STAR NOTE 161
A GRAPHICAL USER INTERFACE FOR TAS
Preliminary Design

Claude A. Pruneau
Physics Department
Wayne State University, Detroit, MI 48201, USA

ABSTRACT

This document summarizes the features desired for the STAR Graphical Interface to the analysis shell TAS. It also suggests a specific implementation using the CERN KUIP/MOTIF builder tools.

1. Introduction

Friendly graphical user interfaces (GUI) are increasingly popular nowadays. The high energy physics/nuclear physics community makes no exception to this recent fad. Witness the very successful development of the PAW++ [1] data analysis program with a graphical interface or the quite useful documentation browser NCSA-Mosaic[3].

In view of the complexity of the STAR analysis software and considering the size of STAR analysis undertaking, the desire for a good organizational and visual tool for data reduction and analysis appears quite natural. STAR physicists and software engineers have already spent a great deal of time designing and building a large body of analysis and monte Carlo software. This effort is only starting and we foresee that the STAR software will grow both in complexity and flexibility. So far, the approach has been to reduce the work in small, well defined components, attacking very specific tasks or problems. In such a context, the analysis of STAR data becomes a matter of choosing the right analysis components and organize them in logical sequence. Also, as the STAR collaboration and simulation effort is distributed across the world, the data reduction, detector calibration and physics analysis will be a highly distributed and complex process. Finally, the collaboration has more or less adopted the analysis shell prototype TAS as a starting point for its future analysis shell.

TAS has quite interesting and unique features such as the capability to turn ON and OFF certain analysis modules, to examine and change the content of data objects (called tables), to book and specify the filling of histograms and n-tuples interactively in addition to the full spectrum of PAW features. In addition, one of the planned extension of the shell is to allow multiple and programmable analysis paths at the shell level rather than the code level. This will allow to execute different analysis sequences depending on the type of event being analyzed or given some other user criteria. It should be possible to treat different types of events such as "normal trigger", "empty event" trigger or calibration events during the same analysis session. Also, one might consider giving a special treatment to some events having some interesting pathology and perform some specific routine calls for these events only. It would therefore being interesting to include in the shell some tools to define multiple analysis paths and specify the calling conditions of these paths. A graphical/symbolic tool would be best as it would allow a simple and quick definition and organization of the calling sequence. The design of such tools is underway but is not included in this document.

Currently, the interaction with the shell is done by means of a single command line user interface based on the the CERN KUIP[2]. Although quite powerful, and flexible, such an interface has been criticized as somewhat difficult to learn for novices and "non-specialists". The user has to be familiar with the overall structure of the analysis package and there is not really any features that guide the user through an analysis Session. Although not intrinsically difficult, the features of the program are numerous and a non frequent user may become lost with the syntax of all the commands. In fact, the desire for a friendlier analysis tool has been expressed repeatedly by many members of the STAR collaboration. Some have expressed the desire for an analysis shell which is "self explanatory", that is simple to use without the guidance of a specialist and yet includes all features one has become accustomed to. It is the purpose of this document to establish the basis of such a "user friendly" TAS graphical user interface.

As an event processing program, the graphical interface must include tools to select input and output event data streams, commands to select, load or Save geometry, calibration and other calibration constants, commands to select/specify the operation and data reduction performed on these events, commands to control which and how many of the events will be processed. And,As an analysis environment, the shell should include commands and tools to visualize the various data objects, i.e. TAS tables, TAS Modules, TAS histogram definitions and graphics. The shell should also include tools to store the commands issued by the user, and or the Error messages produced by the various TAS components and analysis Modules. Many of these features already exist in a primitive form in TAS. We suggest that these features be enhanced and be packaged in a graphical form which not only simplifies their apprenticeship but speed up the debugging and facilitate an analysis Session.

The implementation of the GUI for TAS may become a major task. Given the limited resources available (man power and money) among the collaboration, and given the size of the proposed task, we suggest to proceed to the implementation of a graphical interface in progressive steps first making use of tools and simple GUI builders already available on the market. Also,As to minimize our task and maximize the friendliness of the interface, it would be best to use known and well accepted graphics standards. We therefore propose to base the development of the graphical interface on a X-Motif implementation of the KUIP GUI builder. In the future we may expend the interface using other interface builders(need references).

In this document, we summarize what we believe are the most important features that one should implement. The features are discussed separately according to their functionality. We describe and discuss their implementation in two steps. First, we list and describe the features Associated with a generic function. Then, in a second part, we propose a specific organization and implementation of the features listed in the first section.

The x-motif implementation is described somewhat informally using usual widgets and gadgets concepts and names. No attempt was made to follow the official X-motif nomenclature. We apologize to the Reader if this may introduce any confusion...

As this document is a preliminary design effort of the TAS graphical user interface, we welcome all comments and constructive criticism.

2. Interface Specifications and X-Motif Implementation Definition.

2.0 Summary

The graphical interface should include the following features that we list here without explanations or justifications. They are grouped by functionality and described in more details in later sections. Note that a user need not activate all the windows and features. In all cases however, the executive and event processing window should be present and give access by means of a command-line to all usual TAS features.

1. *Executive or Event Processing Window*
2. *Browsing Window*
3. *Error and Message Logging Window*
4. *Error and Message Summary Window*
5. *Command Transcript and History Window*
6. *Event Processing Status Window*
7. *Module View Window*
8. *Table View Window*
9. *Variable View Window*
10. *Histogram/N-tuple Definition View Window*
11. *Histogram/N-tuple Booking Window*
12. *Graphics Window*
13. *Dialog Windows*

2.1 Executive or Event Processing Window

The Executive/Event Processing window is the main window of the graphical interface version of TAS. It is meant primarily to be used for event process control, to select event input and output streams, to load or Save event input and output calibration and analysis constants, to start, pause, or stop the event processing. However,As TAS is also an analysis tool, the executive window can be used in a manner similar to the PAW++ analysis program as the controlling center of an analysis Session. In addition, the executive window should contain Menus to provide access to the secondary or additional features by Opening special windows.

We list and describe the desired features in section 2.1.1. We suggest an x-motif implementation in section 2.1.2.

2.1.1 Executive/Event Processing Window Specifications

The Executive and Event Processing Window should include the following functionality and features:

1. *Session Management and Control*
2. *Selection of View Windows and other User Activated Features*
3. *Selection and Display of Input and Output Event Data streams*

4. *Selection and Display of Input and Output Static Data Files and TAS Tables*
5. *Event Processing Control*
6. *Standard Line-Command Input*
7. *Session Transcript*
8. *Selection of Options (Fonts, Colors, Other Setup Specifications, etc)*
9. *Help*

Each of these features are described in details in the following.

2.1.1.1 Session Management and Control

The shell must include commands to suspend, terminate or Kill the Session. It should also include commands to inform the user about some of the TAS and subcomponents version number and authors.

1. *About TAS*
   Opens an Info window that lists the TAS and subcomponents version number and author-ships.
2. *Pause/Session Suspension*
   The suspension of the Session implies that the shell windows are temporarily iconized and stored away on the user's screen. Event processing in progress is not interrupted.
3. *Session Termination*
   Termination means Exiting the Session normally and gracefully by first closing all Open files and freeing allocated memory. Optionally, event processing in progress could continue and proceed as a detached process.
4. *Session Killing*
   Killing the Session should be possible if unexpected shell behavior is encountered by the user or if the shell or some of its components fail to respond.

These options should be available by means of line-commands, MENU options and possibly through window Buttons. We propose to refer to them as Session/ABOUT, Session/Iconify, Session/Exit and Session/Kill respectively.

2.1.1.2 Selection of View Windows and other User Activated Features

The TAS graphical interface once completed will include many options. All these need not to be activated at all time and different users depending on their purposes may so desire to activate or deactivate some of these options and features. In addition, it should be feasible to activate multiple copies of some of these windows. Among obvious features, we list:

1. *Browsing Window*
   The Browsing window consists of graphical TAS/paw command command browser. It is described in more details in section 2.2.
2. *Error/Message Logging View Window*
   The Error/Message Window is used to display Error messages issued by the analysis Modules. The window include many features described in section 2.3.
3. *Error/Message Summary Window*
   The Error/Message Summary Window is used to display a summary of the Error messages issued by the analysis routines for user diagnostic. It is described in section 2.4.

