User Manual

Tektronix

Analog Design System
ADS

T0-6360–00

This document applies for software version 5a0 and above.

Please check for change information at the back of this manual.

Second Printing: June 1996

Table of Contents

Getting Started

Tutorial

Window Descriptions

ADS–TekSpice Reference

Application Models

Appendices

List of Figures

List of Tables

Preface

This is the User manual for the Tektronix Analog Design System (ADS).

Section 1 *Getting Started* familiarizes you with the capabilities and terminology of the ADS system.

Section 2 *Tutorial* gives the step by step procedures for creating and simulating a circuit schematic using the ADS system.

Section 3 *Reference* describes the windows and commands available in ADS. There is also a description of the waveform processing functions available from ADS.

Conventions

In the *Getting Started, Tutorial* and *Reference* sections you will find various procedures which contain steps of instructions for you to perform. To keep those instructions clear and consistent, this manual uses the following conventions:

- Names of the Browser, Editor and Dialog Box windows are in initial capitals boldface print for example **Circuit Editor** and **Library Browser**.
- Names of panes within the windows are in initial capitals italics print for example *Schematic Editor Pane*.
- Names of pop–up button menu commands are in boldface print.
- Instruction steps are numbered. The number is omitted if there is only one step.
- When steps require that you make a sequence of selections using the pop–up button menu, a greater than (>) marks the transition between primary and secondary menus for example **zoom > nominal**.

ADS Version Conventions

The ADS, TekSpice and model libraries version naming procedures are as follows:

- The first position is an integer, usually a single digit. This position indicates major releases with a significant change in functionality.
- The second position is a character, usually one. This position indicates user interface modifications and one or more new features.

- The third position is an integer, usually a single digit. This position indicates bug fixes.
- Examples are: ADS 4e0, TekSpice 2d0, CMOS 1c1.
- Fourth, fifth and higher positions indicate alpha and beta test versions.

Related Manuals

The following manuals are related to the ADS User Manual:

- TekSpice Reference Manual (Tektronix part number TO–6370–00) describes the syntax and model details used in the TekSpice circuit simulator.
- QuicKic User Manual (Includes QuERC and Simtoq8) describes the QuicKic IC layout editor and auxillary software.

Getting Started

Overview

The Analog Design System (ADS) provides a workstation–based environment for Analog IC and Systems design.

The purpose of this user manual is 1) to help you set up and get started with ADS; and 2) provide a reference document for the ADS environment including windows, panes and command menus.

Features

Features of the ADS environment include:

- Schematic Capture
- TekSpice simulation and control
- TekSpice results display; plot windows and tables
- TekSpice results processing
- Experiment management
- Device library support
- Printer support
- Interface to IC layout
- Interface to Electrical Rules Checking

The user creates a circuit schematic from a library of symbols representing TekSpice element models such as resistors, capacitors, transistors. Operations in the schematic window include; cut, paste, copy, move and print.

Once the schematic is in a form ready for simulation the user can create control syntax for the type of simulation desired; bias, dc, ac or transient analyses. Default control syntax forms are available.

ADS uses TekSpice, a general purpose circuit simulator developed at Tektronix. The interface between ADS and TekSpice is transparent to the user; ADS generates a netlist from the schematic, invokes TekSpice, and reads the simulation results. The simulation is either executed locally on your workstation or at another Sun/UNIX server on the network (Remote simulation is a non–standard configuration and offered as separate software).

After completing a simulation, the results are plotted, printed or processed by user defined functions or by the built–in waveform functions.

Experiments can be set up, results obtained, compared with previous results and saved for later reference.

There is an extensive library of element models developed for specific IC processes, IC packages and other types of elements.

Printed output is available from any postscript laser printer. Printed output is generated as a postscript file or from a window snapshot.

ADS does not provide layout capability, but can generate a TekSpice netlist for subsequent layout and design verification with an external tool such as the QuicKic layout editor.

An electrical rule checker, QuERC, identifies possible Electrical Rule violations. A netlist conversion program, Simtoq8, maps simulation model names into the proper layout names, as well as extracts parasitic simulation elements.

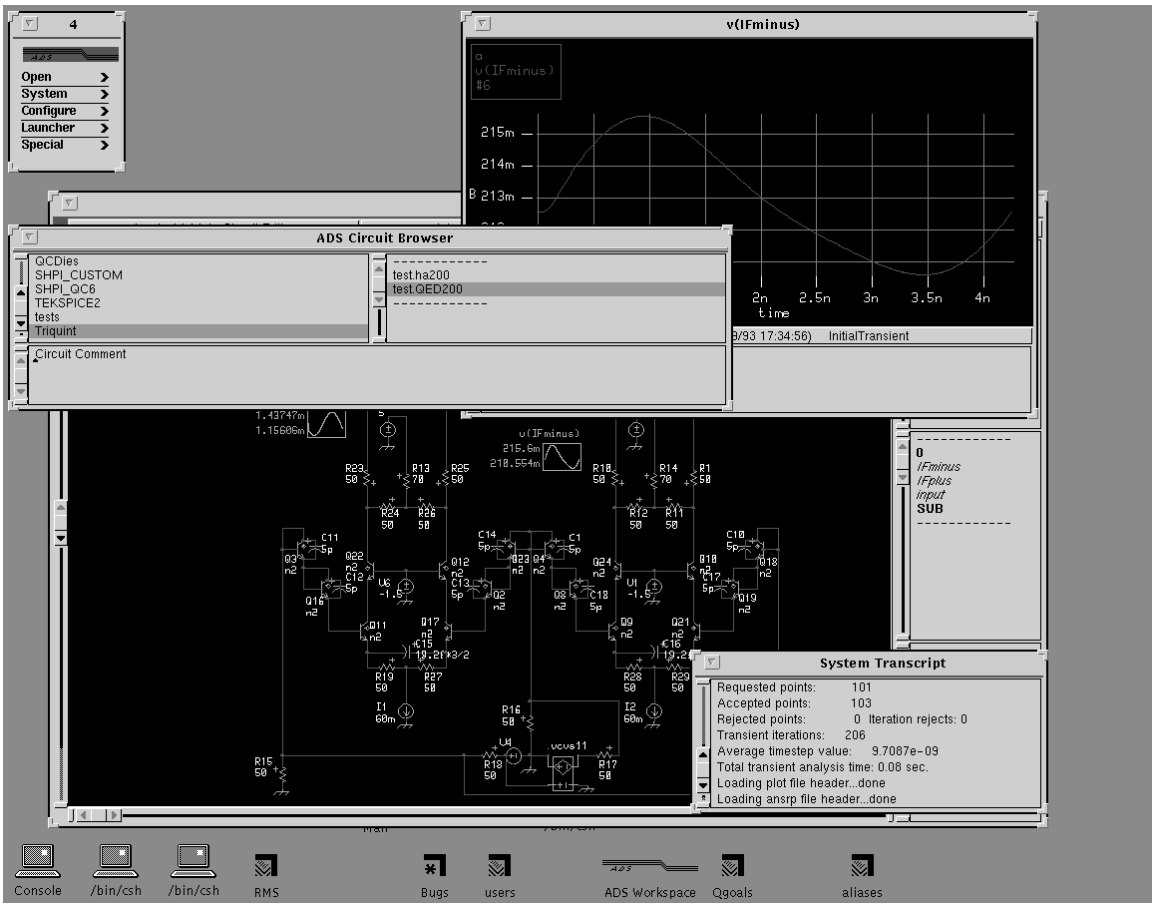


Figure 1–1: Typical User's Display

Figure 1–1 shows the display of a typical ADS user. This is just one snapshot of the many windows and panes available from within ADS.

ADS Support

Support for ADS is available throughout Tektronix, its subsidiaries and their customers. Each user is assigned an ADS support contact. This person is the focal point for support, and gets assistance as needed. In addition to phone support, electronic problem and enhancement requests are available through the bug report/e–mail system within ADS. This should be used whenever possible.

ADS Training

The best way to learn ADS is by hands–on experience. First, read the **Startup** section of this chapter.

Next, step through the **ADS Tutorial** in the *Tutorial* chapter. The tutorial session takes a few hours to complete. During this session you perform a series of basic tasks important to learning the ADS system. You are introduced to many of the important concepts within the ADS system. Later, as you start using ADS for circuit designs, examples in the *Appendix* of this manual are useful references.

ADS Setup

Obtain access to a workstation and contact a System Administration person familiar with setting up new user accounts, disk allocations, network connections, etc. This system administrator should be familiar with **System Administrator Procedures** at the end of this section.

ADS Windows

ADS windows coexist with the Motif or OpenLook window managers. Many of these window manager commands can be used along with ADS. An example of this would be the cut and paste operation between ADS and other windows. Common window commands such as **close**, **resize** and **back** also apply to ADS windows.

The colored title bar at the top of each ADS window helps differentiate ADS from other applications windows. This color can be changed with the **Launcher>change colors** command found in the **Launcher** (See the *Window Descriptions* chapter). If multiple ADS images are running, each image should use a different color bar.

A description of the functionality of the windows within ADS is given in the *Window Descriptions* chapter. Although technically they are all windows, for the purposes of this manual they are divided into launchers, browsers, editors and dialog boxes. The diagram in Figure 1–2 shows at least one path to get to each of the browsers and editors.

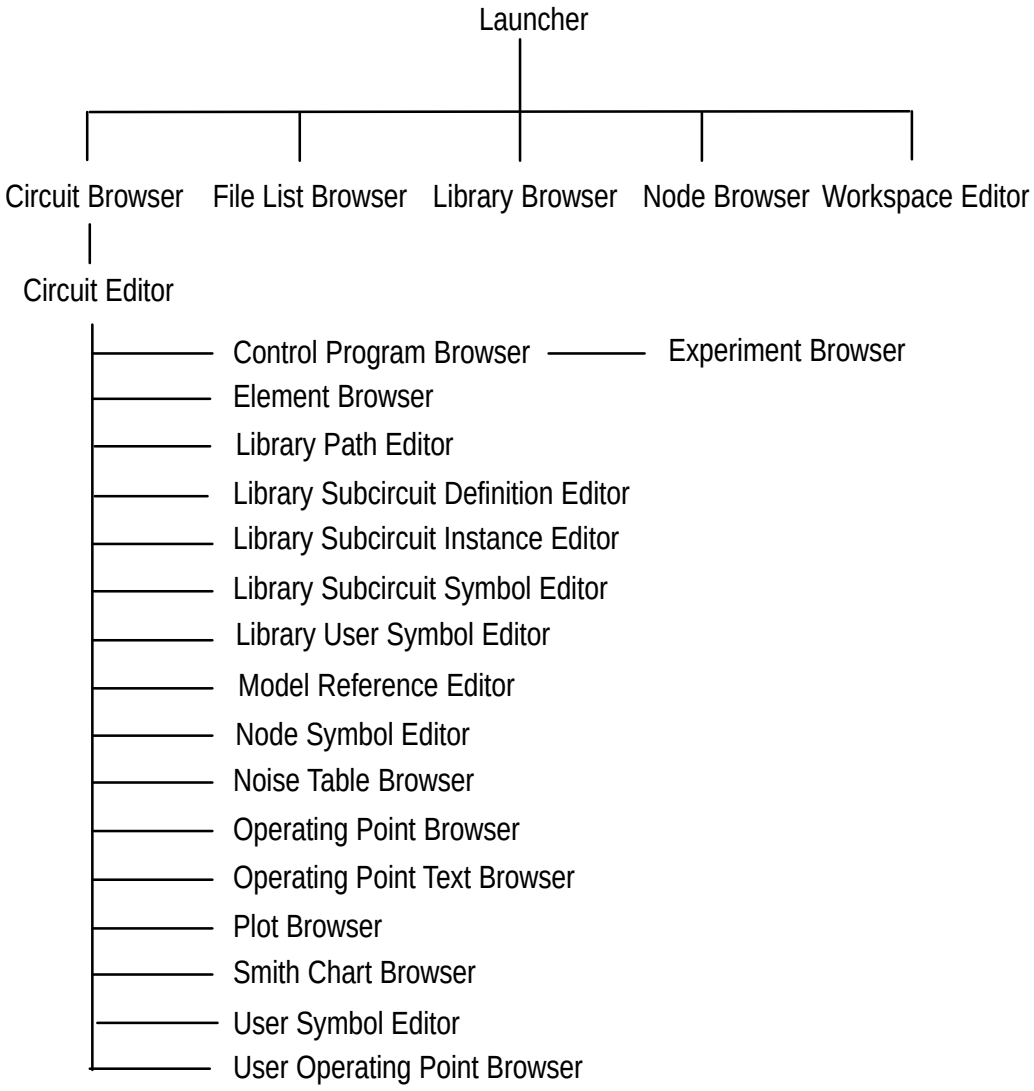


Figure 1–2: Access Path to ADS Browsers and Editors

Startup

Entering ADS

During the setup process, a directory was created to store the ADS image and related files. This directory is named **ads** (by default) and accessed by typing:

```
cd ~/ads
```

ADS is started from this directory by typing:

```
ads5
```

This runs an alias [`alias ads5 (~cae/bin/adsi ads5.im)`] which opens an ads image, ads5.im. To exit ADS see **Saving your ADS Image** later in this section.

The ADS image is a large binary file (starting at about 10 Megabytes) containing the basic ADS application. After a circuit schematic is created the image also contains the circuits and the element libraries required for this circuit. Also stored in this image are the open windows, and window locations. Each time you restart this ADS image, you enter this customized environment just as you left it yesterday, last week, or last year.

Using the Mouse Buttons

ADS uses all three mouse buttons; the Select button, the pop–up button and the special button. Figure 1–3 shows a diagram of the default position of the mouse buttons. The select button and the pop–up button are the most often used buttons.

Select button. The Select button is used to select items. Examples are graphics, text, a new active window, or items in a list pane. The Select button is the left mouse button. There are several methods of selecting:

Select: Point at the desired item and click the select button. (Click means to press and release the button.)

Area Select: It is possible to select items using an area select. This is done by placing the cursor arrow at a corner of the area you want selected; then press and hold the select button as you move the cursor across the area to be selected, and releasing the button. Items completely within the area are selected.

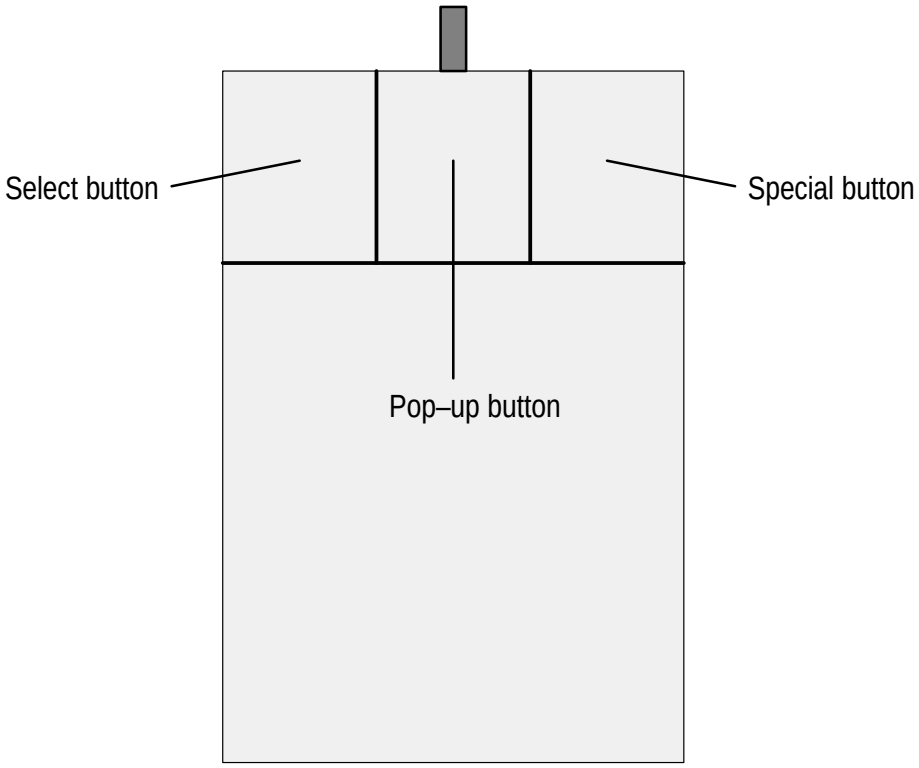


Figure 1–3: Default Location of Mouse Buttons

Multiple Item Select: The standard selection operation selects the item ”clicked on”, and deselects all other items. For additive selection, press the keyboard **Shift key** while clicking on the object. For subtractive selection press the keyboard **Control** key (Ctrl) while clicking on the subtracted object.

For selecting multiple items from a list, such as models from a library, click on items, while holding down the shift key. Each item is noted by a check mark.

Pop–up button. The pop–up button accesses menus and execute commands. In ADS, menus are context–specific. This means that different menus pop–up, depending on the present state of the window and the cursor location when pressing the pop–up button. The pop–up button is the middle mouse button (default).

Special button. The special button accesses menus independent of the cursor location. The special button menu contains the standard window manager commands **Close**, **Move**, **Resize**, **Front**, **Back**, **Refresh** and **Quit**, plus the commands **print window**, **new label** and **minimize**. The default special button is the right mouse button, but can be changed to the center mouse button using the **Configure > user interface** command from the **Launche**r.

Keyboard Macros

Depending on the type of workstation that is used to interface to ADS, different keyboard macros are available. For the Sun Worstations the following keys are available as alternatives to mouse pop–up buttons:

Copy
Cut
Paste

Menus

ADS is a menu–driven system. The menus contain commands related only to the applications supported in the panes. ADS displays only the menus that relate to the pane under the mouse cursor. For many panes, the mode (usually controlled by previously issued commands) and context (the location of the cursor within the pane) affect the pop–up menu contents when the pop–up button is pressed.

A pop–up button menu appears by pressing the mouse cursor. Moving the cursor up and down causes the highlighted (reverse video) menu commands to change one by one. To execute a command, first highlight it, then release the button. To avoid executing a menu item move the cursor off the menu area before releasing the button.

Secondary Menus

Some menus have one or more levels of secondary menus. Selection of these menus are from left to right. Secondary menus are differentiated by a > symbol. Selecting an item that has a secondary menu, causes an additional menu to appear at the right of that menu.

To select a command from a secondary menu, continue to press the mouse button as the cursor is moved across the > symbol and down the secondary menu to the item. To select a command, highlight it and release the mouse button (for example **Special > collapse ADS**).

Windows and Panes

During the course of a design, it is normal to have several ADS windows displayed on a screen at one time. Each window may have one or more panes. Each type of pane has a specific function. See the *Window Descriptions* chapter for detailed description on the types of windows and panes.

An example of two ADS windows are the **Circuit Editor** and **Launcher**. The **Circuit Editor** is one of the more commonly used windows and can create

circuits and subcircuit schematics which in turn create netlists. The netlists are automatically input to the TekSpice circuit simulator. The **Launcher** is a small, postage stamp sized window, usually located in a corner of the display terminal. The **Launcher** windows is the only ADS window that must remain open. It is recommended that the **System Transcript** window remain open.

Opening a Window

The opening of an ADS window is as follows:

1. Select the opening command. At this point the cursor changes from a pointer to a framing rectangle.
2. Move the mouse to position the upper–left corner of the window.
3. Press and hold the select button to pin the upper left corner of the window.
4. Drag the cursor down and to the right with the select button pressed. When the window is the desired size and shape, release the select button. The window is now opened.

Graphics Panes

Graphics panes let the user work with graphical objects such as lines, rectangles and arcs. Graphical objects are added, copied, moved and deleted in a graphics pane. Examples of ADS graphics panes are the *Symbol Editor* and *Schematic Editor* pane.

List Panes

List panes contains a list of information and a scroll bar appearing along one side of the pane. Some have a scroll bar across the bottom. There are dashed lines above the top item and below the bottom item indicating the ends of the list. If the list extends above or below the pane, scroll bars can access unseen items. Scroll bars move by placing the cursor on the bar, holding down the select button and moving the bar with the cursor. These scroll bars have the same operation as the Motif or Open Windows scroll bars.

Selector Panes

Selector panes allows changing from one mode to another. For example clicking on the *Edit/Analysis/Annotation Pane* changes from edit mode to analysis mode in the **Circuit Editor**. Holding down the mouse select button shows all the choices.

Status Panes

Status panes show the status of a selected item. For example *Execution Status* pane in the **Circuit Editor** shows the time and date of the selected experiment in the simulation control program.

Text Panes

Text panes allow editing text. They have a scroll bar allowing the user to view text that is outside the window, and a text cursor indicating the insertion point of text. Text panes resemble list panes except that text panes do not have dashed lines at the top and bottom of the text. The *Control Program Text Pane* in the **ADS Control Program Browser,** for example, allows entering analysis commands.

Text editing commands are the same for all text panes in ADS. Additional commands with other functionality may appear in some menus.

The basic text editing commands are as follows:

1. To **insert** text:
   - **a.** Move the mouse to position the text cursor (^) where you want the new text, and click the select button.
   - **b.** Type in the text. (Or use **paste** as a result of a prior **copy** or **cut** command from this text pane, another text pane or from a non–ADS window.)
2. To **select** text:
   - **a.** Move the text cursor to either end of the text you want selected and press the select button.
   - **b.** Keeping the button pressed, drag the cursor across the text you want to select. The selected text highlights (turns to reverse video).
   - **c.** To select a word, point at the word and click twice.
   - **d.** To select an entire line, move to the beginning or end of the line and click twice
   - **e.** To select all the text in the pane, move to the beginning of the first line or beyond the end of the last line and click twice.
3. To **cut** and **paste**:
   - **a.** Cut or copied text remains in the text buffer and can be inserted at the cursor location with the **paste** command. Each **cut** or **copy** operation replaces the contents of the text buffer.

b. Select (highlight) the text for deletion.

c. Hold down the pop–up button and move the cursor to to highlight the **cut** command. Release the button.

d. Locate the text cursor at the insertion point of the text.

e. Hold down the Pop–up button and move the cursor to highlight the **paste** command. Release the button.

4. To **search** for text:

a. Put the desired text in the text buffer, either by inserting or cutting or copying using the pop–up button **cut**, **copy** or **paste** command.

b. Click on the **again** command in the pop–up button menu.

Accept

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window. A **blue** box indicates that the contents of the pane are ”read only”.

Accept Command

The **accept** command is the only way changes in a design (such as in the *Schematic Editor Pane*) become permanent and known to other windows in ADS. Panes, where this notion applies, list the **accept** command on the pop–up menu.

Cancel Command

The **cancel** command allows rejection of changes and restores the circuit to the state just before the last **accept** command.

Dialog Boxes

Dialog boxes prompt for information needed to complete commands. For example the **Print Options Dialog Box** causes ADS to output schematics and text to a to a printer. The purpose of this dialog box is to obtain size, orientation and file names for the resulting output. ADS automatically opens and sizes

dialog boxes. After the user supplies the requested information and presses the continue button, the box goes away. It is possible to move the box by positioning the cursor on the title bar and dragging it to the desired location.

Saving an ADS Image

An ADS image represents circuits, models and window locations. During an ADS session this image resides in the workstation RAM. For a permanent copy of these changes use the **Special > save as** or **Special > exit** command from the **Launcher** window. This displays an optional default file name. Change this file name to save a backup copy.

To use other names or directories, change the default aliases. See the Installation section of this chapter for more details.

Saving an image is equivalent to taking a snapshot of your work. The image contains the basic ADS application, the circuits you design, the libraries used by these circuits, and the location and settings of the windows at save time. When restarting ADS, you reenter this environment with everything on the screen as you left it.

If you do not save after an ADS session (exit without saving, or computer power down), all changes made during that session are lost.

An automatic save capability is available to minimize lost work due to unforeseen problems. It is started with the **Configure > automatic save** command from the **Launcher**. You can use this save feature in a variety of ways. A common use is to reserve enough disk space for backing up your image, ads4.backup.im for example. Use the **Configure > automatic save** command to set that name and the frequency of save time, for example 60 minutes.

To clear up the screen for other applications, iconify your ADS windows using the **Special > collapse ADS** command. This collapses all ADS windows into one small window. This does not save the image to a file.

EDIF Save

EDIF (Electronic Design Interchange Format) files are an ASCII representation of the circuitry, control program and libraries within the ADS system.

There are different options for EDIF output. All new versions of ADS are able to read in this data, thus EDIF is a safe way to archive data when a design is at certain milestones.

Full EDIF Backup: To save all projects, circuits, libraries and control programs use the **System > backup to EDIF** command from the **Launcher**. This prompts for a directory name, and places all data into that directory and constructed

subdirectories. This command is used when an entire project is complete, or when circuitry requires merging into another image.

Library EDIF files: Libraries of models and individual models may be written with the **write > edif** command within the **Library Browser**. This approach shares libraries or library models with other designers. Care must be taken to see that all model references within the library model are satisfied. If in doubt, the encapsulated option below is a safer approach.

Circuit EDIF files: Circuits are written with the **write > edif** command from the **Circuit Browser,** with the circuit of interest selected. A dialog box opens, allowing selection of write options. Internal libraries are the most common option. Select external libraries only if they are non standard. The full write option writes all models available, whereas the minimal write only writes out models which are actually used in the circuit. A Circuit EDIF is encapsulated into a single UNIX file. On read in, the file expands into appropriate libraries, circuits and subcircuits, preserving library paths. Libraries are given a unique name (using the string ”encapsulated”), to assure they do not overwrite existing libraries reading back in. These libraries should be renamed as appropriate before your design goes into production.

Using the EDIF approach is the simplest method to share circuits between designers, or to archive a circuit at appropriate milestones.

Before sending a design into production, it is recommended that all ”encapsulated” libraries be eliminated from the **Circuit Browser**. This avoids confusion on which libraries are used. These changes are done with the **set libraries** command in the **Circuit Browser**, after the appropriate names are changed and models merged in the **Library Browser**.

To Minimize the Loss of Work

To minimize the loss of work in the event of a disk, system or network crash, use the following guidelines:

Be sure disk files are backed up on a regular basis.

Save images regularly. At least once a day, and more often if possible. The automatic save option automatically saves files at a predefined interval.

Use EDIF files to save circuit data at appropriate milestones.

Bug Reports and Enhancement Requests

ADS supports an e–mail based tracking system for bugs and enhancement requests. This system is normally accessed using the **open > bug report** command from the **Launcher**.

It may also be opened from an **Error Notification** dialog box window. This is an error window that automatically appears when ADS recognizes the occurrence of a bug. When the **Bug Report** dialog box is opened from this window, sections of relevant ADS application code are automatically inserted into the bug report significantly aiding problem diagnosis. For additional information on Bug Reports see the *Window Descriptions* chapter.

ADSI Script

The **adsi** command starts up an interpreter (VM) for ADS. The format for this command is as follows:

```
adsi [-voe <version name>] <image name>
```

The **adsi** command runs the ADS interpreter on a specified image name. The version of the interpreter is specified with the –voe option. No version need be specified. The default version is the latest ADS interpreter. Normally the **adsi** command is invoked from an alias named **ads5**.

Hardware Requirements

ADS requires an above average investment in computer resources. Using minimum RAM memory and disk storage significantly reduces the productivity of the designer using the system.

NOTE. *The recommendations made here are not the minimum system requirements.*

Workstation type: Sun–Sparc 20

The Sun Sparc 20 is the current standard platform. ADS runs on older Sparc workstations at a reduced performance. A color display is required. A Tektronix X–Station can be used as an alternative to the Sun Sparc's display.

RAM: at least 64 MB

The minimum recommended physical memory is 64 megabytes. For large circuits, such as full custom ICs, additional memory reduces design time.

Disk Storage: at least 400 MB

One 400 MB drive is barely adequate. The */home* partition is left with ~165 MB, ADS tools use roughly 22 MB, leaving approximately 140 MB for the user. Adequate disk storage for the ADS user allows space for at least one image that can grow to 20 MB or more. ADS requirements for temporary space on the order of several megabytes, depending on the size of the simulations. For practical use, an additional 400 MB drive or the ability to access a file server is required.

Disk Swap Space: at least 128 MB

On a 400 MB drive, the default swap space is ~82 MB, which is adequate for one ADS user if no other large jobs are running at the same time (e.g., analyzing circuits with TekSpice, running Interleaf, etc.). A rule of thumb for a Sun workstation is that swap space should be at least double the amount of RAM. Hence a workstation with 64 MB of RAM should have 128 MB of swap space.

For SunOS determine how much swap space is installed on your workstation, by typing **pstat –s**. The Swap space is the sum of the 'used' and 'available' numbers.

Printing

ADS supports most Postscript laser printers and Tektronix color Postscript printers.

Operating System

ADS runs on Sun OS versions 4.1.3 and Solaris 2.5. To find out which SunOS version is running, look at the first line in the file */etc/motd* from which SunOS was booted.

System Administration (Support Personnel Only)

The machine where ADS is being installed is referred to as “the workstation”. The setup procedures are different for Tektronix Internal use and External use.

Setup Procedures – Internal Tektronix Only

Network Setup (non–yellow pages)

The network setup procedures if yellow pages are not used are as follows:

1. Get a network address assigned to the workstation.
2. Connect the workstation to a gateway machine on the ethernet network.
3. As *root*, configure your system for operation on the network....
   a. Add the workstation network address to */etc/hosts*.
   b. Start up the network daemons.
   c. Make sure the network daemon *inetd* is running.
   d. Determine the workstation name and network address: use the *hostname* command to get the host name and *hostid* command to get the address. The system name and IP address are set when the workstation is booted.
   e. See the SunOS installation manual for more detail.
4. Add a gateway machine if necessary using the commands given below. Identify the gateway machine. Edit */etc/rc.local*, searching for the “route”.

```
# If an explicit default route is desired, uncomment
# the command lines below,
# substituting the correct gateway host and hop count.
# For example,
# ''/etc/route -f add default gatemach 2''
#
# if [ -f /etc/route ]
# then
# /etc/route -f add default GATEWAY HOP_COUNT >/dev/null 2>&1 &&
# echo -n ' (default route)'
# fi
```

Routing must be established before mounting NFS files if any of the NFS mounted systems are on networks other than the immediate physical ethernet. For instance, inside the T&M group, all *routed* daemons are disabled, and the

route command (in */usr/etc*) is used to setup the default route to some IP router already on the network.

As the comment states, remove the # at the beginning of the lines in the *if* construct, and replace GATEWAY with the name of the host machine and HOP_COUNT with the number of machines on the network between the workstation and the gateway computer (typically 1).

If you had to uncomment the */etc/route* command, it was not run at the latest reboot. Therefore, you should either reboot the machine or run the */etc/route* command, for example:

```
/usr/etc/route -f add default tekigx 1 >/dev/null 2>&1
```

5. Add the following entries in */etc/hosts* file to enable distribution of ADS:

```
128.181.29.2              tekig6
```

Engineering Network Support distributes these addresses as part of the *hosts.std* file for building the *hosts* file. Ask the system administrator of the gateway machine if these entries are installed there.

Network Setup (yellow pages)

If yellow pages is active on your system (a process called *ypbind* is running), the /*etc*/*hosts* file is ignored and need not be modified.

Printing

To get printed output of schematics and other information out of ADS, the system must have access to a printer or printers capable reading PostScript Level 1 format. The procedures for producing both A and B–size output need to be inserted in the designer's image, with the **configure>printer** commands.

Usually the procedure will be either an *lpr* command on the workstation, or an *lpr* on some other UNIX machine. Ideally, line printer daemons would be installed on the workstation itself for accessing a laser printer. As an alternative, *rsh* is used to start the printer daemon on another host. In the latter case, a remote shell is used:

```
cat filename | rsh tekig9 ipr -Plp99
```

Customer Accounts and Files

The customer accounts and file system are setup as follows:

1. If acceptable, change the *umask* of all users to 022 for read access to each others' files. This is valuable when troubleshooting problems, because the customer need not be bothered with changing permissions on files support people may need to access. Those permissions will also be needed for customers to share files.
2. All users should also have the directory */usr/local* in their path, set by the environment and shell variables *PATH* and *path*. This is preferred over

having the *~cae* directories in the path. If the directory */usr/local* does not exist, create it while logged in as root. If the workstation is using a directory other than */usr/local* for non–standard commands, make sure that directory is in the paths, and there is no need to do the above */usr/local* operations.

3. Configure the customer account to start Openwindows automatically. Add the entries we supply in *~cae/EXAMPLE.Xdefaults* to the customer *~/.Xdefaults.* If necessary, set the environmental variables OPENWINHOME and LD_LIBRARY_PATH.

