev = NULL;
        }
        else            /* add to end of list */
        {

                        /* find out what is previous symbol in list */
                ptr_after = head->next;
                ptr_before = head;
                while(ptr_after != NULL)
                {
                        ptr_before = ptr_before->next;
                        ptr_after = ptr_after->next;
                }


                tail->next = new ;
                tail = new ;
                        /* assign pointer to previous symbol */
                new->prev = ptr_before;
        }

}
```

```
/*****************************************************************/
/* upon adding a symbol, the appropriate template is customized for the
** symbol and placed into a file with the appropriate name */
process_template()
{
int     c, d, i;
FILE    *input_fp, *output_fp, *fopen();
char    *in_file,
        *walk_symbol_name;

        switch(symbol_choice) {
                        case 1 :        in_file = controlled_temp;
                                        break;
                        case 2 :        in_file = reader_temp;
                                        break;
                        case 3 :        in_file = input_temp;
                                        break;
                        case 4 :        in_file = output_temp;
                                        break;
                        case 5 :        in_file = monitor_temp;
                                        break;
                        case 6 :        in_file = informer_temp;
                                        break;
                        case 7 :        in_file = modifier_temp;
                                        break;
                        case 8 :        in_file = handler_temp;
                                        break;
                        case 9 :        in_file = blank_temp;
                                        break;
        }
        if ((input_fp = fopen(in_file, "r")) == NULL)
        {
                printf("\n Cannot open input file: %s\n", in_file);
                exit (1);
        }

                        /* clear the out_file array */
        for (i = 0; i < MAX_FILENAME; i++)
                out_file[i] = '\0';
        strcat(out_file, graphic_name_save) ;
        strcat(out_file, "_") ;
        strcat(out_file, input_symbol_name) ;
        strcat(out_file, ".pg");

        if ((output_fp = fopen(out_file, "w")) == NULL)
        {
                printf("\n Cannot open output file: %s\n", out_file);
                exit(1);
        }

                        /* read through the file making substitutions */
        while ((c = getc(input_fp)) != EOF)
        {
                if (c != '~')
                        fputc(( (char) c ), output_fp);
```

```
                else
                {
                        c = getc(input_fp);
                        switch (c) {

                        case '1' : fputs(input_symbol_name, output_fp);
                                   break;
                        case '2' : fputs("  \"Note - Default value\"  ",
                                                output_fp);
                                   break;
                        case '3' : fputs(" = REAL", output_fp);
                                   break;
                        case '6' : for (walk_symbol_name =
                                                input_symbol_name;
                                                *walk_symbol_name !=
                                                        '\0';
                                                  walk_symbol_name++)
                                   {
                                        d = (int) *walk_symbol_name;
                                        if ('a' <= d && d <= 'z')
                                                fputc((char)(d + 'A' -
                                                  'a'), output_fp);
                                        else
                                                fputc(*walk_symbol_name,
                                                    output_fp);
                                    }
                        }

                }
        }
        fclose(input_fp);
        fclose(output_fp);
}
```

```
/*****************************************************************/
/* Generic popup panel routine - centers popup panel on the terminal */
center_popup(any_frame)
Frame any_frame;
{
Rect    *r;
int     width, height;

        r = (Rect *) window_get(any_frame, WIN_SCREEN_RECT);
                width = (int) window_get(any_frame, WIN_WIDTH);
                height = (int) window_get(any_frame, WIN_HEIGHT);
                left = (r->r_width - width) / 2;
                top = (r->r_height - height) / 2;
                if (left < 0)
                        left = 0;
                if (top < 0)
                        top = 0;
}

/*****************************************************************/
/* Generic popup routine - eliminates window i/o lock caused by
** window_loop*/
void
return_to_graphic_from_popup()
{
        window_return();
}
```

```
/*****************************************************************/
/* manages the number of open spec subwindows */
void
advance_to_subwin(symbol_go, sent_sym)
char    *symbol_go;
struct symbol_rec       *sent_sym;
{
int     l, m, found_free_slot;
char    spec_file_name[MAX_FILENAME];

                        /* allow a maximum of text subwindows open at
                                once */
                        /* find an open slot, if available */
                found_free_slot = 0;
                l = 0;
                while((l < MAX_SUBWINS) && (found_free_slot == 0))
                {
                        if((sub_list[l].in_use == -1) && (sent_sym->open
                                == -1))
                        {
                           found_free_slot = 1;
                           sub_list[l].in_use = 1;

                                /* store symbol name */
                           strcpy(sub_list[l].name, symbol_go);

                           for(m = 0; m < MAX_FILENAME; m++)
                                spec_file_name[m] = '\0';

                                /* build filename to pass to subwin */
                           strcat(spec_file_name, graphic_name_save);
                           strcat(spec_file_name, "_");
                           strcat(spec_file_name, symbol_go);
                           strcat(spec_file_name, ".pg");
                                /* mark the symbol as open */
                           sent_sym->open = l;
                           go_to_subwin(l, spec_file_name);
                        }
                        l++;
                }  /* end while */
}
```

```
/*****************************************************************/
/* verifies that criteria has been met before allowing a user to add
channel */
void
advance_to_add_channel()
{
int     i, count_open, first;

        count_open = 0;
        for(i = 0; i < MAX_SUBWINS; i++)
                if(sub_list[i].toggle_value == 1)
                        count_open++;

        if(count_open == 2)
        {
                first = 0;
                        /* identify to which symbols the add channel
                                occurs */
                for(i = 0; i < MAX_SUBWINS; i++)
                        if(sub_list[i].toggle_value == 1)
                                if(first == 0)
                                {
                                        cur_symbol_1  =
                                                sub_list[i].name;
                                        first = 1;
                                }
                                else
                                        cur_symbol_2 = sub_list[i].name;

                go_to_add_channel(cur_symbol_1, cur_symbol_2);
        }
}
```

```
/*****************************************************************/
/* Routine to provide all operations necessary to save a graphic */
void
save_graphic()
{
int     i;
FILE    *graphic_fp, *fopen();
char    graphic_file[MAX_FILENAME];

        new_graphic = 0;
                        /* clear the graphic_file array */
        for (i = 0; i < MAX_FILENAME; i++)
                graphic_file[i] = '\0';
        strcat(graphic_file, graphic_name_save) ;
        strcat(graphic_file, ".gr");

        if ((graphic_fp = fopen(graphic_file, "w")) == NULL)
        {
                printf("\n Cannot open graphic save file: %s\n",
                        graphic_file);
                exit(1);
        }

                        /* traverse list to count number of symbols */
        count_symbols();

                        /* save linked list of symbol names to file */
        fprintf(graphic_fp, "%d\n", count_of_symbols);
        old = head;
        while (old != NULL)
        {
                fprintf(graphic_fp, "%s %d %d %d %d\n",
                        old->symbol_name, old->symbol_type,
                        old->symbol_x, old->symbol_y,
                        old->inverted);
                old = old->next;
        }

                        /* save existing channels */
                        /* traverse list to count # channels */
        count_of_channels = 0;
        chan_old = chan_head;
        while (chan_old != NULL)
        {
                count_of_channels++;
                chan_old = chan_old->next;
        }

                        /* save linked list of symbol names to file */
        fprintf(graphic_fp, "%d\n", count_of_channels);
        chan_old = chan_head;
        while (chan_old != NULL)
        {
                fprintf(graphic_fp, "%s %d\n", chan_old->sym_name,
                        chan_old->interact_type_1);
```

```
            fprintf(graphic_fp, "%s %d\n",
                    chan_old->sym_name_other_end,
                    chan_old->interact_type_2);
            fprintf(graphic_fp, "%s\n", chan_old->chan_name);
            chan_old = chan_old->next;
        }

        fclose(graphic_fp);

                    /* remove system generated % files */
        clear_command_line();
        strcat(command_line, "rm ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, "_*.pg% 2> /dev/null");
        system(command_line);

                    /* copy current save files to an archived copy
                    */
        archive_graphic();
}
```

```
/*****************************************************************/
/* deallocates the linked list of existing symbols on the canvas */
void
delete_symbol_list()
{

                        /* clean up/deallocate linked list to system */
        old = head ;
        while (old != NULL)
        {
                head = old->next ;
                free (old) ;
                old = head ;
        }
        head = NULL;
}
```

```
/*****************************************************************/
/* for previously developed graphics call out the saved graphic image
*/
void
retrieve_graphic()
{
int     i, j, symbol_num;
FILE    *build_graphic_fp, *fopen();
char    *ptr_chan_in, *ptr_s1_in, *ptr_s2_in,
        graphic_file[MAX_FILENAME];

                                /* Prepare file to access */
        strcpy(graphic_file, graphic_name_save) ;
        strcat(graphic_file, ".gr");

        if ((build_graphic_fp = fopen(graphic_file, "r")) == NULL)
        {
                printf("\nCannot retrieve saved graphic: %s\n",
                        graphic_file);
                exit(1);
        }

                        /* build linked list of symbol names from file
                        */
        fscanf(build_graphic_fp, "%d\n", &count_of_symbols);
        if(count_of_symbols != 0)
        {
                for (i = 0; i < count_of_symbols; i++)
                {
                        fscanf(build_graphic_fp, "%s %d %d %d %d\n",
                                name_of_symbol_in, &symbol_choice,
                                &cur_loc_x, &cur_loc_y, &inverted);
                        input_symbol_name = name_of_symbol_in;
                        add_symbol();
                }

                        /* call out channels on the graphic */
                fscanf(build_graphic_fp, "%d\n", &count_of_channels);
                for (i = 0; i < count_of_channels; i++)
                {
                        fscanf(build_graphic_fp, "%s %d\n", sym_end_1,
                                        &int_1);
                        fscanf(build_graphic_fp, "%s %d\n", sym_end_2,
                                        &int_2);
                        fscanf(build_graphic_fp, "%s\n", chan_text);
                        ptr_s1_in = sym_end_1;
                        ptr_s2_in = sym_end_2;
                        ptr_chan_in = chan_text;
                                /* build list of channels */
                        add_chan_to_list(ptr_s1_in, ptr_s2_in, int_1,
                                int_2, ptr_chan_in);
                        add_endpoint(ptr_s1_in, int_1, ptr_chan_in);
                        add_endpoint(ptr_s2_in, int_2, ptr_chan_in);
                }
                        /* close file */
```

```
            fclose(build_graphic_fp);

        }  /* end steps for retrieved graphics with symbols */

                        /* starting symbol is select */
        symbol_choice = 0;
        panel_set_value(symbol_item, 0);

}
```

```
/*****************************************************************/
/* Routine to traverse linked list and redraw graphic ************/
void
redraw_graphic()
{
Pixwin *pw;
int    s1x, s1y, s2x, s2y, canvas_width, canvas_height;

char   *symbol_name_to_print,
       *ptr_symbol_1, *ptr_symbol_2;

       pw = canvas_pixwin(pg_canvas);
       a_symbol_inverted = 0;

                     /* clear background for redraw */
       canvas_width = (int) window_get(pg_canvas, CANVAS_WIDTH);
       canvas_height = (int) window_get(pg_canvas, CANVAS_HEIGHT);
       pw_writebackground(pw, 0, 0, canvas_width, canvas_height,
                               PIX_SRC);

                     /* traverse channel list and draw channels */
       chan_old = chan_head;
       while(chan_old != NULL)
       {
              ptr_symbol_1 = chan_old->sym_name;
              seek_xy(ptr_symbol_1);
              s1x = sym_x;
              s1y = sym_y;
              ptr_symbol_2 = chan_old->sym_name_other_end;
              seek_xy(ptr_symbol_2);
              s2x = sym_x;
              s2y = sym_y;
              pw_vector(pw, s1x, s1y, s2x, s2y, PIX_SRC, 1);
              chan_old = chan_old->next;
       }
                     /* keep an updated count of symbols */
       count_of_symbols = 0;
                     /* draw symbols */
       relative_ptr = head;
       while (relative_ptr != NULL)
       {
              count_of_symbols++;
              cur_loc_x = relative_ptr->symbol_x;
              cur_loc_y = relative_ptr->symbol_y;
              pw_rop(pw, cur_loc_x, cur_loc_y, 64, 64,
                       PIX_SRC, icons[relative_ptr->symbol_type], 0,
                              0);

                     /* invert the symbol, if required */
              if(relative_ptr->inverted == 1)
              {
                            /* record it so can uninvert later */
                     ptr_to_previous = relative_ptr;
                     save_inverted_x = relative_ptr->symbol_x;
                     save_inverted_y = relative_ptr->symbol_y;
```

```
                save_inverted_type = relative_ptr->symbol_type;
                invert_symbol(save_inverted_x, save_inverted_y,
                        save_inverted_type);
                a_symbol_inverted = 1;
            }
            input_symbol_name = relative_ptr->symbol_name;
            center_print_text();
            relative_ptr = relative_ptr->next;
        }
}
```

```
/*****************************************************************/
/* Routine to remove all files from directory. deallocate list, clear
*/
void
delete_graphic_proc()
{
        invoke_delete_popup();
        delete_graphic = (int) window_loop(delete_graphic_frame);

        if (delete_graphic == 1)
        {
                        /* clean up and return to directory */
                delete_pg_files();
                return_to_directory_proc();
        }

        window_set(delete_graphic_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(delete_graphic_frame);
}

/*****************************************************************/
/* removes the newest version of the graphic files */
void
delete_pg_files()
{
                        /* remove .pg files */
        clear_command_line();
        strcat(command_line, "rm ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, "_* 2> /dev/null");
        system(command_line);
                        /* remove .gr file */
        clear_command_line();
        strcat(command_line, "rm ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr 2> /dev/null");
        system(command_line);
}
```

```
/*****************************************************************/
/* creates the popup panel asking the user if they are sure they want
** to delete graphic */
void
invoke_delete_popup()
{
        delete_graphic_frame = window_create(NULL, FRAME,
                        FRAME_SHOW_LABEL,       FALSE,
                        0);

        delete_panel = window_create(delete_graphic_frame, PANEL,
                        0);

        panel_create_item(delete_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING,
                        "Are you sure you want to delete this graphic?",
                0);

        panel_create_item(delete_panel, PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        delete_panel, "Yes", 5, 0),
                PANEL_ITEM_X,   140,
                PANEL_ITEM_Y,   35,
                PANEL_CLIENT_DATA,      1,
                PANEL_NOTIFY_PROC,      response_proc,
                0);

        panel_create_item(delete_panel, PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(
                        delete_panel, "No - Cancel", 5, 0),
                PANEL_ITEM_X,   230,
                PANEL_ITEM_Y,   35,
                PANEL_CLIENT_DATA,      0,
                PANEL_NOTIFY_PROC,      response_proc,
                0);

        window_fit(delete_panel);
        window_fit(delete_graphic_frame);

        center_popup(delete_graphic_frame);

        window_set(delete_graphic_frame, WIN_X, left,
                                WIN_Y,          top,
                                0);
}

/*****************************************************************/
/* returns from a popup while returning whether the user answered yes
** or no */
void
response_proc(item)
Panel_item      item;
{
        window_return(panel_get(item, PANEL_CLIENT_DATA));
}
```

```
/*****************************************************************/
/* Routine to concatenate all specs for a graphic in a file */
void
store_specs_to_file(store_file)
char    *store_file;
{

int     j, k;
FILE    *store_fp, *fopen();
char    temp_file[MAX_FILENAME];

        count_symbols();
        if(count_of_symbols != 0)
        {
                        /* open store_file & place "system
                                specification" */
                if((store_fp = fopen(store_file, "w")) == NULL)
                {
                        printf("\nCannot open temporary file %s",
                                        store_file);
                        exit(1);
                }
                        /* place opening brace */
                fprintf(store_fp, "(");

                old = head;
                while(old != NULL)
                {
                        fprintf(store_fp, "%s-cycle[",
                                        old->symbol_name);
                        if(old->symbol_type == 1)
                                fprintf(store_fp,
                                        "%s-empty-buffer]",
                                                old->symbol_name);
                        else
                                if((old->symbol_type == 3) ||
                                                (old->symbol_type == 6))
                                        fprintf(store_fp, "Null]");
                                else
                                        fprintf(store_fp,
                                                "initial-%s-state]",
                                                 old->symbol_name);

                        if(old->next != NULL)
                                fprintf(store_fp, ",\n");
                        else
                                fprintf(store_fp, "\n");

                        old = old->next;
                }  /* end while */

                fprintf(store_fp, ");\n\n");
                fclose(store_fp);

                build_file_list();
```

```
                temp_fp = fopen("temp_file.pg", "r");

                for(j = 0; j < count_of_symbols; j++)
                {
                        clear_command_line();
                        strcat(command_line, "cat ");

                                /* clear the temp_file array */
                        for(k = 0; k < MAX_FILENAME; k++)
                                temp_file[k] = '\0';

                        fscanf(temp_fp, "%s\n", temp_file);
                        strcat(command_line, temp_file);

                        strcat(command_line, " >> ");
                        strcat(command_line, store_file);
                        system(command_line);
                }
                                /* close pipe */
                fclose(temp_fp);
        }  /* end if symbols */
}
```

```
/*****************************************************************/
/* creates a list of all the spec filenames that are in the graphic */
void
build_file_list()
{
                /* build command to put spec names into temp file */
        clear_command_line();
        strcat(command_line, "ls -c ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, "_*.pg > temp_file.pg");
        system(command_line);
}

/*****************************************************************/
/* clears the command line array */
void
clear_command_line()
{
int     m;

        for(m = 0; m < MAX_COMMAND; m++)
                command_line[m] = '\0';
}
```

```
/*****************************************************************/
/* Allow user to create standard rasterfile for imagen printer */
void
print_screen_proc()
{
char    *screen_file;

        entry_made = 0;
        init_screen_popup();

        while(entry_made == 0)
        {
                entry_made = 1;

                        /* force user to provide a filename or cancel */
                window_loop(screen_file_frame);

                if(cancel == 0)
                {
                                /* obtain name that user input */
                        screen_file = panel_get_value(screendump_item);
                        if(strlen(screen_file) != 0)
                        {
                                /* make and fill the screendump file */
                                clear_command_line();
                                strcat(command_line, "screendump > ");
                                strcat(command_line, screen_file);

                                        /* run the command */
                                system(command_line);
                        }
                        else
                                entry_made = 0;
                }
                else
                        cancel = 0;

        }  /* end while no entry */

        remove_screen_popup();
}

/* separate routine needed to eliminate the print screen popup because
** it must be destroyed before the screendump is done */
void
remove_screen_popup()
{
                /* get rid of popup */
        window_set(screen_file_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(screen_file_frame);
}
```