4. *Command Transcript and Summary Window*
   The Command Transcript and Summary Window Window is used to display a transcript of the line and gui commands issued by the user. It is described in section 2.5.

5. *Module View Window*
   The Module View Window is used to display the current status of user analysis Modules. It is described in details in section 2.6.
6. *Table View Window*
   The Table View Window is used to display the current status of the analysis tables. It is described in section 2.7.
7. *Variable View Window*
   The Variable View Window is used to display the current content of TAS tables/variables. It is described in section 2.8.
8. *Histogram/N-tuple Definition View Window*
   The Histogram/N-tuple Definition View Window is used to display current definitions of TAS histograms and N-tuples. It is described in detail in section 2.9.
9. *Processing Status View*
   The Processing Status Window is used to display the status of event processing. It identifies the I/O Streams, the source of static tables, the number of events to analyze, the number currently analyzed, the volume of data Read, etc... It is described in detail in section 2.10.

These options should be accessible by means of line-commands and Menu options. We propose to refer to them as View/Processing, View/Browser, View/Error, View/Summary, View/Module, View/Table, View/Variable, View/Histo. In addition, one could also include a Panel option similar to that used in PAW++.

2.1.1.3 Selection of Input and Output Event Data streams

The User may analyze or more generally process events from various sources (disk files, tape, network) and of various types (raw data, pre-analyzed, dst, physics etc). The result of this process may be any reduced data set in the form of TAS tables. These can be stored in principle to any media in user selected format. In addition, in the case of Monte Carlo simulations there may not be necessarily a real input from TAS tables. The interface must therefore include the following features:

1. *Open Input Stream*
   The User must select the source (medium, file name and format) of the events to be processed (analyzed) or specify, using the next option, that Faked events are used. One could envision the possibility to Open multiple input streams simultaneously. In such context, one would have to include commands to select the active input stream and de-select other not in use. The same applies for output stream. For simplicity, we will such options in this preliminary design report. We will therefore assumes that the Opening of an input (output) stream while one is already Opened results in the closing of the one currently Opened.

2. *Fake Input Stream*
   The User may use a Module which actually generates Monte Carlo data or Reads data from a user file in a non standard way. In this case, some of the normal actions of the shell are skipped.
3. *ReOpen Input Stream*
   This option is used to restart the analysis at the beginning of an input stream file and essentially consists in rewind of the medium or file.
4. *Close Input Stream*
   The User may explicitly Close an input stream prior to other subsequent actions.
5. *Open Output Stream*
   The User may select TAS tables for output after the processing of each events. In addition, the User must specify the medium, file name and format for output.
6. *Close Output Stream*
   The User may explicitly Close an output stream prior to other subsequent actions.
7. *Suspend Event Output*
   Once an output stream has been defined, the User may suspend temporarily the table output after the processing of each events.
8. *Resume Event Output*
   Once the output of table output has been suspended, it should be possible to resume the output.

These options should be accessible by means of line-commands and MENU options. We propose to refer to these as Event-io/Open_Input, Event-io/Fake_Input, Event-io/Reopen, Event-io/Close_Input, Event-io/Open_Output, Event-io/Close_Output, Event-io/Suspend_Output, Event-io/Resume_Output.

2.1.1.4 Selection of Input and Output Static Data Files and TAS tables

The processing of events requires external geometry constants, calibration constants and various other parameters stored in various forms (among them TAS tables). The shell includes functions to Read from and write to file these "static" data. Internally, the data are stored as TAS tables. Externally, they could be stored in many forms. The user should have the ability to select among the possible many forms defined and accepted by the collaboration. The interface must therefore included the following features:

1. *Load Static Table*
   The tables are loaded from a User selected file according to the user selected format.
2. *ReLoad Static Table*
   The tables are reloaded from the User selected file according to the user selected format.

3. *Save Static Table*
   An updated version of Selected tables is Saved to a preselected and Opened file. This operation defaults to Save to Save Static Table As... if no output file or tables have been selected by the user.
4. *Save Static Table As...*
   The User selects static tables and a file name for output. The Use also selects the output format. The shell Opens the file and Save the selected tables.

5. *Reset Static Table*
   The User Selects one or few or all static tables to be reset to some User's preset value.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them as Static-io/Load_Static, Static-io/Reload_Static, Static-io/Save_Static, Static-io/Save_Static... and Static-io/Reset_Static.

2.1.1.5 Event Processing Control

The User may start, pause or stop the event processing. He may also specify how many events are to be processed and what type of events are to be processed (this particular feature needs further design). In addition, the shell should display on screen the status of the event processing, the number of events to be analyzed, the number of events skipped, the number of events actually analyzed etc. Also, the shell should display the name of the input and output event streams and the source (file name) of the static tables (and their names) currently loaded. The shell should include the following commands and features.

1. *Go All*
   The Go All command request the processing of all events in the currently selected input stream.
2. *Go...*
   With the Go... command, the user must specify how many events and the event type of the currently selected input stream must be analyzed.
3. *Pause*
   The Pause command suspends momentarily the processing of events. All Counters values are Saved.
4. *Resume*
   The Resume command restart the processing of events once it has been suspended with the Pause command.
5. *Stop*
   The Stop command interrupts the processing of events. Counter values may be altered.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them as Process/Go all, Process/Go..., Process/Pause, Process/Resume, and Process/Stop.

2.1.1.6 Standard Line-Command Input and Session Transcript

The shell must include a conventional line-command interface to the event processor and analysis programs. This could be achieved in a manner similar to that used in PAW++. The Executive window should therefore include an input field and an output field or transcript pad. The transcript pad displays both the user commands and the output resulting from these commands. Both commands typed explicitly in the input pad and commands selected by means of Menu or Buttons should be displayed (as well as their results) in the transcript pad. The commands issued from the command line could be differentiated from those issued through menus by displaying them with different Colors or with different Fonts or by pre-appending some symbol.

2.1.1.7 Help

The Executive window should have a help Menu option which gives access to help and documentation on the various features of TAS, KUIP, PAW commands, the executive window and the other windows.

2.1.2 Executive/Event Processing Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last. The proposed Menu-bar item names are given in italic.

2.1.2.1 Window Title: *TAS Executive Window*

2.1.2.1 Menus

2.1.2.1.1 Session Control: *File*

1. *About KxTerm*
   Opens an Info Window and Displays version of x-terminal software used.
2. *About TAS*
   Opens an Info Window and Displays version/authors of current version of TAS and major subcomponents.
3. *Iconify*
   Pause the Interactive Session, Iconify all TAS windows and stored them away on User's screen. Does not stop event processing in progress if any.
4. *Exit*
   Opens Alarm Dialog requesting authorization from user to proceed with Exiting procedure. If authorization is granted by the user, Detach event processing process as stand-alone process if any, Close necessary files, Close windows, deallocate memory, Exit program.
5. *Kill*
   Opens Alarm Dialog requesting authorization from user to proceed with Exiting procedure. If authorization is granted by the user, Terminate event processing, Close all files, Close all windows, deallocate memory, Exit program.
6. *Save Transcript As...*
7. *Save Setup*
8. *Save Setup As...*
9. *Load Setup*

2.1.2.1.2 Event Stream: *Event I/O*

1. *Open Input Stream...*
   Opens an "Open Input Stream Dialog" window and upon successful completion of the dialog, Open new Input Stream. Close previously Opened Input Stream, Enable ReOpen Input Stream and Close Input Stream commands. Update display of input stream name.

2. *Fake Input Stream*
   Requests the bypass of event Reading by the event processor. Close previously Opened Input Stream if any, Update display of Input Stream as "FAKE INPUT".
3. *Close Input Stream*
   Close currently Opened input stream. Update display of Input Stream As "NONE".
4. *ReOpen Input Stream*
   Rewind Input Stream.
5. *Open Output Stream...*
   Opens an "Open Output Stream Dialog" window and upon successful completion of the dialog, Open Output Stream. Close previously Opened Output stream if any. Update display of output stream name. Enable Close Output Stream command.
6. *Close Output Stream*
   Close currently Opened output stream, suspend output of event. (dynamic) tables.
7. *Suspend Event Output*
   Suspend output of event (dynamic) tables. Update displayed event process status As "Event Output Suspended".
8. *Resume Event Output*
   Resume Output of event tables. Update displayed event process status As "Event Output Selected".