4. Create the account *cae*. Assign it user–ID (*uid*) 262. This is the third field in the */etc/passwd* file. Also, assign the account a login shell of *csh*. Log in temporarily (**su cae**) to set the password, then exit to *root*. Please tell the ADS support person the password so he/she can set it again. If the *~cae* account already exists but the *uid* must be changed, all files owned by the new uid must also be changed:

   ```
   cd ~cae/etc/chown ~cae. *.??* */* */.??*
   ```

5. Run the Cshell script *caeLinks* supplied in ~cae. This sets up links to the ADS4 and TekSpice commands and the man pages from */usr/local*. For a different directory, refer to the comments at the beginning of the script and change the value of the *csh* variable *localdir* in the script.

6. Arrange for daily backups of customer files. This is important to reduce risk of losing work done within all CAD applications.

Setup Procedure– External Tektronix Customers

Network Setup

The network setup will vary depending on the location and hardware.

Printer

To get printed output of schematics and other information out of ADS, the system must have access to a printer or printers capable of reading PostScript Level 1 format. The procedures for producing both A and B–size output need to be inserted into the designer's image with the **configure>print** command within ADS.

Usually the procedure will be either an *lpr* command on the workstation, or an *lpr* on some other UNIX machine. Ideally, line printer daemons would be installed on the workstation itself for accessing a laser printer. As an alternative, *rsh* is used to start the printer daemon on another host. In the latter case, a remote shell is used:

```
cat filename | rsh tekig9 ipr -Plp99
```

Customer Accounts and Files

The ADS account name *tekcad*, may vary with customer site locations. The instructions assume the use of the olwm (OpenLook) window manager. An optional window manager, mwm (Motif), could also be used.

1. Configure the customer account to start Openwindows automatically. Add the entries we supply in *~cae/EXAMPLE.Xdefaults* to the customer *~/.Xdefaults.* If necessary, set the environmental variables OPENWINHOME and LD_LIBRARY_PATH.
2. Create the account *tekcad*.
3. Run the Cshell script *caeLinks* supplied in ~cae. This sets up links to the ADS and TekSpice commands and the man pages from */usr/local*. For a different directory, refer to the comments at the beginning of the script and change the value of the *csh* variable *localdir* in the script.
4. Arrange for daily back ups of customer files. This is important to reduce risk of losing work done with all applications.

Installation Procedures (Support Personnel Only)

Before You Go

Before you go to the customers site:

1. Obtain the IP network address, hostid, and machine name. Request that operations add the machine to the */etc/hosts* file.
2. Supply the hostid, host name and email address to the Tektronix CAD Manager, MS 39–717.
3. Send a copy of this chapter to the System Administrator.
4. Consult with customer on EDIF backup, if necessary. (Relevant only if wishing to start from a previous ADS image.)

After Cae Account is Setup[1]

After the cae account is setup by the System Administrator:

1. Using the command **pstat –s**, check the paging space on the target workstation. This should show at least 83 megabytes of swap space.
2. Check the umask on the target machine to make sure all the files shipped will be readable and executable by the users. The *~cae* umask must be 022 (or 22) or the installation script will not work.
3. Determine if *caeLinks* has been run; */usr/local/tekspice* will be linked to */u/cae/bin/tspice* if it has run.
4. If this is a version upgrade for an existing customer be sure the EDIF backup is ready.

Installing ADS and TekSpice

The procedure for setting up ADS and TekSpice 2 is as follows:

1. Make sure there is at least 25 megabytes of diskspace available for the ADS installation files. If space is limited, the files can be shipped omitting the ADS image.
2. Login as caedev and install ADS and TekSpice files using the installation script. The easiest method is to modify the *~/RunCAXpro.ads4* script used for the previous installation. Copy this script, comment out the last line, and modify the workstation name.

The following is an example of an installation script that omits the ADS image and sends the 4d0 version of ADS to workstation **wu**, including TekSpice2 and the capabilities file (there are no carriage returns in this command line):

[1] **cae for internal Tektronix customers, tekcad for external Tektronix customers.**

```
csh -vx devsys/CAXpro.ads4 wu 4d0 makeversions -cap
cap55425652 -v 1g0 -v2 2b1
```

3. For first time ADS installations, the libraries need to be distributed. Look for an appropriate *RunCAXpro* script, such as *RunCAXpro.ICO*, and others ending in *VENDOR*, and *TekSpice2*. If this completes successfully setup is complete.

Completing Installation at Customer Site

Once ADS and TekSpice are installed the following details need completed:

1. Check that there is enough disk space for the image on the customer's account—at least 15Megabytes.

2. Login to the customer's account and create the ADS directories:

```
~/ads
~/ads/SimulationResults.ADS
~/ads/Genlib.ADS
~/ads/Scratch.ADS
~/ads/lib
```

3. Modify the *.cshrc* and *.login* files.

   a. Check the .cshrc file PATH statement. It should include the paths: */usr/etc*, and *$OPENWINHOME/bin*. Type

   ```
   . source
   ```

   to re–execute the .cshrc file and put these changes into effect. ADS uses those two paths to run *pstat –s* and *xrdb –symbols*. These programs extract information about the customers workstation when submitting a bug report.

   b. Modify the .cshrc file to unlimit core dump files and an alias for starting ADS as follows:

   ```
   limit coredump unlimited
   alias ads4 /u/cae/bin/st ads4.im
   ```

   Note that the ".im" suffix is automatically added.

   c. Remove the *cd* command from the .cshrc file. Aliases involving *cd* are permitted.

   d. The *stty* command should be in the .login file.

There are sample files in the cae account for additional help: see *EXAMPLE.cshrc* and *EXAMPLE.login* in *~cae.* These are available after shipping via **CAXpro.ads4**.

4. Check that the customer has Openwindows running. The environmental variables: LD_LIBRARY_PATH and OPENWINHOME should be set in the .cshrc file.

5. Check that OpenWindows parameters either required or recommended in the *EXAMPLES.Xdefaults* are installed in the customer *~/.Xdefaults* file.

6. Copy the ADS image from the ~cae directory into the customers *~/ads* directory as *ads4.im*. If the image was not installed in the ~cae directory by the CAXpro script, it can be copied across the network using **ftp** command in binary mode (bin). It is the most recent image in the *~caedev* directory:

   caedev/lib/ads/4PPS

7. Start up ADS (window manager must be running at this point). This exercises the interpreter and checks that the link and alias are proper.

8. Change settings in the **configure > system** panel. Start by clicking on **factory reset**. Of particular interest are the settings for printing and simulator versions.

9. To enable remote simulation, add a line in the *~/.rhosts* file for the remote machine. The line contains only the remote machine name and account name there, separated by a space. Then have the customer log onto the remote machine and update the *~/.rhosts* there to identify the local machine. After this is completed, update the **configure > simulator** panel.

10. Update the **configure > printer (graphics) script** panel. The Postscript file (”out.gpr” by default) created by a print operation is stored in the directory ~/ads/Scratch.ADS. The script created in debug mode, called *hardcopy.exec* may be helpful to debug problems. Test the printer installation by making test copies with all the paper sizes.

11. Configure text printer.

12. Check that email for bug and enhancement requests will work by clicking on **configure > bug report system,** then reset.

13. Help the customer restore circuits from the EDIF files written earlier.

14. Run a simulation and view results.

15. Save the ADS image.

16. Give customer an ADS manual.

Color Allocation Problems Using ADS Concurrently with other software

If your workstation colors are not properly allocated when using ADS and other software concurrently, try the following procedure:

1. Quit one of the programs which uses colors.

2. Re–load colors for ADS. See **Emergency Checker Dialog box**, Reset colors option.

Tutorial

Starting Your ADS Image

This chapter is a tutorial on the operation of the Analog Design System (ADS). By following the step–by–step procedure in this tutorial, you will practice many tasks common in entering and analyzing a circuit design.

The *Window Descriptions* chapter contains more detailed information about windows and commands introduced in this tutorial.

Your workstation should be set up as recommended in the *Getting Started* chapter, System Administration section. The mouse button key functions are described in the *Getting Started* chapter, Startup section under Using the Mouse Buttons.

Tutorial Overview

Specifically, this tutorial will lead you through:

- Getting started, installing process libraries, and beginning a circuit.
- Creating a simple analog differential amplifier schematic.
- Running bias, dc, and ac simulations on the amplifier.
- Probing the simulation results and plotting waveforms.
- Changing the schematic, resimulating, and comparing the results.
- Creating a subcircuit using input and output parameters.
- Sweeping circuit variables.

The ADS Working Directory

ADS is normally run from the sub–directory,

```
~/ads
```

The ADS image file, *ads5.im*, resides in this directory.

ADS manages three subdirectories under the ads directory. Do not modify these directories using UNIX commands.

1. `~/ads/SimulationResults.ADS.` This is where ADS stores files from TekSpice simulations.

2. `~/ads/Genlib.ADS.` This is where ADS stores binary TekSpice files resulting from an EDIF (Electronic Data Interchange Format) write. These are only the files which involve external libraries.

3. `~/ads/Scratch.ADS.` This is used by ADS to store temporary files.

Starting ADS

In order to run ADS, you must first change directories to the ads sub–directory:

```
cd ~/ads
```

To start the ADS software, type:

```
ads5
```

The session begins (after displaying a copyright window) by displaying the ADS **Launcher Window** and the **System Transcript Window** as shown in Figure 2–1. The **System Transcript Window** may appear as an iconified window.

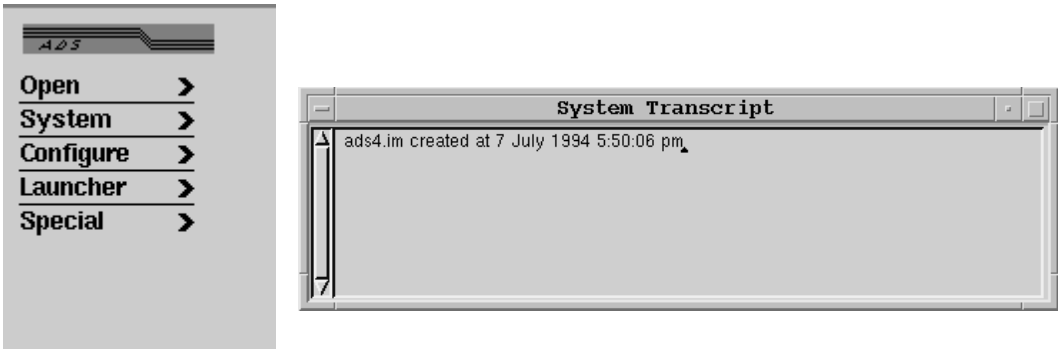


Figure 2–1: ADS Launcher and System Transcript Windows

Install Process Libraries

The first step in using ADS, is to load the appropriate process libraries for your design. This is accomplished from the **Library Browser** as follows:

1. Open the **Library Browser** with **Open>library browser** command from the **Launcher Window**. Use the select or popup mouse button to access the menu (See Figure 1.3 in Chapter 1).

2. ADS now expects you to position and size the **Library Browser**. Move the mouse cursor to the desired position on the screen and *hold down* the select mouse button. This positions the top–left corner of the window. The window can now be sized by moving the mouse, or to use the default window size,

simply release the mouse select button. The **Library Browser** is shown in Figure 2–2. The upper left pane of this browser is the *Process List Pane.* The names of other panes can be found in the Window Descriptions Chapter of this manual.

All ADS windows are opened and placed in the same manner.

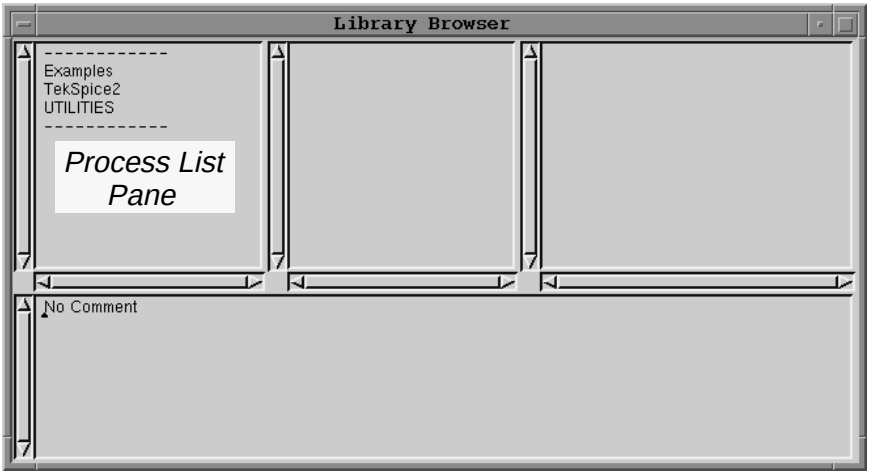


Figure 2–2: Library Browser showing the Process List Pane

3. Next, with the cursor over the *Process List Pane*, select the **install standard process** command from the pop–up mouse button menu. This displays a **Selection Dialog Box** as shown in Figure 2–3.

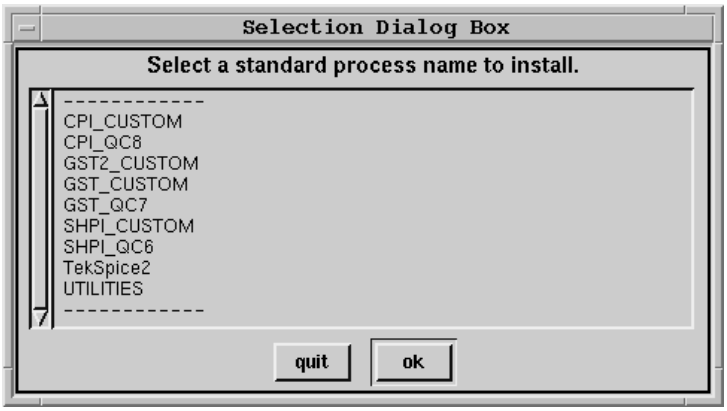


Figure 2–3: Selection Dialog Box for Install Standard Process

4. Select the **SHPI_QC6** process from the list and click on **ok**.

A list of the available libraries for this process is displayed. The processes available for the SHPI_QC6 processes are shown in Figure 2–4.

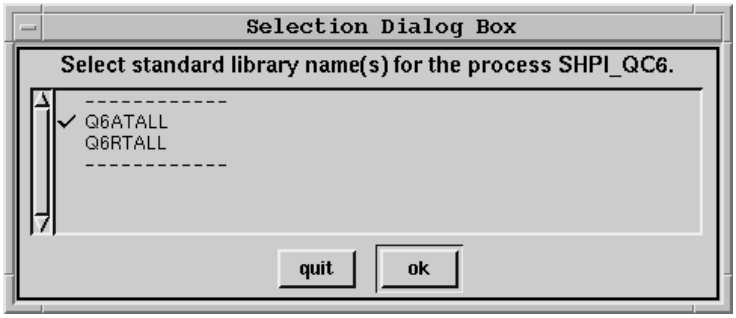


Figure 2–4: Available SHPI_QC6 Libraries

5. Select **Q6ATALL** and click on **Ok**. ADS then reads in all QuickChip 6 schematic symbols. When ADS is finished, the libraries are available for use.

6. Close the **Library Browser**.

Create Circuit Project

Next, create a project for your tutorial schematic. This is done from the **Circuit Browser**.

1. Open the **Circuit Browser** with the **Open>circuit browser** command from the **Launcher Window**.

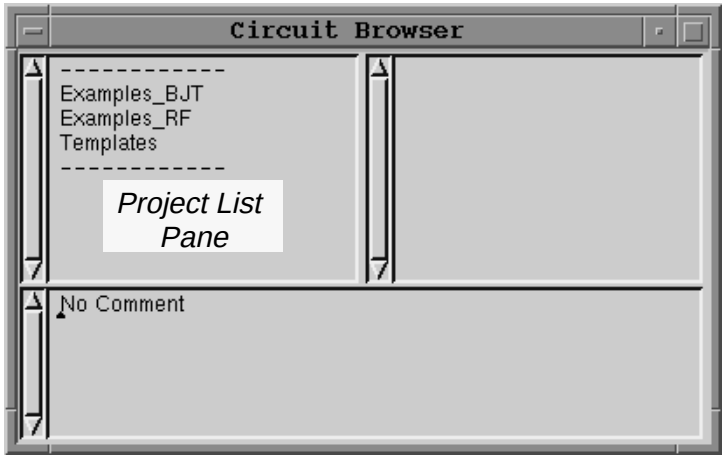


Figure 2–5: Circuit Browser Showing the Project List Pane

2. Create a project for your circuit with the **add project** command accessed from the *Project List Pane* popup menu. The Dialog box shown in Figure 2–6 Appears.

Figure 2–6: Add Project

In the Text window, type in the name **SHPI_QC6** and click on **ok**. This creates an empty project for the tutorial schematic.

Copy Process Template For Schematic

1. Select the **Templates** project from the **Circuit Browser** as shown in Figure 2–7.

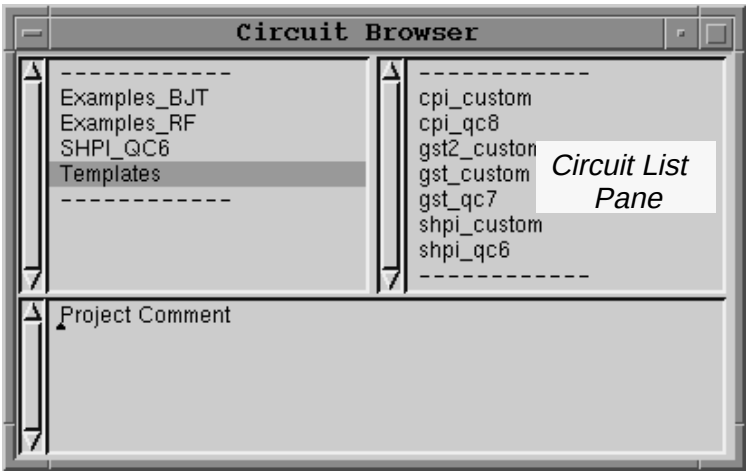


Figure 2–7: Circuit Templates

2. Select the **shpi_qc6** circuit template from the *Circuit List Pane* and select the popup button command **copy>copy to project**. A Dialog box is opened. When ADS prompts for a project name in this box, select your newly created **SHPI_QC6** project and select the ok button.

3. Then, select your **SHPI_QC6** project from the *Project List Pane* of the **Circuit Browser**. Rename the template circuit to **Tutorial** by selecting it in the **Circuit List Pane** and accessing **rename circuit** command from the popup menu.

You are now ready to begin editing the circuit. Your SHPI_QC6 project in the **Circuit Browser** should look Figure 2–8.

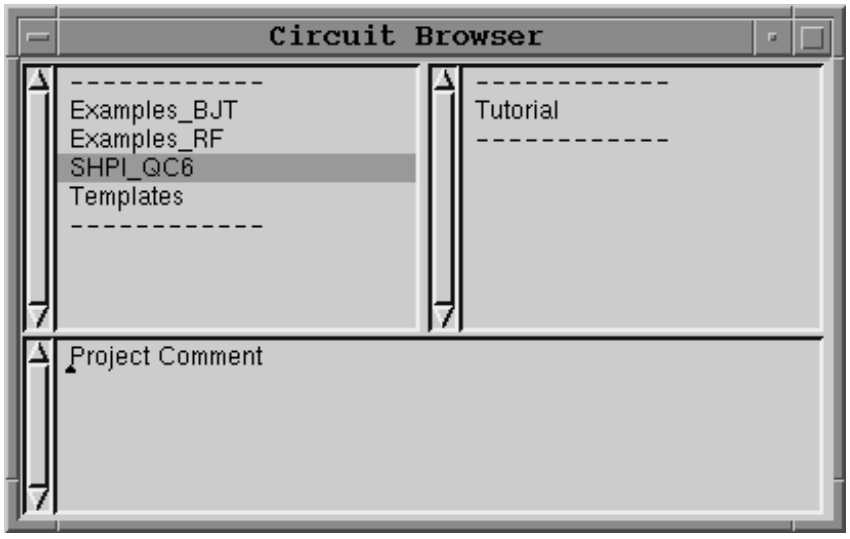


Figure 2–8: Circuit Browser with SHPI_QC6 Project

Creating a Circuit Schematic

In order to analyze a circuit using TekSpice, you must first create a circuit schematic. This section of the tutorial shows you how to create the schematic of a bipolar differential amplifier circuit.

Opening the Schematic Circuit Editor

Open the **Circuit Editor** from the **Circuit Browser** window as follows:

1. With the **SHPI_QC6** project and the **Tutorial** schematic highlighted in the **Circuit Browser**, access the **edit circuit** command from the *Circuit List Pane*.

2. Position and size the **Circuit Editor** window. Figure 2–9 shows the **Circuit Editor** for the Tutorial schematic.

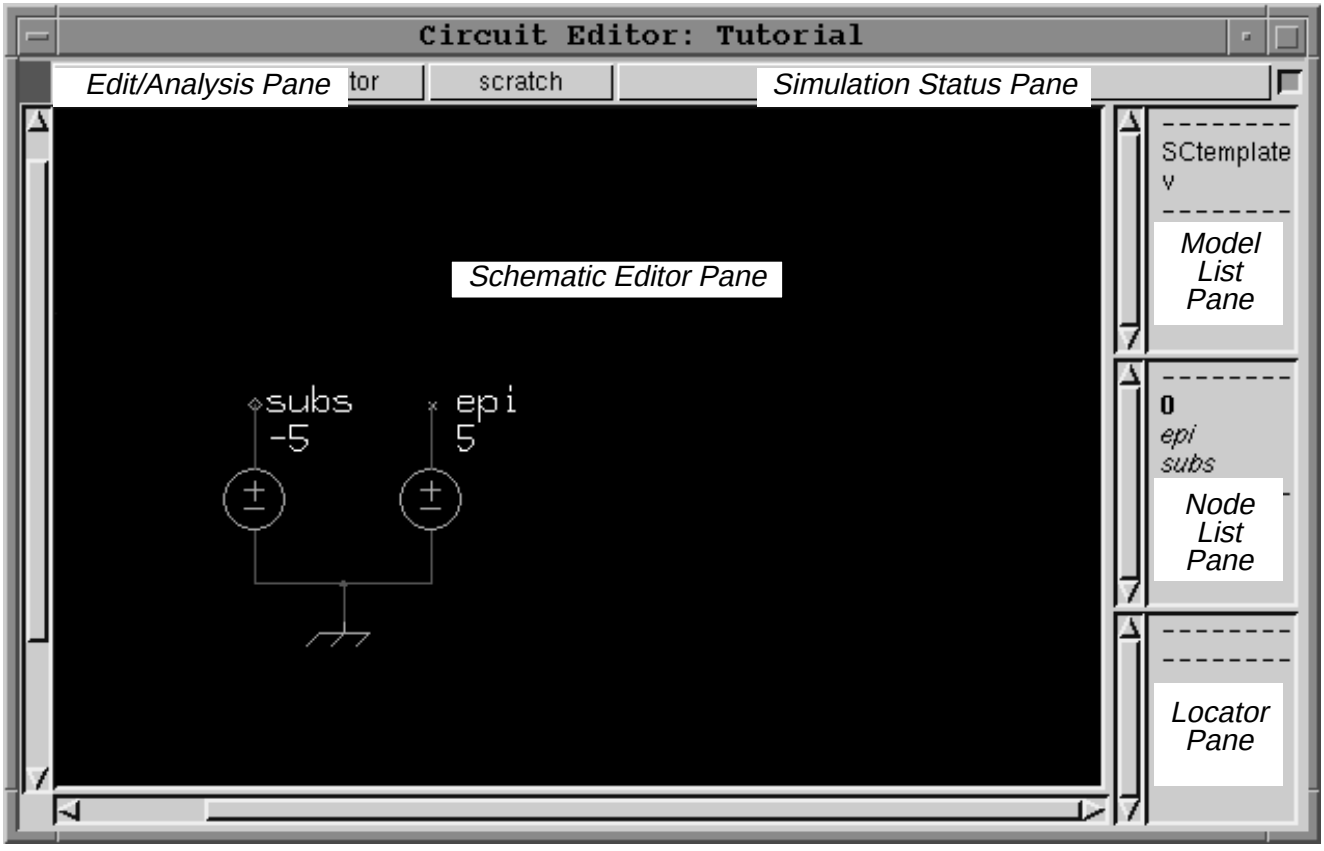


Figure 2–9: Circuit Editor with Pane Names

Choosing the Elements

In the previous section you selected the element libraries you needed for your schematic, now you need to choose the elements from the element libraries used in your schematic. To do this follow the steps below:

1. In the *Model List Pane* with nothing selected, access the **library** command from the popup menu. This displays a list of elements as shown on the Selection Dialog Box in Figure 2–10.

***NOTE**. Be sure that no items in the Model List Pane are selected when accessing the library command. Popup menus in ADS change when items are selected. To deselect an item, simply select it again with the special button.*

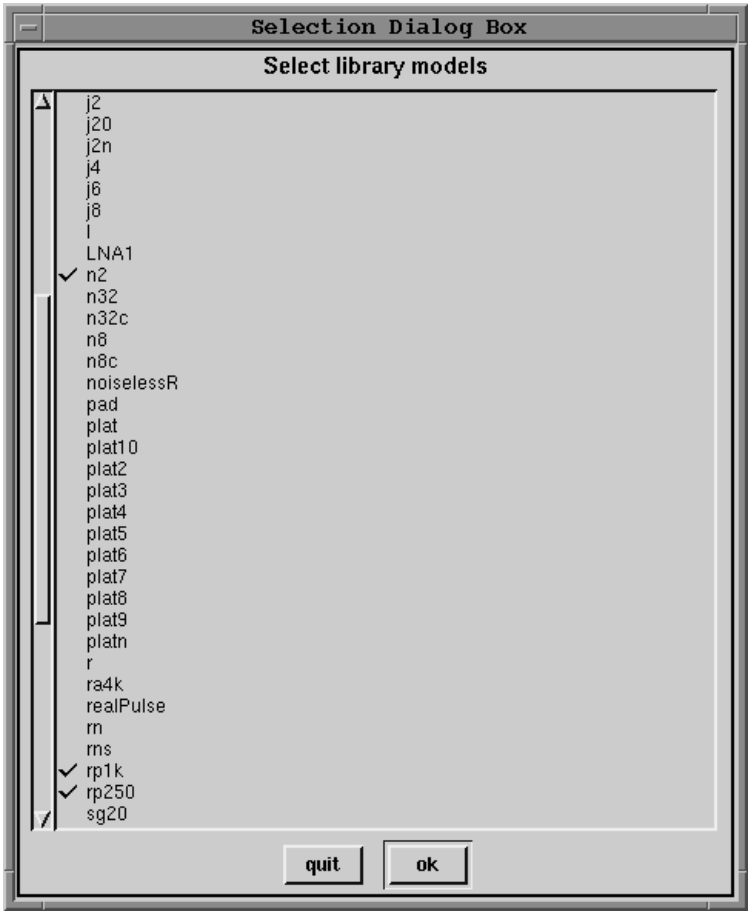


Figure 2–10: List of available elements

2. Search through the list of models and click on **i**, **n2**, **rp1k**, and **rp250**. (See Figure 2–10.) Use the keyboard shift key to select multiple items.

3. Click on **ok**. The selected models now appear in the *Model List Pane* and can be placed in the schematic.

Placing Elements on the Schematic

The next step in creating a circuit schematic is to place (paste) elements into the ADS **Circuit Editor**. Refer to Figure 2–11 for approximate arrangement of components.

NOTE*. The component names in your schematic probably will not match the tutorial names. Don't worry about the exact element names in your schematic. You can, if you wish, rename the elements; the tutorial will later show you how to do this.*

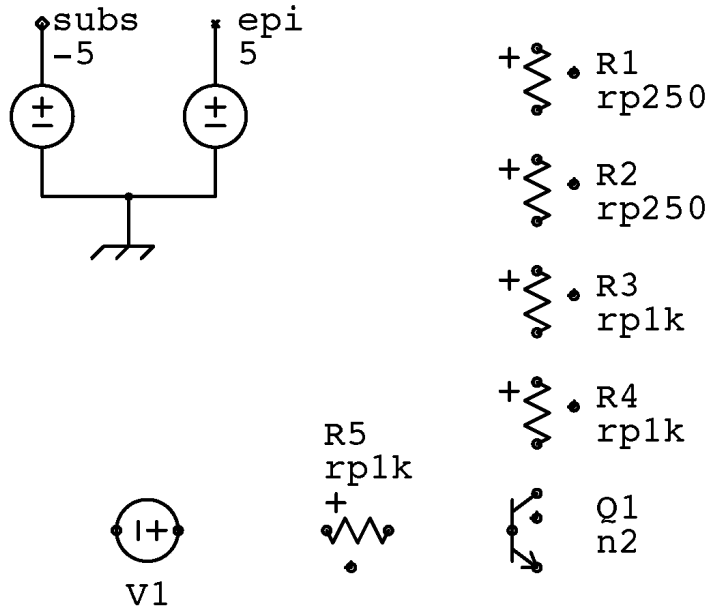


Figure 2–11: Component Layout for 1/2 of Differential Amplifier

NOTE. *The pop–up menu within the Schematic Editor Pane has several window scaling commands:* ***zoom*** *displays a secondary menu consisting of* ***zoom in*** *and* ***zoom out*** *causing a 2X scale change;* ***overview*** *fills the Schematic Editor Pane with the entire schematic diagram and* ***window*** *expands a framed area.*

To place elements in the schematic, follow the steps below:

1. Move the cursor to the *Model List Pane* and click on **n2**.
2. Select the **paste** command from the pop–up mouse button menu. Drag the transistor symbol for **n2** to the *Schematic Editor Pane* and place it with the select button. ADS assigns a name of **Q1**. Notice that the transistor remains highlighted in red. This means Q1 is selected; editing commands in the popup menu are available to immediately operate on this device.
3. Highlight the model **rp250** in the *Model List Pane.* Paste the two **rp250** resistors R1 and R2.
4. Repeat the previous step and add two **rp1k** resistors R3 and R4.
5. **Paste** resistor R5 from the *Model List Pane*. Before placing, use the popup menu to **rotate>90** and **mirror>left–right** the resistor.

   The plus sign on the symbol indicates the polarity of resistors and other elements.
6. Highlight the model **v** in the *Model List Pane*. **Paste** and **rotate>90** the element and place it to the left of resistor R5.
7. Labels that appear on top of a device may be moved to another location. With the cursor over an element, select the popup menu **move label** command.

NOTE. *Popup menus in the Schematic Editor Pane are position sensitive. The contents of the popup menu vary depending on what is under the mouse cursor. For example, the popup menu over an element accesses commands to operate on elements; whereas, the popup menu over open space accesses generic editing commands.*

Connecting Components on the Schematic

Use the popup button **wire** command to make circuit connections. Notice that while in wire entry mode the cursor changes to a crosshair.

Wire the elements as shown below. The wiring snap in ADS is one half of a grid. To exit wire mode, use the popup **select** command.

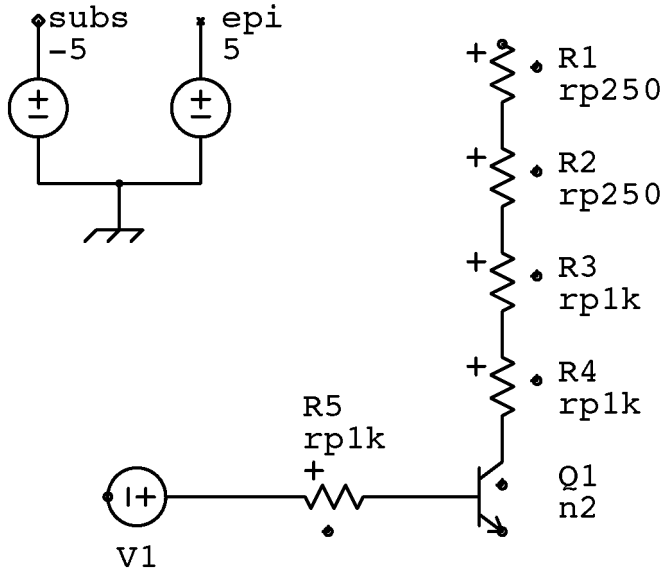


Figure 2–12: Components with wires

Wire resistors R1, R2, R3 and R4 in series as follows:

1. Move the cursor to the first terminal; notice an intensified square. This indicates that the wire will make a connection to the terminal.
2. Click the mouse select button to start the wire.
3. Move the cursor to the next terminal. Click the select button to place the wire segment.
4. To terminate the wire, click a second time at the same location. An alternative method for terminating the wire is to exit wire mode with the **select** command.
5. Complete the remaining connection shown above in the same manner. Note that crossing a wire segment over an existing wire does not make a

connection. To make a connection the wire command click must be on the existing wire.

Setting Element Parameters

Voltage sources have default parameter values assigned at the time they are placed. Use the **Element Browser** dialog box to change these values as follows:

1. Move the cursor over voltage source V1 and access **browse** from the popup menu. This displays the **Element Browser.**
2. In the **Element Browser** *Text Pane*, type dc=dcin as shown in Figure 2–13. The variable dcin allows the dc voltage of V1 to be assigned in the simulation control program.