```
/*****************************************************************/
/* create the popup asking for user to input name of file for
** screendump*/
void
init_screen_popup()
{
        screen_file_frame = window_create(NULL, FRAME,
                                FRAME_SHOW_LABEL,       FALSE,
                                0);


        screen_panel = window_create(screen_file_frame, PANEL,
                                0);

        panel_create_item(screen_panel, PANEL_MESSAGE,
                PANEL_LABEL_STRING, "Enter Filename for Screendump: ",
                0);

        screendump_item = panel_create_item(screen_panel, PANEL_TEXT,
                PANEL_VALUE_DISPLAY_LENGTH,     MAX_INPUT_FILENAME,
                PANEL_VALUE_STORED_LENGTH,      MAX_INPUT_FILENAME,
                0);

        panel_create_item(screen_panel, PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(screen_panel,
                        "Done", 5, 0),
                PANEL_ITEM_X,   240,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      return_to_graphic_from_popup,
                0);

        panel_create_item(screen_panel, PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image(screen_panel,
                        "Cancel", 5, 0),
                PANEL_ITEM_X,   140,
                PANEL_ITEM_Y,   35,
                PANEL_NOTIFY_PROC,      cancel_proc,
                0);

        window_fit(screen_panel);
        window_fit(screen_file_frame);

        center_popup(screen_file_frame);

        window_set(screen_file_frame, WIN_X,    left,
                                WIN_Y,          top,
                                0);

}
```

```
/*****************************************************************/
/* make a duplicate copy of all files associated with a graphic */
void
archive_graphic()
{
int     i, j;
char    cur_file_name[MAX_FILENAME];
FILE    *temp_fileptr, *fopen();

                        /* count the number of files/symbols in the
                                graphic */
        count_symbols();

                        /* only archive symbols if some out there */
        if(count_of_symbols != 0)
        {
                clear_command_line();
                strcat(command_line, "ls ");
                strcat(command_line, graphic_name_save);
                strcat(command_line, "_*.pg > pg_graphics.pg 2>
                                /dev/null");
                system(command_line);

                        /* use stored filenames to make archive copy */
                temp_fileptr = fopen("pg_graphics.pg", "r");
                for(i = 0; i < count_of_symbols; i++)
                {
                                /* clear the file array */
                        for(j = 0; j < MAX_FILENAME; j++)
                                cur_file_name[j] = '\0';
                        fscanf(temp_fileptr, "%s\n", cur_file_name);
                        clear_command_line();
                        strcat(command_line, "cp ");
                        strcat(command_line, cur_file_name);
                        strcat(command_line, " saved_");
                        strcat(command_line, cur_file_name);
                        strcat(command_line, " 2> /dev/null");
                        system(command_line);
                }

                fclose(temp_fileptr);
                system("rm pg_graphics.pg");

        }  /* end if symbols in the graphic */

                        /* save the .gr file */
        clear_command_line();
        strcat(command_line, "cp ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr");
        strcat(command_line, " saved_");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr");
        system(command_line);
}
```

```
/*****************************************************************/
/* overlays the original version of the graphic back onto newest
** version */
void
unarchive_graphic()
{
int     i, j, skip;
char    graphic_file[MAX_FILENAME], saved_file[MAX_FILENAME];
FILE    *saved_fp, *fopen();

                        /* restore the original .gr file */
        clear_command_line();
        strcat(command_line, "mv saved_");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr ");
        system(command_line);

                        /* clear graphic_file */
        for(j = 0; j < MAX_FILENAME; j++)
                graphic_file[j] = '\0';
        strcat(graphic_file, graphic_name_save);
        strcat(graphic_file, ".gr");
        if ((saved_fp = fopen(graphic_file, "r")) == NULL)
        {
            printf("\n Cannot retrieve archived graphic:%s\n",
                                graphic_file);
            exit(1);
        }

                        /* go through the file */
        fscanf(saved_fp, "%d\n", &count_of_symbols);
        for(i = 0; i < count_of_symbols; i++)
        {
                        /* clear temp_file */
                for(j = 0; j < MAX_FILENAME; j++)
                        saved_file[j] = '\0';
                        /* read in file name */
                fscanf(saved_fp, "%s %d %d %d %d\n", saved_file, &skip,
                        &skip, &skip, &skip);

                        /* copy the saved files back to original files
                        */
                clear_command_line();
                strcat(command_line, "mv saved_");
                strcat(command_line, graphic_name_save);
                strcat(command_line, "_");
                strcat(command_line, saved_file);
                strcat(command_line, ".pg ");
                strcat(command_line, graphic_name_save);
                strcat(command_line, "_");
                strcat(command_line, saved_file);
                strcat(command_line, ".pg");
                system(command_line);
```

```
        }  /* end for all files */
}
```

```
/*************************************************************************/
/* remove the archived files of a graphic */
void
remove_archive()
{
                /* remove .pg files */
        clear_command_line();
        strcat(command_line, "rm saved_");
        strcat(command_line, graphic_name_save);
        strcat(command_line, "_* 2> /dev/null");
        system(command_line);

                /* remove the .gr file */
        clear_command_line();
        strcat(command_line, "rm saved_");
        strcat(command_line, graphic_name_save);
        strcat(command_line, ".gr 2> /dev/null");
        system(command_line);
}

/*************************************************************************/
/* counts the symbols on the graphic */
void
count_symbols()
{
        count_of_symbols = 0;
        old = head;
        while (old != NULL)
        {
                count_of_symbols++;
                old = old->next;
        }

}
```

```
/*****************************************************************/
/* manages all the restriction, clean-up and rearranging of the symbol
** list when a symbol is deleted */
void
delete_symbol_proc()
{
int     allow_delete1, allow_delete2, keep_looping, found_inv;
char    *ptr_cha1, *ptr_cha2, *cur_sym_inverted;

        if(a_symbol_inverted == 1)
        {
                        /* do not allow delete if there are channels to
                                sym */
                allow_delete1 = 1;
                allow_delete2 = 1;
                keep_looping = 1;
                chan_old =  chan_head;
                while((chan_old != NULL) && (keep_looping == 1))
                {
                        ptr_cha1 = chan_old->sym_name;
                        ptr_cha2 = chan_old->sym_name_other_end;
                        cur_sym_inverted = ptr_to_previous->symbol_name;
                        allow_delete1 = strcmp(ptr_cha1,
                                                cur_sym_inverted);
                        allow_delete2 = strcmp(ptr_cha2,
                                                cur_sym_inverted);

                        if((allow_delete1 == 0) || (allow_delete2 ==
                           0))
                                keep_looping = 0;
                        chan_old = chan_old->next;
                }
                if(keep_looping == 1)
                {
                if((ptr_to_previous->next == NULL) &&
                        (ptr_to_previous->prev == NULL))
                                 /* only one in list */
                        {
                                head = NULL;
                                tail = NULL;
                                remove_element(ptr_to_previous);
                        }
                                /* 1st in list */
                        else if(ptr_to_previous->prev == NULL)
                        {
                                head = ptr_to_previous->next;
                                head->prev = NULL;
                                remove_element(ptr_to_previous);
                        }
                        else
                        {
                                        /* in middle or end of list */
                                ptr_before = ptr_to_previous->prev;
                                ptr_before->next =
                                        ptr_to_previous->next;
```

```
                                        /* handle middle of list case */
                                    if(ptr_to_previous->next != NULL)
                                    {
                                        ptr_after = ptr_to_previous->next;
                                        ptr_after->prev = ptr_before;
                                    }
                                    else
                                        /* handle case of last in list */
                                            tail = ptr_before;
                                    remove_element(ptr_to_previous);
                            }  /* end else */

                            redraw_graphic();

                    }  /* end if not empty list */
            }  /* end if a symbol is inverted */
}
```

```
/*****************************************************************/
/* deletes everyting associated with a symbol */
void
remove_element(sy_node)
struct symbol_rec       *sy_node;
{
Pixwin *pw;
char    *s_name;
int     extra_char, pixels, num_char;

                        /* remove spec file for the node */
        clear_command_line();
        strcat(command_line, "rm ");
        strcat(command_line, graphic_name_save);
        strcat(command_line, "_");
        strcat(command_line, sy_node->symbol_name);
        strcat(command_line, ".pg");
        system(command_line);

        a_symbol_inverted = 0;
        free(sy_node);
}
```

```
#include        "./pg.h"

                        /* function declarations */
void    close_view();
void    user_done_proc();
void    advance_to_show_endpoint();
void    go_to_show_endpoint();
void    remove_subwin();
void    toggle_proc();
void    reset_symbol();
void    save_no_quit();
void    scan_channels();
void    add_endpoint();
extern void     count_symbols();
extern void     add_chan_to_list();
void    delete_end_list();
extern void     redraw_graphic();
void    convert_end_to_chan();
get_window_num();
void    find_primary();
void    find_secondary();
extern void     response_proc();
extern void     build_file_list();
extern void     clear_command_line();

                        /* graphic name */
extern char    graphic_name_save[];

                        /* declarations specific to showing other end */
extern void     end_proc();
/*****************************************************************/
/* variables used in establishing a channel */
int     exch1, exch2;
char    *ps1, *ps2, *pcn;
```

```
/*****************************************************************/
/* creates the specification edit environment consisting of a panel
** with buttons and a textsubwindow  */
void
go_to_subwin(subwin_num, spec_file)
int     subwin_num;
char    spec_file[];
{
char            *spec_filename, *buf;
unsigned        flags, buf_len;
Textsw_index    first, last_plus_one;
int             bytes_found;

        pg_subwin_frame[subwin_num] = window_create(pg_base_frame,
                FRAME,
                FRAME_DONE_PROC,        user_done_proc,
                WIN_CLIENT_DATA,        subwin_num,
                0);

        select_panel[subwin_num] = window_create(
                        pg_subwin_frame[subwin_num], PANEL,
                WIN_HEIGHT,     30,
                PANEL_CLIENT_DATA,      subwin_num,
                0);

        panel_create_item(select_panel[subwin_num], PANEL_BUTTON,
                PANEL_LABEL_IMAGE,
                panel_button_image(select_panel[subwin_num], "Show Other
                                        End", 0, 0),
                PANEL_ITEM_X,   5,
                PANEL_NOTIFY_PROC, advance_to_show_endpoint,
                0);

        panel_create_item(select_panel[subwin_num], PANEL_BUTTON,
                PANEL_LABEL_IMAGE, panel_button_image(
                        select_panel[subwin_num], "Save/Scan-No Quit",
                                0, 0),
                PANEL_ITEM_X,   185,
                PANEL_NOTIFY_PROC, save_no_quit,
                0);

        panel_create_item(select_panel[subwin_num], PANEL_BUTTON,
                PANEL_LABEL_IMAGE, panel_button_image(
                        select_panel[subwin_num], "Save/Quit-Close
                                        View", 0, 0),
                PANEL_ITEM_X,   395,
                PANEL_NOTIFY_PROC, close_view,
                0);

        panel_create_item(select_panel[subwin_num], PANEL_TOGGLE,
                PANEL_CHOICE_STRINGS,   "Add Channel Toggle",
                                        0,
                PANEL_TOGGLE_VALUE,     0, FALSE,
                PANEL_ITEM_X,   645,
                PANEL_ITEM_Y,   5,
```

```
            PANEL_NOTIFY_PROC,      toggle_proc,
            0);

    spec_filename = spec_file;
    textsw[subwin_num] = window_create(
                    pg_subwin_frame[subwin_num], TEXTSW,
            WIN_BELOW,      select_panel[subwin_num],
            WIN_X,          0,
            WIN_WIDTH,      900,
            WIN_HEIGHT,     350,
            TEXTSW_BLINK_CARET,     TRUE,
            TEXTSW_DISABLE_CD,      TRUE,
            TEXTSW_DISABLE_LOAD,    TRUE,
            TEXTSW_STORE_CHANGES_FILE,      FALSE,
            TEXTSW_STORE_SELF_IS_SAVE,      TRUE,
            TEXTSW_FILE,    spec_filename,
            0);

    t_menu = window_get(textsw[subwin_num], TEXTSW_MENU);
    m_item = menu_get(t_menu, MENU_NTH_ITEM, 1);
    menu_set(m_item, MENU_INACTIVE, TRUE, 0);
    m_item = menu_get(t_menu, MENU_NTH_ITEM, 10);
    menu_set(m_item, MENU_INACTIVE, TRUE, 0);

    window_fit(pg_subwin_frame[subwin_num]);
    window_set(pg_subwin_frame[subwin_num], WIN_SHOW, TRUE, 0);

                    /* advance to the other end of the channel if
                    ** the request is to show other end */
    if(showing_other_end == 1)
            end_proc();
}
```

```
/*****************************************************************/
/* limits the number of open endpoint panels */
void
advance_to_show_endpoint(item)
Panel_item      item;
{
        if(chan_end_displayed == 0)
                go_to_show_endpoint(item);
}

/*****************************************************************/
/*  routine to destroy text window if user chooses "done" off frame
** menu ***/
void
user_done_proc(frame_done)
Frame   frame_done;
{
        remove_subwin(frame_done);
}

/*****************************************************************/
/* identifies which frame the request to close view was called from */
void
close_view(item)
Panel_item      item;
{
Frame   frame;
Panel   own_panel;
                        /* determine what panel owns panel item */
        own_panel = panel_get(item, PANEL_PARENT_PANEL);
                        /* determine what frame owns panel */
        frame = window_get(own_panel, WIN_OWNER);
        remove_subwin(frame);
}
```

```
/*****************************************************************/
/* determines the number of the textsubwindow that requested it */
get_window_num(ref_item)
Panel_item      ref_item;
{
Panel   own_panel;
int     desired_num;

                        /* determine what panel owns panel item */
        own_panel = panel_get(ref_item, PANEL_PARENT_PANEL);
                        /* update list of open subwins */
        desired_num = (int) panel_get(own_panel, PANEL_CLIENT_DATA);
        return(desired_num);
}

/*****************************************************************/
/* routines to manage the closing out of a specification window */
void
remove_subwin(frame_to_kill)
Frame   frame_to_kill;
{
int     ret_subwin_num;

                        /* save edits for the user */
        ret_subwin_num = (int) window_get(frame_to_kill,
                                WIN_CLIENT_DATA);
        textsw_save(textsw[ret_subwin_num], 100, 100);

        reset_symbol(ret_subwin_num);

        window_set(pg_subwin_frame[ret_subwin_num], FRAME_NO_CONFIRM,
                                TRUE, 0);
        window_destroy(pg_subwin_frame[ret_subwin_num]);

}
```

```
/*****************************************************************/
/* reset the factors for the symbol's spec window that has been closed
*/
void
reset_symbol(subwin_in)
int     subwin_in;
{
int     u, found;

                        /* reset on list of symbols */
        found = 1;
        rel_ptr = head;
        while((rel_ptr != NULL) && (found != 0))
        {
                if(rel_ptr->open == subwin_in)
                {
                        rel_ptr->open = -1;
                        found = 0;
                }
                rel_ptr = rel_ptr->next;
        }

                        /* resets on array of open windows */

                        /* clear name */
        for(u = 0; u < MAX_SYMBOL_NAME; u++)
                sub_list[subwin_in].name[u] = '\0';
                        /* clear other variables */
        sub_list[subwin_in].toggle_value = 0;
        sub_list[subwin_in].in_use = -1;
}
```

```
/*****************************************************************/
/* sets an array element which keeps track of the textsw that are
** currently eligible for the adding of a channel */
void
toggle_proc(item)
Panel_item      item;
{
int     tog_val,
        sub_num;

        tog_val = (int) panel_get(item, PANEL_TOGGLE_VALUE, 0);
        sub_num = get_window_num(item);
        sub_list[sub_num].toggle_value = tog_val;
}
```

```
/*****************************************************************/
/* searches for channels in all spec files for a graphic and builds an
** endpoint list */
void
scan_channels()
{
int     p, q, r, s, t, typex, prefix, num_matches, found_chn;
char    read_file[MAX_FILENAME], chan_line[MAX_LINE], *ptr_read_file,
        *ptr_ch_name, *only_symbol, *store_chan;
FILE    *chan_fp, *count_pipe, *fopen();

                        /* eliminate the channel list out there */
        delete_chan_list();
        delete_end_list();
        chan_head = NULL;
        end_head = NULL;
                /* list of current graphics files in pg_temp.pg */
        build_file_list();

                        /* count the number of files */
        count_symbols();

        temp_fp = fopen("temp_file.pg", "r");

                /* have operating system scan files for channels */
        for(p = 0; p < count_of_symbols; p++)
        {
                        /* clear array */
                for(q = 0; q < MAX_FILENAME; q++)
                        read_file[q] = '\0';
                        /* place filename into array */
                fscanf(temp_fp, "%s\n", read_file);
                        /* scan the file for channels */
                clear_command_line();
                strcat(command_line, "grep x- ");
                strcat(command_line, read_file);
                strcat(command_line, " > pg_chans.pg 2> /dev/null");
                system(command_line);
                clear_command_line();
                strcat(command_line, "grep x[rm]- ");
                strcat(command_line, read_file);
                strcat(command_line, " >> pg_chans.pg 2> /dev/null");
                system(command_line);

                        /* determine how many channels were found */
                count_pipe = popen("wc -l pg_chans.pg", "r");
                fscanf(count_pipe, "%d", &num_matches);
                pclose(count_pipe);

                if(num_matches > 0)
                {
                                /* open the file and read the chans */
                        chan_fp = fopen("pg_chans.pg", "r");

                        for(q = 0; q < num_matches; q++)
```

```
{
        for(r = 0; r < MAX_LINE; r++)
                chan_line[r] = '\0';

        found_chn = 0;
        while(found_chn == 0)
        {
                fscanf(chan_fp, "%s",
                                chan_line);
                /* determine if its string
                        with x in it */
                ptr_ch_name = chan_line;
                found_chn = index(ptr_ch_name,
                                'x');
        }
                /* move to next line */
        fscanf(chan_fp, "\n");

                /* get a ptr to just the symbol
                        name */
        r = 0;
        while(read_file[r] != '.')
                r++;
                /* clear to end of line */
        for(s = 0; s < 3; s++)
        {
                read_file[r] = '\0';
                r++;
        }
        prefix = strlen(graphic_name_save) + 1;
        ptr_read_file = read_file;
                /* this is the clean symbol name
                */
        only_symbol = (ptr_read_file + prefix);

                /* identify the type of exchange
                */
        r = 0;
        while(chan_line[r] != 'x')
                r++;
        r++;
        switch(chan_line[r])
        {
                case '-' : typex = 0;
                           break;
                case 'r' : typex = 1;
                           r++;
                           break;
                case 'm' : typex = 2;
                           r++;
                           break;
        }
        r++;

                /* clear to end of channel name
```

```
                                        line */
                                s = 0;
                                t = r + s;
                                while((chan_line[t] != '[') &&
                                        (chan_line[t] != ' ') &&
                                           (chan_line[t] != '\n') &&
                                           (chan_line[t] != '\0') &&
                                           (s < MAX_CHANNEL_NAME) )
                                {
                                        s++;
                                        t = r + s;
                                }
                                chan_line[t] = '\0';

                                ptr_ch_name = chan_line;
                                        /* this is the clean channel
                                                name */
                                store_chan = (ptr_ch_name + r);

                                add_endpoint(only_symbol, typex,
                                                store_chan);

                        } /* end for all matching lines in template */

                        fclose(chan_fp);
                        system("rm pg_chans.pg");
                } /* end if */
        } /* end for all files in graphic */

        fclose(temp_fp);
        system("rm temp_file.pg");

                /* make new list of channels from updated list of
                        endpoints */
        convert_end_to_chan();
}
```