2.1.2.1.3 Static Tables I/O: *Tables*

1. *Load Static Table...*
   Opens a "Load Static Table Dialog" window and upon successful completion of the dialog, Load the Static table from user selected file.
2. *Save Static Table*
   Save the user selected tables to the user selected file. If no file or tables are selected, defaults to Save Static Table As....
3. *Save Static Table As...*
   Opens a "Save Static Table Dialog" window and upon successful completion of the dialog, Save the user selected Static tables to the user selected file.
4. *Reset Static Table...*
   Opens a "Reset Static Table Dialog" window and upon successful completion of the dialog, Reset the user selected Static tables.

2.1.2.1.4 Event Processing Control: *Process*

1. *Go All*
   Starts Event Processing of all events in the current input stream. Enables the Pause and Stop commands. The GO commands are enabled only if valid analysis Modules are selected and if an invent stream has been selected by the user (or the fake event option has been selected), Opened successfully and has not reached the end of file.
2. *Go...*
   Opens a "Go..." Dialog window and upon successful completion of the dialog, Starts Processing of the user selected number of events and event type from the current input stream. Enables the Pause and Stop commands.

3. *Pause*
   Suspend Processing of events. This command is disabled after it has been selected. It is enabled by the Go commands and re-enabled by the Resume command. It enables the Resume command.
4. *Resume*
   Resume Processing of events. By default this command is not enabled. It is enabled when the user suspends the event processing with the Pause command.
5. *Stop*
   Interrupts the processing of events. The command is enabled by the Go commands and disables itself upon selection.

2.1.2.1.5 View: *View*

1. *Processing Status*
   Open a Processing Status View window.
2. *Browser*
   Open a Browser window.
3. *Error Logging*
   Open a Error Logging window.
4. *Error Summary*
   Open a Error Summary window.
5. *Module View*
   Open a Module View window.
6. *Table View*
   Open a Table View window.
7. *Variable View*
   Open a Variable View window.
8. *Histo View*
   Open a Histo View window.

2.1.2.1.5 Help: *Help*

1. *On xterm*
   Open an edit/Read Only Window showing help on xterm handling.
2. *On TAS*
   Open an edit/Read Only Window showing help on TAS.
3. *On TAS Resources*
   Open an edit/Read Only Window showing help on TAS resources.
4. *On KUIP Resources*
   Open an edit/Read Only Window showing help on KUIP Resources.
5. *On Browser*
   Open an edit/Read Only Window showing help on the Browser Window.
6. *On Panel*
   Open an edit/Read Only Window showing help on the Panel Window.
7. *On Error Logging View*
   Open an edit/Read Only Window showing help on Error Logging Window.

8. *On Error Summary View*
   Open an edit/Read Only Window showing help on Error Summary Window.
9. *On Module View*
   Open an edit/Read Only Window showing help on Module View Window.
10. *On Table View*
    Open an edit/Read Only Window showing help on Table View Window.
11. *On Variable View*
    Open an edit/Read Only Window showing help on Variable View Window.
12. *On Histo View*
    Open an edit/Read Only Window showing help on Histo View Window.
13. *On KUIP Functions*
    Open an edit/Read Only Window showing help on KUIP Functions.

2.1.2.2 Buttons

1. *Iconify*
   Same action as Session/Iconify command.
2. *Exit*
   Same action as Session/Exit command.
3. *Kill*
   Same action as Session/Kill command.

2.1.2.3 Window Layout

Figure 1 displays the suggested layout of the executive/event processing window.

2.2 Browsing Window

We suggest to use an implementation identical to that used in PAW++. We therefore refer the Reader to PAW++ for the definition of this window.

2.3 Error Message Logging Window

We list and describe the desired features in section 2.3.1. We suggest an x-motif implementation in section 2.3.2.

2.3.1 Error Message Logging Window Specifications

The shell should include commands to select where Error message from analysis Modules are directed and the level of severity reported.

The Error Message Logging Window should include the following functionality and features:

1. *Selection of output file*
2. *Message Severity and Type Selection*
3. *Help*
4. *Error Message Display*

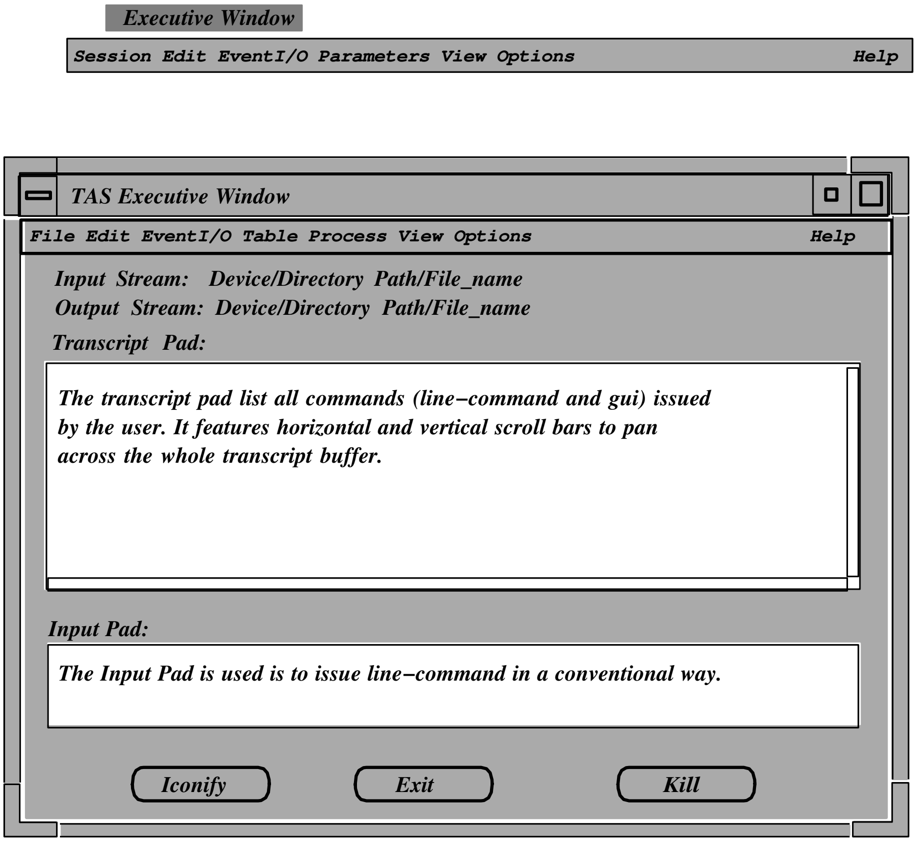

Fig. 1: Executive/Event Processing Window

2.3.1.1 Selection of output file

By default, the Error message are output in the transcript pad of the executive window and to the Error logging window buffer. The user has a choice to redirect the output only to either windows or to Save the Error message logging buffer to a user selected file. The interface must therefore included the following features:

1. *Save Error Message Log As...*
   The User must specify a file name under which the Error log buffer will be Saved.
2. *Save Error Message Log*
   The User may Save/update the Error log buffer to file. This commands default to the Save Error Message Log As... if no output file has been Opened.
3. *Suspend Error Message Logging*
   Suspend Error message logging.
4. *Resume Error Message Logging*
   Resume Error message logging once it has been interrupted with the suspend command.
5. *Close Error Message Log*
   Close the currently Opened Error log file and suspend Error logging.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them as Error Log/Save As..., Error Log/Save, Error Log/Suspend, Error Log/Resume and Error Log/Close.

2.3.1.2 Selection of Severity Level

By default, all Error messages are written to the Error logging buffer. The user has a choice to filter the messages written to the buffer and Saved to file.

2.3.1.3 Help

An Help command should be included to document the features of this window.

2.3.2 Error Message Logging Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.3.2.1 Window Title: *Error Message Log View*

2.3.2.1 Menus

2.3.2.1.1 File

1. *Save Error Log*
   Save/Update current buffer to Error Log File. If no file is Opened, execute Save Error Log As... command.
2. *Save Error Log As*
   Opens an "Save Error Log As Dialog" window and upon successful completion of the dialog, Open the Output file, See Description below.
3. *Suspend Error Logging*
   Suspend Error Logging. Disable "Suspend Error Logging" menu item. Enable "Resume Error Logging" menu item.
4. *Resume Error Logging*
   Resume Error Logging. Disable "Resume Error Logging" menu item. Enable "Suspend Error Logging" menu item.
5. *Close Error Log*
   Close Error log file. Disable "Suspend Error Logging" and "Resume Error Logging" items.