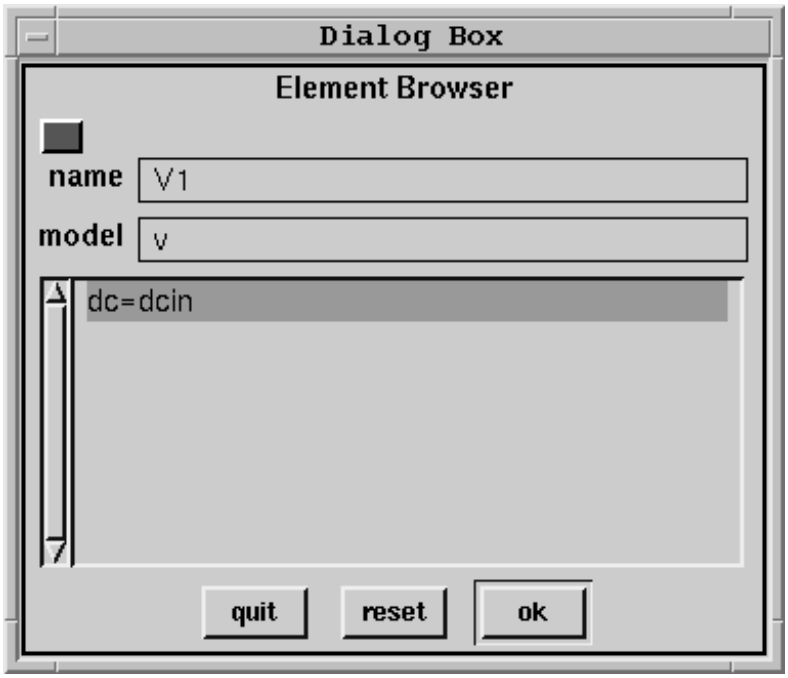


Figure 2–13: Using the Element Browser to Assign Dc Parameter

3. Click on **ok**. Notice that the schematic displays the value of V1 as dcin.

   You can also change the element name (V1) and the model (v) within this browser by selecting the text in the *Name* or *Model* panes and entering the new names.

Connecting Substrate, Epi, and Ground Nodes

The template circuit, **shpi_qc6**, copied to create the **Tutorial** provides the substrate and epi local nodes along with their voltage sources. Connect these nodes as follows:

1. Move the cursor to the *Node List Pane* and highlight ***subs***.
2. From the *Node List Pane*, select the pop–up button **paste** command.
3. Move the cursor over the small substrate node (fourth node) of a transistor. At the intensified substrate terminal, click to paste down the substrate symbol. See Figure 2–14.

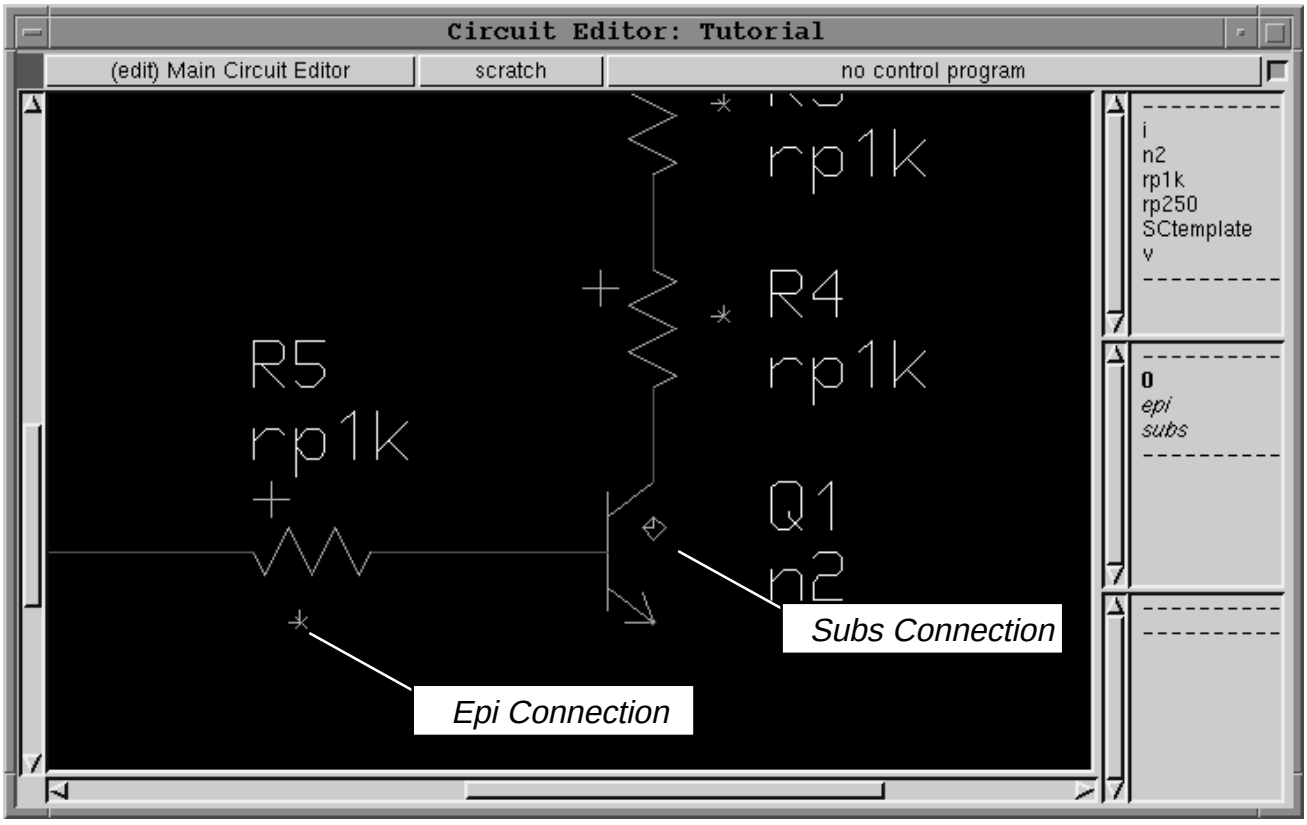


Figure 2–14: Pasting the Subs Node to the Substrate Terminal of Q1

4. Move the cursor to the *Node List Pane* and highlight ***epi***.
5. Select the pop–up button **paste** command.

6. Move the cursor over the small epi node (third terminal) of a resistor. At the intensified epi terminal, click to paste down the epi symbol. Repeat for all of the resistors.

7. **0**, the reference ground potential, is a global node predefined at all circuit levels. It always appears in the node list when you open a **Circuit Editor**. Global nodes appear in bold typeface, local nodes in italics. Select **0** from the *Node List Pane* and **paste** it below V1.

Copying Schematic Elements

Use the following steps to copy the left half of the differential pair to make the right half of the circuit schematic. See Figure 2–15 for placement.

1. If you have not already done this, terminate wire mode and activate selection with the popup menu **select** command. Select all components and connections by dragging the cursor across them while pressing the select button. The selected objects turn red. To add to the currently selected objects, hold down the *shift* key while selecting.

2. Make room for a copy of the selected components using the **zoom** command.

3. Use the popup menu to access **copy** and then **paste**. Before placing the copied portion, use **mirror>left–right** to reorient the components. Click the left button to place the copied items.

   The copied components are numbered automaticically and the parameter values for the copied circuit are the same as the original.

Finishing the Circuit

Finish the differential amplifier circuit shown in Figure 2–15 with the following steps:

1. Pointing at the right–hand voltage source V2, select the **browse** command and change the dc parameter from **dcin** to **–dcin**.

2. Add the collector voltage source V3, and the emitter current source I1, to the schematic as shown in Figure 2–15. Following the same steps you used earlier, wire the two halves of the circuit together.

3. Open the **Element Browser** for V3 and set the DC voltage to 5. Set the DC current value of I1 to 2m (two milli).

4. Paste the ground symbol (node **0**) to connect V3 and I1 to ground.

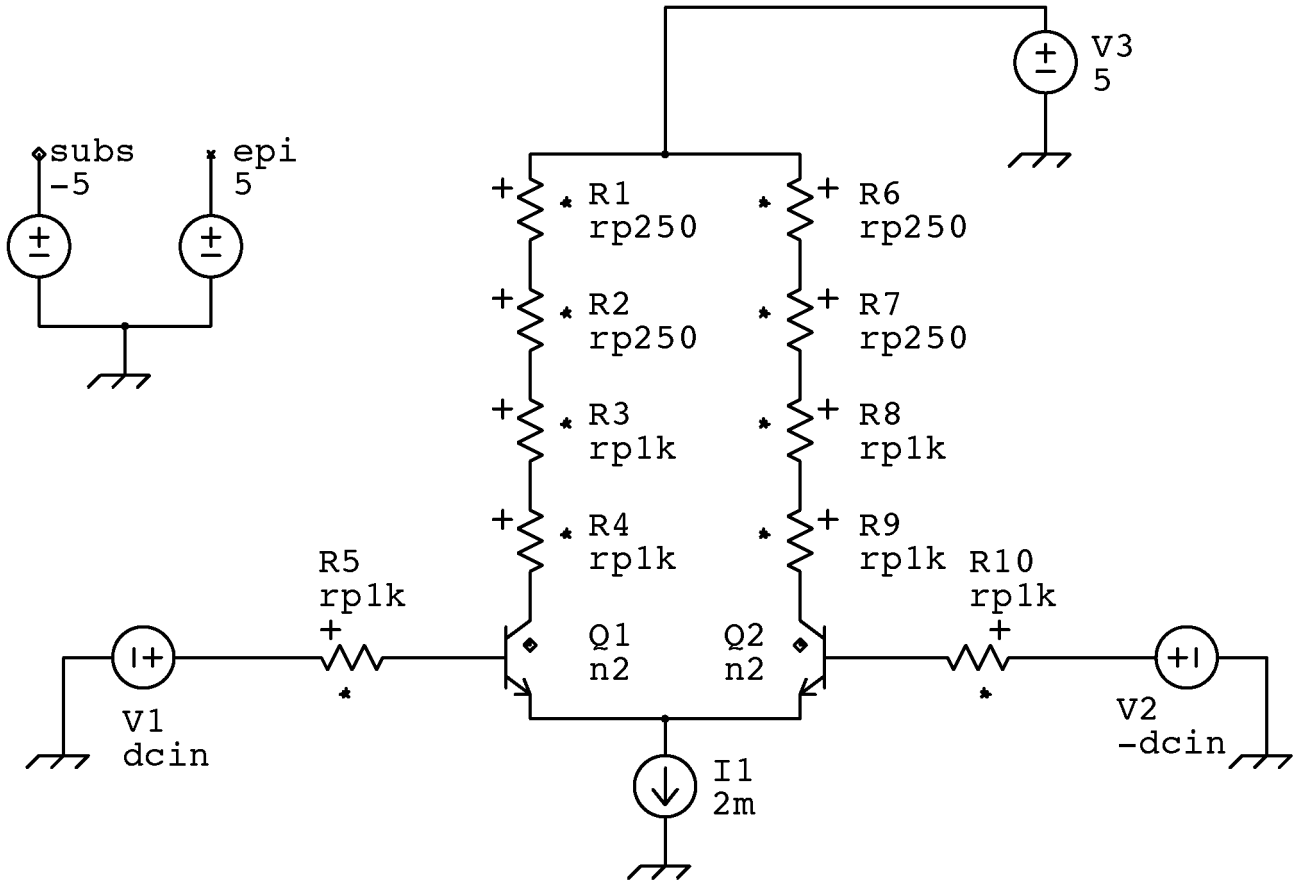


Figure 2–15: Complete Differential Amplifier Circuit

Accepting the Circuit

After making any change in the schematic, the *Accept Pane* (small colored square in the upper–left corner of the Schematic **Circuit Editor**) turns yellow. Before a simulation can be run on these changes, the circuit must be ”accepted”. Use the popup button **accept** command.

If the schematic has open connections, the **accept** command displays a warning message and lists the incomplete nodes in the *Locator Pane*. Select a node name in the *Locator Pane* and use the popup **locate** command to zoom the mouse to the error.

Saving the ADS Image

Once all schematic connection errors have been corrected, you should save your work.

ADS stores your circuit data and current design environment, including all opened window sizes and positions, to an ADS image file. The default name of this file is *ads5.im* located in your ads directory. To save your image use the following steps:

1. From the **Launcher** access the **Special>save as** command. Figure 2–16 shows this pop–up menu.

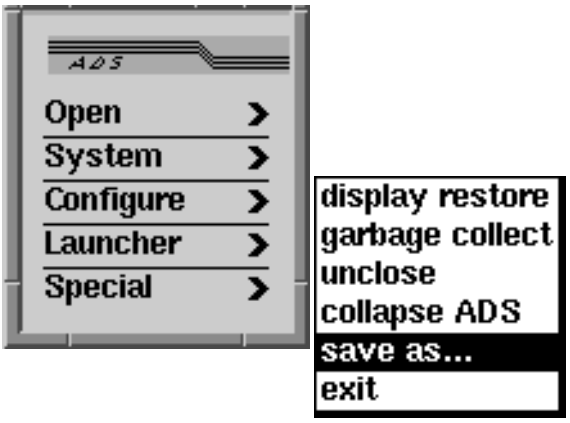


Figure 2–16: Saving ADS Image Using Launcher Command

2. Use the default file name, *ads5*, by clicking on **ok** in the resulting dialog box. (ADS assumes the suffix *im* for image files)

Simulating Your Circuit

Once the circuit schematic is complete, you can run the circuit simulator, TekSpice to determine the ac, dc or transient behavior of your circuit. This part of the tutorial shows how to run simple bias, dc, and ac simulations.

Running a Bias Analysis

Simulations and simulation results are controlled and managed from the **Control Program Browser**. Simulate your circuit by following the steps below:

1. Move the cursor to the *Simulation Status Pane* of the **Circuit Editor.** Access the **edit control programs** command from the popup button menu. Position and place the **Control Program Browser**. See Figure 2–17.

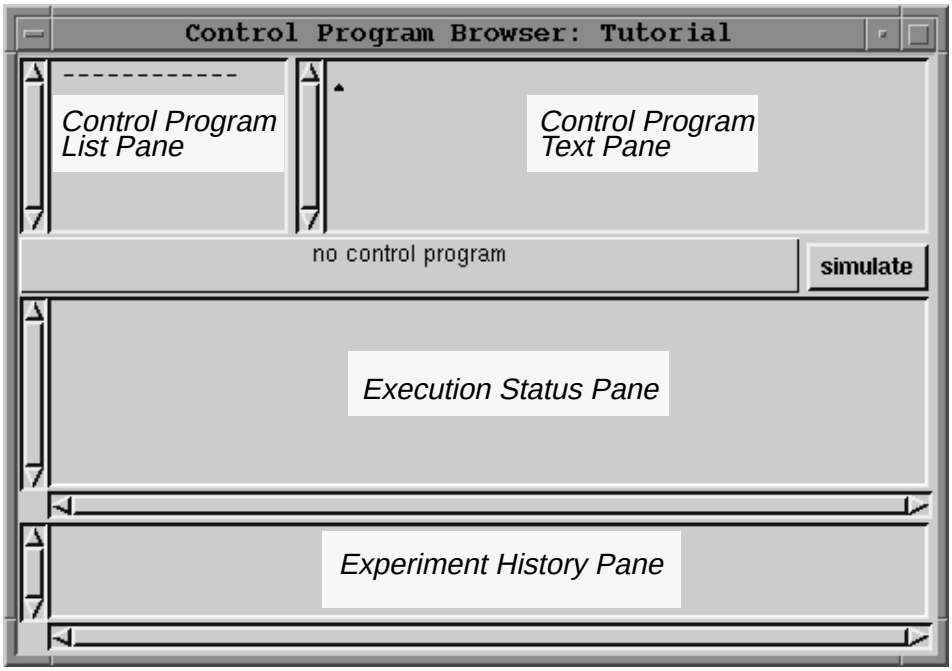


Figure 2–17: Control Program Browser

2. From the *Control Program List Pane*, execute **add bias** from the popup menu. This inserts a default bias analysis into the browser as shown in Figure 2–18.

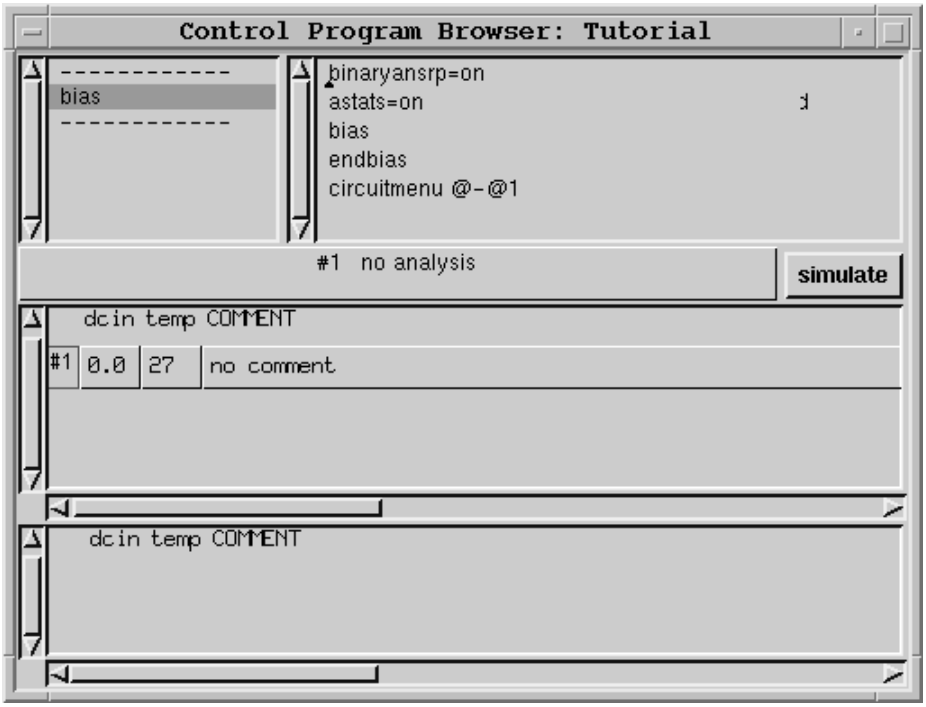


Figure 2–18: Control Program Browser with Default Bias Control Program

3. Simulate the circuit with all default values (dcin=0, temp=27) by clicking on the **simulate** button in the **Control Program Browser**.

4. The **Simulation Status** dialog box opens and tracks the simulation progress, and displays status and error messages. See Figure 2–19. This window automatically closes when the simulation is complete, and transfers relevant messages to the **System Transcript Window**.

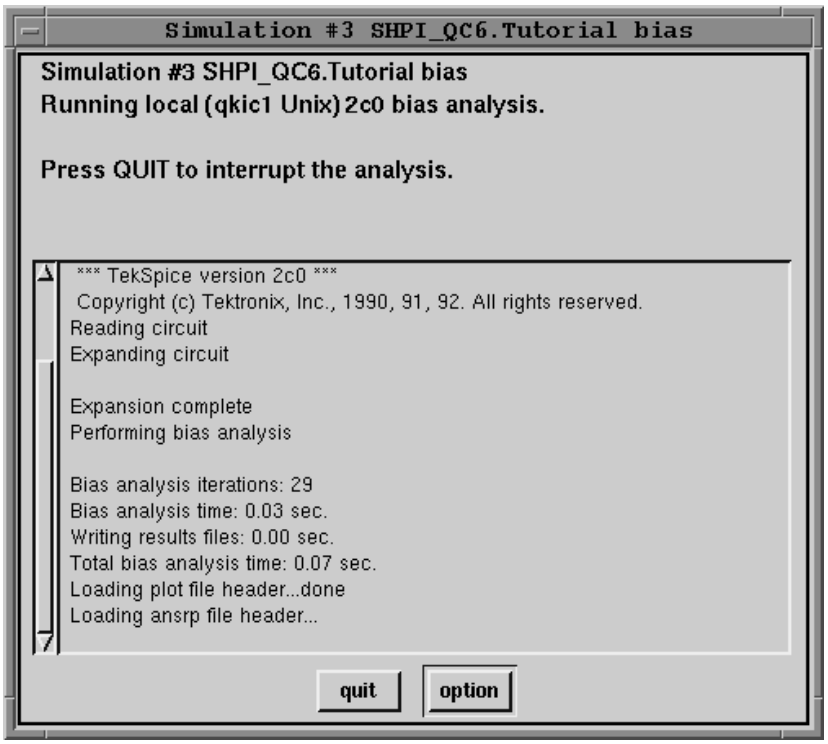


Figure 2–19: Simulation Status Dialog Box

Examining Analysis Results

After the simulation is complete, results can be probed directly off of the schematic.

***NOTE**. Running a simulation automatically switches the **Circuit Editor** to analysis mode. Analysis mode changes all Circuit Editor menus to analysis commands. Remember that popup menus in the Schematic Editor Pane are position sensitive. The contents of the popup menu vary depending on what is under the mouse cursor.*

Follow these steps to probe the results from the simple bias analysis:

1. Examine the bias voltages by pointing at a wire connected to a transistor collector and selecting the pop–up menu **bv()** command. Move the cursor to position the value and click to place it. Verify that the circuit operating points are reasonable.

2. Point to one of the collector resistors and select the pop–up button **functions>I** command as shown in Figure 2–20. Move the cursor to position the resistor bias current text and click to place it. Try other output functions, such as **functions>R** to obtain the calculated value of the resistor.

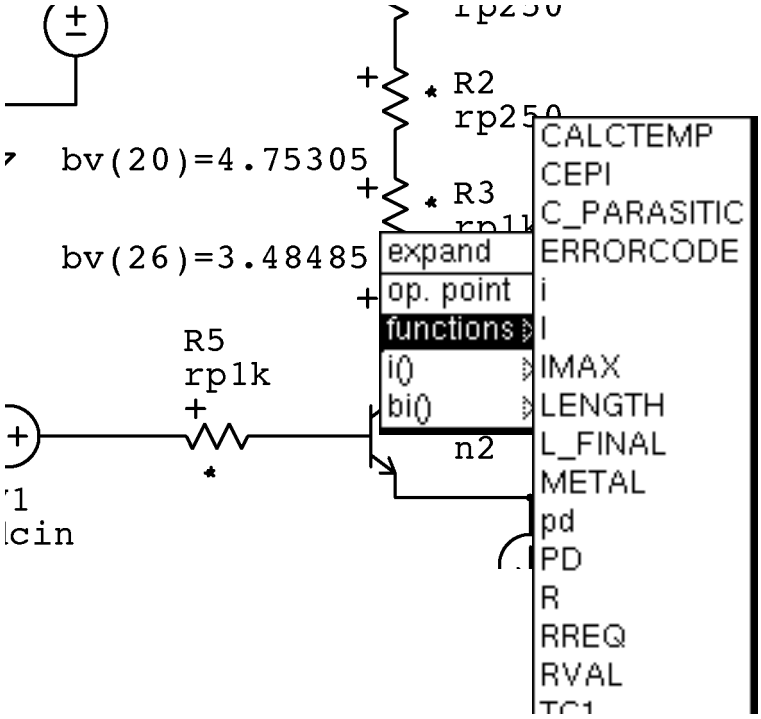


Figure 2–20: Viewing Output Parameter for Library Models

3. Examine some of the output function for the n2 transistors. Transistor models have many output functions defined, such as terminal currents, transconductance, and beta.

Changing the Input and Running Another Analysis

The following steps show how to change the value of dcin:

1. In the *Control Program Text Pane* enter the assignment statement dcin=50m before the bias command line as shown below.

```
clear
 dcin=50m
bias
endbias
circuitmenu @–@1
```

   While still in the *Control Program Text Pane*, select the pop–up button **accept** command.

2. Click on the **simulate** button to re–simulate the circuit.

3. After the simulation is complete, notice that **50m** is displayed as the value of dcin in the *Execution Status Pane*. Return to the **Circuit Editor** window.

4. Look at the collectors of the transistors to determine how the $\pm$ 50 millivolt differential pair operates:

   a. With the cursor over an empty circuit area, select the pop–up menu **functions>@–@1** command. The cursor changes to a crosshair. This activates a two point differential probing command.

   b. While pointing at the collector wire of Q1, use the pop–up button menu to select the **bv** command.

   c. Do the same for the collector of Q2.

   d. Click the select button to paste down the differential bias voltage text.

Running a DC Analysis

The dc analysis sweeps the dc input voltage sources. The dc analysis is accomplished by the following steps:

1. In the **Control Program Browser**, deselect the **bias** control program in the *Control Program List Pane* by clicking on it.

2. Select the pop–up menu command **add dc**. This adds the default dc control program to the browser.

3. In the *Control Program Text Pane* change the dc sweep statement to read:

```
dc analysis dcin: –100m 100m 200m/50
```

This will sweep dcin from –100m volts to 100m volts with a step size of 4m volts (200m volt sweep in 50 points). Select the pop–up menus **accept** command from the *Control Program Text Pane* to accept this change.

4. Click on the **simulate** button. Wait for the simulation to finish.

5. Return to the **Circuit Editor**. Over an empty circuit area, select the pop–up menu **functions>@–@1** command. Probe the two collector voltages with the **v** function. Move the cursor to place the *postage stamp* plot and click to paste it on the schematic. See Figure 2–21.

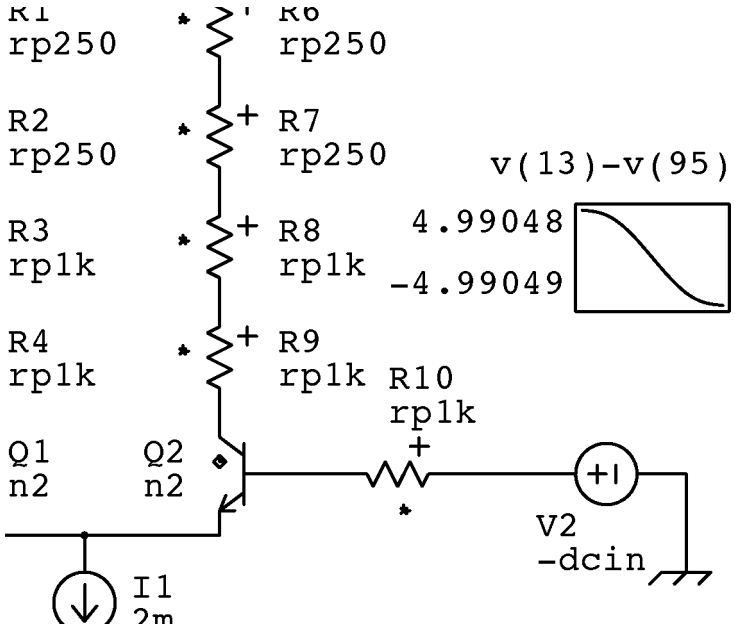


Figure 2–21: Postage Stamp Plot of DC Analysis Simulation

6. Try some probing some of the device output functions such as the n2 collector current, **ic**.

Displaying Waveforms in a Plot Browser

To obtain waveform plots larger than the postage stamps plots, follow the step below:

1. Position the cursor over the postage stamp plot and access the popup menu **plot** command. This opens a **Plot Browser** filled with the postage stamp trace.
2. Position and place the **Plot Browser**.

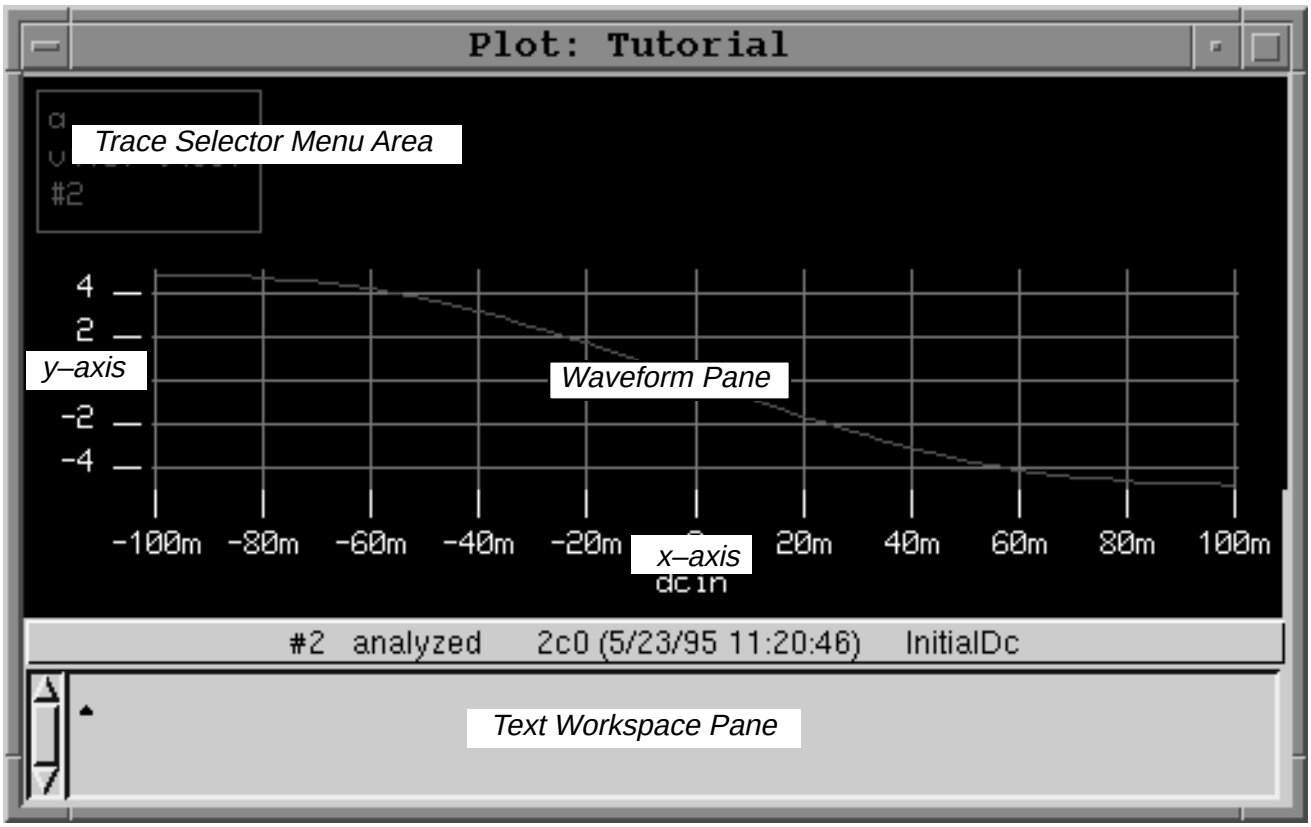


Figure 2–22: Plot Browser

Now that this trace is in the **Plot Browser**, it can be manipulated using a wide variety of waveform processing functions. To see the differential of the output voltage, i.e., the large–signal DC gain vs input, do the following:

1. In the *Text Workspace Pane*, type the expression **dif(a)**.
2. Select the dif(a) text by clicking and dragging over it and execute the popup menu **plot it** command. The new trace for dif(a) has been created and named trace **b**.

3. This new trace is not visible because it does not fit on the existing y–axis. To give the trace a new axis, first select the dif(a) trace box from the *Trace Selector Menu Area*. Then from the *Waveform Pane* select the pop–up **new axis>y** command. Both traces should now be visible as shown in Figure 2–23.