```
/*****************************************************************/
/* makes a linked list entry for each channel endpoint */
void
add_endpoint(sy, ty, ch)
char    *sy, *ch;
int     ty;
{
int s, found_end, check_sy, check_ch;

                        /* make sure the endpoint doesn't exist already
                        */
        found_end = 0;
        end_old = end_head;
        while((end_old != NULL) && (found_end == 0))
        {
                check_sy = strcmp(end_old->sy_name, sy);
                check_ch = strcmp(end_old->ch_name, ch);
                if((check_sy == 0) && (check_ch == 0) && (end_old->inter
                                == ty))
                        found_end = 1;
                else
                        end_old = end_old->next;
        }

                        /* continue only if the endpt. not in list */
        if(found_end == 0)
        {
                end_new = (struct end_rec *) malloc (sizeof (struct
                                end_rec));
                strcpy(end_new->sy_name, sy);
                end_new->inter = ty;
                strcpy(end_new->ch_name, ch);
                end_new->u_num = 0;
                end_new->next = NULL;

                        /* first in list */
                if (end_head == NULL)
                {
                        end_head = end_new ;
                        end_tail = end_new ;
                }
                else                    /* add to end of list */
                {
                        end_tail->next = end_new ;
                        end_tail = end_new ;
                }
        }  /* end if not already in list */
}
```

```
/*************************************************************************/
/* deletes the entire channel endpoint list */
void
delete_end_list()
{
        end_old = end_head;
        while(end_old != NULL)
        {
                end_head = end_old->next;
                free(end_old);
                end_old = end_head;
        }
}

/*************************************************************************/
/* function to save all the open textsw's, scan them for
** channels, construct channel list and redraw graphic */
void
save_no_quit()
{
int     v, window_num;

        for(v = 0; v < MAX_SUBWINS; v++)
                if(sub_list[v].in_use != -1)
                                /* store the specification to file if
                                        open */
                        textsw_save(textsw[sub_list[v].in_use], 100,
                                        100);

                        /* scan for channels */
        scan_channels();
}
```

```
/*****************************************************************/
/* traverses the channel endpoint list to create list of channels (two
** endpts)*/
void
convert_end_to_chan()
{
int     c, w, x, y, un_count, check_c, count_of_ends, first, tyx, tyr,
tym,
        char_to_find;

        end1 = end_head;
        un_count = 1;
                        /* identify and record unique channel names */
        while(end1 != NULL)
        {
                if(end1->u_num == 0)
                {
                        end2 = end1->next;
                        check_c = -1;
                        while(end2 != NULL)
                        {
                                check_c = strcmp(end1->ch_name,
                                                end2->ch_name);
                                if(check_c == 0)
                                {
                                        end2->u_num = un_count;
                                        end1->u_num = un_count;
                                }
                                end2 = end2->next;
                        } /* end while */

                        if(end1->u_num != 0)
                                un_count++;
                }  /* end if */
                end1 = end1->next;

        }  /* end while */

                        /* no need to build chan list if not enough
                                endpts */
        if(un_count != 1)
        {
                for(w = 1; w < un_count; w++)
                {
                                /* count the no. of ends on each chan */
                        count_of_ends = 0;
                        tyx = 0;
                        tyr = 0;
                        tym = 0;
                        end1 = end_head;
                        while(end1 != NULL)
                        {
                                if(end1->u_num == w)
                                        count_of_ends++;
```

```
                        /* count type of exchanges */
                        switch(end1->inter)
                        {
                                case 0 : tyx++;
                                         break;
                                case 1 : tyr++;
                                         break;
                                case 2 : tym++;
                                         break;
                        }
                end1 = end1->next;
        }

        if( (count_of_ends == 2) && ((tyx == 2) ||
                ((tyx == 1) && (tyr == 1)) ||
                ((tym == 1) && (tyr == 1)) ) )
        {
                        /*if a correct case make channel
                                entry*/
                first = 0;
                end1 = end_head;
                while((end1 != NULL) && (first != 2))
                {
                        if(end1->u_num == w)
                                if(first == 0)
                                {
                                  ps1 = end1->sy_name;
                                  pcn = end1->ch_name;
                                  exch1 = end1->inter;
                                  first = 1;
                                }
                                else
                                {
                                  ps2 = end1->sy_name;
                                  exch2 = end1->inter;
                                  first = 2;
                                }
                        end1 = end1->next;
                }  /* end while */
                add_chan_to_list(ps1, ps2, exch1, exch2,
                                        pcn);

        }  /* end if */
        else
        {
                        /* 1 x and >1 r */
                if((tyx == 1) && (tyr > 1) && (tym ==
                        0))
                {
                    char_to_find = 0;      /* x */
                    find_primary(w, char_to_find,
                                        end_head);
                    char_to_find = 1;      /* xr */
                    find_secondary(w, char_to_find,
                                        end_head);
```

```
}

        /* 1 x and >1 m */
if((tyx == 1) && (tyr == 0) && (tym >
        1))
{
    char_to_find = 0;       /* x */
    find_primary(w, char_to_find,
                        end_head);
    char_to_find = 2;       /* xm */
    find_secondary(w, char_to_find,
                        end_head);
}

        /* all x's */
if((tyx > 2) && (tyr == 0) && (tym ==
        0))
{
    starter = end_head;
    for(c = 0; c < (tyx - 1); c++)
    {
        /* connect each x to all the
                other x's */
        char_to_find = 0;      /* x */
        find_primary(w, char_to_find,
                        starter);
        find_secondary(w, char_to_find,
                        primary->next);
        starter = primary->next;
    }
}
        /* no x's, but >1r and >1m */
if((tyx == 0) && (tyr > 1) && (tym > 1))
{
   starter = end_head;
   for(c = 0; c < tyr; c++)
   {
        /* connect each xr to all the
                other xm's */
     char_to_find = 1;     /* xr */
     find_primary(w, char_to_find,
                        starter);
     char_to_find = 2;      /* xm *
     find_secondary(w, char_to_find,
                        end_head);
     starter = primary->next;
    }
}  /* end if */

        /* 1 r and >1 m */
if((tyx == 0) && (tyr == 1) && (tym >
        1))
{
    char_to_find = 1;      /* x */
    find_primary(w, char_to_find,
```

```
                                        end_head);
                                char_to_find = 2;     /* xr */
                                find_secondary(w, char_to_find,
                                        end_head);
                            }

                                /* >1 r and 1 m */
                            if((tyx == 0) && (tyr > 1) && (tym ==
                                1))
                            {
                                char_to_find = 2;     /* x */
                                find_primary(w, char_to_find,
                                                end_head);
                                char_to_find = 1;     /* xr */
                                find_secondary(w, char_to_find,
                                                end_head);
                            }

                        } /*end else */
                } /* end for */

                redraw_graphic();
        } /* end if */
}
```

```
/*********************************************************************/
/*  routine to start at end_s and find next occurrence of s_num and
** find_char interaction type in endpoint list */
void
find_primary(s_num, find_char, end_s)
int     s_num, find_char;
struct end_rec *end_s;
{
int     found_prim;

        end1 = end_s;
        found_prim = 0;
        while((end1 != NULL) && (found_prim == 0))
        {
                if((end1->inter == find_char) && (end1->u_num == s_num)
                        && (found_prim == 0))
                {
                        found_prim = 1;
                        primary = end1;
                }
                end1 = end1->next;
        }
}
```

```
/*************************************************************************/
/*  routine to start at end_f and find next occurrence of f_num and
** f_char interaction type in endpoint list */
void
find_secondary(f_num, f_char, end_f)
int     f_num, f_char;
struct end_rec *end_f;
{

        end1 = end_f;
        while(end1 != NULL)
        {
                if((end1->inter == f_char) && (end1->u_num ==f_num))
                {
                        ps1 = primary->sy_name;
                        ps2 = end1->sy_name;
                        pcn = primary->ch_name;
                        exch1 = primary->inter;
                        exch2 = end1->inter;
                        add_chan_to_list(ps1, ps2, exch1, exch2, pcn);
                }
                end1 = end1->next;
        }
}
```

```
#include "./pg.h"

/*****************************************************************/
/* function declarations */
void    init_add_panel();
void    post_channel();
void    seek_xy();
void    add_chan_to_list();
void    delete_chan_list();
build_insert_line();
find_index_to_subwin();
extern void     redraw_graphic();
extern void     add_endpoint();

                /* objects involved in the Add Channel feature */
Panel   add_chan_panel;
Frame   add_chan_frame;
Canvas  pg_canvas;
Panel_item      interact_item_1, interact_item_2, name_item;

                        /* misc variables */
int     sym_x, sym_y,
        cancel;

                        /* pointers to the symbol names */
char    *ptr_sym_1, *ptr_sym_2,
                        /* pointer to channel name */
        *input_channel_name,
                        /* channel name to be inserted into spec */
        *ptr_insert_line, insert_line[MAX_CHANNEL_NAME];

                /* vars used to pass channel stuff between files */
int     int_1, int_2, saved_yet;
char    sym_end_1, sym_end_2, chan_text;
```

```
/*****************************************************************/
/* manages the popup panel for the add channel feature including
** interaction type choice switches, channel name input and management
** buttons */
void
go_to_add_channel(in_sym_1, in_sym_2)
char    *in_sym_1, *in_sym_2;
{
int     entry_m;

        ptr_sym_1 = in_sym_1;
        ptr_sym_2 = in_sym_2;

                        /* bring panel up for add of channel */
        init_add_panel();

        entry_m = 0;
        while(entry_m == 0)
        {
                entry_m = 1;
                        /* limit user to that window */
                window_loop(add_chan_frame);

                input_channel_name = panel_get_value(name_item);
                if(strlen(input_channel_name) != 0)
                        post_channel(input_channel_name);
                else
                        entry_m = 0;

        }  /* end while no channel name entry made */
                        /* On the return, get rid of add_chan_frame */
        window_set(add_chan_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(add_chan_frame);
}
```

```
/*************************************************************************/
/* creates the popup panel for adding a channel */
void
init_add_panel()
{
int     ch1, ch2, count_match;
char    channel_name[MAX_CHANNEL_NAME], *ptr_chan_name, *ptr_int,
        ptr_i[MAX_INT];


        add_chan_frame = window_create(NULL, FRAME,
                        FRAME_LABEL,    "          Add Channel
                                Interaction",
                        WIN_X,          0,
                        WIN_Y,          0,
                        0);

        add_chan_panel = window_create(add_chan_frame, PANEL,
                        0);

        panel_create_item(add_chan_panel, PANEL_MESSAGE,
                        PANEL_LABEL_STRING,     ptr_sym_1,
                        PANEL_ITEM_X,   25,
                        PANEL_ITEM_Y,   5,
                        0);

        interact_item_1 = panel_create_item(add_chan_panel,
                        PANEL_CHOICE,
                        PANEL_CHOICE_STRINGS,   "x", "xr", "xm", 0,
                        PANEL_ITEM_X,   85,
                        PANEL_ITEM_Y,   35,
                        PANEL_LAYOUT,   PANEL_VERTICAL,
                        0);

        panel_create_item(add_chan_panel, PANEL_MESSAGE,
                        PANEL_LABEL_STRING,     ptr_sym_2,
                        PANEL_ITEM_X,   200,
                        PANEL_ITEM_Y,   5,
                        0);

        interact_item_2 = panel_create_item(add_chan_panel,
                        PANEL_CHOICE,
                        PANEL_CHOICE_STRINGS,   "x", "xr", "xm", 0.
                        PANEL_ITEM_X,   250,
                        PANEL_ITEM_Y,   35,
                        PANEL_LAYOUT,   PANEL_VERTICAL,
                        0);

        panel_create_item(add_chan_panel, PANEL_MESSAGE,
                        PANEL_LABEL_STRING,     "Channel Name:",
                        PANEL_ITEM_X,   65,
                        PANEL_ITEM_Y,   110,
                        0);

        name_item = panel_create_item(add_chan_panel, PANEL_TEXT,
```

```
                PANEL_ITEM_X,   220,
                PANEL_ITEM_Y,   110,
                PANEL_VALUE_DISPLAY_LENGTH, MAX_CHANNEL_NAME,
                PANEL_VALUE_STORED_LENGTH,  MAX_CHANNEL_NAME,
                0);

                /* scan the list of channels for occurrences of
                ** an existing channel between these two symbols
                */
count_match = 0;
chan_old = chan_head;
while(chan_old != NULL)
{
        ch1 = strcmp(chan_old->sym_name, ptr_sym_1);
        if(ch1 == 0)
        {
                        /* compare with other end */
                ch2 = strcmp(chan_old->sym_name_other_end,
                                ptr_sym_2);
                if(ch2 == 0)
                        count_match++;
        }
        else
                        /* no match, check other end */
        {
                ch1 = strcmp(chan_old->sym_name_other_end,
                                ptr_sym_1);
                if(ch1 == 0)
                {
                        ch2 = strcmp(chan_old->sym_name,
                                ptr_sym_1);
                        if(ch2 == 0)
                                count_match++;
                }
        }
        chan_old = chan_old->next;
}
count_match++;
sprintf(ptr_i, "%d", count_match);
ptr_int = ptr_i;
                /* set default channel name */
strcat(channel_name, ptr_sym_1);
strcat(channel_name, "-");
strcat(channel_name, ptr_sym_2);
strcat(channel_name, "-chan-");
strcat(channel_name, ptr_int);
ptr_chan_name = channel_name;
panel_set_value(name_item, ptr_chan_name);

panel_create_item(add_chan_panel, PANEL_BUTTON,
                PANEL_LABEL_IMAGE,      panel_button_image
                        (add_chan_panel, "Done - Return to
                                Graphic", 5, 0),
                PANEL_ITEM_X,   330,
                PANEL_ITEM_Y,   50,
```

```
                    PANEL_NOTIFY_PROC, return_to_graphic_from_popup,
                    0);

        panel_create_item(add_chan_panel, PANEL_BUTTON,
                    PANEL_LABEL_IMAGE,      panel_button_image
                            (add_chan_panel, "Cancel",
                                    5, 0),
                    PANEL_ITEM_X,   330,
                    PANEL_ITEM_Y,   80,
                    PANEL_NOTIFY_PROC,      cancel_proc,
                    0);

        window_fit(add_chan_panel);
        window_fit(add_chan_frame);
}
```

```
/*****************************************************************/
/* posts new channel to storage and the channel name to specifications
** in the text windows */
void
post_channel(p_chan)
char   *p_chan;
{
Pixwin *pw;
int    int1, int2, insert_index, buf_len;

                    /* update list of channels */
       int1 = (int) panel_get_value(interact_item_1);
       int2 = (int) panel_get_value(interact_item_2);

       if( ((int1 == 0) && (int2 == 0)) || ((int1 == 0) && (int2 == 1))
              || ((int1 == 1) && (int2 == 2)) ||
                    ((int1 == 2) && (int2 == 1)) ||
                          ((int1 == 1) && (int2 == 0)) )
                    /* add channel to lists of channels */
       add_endpoint(ptr_sym_1, int1, input_channel_name);
       add_endpoint(ptr_sym_2, int2, input_channel_name);
       add_chan_to_list(ptr_sym_1, ptr_sym_2, int1, int2, p_chan);

                    /* redraw entire graphic, lines first */
       redraw_graphic();

                    /* write channel name with selected exchange
                    ** type into specification */
       insert_index = find_index_to_subwin(ptr_sym_1);
       buf_len = build_insert_line(ptr_sym_1, int1);
       textsw_insert(textsw[insert_index], ptr_insert_line, buf_len);

       insert_index = find_index_to_subwin(ptr_sym_2);
       buf_len = build_insert_line(ptr_sym_2, int2);
       textsw_insert(textsw[insert_index], ptr_insert_line, buf_len);
}
```

```
/*****************************************************************/
/* routine to determine x,y location of symbol on canvas **/
void
seek_xy(ptr_to_sym_name)
char    *ptr_to_sym_name;
{
int     found_it;
char    *ptr_to_list_name;

        found_it = 1;
        traverse = head;
        while((traverse != NULL) && (found_it != 0))
        {
                ptr_to_list_name = traverse->symbol_name;
                found_it = strcmp(ptr_to_list_name, ptr_to_sym_name);
                if(found_it == 0)
                {
                        sym_x = traverse->symbol_x + 32;
                        sym_y = traverse->symbol_y + 32;
                }
                traverse = traverse->next;
        }  /* end while */
}

/*****************************************************************/
/* routine to add to list of channels **/
void
add_chan_to_list(sym_1, sym_2, inter1, inter2, chn_name)
char    *sym_1, *sym_2;
int     inter1, inter2;
char    *chn_name;
{
int k;

        chan_new = (struct chan_rec *) malloc (sizeof (struct
                                chan_rec));

        for(k = 0; k < MAX_SYMBOL_NAME; k++)
                chan_new->sym_name[k] = '\0';
        strcpy(chan_new->sym_name, sym_1);
        strcpy(chan_new->sym_name_other_end, sym_2);
                        /* identify and record interactions */
        chan_new->interact_type_1 = inter1;
        chan_new->interact_type_2 = inter2;
        strcpy(chan_new->chan_name, chn_name);
        chan_new->next = NULL;

        if(chan_head == NULL)
        {
                chan_head = chan_new;
                chan_tail = chan_new;
        }
        else
        {
                chan_tail->next = chan_new;
```

```
                chan_tail = chan_new;
        }
}
```

```
/*****************************************************************/
/* obtains the number of the text subwindow for a given symbol name */
find_index_to_subwin(ref_name)
char    *ref_name;
{
int     counter, found_index;


                /* identify symbol in list of symbols */
        counter = 0;
        found_index = 1;
        while((counter < MAX_SUBWINS) && (found_index != 0))
        {
                found_index = strcmp(sub_list[counter].name, ref_name);
                if(found_index == 0)
                                /* obtain the index */
                        return(counter);
                counter ++;
        }
}

/*****************************************************************/
/* concatenates the channel interaction type and channel name into an
** array for eventual insert into the specification */
build_insert_line(ref_sym_name, exchange_type)
char    *ref_sym_name;
int     exchange_type;
{
int     n, length;
char    *exchange;

                        /* clear insert_line array */
        for(n = 0; n < MAX_CHANNEL_NAME; n++)
                insert_line[n] = '\0';

        switch (exchange_type)
        {
                case 0 : exchange = "";
                         break;
                case 1 : exchange = "r";
                         break;
                case 2 : exchange = "m";
        }

                        /* build the line */
        strcat(insert_line, "x");
        strcat(insert_line, exchange);
        strcat(insert_line, "-");
        strcat(insert_line, input_channel_name);
        ptr_insert_line = insert_line;
        length = strlen(ptr_insert_line);
        return(length);
}
```

```
/*****************************************************************/
/* deallocates the linked list of existing channels on the canvas */
void
delete_chan_list()
{
                /* clean up and deallocate linked list to system */
        chan_old = chan_head;
        while(chan_old != NULL)
        {
                chan_head = chan_old->next;
                free(chan_old);
                chan_old = chan_head;
        }
}
```

```
#include "./pg.h"

void    action_menu_event_proc();
void    chan_panel_bg_proc();
void    show_other_end();
void    create_choices();
void    make_item();

void    get_select();
void    end_proc();

char * get_sel();

extern void    advance_to_subwin();
extern get_window_num();