2.3.2.1.2 Severity: *Severity*

1. *Items are to be defined*

2.3.2.1.3 Options: *Options*

1. *Items are to be defined*

2.3.2.1.4 Help: *Help*

1. *Items are to be defined*

2.3.2.2 Buttons

1. *Iconify* Iconify the Error Message Log View window and store away icon on user's screen.
2. *Close* Close the Error Message Log View window.

2.3.2.3 Window Layout

Figure 2 displays the suggested layout of the Error logging window.

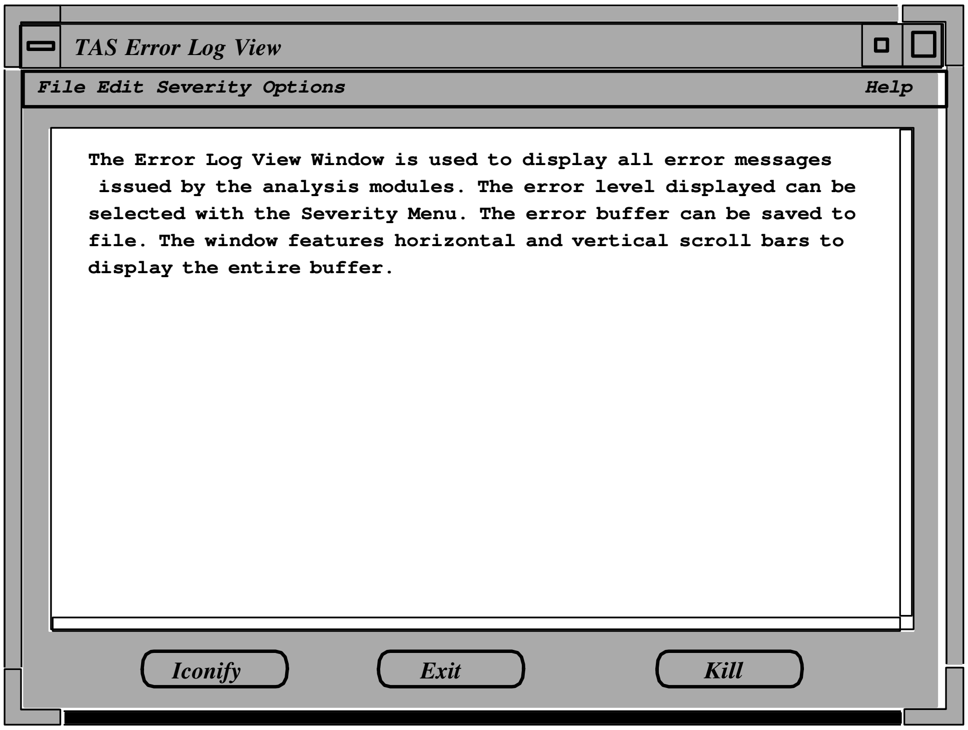

Fig. 2: Error Logging View Window

2.4 Error Message Summary Window

The Error message report package keeps track of the number of occurrences of all messages and can be prompted for a summary of all messages and their number of occurrences. The current version of TAS produces a journal file that lists this summary. IN the graphical Interface Version of TAS, we propose to include an Error Summary Window to display this summary and allow the user to Save the summary or parts of it to a user selected file.

We list and describe the desired features in section 2.4.1. We suggest an x-motif implementation in section 2.4.2.

2.4.1 Error Message Summary Window Specifications

The shell should include commands to display, Save the Error summary, and to select the level of severity reported.

The Error Summary Window should include the following functionality and features:

1. *Selection of output file*
2. *Type and Severity Selection*
3. *Help*
4. *Error Message Summary Display*

2.4.1.1 Selection of output file

By default, the Error Summary is shown in the Summary window and is not Saved to file after an analysis Session. The user may however request the shell to Open a file and Save the Error Summary for future consideration. Doing so, the user should also select the level and type of Error reported. One should include the following features:

1. *Save Error Summary As...*

   The User must specify a file name under which the Error Summary will be Saved.

2. *Save Error Summary*
   The User may Save/update the Error Summary to file. This commands default to the Save Error Summary As if no output file has been Opened.
3. *Close Error Summary*
   Close the currently Opened Error Summary file and Close the Error Summary window.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them as Summary/Save As..., Summary/Save, Summary/Close.

2.4.1.2 Selection of Severity Level

By default, all Error messages are reported in the Summary. The user has a choice to filter the messages displayed and Saved to file.

2.4.1.3 Help

An Help command should be included to document the features of this window.

2.4.2 Error Message Logging Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.4.2.0 Window Title: *Error Message Summary View*

2.4.2.1 Menus

2.4.2.1.1 File Selection: *File*

1. *Save Error Summary*
   Save/Update current buffer to Error Summary File. If no file is Opened, execute Save Error Summary As... command.
2. *Save Error Summary As...*
   Opens an "Save Error Summary As..." Dialog and upon successful completion of the dialog, Open the Output file and Save the Error Summary, See Description below.
3. *Close Error Summary*
   Close Error Summary file and window.

2.4.2.1.2 Severity

1. *Items are to be defined*

2.4.2.1.3 Help: *Help*

1. *Items are to be defined*

2.4.2.2 Buttons

1. *Iconify*
2. *Close*

2.4.2.3 Window Layout

The Figure 3 displays the suggested layout of the Error Summary View window.

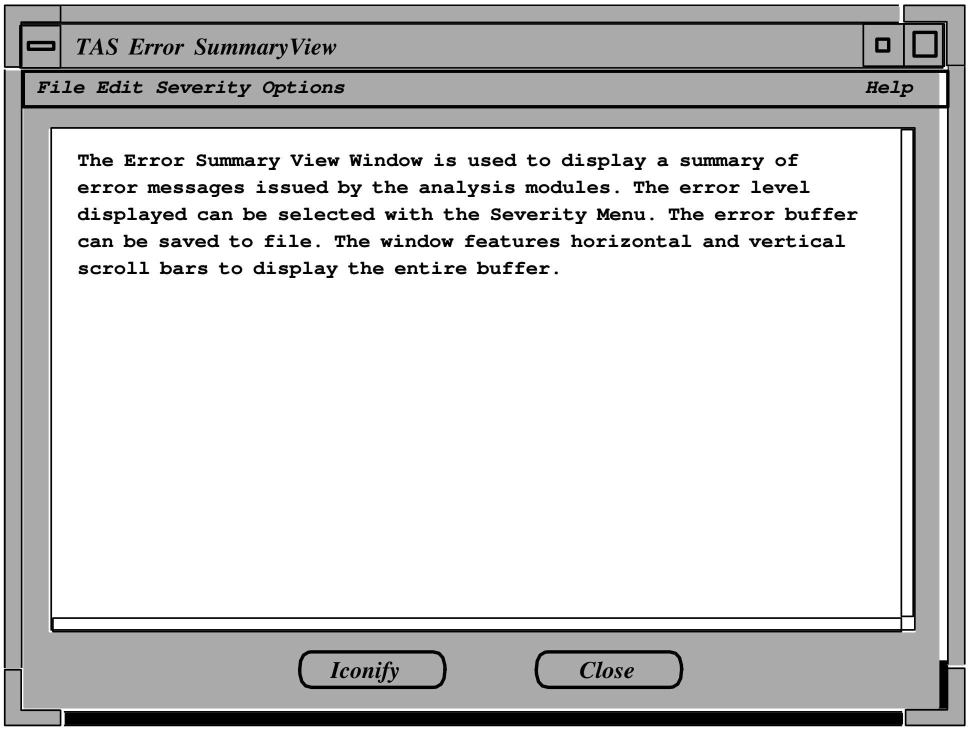

Fig. 3: Error Summary View Window

2.5 Command Transcript and History Window

We list and describe the desired features in section 2.5.1. We suggest an x-motif implementation in section 2.5.2.