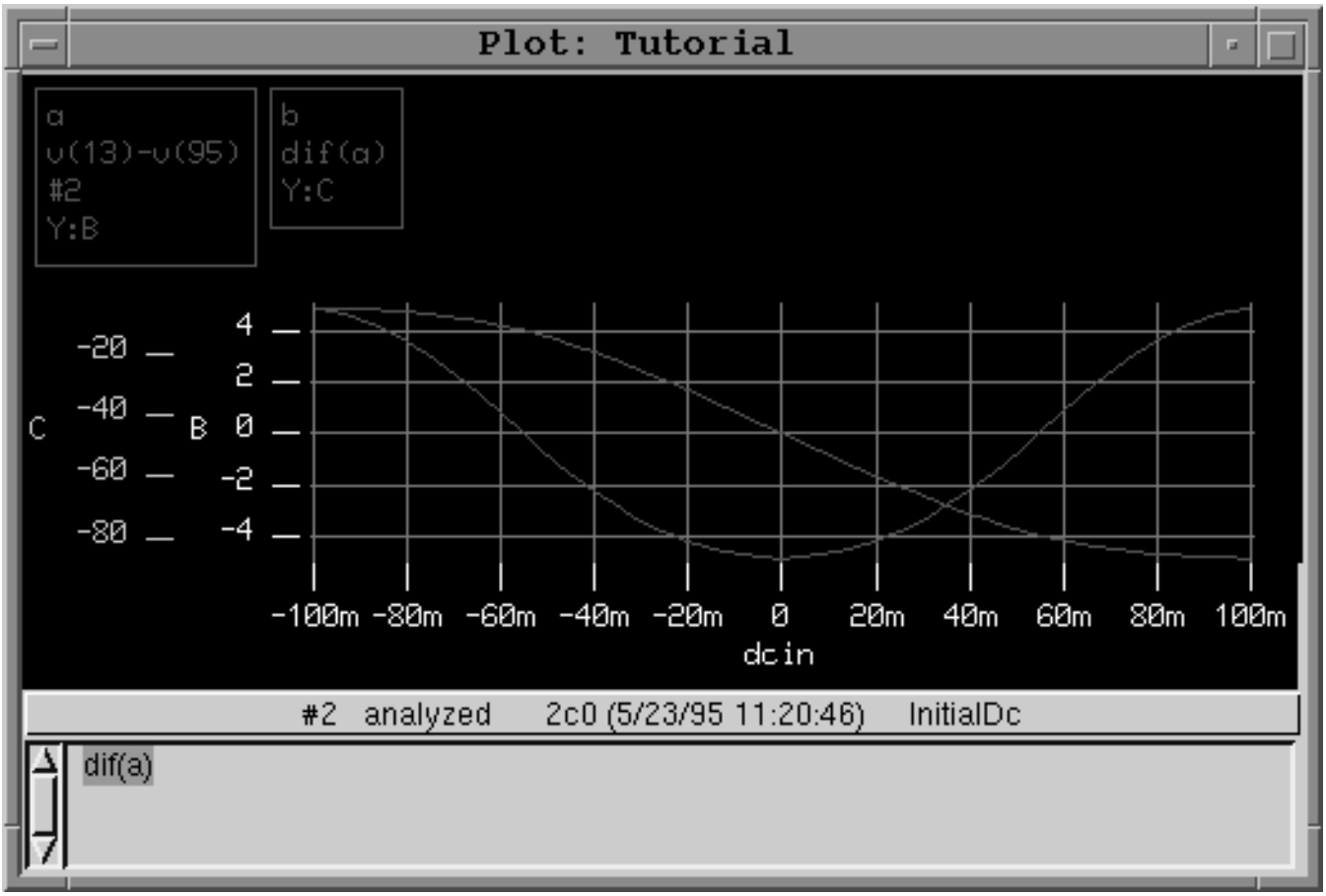


Figure 2–23: Differentiated Plot Trace

Another useful feature allows you to **copy** postage stamp traces and **paste** them into opened **Plot Browsers**. Try this with a different postage stamp plot.

Viewing Operating Point Information

As you have seen, output parameters can be interactively probed and placed on the schematic. The **User Operating Point Browser** provides access to all of these output parameters at once.

The **User Operating Point Browser** is opened from the **Circuit Editor** while in the analysis mode. This data is viewed using the following steps:

1. From the *Schematic Editor Pane* (not pointing at the circuit), select the pop–up menu **open>user operating point table** command. Size and place the browser. See Figure 2–24.

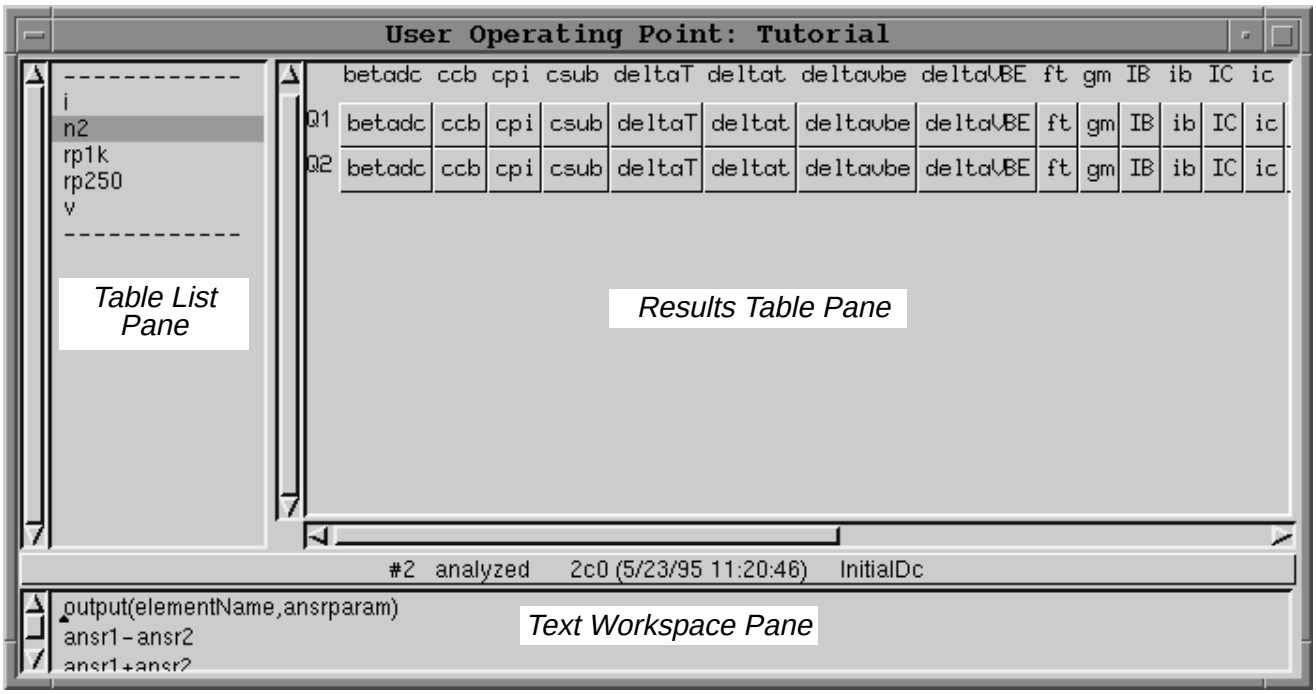


Figure 2–24: User Operating Point Table with Variable Names

2. From the *Table List Pane*, select device **n2**. The *Results Table Pane* in this window now contains output variable names for the n2 devices (Q1 and Q2).

   The table now provides direct access to all of the n2 output parameters. To display a differential collector current plot, use the temporary storage registers *ansr1* and *ansr2* as follows:

3. In the Results Table Pane, click on **ic** in the Q1 row. The box displays as highlighted in red.

4. Copy **ic** for Q1 to the register *ansr1* with the popup menu **ansr1** command. **ic** for both transistors evaluate to the displayed plots.

5. Copy **ic** for Q2 to the *ansr2* register.

6. In the *Text Workspace Pane*, select the text for **ansr1–ansr2**, and select the pop–up menu **plot it** command. A **Plot Browser** filled with the differential trace appears.

7. To display values for all parameters of Q1 and Q2, execute the **evaluate** command from the *Table List Pane*. Figure 2–25 shows the result.

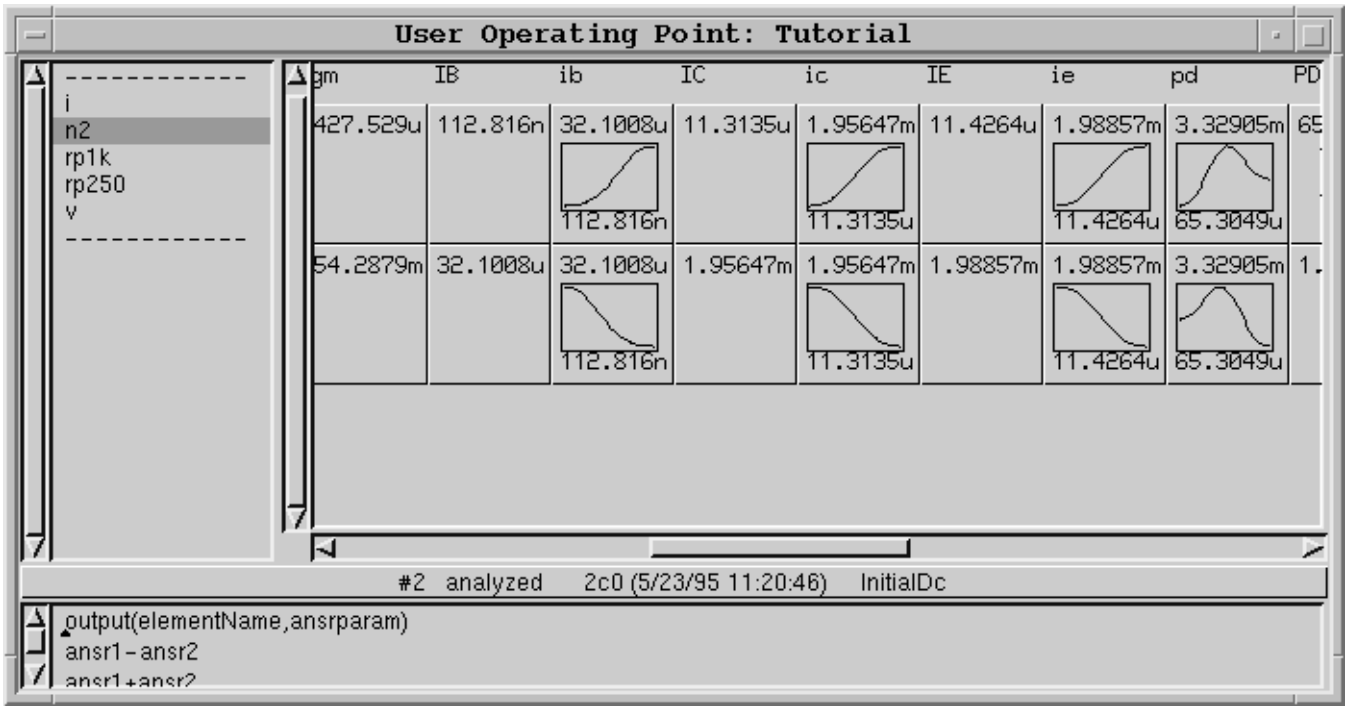


Figure 2–25: User Operating Point Browser with Parameters Values

Saving Results for Later Use

By default, simulation results are overwritten every time a control program is resimulated. To save the above dc results for later comparison:

1. Simulation and experiment results are managed from the **Control Program Browser**. Locate or re–open the **Control Program Browser** for the Tutorial schematic.

2. From the *Execution Status Pane*, select the pop–up menu **move experiment to history** command.

3. Notice that the analysis entry from the *Execution Status Pane* is moved to the *Experiment History Pane*. These results are now saved and will not be overwritten by future dc simulations.

Running an AC Analysis

To simulate the ac response of the differential amplifier follow the steps below:

1. Switch back to *edit mode* in the **Circuit Editor**. The mode can be toggled between *edit* and *analysis* by clicking the left mouse button on the *Edit / Analysis Pane*. Using the **browse** command open an **Element Browser** on the voltage source V1. Add the parameter ac=1. Click on **ok**. Repeat this for V2, also adding acp=180.

***NOTE**. If you do not know the avaulable input parameters for a device, the* **Element Browser** *command* **default parameters** *accessed from the browser Text Pane inserts all default input parameters.*

---

2. Accept these changes with the pop–up menu **accept** command.
3. With no analysis selected in the *Control Program List Pane* of the **Control Program Browser**, access the **add ac** command. In the *Control Program Text Pane*, change the ac analysis parameters to:

   ```
   ac analysis freq: 100k 10g 10 type=dec
   ```

   This will perform an ac frequency sweep from 100k Hz to 10g Hz using 10 points per decade.
4. From the *Control Program Text Pane* **accept** the changes.
5. Click the **simulate** button to run the ac analysis and wait for the analysis to finish. As before, this automatically switches the **Schematic Circuit Editor** to analysis mode.
6. Observe the differential ac gain by:
   a. Point at a collector node and select the pop–up menu **functions**> **mag(@–@1)** command.
   b. Click the left button on the other collector node.
   c. Paste the postage stamp plot.
7. Try some of the other output functions, such as **db**.

End of Tutorial

You have completed the ADS Training Tutorial. Feel free to revisit any of the areas of the program that this tutorial has introduced. Try experimenting with other features and plotting other output functions available on the device models.

Running Electrical Checks (optional)

QuERC, a proprietary electrical rules checker, examines the bias conditions as provided by the circuit simulator. Using a data base of process and device limitations, QuERC identifies potential design problems early in the development process. QuERC checks for:

- Junctions biased above breakdown
- Transistors biased in saturation
- Transistors biased in region of low f_T
- Excessive power dissipation
- Excessive device current
- Incorrect device polarity
- Substrate not at lowest voltage
- Resistor bias modulation

QuERC can be run directly from ADS, or as a stand–alone program. ADS provides a special simulation control program for running QuERC. The procedure is to simulate the circuit with this control program, and then display the QuERC results.

Running QuERC

To create some electrical violations in the differential amplifier:

1. Switch the Tutorial **Circuit Editor** to edit mode.
2. On the current source I1, open the **Element Browser** with **browse**.
3. Change the dc current from 2m to 5m and click on **ok**.

Add QuERC Control Program

From the *Control Program List Pane* in the **Control Program Browser**, run **add querc** to add a QuERC control program.

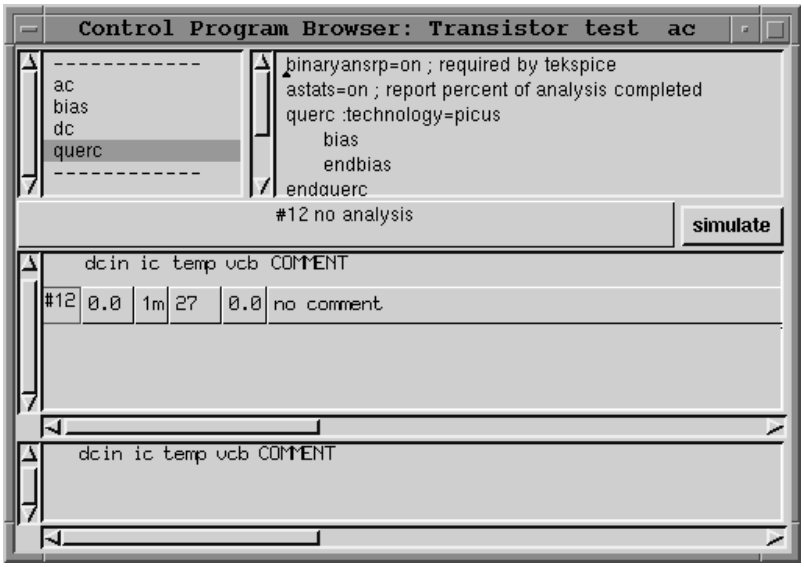


Figure 2–26: Control Program Browser with QuERC Control Program

Set Technology Name

In the *Control Program Text Pane*, change the technology to:

```
querc :technology=qc6
```

Save the change with the popup button **accept** command.

Simulate The Circuit

Click on **simulate** to run a QuERC simulation.

Display QuERC Results

When the simulation is complete, open the **QuERC Results** window by accessing **open>querc text** from the right button popup menu in the *Schematic Editor Pane*. You should see the following errors:

```
Rule 'Saturation', model 'N2':

   NPN transistor saturation. Vbc = 0.5055V > 0.3V.

Rule 'Ft', model 'N2':

   Transistor in region where Ft is falling rapidly. Ic = 2.25mA > 2mA.
```

Close the **QuERC Results** window when finished. For more information on QuERC, or how to run QuERC as a stand–alone program, see the QuERC man page in the *QuicKic User Manual*.

Adding Hierarchy to the Schematic

Defining a Subcircuit

In this section, you will replace the ideal current source (I1) with an IC transistor version of a current mirror. Defining a subcircuit involves the following steps:

- Copy the predefined subcircuit template, **SCtemplate**. As with the circuit template, this subcircuit template contains predefined *epi* and *subs* nodes.
- Create the subcircuit symbol. This can be done by drawing the symbol with lines, rectangles, arcs, and text, or by copying the graphics from existing library symbols.
- Define the subcircuit ports and position them on the symbol.
- Create the subcircuit schematic, including pasting all subcircuit ports.
- Add subcircuit input parameters, output parameters, and local variables.

  Input Parameters: Allow variables to be passed into the subcircuit. The voltage source parameter **dc** is an example of an input parameter.

  Output Parameters: Allow subcircuit simulation results to be probed on the subcircuit symbol. The **functions>I** command used in the previous section is an example of an output parameter.

  Local Variables: Allow local expression to be defined for the subcircuit.

Copy the Subcircuit Template

From the **Circuit Editor**, select **SCtemplate** from the *Model List Pane*. Access the **copy** command from the popup menu. Type the name **isource** in the dialog box and click on **ok**. When ADS asks for a library name, select **Clipboard**. **isource** now appears in the *Model List Pane*.

Create the Subcircuit Symbol

Create the IC transistor current mirror symbol with the following steps:

1. With **isource** still selected in the *Model List Pane*, execute the **edit** command. This opens the **Library Subcircuit Definition Editor**. Position and size the window. See Figure 2–27.

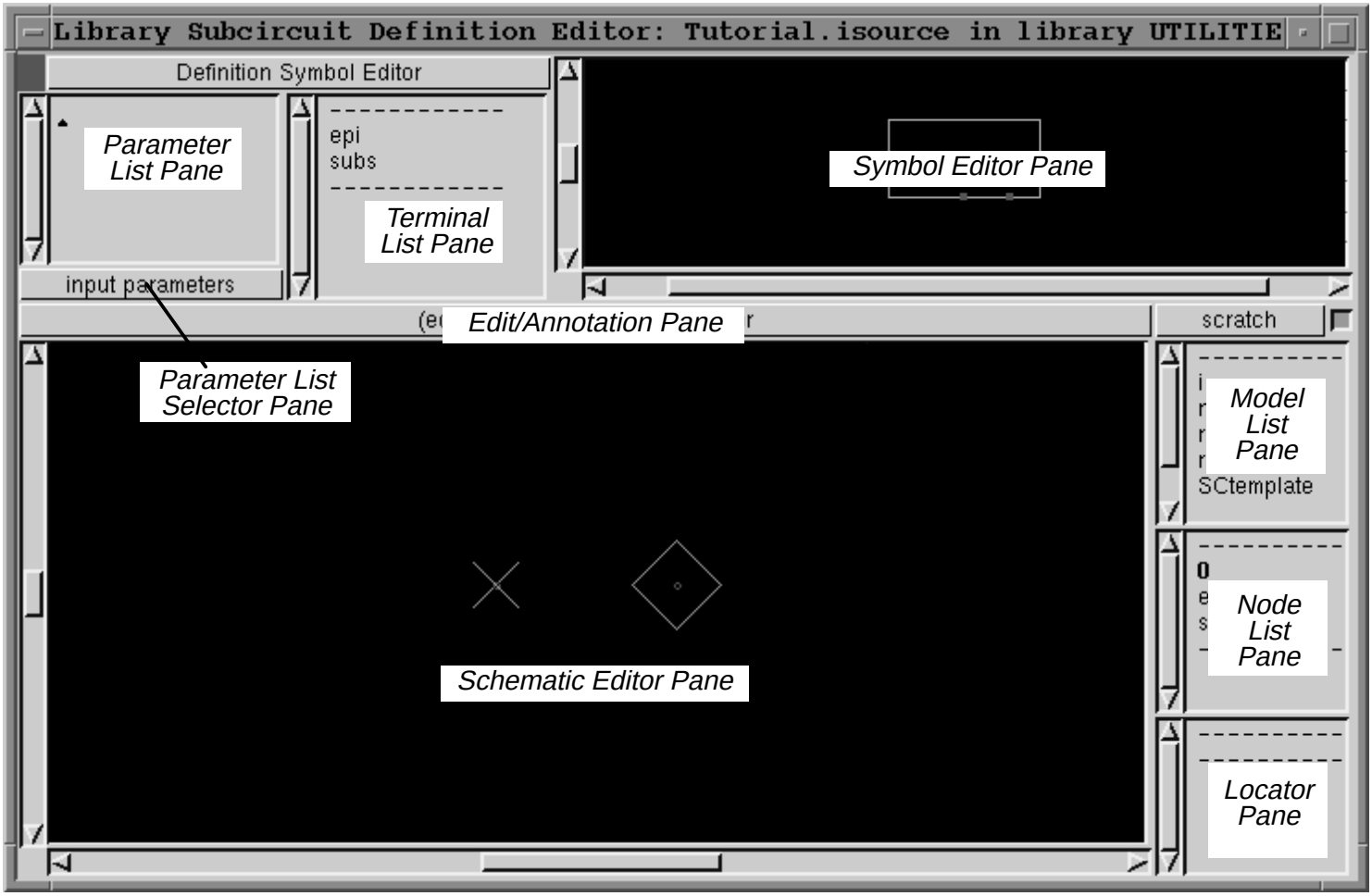


Figure 2–27: Library Subcircuit Definition Editor

2. To create the subcircuit symbol for **isource**, we will copy the graphics symbol from the primitive current source, **i**. Select **i** in the *Model List Pane* and run the **edit** command. Position and size the editor window.

3. Select all symbol graphics for **i** by dragging the cursor over all graphics with the select mouse button down. Execute the popup menu **copy** command. Close the editor window for **i**.

4. Back in the **Library Subcircuit Definition Editor** for **isource**, **paste** the current source symbol graphics in the *Symbol Editor Pane*. See Figure 2–28.

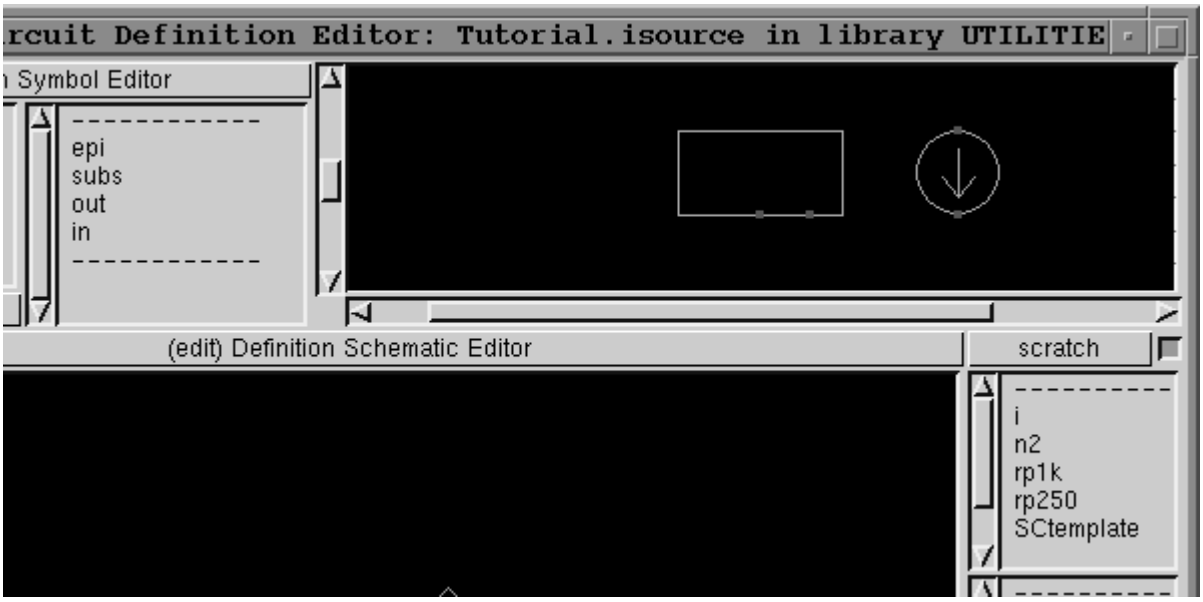


Figure 2–28: Pasting Symbol Graphics

5. Define the subcircuit ports and their order. From the *Terminal List Pane*, run the **add terminal** command. In the dialog box type: **in>out>epi>subs** as shown below and click on **ok**. Notice that the terminal list order updates and that **in** and **out** now appear in the *Node List Pane*.

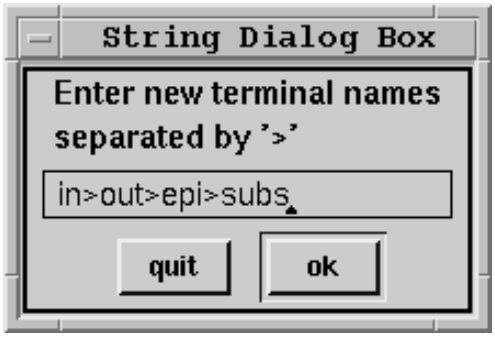


Figure 2–29: Add Terminals Dialog Box

6. Move the epi and subs symbol ports from the template symbol box to the current source symbol as follows:

   a. Select the **epi** port from the *Terminal List Pane*. The port should highlight in the *Symbol Editor Pane*. From the *Symbol Editor Pane*, use the **move** command to reposition the epi port as shown in Figure 2–30.

b. Repeat these steps to move the **subs** port.

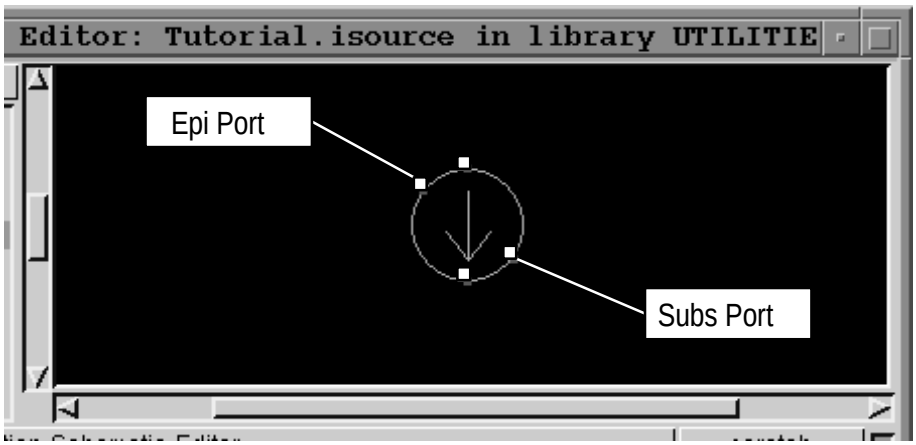


Figure 2–30: Finished Current Source Subcircuit Symbol

7. Finally, **select** and **cut** the old template rectangle symbol.

8. Use the popup **overview** command to view the symbol as shown above.

Create the Subcircuit Schematic

Build the isource schematic in the *Schematic Editor Pane* as shown in Figure 2–31. Remember to **paste** the *epi* and *subs* connections on the resistors and transistors.

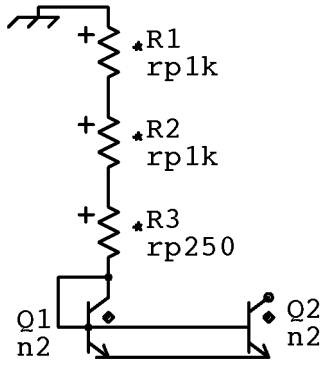


Figure 2–31: Circuit Schematic for Current Source Subcircuit

Add Subcircuit Ports

Select the **in** node from the *Node List Pane* and **paste** it on the schematic as shown in Figure 2–32. Set the correct orientation with the **mirror>top–bottom** command while the **in** port is dragging with the cursor. Repeat to paste the **out** port.

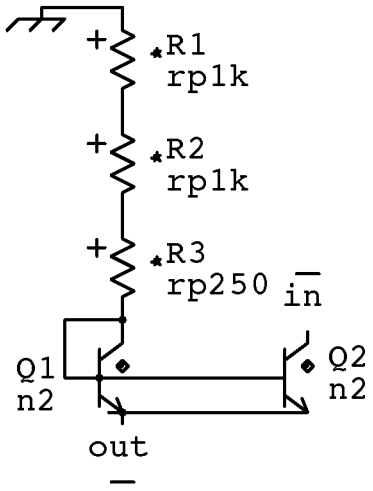


Figure 2–32: Complete Current Source Subcircuit with Ports

Define Output Parameters

Set the *Parameter List Selector Pane* to **output parameter** by clicking twice on the pane with the select mouse button. Enter the following two output parameters in the *Parameter List Pane*.

```
biout=output(Q2,IC)
iout=output(Q2,ic)
```

Here we have defined the current for the current source as the collector current of Q2. From this pane access the **accept** command to accept the new parameter definitions.

Accept the Subcircuit

In the *Schematic Editor Pane*, **accept** the subcircuit schematic. Correct any errors as you did with the Tutorial schematic. Close down the isource **Library Subcircuit Definition Editor** when finished.

Using The New Subcircuit

Now modify your original circuit to use the new isource subcircuit. Return to the **Circuit Editor** for the **Tutorial** schematic and follow the steps below:

1. Make sure the editor is set to *edit mode*. Select the current source **I1** and the connecting ground symbol. Delete them with the **cut** command.
2. Select **isource** from the *Model List Pane.* Use **paste** to place an instance of **isource** as shown in Figure 2–33.
3. **Paste** the ***epi*** and ***subs*** nodes on the appropriate ports of the isource symbol.
4. Add a voltage source (**v**) below the current source and using the **Element Browser** set the **dc** input parameter to **–5**. See Figure 2–33.
5. **Wire** the components and **accept** the changes.

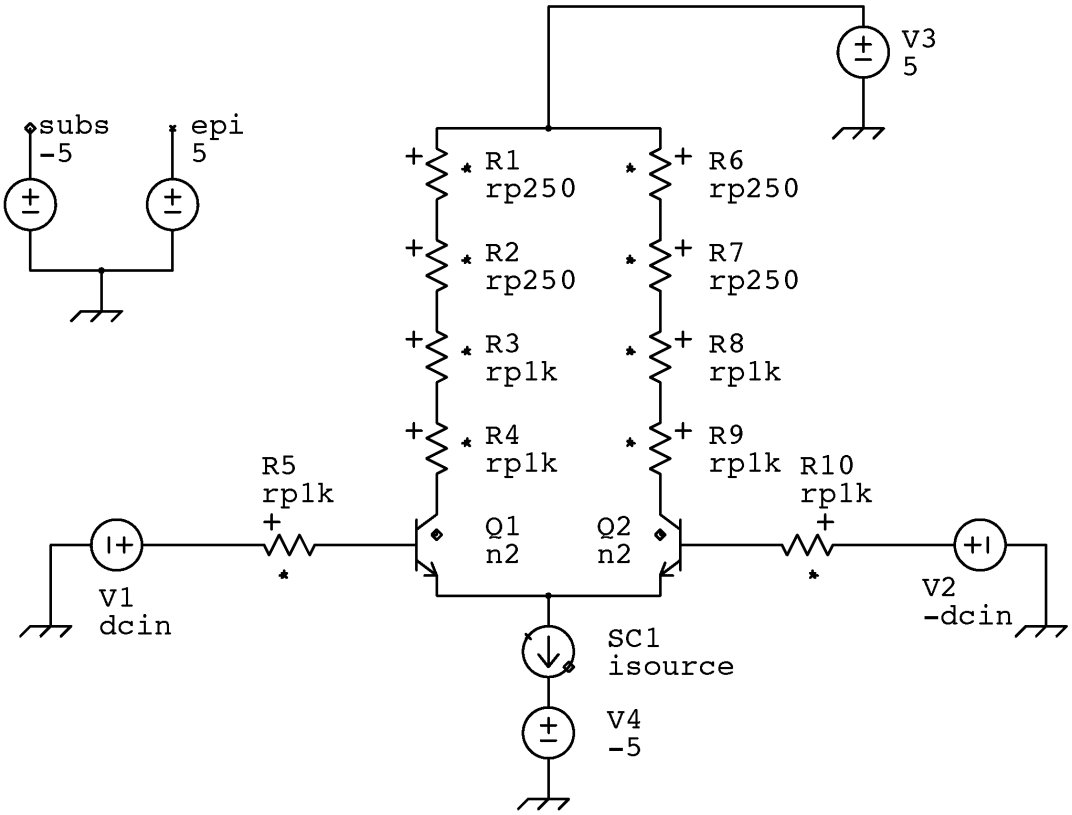


Figure 2–33: Tutorial Schematic Using isource

Simulate The New Circuit

1. From the **Control Program Browser**, select **dc** in the *Control Program List Pane*. Simulate the new circuit by clicking on the **simulate** button. Wait for the simulation to complete.

2. With the cursor over **isource**, access the **functions>biout** command to verify that the bias current of **isource** is close to 2m. Also look at the **functions>iout** output parameter.

3. Probe inside the isource subcircuit by selecting the popup menu **expand** command. This opens an editor window for isource. Examine the currents and voltages in the subcircuit. Close the isource window when finished.

4. Back in the Tutorial **Circuit Editor**, use the **functions>@–@1** command on the transistor collector nodes. Place the postage stamp plot. Now compare this trace with the results from the ideal current source circuit as follows:

   a. With the cursor over the postage stamp, open a **Plot Browser** for the trace using the **plot** command.

   b. From the **Control Program Browser** click on the experiment number of the saved dc analysis. See Figure 2–34.

