CHAPTER 3 SCRIPT

The Script Utility (see Figure 3-1) allows us to record the process of solving a model, or any other user interface process. Once this process is recorded in the form of a script, the script can be repeated either fully or in part. The solution process for a given model can be communicated to another modeler by distributing a script saved to a file. Following is an outline of the various menus and buttons on the script window along with a library of the ASCEND commands which can be recorded.

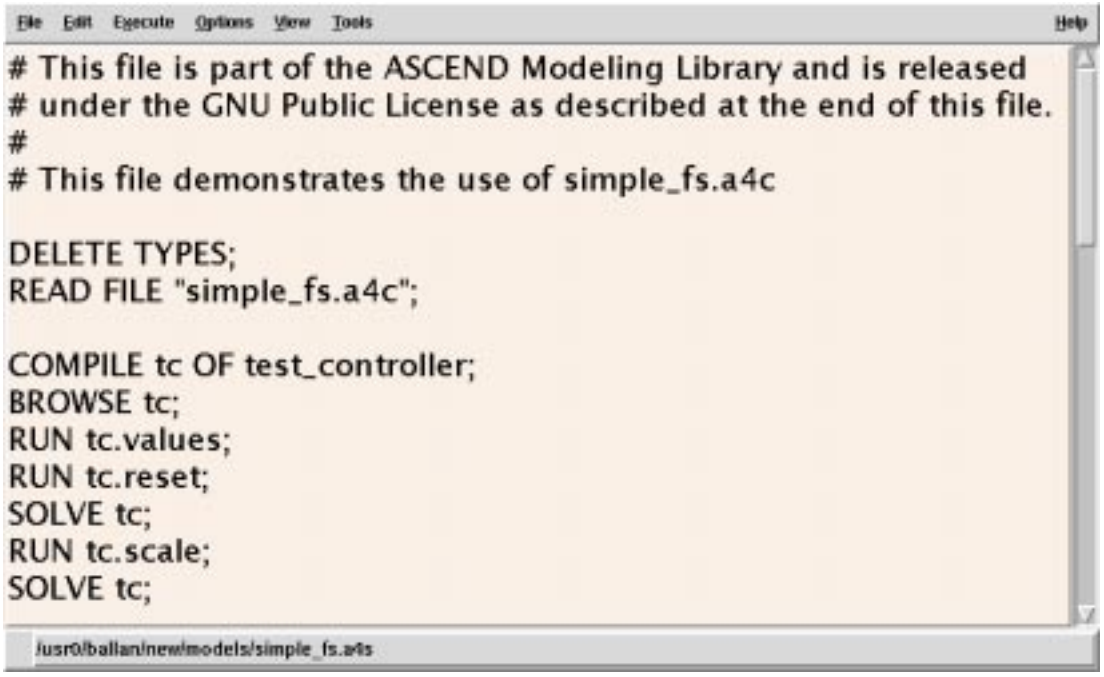

Figure 3-1 ASCEND's Script Window

3.1 THE SCRIPT MENU BAR

3.1.1 SCRIPT FILE MENU

The script file menu provides several functions for managing script files. The script utility may contain multiple scripts but will only display one at any given time. Upon startup a scratch workspace is provided

New File Request a buffer name and creates a new buffer with this name.

Read File Requests a filename through the file selection box and proceeds to load this file (which is assumed to contain ASCEND Script and/or Tcl statements) into a new Script window buffer. No error checking is performed on the loaded file.

Import File Requests a filename through the file selection box and appends the text contained in this file to the end of the current buffer.

Exit ASCEND Exit the ASCEND system. You will asked to confirm that you wish to do this.

Save Saves the text in the current script buffer window to the current script file (indicated by the filename at the bottom of the script window). The existing file is overwritten.

Save As Request a filename through the file selection box and saves the text in the current script buffer window to this file. If the specified file exists, it is overwritten.

Buffer List A list of scripts used in the current ASCEND session is displayed at the bottom of the file menu. A script can be redisplayed in the script window by selecting it from the buffer list. This window contains the words “License-Warranty.tcl” when you first start ASCEND (which is the initial contents of the Script buffer).