2.5.1 Command Transcript and History Window Specifications

It is useful to record all commands issued by the user as explicit commands on the command line or as GUI commands through Menus or Buttons. The command History should be directed to an History buffer Save-able upon user's request. The user should be able to select the file name under which the buffer is Saved. He should be able to suspend/resume the History logging. He should also be able to edit/cut complete or parts of commands from the History buffer. The Command Transcript and History Window should include the following functionality and features:

1. *Selection of a new buffer*
2. *Selection of an output file to Save current History buffer*
3. *Close current History buffer and file.*
4. *Selection of font types, Colors and sizes used to display text in the History window.*

2.5.1.1 History File Management

1. *New History Buffer*
   The current History buffer is deactivated, and a new History buffer is Opened.
2. *Open History...*
   An old History file is Opened into a separate History buffer and window.

3. *Save History*
   An updated version of the History buffer is Saved to a preselected and Opened file. This command defaults to the "Save History As..." command if no file output file is Opened for current buffer.
4. *Save History As...*
   The User selects a file name for output and the shell Saves the current History buffer.
5. *Reset History*
   The History buffer is emptied.
6. *Close History*
   The current History buffer/file is Closed and the History window is Closed.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them as History/New, History/Open, History/Save, History/Save As..., History/Reset, History/Close.

2.5.1.2 History View Options

1. *Fonts*
   The user is presented with a series of font options he has to select from. The selection should include a "default" option which allows a quick re-establishment of the default options.
2. *Colors*
   The user is presented with a series of color selections to customize the History window. The selection should include a "default" option which allows a quick re-establishment of the default options.

These options should be accessible by means of line-commands and MENU options. We propose to refer to them As History/Fonts, History/Colors.

2.5.2 Executive/Event Processing Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.5.2.1 Window Title: *History View*

2.5.2.1 Menus

2.5.2.1.1 File: *File*

1. *New History*
   Open a new History buffer.
2. *Save History*
   Save the History buffer to file. If no file has been Opened, execute Save History As command.
3. *Save History As...*
   Opens an "Save History As..." Dialog and upon successful completion of the dialog, Open History file, and Save History buffer.

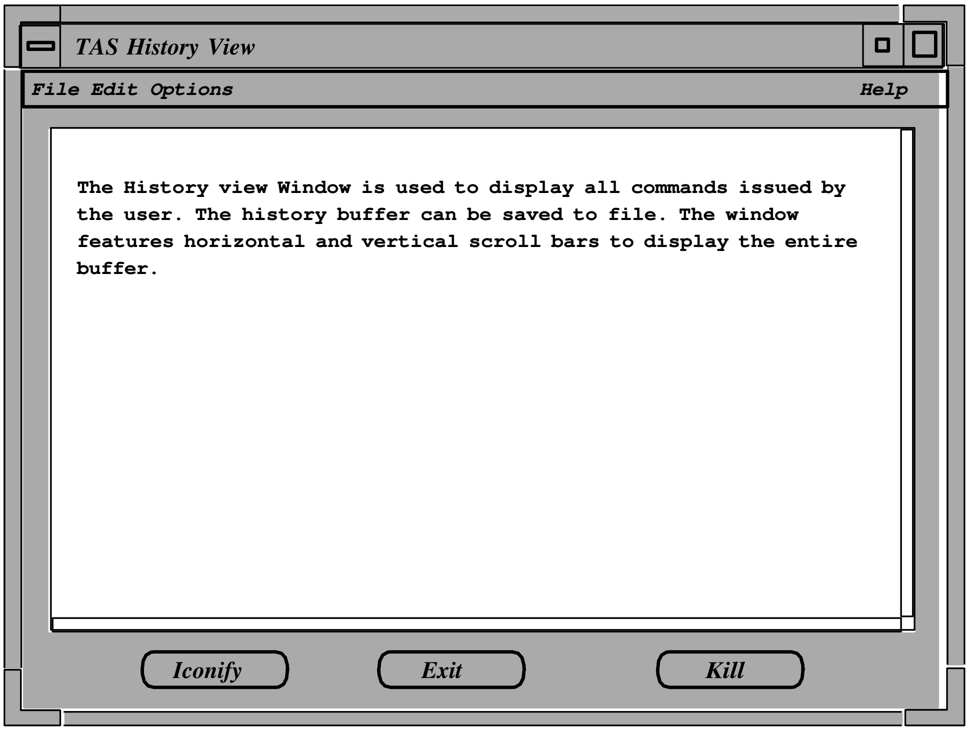

Fig. 4: History View Window

4. *Close History*
   Close currently Opened History file and window.
5. *Reset History*
   Empty History buffer.

2.5.2.1.1 Window settings: *Options*

1. *Fonts*
   Open a font selection window.
2. *Colors*
   Open a color selection window.

2.5.2.2 Buttons

1. *Iconify*
   Same action as History/Iconify command.
2. *Exit*
   Same action as History/Close command.
3. *Cancel*
   Same action as History/Close command.

2.1.2.3 Window Layout

The Figure 4 displays the suggested layout of the History view window.

2.6 Module View Command

We list and describe the desired features in section 2.6.1. We suggest an x-motif implementation in section 2.6.2.

2.6.1 Module View Window Specifications

TAS include a certain number of commands to view, select, activate or deactivate analysis Modules. These commands should remain available for line-commands. It would be useful to add a Module View Window in which the status of all analysis Modules is displayed. There should be a brief mode and a verbose mode. In the brief mode, only the name, the activation status, the number of arguments, the called status are shown. In the verbose mode, the Module's arguments, their type (static or dynamic) and their scope (Read, write, update) are listed as well. In addition, it should be possible to activate or deactivate a Module by selecting its name with the mouse, and pressing the activate or deactivate buttons located in the same window. An alternative method of activation/deactivation would be to "double click" with the mouse on the name of the Modules. Also, it should be possible to select a large number of Module with the mouse and activate or deactivate them collectively with the activate/deactivate buttons.

1. *Selection of activation state (on/off) of all analysis Modules.*
2. *Selection of an output file to Save current History buffer*
3. *Close current History buffer and file.*
4. *Selection of font types, sizes and Colors used to display the Module status.*

2.6.1.1 Module Activation

The user should be able to use a menu item to turn all Modules ON or OFF at the same time. In addition, he should be able to activate or deactivate one or few Modules using the mouse to select their name in the view field and "click" on deactivate/activate buttons located in the "Module View" window. We propose to use Module/ON and Module/OFF command names.

2.6.1.2 File management

The user should be able to store away to an ASCII file the current status displayed by the window. There should be commands to Open a previously Saved file, Open/create a new file, and commands to update the status Saved in a file. Analogously to other windows defined, we suggest to use the following commands: Module/Open, Module/Close, Module/Save, Module/Save As..., and Module/Clone.

2.6.1.3 Options selection

The user should be able to select the font type, size and Colors used in the various fields of the "View/Module" window.

2.6.2 Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.6.2.1 Window Title: *Module View*