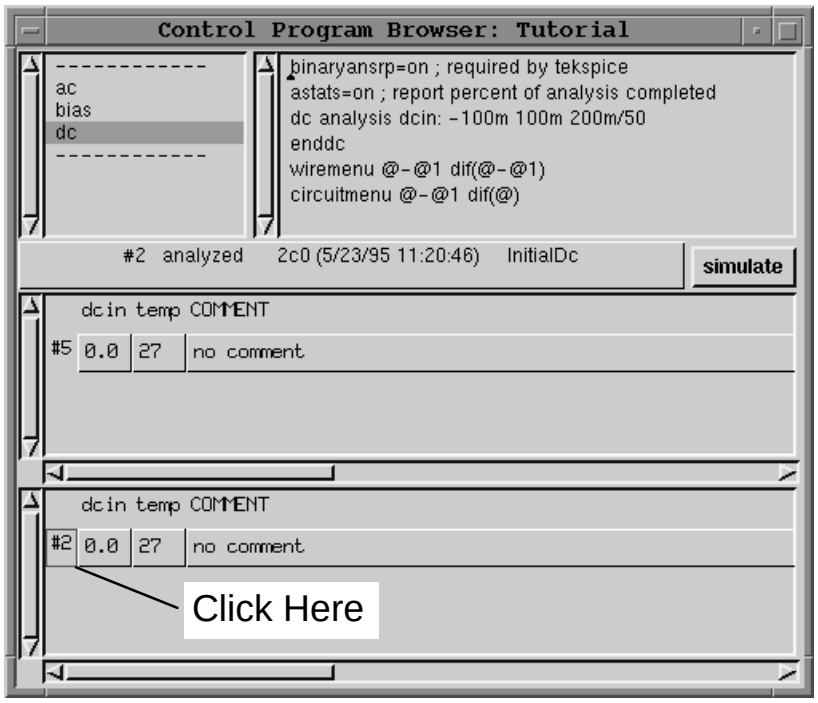


Figure 2–34: Control Program Browser with Saved Experiment

c. With this experiment selected, all analysis probing in the **Circuit Editor** will plot results from the saved simulation. Repeat the **functions>@–@1** command and paste the new postage stamp.

d. With the cursor over the new postage stamp, **copy** the trace and **paste** it into the opened **Plot Browser**. The two traces should match fairly closely.

e. To see how closely, type **a–b** in the *Text Workspace Pane*, select the text and execute **plot it**. Give the resulting trace *c* a new axis with **new axis>y**.

Writing A Layout Netlist For QuicKic (optional)

The *QuicKic Layout With Verification Editor* uses the simulation netlist to lead the designer through the layout process by highlighting nodes and preventing improper connections. Generating a QuicKic netlist from ADS is accomplished with the following steps:

1. Access **info>layout info** from the *Edit/Analysis Pane* of the **Circuit Editor** window as shown in Figure 2–35.

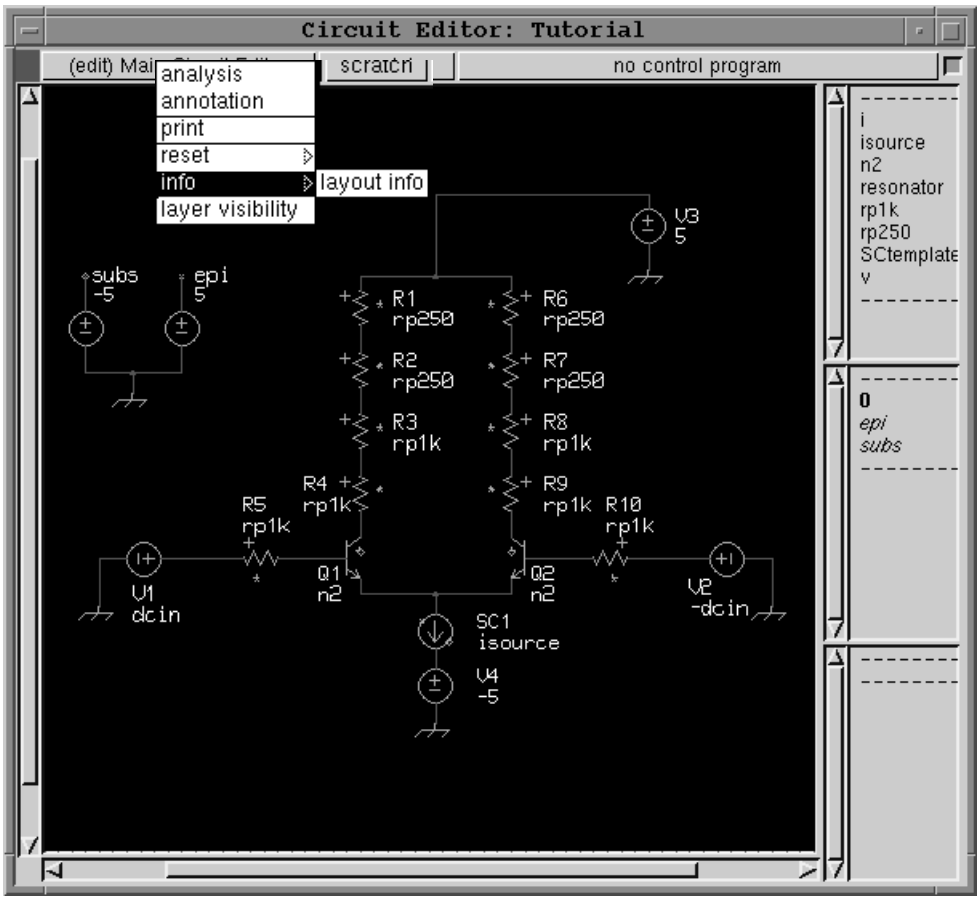


Figure 2–35: Accessing info>layout info

2. This opens the **Netlist To Layout Dialog Box**.

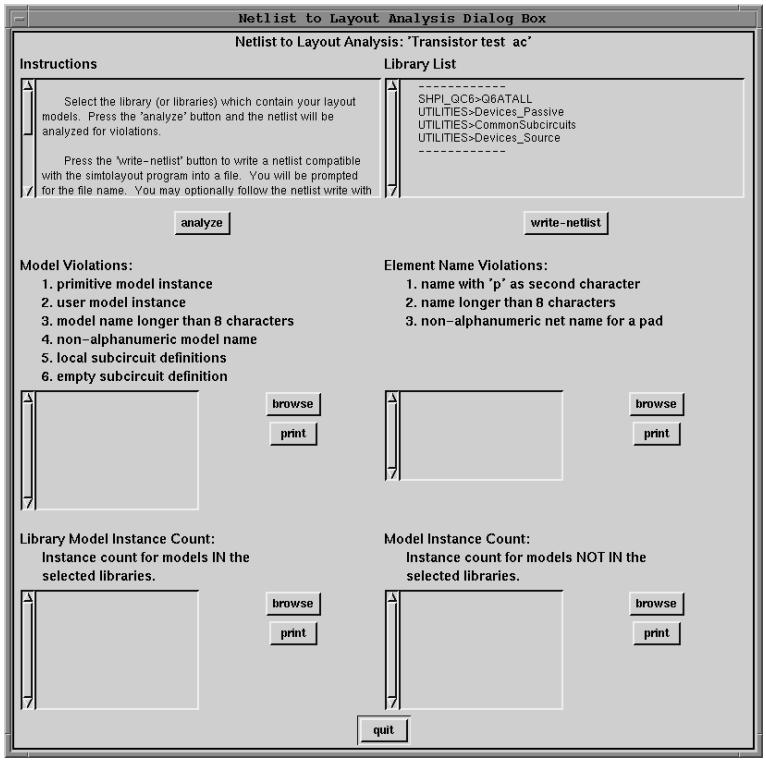


Figure 2–36: Netlist To Layout Dialog Box

Ignore most of this window and click on **Write Netlist**.

3. When the **Netlist generation** window appears click on **ok** to generate the file **Tutorial.tsp**.

4. The netlist has been written to the ADS working directory. Close the **Netlist To Layout Dialog Box** when finished.

NOTE. *This Netlist generation window also provides an analyze command to check the circuit for possible netlist problems. Select the* ***SHPI_QC6>Q6ATALL*** *library and click on* ***analyze*** *to try out the command.*

The *QuicKic Training Tutorials* provide the remaining steps to get this ADS generated netlist into QuicKic.

Using Subcircuit Parameters

Two capabilities of ADS enable analysis of large–scale circuit parameter variations:

1. At simulation time, you can pass parameters into the subcircuit that determine element values in that subcircuit.
2. You can automatically vary parameter values for successive simulations. When finished, you can observe the variations directly on family curve plots with traces corresponding to each parameter setting.

The example subcircuit constructed below creates a resonant response for the differential amplifier circuit. You will make this subcircuit with input parameters for unloaded Q, center frequency, and shunt resistance. Local variables are used for intermediate calculations of the parameter values for Q and center frequency. These local variables can also control the related element parameters.

Create The Resonator

The procedure to create this subcircuit is similar to the previous current mirror exercise. Create the resonant circuit by following the steps below:

1. Create a new subcircuit named **resonator**, by copying the **SCtemplate** as you did earlier to create **isource**.
2. Open a **Subcircuit Definition Editor** window with the **edit** command.
3. This subcircuit does not use the epi or subs terminals. Remove them with the following steps:
   a. In the *Terminal List Pane*, click on the name ***epi*** and execute **cut** from the *Symbol Editor Pane*. Repeat for ***subs***.
   b. Select the ***epi*** and ***subs*** connection in the *Schematic Editor Pane* and **cut** them.
   c. With ***epi*** still highlighted in the *Terminal List Pane*, execute **remove terminal**. Repeat for ***subs***.
4. Now run **add terminal** from the *Terminal List Pane*. Enter **term1>term2** and click on **ok**.
5. Use **paste terminal** to add the terminals to the symbol as shown in Figure 2–37.

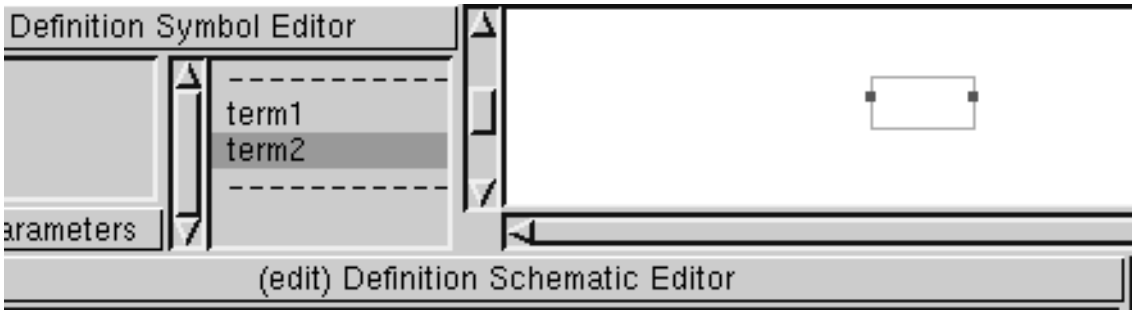


Figure 2–37: Defining Resonator Terminals

6. The *Parameter List Selector Pane* should be set to **input parameters**. (If not, click on the pane until it is correct) Enter the following input parameters in the *Parameter List Pane*. The values after the equal signs are defaults for impedance, resonant frequency, and Q.

```
rval=1k
f0=100meg
qu=10
```

7. While still in the *Parameter List Pane*, **accept** the new parameters.

8. Change the *Parameter List Selector Pane* to **local variables**.

9. Enter local variables as stated below:

```
w0=twopi*f0
cval=qu/(w0*rval)
lval=1/(w0^2*cval)
```

   twopi is a built–in constant equal to 2*π.

10. **Accept** the new local variables.

11. Add models l, c, and r to the *Model List Pane* using the popup menu **library** command.

12. Create the schematic as shown in Figure 2–38.

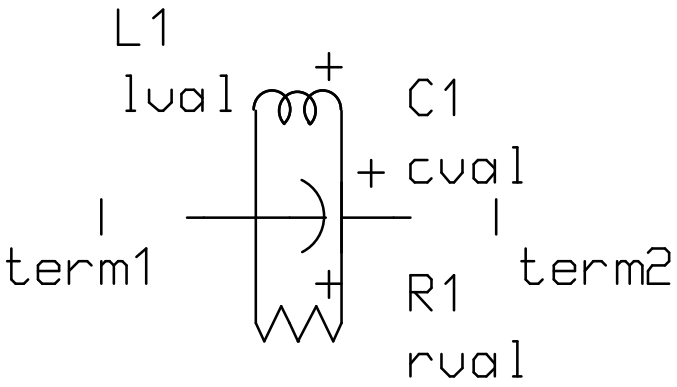


Figure 2–38: Resonator Subcircuit Schematic

13. Use the **browse** command to set the r, l and c input parameters to *r=rval*, *l=lval*, and *c=cval* respectively.

14. Connect the components using the **wire** command.

15. Select and **paste** the subcircuit terminals from the *Node List Pane*.

16. **Accept** the circuit from the *Schematic Editor Pane* and correct any errors.

17. Set the *Parameter List Selector Pane* to **output parameters**. Add the following parameters and expressions to the *Parameter List Pane*:

```
l=output(l1,l)
c=output(c1,c)
r=output(r1,r)
```

These parameters will now appear as output **functions** while in analysis mode.

18. **Accept** the output parameters in the *Parameter List Pane* and then **accept** the circuit in the *Schematic Editor Pane*. Quit the resonator **Subcircuit Editor**.

Place The Resonator

Place the new resonator in the Tutorial schematic **Circuit Editor** (*edit mode set*) as follows:

1. Select **resonator** subcircuit from the *Model List Pane* and **paste** it between the collectors of transistors Q1 and Q2.

2. Use the **wire** command to connect each terminal of the resonator to the adjacent collector nodes. See Figure 2–39.

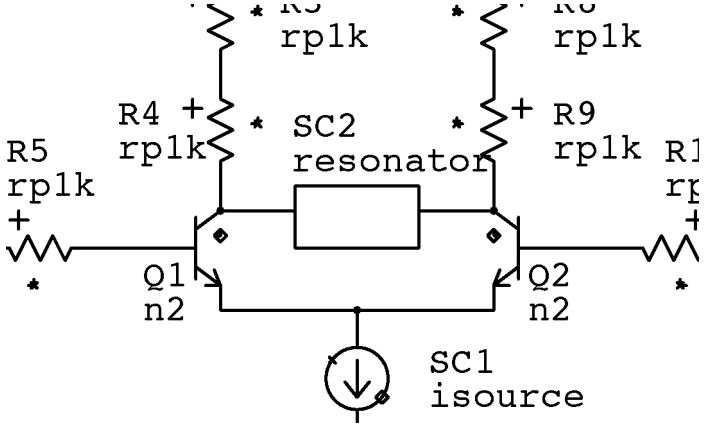


Figure 2–39: Placed Resonator Subcircuit

3. Open the **Element Browser** for the resonator and enter the following input parameters:

   rval=10
   f0=1meg
   qu=20

4. Click on **ok** and **accept** the circuit.

Simulate With The Resonator

Simulate the new circuit as follows:

1. From the **Control Program Browser** modify the **ac** analysis parameters to say:

   ```
   ac analysis freq: 500k 1.5meg 1meg/100
   ```

   This will perform an ac frequency sweep from 500k Hz to 1.5meg Hz with a step size of 10k Hz.

2. **Accept** the control program.

3. **Simulate** and wait for it to finish.

4. Observe the output between the transistor collectors as **mag(@–@1)**.

5. The resonator output variables for r, c, and l appear under the **functions** popup menu command. With the cursor over the resonator, examine these parameters.

Sweeping Circuit Variables

As an additional exercise, determine how the ac response of the resonator circuit changes with the parallel resistance *rval*. This can be done by sweeping the variable *rval* as shown in the steps below:

1. Enter **var sweep** commands for the variable sweep block to the existing ac analysis commands as shown below. Also, add the **bandpass** function to the wire menu as shown. Do not put a carriage return between the **mag(@)** and **mag(@–@1)** functions.

```
binaryansrp=on ; required by TekSpice
astats=on ; report percent of analysis completed
dcin=0
var sweep shuntr : 10 50 10
ac analysis freq: 500k 1.5meg 1meg/100
endac
endvar
wiremenu @–@1 db(@) db(@–@1) phase(@) phase(@–@1) mag(@)
mag(@–@1) lowpass(@) lowpass(@–@1) bandpass(@–@1)
circuitmenu @–@1 db(@) phase(@) mag(@) lowpass(@)
```

2. **Accept** the control program.

3. From the **Circuit Editor** (*edit mode set*) **browse** the resonator to open the **Element Browser**.

4. Assign a new variable *shuntr* to the value of rval. Click on **ok**. At the prompt, declare *shuntr* as a circuit variable with a default value of 10.

5. **Accept** the circuit.

6. **Simulate** the circuit.

7. Observe the results using the analysis functions in the *Schematic Editor Pane* popup menus. Plots of the differential output are families of curves, conveniently displaying in the same window as the outputs for the parameter values. See Figure 2–40. The added **wiremenu** command for **bandpass** reduces the output to a single trace representing the change in filter response vs frequency.

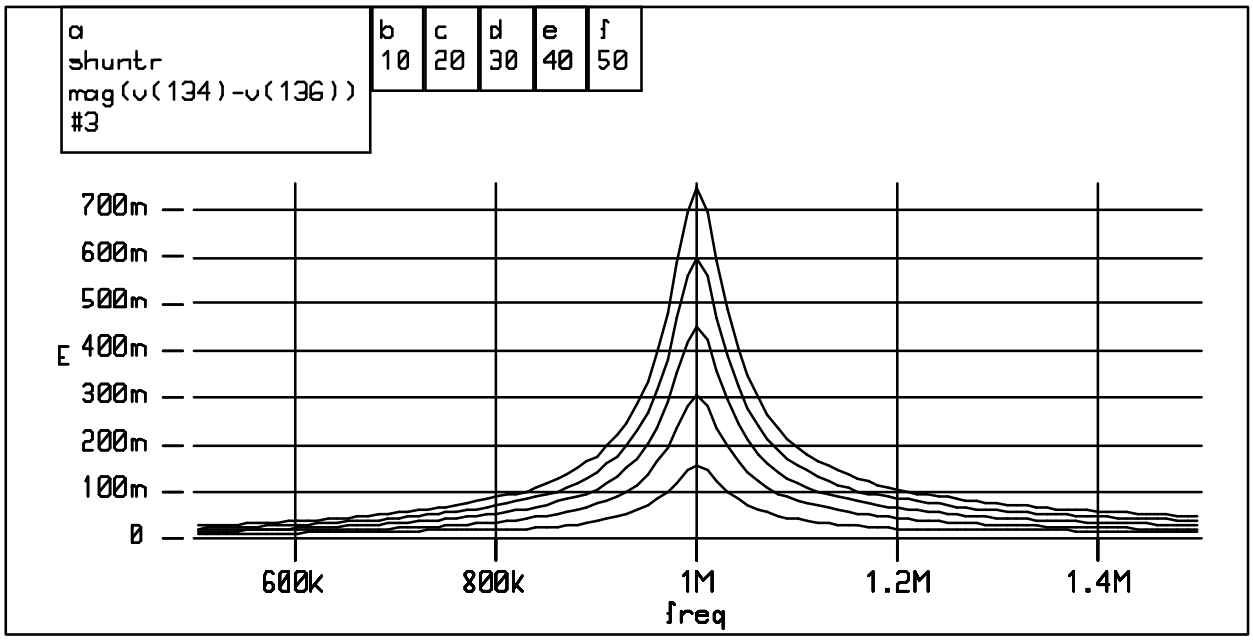


Figure 2–40: Family of Curves from "function > mag(@–@1)"

End of Tutorial

You have completed the *ADS Tutorial* and have been introduced to the basic features of the ADS program. Feel free to experiment with other features and plotting functions.

See the *ADS User Manual* and the *TekSpice2 Reference Manual* for further information when designing your own circuits.

Window Descriptions

Launchers

This reference chapter describes the functionality of the windows within the ADS system. Although technically they are all windows, for the purposes of this manual they are divided into Launchers, Browsers, Editors and Dialog Boxes. Also described in this reference are panes and pop–up menus which contain commands and sub–menus accessible from within the windows.

There is only a single launcher described in this section.

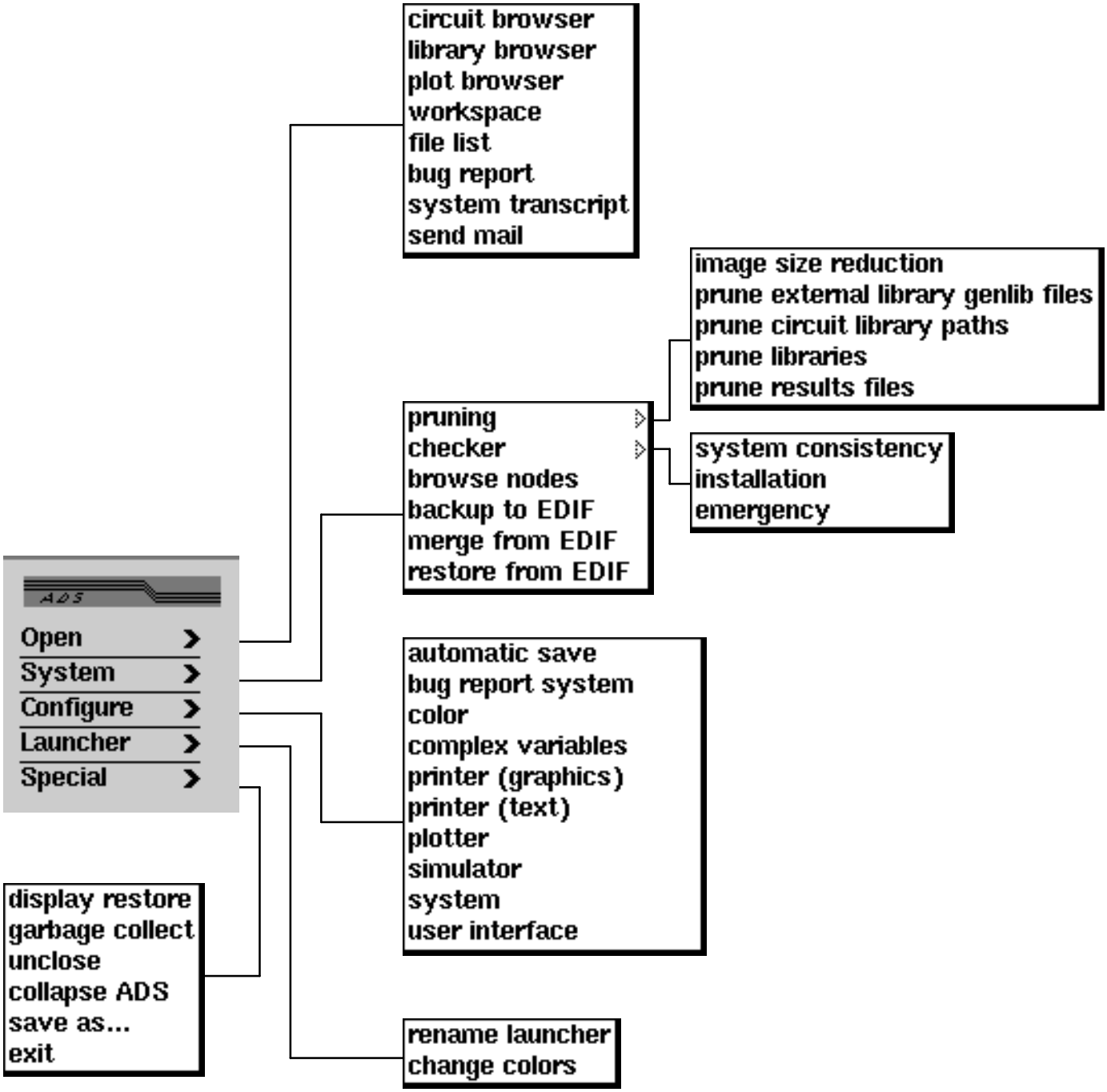


Figure 3–1: Launcher Pull–out Commands

Launcher

The **Launcher** is the main window in ADS. From the **Launcher** all other window are started. The **Launcher** appears in the initial ADS image as a small window with a title bar containing the ADS version. This window must remain open or as an icon during the entire ADS session.

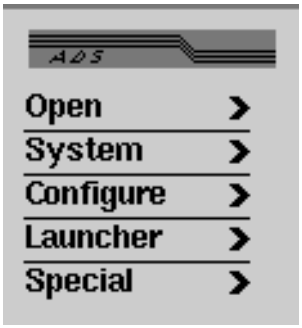


Figure 3–2: The Launcher

Access

The **Launcher** is the main ADS window and is always available.

Description

The commands listed on the Launcher and their sub–commands are described briefly below:

Open. This is the entry point to the following windows (See Figure 3–1):

Circuit Browser.

Library Browser.

Plot Browser.

Annotation Editor.

Workspace Editor.

File List Browser.

Bug Report dialog box.

System Transcript dialog box.

Send Mail Browser.

System. This is the entry point for the following commands (See Figure 3–1):

pruning. The pruning command allows access to the following sub–commands:

image size reduction. See **Image Reduction Dialog Box.**

prune external library genlib files. Presents a list of files that are external library genlib files no longer referenced by this image and allows you to remove them.

prune circuit library paths. Removes library path entries from circuits when the library no longer exists in the image.

prune libraries. Removes libraries when the contents of the library are no longer used by circuits in the image.

prune results files. Presents a list of files in the simulation results directory that are no longer referenced by this image as results.

checker. The checker command allows access to the following sub–commands (See Figure 3–1):

system consistency. See **System Consistency Checker Dialog Box**.

installation. See **System Installation Checker Dialog Box**.

emergency. Used for running emergency operations usually upon advice from your ADS support person. See **Emergency Checker Dialog Box**.

browse nodes. Opens the **Node Browser**.

backup to EDIF. Used to write all the circuit data in the image to an EDIF file.

merge from EDIF. Used to merge the circuit data from a ”backup to EDIF” file into the current image.

restore from EDIF. Used to replace all the circuit data in the image with the data found in the ”backup to EDIF” file.

Configure. This is the entry point to the following commands (See Figure 3–1):

automatic save. See **Automatic Save Configuration Dialog Box**.

bug report system. See **Bug Report Configuration Dialog Box**.

color. See **Color Configuration Dialog Box**.

complex variables. See **Complex Variable Configuration Dialog Box**.

printer (graphics). See **Printer Options (Graphics) Dialog Box**.

printer (text). See **Printer Options (Text) Dialog Box**.

plotter. See **Plotter Configuration Dialog Box**.

simulator. See **Simulator Configuration Dialog Box**.

system. See **System Configuration Dialog Box**.

user interface. See **User Interface Dialog Box**.

Launcher. Allows access to the following commands (See Figure 3–1):

rename launcher. Allows changing the text in the title bar of the **Launcher.**

change colors. Allows changing the color of the title bar of all windows in the image (This is useful if more than one image is active on the same monitor).

Special. This command allows access to the following commands (See Figure 3–1):

display restore. Restores all closed ADS windows from the **collapse ADS** command.

garbage collect. Perform a garbage collection of the ADS image printing the results to the **System Transcrip**t dialog box.

unclose. Reopens the window that was most recently closed. This will reopen the window in its closed state even if it was not accepted.

collapse ADS. Collapse all the ADS windows and open a new **Launcher**. You can **expand** this **Launcher** window to regenerate all ADS windows to the previous state.

save as.... Saves the ADS image to a file name. The default file name is the one used to start ADS.

exit. Exits ADS

Browsers

This reference describes the functionality of the windows within the ADS system. Although technically they are all windows, for the purposes of this manual they are divided into Launchers, Browsers, Editors and Dialog Boxes. Also described in this reference are panes and pop–up menu commands.

In this section the ADS browsers are presented alphabetically by name. Each browser description contains the following information:

1. A brief description of each browser.
2. A figure showing the location of each pane in the browser.
3. The path to access the browser.
4. A description of each pane in the browser.

Listed below are the browsers described in this section. The more commonly used browsers are in **bold** typeface. The description of these browsers include a figure showing the available pop–up button menus.

- **Circuit Browser**
- **Control Program Browser**
- Experiment Browser
- File List Browser
- **Library Browser**
- Node Browser
- Noise Table Browser
- Operating Point Browser
- **Plot Browser**
- Send Mail Browser
- Smith Chart Browser
- User Operating Point Browser

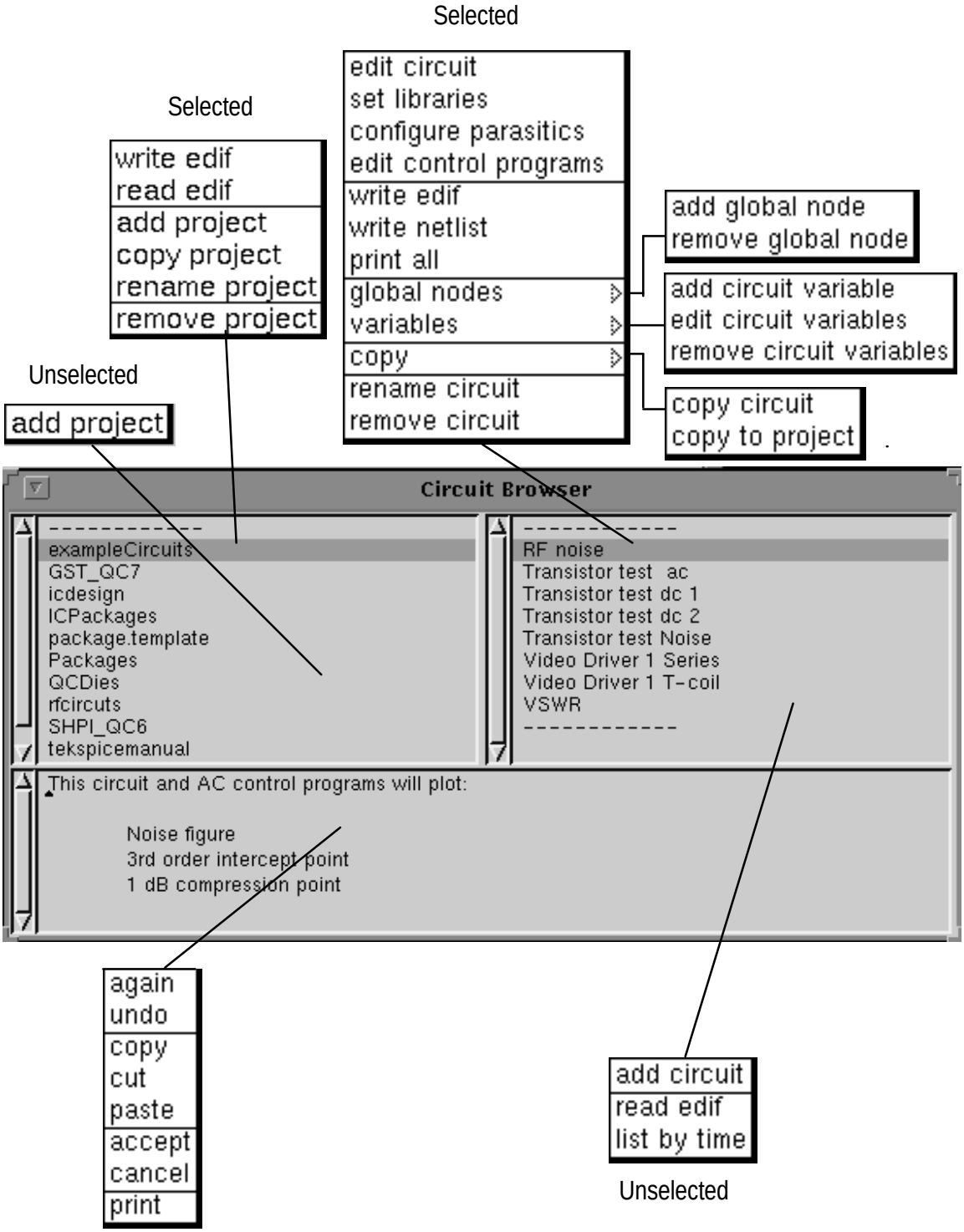


Figure 3–3: Circuit Browser Commands

Circuit Browser

The **Circuit Browser** is the access point for all circuits which exist in ADS. There are two levels of hierarchy within this browser; the project level which defines a group of circuits within the project; and the circuit level which contains the individual circuits. Figure 3–3 shows the pop–up button commands available from the **Circuit Browser**.