                /* objects included in the Show Other End feature */
Panel  show_chan_panel;
Panel_item action_item;
Panel_item select_item;
Menu   action_menu;
Frame  show_chan_frame;

int     left, top,      /* for centering the popup */
        count_chan,     /* count of channels as adding to panel choices
                        */
        selected,       /* chosen symbol from list of symbols on chan */
        match,          /* to determine if have found the desired string
                        */
        index_to_windows;       /* which textsw made request */

                        /* the current symbol for which user is showing
                                end */
char    *ptr_symbol_name;

                        /* list of current choices on the other end
                                popup */
struct chan_choice {
                        char    symbol[MAX_SYMBOL_NAME];
} chan_list[MAX_CHANNELS];
```

```
/*****************************************************************/
/* creates the panel containing a list of the symbols and interaction
** types at the other end of a given symbol/channel combination */
void
go_to_show_endpoint(chosen_item)
Panel_item      chosen_item;
{
                        /* Indicate that endpoint popup is open */
        chan_end_displayed = 1;
        index_to_windows = get_window_num(chosen_item);
        show_chan_frame =
           window_create(pg_subwin_frame[index_to_windows],
                        FRAME,
                        FRAME_DONE_PROC,        end_proc,
                        0);

        show_chan_panel = window_create(show_chan_frame, PANEL,
                        WIN_HEIGHT,             100,
                        WIN_VERTICAL_SCROLLBAR, scrollbar_create(
                                SCROLL_PLACEMENT,       SCROLL_EAST,
                                 0),
                        PANEL_BACKGROUND_PROC,  chan_panel_bg_proc,
                        PANEL_EVENT_PROC,       action_menu_event_proc,
                        0);

        panel_create_item(show_chan_panel,      PANEL_BUTTON,
                        PANEL_LABEL_IMAGE,      panel_button_image(
                                show_chan_panel, "Cancel", 5, 0),
                        PANEL_ITEM_X,   110,
                        PANEL_ITEM_Y,   15,
                        PANEL_NOTIFY_PROC,      end_proc,
                        0);

        select_item = panel_create_item(show_chan_panel, PANEL_CHOICE,
                        PANEL_LABEL_STRING,     "     Symbol Name
                                Exchange",
                        PANEL_DISPLAY_LEVEL,    PANEL_ALL,
                        PANEL_FEEDBACK, PANEL_MARKED,
                        PANEL_LAYOUT,   PANEL_VERTICAL,
                        PANEL_CHOICE_STRINGS,   "", "", "", "", "", "",
                                "", "", "", "", "", "", "", "", "", 0,
                        PANEL_ITEM_X,   25,
                        PANEL_ITEM_Y,   55,
                        0);

        action_menu = menu_create(MENU_TITLE_ITEM,      "Action Menu",
                MENU_ACTION_ITEM,       "Show New Specification",
                        show_other_end,
                MENU_ACTION_ITEM,       "Return to Specification",
                        end_proc,
                0);

        window_fit(show_chan_panel);
        window_fit(show_chan_frame);
```

```
        center_popup(show_chan_frame);

        window_set(show_chan_frame, WIN_X,      left,
                                    WIN_Y,      top,
                                    WIN_SHOW,   TRUE,
                                    0);

                        /* place appropriate choices on panel */
        create_choices();
}
```

```
/*************************************************************************/
/* terminates the channel endpoint panel */
void
end_proc()
{
        chan_end_displayed = 0;
                        /* On the return, get rid of show_chan_frame */
        window_set(show_chan_frame, FRAME_NO_CONFIRM, TRUE, 0);
        window_destroy(show_chan_frame);
}

/*************************************************************************/
/* manages the mouse interactions on the channel endpoint panel */
void
action_menu_event_proc(item, event)
Panel_item      item;
Event   *event;
{
        chan_panel_bg_proc();
        if (event_id(event) == MS_RIGHT)
        {
                Event   *adjusted_event;
                        adjusted_event = panel_window_event(
                                        show_chan_panel, event, 0);
                        chan_panel_bg_proc(item, event);
        }
        else
                panel_default_handle_event(item, event);
}
```

```
/*************************************************************************/
/* handles the display of the action menu on the channel endpoint panel
*/
void
chan_panel_bg_proc(panel, event)
Panel  panel;
Event  *event;
{
        if(event_id(event) == MS_RIGHT)
                menu_show(action_menu, show_chan_panel, event, 0);
}

/*************************************************************************/
/* pulls up a new spec subwindow for the end of the channel desired to
** view */
void
show_other_end()
{
int     found_sym;
char    *name_selected;

        get_select();

                        /* make sure its a valid selection */
        if(selected <= (count_chan - 1))
        {
                name_selected = chan_list[selected].symbol;

                        /* find the structure for that symbol */
                found_sym = 1;
                traverse = head;
                while((traverse != NULL) && (found_sym != 0))
                {
                        found_sym = strcmp(name_selected,
                                        traverse->symbol_name);
                        if(found_sym != 0)
                                traverse = traverse->next;
                }
                showing_other_end = 1;
                        /* call go_to_subwin() for that symbol */
                advance_to_subwin(name_selected, traverse);
        }   /* end if a valid selection */
}

/*************************************************************************/
/* obtains the parameter of the user's selection from the list of
** choices displayed */
void
get_select()
{
        selected = (int) panel_get_value(select_item);
}
```

```
/*****************************************************************/
/* traverses the list of endpoints to find all occurrences of a channel
** and place them on the channel endpoint panel */
void
create_choices()
{

int     found, caught;
char    *compared_chan, *actual_chan;

                        /* get selection from user */
        actual_chan = get_sel();

        strcpy(save_this_channel, actual_chan);

                        /* get name of symbol being dealt with */
        ptr_symbol_name = sub_list[index_to_windows].name;

                        /* search chan endpoint list for occurrences of
                                chan */
        count_chan = 0;
        end_old = end_head;
        while((end_old != NULL) && (count_chan < MAX_CHOICES))
        {
                compared_chan = end_old->ch_name;
                found = strcmp(actual_chan, compared_chan);
                if(found == 0)
                {
                                /* decide/add symbol as a panel item */
                        make_item(end_old->sy_name,
                                end_old->inter);
                        if (match != 0)
                                count_chan++;
                }
                end_old = end_old->next;
        }
}
```

```
/*****************************************************************/
/* sets the symbol name and interaction type on a channel endpoint
** panel item */
void
make_item(name_of_chan, interact)
char    name_of_chan[];
int     interact;
{
int     k, num_char, offset_char, stagger;
char    *ptr_endpoint, build_title[MAX_COMMAND], *ptr_title;

                        /* assign a pointer to open symbol name */
        ptr_endpoint = name_of_chan;
        match = strcmp(ptr_symbol_name, ptr_endpoint);
        if(match != 0)  /* choose symbol that is not the one user is in
                        */
        {
                        /* make title string */
                for(k = 0; k < MAX_COMMAND; k++)
                        build_title[k] = '\0';
                        /* calculate offset for length of symbol name */
                num_char = strlen(ptr_endpoint);
                offset_char = (15 - num_char) / 2;
                for(k = 0; k < offset_char; k++)
                        strcat(build_title, " ");
                strcat(build_title, ptr_endpoint);
                stagger = (15 - offset_char - num_char);
                for(k = 0; k < stagger; k++)
                        strcat(build_title, " ");
                strcat(build_title, "    ");
                switch(interact)
                {
                        case 0 : strcat(build_title, "x");
                                 break;
                        case 1 : strcat(build_title, "xr");
                                 break;
                        case 2 : strcat(build_title, "xm");
                }
                ptr_title = build_title;

                        /* make item */
                panel_set(select_item, PANEL_CHOICE_STRING, count_chan,
                        ptr_title, 0);

                        /* remember the choice */
                strcpy(chan_list[count_chan].symbol, ptr_endpoint);
        }
}
```

```
/*****************************************************************/
/* returns the current selection on the terminal */
char *
get_sel()
{
static char     ret_string[MAX_CHANNEL_NAME];
Seln_holder     holder;
Seln_request    *buffer;

        holder = seln_inquire(SELN_PRIMARY);
        buffer = seln_ask(&holder, SELN_REQ_CONTENTS_ASCII, 0, 0);

        strncpy(ret_string,
                buffer->data + sizeof(Seln_attribute),
                MAX_CHANNEL_NAME);

                /* returns empty string if no selection */
        return(ret_string);
}
```

Directory: pg_icons

File: sy_PyGraph.picon - ICONIC SYMBOL FOR PYGRAPH IN ITS CLOSED STATE

```
/* Format_version=1, Width=64, Height=64, Depth=1, **
Valid_bits_per_item=16 */
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x81E0,0x0380,0x0000,0x4001,0x8110,0x0440,0x0000,0x4001,
        0x8112,0x240B,0x0E2C,0x5801,0x8112,0x240C,0x9132,0x6401,
        0x81E1,0x24C8,0x0122,0x4401,0x8101,0x2448,0x0F22,0x4401,
        0x8100,0xA448,0x1122,0x4401,0x8100,0xC4C8,0x1132,0x4401,
        0x8100,0x4348,0x0F2C,0x4401,0x8000,0x4000,0x0020,0x0001,
        0x8002,0x8000,0x0020,0x0001,0x8009,0x0000,0x0020,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0800,0x0001,0x8008,0x0000,0x1400,0x0001,
        0x8008,0x0000,0x1200,0x0001,0x8008,0x0000,0x2200,0x0001,
        0x8008,0x0000,0x4100,0x0001,0x8008,0x0000,0x4080,0x0001,
        0x8008,0x0000,0x8040,0x0001,0x8008,0x0001,0x0020,0x0001,
        0x8008,0x0002,0x0010,0x0001,0x8008,0x0002,0x0010,0x0001,
        0x8008,0x0004,0x0008,0x0001,0x8008,0x0008,0x0004,0x0001,
        0x8008,0x0008,0x0002,0x0001,0x8008,0x0010,0x0001,0x0001,
        0x8008,0x0020,0x0000,0x8001,0x8008,0x0020,0x0000,0x8001,
        0x8008,0x0040,0x0000,0x4001,0x8008,0x0080,0x0000,0x2001,
        0x8008,0x0100,0x0000,0x1001,0x8008,0x0100,0x0000,0x0801,
        0x8008,0x0200,0x0000,0x0401,0x8008,0x0400,0x0000,0x0401,
        0x8008,0x0400,0x0000,0x0201,0x8008,0x0FFF,0xFFFF,0xFF01,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0x8008,0x0000,0x0000,0x0001,
        0x8008,0x0000,0x0000,0x0001,0xFFFF,0xFFFF,0xFFFF,0xFFFF
```

File: sy_blank.picon - ICON FOR BLANK SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0007,0xFFFF,0xFFFF,0xE000,
        0x0038,0x0000,0x0000,0x1C00,0x00C0,0x0000,0x0000,0x0300,
        0x0100,0x0000,0x0000,0x0080,0x0600,0x0000,0x0000,0x0060,
        0x0800,0x0000,0x0000,0x0010,0x0800,0x0000,0x0000,0x0010,
        0x1000,0x0000,0x0000,0x0008,0x2000,0x0000,0x0000,0x0004,
        0x2000,0x0000,0x0000,0x0004,0x4000,0x0000,0x0000,0x0002,
        0x4007,0x8401,0x8424,0x2002,0x4004,0x4401,0x8624,0x4002,
        0x8004,0x4401,0x8624,0x8001,0x8004,0x4402,0x4525,0x0001,
        0x8007,0x8402,0x4526,0x0001,0x8004,0x4402,0x44A7,0x0001,
        0x8004,0x2404,0x24A5,0x8001,0x8004,0x2407,0xE464,0xC001,
        0x8004,0x2404,0x2464,0x6001,0x4007,0xC7E4,0x2424,0x2002,
        0x4000,0x0000,0x0000,0x0002,0x4000,0x0000,0x0000,0x0002,
        0x2000,0x0000,0x0000,0x0004,0x2000,0x0000,0x0000,0x0004,
        0x1000,0x0000,0x0000,0x0008,0x0800,0x0000,0x0000,0x0010,
        0x0800,0x0000,0x0000,0x0010,0x0600,0x0000,0x0000,0x0060,
        0x0100,0x0000,0x0000,0x0080,0x00C0,0x0000,0x0000,0x0300,
        0x0038,0x0000,0x0000,0x1C00,0x0007,0xFFFF,0xFFFF,0xE000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
```

File: sy_controlled.picon - ICON FOR CONTROLLED OBJECT SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0001,0x8000,0x0000,0x0000,0x0002,0x4000,0x0000,
        0x0000,0x0004,0x2000,0x0000,0x0000,0x0008,0x1000,0x0000,
        0x0000,0x0010,0x0800,0x0000,0x0000,0x0020,0x0400,0x0000,
        0x0000,0x0040,0x0200,0x0000,0x0000,0x0080,0x0100,0x0000,
        0x0000,0x0100,0x0080,0x0000,0x0000,0x0200,0x0040,0x0000,
        0x0000,0x0400,0x0020,0x0000,0x0000,0x0800,0x0010,0x0000,
        0x0000,0x1000,0x0008,0x0000,0x0000,0x2000,0x0004,0x0000,
        0x0000,0x4000,0x0002,0x0000,0x0000,0x8000,0x0001,0x0000,
        0x0001,0x0000,0x0000,0x8000,0x0002,0x0000,0x0000,0x4000,
        0x0004,0x0000,0x0000,0x2000,0x0008,0x0000,0x0000,0x1000,
        0x0010,0x0000,0x0000,0x0800,0x0020,0x0000,0x0000,0x0400,
        0x0040,0x0000,0x0000,0x0200,0x0080,0x0000,0x0000,0x0100,
        0x0100,0x0000,0x0000,0x0080,0x0200,0x0000,0x0000,0x0040,
        0x0400,0x0000,0x0000,0x0020,0x0C31,0x17DC,0x3104,0x1E70,
        0x1249,0x9112,0x4904,0x1048,0x3049,0x5112,0x4904,0x1C4C,
        0x5049,0x311C,0x4904,0x104A,0x9249,0x1114,0x4904,0x1049,
        0x8C31,0x1112,0x31E7,0x9E71,0x4000,0x0000,0x0000,0x0002,
        0x2000,0xC70E,0x78C7,0xC004,0x1001,0x2482,0x4121,0x0008,
        0x0801,0x2702,0x7101,0x0010,0x0401,0x2482,0x4101,0x0020,
        0x0201,0x2492,0x4121,0x0040,0x0100,0xC70C,0x78C1,0x0080,
        0x0080,0x0000,0x0000,0x0100,0x0040,0x0000,0x0000,0x0200,
        0x0020,0x0000,0x0000,0x0400,0x0010,0x0000,0x0000,0x0800,
        0x0008,0x0000,0x0000,0x1000,0x0004,0x0000,0x0000,0x2000,
        0x0002,0x0000,0x0000,0x4000,0x0001,0x0000,0x0000,0x8000,
        0x0000,0x8000,0x0001,0x0000,0x0000,0x4000,0x0002,0x0000,
        0x0000,0x2000,0x0004,0x0000,0x0000,0x1000,0x0008,0x0000,
        0x0000,0x0800,0x0010,0x0000,0x0000,0x0400,0x0020,0x0000,
        0x0000,0x0200,0x0040,0x0000,0x0000,0x0100,0x0080,0x0000,
        0x0000,0x0080,0x0100,0x0000,0x0000,0x0040,0x0200,0x0000,
        0x0000,0x0020,0x0400,0x0000,0x0000,0x0010,0x0800,0x0000,
        0x0000,0x0008,0x1000,0x0000,0x0000,0x0004,0x2000,0x0000,
        0x0000,0x0002,0x4000,0x0000,0x0000,0x0001,0x8000,0x0000
```

File: sy_handler.picon - ICON FOR HANDLER SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,
        0x0000,0x0002,0x4000,0x0000,0x0000,0x0002,0x4000,0x0000,
        0x0000,0x0004,0x2000,0x0000,0x0000,0x0004,0x2000,0x0000,
        0x0000,0x0008,0x1000,0x0000,0x0000,0x0008,0x1000,0x0000,
        0x0000,0x0010,0x0800,0x0000,0x0000,0x0010,0x0800,0x0000,
        0x0000,0x0020,0x0400,0x0000,0x0000,0x0020,0x0400,0x0000,
        0x0000,0x0040,0x0200,0x0000,0x0000,0x0040,0x0200,0x0000,
        0x0000,0x0080,0x0100,0x0000,0x0000,0x0080,0x0100,0x0000,
        0x0000,0x0100,0x0080,0x0000,0x0000,0x0100,0x0080,0x0000,
        0x0000,0x0200,0x0040,0x0000,0x0000,0x0200,0x0040,0x0000,
        0x0000,0x0400,0x0020,0x0000,0x0000,0x0400,0x0020,0x0000,
        0x0000,0x0800,0x0010,0x0000,0x0000,0x0800,0x0010,0x0000,
        0x0000,0x1000,0x0008,0x0000,0x0000,0x1000,0x0008,0x0000,
        0x0000,0x2000,0x0004,0x0000,0x0000,0x2000,0x0004,0x0000,
        0x0000,0x4000,0x0002,0x0000,0x0000,0x4000,0x0002,0x0000,
        0x0000,0x8000,0x0001,0x0000,0x0000,0x8000,0x0001,0x0000,
        0x0001,0x0000,0x0000,0x8000,0x0001,0x0000,0x0000,0x8000,
        0x0002,0x0000,0x0000,0x4000,0x0002,0x0000,0x0000,0x4000,
        0x0004,0x0000,0x0000,0x2000,0x0004,0x0000,0x0000,0x2000,
        0x0008,0x0000,0x0000,0x1000,0x0008,0x0000,0x0000,0x1000,
        0x0010,0x0000,0x0000,0x0800,0x0010,0x0000,0x0000,0x0800,
        0x0020,0x0000,0x0000,0x0400,0x0020,0x0000,0x0000,0x0400,
        0x0040,0x0000,0x0000,0x0200,0x0040,0x0000,0x0000,0x0200,
        0x0080,0x0000,0x0000,0x0100,0x0080,0x0000,0x0000,0x0100,
        0x0100,0x0000,0x0000,0x0080,0x0088,0xE627,0x881F,0x3C80,
        0x0289,0x1624,0x4810,0x2240,0x0289,0x1524,0x4810,0x2240,
        0x04F9,0x1524,0x481E,0x3C20,0x0489,0xF4A4,0x4810,0x2820,
        0x0889,0x14A4,0x4810,0x2410,0x0889,0x1464,0x4810,0x2210,
        0x1089,0x1467,0x8F9F,0x2208,0x1000,0x0000,0x0000,0x0008,
        0x2000,0x0000,0x0000,0x0004,0x2000,0x0000,0x0000,0x0004,
        0x4000,0x0000,0x0000,0x0002,0x4000,0x0000,0x0000,0x0002,
        0x8000,0x0000,0x0000,0x0001,0xFFFF,0xFFFF,0xFFFF,0xFFFF
```