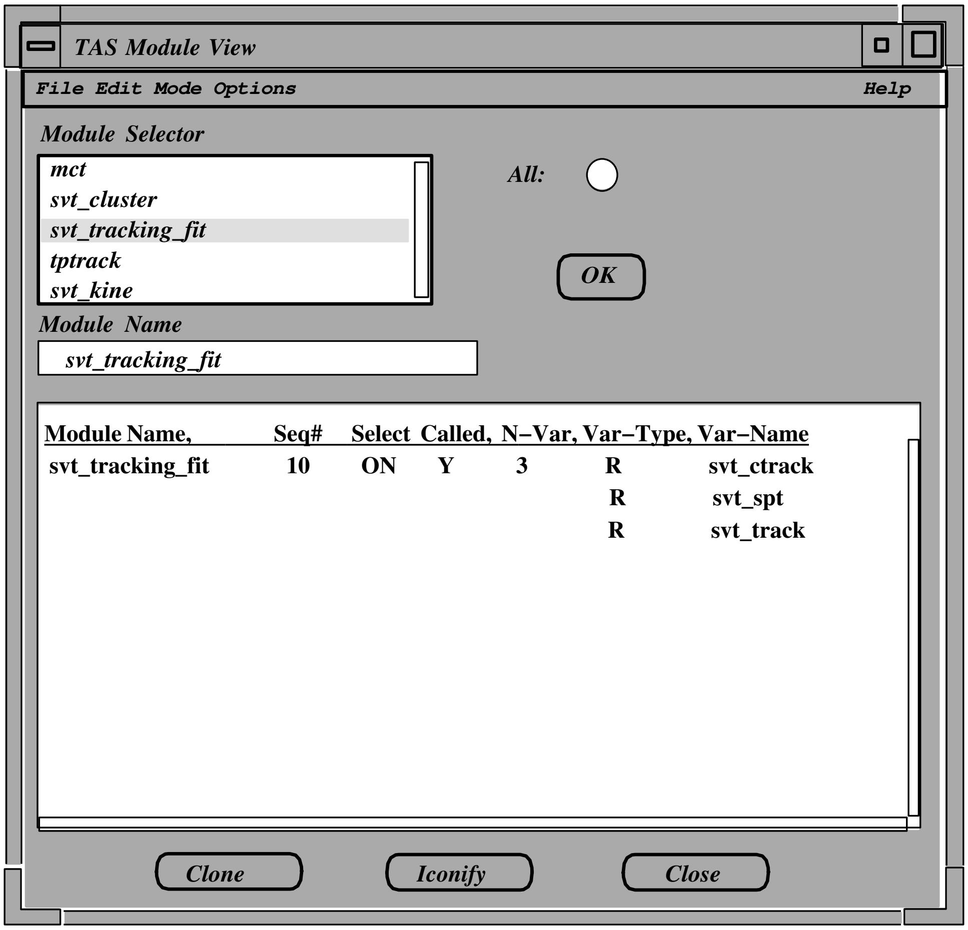

Fig. 5: Module View Window

2.6.2.1 Menus

2.6.2.1.1 File Management: *File*

1. *Open*
2. *Save*
3. *Save As...*
4. *Close*
5. *Clone*

2.6.2.2 Buttons

1. *Close*
2. *Clone*
3. *Iconify*

2.6.2.3 Window Layout

The Figure 6 displays the suggested layout of the Module view window.

2.7 Table View Command

We list and describe the desired features in section 2.7.1. We suggest an x-motif implementation in section 2.7.2.

2.7.1 Table View Window Specifications

TAS include a certain number of commands to view the current status of selected tables. These commands should remain available as line-commands. It would be useful to add a Table View Window in which the status of all tables is displayed. In the brief mode, The view field should display the table names, their entry number, the number of variables they contain, the number of OK rows, i.e. the number of rows properly filled, the current state of the table (filled:F or not filled:N) the maximum number of rows (Maxlen), the table type (static or dynamic), the modification count (ModCnt) and the version number. In the verbose mode, additional information would include the variable names and sizes. In addition, it should be possible to select or de-select table for output in output io stream. The selection would be done with the mouse or by means of special menus. Finally, it should be possible to Save the current status of tables to file or to Open an old status file and display its content in an independent "Table View" window.

1. *Selection of tables to be written out in the event output stream.*
2. *Selection of an output file to Save current table view buffer*
3. *Close current table view buffer and file.*
4. *Selection of font types, sizes and Colors used to display the table status.*

2.7.1.1 Table Selection

The user should be able to use a menu item to select or de-select all tables for output in the event stream. In addition, he should be able to select/de-select one or few tables using the mouse to select their name in the view field and "click" on select/de-select buttons located in the "Table View" window. We propose to use TABLE/SELECT and TABLE/DE-SELECT command names.

2.7.1.2 File management

The user should be able to store away to an ASCII file the current status displayed by the window. There should be commands to Open a previously Saved file, Open/create a new file, and commands to update the status Saved in a file. Analogously to other windows defined, we suggest to use the following commands: TABLE/Open, TABLE/Close, TABLE/Save, TABLE/Save As, and TABLE/Clone.

2.7.1.3 Options selection

The user should be able to select the font type, size and Colors used in the various fields of the "View TABLE" window.

2.7.2 Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.7.2.1 Window Title: *Table View*

2.7.2.1 Menus

2.7.2.1.1 FILE

1. *Open*
2. *Save*
3. *Save As...*
4. *Close*
5. *Clone*

2.7.2.1.2 Table Selection: Select

1. *Select All*
2. *De-Select All*
3. *Select*
4. *De-Select*

2.7.2.1.3 Option Selection: Options

1. *Fonts*
2. *Colors*

2.7.2.2 Buttons

1. *Close*
2. *Clone*
3. *Iconify*

2.7.2.3 Window Layout

The Figure 6 displays the suggested layout of the Module view window.

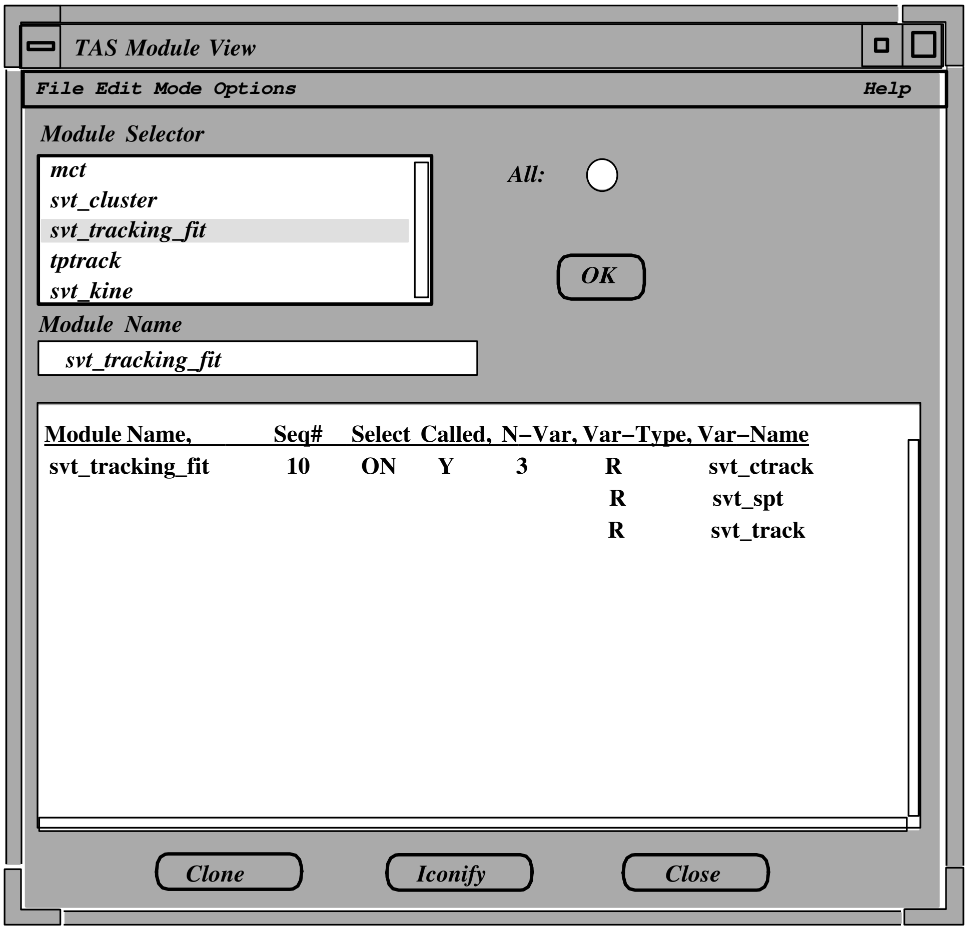

Fig. 6: Module View Window

2.8 Variable Value View Command

We list and describe the desired features in section 2.8.1. We suggest an x-motif implementation in section 2.8.2.

2.8.1 Variable Value View Window Specifications

TAS include a certain number of commands to view and set the content of selected variables of a table. These commands should remain available as line-commands. It would be useful to add a Variable View Window in which the content of selected variables of selected tables is displayed.

We envision at least two possible modes of display. In one, the variable are displayed in manner similar to the current method used by TAS to display the content of variables. A more interesting way to list the variable content would be to show them on a per row basis. In such a mode, a title line would give the name of the variables actually displayed and the following lines would show the content of the table row by row showing one and only table row per window line. Additionally, there could be two modes of operation: update and continuous. In the update mode, only the current request is shown in the window. In the continuous mode, the current request is appended to the bottom of the variable view buffer. We list the desired features:

1. *Selection of display mode: per variable mode or per row mode.*
2. *Selection of update mode: update mode or continuous mode.*
3. *Selection of an output file to Save current variable view buffer.*
4. *Close current variable view buffer and file.*
5. *Open/Close current table view buffer and file.*
6. *Selection of font types, sizes and Colors used to display the table status.*

2.8.1.1 Display Mode Selection

The user should be able to use a menu item to select the display mode. In the "update mode", only the current "show/variable" request is shown on the screen. That is the buffer is reset/emptied and the new list is shown. In the continuous mode, the new request output is appended at the end of the current buffer.