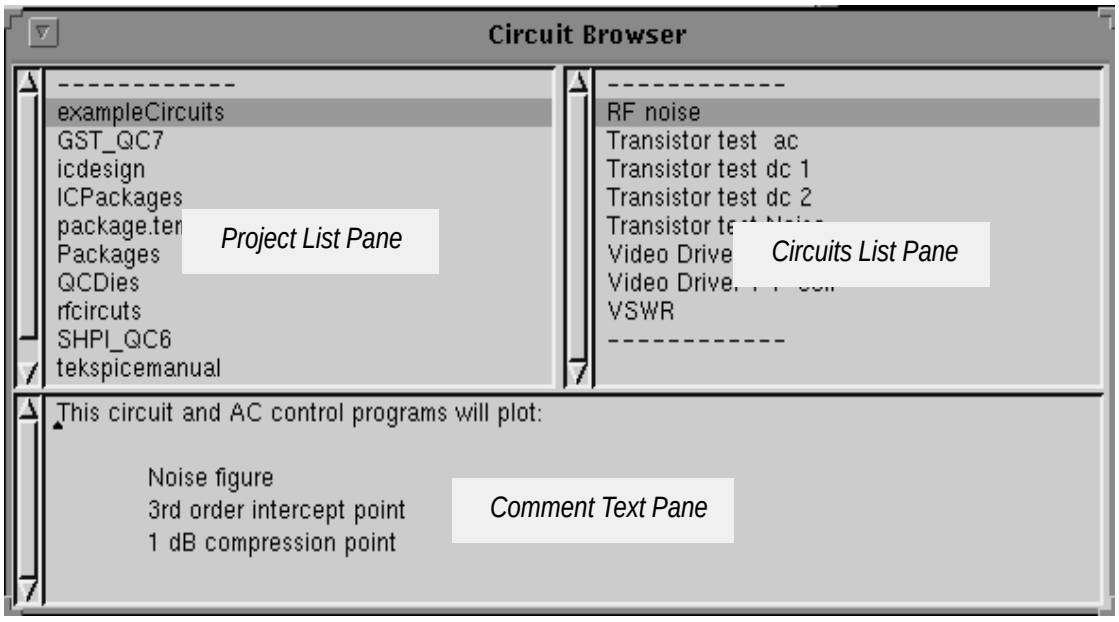


Figure 3–4: Circuit Browser Panes

Access

Access the **Circuit Browser** from the **Launcher** using the **Open > circuit browser** command.

Panes

Figure 3–4 shows the location of the panes in the **Circuit Browser**.

A description of the panes in this window follows:

Circuits List Pane

The *Circuit List Pane* lists the circuit names in the present ADS image. Pop–up button menu commands accessed within this pane support adding, renaming, copying, and removing circuits. They also allow reading and writing circuit files in EDIF format as well as adding or deleting simulation libraries, global nodes and circuit variables. See **EDIF Dialog Boxes**.

Comment Text Pane

The Comment Text Pane contains text describing the item selected in the active pane. Use the standard text editing commands in this pane. See the **Getting Started** chapter.

Project List Pane

The *Project List Pane* lists the project names in the present ADS image. Pop–up menu commands within this pane support adding, renaming, copying, and removing projects. They also allow reading and writing project files in EDIF format.

(This page intentionally left blank)

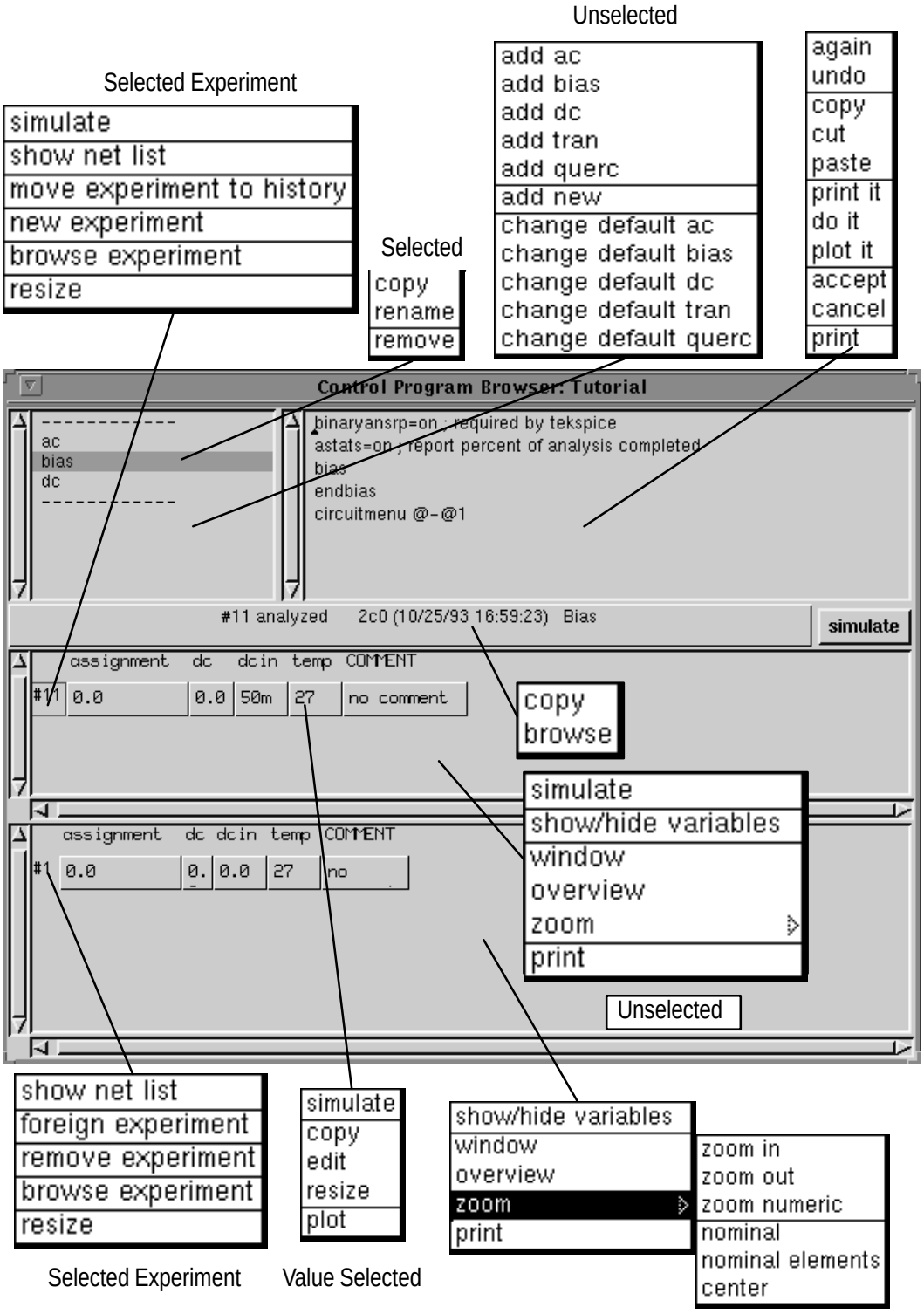


Figure 3–5: Control Program Browser Commands

Control Program Browser

The **Control Program Browser** allows the selection and editing of control statements (control programs) to be used when running the simulator TekSpice. Control programs are organized by name and listed in the *Control Program List Pane*. The contents of the selected control program are shown in the *Control Program Text Pane*. Each control program associated with an experiment table displayed in the *Experiment Table Pane*. Figure 3–5 shows the pop–up button commands available from the **Control Program Browser**.

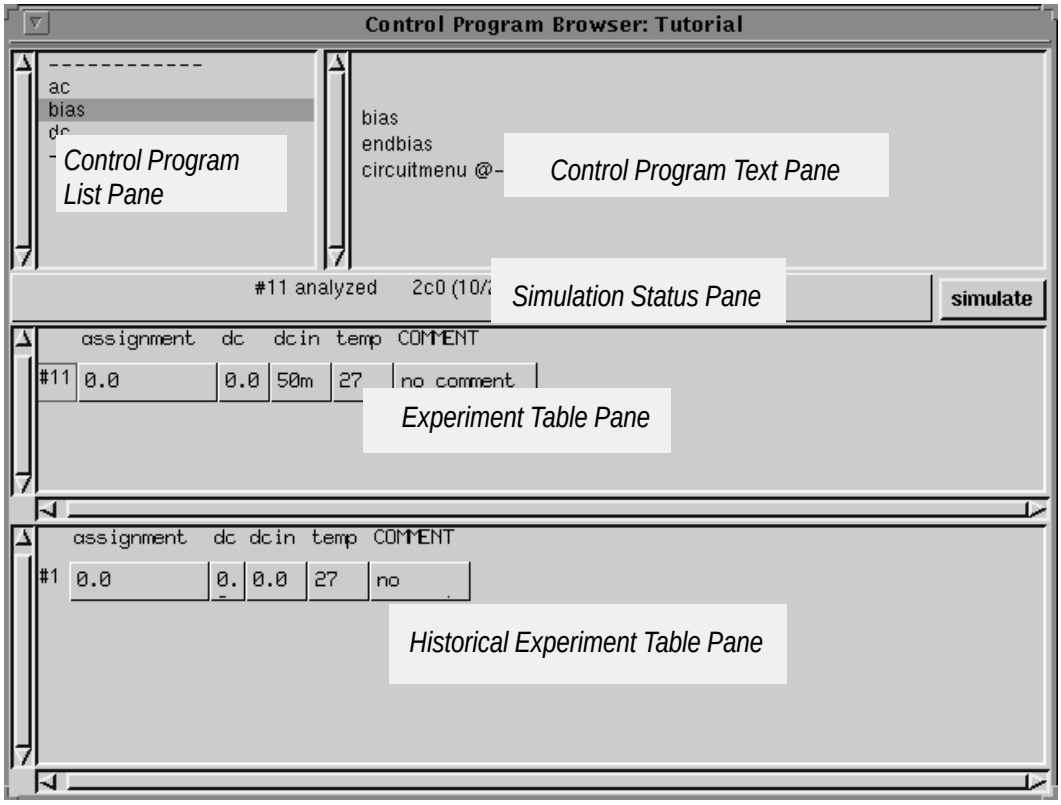


Figure 3–6: Control Program Browser Panes

Access

Access the **Control Program Browser** either 1) from the **Circuit Editor**'s *Simulation Status Pane* using the pop–up menu **edit control programs** command, or 2) from the *Circuit List Pane* of the **Circuit Browser** using the pop–up button command **edit control programs**.

Panes

Figure 3–6 shows the location of panes in the **Control Program Browser**.

A description of the panes in this window follows:

Control Program List Pane

The *Control Program List Pane* lists the presently defined control programs.

Control Program Text Pane

Control Program Text Pane displays the contents of a selected control program. Use the standard text editing commands for modifying the control programs. The first line of all control programs contain the statement **clear**, which resets all the variable assignments to zero, before a simulation. The contents of the control program may include (order dependent): Variable assignments (if any); TekSpice simulation command blocks; and Post–simulation commands and assignments. See also **circuitmenu**, **wiremenu**, and Waveform Functions.

Experiment Table Pane

The *Experiment Table Pane* sets up and execute simulations for a selected control program. This pane displays one row for each simulation command block in an analyzed simulation control program. The Experiment Table shows the selected variable values, providing a method for viewing the variable assignments used in the simulation. Control programs can be executed and the Experiment Table format can be modified from this pane.

This table displays preset variables and user defined circuit variables, and local variables. Rows of experiments associate simulation results with the values for those variables. The displayed items can be selected from all available variables. The selected variable names appear above the values assigned to them. Experiments are further identified by an automatically generated number and by the analysis categories; ac, bias, dc and tran.

Historical Experiment Table Pane

A *Historical Experiment Table Pane* is located in the bottom pane of the **Control Program Browser**. The most recent working experiment number is reused if that experiment has no results. An experiment can be copied from the working table to the *Historical Experiment Table Pane* with the pop–up button menu **move experiment to history** command. Do this to save results. The **foreign experiment** command is added to the historical table for easy access to simulation results from a circuit other than the active circuit in the **Control Program Browser**.

Simulate Button

Press the *Simulate Button* to run the selected control program.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)

- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

(This page intentionally left blank)

Experiment Browser

The **Experiment Browser** displays more detailed information about an experiment (simulation).

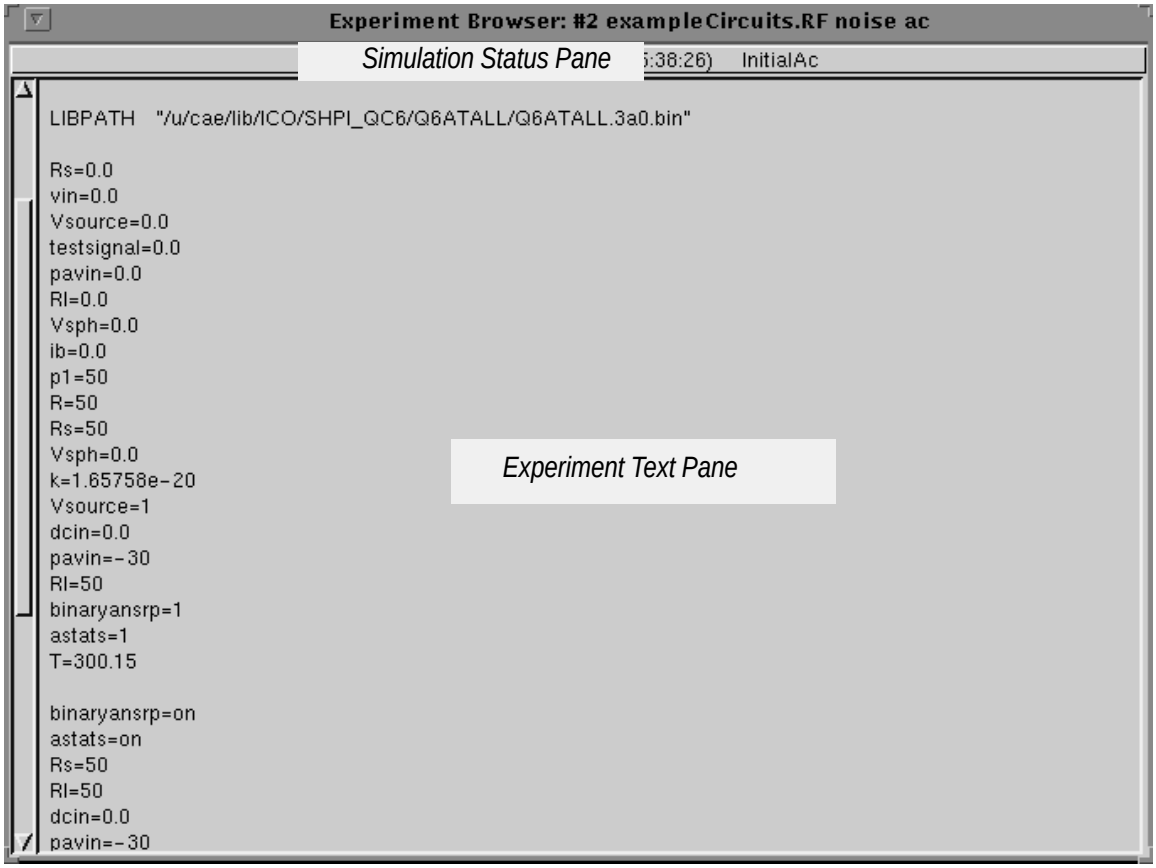


Figure 3–7: Experiment Browser Panes

Access

Access the **Experiment Browser** from the **Circuit Editor** *Simulation Status Pane* (or any *Simulation Status Pane*) using the pop–up button menu command **browse experiment** or **browse**.

Panes

Figure 3–7 shows the location of panes accessed from the **Experiment Browser**. The panes available from this browser are the *Simulation Status Pane* and the *Experiment Browser Pane.*

A description of the panes in this window follows:

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Experiment Text Pane

The *Experiment Text Pane* shows the library path, the variable assignments, and the TekSpice control program commands used to run the simulation. Also shown are the results file names and simulation descriptions. If a simulation was not run, the commands which will be used are shown instead.

File List Browser

The **File List Browser** serves as a window into the operating system of the workstation. It allows updating new versions of ADS and reading EDIF files into libraries. It is also used for general purpose file access and as a text editor.

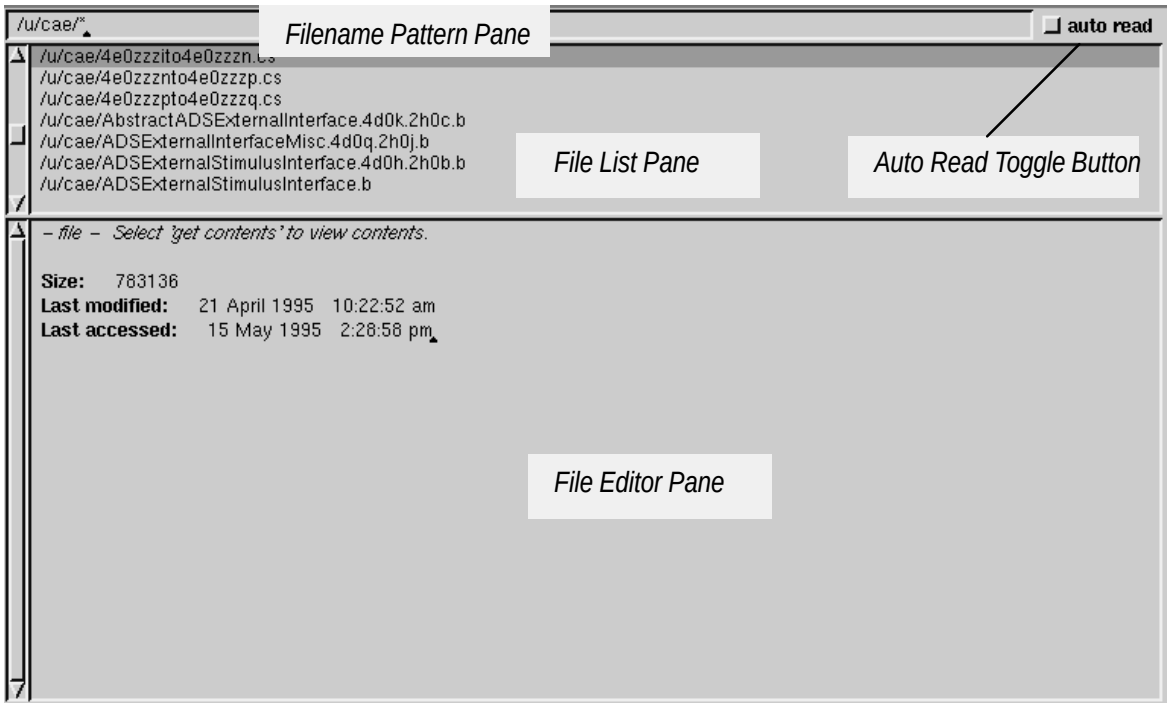


Figure 3–8: File List Browser

Access

Access the **File List Browser** from the **Launcher** using the **Open > file list** command.

Panes

Figure 3–8 shows the location of the panes in the **File List Browser**.

A description of the panes in this window follows:

Auto Read Toggle Button

Displays the contents of the selected file when selected. otherwise the file's *get info* information is displayed.

File Editor Pane

The *File Editor Pane* displays and allows editing the contents of files selected in the *File List Pane*. Use the standard text editing commands in this pane.

File List Pane

The *File List Pane* shows the files selected from an **accept** command in the File Name Pattern Pane.

File Name Pattern Pane

The *File Name Pattern Pane* allows typing existing or new file names or patterns for file names. Special characters * and # are permitted. ”* ” matches any sequence of none, one, or more characters. ”#” matches any one character. Issuing the pop–up button **accept** command displays the name or pattern into a list of matching directories, files, or both in the *File List Pane*.

(This page intentionally left blank)

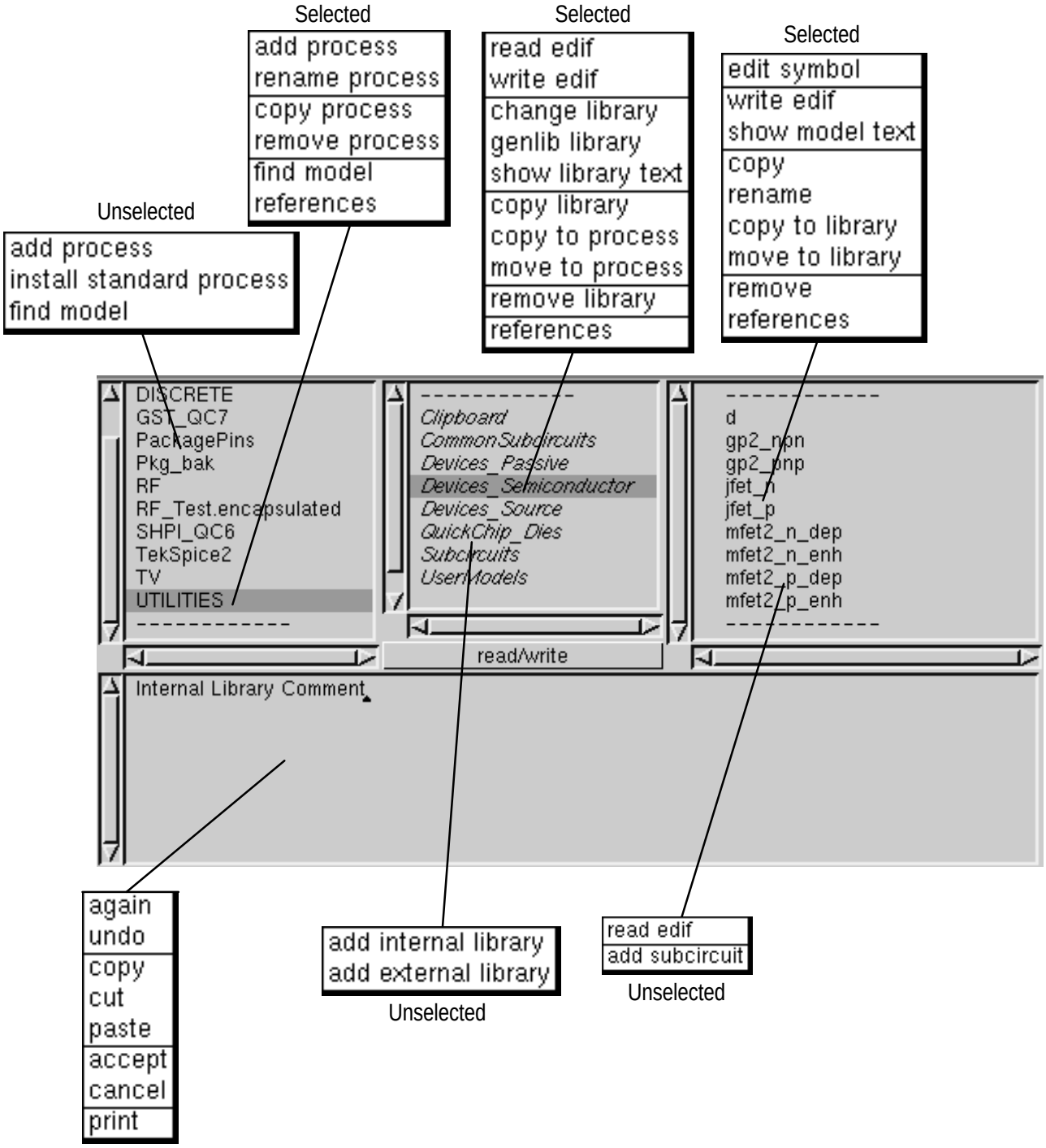


Figure 3–9: Library Browser Commands

Library Browser

The **Library Browser** supports the addition and removal of all libraries defined within the active ADS image. The models in the libraries defined through this window are available for use by any circuit within the ADS image. The libraries for a particular circuit are selected using the **Library Path Editor** and opened using the **set libraries** command in the *Circuit List Pane* of the **Circuit Editor**.

The **Library Browser** supports three levels of hierarchy; the library Process (displayed in the *Process List Pane*) which displays the process used; the Libraries (displayed in the *Library List Pane*) which displays the libraries available for the process; and the Library Models (displayed in the *Library Model List Pane*) which displays the models available from the Library. The library models may be subcircuits or user models. Figure 3–9 shows the pop–up button commands available from the **Library Browser**.

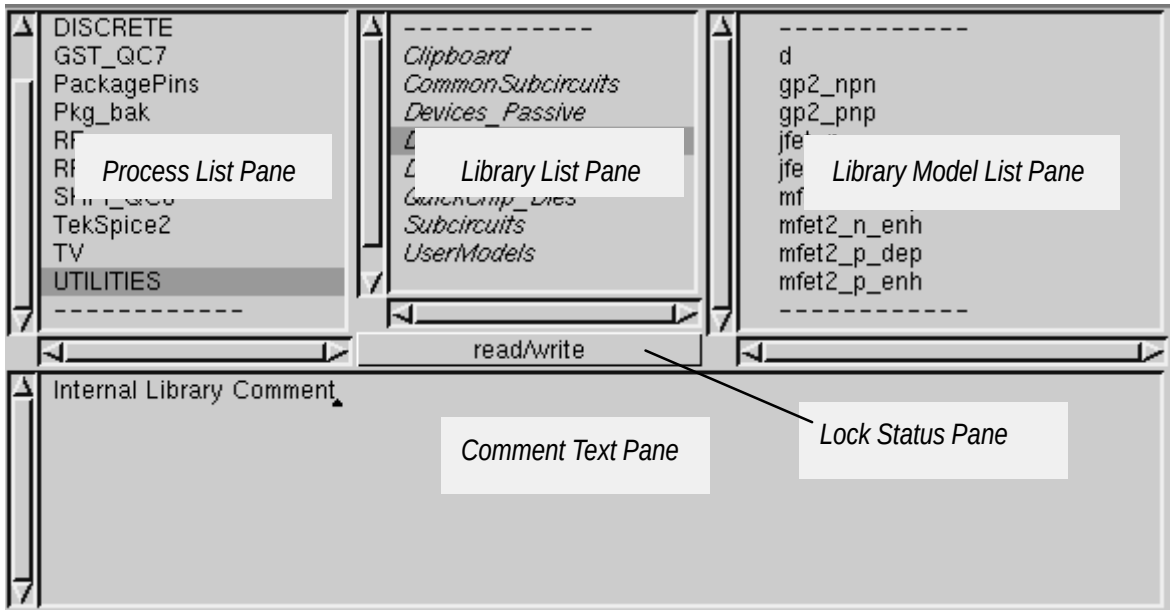


Figure 3–10: Library Browser Panes

Access

Access the **Library Browser** from the **Launcher** using the **Open > library browser** command.

Panes

Figure 3–10 shows the location of the panes in the **Library Browser**.

A description of the panes in this window follows:

Comment Text Pane

The Comment Text Pane contains text describing the item selected in the active pane. Use the standard text editing commands in this pane. See the **Getting Started** chapter.

Library List Pane

The *Library List Pane* displays the list of libraries available for the selected process. Libraries on this list may be internal libraries or external libraries. Selecting a library lists the available models in the *Library Model List Pane*.

Library Model List Pane

The *Library Model List Pane* provides the list of user and subcircuit models available for use in the creation of a circuit schematic. The highlighted library in the *Library List Pane* determines the names on this list. Library models can be added, modified or removed from this list. The typface font of model names indicate the model type: *italics* font indicates a local model, a regular font indicates a library model; and **bold regular** font indicates a primitive model. ADS version 4d0 and above promote using only the library model.

Lock Status Pane

The *Lock Status Pane* indicates the status of the selected library in the *Library List Pane* of the **Library Browser**. The status can be **read/write** or **read only**. This status can be changed using the popup button command **change library** in the *Library List Pane* of the **Library Browser**.

Process List Pane

The *Process List Pane* lists the processes available in the current ADS image. Selecting a process lists the available libraries in this process in the *Library List Pane*.

Node Browser

The **Node Browser** lists the names given to node types in the current ADS image. The symbols representing these node types can be viewed and edited using the **Node Symbol Editor**. Editing the symbols listed in the **Node Browser** changes the default symbol for that node type. It does not change the symbol for existing instances.

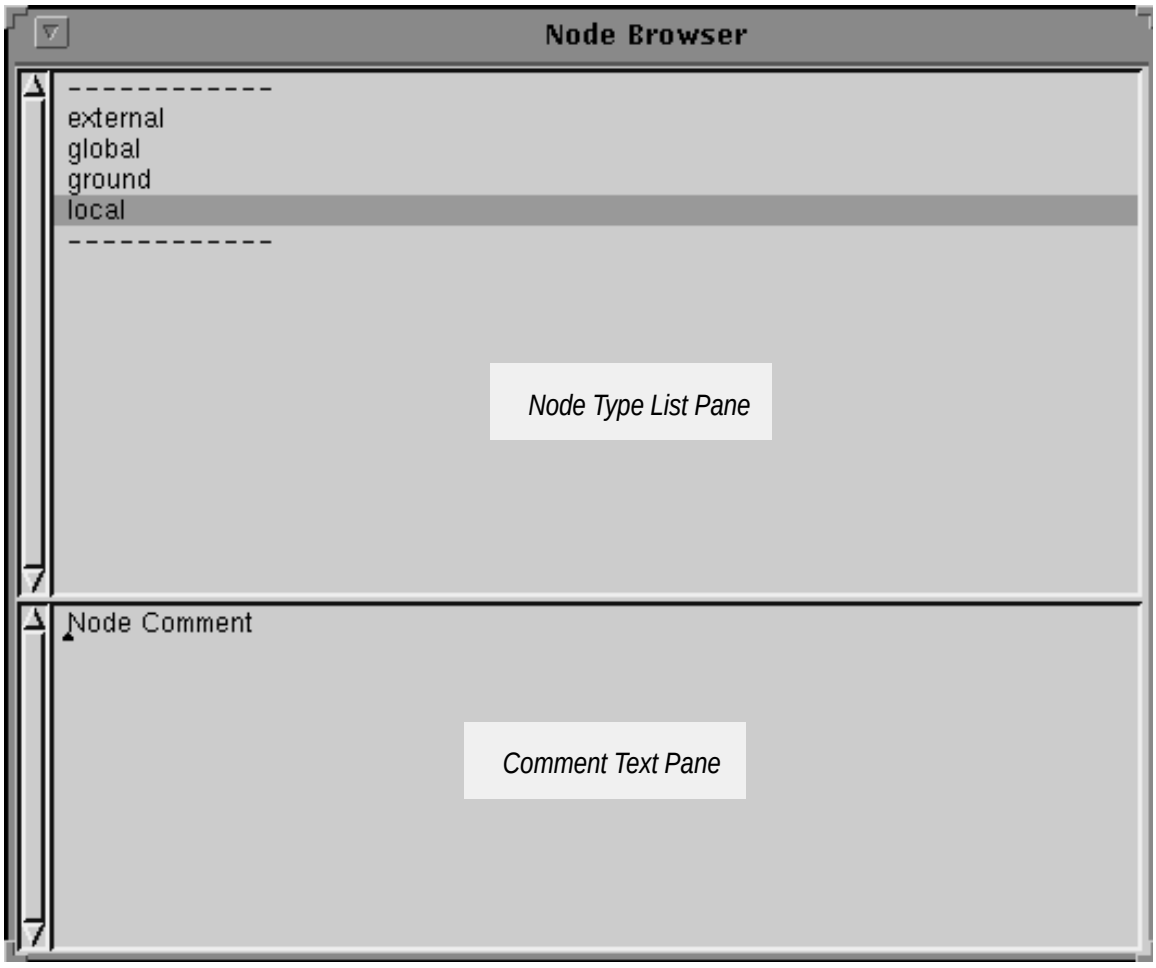


Figure 3–11: Node Browser Panes

Access

Access the **Node Browser** from the **Launcher** using the **System > browse nodes** command.

Panes

Figure 3–11 shows the location of the panes in the **Node Browser**.

A description of the panes in this window follows:

Comment Text Pane

The Comment Text Pane contains text describing the item selected in the active pane. Use the standard text editing commands in this pane. See the **Getting Started** chapter.

Node Type List Pane

The *Node Type List Pane* provides the names of the node types used in the ADS image. The node types are: external, global, ground and local. **Node Symbol Editor** allows viewing the schematic symbols representing these nodes. An external node is used to define the ports of a subcircuit. Global nodes are defined only once throughout the circuit hierarchy. The ground node is the special case for a global node. Local nodes are present only in the displayed schematic.

Noise Table Browser

The **Noise Table Browser** allows viewing the results of a TekSpice Noise analysis. This browser allows examination of: 1) noise contribution values for all circuit elements, 2) summary information for each element, and 3) overall summary information for the entire circuit. A cutoff noise value threshold can be set to view only those elements whose noise contributions are above this cut–off value. This dramatically reduces the amount of data displayed.