File: sy_informer.picon - ICON FOR INFORMER SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8F91,0x3E38,0xF213,0xE781,
        0x8219,0x2044,0x8A12,0x0441,0x8219,0x2044,0x8B32,0x0441,
        0x8215,0x2044,0x8B32,0x0441,0x8215,0x3C44,0xF2D3,0xC781,
        0x8213,0x2044,0x92D2,0x0481,0x8213,0x2044,0x8A12,0x0441,
        0x8211,0x2044,0x8A12,0x0441,0x8F91,0x2038,0x8A13,0xE441,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0x8000,0x0000,0x0000,0x0001,0x8000,0x0000,0x0000,0x0001,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
```

File: sy_input.picon - ICON FOR INPUT SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0000,0x0000,0x0003,0x0000,0x0000,0x0000,0x000D,
        0x0000,0x0000,0x0000,0x0031,0x0000,0x0000,0x0000,0x00C1,
        0x0000,0x0000,0x0000,0x0301,0x0000,0x0000,0x0000,0x0C01,
        0x0000,0x0000,0x0000,0x3001,0x0000,0x0000,0x0000,0xC001,
        0x0000,0x0000,0x0003,0x0001,0x0000,0x0000,0x000C,0x0001,
        0x0000,0x0000,0x0030,0x0001,0x0000,0x0000,0x00C0,0x0001,
        0x0000,0x0000,0x0300,0x0001,0x0000,0x0000,0x0C00,0x0001,
        0x0000,0x0000,0x3000,0x0001,0x0000,0x0000,0xC000,0x0001,
        0x0000,0x0003,0x0000,0x0001,0x0000,0x000C,0x0000,0x0001,
        0x0000,0x0030,0x0000,0x0001,0x0000,0x00C0,0x0000,0x0001,
        0x0000,0x0300,0x0000,0x0001,0x0000,0x0C00,0x0000,0x0001,
        0x0000,0x3000,0x0000,0x0001,0x0000,0xC000,0x0000,0x0001,
        0x0003,0x0000,0x0000,0x0001,0x000C,0x0000,0x0000,0x0001,
        0x0030,0x0000,0x0000,0x0001,0x00C0,0x003E,0x44F1,0x13E1,
        0x0300,0x0008,0x6489,0x1081,0x0C00,0x0008,0x6489,0x1081,
        0x3000,0x0008,0x5489,0x1081,0xC000,0x0008,0x54F1,0x1081,
        0xC000,0x0008,0x4C81,0x1081,0x3000,0x0008,0x4C81,0x1081,
        0x0C00,0x0008,0x4481,0x1081,0x0300,0x003E,0x4480,0xE081,
        0x00C0,0x0000,0x0000,0x0001,0x0030,0x0000,0x0000,0x0001,
        0x000C,0x0000,0x0000,0x0001,0x0003,0x0000,0x0000,0x0001,
        0x0000,0xC000,0x0000,0x0001,0x0000,0x3000,0x0000,0x0001,
        0x0000,0x0C00,0x0000,0x0001,0x0000,0x0300,0x0000,0x0001,
        0x0000,0x00C0,0x0000,0x0001,0x0000,0x0030,0x0000,0x0001,
        0x0000,0x000C,0x0000,0x0001,0x0000,0x0003,0x0000,0x0001,
        0x0000,0x0000,0xC000,0x0001,0x0000,0x0000,0x3000,0x0001,
        0x0000,0x0000,0x0C00,0x0001,0x0000,0x0000,0x0300,0x0001,
        0x0000,0x0000,0x00C0,0x0001,0x0000,0x0000,0x0030,0x0001,
        0x0000,0x0000,0x000C,0x0001,0x0000,0x0000,0x0003,0x0001,
        0x0000,0x0000,0x0000,0xC001,0x0000,0x0000,0x0000,0x3001,
        0x0000,0x0000,0x0000,0x0C01,0x0000,0x0000,0x0000,0x0301,
        0x0000,0x0000,0x0000,0x00C1,0x0000,0x0000,0x0000,0x0031,
        0x0000,0x0000,0x0000,0x000D,0x0000,0x0000,0x0000,0x0003
```

File: sy_modifier.picon - ICON FOR MODIFIER SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0xFFFF,0xFFFF,0xFFFF,0xFF00,0x8000,0x0000,0x0000,0x0100,
        0x8000,0x0000,0x0000,0x0100,0x8000,0x0000,0x0000,0x0100,
        0x4000,0x0000,0x0000,0x0080,0x4000,0x0000,0x0000,0x0080,
        0x4000,0x0000,0x0000,0x0080,0x4000,0x0000,0x0000,0x0080,
        0x4000,0x0000,0x0000,0x0080,0x4000,0x0000,0x0000,0x0080,
        0x4000,0x0000,0x0000,0x0080,0x2000,0x0000,0x0000,0x0040,
        0x2000,0x0000,0x0000,0x0040,0x2000,0x0000,0x0000,0x0040,
        0x2000,0x0000,0x0000,0x0040,0x2000,0x0000,0x0000,0x0040,
        0x2000,0x0000,0x0000,0x0040,0x2000,0x0000,0x0000,0x0040,
        0x1000,0x0000,0x0000,0x0020,0x1000,0x0000,0x0000,0x0020,
        0x1000,0x0000,0x0000,0x0020,0x1000,0x0000,0x0000,0x0020,
        0x106C,0xC70E,0x7CE7,0x9C20,0x106D,0x2484,0x4044,0x1220,
        0x1055,0x2484,0x7847,0x1220,0x0855,0x2484,0x4044,0x1C10,
        0x0845,0x2484,0x4044,0x1410,0x0844,0xC70E,0x40E7,0x9210,
        0x0800,0x0000,0x0000,0x0010,0x0800,0x0000,0x0000,0x0010,
        0x0800,0x0000,0x0000,0x0010,0x0400,0x0000,0x0000,0x0008,
        0x0400,0x0000,0x0000,0x0008,0x0400,0x0000,0x0000,0x0008,
        0x0400,0x0000,0x0000,0x0008,0x0400,0x0000,0x0000,0x0008,
        0x0400,0x0000,0x0000,0x0008,0x0400,0x0000,0x0000,0x0008,
        0x0200,0x0000,0x0000,0x0004,0x0200,0x0000,0x0000,0x0004,
        0x0200,0x0000,0x0000,0x0004,0x0200,0x0000,0x0000,0x0004,
        0x0200,0x0000,0x0000,0x0004,0x0200,0x0000,0x0000,0x0004,
        0x0200,0x0000,0x0000,0x0004,0x0100,0x0000,0x0000,0x0002,
        0x0100,0x0000,0x0000,0x0002,0x0100,0x0000,0x0000,0x0002,
        0x0100,0x0000,0x0000,0x0002,0x0100,0x0000,0x0000,0x0002,
        0x0100,0x0000,0x0000,0x0002,0x0100,0x0000,0x0000,0x0002,
        0x0080,0x0000,0x0000,0x0001,0x0080,0x0000,0x0000,0x0001,
        0x0080,0x0000,0x0000,0x0001,0x00FF,0xFFFF,0xFFFF,0xFFFF,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
```

File: sy_monitor.picon - ICON FOR MONITOR SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x001F,0xFC00,0x0000,0x0000,0x01E0,0x03C0,0x0000,
        0x0000,0x0E00,0x0038,0x0000,0x0000,0x3000,0x0006,0x0000,
        0x0000,0xC000,0x0001,0x8000,0x0001,0x0000,0x0000,0x4000,
        0x0002,0x0000,0x0000,0x2000,0x000C,0x0000,0x0000,0x1800,
        0x0010,0x0000,0x0000,0x0400,0x0020,0x0000,0x0000,0x0200,
        0x0040,0x0000,0x0000,0x0100,0x0080,0x0000,0x0000,0x0080,
        0x0080,0x0000,0x0000,0x0080,0x0100,0x0000,0x0000,0x0040,
        0x0200,0x0000,0x0000,0x0020,0x0400,0x0000,0x0000,0x0010,
        0x0400,0x0000,0x0000,0x0010,0x0800,0x0000,0x0000,0x0008,
        0x0800,0x0000,0x0000,0x0008,0x1000,0x0000,0x0000,0x0004,
        0x1000,0x0000,0x0000,0x0004,0x1000,0x0000,0x0000,0x0004,
        0x2000,0x0000,0x0000,0x0002,0x2000,0x0000,0x0000,0x0002,
        0x2000,0x0000,0x0000,0x0002,0x2000,0x0000,0x0000,0x0002,
        0x4000,0x0000,0x0000,0x0001,0x4084,0x7113,0xE7C7,0x1E01,
        0x4084,0x8990,0x8108,0x9101,0x40CC,0x8990,0x8108,0x9101,
        0x40CC,0x8950,0x8108,0x9101,0x40B4,0x8950,0x8108,0x9E01,
        0x40B4,0x8930,0x8108,0x9201,0x4084,0x8930,0x8108,0x9101,
        0x4084,0x8910,0x8108,0x9101,0x4084,0x7113,0xE107,0x1101,
        0x4000,0x0000,0x0000,0x0001,0x2000,0x0000,0x0000,0x0002,
        0x2000,0x0000,0x0000,0x0002,0x2000,0x0000,0x0000,0x0002,
        0x2000,0x0000,0x0000,0x0002,0x1000,0x0000,0x0000,0x0004,
        0x1000,0x0000,0x0000,0x0004,0x1000,0x0000,0x0000,0x0004,
        0x0800,0x0000,0x0000,0x0008,0x0800,0x0000,0x0000,0x0008,
        0x0400,0x0000,0x0000,0x0010,0x0400,0x0000,0x0000,0x0010,
        0x0200,0x0000,0x0000,0x0020,0x0100,0x0000,0x0000,0x0040,
        0x0080,0x0000,0x0000,0x0080,0x0080,0x0000,0x0000,0x0080,
        0x0040,0x0000,0x0000,0x0100,0x0020,0x0000,0x0000,0x0200,
        0x0010,0x0000,0x0000,0x0400,0x000C,0x0000,0x0000,0x1800,
        0x0002,0x0000,0x0000,0x2000,0x0001,0x0000,0x0000,0x4000,
        0x0000,0xC000,0x0001,0x8000,0x0000,0x3000,0x0006,0x0000,
        0x0000,0x0E00,0x0038,0x0000,0x0000,0x01E0,0x03C0,0x0000,
        0x0000,0x001F,0xFC00,0x0000,0x0000,0x0000,0x0000,0x0000
```

File: sy_output.picon - ICON FOR OUTPUT SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0xC000,0x0000,0x0000,0x0000,0xB000,0x0000,0x0000,0x0000,
        0x8C00,0x0000,0x0000,0x0000,0x8300,0x0000,0x0000,0x0000,
        0x80C0,0x0000,0x0000,0x0000,0x8030,0x0000,0x0000,0x0000,
        0x800C,0x0000,0x0000,0x0000,0x8003,0x0000,0x0000,0x0000,
        0x8000,0xC000,0x0000,0x0000,0x8000,0x3000,0x0000,0x0000,
        0x8000,0x0C00,0x0000,0x0000,0x8000,0x0300,0x0000,0x0000,
        0x8000,0x00C0,0x0000,0x0000,0x8000,0x0030,0x0000,0x0000,
        0x8000,0x000C,0x0000,0x0000,0x8000,0x0003,0x0000,0x0000,
        0x8000,0x0000,0xC000,0x0000,0x8000,0x0000,0x3000,0x0000,
        0x8000,0x0000,0x0C00,0x0000,0x8000,0x0000,0x0300,0x0000,
        0x8000,0x0000,0x00C0,0x0000,0x8000,0x0000,0x0030,0x0000,
        0x8000,0x0000,0x000C,0x0000,0x8000,0x0000,0x0003,0x0000,
        0x8000,0x0000,0x0000,0xC000,0x8000,0x0000,0x0000,0x3000,
        0x8000,0x0000,0x0000,0x0C00,0x8E22,0x7CF1,0x13E0,0x0300,
        0x9122,0x1089,0x1080,0x00C0,0x9122,0x1089,0x1080,0x0030,
        0x9122,0x1089,0x1080,0x000C,0x9122,0x10F1,0x1080,0x0003,
        0x9122,0x1081,0x1080,0x0003,0x9122,0x1081,0x1080,0x000C,
        0x9122,0x1081,0x1080,0x0030,0x8E1C,0x1080,0xE080,0x00C0,
        0x8000,0x0000,0x0000,0x0300,0x8000,0x0000,0x0000,0x0C00,
        0x8000,0x0000,0x0000,0x3000,0x8000,0x0000,0x0000,0xC000,
        0x8000,0x0000,0x0003,0x0000,0x8000,0x0000,0x000C,0x0000,
        0x8000,0x0000,0x0030,0x0000,0x8000,0x0000,0x00C0,0x0000,
        0x8000,0x0000,0x0300,0x0000,0x8000,0x0000,0x0C00,0x0000,
        0x8000,0x0000,0x3000,0x0000,0x8000,0x0000,0xC000,0x0000,
        0x8000,0x0003,0x0000,0x0000,0x8000,0x000C,0x0000,0x0000,
        0x8000,0x0030,0x0000,0x0000,0x8000,0x00C0,0x0000,0x0000,
        0x8000,0x0300,0x0000,0x0000,0x8000,0x0C00,0x0000,0x0000,
        0x8000,0x3000,0x0000,0x0000,0x8000,0xC000,0x0000,0x0000,
        0x8003,0x0000,0x0000,0x0000,0x800C,0x0000,0x0000,0x0000,
        0x8030,0x0000,0x0000,0x0000,0x80C0,0x0000,0x0000,0x0000,
        0x8300,0x0000,0x0000,0x0000,0x8C00,0x0000,0x0000,0x0000,
        0xB000,0x0000,0x0000,0x0000,0xC000,0x0000,0x0000,0x0000
```

File: sy_reader.picon - ICON FOR READER SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0x4000,0x0000,0x0000,0x0001,
        0x4000,0x0000,0x0000,0x0002,0x2000,0x0000,0x0000,0x0002,
        0x2000,0x0000,0x0000,0x0004,0x100F,0x1F08,0x70F9,0xE004,
        0x1008,0x9008,0x4881,0x1008,0x0808,0x9014,0x4481,0x1008,
        0x0808,0x9014,0x4481,0x1010,0x040F,0x1E14,0x44F1,0xE010,
        0x0409,0x1022,0x4481,0x2020,0x0208,0x903E,0x4481,0x1020,
        0x0208,0x9022,0x4881,0x1000,0x0108,0x9F22,0x70F9,0x1040,
        0x0100,0x0000,0x0000,0x0080,0x0080,0x0000,0x0000,0x0080,
        0x0080,0x0000,0x0000,0x0100,0x0040,0x0000,0x0000,0x0100,
        0x0040,0x0000,0x0000,0x0200,0x0020,0x0000,0x0000,0x0200,
        0x0020,0x0000,0x0000,0x0400,0x0010,0x0000,0x0000,0x0400,
        0x0010,0x0000,0x0000,0x0800,0x0008,0x0000,0x0000,0x0800,
        0x0008,0x0000,0x0000,0x1000,0x0004,0x0000,0x0000,0x1000,
        0x0004,0x0000,0x0000,0x2000,0x0002,0x0000,0x0000,0x2000,
        0x0002,0x0000,0x0000,0x4000,0x0001,0x0000,0x0000,0x4000,
        0x0001,0x0000,0x0000,0x8000,0x0000,0x8000,0x0000,0x8000,
        0x0000,0x8000,0x0001,0x0000,0x0000,0x4000,0x0001,0x0000,
        0x0000,0x4000,0x0002,0x0000,0x0000,0x2000,0x0002,0x0000,
        0x0000,0x2000,0x0004,0x0000,0x0000,0x1000,0x0004,0x0000,
        0x0000,0x1000,0x0008,0x0000,0x0000,0x0800,0x0008,0x0000,
        0x0000,0x0800,0x0010,0x0000,0x0000,0x0400,0x0010,0x0000,
        0x0000,0x0400,0x0020,0x0000,0x0000,0x0200,0x0020,0x0000,
        0x0000,0x0200,0x0040,0x0000,0x0000,0x0100,0x0040,0x0000,
        0x0000,0x0100,0x0080,0x0000,0x0000,0x0080,0x0080,0x0000,
        0x0000,0x0080,0x0100,0x0000,0x0000,0x0040,0x0100,0x0000,
        0x0000,0x0040,0x0200,0x0000,0x0000,0x0020,0x0200,0x0000,
        0x0000,0x0020,0x0400,0x0000,0x0000,0x0010,0x0400,0x0000,
        0x0000,0x0010,0x0800,0x0000,0x0000,0x0008,0x0800,0x0000,
        0x0000,0x0008,0x1000,0x0000,0x0000,0x0004,0x1000,0x0000,
        0x0000,0x0004,0x2000,0x0000,0x0000,0x0002,0x2000,0x0000,
        0x0000,0x0002,0x4000,0x0000,0x0000,0x0001,0x4000,0x0000,
        0x0000,0x0001,0x8000,0x0000,0x0000,0x0000,0x8000,0x0000
```

File: sy_select.picon - ICON FOR SELECT SYMBOL

```
/* Format_version=1, Width=64, Height=64, Depth=1,
** Valid_bits_per_item=16 */
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0x8888,0x8888,0x8888,0x8889,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0x8888,0x8888,0x8888,0x8889,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0x88FF,0xFFFF,0xFFFF,0xFF89,0x8880,0x0000,0x0000,0x0189,
        0xA280,0x0000,0x0000,0x0123,0xA280,0x0000,0x0000,0x0123,
        0x8880,0x0000,0x0000,0x0189,0x8880,0x0000,0x0000,0x0189,
        0xA280,0x0000,0x0000,0x0123,0xA280,0x0000,0x0000,0x0123,
        0x8800,0x0000,0x0000,0x0089,0x8800,0x0000,0x0000,0x0089,
        0xA23C,0x7E40,0x7E3C,0xFE23,0xA242,0x4040,0x4042,0x1023,
        0x8842,0x4040,0x4042,0x1089,0x8820,0x4040,0x4040,0x1089,
        0xA218,0x7C40,0x7C40,0x1023,0xA204,0x4040,0x4040,0x1023,
        0x8802,0x4040,0x4040,0x1089,0x8842,0x4040,0x4042,0x1089,
        0xA242,0x4040,0x4042,0x1023,0xA23C,0x7E7E,0x7E3C,0x1023,
        0x8800,0x0000,0x0000,0x0089,0x8800,0x0000,0x0000,0x0089,
        0xA200,0x0000,0x0000,0x0023,0xA200,0x0000,0x0000,0x0023,
        0x8880,0x0000,0x0000,0x0189,0x8880,0x0000,0x0000,0x0189,
        0xA280,0x0000,0x0000,0x0123,0xA280,0x0000,0x0000,0x0123,
        0x8880,0x0000,0x0000,0x0189,0x8880,0x0000,0x0000,0x0189,
        0xA280,0x0000,0x0000,0x0123,0xA2FF,0xFFFF,0xFFFF,0xFF23,
        0x8888,0x8888,0x8888,0x8889,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0x8888,0x8888,0x8888,0x8889,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0x8888,0x8888,0x8888,0x8889,0x8888,0x8888,0x8888,0x8889,
        0xA222,0x2222,0x2222,0x2223,0xA222,0x2222,0x2222,0x2223,
        0xFFFF,0xFFFF,0xFFFF,0xFFFF,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
```