Note: if you alter the contents (for example, clear it), the system will restore the modified contents and not the original contents. The file you originally read remains unchanged unless you save to it.

3.1.2 SCRIPT EDIT MENU

Record actions When the record function is activated a log of interface events with defined ASCEND Script commands is appended to the end of the current script window buffer. Most, but not all, interface events have

corresponding script commands. The record function can be turned on and off by toggling the pull down button on the grill menu or the record button at the bottom of the script window.

Select all Selects (highlights) all text in the window. (A known bug exists here -- if you do not place the cursor into the text buffer the first time you use this tool, the highlighting may not occur although the text is in fact highlighted.)

Delete statements Removes (cuts) the *selected* text. The removed text is NOT saved for later pasting.

Cut Cut highlighted text to the computer paste buffer. You can paste this text into any application that supports cut, copy and paste -- e.g., into Framemaker or Excel.

Copy Copy highlighted text to the computer paste buffer. You can paste this text into any application that supports cut, copy and paste -- e.g., into Framemaker or Excel.

Paste Paste the contents of the computer paste buffer into the Scipt buffer at the point of the cursor.

3.1.3 SCRIPT EXECUTE MENU

Run statements selected This button takes the selected text, breaks it into statements delimited by any semicolons (;) that appear in the selection, and executes each statement in the Tcl global environment.

Step through statements selected This button allows you to single step through the highlighted statements. Two windows open: a small window that allows you to proceed to the next statement (next button), change from single step to running the rest of the script automatically (go button) or stop (stop button) and the Display where, while single stepping, you will see the statement being executed.

3.1.4 SCRIPT OPTIONS WINDOW

Save all options and appearances for all windows This tool saves the complete current appearance of the ASCEND windows and all the options selected anywhere in the system. It writes the text file *ascend.ad* and a number of text files ending with *.a4o* into the subdirectory *ascdata* in your “home” directory. The ascend.ad file is a collection of the information placed into the .a4o files at the time you run this instruction. Its main role is to aid in dubugging.

At any time, you can go into any of the windows and use the appropriate Save “window” appearance button to write a new .a4o file for that window. You can also save new options where such tools exist throughout the system, resulting in other .a4o files. The new information in these .a4o files will not be reflected in ascend.ad file, but it will be what is used to set window positions, etc., overriding what is in the ascend.ad file.

3.1.5 SCRIPT VIEW WINDOW

Font Opens the window that lets you reset the fonts for this window. You can select the type of font, the style (bold, etc.) and the size for the font.

Save Script appearance Saves the current settings for this window for font settings and window size and placement on your computer screen. These become the default settings for opening this window in the future. These settings are saved in a *.a4o* text file for this window which the sytem stores in the subdirectory *ascdata* in your “home” directory.

Save all appearances Saves a .a4o file for all the ASCEND windows. (See the above instruction to see the purpose of these files.)

3.1.6 SCRIPT TOOLS WINDOW

Use the tools in this menu to open other windows in ASCEND when they are closed or iconified.

This menu lists the major ASCEND windows. Selecting one of them will open that window on your screen. See the help manuals for these windows to find out more about them. This menu has almost exactly the same content as the ASCEND toolbox window. See the documentation corresponding to the toolbox for more details.

3.1.7 SCRIPT HELP MENU

On SCRIPT Brings up a text description of where to look for help on this window (i.e., it points to the pdf version of this document on the WWW.) You may, of course, look into the section mentioned in any local (but perhaps outdated) copy of the documentation.

On getting started with ASCEND Brings up a text description of where to look for help on getting started with using ASCEND -- the howto book (i.e., it points to the pdf version of this document on the WWW.) You may, of course, look in any local (but perhaps outdated) copy of the documentation.

About ASCEND IV Brings up a window telling you briefly about the GNU license and other information about the ASCEND IV software. This same window opens the first time you start ASCEND on your computer.