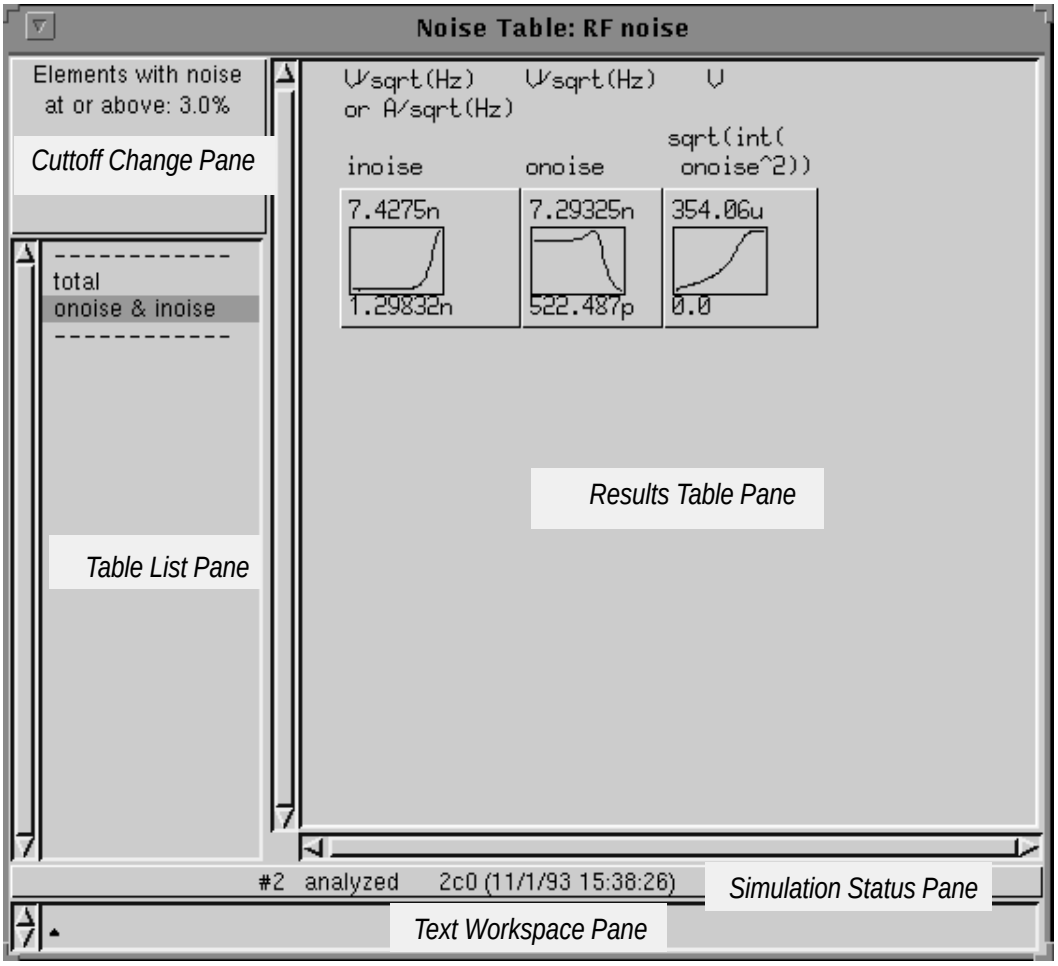


Figure 3–12: Noise Table Browser Panes

Access

Access the **Node Table Browser** from the **Circuit Editor** after an ac analysis. The command from the pop–up menu is **open > noise table**. This appears only if the ac analysis contains the TekSpice NOISE command

Panes

Figure 3–12 shows the location of the **Noise Table Browser** panes.

A description of the panes in this window follows:

Cutoff Change Pane

The *Cutoff Change Pane* allows displaying only those elements whose noise sums are above a certain percentage of the circuit's total output noise. This is helpful in a large circuit, or a circuit where many elements have noise sums of zero. The pop–up button **change cutoff** command allows changing the cutoff percentage at any time. The cutoff percentage does not take affect until "total" is selected (see *Table List Pane*).

The normalized total noise contribution (in %) for element E, $n_{TCE,}$ of any circuit element is defined as follows.

$$n_{TCE}\ (\%) = \sqrt{\frac{\int \sum n_{cE}}{\int (onoise)^2}}$$

where; n_{cE} is the noise contribution for element E; and *onoise* is the circuit's output noise. In a TekSpice **varsweep** analysis, both n_{cE} and *onoise* are families of waveforms. The maximum value for n_{TCE} across all var sweep values is used for each element.

Results Table Pane

The *Results Table Pane* contains data from the selected table. If there is no data for the rows, labels appear instead of numbers. The pop–up button menu commands change depending on the selection of a row label, a column label, a single cell or nothing.

In the **Noise Table Browser** the *Results Table Pane* is a set of postage stamp plots of noise versus frequency. The particular noise parameters are identified by the column headings. The contents of the *Results Table Pane* are controlled by the *Table List Pane* and the *Cutoff Change Pane*. At the top of each column heading are the units on the vertical axis of those postage stamps, for example V*V/Hz is volts squared per hertz. TekSpice only provides voltage noise outputs; input noise may be a current, so amps per square root of hertz is shown as one of two possible units on the inoise column.

Each row in the *Results Table Pane* represents one circuit element, except when "onoise and inoise" is selected in the *Table List Pane.* Selecting "onoise & inoise", displays input and output noise for the entire circuit. For more information on the meaning of inoise, onoise, and the noise contributions, see the description of the TekSpice NOISE command.

Selecting the pop–up button command **total** in the *Table List Pane*, causes additional model names to appear on the list and the *Results Table Pane* to

display total noise and n_{TCE} for each element. Selecting a model name from this list, displays circuit elements of that model type; all noise contributions; the sum of their noise contributions; and n_{TCE}.

Elements in the *Results Table Pane* are sorted from highest to lowest n_{TCD}. This is the same noise contribution number used to determine the cutoff, as described under *Cutoff Change Pane*.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Table List Pane

The *Results Table Pane* displays the table highlighted in the *Table List Pane*. Only data for the specified output parameters are computed.

For the **Noise Table Browser** the *Table List Pane* initially has the “onoise & inoise” table selected. In order to display noise contributions of individual elements, select “total” in the *Table List Pane*. This displays total noise for all elements above the cutoff (see *Cutoff Change Pane*).

Selecting “total” for the first time in a **Noise Table Browser** causes loading the individual elements’ noise contributions into ADS. This step brings up a quit–button dialog box, since there may be a delay while loading results. A higher cutoff value (see *Cutoff Change Pane*) may give a shorter delay.

After loading the individual elements’ noise contributions, the *Table List Pane* lists the model names for all elements passing the cutoff. Selecting a model name gives detailed information on all elements of that model.

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the

Simulation Status Pane. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

(This page intentionally left blank)

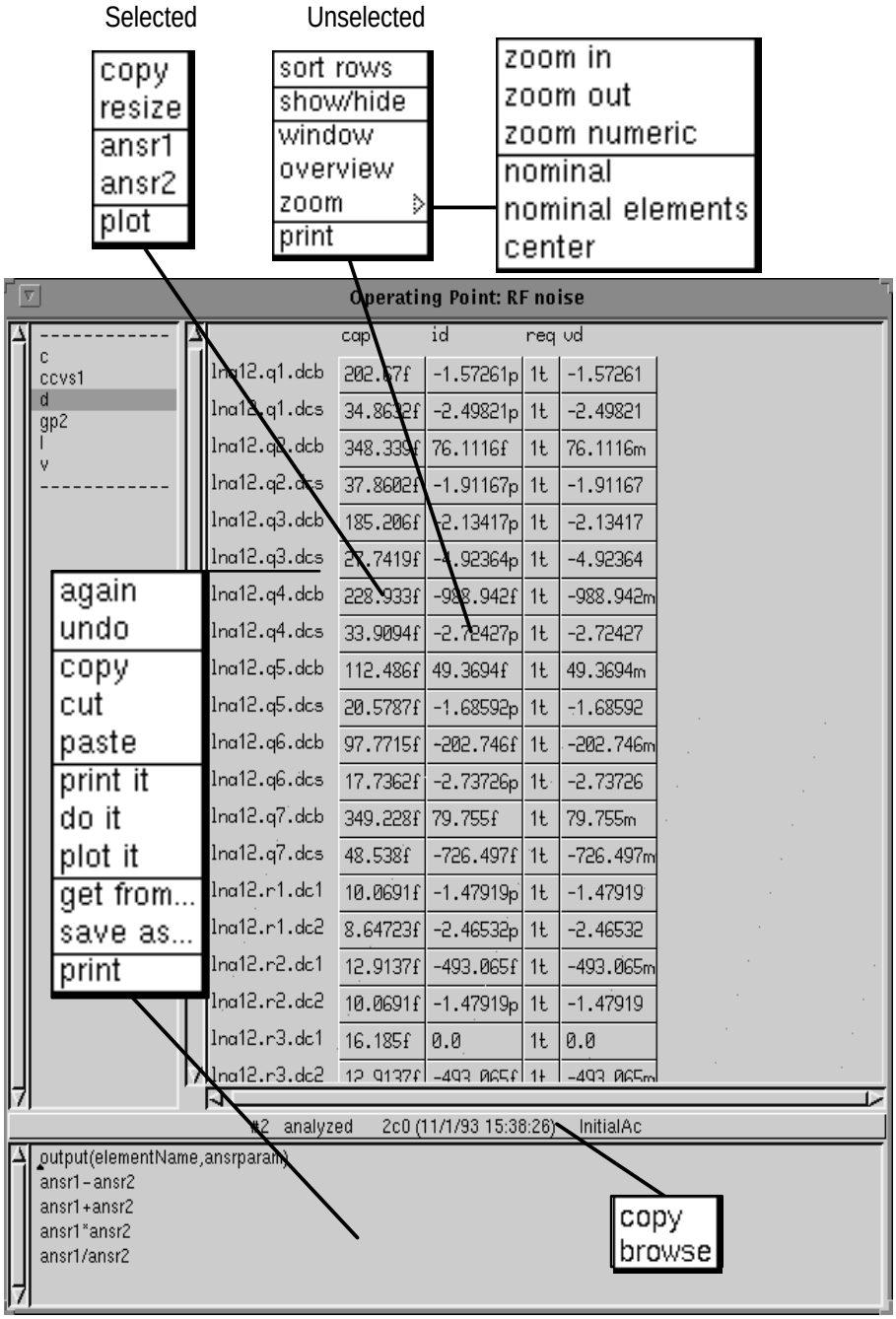


Figure 3–13: Operating Point Browser Commands

Operating Point Browser

The **Operating Point Browser** allows examination of simulation output in a tabular format. This information is equivalent to that provided by TekSpice in the files biansrp, dcansrp, acansrp, or transrp, corresponding to bias, dc, ac, or transient analysis. The **Operating Point Browser** opens only when data is available.

By default, columns for all data are visible. The **hide** and **show** commands determine the displayed columns (to reduce the size of the tables). This is convenient when preparing the table for printing. Data in the *Results Table Pane* comes from the analysis selected in the **Control Program Editor**. Figure 3–13 shows the pop–up button commands available from the **Operating Point Browser**.

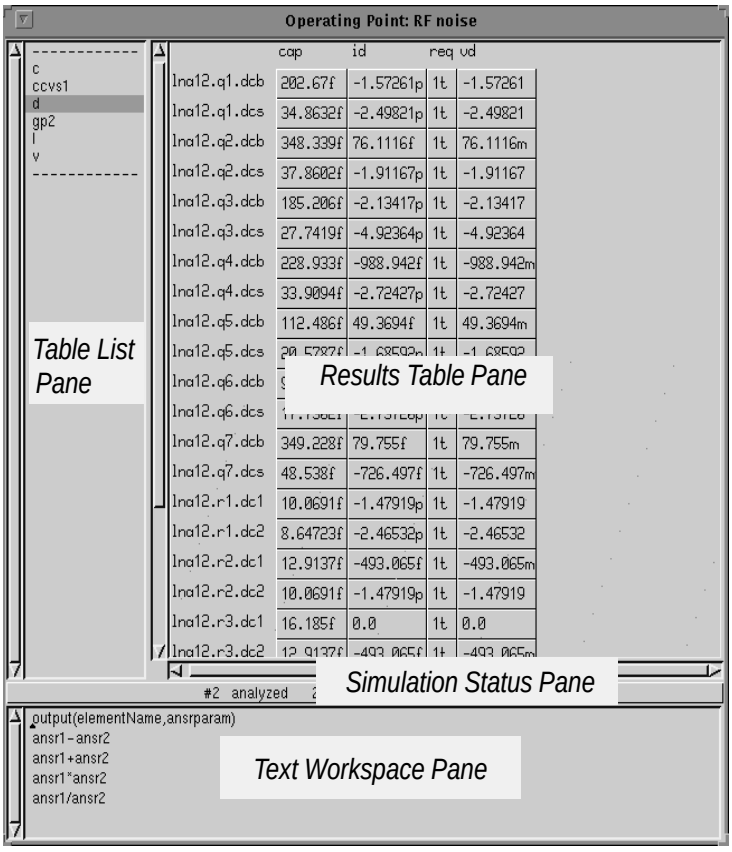


Figure 3–14: Operating Point Browser Panes

Access

Access the **Operating Point Browser** from either 1) the **Circuit Editor's** (in analysis mode) *Schematic Editor Pane* using the pop–up button **open > operating point table** command or 2) from the **Subcircuit Instance Editor**.

Panes

Figure 3–14 shows the Panes accessed from the **Operation Point Browser**.

A description of the panes in this window follows:

Results Table Pane

The *Results Table Pane* contains data from the selected table. If there is no data for the rows, labels appear instead of numbers. The pop–up button menu commands change depending on the selection of a row label, a column label, a single cell or nothing.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Table List Pane

The *Results Table Pane* displays the table highlighted in the *Table List Pane*. Only data for the specified output parameters are computed.

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

(This page intentionally left blank)

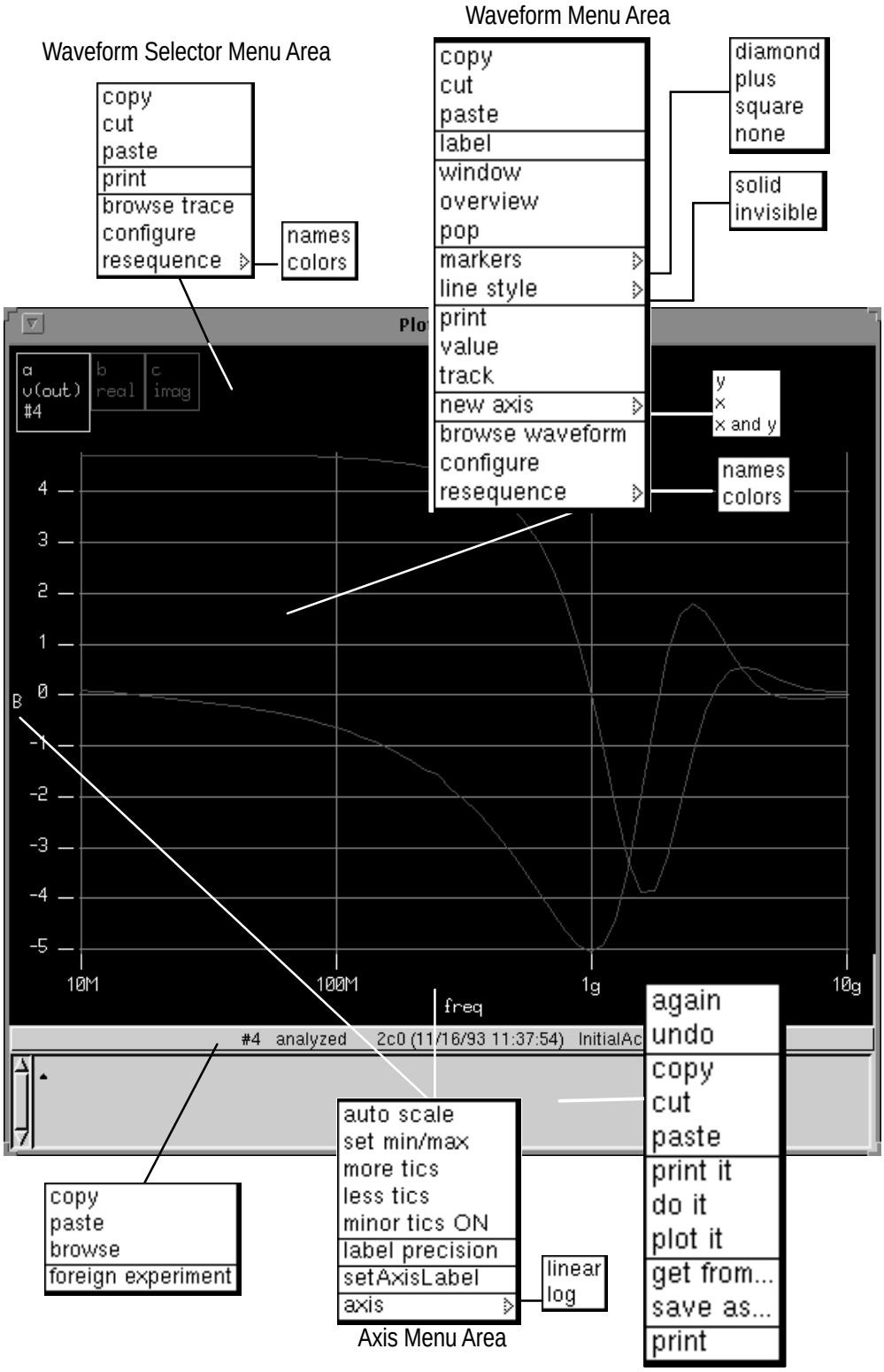


Figure 3–15: Plot Browser Commands

Plot Browser

The **Plot Browser** allows the designer to examine and manipulate output data generated by the TekSpice simulator or from external files. Any number of **Plot Browser** windows can be opened at one time. The **Plot Browser** can be configured for a analog or digital plots.

Highlighting an expression in any workspace and using the pop–up button menu **plot it** command opens a **Plot Browser**. Highlighting an expression in any workspace and clicking on **do it** causes evaluation of the expression and the result put in a buffer. This can then be pasted onto **Plot Browser** as a plot or into a *Schematic Editor Pane* as a postage stamp. The pop–up button commands available from the **Plot Browser** are shown in Figure 3–15.

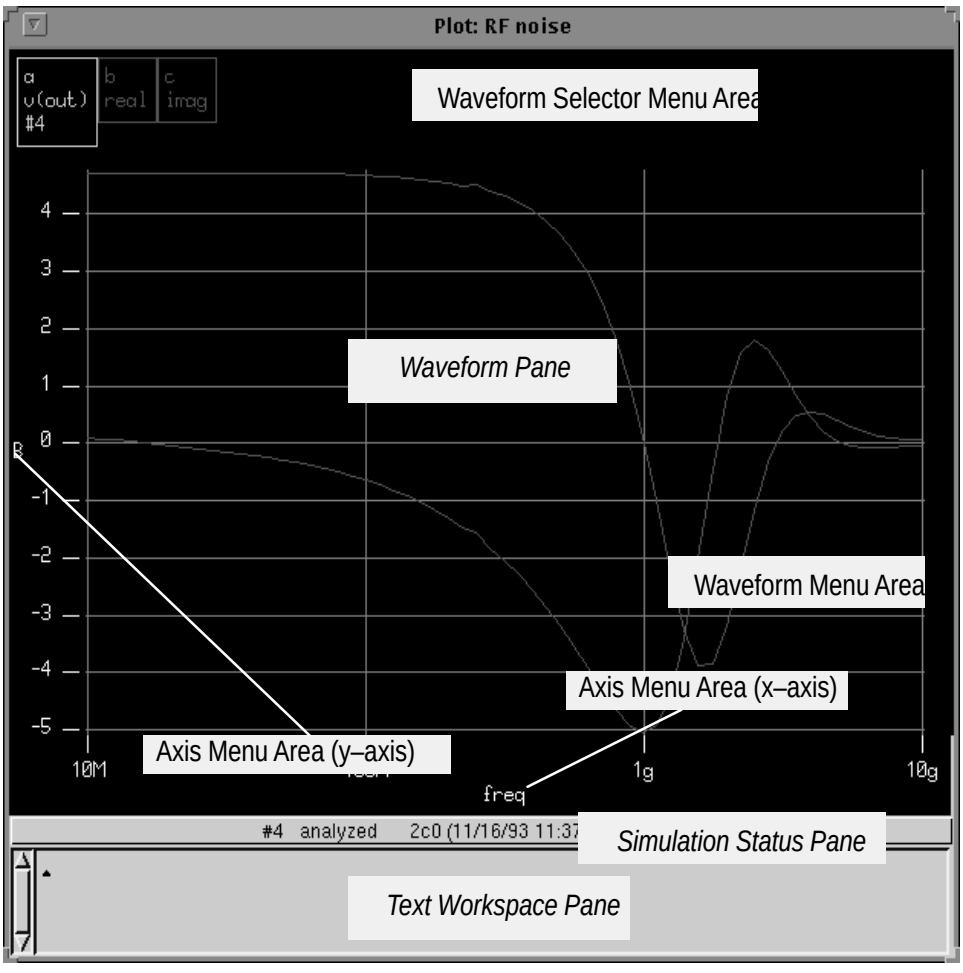


Figure 3–16: Analog Plot Browser Panes and Menu Areas

Access

Access the **Plot Browser** from several locations: 1) the *Schematic Editor Pane* of the **Circuit Editor** while in the Analysis Mode. Open the window using the pop–up button menu commands **plot** or **plot browser**; 2) by issuing a TekSpice plot command in the *Control Program Text Pane* of the **Control Program Browser** and selecting the **simulate** command; 3) from the **Launcher** using the **Open > plot browser** commands; 4) using the pop–up button menu command **plot** over a postage stamp; 5) using the pop–up button menu command **plot it** in any *Workspace Pane*.

Panes

Figure 3–16 shows the location of the panes in the **Plot Browser**.

A description of the panes in this window follows:

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

Valuator Pane

The *Valuator Pane* displays the x and y values of a selected position on a waveform in the *Waveform Pane* of a **Plot Browser**. This pane is a **transient pane and appears only when invoked by the popup button command** track **or** value **from the *Waveform Pane* of a** Plot Browser. This pane is terminated using the popup button command **done**.

Waveform Pane

The *Waveform Pane* is the output plot area for the **Plot Browser**. It is divided into three areas: the Axis Menu Area; the Waveform Menu Area; and the Waveform Selector Menu Area. Pop–up button menu commands depend on the location of the cursor.

Axis Menu Area. The Axis Menu Area commands allow the changing of the plot scale, plot labels and the general appearance of the axes. If the digital mode option is selcected, there can be multiple Y–axes, each of which can have a different scale, label, min/max, etc.

Waveform Menu Area. The Waveform Menu Area commands allow for editing and operating on plot waveforms. Adding labels, re–scaling axes and cutting and pasting of waveforms are allowable. The **track** mode allows reading data values following a waveform. Clicking on a point causes printing of the x and y values on the plot. The **delta** command reports value of an offset from the last point marked on the plot. The **value** mode is similar except that it is not constrained to follow a waveform. The **clear** command removes the tic marks.

Waveform Selector Menu Area. The Waveform Selector Menu Area commands allows adding and removing plot waveforms in the Waveform menu Area. When managing many waveforms in a window, those not of immediate interest can be invisible and later restored with a button click. Waveforms can be cut from a plot, and the remaining waveforms can be re–sequenced for improved clarity of colors, labels, and line styles. A label for all waveforms in a family of waveforms allows the total selection for **cut, copy** and make **invisible** commands.

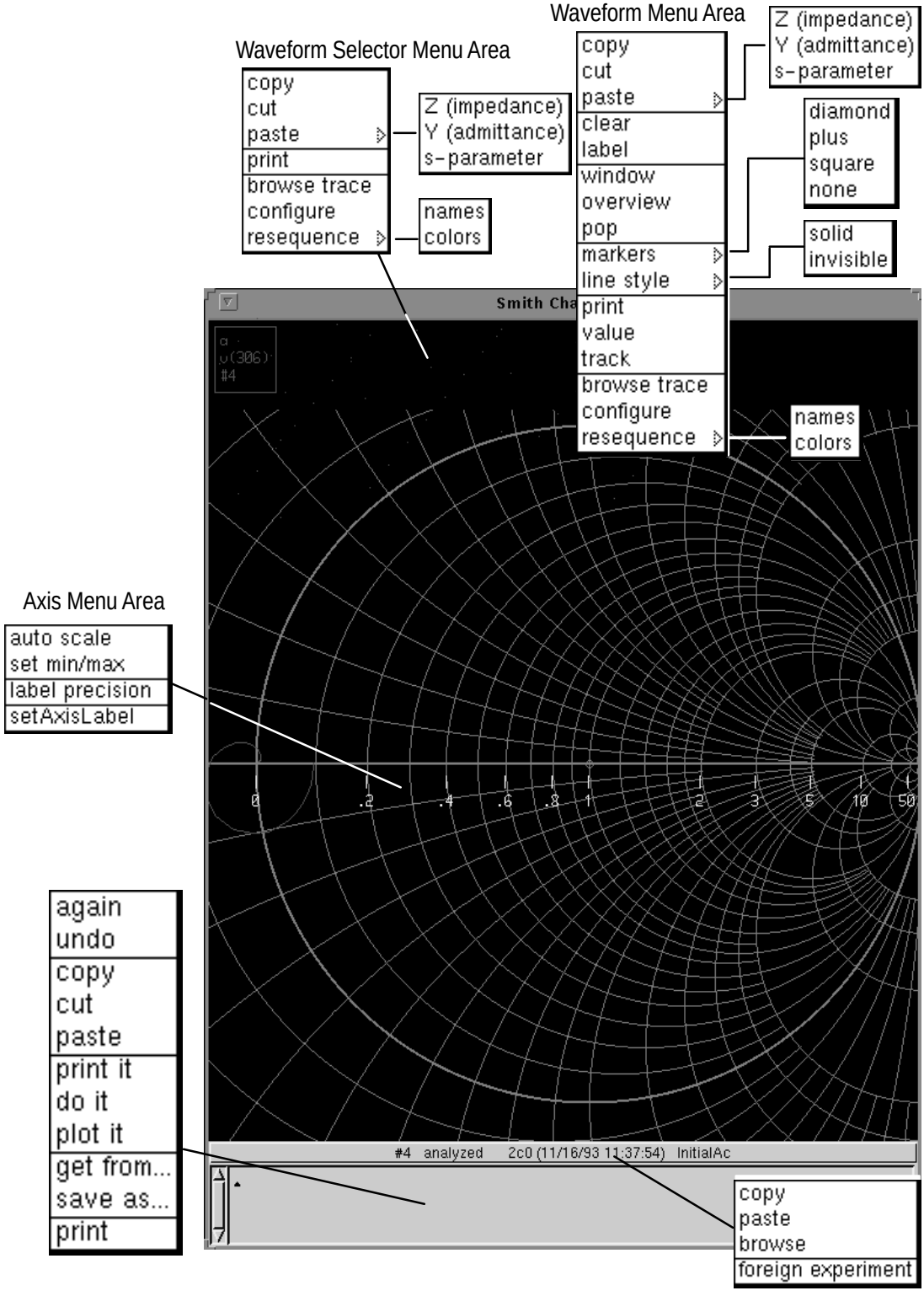


Figure 3–17: Smith Chart Browser Commands

Smith Chart Browser

The **Smith Chart Browser** is used for viewing S–parameter and Impedance (real and imaginary) data. This data can be generated from TekSpice outputs or from external files. Figure 3–17 shows the pop–up button commands available from the **Smith Chart Browser**.

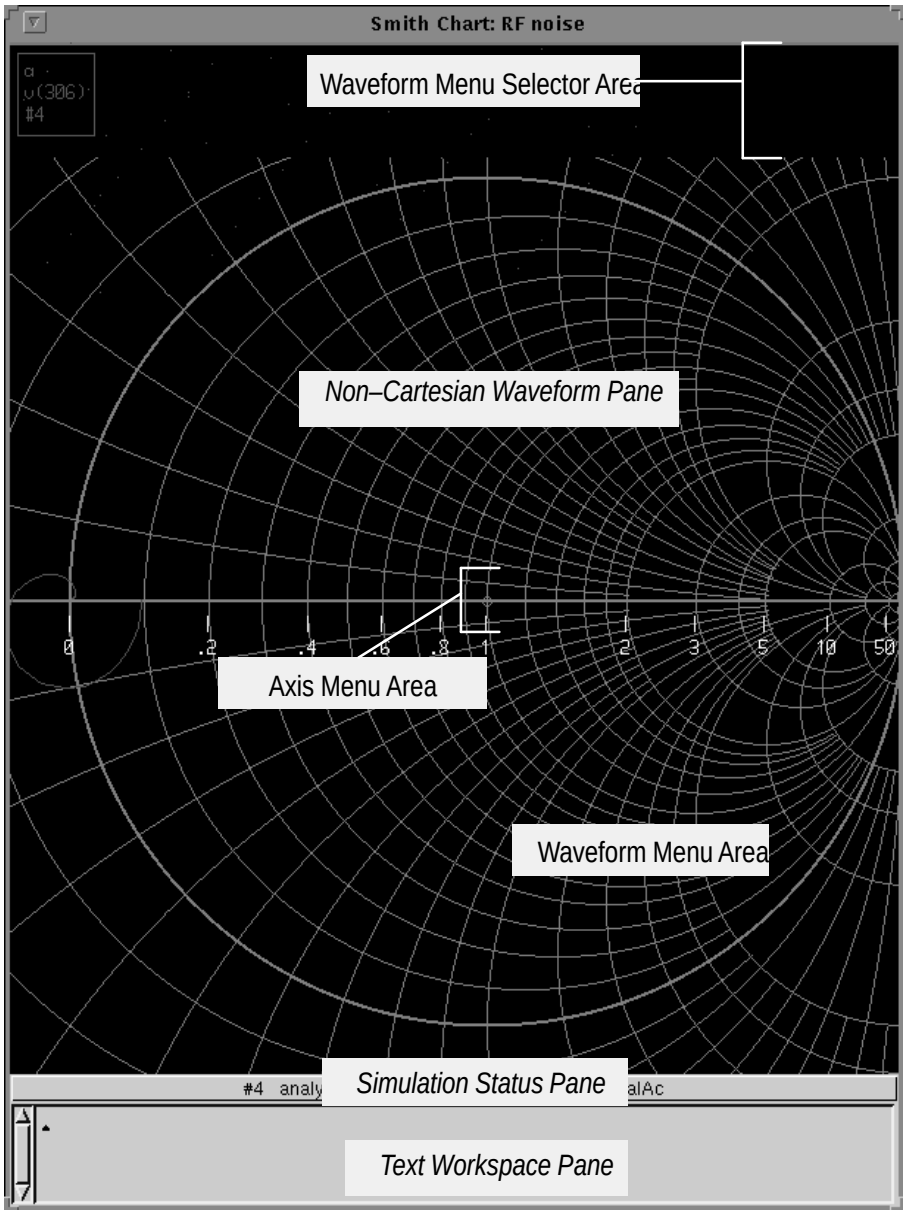


Figure 3–18: Smith Chart Browser Panes and Areas

Access

Access the **Smith Chart Browse**r from the **Circuit Editor** in the Analysis Mode. Select the pop–up button **Smith chart browser** command. To get a waveform in the *Waveform Pane*, place a postage stamp plot in the *Schematic Editor Pane* of the desired node, then **cut** (or **copy**) and **paste** this into the **Smith Chart Browser** *Waveform Pane*.

Panes

Figure 3–18 shows the location of the panes in the **Smith Chart Browser**.

A description of the panes in this window follows:

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

Non–Cartesian Waveform Pane

The *Non–Cartesian Waveform Pane* is the output plot area for the **Smith Plot Browser**. It is divided into three areas; the Axis Menu Area; the Waveform Menu Area; and the Waveform Selector Menu Area.

Axis Menu Area. The Axis Menu Area commands allow the changing of the plot scale, plot labels and the general appearance of the axes.

Waveform Menu Area. The Waveform Menu Area commands allow for editing and operating on plot waveforms. Adding labels, re–scaling axes and cutting and pasting of waveforms are allowable. The **track** mode allows reading data values

following a waveform. Clicking on a point causes printing of the x and y values on the plot. The **delta** command reports value of an offset from the last point marked on the plot. The **value** mode is similar except that it is not constrained to follow a waveform. The **clear** command removes the tic marks.

Waveform Selector Menu Area. The Waveform Selector Menu Area commands allow adding and removing plot waveforms in the Waveform Menu Area. When managing many waveforms in a window, those not of immediate interest can be made invisible and later restored with a button click. Waveforms can be cut from a plot, and the remaining can be re–sequenced for improved clarity of colors, labels, and line styles. A label for all waveforms in a family of waveforms allows the total selection for **delete/copy** and make **visible/invisible** commands.

Valuator Pane

The *Valuator Pane* displays the x and y values of a selected position on a waveform in the *Waveform Pane* of a **Plot Browser**. This pane is a **transient pane and appears only when invoked by the popup button command** track **or** value **from the *Waveform Pane* of a** Plot Browser. This pane is terminated using the popup button command **done**.