File: pg.picon - SYMBOL FOR THE CURSOR ON THE GRAPHIC DEVELOPMENT CANVAS

```
/* Format_version=1, Width=16, Height=16, Depth=1,
** Valid_bits_per_item=16 */
        0xFFFF,0xFFFF,0xC000,0xC000,0xC000,0xC000,0xC000,0xC000,
        0xC000,0xC000,0xC000,0xC000,0xC000,0xC000,0xC000,0xC000
```

```
                                                "** ~1 process "

     initial-~1-state:  --> ~6-STATE;

"**" ~6-STATE~3;
"**" ~6-FEEDBACK~3;
"**" ~6-SENSOR-VALUE~3;

     ~1-cycle : ~6-STATE --> ~6-STATE;
     ~1-cycle[o] =
        proj[(1,(simulate-~1-object
                        [(o,obtain-~1-feedback)],
                 provide-~1-sensor-values[o]
                 ) )];

        obtain-~1-feedback:  --> ~6-FEEDBACK-LIST;
"**"    obtain-~1-feedback = (j#1..1~2
                        < , obtain-~1-j-feedback>);

                 ~6-FEEDBACK-LIST =
"**"                    j#1..1~2
                                < * (~6-FEEDBACK
                                                | FILLER >;

"**"             j#1..1~2
                        < ; obtain-~1-j-feedback:
                                --> ~6-FEEDBACK
                                                | FILLER >;

        simulate-~1-object:
                 ~6-OBJECT-STATE *
                        ~6-FEEDBACK-LIST
                                --> ~6-STATE;

        provide-~1-sensor-values:
                        ~6-STATE --> ~6-STATE;
        provide-~1-sensor-values[o] =
                 proj[(1,(Null)
"**"                    (j#1..1~2< ,
                            provide-~1-sensor-j-value[o] >)
                        ) )];

"**"             j#1..1~2
                 < ;  provide-~1-sensor-j-value:
                                ~6-STATE --> FILLER;
                        provide-~1-sensor-j-value[o] =
                                provide-~1-j-value
                                        [j-~1-sense[o]];

                                j-~1-sense:  ~6-STATE
                                        --> ~6-SENSOR-VALUE;
                        "This should be left undefined."
```

```
                                    provide-~1-j-value:
                                            ~6-SENSOR-VALUE
                                            --> FILLER
"**"                                "Potential interaction site:
                                    provide-~1-j-value[s] =
                                                    x?[s];"

>;
```

```
                                                "** ~1 process "

"**" ~6-BUFFER~3;
"**" ~6-ITEM~3;

     ~1-empty-buffer: --> ~6-BUFFER;

     ~1-cycle : ~6-BUFFER --> ~6-BUFFER;
     ~1-cycle[b] =
        ~1-possible-consumption[
                ~1-possible-production[b]];

        ~1-possible-production:
                ~6-BUFFER --> ~6-BUFFER;
        ~1-possible-production[b] =

                /full[b]: b,

                 True :
~1-append-item-if-any[(b,accept-item-if-any)]
                /;


                ~1-full:
                        ~6-BUFFER --> BOOLEAN;


                ~1-accept-item-if-any:
                        --> ~6-ITEM | FILLER
"**"            "Potential interaction site:
                        ~1-accept-item-if-any[s] = x?[s];"

                ~1-append-item-if-any:

                        ~6-BUFFER *
                                (~6-ITEM | FILLER)
                                        --> ~6-BUFFER;


        ~1-possible-consumption:
                ~6-BUFFER --> ~6-BUFFER;
        ~1-possible-consumption[b] =
                /empty[b]: b,

                 True : ~1-possible-buffer-restore

                     [(b,~1-offer-consumable-item

                          [~1-highest-priority-item[b]])]

                /;
```

```
                ~1-empty:
                        ~6-BUFFER --> BOOLEAN;


                ~1-highest-priority-item:
                        ~6-BUFFER --> ~6-ITEM;

                ~1-offer-consumable-item:
                        ~6-ITEM --> FILLER | ~6-ITEM;
"***"           "Potential interaction site:
                        ~1-offer-consumable-item[s] = x?[s];"

                ~1-possible-buffer-restore:

                        ~6-BUFFER *
                                (FILLER | ~6-ITEM)
                                        --> ~6-BUFFER;

                ~1-possible-buffer-restore[(b,i)] =

                        /null[i]: b,

                         True : ~1-restore-to-buffer[(b,i)]

                        /;


                ~1-restore-to-buffer:
                        ~6-BUFFER *
                                ~6-ITEM
                                    --> ~6-BUFFER;
```

```
                                                "** ~1 process "

~6-STATE =      ~6-SWITCH *
                ~6-TIMING-INFO *
                ~6-VALIDITY-INFO *
                ~6-STANDARDS *
                ~6-HISTORY;

"**: ~6-SWITCH = { On, Off };
"**" ~6-STATE~3;
"**" ~6-STANDARDS~3;
"**" ~6-HISTORY~3;
"**" ~6-VALIDITY-INFO~3;
"**" ~6-TIMING-INFO~3;

     initial-~1-state:  --> ~6-STATE;
     initial-~1-state =
       (initial-~1-switch,
       initial-~1-timing-info,
       initial-~1-standards,
       initial-~1-history
       );

       initial-~1-switch: --> ~6-SWITCH;

       initial-~1-timing-info: --> ~6-TIMING-INFO;

       initial-~1-standards: --> ~6-STANDARDS;

       initial-~1-history: --> ~6-HISTORY;

~1-cycle:     ~6-STATE --> ~6-STATE;
~1-cycle[(s,t,d,h)] =
       (new-~1-switch[s],
       new-~1-timing-info[t],
       new-~1-standards[d],
       ~1-function[(s,t,d,h)],
       );

       new-~1-switch: ~6-SWITCH --> ~6-SWITCH;

       new-~1-timing-info: ~6-TIMING-INFO --> ~6-TIMING-INFO;

       new-~1-standards: ~6-STANDARDS --> ~6-STANDARDS;

       ~1-function:
               ~6-SWITCH *
               ~6-TIMING-INFO *
               ~6-STANDARDS *
               ~6-HISTORY
       --> ~6-HISTORY;
       ~1-function[(s,t,d,h)] =
               /equal[(s,Off)]:  h,
```

```
                 True          : ~1-process-readings[(d,h,
                                    [~1-obtain-readings[
                                            ~1-delay[t]])]
                /;

                ~1-delay:
                        ~6-TIMING-INFO --> FILLER;

                ~6-READING-LIST =
"**"                    j#1..1~2
                                < * ~6-READING >;

                obtain-~1-readings:
                        FILLER --> ~6-READING-LIST;
                obtain-~1-readings[null] =
"**"                    (j#1..1~2
                                < , obtain-~1-j-reading >);

"**"                    j#1..1~2
                        < ; obtain-~1-j-reading:
                                --> ~6-READING
                        >;
"**"                  "Potential Interaction Site:
                        obtain-~1-j-reading[s] = x?[s];"

                process-~1-readings:
                        ~6-STANDARDS *
                        ~6-HISTORY *
                        ~6-READINGS
                --> ~6-HISTORY;
                process-~1-readings[(s,h,r)] =
                        proj[(1,(update-~1-history[(h,r)],
                            provide-~1-feedback
                        [check-for-~1-condition[(s,h,r)]]
                                ) )];

                        update-~1-history:
                                ~6-HISTORY *
                                ~6-READINGS
                                        --> ~6-HISTORY;

                        check-for-condition:
                                        ~6-STANDARDS *
                                ~6-HISTORY *
                                        ~6-READINGS

                        --> ~6-RESPONSE-LIST;


                                ~6-RESPONSE-LIST =
                                        j#1..1~2
"**"                                    < * (~6-FEEDBACK |
                                        ~6-WARNING | FILLER)

>;
                        provide-~1-feedback:
```

```
                                ~6-RESPONSE-LIST --> FILLER;

                        provide-~1-feedback[f] =
                                    proj[(1,(Null,

"**"                                j#1..1~2
                                    < ,
                                    provide-~1-j-feedback-if-any
                                              [proj[(1,f)]]

                                    >
                                           )
                                ) )];


"**"                    j#1..1~2
                        < ; provide-~1-j-feedback-if-any:
                             ~6-FEEDBACK | ~6-WARNING | FILLER
                                    --> FILLER;


                       provide-~1-j-feedback-if-any[f] =
                                    /null[f]: Null,
                                     True :
                                        provide-~1-j-feedback[f]
                                    /;


                                    provide-~1-j-feedback:
                                            ~6-FEEDBACK U
                                                    ~6-WARNING
                                            --> FILLER
"**"                                  "Potential interaction site:
                                provide-~1-j-feedback[s] = x?[s];"

                      >;
```

```
                                                "** ~1 process "

~6-STATE =      ~6-TIMING-INFO *
                ~6-CONFIGURATION;

"**" ~6-STATE~3;
"**" ~6-TIMING-INFO~3;
"**" ~6-CONFIGURATION~3;
"**" ~6-MODIFICATION~3;

     initial-~1-state:  --> ~6-STATE;
     initial-~1-state =
       (initial-~1-timing-info,
       initial-~1-configuration
       );

       initial-~1-timing-info:
               --> ~6-TIMING-INFO;

       initial-~1-configuration:
               --> ~6-CONFIGURATION;

~1-cycle: ~6-STATE --> ~6-STATE;
~1-cycle[(t,c)] =
       (new-~1-timing-info[t],
       ~1-function[(t,c)],
       );

       new-~1-timing-info:
               ~6-TIMING-INFO --> ~6-TIMING-INFO;

       ~1-function:
               ~6-TIMING-INFO *
               ~6-CONFIGURATION
       --> ~6-CONFIGURATION;
       ~1-function[(t,c)] =
               ~1-process-modification[
                       (c,obtain-~1-modification[t])];

       obtain-~1-modification:
               ~6-TIMING-INFO --> ~6-MODIFICATION;

       ~1-process-modification:
               ~6-CONFIGURATION *
               ~6-MODIFICATION
       --> ~6-CONFIGURATION;
       ~1-process-modification[(c,m)] =
               /~1-valid-modification[(c,m)]:

                  proj[(1,(next-~1-configuration[(c,m)],

                       ~1-respond-to-modifier[(c,m)],
```

```
                provide-~1-updates
                [generate-~1-updates[(c,m)]]
                ) )],

         True : refuse-modification[(c,m)]
        /;


        valid-modification:

                ~6-CONFIGURATION *
                        ~6-MODIFICATION
        --> BOOLEAN;

        next-~1-configuration:

                ~6-CONFIGURATION *
                        ~6-MODIFICATION

        --> ~6-CONFIGURATION


        ~1-respond-to-modifier:

                ~6-CONFIGURATION*
                 ~6-MODIFICATION
        --> FILLER;

        generate-~1-updates:

                ~6-CONFIGURATION *
                        ~6-MODIFICATION

        --> ~6-UPDATE-LIST;


"**"    ~6-UPDATE-LIST = j#1..1~2
                < , (~6-UPDATE | FILLER) >;


        provide-~1-updates:
                ~6-UPDATE-LIST --> FILLER;
        provide-~1-updates[u] =

                proj[(1,(Null,

"**"                    j#1..1~2
                < ,provide-~1-j-update-if-any
                        [proj[(1,u)]]
                           >

                    )
                    ) )];
```

```
"**"                    j#1..1~2
                        < ; provide-~1-j-update-if-any:

                                ~6-UPDATE | FILLER
                                                --> FILLER;


                           provide-j~1--update-if-any[u] =

                                /null[u]: Null,
                                 True : provide-~1-j-update[u]

                                /;


                                provide-~1-j-update:
                                        ~6-UPDATE --> FILLER

"**"            "Potential interaction site:
                        provide-~1-j-update[s] = x?[s];"

                        >;


        refuse-modification:
                ~6-CONFIGURATION *
                        ~6-MODIFICATION --> FILLER;
```

OUTPUT DEVICE TEMPLATE

```
                                                "** ~1 process "

"**" ~6-ITEM~3;

     ~1-cycle: FILLER --> FILLER;
     ~1-cycle[null] =
       ~1-output[get-~1-output];


       get-~1-output: --> ~6-ITEM;

"**"            "Potential interaction site:
                         get-~1-output[s] = x?[s];"

       ~1-output: ~6-ITEM --> FILLER;
```

```
                                                    "** ~1 process "

"**" ~6-ITEM~3;

     ~1-cycle: FILLER --> FILLER;
     ~1-cycle[null] =
       ~1-give-input[~1-input];


       ~1-input: --> ~6-ITEM;


       ~1-give-input: ~6-ITEM --> FILLER;
"**"            "Potential interaction site:
                        ~1-give-input[s] = x?[s];"
```

```
                                                "** ~1 process "

~6-STATE =
        ~6-SWITCH *
        ~6-TIMING-INFO *
        ~6-VALIDITY-INFO *
        ~6-CONVERSION-INFO *
        ~6-READING;

      ~6-SWITCH = { On, Off };
"**" ~6-STATE~3;
"**" ~6-TIMING-INFO~3;
"**" ~6-VALIDITY-INFO~3;
"**" ~6-CONVERSION-INFO~3;
"**" ~6-READING~3;
"**" ~6-SENSOR-VALUE~3;

      initial-~1-state:  --> ~6-STATE;
      initial-~1-state =
        (initial-~1-switch,
        initial-~1-timing-info,
        initial-~1-validity-info,
        initial-~1-conversion-info,
        initial-~1-reading,
        );

        initial-~1-switch: --> ~6-SWITCH;

        initial-~1-timing-info: --> ~6-TIMING-INFO;

        initial-~1-validity-info: --> ~6-VALIDITY-INFO;

        initial-~1-conversion-info: --> ~6-CONVERSION-INFO;

        initial-~1-reading: --> ~6-READING;

~1-cycle:      ~6-STATE --> ~6-STATE;
~1-cycle[(s,t,v,c,r,)] =
        (new-~1-switch[s],
        new-~1-timing-info[t],
        new-~1-validity-info[v],
        new-~1-conversion-info[c],
        ~1-function[(s,t,v,c,r)],
        );

        new-~1-switch: ~6-SWITCH --> ~6-SWITCH;

        new-~1-timing-info: ~6-TIMING-INFO --> ~6-TIMING-INFO;

        new-~1-validity-info: ~6-VALIDITY-INFO-->
                ~6-VALIDITY-INFO;

        new-~1-conversion-info: ~6-CONVERSION-INFO -->
```

```
                ~6-CONVERSION-INFO;

~1-function:
        ~6-SWITCH *
        ~6-TIMING-INFO *
        ~6-VALIDITY-INFO *
        ~6-CONVERSION-INFO *
        ~6-READING;
--> ~6-READING;
~1-function[(s,t,v,c,r)] =
        /equal[(s,Off)]:  r,
        True          : ~1-distribute-reading
                                [make-~1-reading[(t,v,c,r)]]
        /;

        distribute-~1-reading:  ~6-READING --> ~6-READING;
        distribute-~1-reading[r] =
                proj[(1,(new-~1-reading[r],
"**"                    (j#1..1~2
                           < ,provide-~1-reading-j[r] >)
                        ) )];

                new-~1-reading:
                        ~6-READING
                                --> ~6-READING;

"**"            j#1..1~2
                < ;  provide-~1-reading-j:
                        ~6-READING
                                --> FILLER >;
                "Potential interaction site:
                        provide-~1-reading-j[s] =
                                        x?[s];"

make-~1-reading:
        ~6-TIMING-INFO *
        ~6-VALIDITY-INFO *
        ~6-CONVERSION-INFO *
        ~6-READING;
--> ~6-READING;
make-~1-reading[(t,v,c,r)] =
        do-~1-conversion[(c,r,
                check-~1-validity[(v,
                        obtain-~1-value[
                                ~1-delay[t]])])];

        ~1-delay:  ~6-TIMING-INFO --> FILLER;

        obtain-~1-value:  FILLER --> ~6-SENSOR-VALUE;

        check-~1-validity:
                ~6-VALIDITY-INFO *
                        ~6-SENSOR-VALUE
        --> ~6-SENSOR-VALUE | FILLER;
        check-~1-validty[(v,s)] =
```

```
/valid-~1-value[(v,s)]: s,
True: proj[(1,(Null,
                ~1-invalidity-warning[(v,s)]
                ) )]
/;

valid-~1-value:
        ~6-VALIDITY-INFO *
        ~6-SENSOR-VALUE --> BOOLEAN;

~1-invalidity-warning:
        ~6-VALIDITY-INFO *
        ~6-SENSOR-VALUE --> FILLER;

do-~1-conversion:
        ~6-CONVERSION-INFO *
        ~6-READING*
        (~6-SENSOR-VALUE | FILLER)
--> ~6-READING;
```

```
                                                "** ~1 process "

~6-STATE =
        ~6-SWITCH *
        ~6-TIMING-INFO*
        ~6-REPORT-INFO *
        ~6-STORAGE-INFO *
        ~6-DATABASE;

"**" ~6-STATE~3;
"**" ~6-TIMING-INFO~3;
"**" ~6-REPORT-INFO~3;
"**" ~6-STORAGE-INFO~3;
"**" ~6-DATABASE~3;
"**" ~6-REPORT~3;

      initial-~1-state:  --> ~6-STATE;
      initial-~1-state =
        (initial-~1-switch,
        initial-~1-timing-info,
        initial-~1-report-info,
        initial-~1-storage-info,
        initial-~1-database

        );

        initial-~1-switch:
                                  --> ~6-SWITCH;

        initial-~1-timing-info:
                                  --> ~6-TIMING-INFO;

        initial-~1-report-info:
                                  --> ~6-REPORT-INFO;

        initial-~1-storage-info:
                                  --> ~6-STORAGE-INFO;

        initial-~1-database:
                                  --> ~6-DATABASE;


~1-cycle: ~6-STATE --> ~6-STATE;
~1-cycle[(s,t,r,g,d)] =
        (new-~1-switch[s],
        new-~1-timing-info[t],
        new-~1-report-info[r],
        new-~1-storage-info[g],
        ~1-function[(s,t,r,g,d)],
        );

        new-~1-switch:
                ~6-SWITCH --> ~6-SWITCH;
```

```
        new-~1-timing-info:
                ~6-TIMING-INFO --> ~6-TIMING-INFO;

        new-~1-report-info:
                ~6-REPORT-INFO --> ~6-REPORT-INFO;

        new-~1-storage-info:
                ~6-STORAGE-INFO --> ~6-STORAGE-INFO;

~1-function:
        ~6-SWITCH *
        ~6-TIMING-INFO *
        ~6-REPORT-INFO *
        ~6-STORAGE-INFO*
        ~6-DATABASE

 --> ~6-DATABASE;

~1-function[(s,t,r,g,d)] =
        /equal[(s,Off)]:  d,
           True       : ~1-process-request
                                [(r,g,d,obtain-~1-request[t])]
                /;


        obtain-~1-request:
                ~6-TIMING-INFO
                        --> ~6-REQUEST | FILLER;

        process-~1-request:

                ~6-REPORT-INFO *
                ~6-STORAGE-INFO *
                ~6-DATABASE *
                (~6-REQUEST | FILLER)

        --> ~6-DATABASE;

        process-~1-request[(r,g,d,q)] =
                /Null[q]: maintain-~1-database[d],

                 True : proj[(1,
                        (update-~1-database[(g,d,q)],
                           give-~1-reports
                                [generate-~1-reports[(r,d,q)]]
                        ) )]
                /;


                maintain-~1-database:
                        ~6-DATABASE --> ~6-DATABASE;

                update-~1-database:
```

```
                        ~6-STORAGE-INFO *
                        ~6-DATABASE *
                        ~6-REQUEST
                --> ~6-DATABASE;

                generate-~1-reports:

                        ~6-REPORT-INFO *
                        ~6-DATABASE *
                        ~6-REQUEST
                --> ~6-REPORT-LIST;

"**"
~6-REPORT-LIST = j#1..1~2
        < * (~6-REPORT | FILLER) >;

give-~1-reports:
        ~6-REPORT-LIST --> FILLER;
give-~1-reports[r] =
        proj[(1,(Null,
"**"            (j#1..1~2
                    < ,give-j-~1-report-if-any[proj[(1,r)]] >)
                ) )];


"**"    j#1..1~2

        < ; give-j-~1-report-if-any:
                ~6-REPORT | FILLER --> FILLER;
        give-j-~1-report-if-any[r] =
                /Null[r]: Null,

                 True : give-~1-j-report[r]

                /;


                give-~1-j-report:
                        ~6-REPORT --> FILLER
"**"            "Potential interaction site:
                        give-~1-j-report[s] = x?[s];"
        <;
```