2.8.1.2 File management

The user should be able to store away to an ASCII file the current variable view buffer. There should be commands to Open a previously Saved files, Open/create a new file, and commands to update the content Saved in a file. Analogously to other windows defined, we suggest to use the following commands: VARIABLE/Open, VARIABLE/Close, VARIABLE/Save, VARIABLE/Save As, and VARIABLE/Clone.

2.8.1.3 Mode selection

The user should be able to use menu items to select either the "update" or the "continuous" mode and the "per variable" or "per row row" mode. The menu should act like toggle switches.

2.8.1.4 Options selection

The user should be able to select the font type, size and Colors used in the various fields of the "VIEW VARIABLE" window.

2.8.2 Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.8.2.1 Window Title: *Variable View*

2.8.2.1 Menus

2.8.2.1.1 FILE

1. *Open*
2. *Save*
3. *Save As...*
4. *Close*
5. *Clone*

2.8.2.1.2 Mode Selection: Modes

1. *Update*
2. *Continuous*
3. *Per variable*
4. *Per row*

2.8.2.1.3 Option Selection: Options

1. *Fonts*
2. *Colors*

2.8.2.2 Buttons

The buttons of this window have similar action as those defined for all other view windows.

1. *Close*
2. *Clone*
3. *Iconify*

2.8.2.3 Window Layout

The Figure 7 displays the suggested layout of the Variable view window. Note that the variable title line appears only in the "per row" mode.

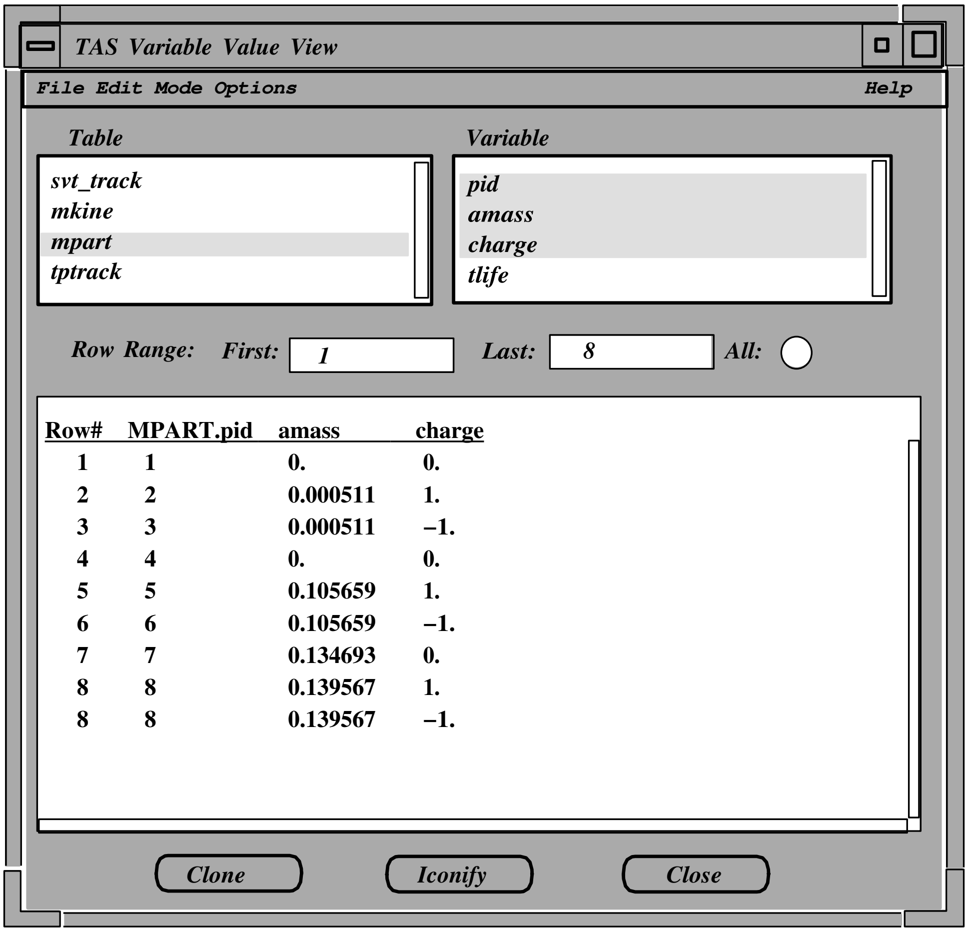

Fig. 7: Variable View Window

2.9 Histogram/N-Tuple Definition View Window

We list and describe the desired features in section 2.9.1. We suggest an x-motif implementation in section 2.9.2.

2.9.1 Histogram/N-Tuple View Window Specifications

TAS include a certain number of commands to book, set the filling of TAS histograms and view the histograms once they have been filled. These commands should remain available as line-commands. It would also be useful to add a Histogram Definition View Window in which the booking parameters and the filling variable associated with histograms and n-tuples would be displayed.

In a brief mode, one could display only the booking parameters, i.e. the histogram id, the title, the number of bins, the minimum and maximum of the axis and the the maximum content of a bin. In the verbose mode, one would display the table.variable names associated with the filling of the histograms. Additionally, it should be possible to store the current buffer of Open a previously Saved Histo view buffer. We list the desired features:

1. *Selection of display mode: brief or verbose mode.*
2. *Selection of an output file to Save current Histo view buffer.*
3. *Close current Histo view buffer and file.*
4. *Close Histo view buffer and file.*
5. *Selection of font types, sizes and Colors used to display the table status.*
6. *Clone current window or Open new window.*

2.9.1.1 Display Mode Selection

The user should be able to use a menu item to select the display mode to brief or verbose. In the "brief" mode, only the booking parameters are shown. In the "verbose mode", the link with table.variables is explicitly shown. We suggest to use the following command names: Histo/View/Brief and Histo/View/Verbose. The verbose mode would be the default.

2.9.1.2 File management

The user should be able to store away to an ASCII file the current Histo definition view buffer. There should be commands to Open a previously Saved files, Open/create a new file, and commands to update the content Saved in a file. Analogously to other windows defined, we suggest to use the following commands: Histo/View/Open, Histo/View/Close, Histo/View/Save, Histo/View/Save As..., and Histo/View/Clone.

2.9.1.3 Options selection

The user should be able to select the font type, size and Colors used in the various fields of the "Histo definition view" window.

2.9.2 Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.9.2.1 Window Title: *Histo Definitions View*

2.9.2.1 Menus

2.9.2.1.1 File Management: *File*

1. *Open*
2. *Save*
3. *Save As...*
4. *Close*
5. *Clone*

2.9.2.1.2 Mode Selection: Modes

1. *Brief*
2. *Verbose*

2.9.2.1.3 Option Selection: Options

1. *Fonts*
2. *Colors*

2.9.2.2 Buttons

The buttons of this window have similar actions as those defined for all other view windows.

1. *Close*
2. *Clone*
3. *Iconify*

2.9.2.3 Window Layout

The Figure 8 displays the suggested layout of the Histogram view window.

2.10 Processing Status View and Control Window

We list and describe the desired features in section 2.10.1. We suggest an x-motif implementation in section 2.10.2.

2.10.1 Processing Status and Control Specifications

This window is created for practical purposes to provide a display of the processing status and to provide control over the processing parameters and task. It is in many ways redundant as it gives direct access to commands accessible from other windows as well.

1. *Processing Status Display*
2. *Processing Control*
3. *Event Stream Selection, Input and Output*

2.10.2 Window Implementation

We list/describe the Window title, the MENU, the Buttons, Input/Output fields separately below. The proposed window layout is shown last.