3.2 THE SCRIPT LANGUAGE

3.2.1 SUMMARY

Script keywords are commands defined for ASCEND (in CAPS) which may be used on the commandline or in the Script. Keywords are actually Tcl functions which encapsulate one or more of the C primitives and other Tcl procedures, so that the user can conveniently emulate button presses. A working knowledge of tcl is not necessary to benefit from the Script's functionality; however, the tcl literate user will be able to create very powerful scripts.

Each keyword takes 0 or more arguments. The use of arguments is given in the following syntax:

<arg> indicates the use of **arg** is required.

<a1,a2> indicates that the use of either **a1** or **a2** is required

<a1 a2> indicates use of both **a1** and **a2** required. Usually written `<a1> <a2>`

[a1] indicates the use of **a1** is optional.

[a,b] indicates that either **a** or **b** is optional, but not both.

qlfdid is short for 'QuaLiFieD IDentifier'

qlfpid is short for 'QuaLiFied Procedure IDentifier'

OF, WITH, TO, and other args in all CAPS are modifiers to the keyword which make it do different things.

{} It is generally best to **enclose all object names and units** in {braces} to prevent Tcl from performing string substitution.

3.2.2 QUICK REFERENCE:

ASSIGN	Sets the value of something atomic
BROWSE	Exports an object to the browser
CLEAR_VARS	Sets all the fixed flags to FALSE
COMPILE	Compiles a simulation of a given type
DELETE	Deletes a simulation or the type library
DISPLAY*	Displays something
INTEGRATE	Runs an IVP integrator
MERGE	Performs an ARE_THE_SAME
PLOT	Creates a plot file
PRINT	Prints one of the printable windows
PROBE	Exports an object to the probe
READ	Reads in a model, script, or values file.
REFINE	Performs an IS_REFINED_TO
RESTORE*	Reads a simulation from disk.
RESUME	Resumes compiling a simulation
RUN	Runs a procedure
SAVE*	Writes a simulation to disk
SHOW	Calls a unix plot program on a file from PLOT
SOLVE	Runs the solver
WRITE	Writes values in Tcl format to disk

* Items not yet implemented.

3.2.3 COMMANDS

ASSIGN

```
ASSIGN <qlfdid> <value> [units]
```

Sets the value of atom ‘qlfdid’ from the script. If value is real, it is required to give a set of units compatible with the dimensions of the variable. If the variable has no dimensions yet, ASSIGN will fix the dimensions.

BROWSE

```
BROWSE <qlfdid>
```

Exports qlfdid to the browser, displaying it as the current instance in the browser.

CLEAR_VARS

```
CLEAR_VARS <qlfdid>
```

Sets the value of the fixed flag to FALSE for all the variables on qlfdid.

COMPILE

```
COMPILE <simname> [OF] <type>
```

Build a simulation of the type given with name simname. You can get away with leaving out OF or spelling it wrong.

DELETE

```
DELETE <TYPES,simname>
```

The modifier TYPES will cause all simulations to be deleted. If a simulation name (simname) is specified only that simulation will be deleted.

DISPLAY

```
DISPLAY <kind> [OF] <qlfdid>
```

How qlfdid is displayed varies with kind. kinds are: VALUE ATTRIBUTES CODE ANCESTRY

INTEGRATE

```
INTEGRATE {qlfdid args}
```

Runs an integrator on qlfdid. There are several permutations on the syntax. It is best to have solved qlfdid before hand to have good initial values.

```
INTEGRATE qlfdid (assumes LSODE and entire range)
INTEGRATE qlfdid WITH (assumes entire range)
INTEGRATE qlfdid FROM n1 TO n2 (assumes lsode)
INTEGRATE qlfdid FROM n1 TO n2 WITH integrator
```

Requires:

- n1 < n2
- qlfdid be of an integrable type (a refinement of ivp.)

MERGE

```
MERGE <qlfdid1> [WITH] <qlfdid2>
```

ARE_THE_SAME qlfdid1 and qlfdid2 if possible.

OBJECTIVE

```
OBJECTIVE
```

Semantics of OBJECTIVE that will be supported are unclear as no OBJECTIVE other than the declarative one is yet supported. Not implemented yet

PLOT

```
PLOT <qlfdid> [filename]
```

Writes plot data from qlfdid, which must be a plottable instance, to filename.