BLANK TEMPLATE

" ~1 process "

PyGraph User's Guide

Table of Contents

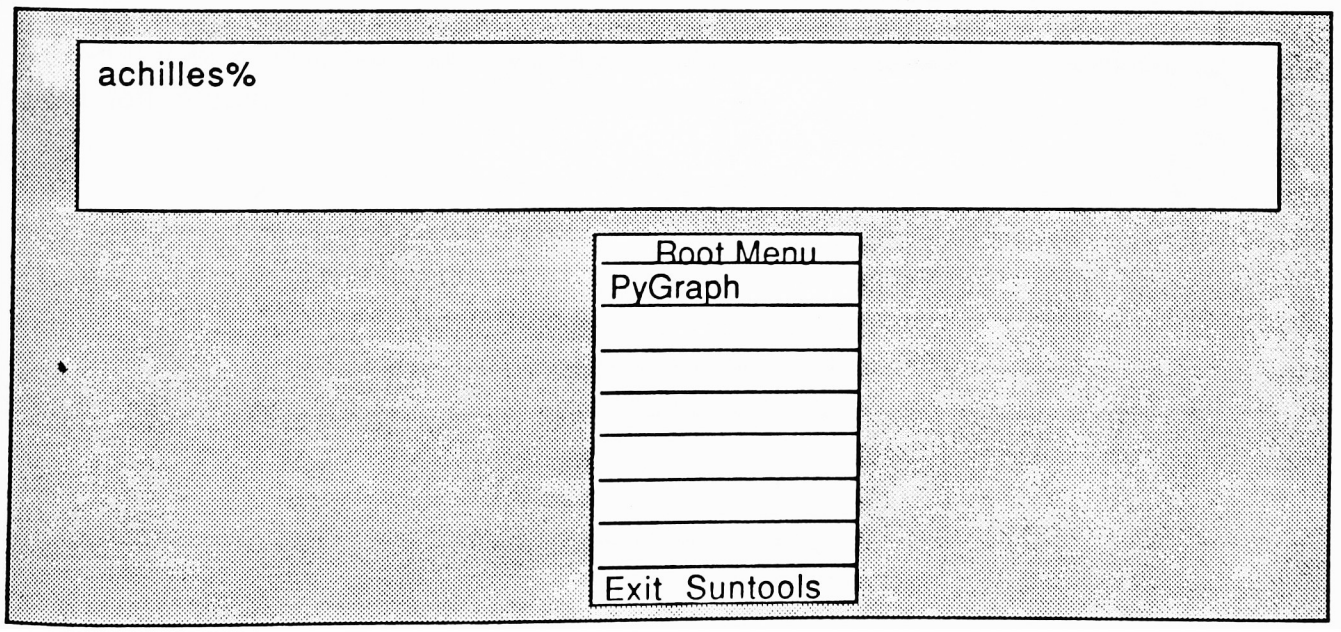
Topic	Page
1. Accessing Suntools	2
2. Accessing PyGraph	3
3. How to Use PyGraph - Getting Started	4
4. Graphic Development - Manipulating Symbols	5
5. Graphic Development - Administration	7
6. Viewing/Modifying the Textual Specification	9
7. PyGraph Special Features	11
I. Movement within the Specification	11
II. Posting of Interactions between Symbols	12
III. Showing the Other End of Channel	14
8. Current PyGraph Constraints and Limitations	16

1. Accessing Suntools

Sunview (Sun Visual/Integrated Environment for Workstations) is an environment for the Sun workstation (Release 3.2) which supports interactive, graphics based applications. It is one of an array of packages under the general name of Suntools. In order to use PyGraph it is necessary to first enter Suntools. This can be accomplished in one of two ways: 1) At the UNIX command prompt type **Suntools** or 2) Place the following command sequence into your .login file.

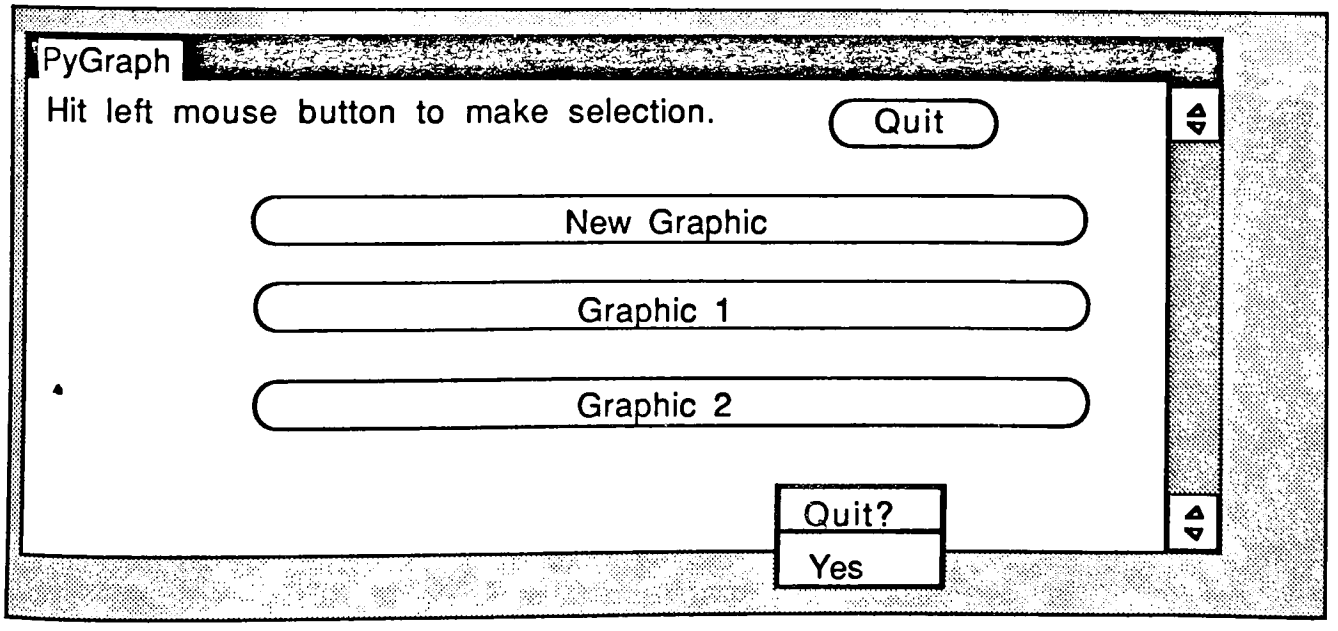
```
if ("`tty`" != "/dev/console") exit
echo -n "Suntools? (^C to interrupt)
sleep 5
suntools
```

Alternative 2 above gives you 5 seconds to abort entry into Suntools by typing ^C after you see Suntools?. After 5 seconds, you will see the top window of the following.



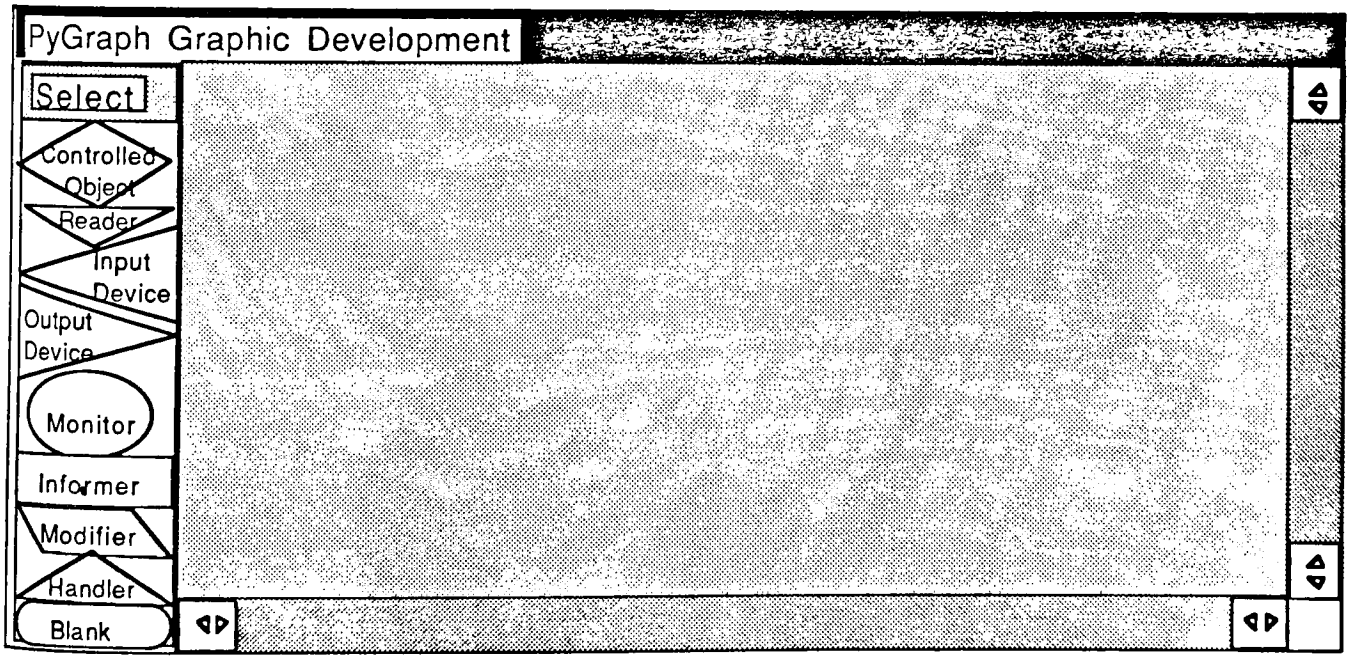
2. Accessing PyGraph

Your UNIX command prompt is shown at the top-left. Several icons may be present at the top-right of the screen for other Sun application tools. PyGraph can be accessed in one of two ways: 1) Type the command **PyGraph** at the command line (be sure to place the cursor inside the command window) or 2) Depress the right mouse button to obtain the "Root Menu" (be sure the cursor is residing outside the command window). Selecting the PyGraph option brings up the first PyGraph window, called the PyGraph Directory. An example view of this is shown below. If it is desired to terminate the PyGraph session at this time, a menu can be obtained by depressing the right mouse button while the cursor is located in the PyGraph Directory window. Quit can also be obtained by clicking the left mouse button on the Quit button.



3. How to Use PyGraph - Getting Started

The PyGraph Directory displays a button for each of the available (previously generated) graphics. Note the scrollbar to the right of the directory. If there are more buttons than can fit on the visible directory window, the standard mouse buttons can be used to scroll the view to the desired location. Also available is a button called "New Graphic" for the generation of a completely new graphic. If it is desired to access one of these, it should be activated by clicking the left mouse button while pointing to it with the cursor. The button will then turn to inverse gray video indicating that it is selected. PyGraph will then proceed to the Graphic Development Environment for the chosen graphic. An example of the environment you will see follows.



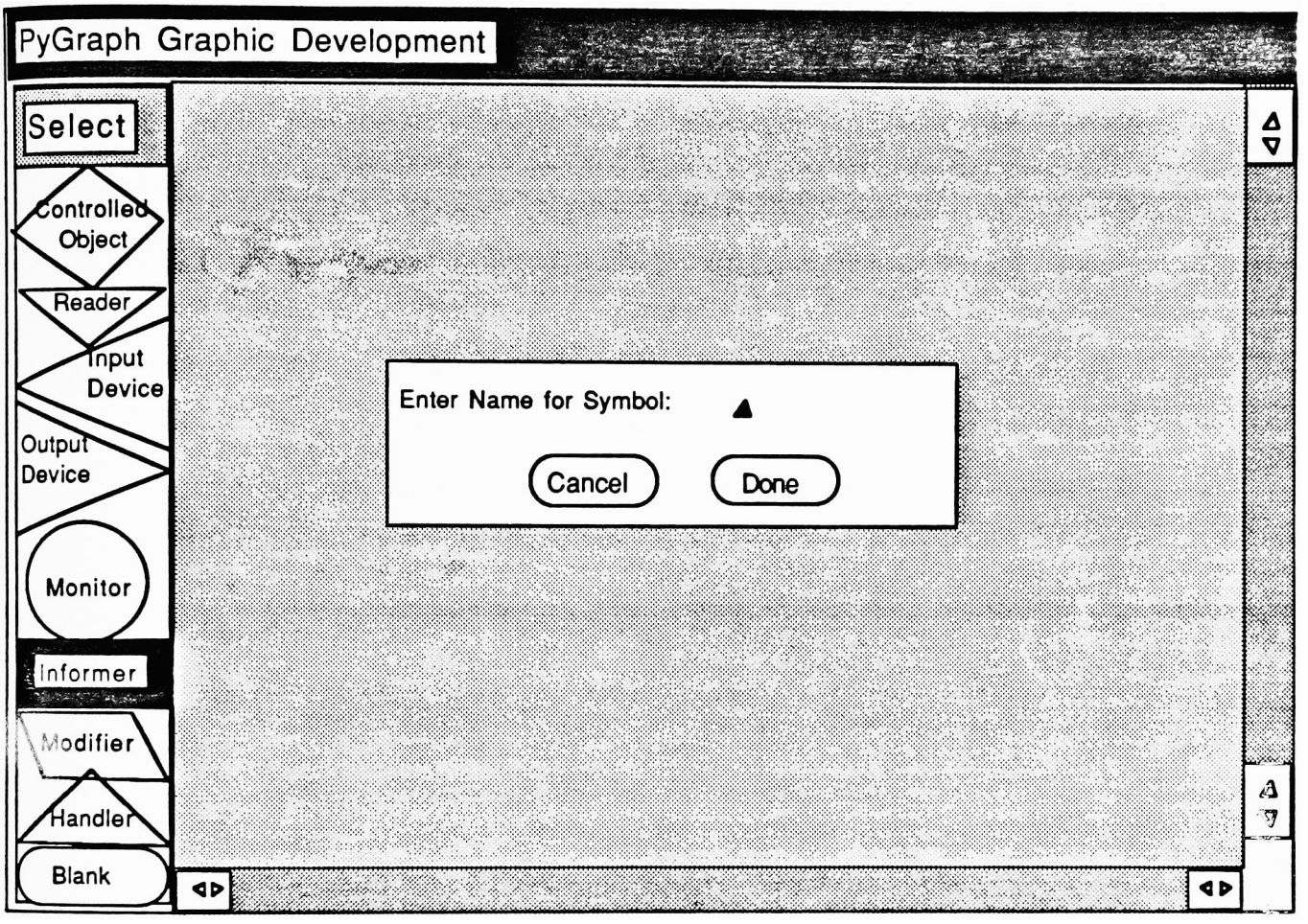
4. Graphic Development - Manipulating Symbols

The Graphic Development Environment consists of a symbol panel and a drawing canvas. The symbol panel is a palette consisting of all the available symbols which can be posted onto the drawing canvas in specifying your application. To post one of the symbols onto the canvas, place the cursor over the symbol and click the left mouse button. The symbol will turn to inverse video. Next, move the cursor to the drawing canvas and click the left mouse button at the location that you desire to post the symbol. A popup will appear asking for the input of a symbol name. This name must be unique among all of the symbols on the graphic. After entering the name, click mouse left on the "Done" button. The popup will disappear, the symbol and symbol name are posted on the canvas, and graphic development can continue.

Once a symbol has been placed on the drawing canvas, it can be moved as desired. To move a symbol it must first be activated by clicking mouse left over it. Once activated, it can be moved freely on the canvas by depressing and holding down the middle mouse button.

The default size of the drawing canvas is larger than the terminal screen. To move the viewport to a different location on the canvas two scrollbars are provided along the right and bottom of the canvas. The middle mouse button can be clicked within the scrollbar for this purpose.

A picture of major elements described above follows.