2.10.2.1 Window Title: *Process View and Control*

2.10.2.1 Menus

2.10.2.1.1 Event Stream Selection: *Event I/O*

1. *Open Input Stream...*
   Opens an "Open Input Stream Dialog" window and upon successful completion of the dialog, Open new Input Stream. Close previously Opened Input Stream, Enable ReOpen Input Stream and Close Input Stream commands. Update display of input stream name.
2. *Fake Input Stream*
   Requests the bypass of event Reading by the event processor. Close previously Opened Input Stream if any, Update display of Input Stream as "FAKE INPUT".
3. *Close Input Stream*
   Close currently Opened input stream. Update display of Input Stream As "NONE".
4. *ReOpen Input Stream*
   Rewind Input Stream.
5. *Open Output Stream...*
   Opens an "Open Output Stream Dialog" window and upon successful completion of the dialog, Open Output Stream. Close previously Opened Output stream if any. Update display of output stream name. Enable Close Output Stream command.
6. *Close Output Stream*
   Close currently Opened output stream, suspend output of event. (dynamic) tables.
7. *Suspend Event Output*
   Suspend output of event (dynamic) tables. Update displayed event process status as "Event Output Suspended".
8. *Resume Event Output*
   Resume Output of event tables. Update displayed event process status as "Event Output Selected".

2.10.2.1.2 Static Table I/O: Tables

1. *Load Static Table...*
   Opens a "Load Static Table Dialog" window and upon successful completion of the dialog, Load the Static table from user selected file.
2. *Save Static Table*
   Save the user selected tables to the user selected file. If no file or tables are selected, defaults to Save Static Table As....
3. *Save Static Table As...*
   Opens a "Save Static Table Dialog" window and upon successful completion of the dialog, Save the user selected Static tables to the user selected file.
4. *Reset Static Table...*
   Opens a "Reset Static Table Dialog" window and upon successful completion of the dialog, Reset the user selected Static tables.

2.10.2.1.3 Event Processing Control: *Process*

1. *Go All*
   Starts Event Processing of all events in the current input stream. Enables the Pause and Stop commands. The GO commands are enabled only if valid analysis Modules are selected and if an invent stream has been selected by the user (or the fake event option has been selected), Opened successfully and has not reached the end of file.
2. *Go...*
   Opens a "Go..." Dialog window and upon successful completion of the dialog, Starts Processing of the user selected number of events and event type from the current input stream. Enables the Pause and Stop commands.
3. *Pause*
   Suspend Processing of events. This command is disabled after it has been selected. It is enabled by the Go commands and re-enabled by the Resume command. It enables the Resume command.
4. *Resume*
   Resume Processing of events. By default this command is not enabled. It is enabled when the user suspends the event processing with the Pause command.
5. *Stop*
   Interrupts the processing of events. The command is enabled by the Go commands and disables itself upon selection.

2.10.2.1.3 Options

1. *To be defined*

2.10.2.2 Buttons

The Close, Clone, Iconify buttons of this window have similar actions as those defined for all other view windows.

1. *Close*
2. *Clone*
3. *Iconify*
4. *Pause* Same action as the Process/Pause Menu command.

5. *Stop* Same action as the Process/Stop Menu command.
6. *Go* Same action as the Process/Go Menu command.
7. *Resume* Same action as the Process/Resume Menu command.

2.10.2.3 Window Layout

The Figure 9 displays the suggested layout of the Processing Status view window.

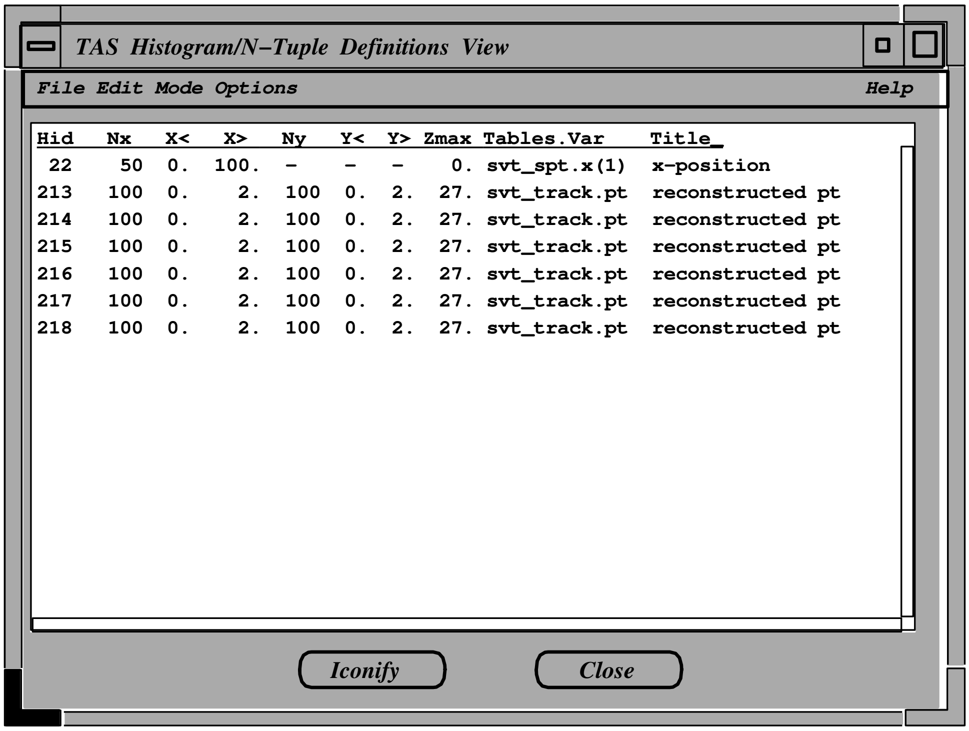

Fig. 8: Histogram Definitions View Window

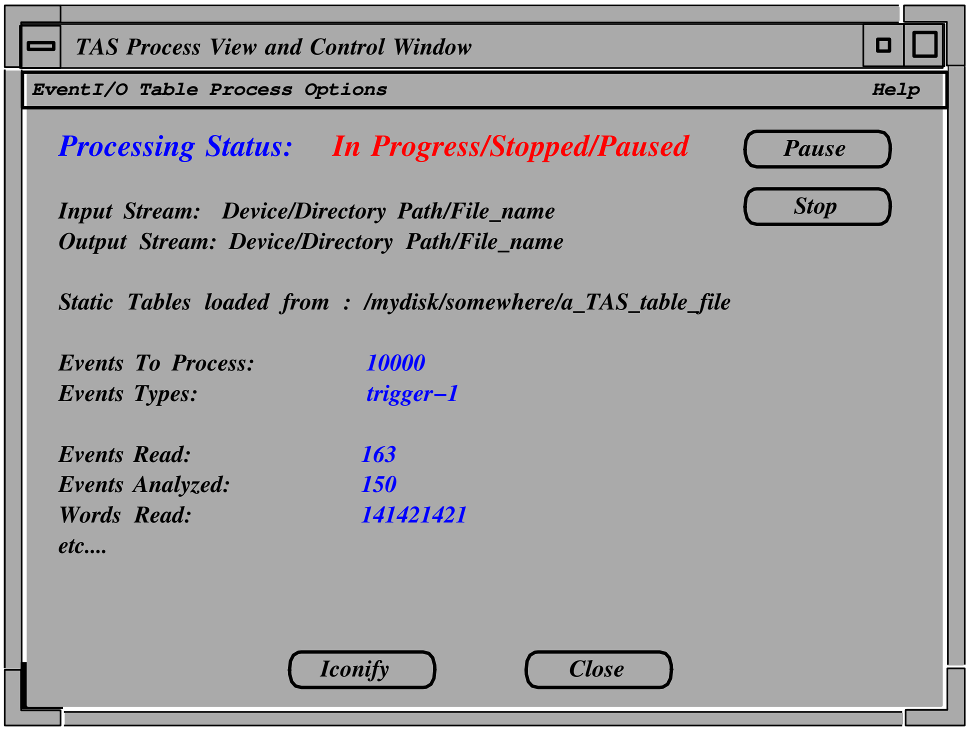

Fig. 9: Process View and Control Window

2.11 Histogram/N-Tuple Booking Window

This window will be included in the next version of this document.

2.12 Graphics Window

We propose to use, at least for the time being, the PAW HIGZ graphics window implementation. This window is basically featureless apart from being a move-able, re-sizeable, iconizable HIGZ window. We refer the Reader to the paw documentation for the definition of the features and properties of this window.

2.13 Dialog Windows

There are a few of them. They are Opened upon user's selection of some menu items to set various options. They will be defined/expended in the next version of this document.

4. References

1. R. Brun et al, CERN, PAW and Paw++ User Guide.
2. R. Brun etal. CERN, KUIP User Guide.
3. NCSA-Mosaic, Software Development Group, National Center for Supercomputer Applications.