PRINT

```
PRINT <PROBE,DISPLAY>
```

Prints out the text currently in the Probe or Display.

PROBE

```
PROBE ONE qlfdid
```

Exports the item qlfdid to the Probe.

```
PROBE ALL qlfdid
PROBE qlfdid
```

Exports items found in qlfdid matching the current specifications of Visit in the Browser. By default, all variables and relations.

Items always go to currently selected probe context.

READ

```
READ [FILE,<VALUES,SCRIPT>] <filename>
```

Loads a file from disk. Searches for files in directories (Working directory):.:$ASCENDLIBRARY unless a full path name is given for filename.

The modifier FILE is used to indicate that the file contains ASCEND source code (ASCEND source code files normally have a .asc extension).

The modifier VALUES is used to indicate that the file contains variable data written by WRITE VALUES (These files normally have a .values extension).

The modifier SCRIPT is used to indicate that the file is a script file to be loaded at the end of the Script window (Script files normally have a .s extension).

If neither VALUES nor SCRIPT are found, FILE will be assumed. Note: You will get quite a spew from the parser if you leave out the SCRIPT or VALUES modifier by accident.

REFINE `REFINE <qlfdid> [TO] <type>`

Refines qlfdid to given type if they are conformable.

RESTORE `RESTORE <file>`

Reloads a simulation from disk

RESUME `RESUME <simname>`

Reinvokes compiler on simname.

RUN `RUN <qlfpid>`

Run the procedure qlfpid as if from the browser Initialize button.

SAVE `SAVE <sim> [TO] <filename>`

Filename will be assumed to be in Working directory (on utils page) unless it starts with a / or a ~. Not implemented yet.

SHOW `SHOW <filename,LAST>`

Invokes the plotter program on the filename given or on the file LAST generated by PLOT.

SOLVE `SOLVE <qlfdid> [WITH] [solvername]`

Exports qlfdid to the solver and attempts to solve it with the default solver (usually QRSlv) or the solver indicated by the optional solvername argument. Solvername must be given as it appears on the menu buttons. Bugs: Should use current solver rather than default.

WRITE `WRITE <kind> <qlfdid> <file> [args]`

Write something (what sort of write indicated by kind) about qlfdid to a file. args may modify as determined by kind. At present only VALUES is supported. SYSTEM (for solver dump) would be nice.

```
WRITE VALUES filename.
```

Filename must be a full path name or in the pwd, also known as '.' .

3.3 SCRIPT WINDOW BINDINGS

In the event binding descriptions that follow, M1 is short for mouse-button-1 (the left mousebutton), M2 is the middle button, and M3 is the right mouse button. On machines with no middle button, M3 is still the right mouse-button and M2 is unavailable.

M1 repositions the cursor.

M1-Drag selects text.

Shift-M1[-Drag] extends the selection.

Double-M1 selects the nearest word.

Double-M1-Drag selects the nearest word and those you drag over, whole words at a time.

Triple-M1 selects the nearest line.

Triple-M1-Drag selects the nearest line and those you drag over, whole lines at a time.

M2 does nothing.

M2-Held-Down has an effect similar to the scrollbar.

M3 does nothing.

Control-M1 Starts another part of a disjoint selection.

UNIX bindings: The text widgets in ASCEND share a common stack of cut/copy/paste text pieces. This is a CMU extension of the text bindings, not default Tk behavior, and it is EMACS-like, but not EMACS (EMACS uses a

	ring, not a stack.) When the stack is empty, Paste does nothing. This is a design decision. The Tcl function ascPopText can be changed to behave differently.
Control-k	Cuts text to the end of the current line, putting it on the stack.
Control-w	Cuts the selected text, putting it on the stack.
Meta-w	(e.g. diamond-w on most Sun keyboards) Copies the selected text onto the stack.
Control-y	Pastes the most recent text added to the stack, and removes it from the stack.
Meta-y	Not supported.
MSW bindings:	The standard Control-X, Control-C, Control-V bindings of Microsoft Windows clipboard apply to text widgets. The UNIX text stack is not available.