PyGraph Graphic Development
Select
Controlled Object
Reader
Input Device
Output Device
Monitor
Informer
Modifier
Handler
Blank
Enter Name for Symbol:
Cancel
Done

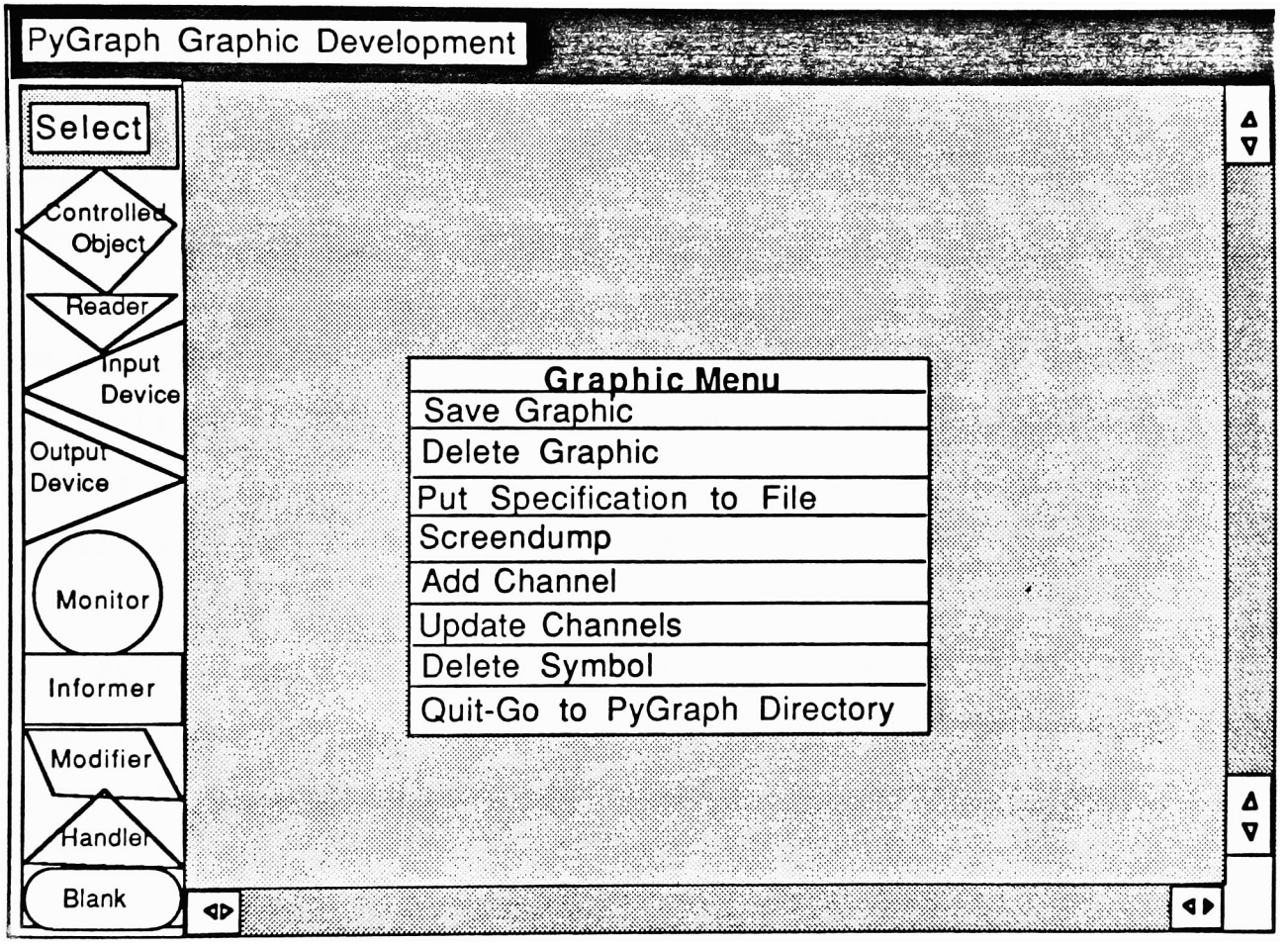
5. Graphic Development - Administration

At any time during graphic development, a graphic menu can be called up that will enable various administrative actions. It is obtained by depressing the right menu button while the cursor is in the drawing canvas or symbol panel. Actions that can be taken from the menu include:

1. Save graphic - Saves the graphic to the PyGraph Directory Panel and returns the user to that Directory for further action.
2. Delete graphic - Deletes the graphic and returns the user to the Directory for further action.
3. Put specification to file - Will cause a copy of the complete textual specification to be saved into a file of the user's choice. A popup will appear when you choose this option requesting entry of a filename.
3. Screen Dump - A screen dump is generated and sent to a file of the user's choice.
4. Add Channel - A feature that simplifies the process of posting channels into specifications and lines on the graphic. (See section 7.II.)
5. Update Channels - Scans the entire specification for channels and draws the corresponding lines on the graphic.
6. Delete Symbol - Deletes the inverted symbol provided that there are no channels drawn to it.
7. Quit/Return to PyGraph Directory - Prompts the user

as to whether changes should be saved, then returns the user to the PyGraph Directory Panel.

An example screen showing the key elements follows.



6. Viewing/Modifying the Textual Specification

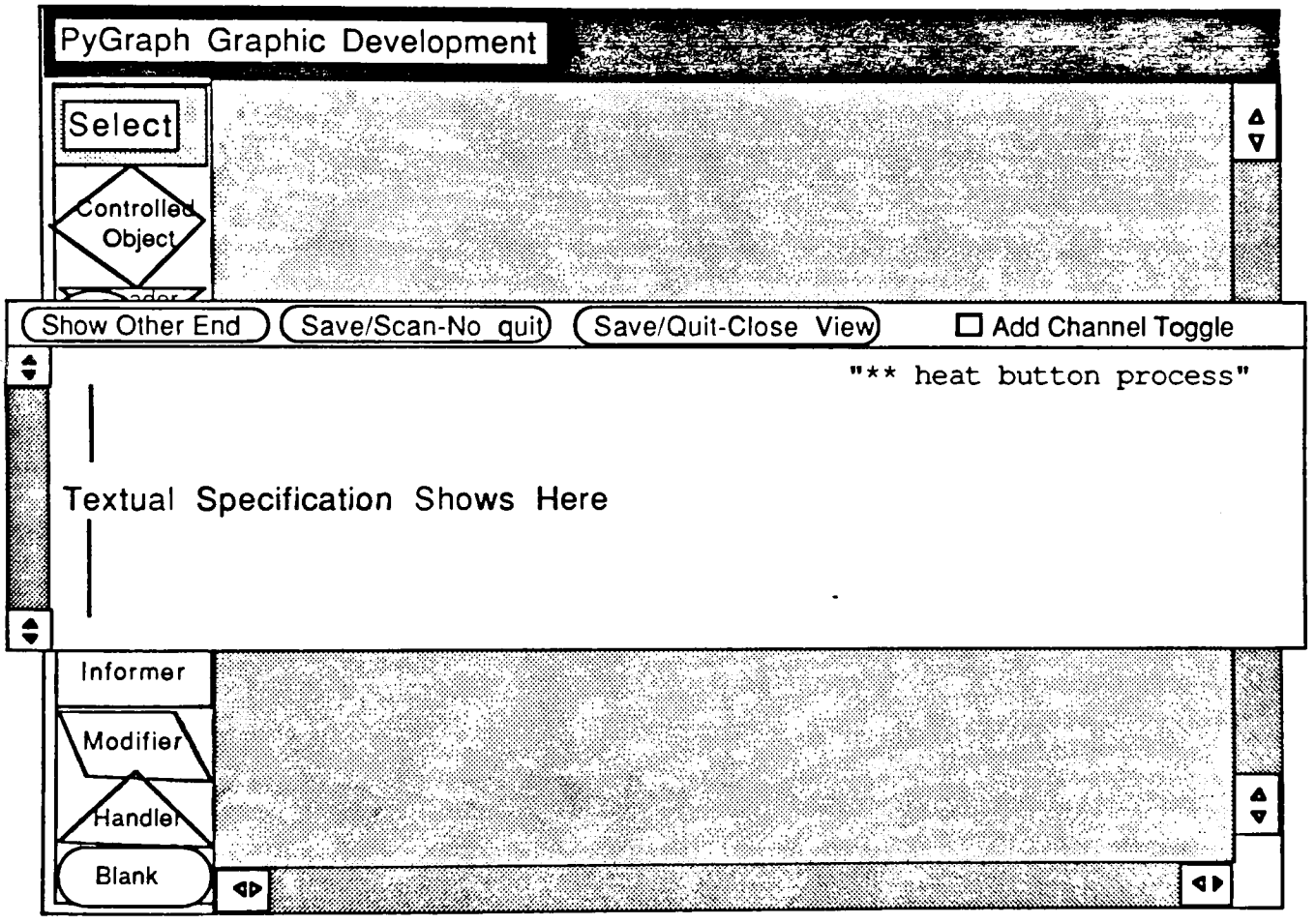
Upon completion of graphic development, or at any time after the first symbol has been posted onto the canvas, the textual specification for each symbol can be viewed and modified. The specification for a symbol is obtained by first activating the symbol by clicking the left mouse button while the cursor is positioned over the symbol (shown by inverse video). Clicking the left mouse button again while holding the shift key down provides a new window showing the specification. It contains the textual specification which can be modified with all of the standard Sun Workstation editing features. As the specification is being developed, channels will often be added to model interaction between processes. There are two ways that PyGraph facilitates the development of channel interactions.

1. PyGraph provides a feature that structures and simplifies the process of posting channels onto the specifications of interacting processes. This feature is described in section 7.II.

2. If the user elects to add channels directly to the specification, PyGraph can tabulate these additions and post the corresponding lines on the drawing canvas. The tabulation process is invoked from the specification window by pressing the button titled "Save - No Quit". Here the currently open specifications are saved and the

specifications for all of the symbols in the graphic are scanned for channels. Ultimately, the drawing canvas is updated per the scanning process.

The key elements of the above process are shown below.



7. PyGraph Special Features

There. are three features of PyGraph that need further description. These are 1) Movement within the textual specification, 2) Posting of interactions between symbols, and 3) Showing the Other End of Channels.

I. Movement within the specification

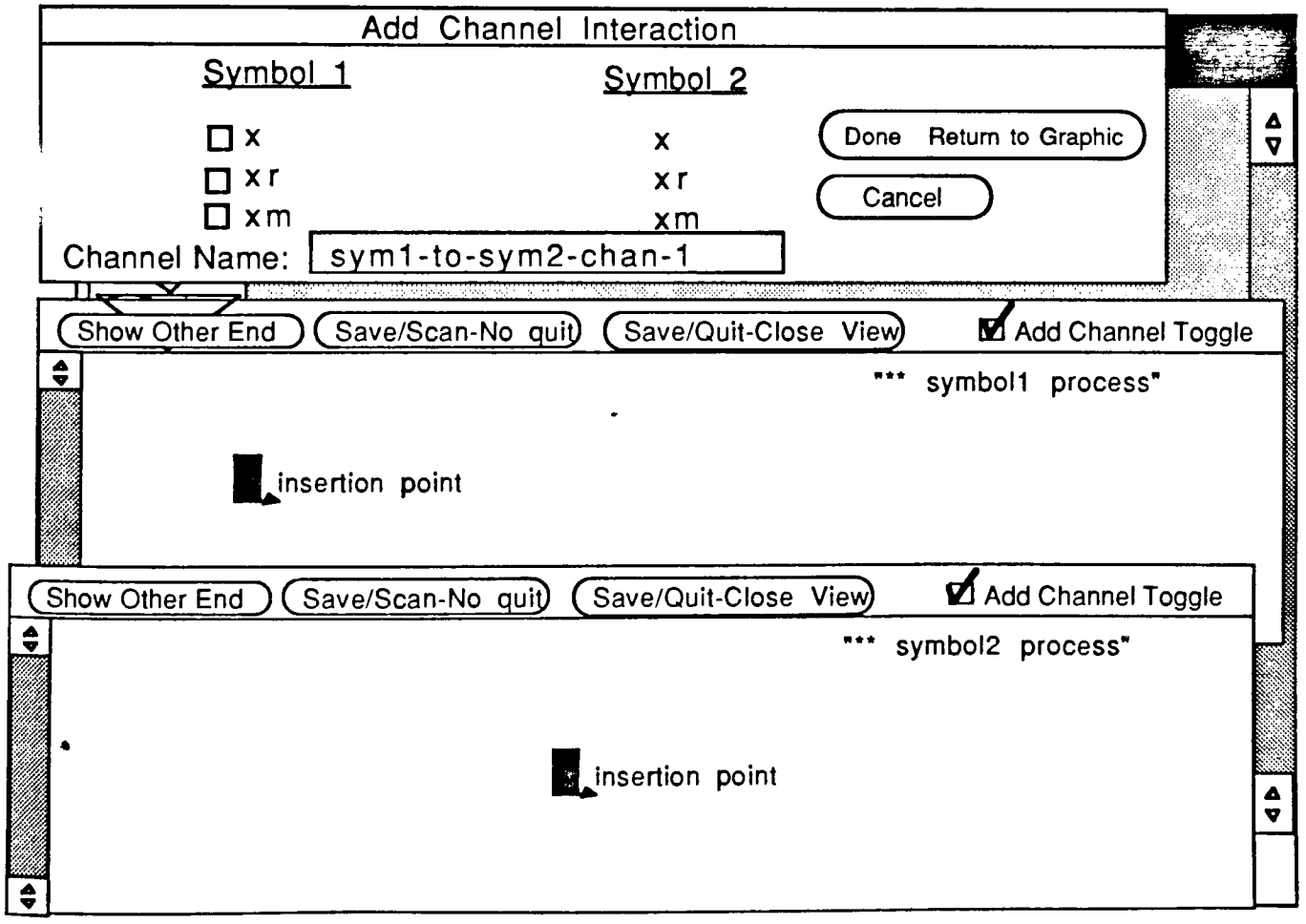
Designed into the PyGraph templates is an accelerator mechanism that enables the user to move quickly to items that have been predetermined by PyGraph. Each one of these lines has a "**" next to it. The "Find" mechanism built into the Sun Workstation (lower left corner of the Sun 3 keyboard) can be used to quickly progress to these sites. The find capability can also be used to trace function calls. This is accomplished by pressing the left mouse button on the first letter of the function name, then the middle mouse button on the last letter of the function name, thus highlighted the name. Pressing the find key will move the viewport to the next occurrence of that name. For further information on this and many other innate capabilities, refer to the user documentation on the Sun textedit subwindows.

II. Posting of Interactions between Symbols

Ultimately, the user will need to identify interactions between symbols on the graphic. This implies two tangible outcomes: 1) PyGraph must place a line on the graphic between the two symbols and 2) Exchange functions over channels must be posted at specific locations in the specification for each symbol. A feature has been built into PyGraph which automates most of this task. The following steps are involved:

1) Open a view into the textual specification for both of the symbols in which it is desired to post interactions (same functionality as viewing and modifying a specification described in section 6).
2) On each of the two specifications, place an insertion point (click mouse left to place a small caret below and to the right or left of a character or space) where you want to post the channel.
3) Place a check mark in the "Add Channel Toggle" at the upper right corner of the specification subwindow. This is done by pointing to the box and clicking the left mouse button. The same operation can be used to remove an existing check mark.
4) Depress the right mouse button while the cursor is on the Graphic Development Canvas to obtain the Graphic Menu (shown in section 5).
5) Choose the Graphic Menu option to Add Channel. A panel labeled Add Channel Interaction will appear

for input in the upper left hand corner of the terminal. On this panel, indicate type of exchange at each symbol. Additionally, if desired, the default channel name can be modified by the user. Finally, the user can press the left mouse button on the Done button to accept the input This will post the channel name at the insertion point in each specification and draw the corresponding line on the graphic.

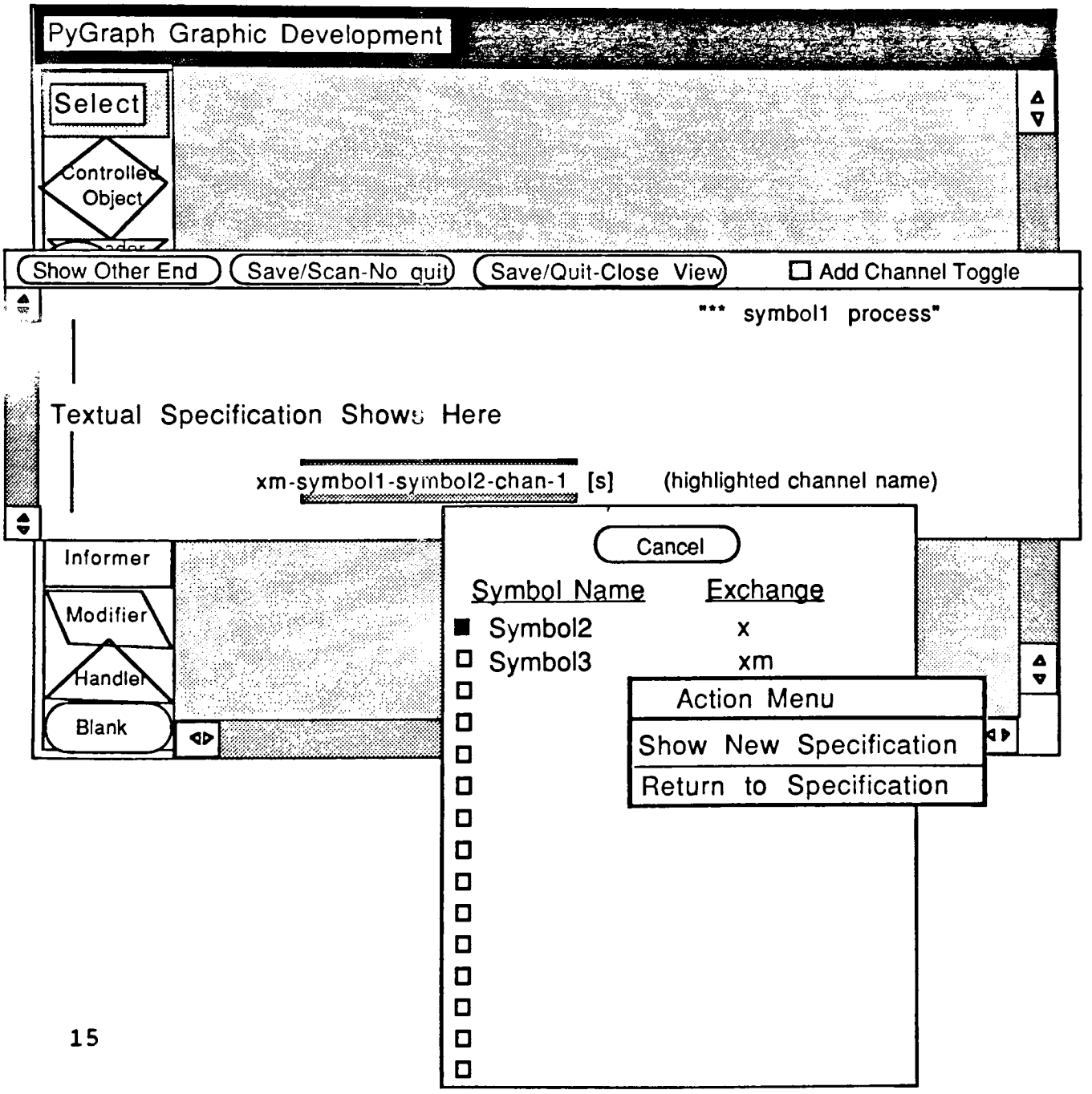


III. Show Other End

Once channels have been established on the graphic, another feature exists to quickly determine where (which symbol) other endpoints of a channel exist and what type of interaction is occurring. This feature is accessed with the following process:

1) Enter a specification for a symbol using the standard process described in section 6.
2) Highlight a channel name which it is desired to see the other end. This done by clicking left mouse on the left end of the channel name and clicking middle mouse button on the right end of the name (channel name in the specification must be followed by a space or a left square bracket).
3) Click the button at the top of the specification window titled "Show Other End".
4) A list of interactions at the other end of the channel will appear in a popup, if any exist.
5) To proceed to the specification for one of the choices listed, obtain the action menu by depressing the right mouse button while the cursor is inside the choice panel.
6) The second option on the menu is titled "Show Other End". Choosing that action menu option will cause another specification window to appear that contains the other end of the channel that was desired.

7) The other menu option is that of returning to the specification. This option returns the user back to the Specification window from where the request started. A diagram including the key aspects of the process follows.



8. Current PyGraph Constraints and Limitations

1) Two specification text subwindows may be open at a time. Once two are open, PyGraph will not allow any more to be opened.

2) Thirty graphics are allowed in the directory at a time.

3) The drawing canvas is constrained to 1000 X 1000 pixels.

4) A channel can have at most 15 "other ends".