(This page intentionally left blank)

User Operating Point Browser

The **User Operating Point Browser** scans simulation outputs as specified in the model definitions. Information in the **User Operating Point Browser** is calculated directly from data in the ‘TekSpice results files (e.g., transrp, trfile) and presented in tabular format. Data for the table comes from the analysis selected in the **Control Program Editor**. If data are not available, the window does not open. This browser is useful for plotting parameter variations between circuit elements or evaluating expressions using data from the tables.

This browser differs from the **Operating Point Browser** in that the **Operating Point Browser** scans the simulation outputs as specified on the simulator’s built–in models.

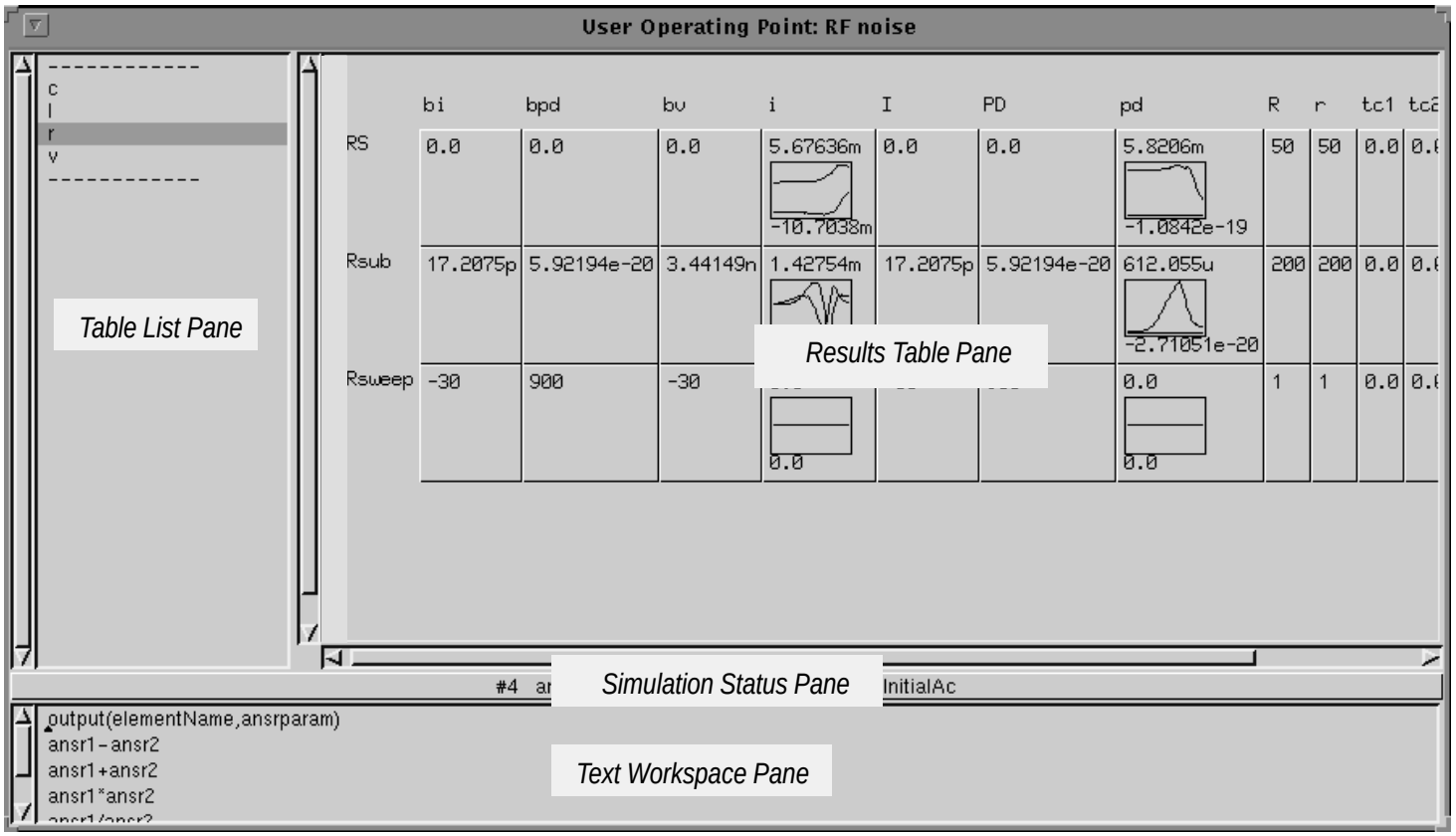


Figure 3–19: User Operating Point Browser Panes

Access

Access the **User Operating Point Browser** from; 1) in the **analysis mode** from the **Circuit Editor’s** *Schematic Editor Pane* using the pop–up button menu **open > user operating point table** command **;** 2) from the **Library Subcircuit Instance Editor**.

Panes

Figure 3–19 shows the location of the panes in the **User Operating Point Browser**.

A description of the panes in this window follows:

Results Table Pane

The *Results Table Pane* contains data from the selected table. If there is no data for the rows, labels appear instead of numbers. The pop–up button menu commands change depending on the selection of a row label, a column label, a single cell or nothing.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Table List Pane

The *Results Table Pane* displays the table highlighted in the *Table List Pane*. Only data for the specified output parameters are computed.

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

Editors

This reference describes the functionality of the windows within the ADS system. Although technically they are all windows, for the purposes of this manual they are divided into Launchers, Browsers, Editors and Dialog Boxes. Also described in this reference are panes and pop–up menu commands.

In this section the Editors are presented alphabetically by name. Each editor description contains the following information:

1. A brief description of each editor.
2. A figure showing the location of each pane in the editor.
3. The path to access the editors.
4. A description of each pane in the editor.

Listed below are the editors described in this section (in alphabetical order). The more commonly used editors are in **bold** typeface. The description of these editors include a figure showing the available pop–up button menus.

- **Circuit Editor**
- Library Path Editor
- Library Subcircuit Definition Editor
- Library Subcircuit Instance Editor
- **Library Subcircuit Symbol Editor**
- Library User Symbol Editor
- Model Reference Editor
- Node Symbol Editor
- User Symbol Editor
- Workspace Editor

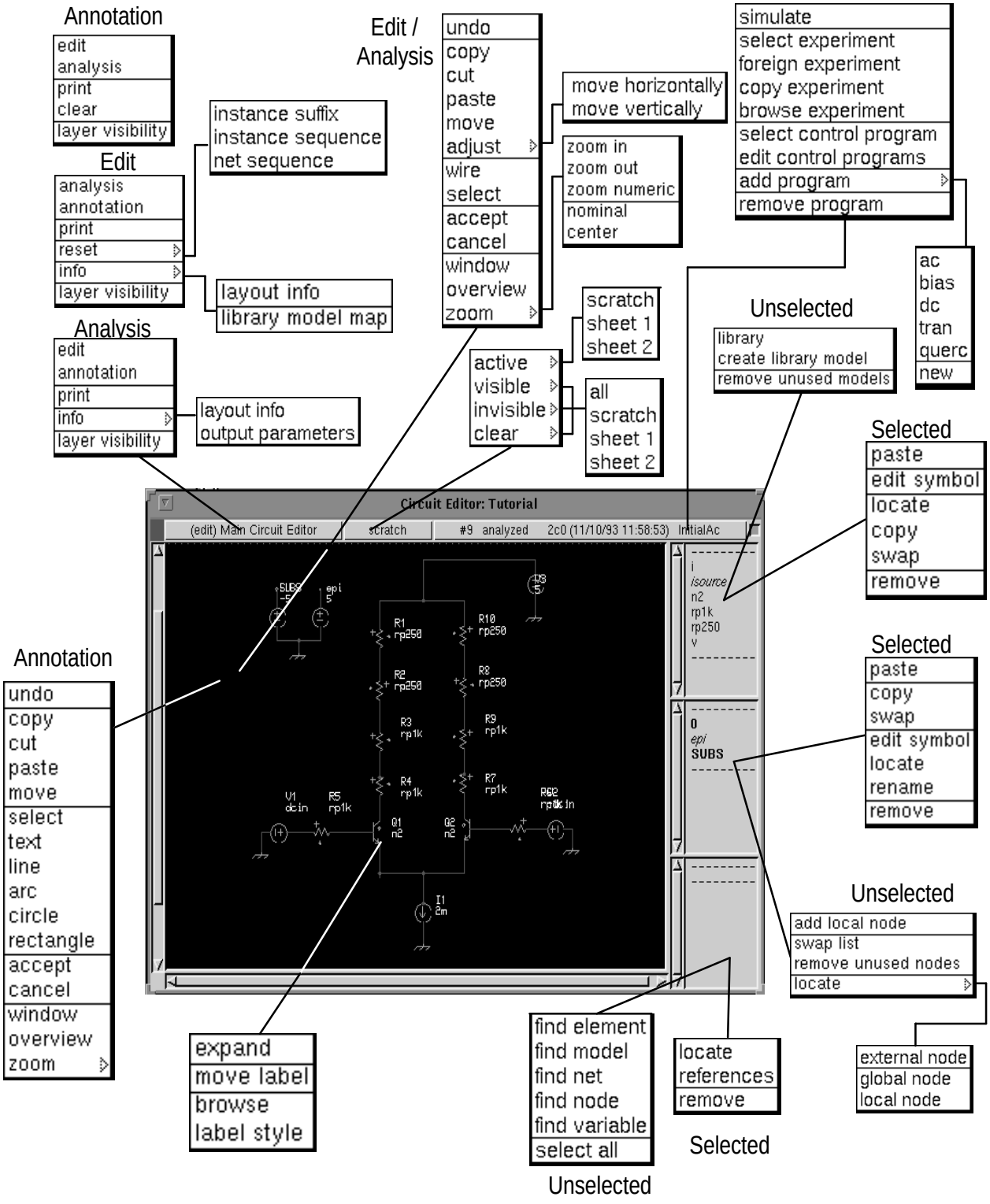


Figure 3–20: Circuit Editor Commands

Circuit Editor

The **Circuit Editor** is the main editor within the ADS system. This editor provides access to the majority of the browsers and editors within ADS. The **Circuit Editor** allows the creation and modification of circuit schematic diagrams, control of the simulation, and the display of simulation results (postage stamp plots) on the schematic. Figure 3–20 shows the commands available from the **Circuit Editor**.

Circuit schematics are developed by adding instances of elements, models and subcircuits, that are joined with wires or connectors as needed. Simulations may then be defined and executed, and outputs displayed directly on the schematic. The simulation results can also be saved for future reference.

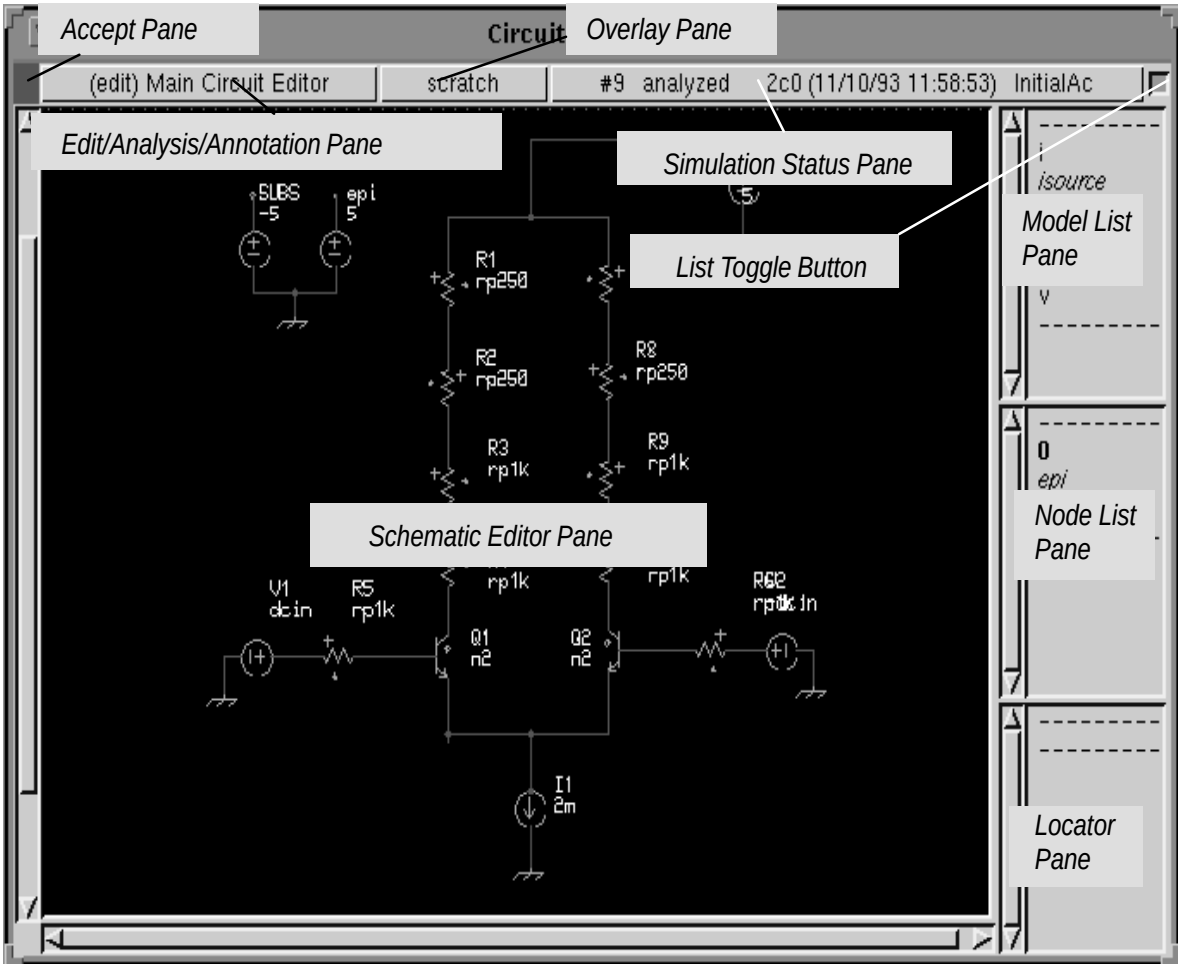


Figure 3–21: Circuit Editor Panes

Access

Access the **Circuit Editor** from the **Circuit Browser**. Select a circuit name from the **Circuit Browser** and select the pop–up button menu **edit circuit** command.

Panes

Figure 3–21 shows the location of the panes accessed from the **Circuit Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Edit/Analysis/ Annotation Pane

The *Edit/Analysis/Annotation Pane* affects the operation of the *Schematic Editor Pane*. Toggling this pane with the select button changes the displayed text between EDIT and ANALYSIS. The pop–up button allows selection of EDIT, ANALYSIS or ANNOTATION

List Toggle Button

The *List Toggle Button* either displays or hides the contents of the *Model List Pane*, *Node List Pane* and *Locator Pane*.

Locator Pane

The *Locator Pane* provides the ability to locate circuit elements (instances), models, nets and nodes. Locating these elements models or nodes is possible in the local circuit or at other levels of hierarchy. The symbol of the selected model element or node is identified by highlighting on the circuit.

Model List Pane

The *Model List Pane* provides the list of library models available for use in the creation of a circuit schematic. Pop–up button menu commands allow adding and removing models from this list.

Node List Pane

The *Node List Pane* provides the names of local and global nodes used in the circuit. Schematic symbols representing these nodes are attached to wires or terminals in the circuit to establish the node (net) name. The local nodes listed are those only used in the schematic displayed in the *Schematic Editor Pane*. Global nodes are listed in boldface print, local nodes listed in italics print. Global nodes are constant throughout the circuit hierarchy.

Overlay Pane

The *Overlay Pane* shows the selected sheet names and provides for their control. Sheets may be thought of as two layers of graphics overlaying the *Schematic Editor Pane*. The user may place simulation output postage stamp plots on either or both sheets for comparing outputs of one simulation to another. A third layer, scratch, is also available. This layer is less permanent and is erased every time a simulation is run.

Schematic Editor Pane

The *Schematic Editor Pane* is the main pane for creating and modifying circuit schematics and viewing simulation results. The operation of this pane depends on the mode of the *Edit/Analysis/Annotation Pane*. Elements are added to the *Schematic Editor Pane* by selecting a model in the *Model List Pane* and using the pop–up button menu **paste**. When pasting elements in the *Schematic Editor Pane*, ADS replaces the arrow cursor with the symbol of the selected model. Before placement, a pop–up button menu is available for orienting the symbol. For copying, moving and deleting wires and elements use the **copy**, **cut**, **paste**, **move** and **adjust** commands.

Edit Mode. The EDIT mode allows editing the circuit schematic in the *Schematic Editor pane*.

Analysis Mode. The Analysis mode allows displaying the results of a simulation in the *Schematic Editor Pane*.

Annotation Mode. The Annotation mode allows adding text to a schematic in the *Schematic Editor Pane*. This text is on a separate ”layer” and can be erased or made invisible. Editing circuit graphics is not possible in the Annotation mode. Editing annotation text is not possible in the Edit or Analysis mode.

Text is added by selecting the **text** command from the pop–up button menu while in the *Schematic Editor Pane*. The cursor switches to an I–Beam. Click the cursor at the point where the text should start and begin typing. The right margin is defined by pressing the **Enter** key on the numeric keypad and continue typing. Clicking on existing text, opens an editor for that text; the **resize** command in the text menu allows re–sizing the text; the **select** command accepts the current text and goes to select mode. The **accept** command accepts the text and closes the editor on the current text item, but stays in the text mode. The bounding box of each line of text is separately selected for editing.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

(This page intentionally left blank)

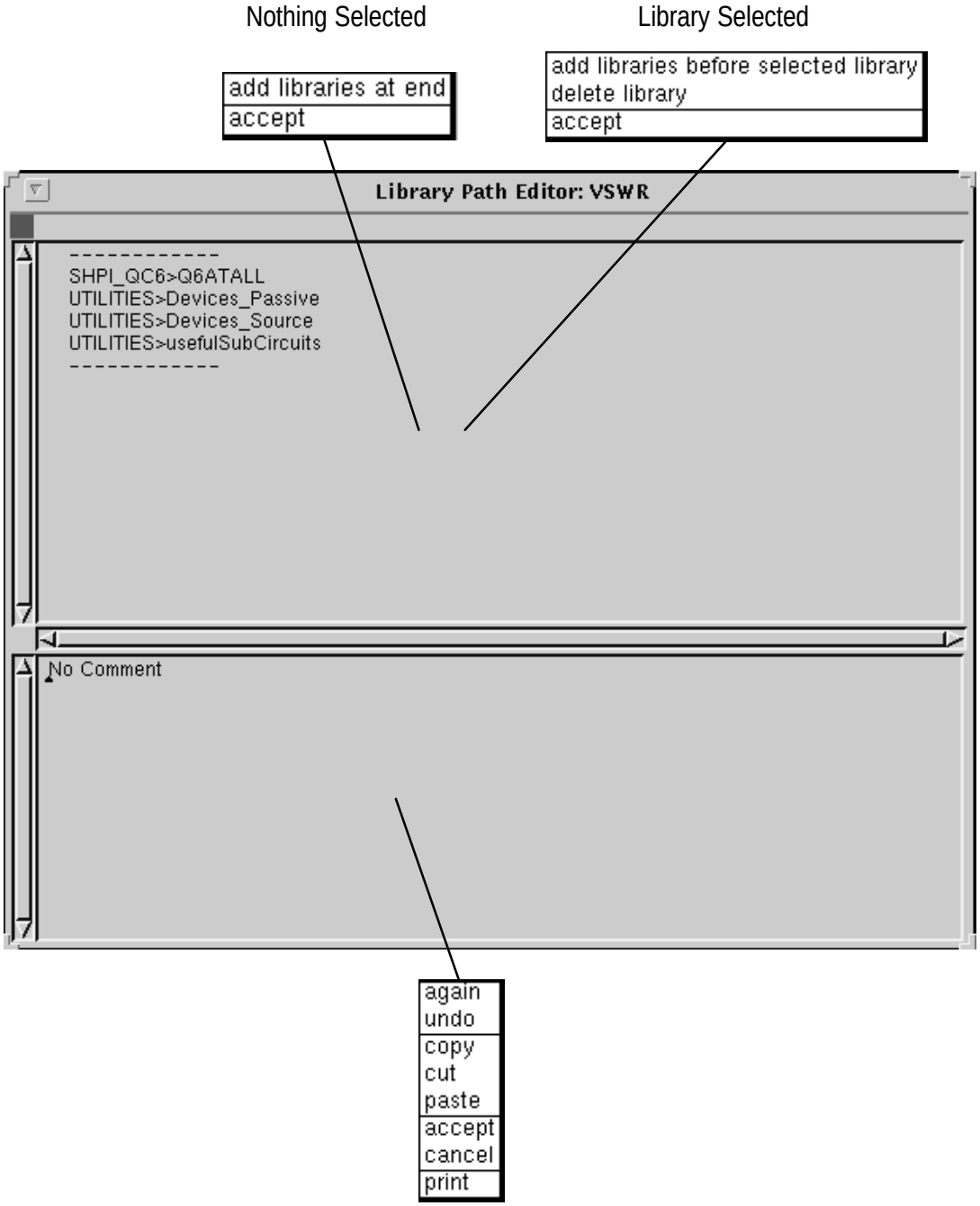


Figure 3–22: Library Path Editor Commands

Library Path Editor

The **Library Path Editor** lists the selected libraries used for the circuit specified by the **Circuit Browser**. Both internal and external libraries can be specified. For external libraries the full filename for the corresponding external library file appears on a LIBPATH statement in the generated TekSpice netlist. The order of the files in *Library Path Pane* determines the search order; the first instance of the library model found is the one used.

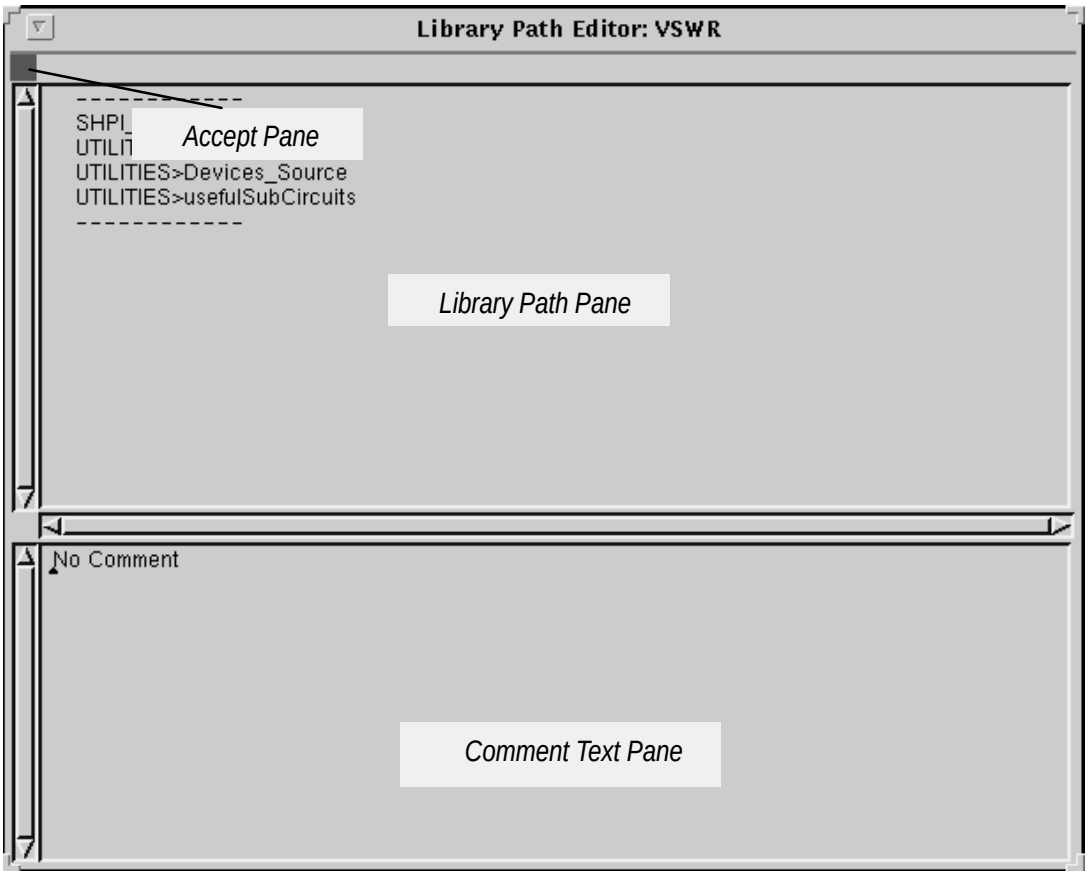


Figure 3–23: Library Path Editor Panes

Access

Access the **Library Path Editor** from the **Circuit Browser** using the pop–up button **set libraries** command with a circuit selected.

Panes

The **Library Path Editor** has two panes; the *Library Path Pane* and *Comment Text Pane*. Figure 3–23 shows the location of these panes.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Library Path Pane

The *Library Path Pane* lists the libraries available to the circuit specified by the **Circuit Browser**. Pop–up button commands allow adding or deleting libraries from this list. Any library listed in the **Library Browser** can be added to this list. Selecting a displayed library(s) and using the pop–up button accept command adds these library models to those available from the Circuit Editors’ *Model List Pane* pop–up button **library** command. The order of the libraries is important. If more than one of the libraries listed includes a model definition with the same name, the first occurrence in the list is used.The order of the library list can be changed by editing.

Comment Text Pane

The Comment Text Pane contains text describing the item selected in the active pane. Use the standard text editing commands in this pane. See the **Getting Started** chapter.

Library Subcircuit Definition Editor

The **Library Subcircuit Definition Editor** allows the defining of the element level representation of the subcircuit. It represents the combination of the **Circuit Editor** and the **Symbol Editor**. This subcircuit can access the library models listed in the *Model List Pane.* The subcircuit symbol can be edited in the *Symbol Editor Pane.* See Figure 3–24.

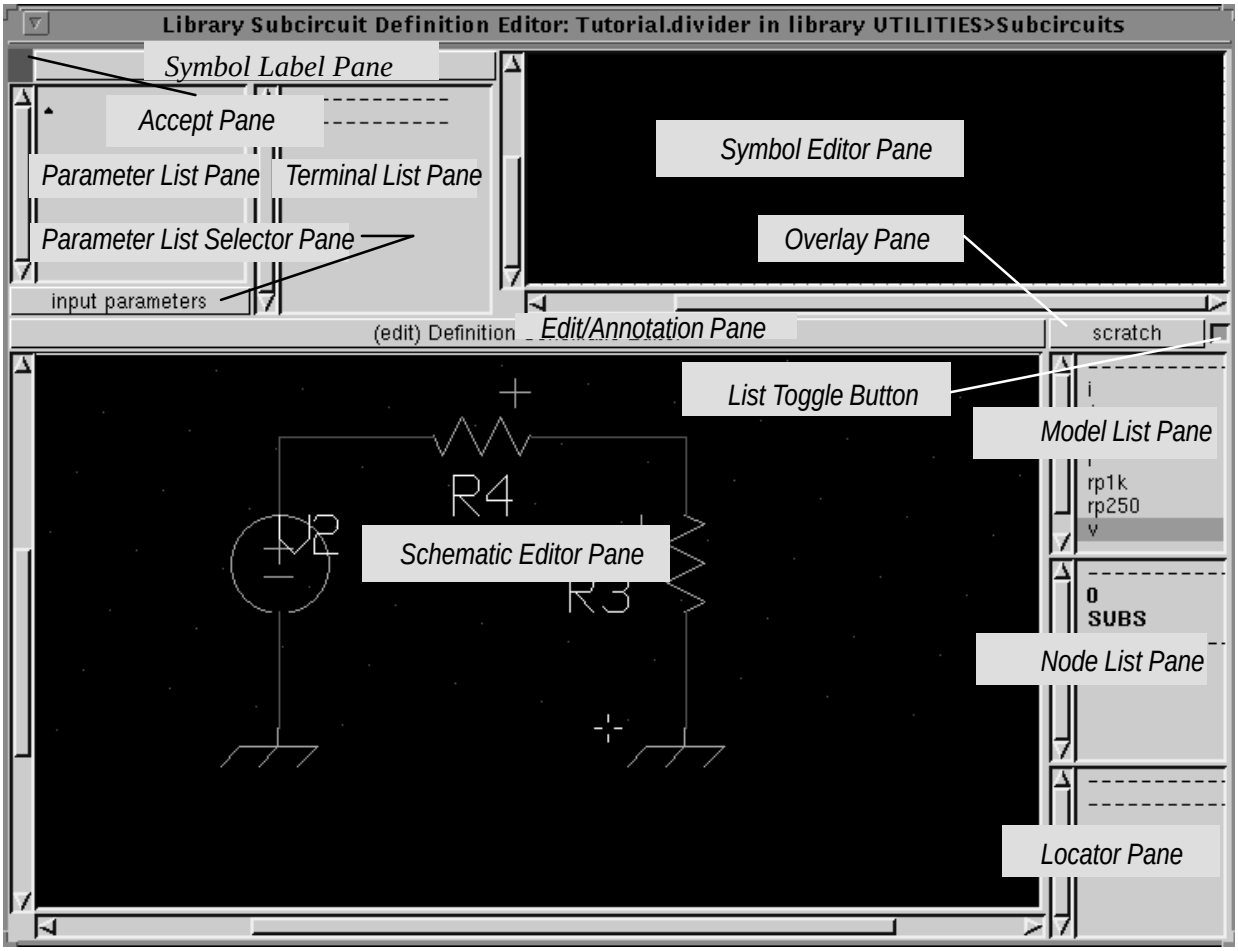


Figure 3–24: Library Subcircuit Definition Editor Panes

Access

Access the **Library Subcircuit Definition Editor** from the **Circuit Editor**'s *Model List Pane*. Highlight the *model* or *subcircuit* name and select the pop–up button **edit** command.

Panes

Figure 3–24 shows the location of the panes accessed from the **Library Subcircuit Definition Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane*. If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

List Toggle Button

The *List Toggle Button* either displays or hides the contents of the *Model List Pane*, *Node List Pane* and *Locator Pane*.

Locator Pane

The *Locator Pane* provides the ability to locate circuit elements (instances), models, nets and nodes. Locating these elements models or nodes is possible in the local circuit or at other levels of hierarchy. The symbol of the selected model element or node is identified by highlighting on the circuit.

Model List Pane

The *Model List Pane* provides the list of library models available for use in the creation of a circuit schematic. Pop–up button menu commands allow adding and removing models from this list.

Node List Pane

The *Node List Pane* provides the names of local and global nodes used in the circuit. Schematic symbols representing these nodes are attached to wires or terminals in the circuit to establish the node (net) name. The local nodes listed are those only used in the schematic displayed in the *Schematic Editor Pane*. Global nodes are listed in boldface print, local nodes listed in italics print. Global nodes are constant throughout the circuit hierarchy.

Parameter List Pane

The *Parameter List Pane* defines input parameters, assign values to them and defines output parameters. The displayed names are either input or output parameter names or local variable names depending on the mode set in the *Parameter List Selector Pane*. If a user symbol is being created, local variables may not be created and only value changes can be made for the input parameters.

The syntax for defining output parameters for subcircuit models is as follows:

output(element, parameter) is the value of the named **parameter** on the named **element**. Element name is relative to the definition of the subcircuit where the output function is used.

v(N) is the swept voltage on node N, where N is named relative to the definition of the subcircuit where **v()** is used.

bv(N) is the bias (operating point) voltage on node N where N is named relative to the definition of the subcircuit where the **bv()** is used.

i(element,terminal number) is the swept current through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where **i()** is used.

bi(element,terminal number) is the bias (operating point) cuirrent through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where the **bi()** is used.

The syntax for primitive and user models is as follows (where T is an integer greater than 0 and less than or equal the terminal count on the model):

primitivebv(T) is the operating point voltage at terminal T.

primitivebi(T) is the operating point current into terminal T.

primitivev(T) is the swept voltage at terminal number T of the given element.

primitiveoutput(P) is the swept value of parameter P defined in the primitive model for the given element from the swept analysis (Ac, dc, or Tran). If not defined here then it is undefined.

primitiveparameter(P) is the value of parameter P defined in the primitive model for the given element from the initial (bias, dc, ac or tran) analysis. If not defined here then it is from the 'tempAdjust' outputs. If not defined here then it is from the initial parameter values. If not defined here then it is undefined.

Parameter List Selector Pane

The *Parameter List Selector Pane* allows setting the operating mode of the *Parameter List Pane*. Setting the mode to **input parameters** causes the parameter names and value in the *Parameter List Pane* to represent subcircuit input parameters. Setting the mode to **output parameters** causes the parameter names and values in the *Parameter List Pane* to represent subcircuit output parameters. Setting the mode to **local variables** causes the parameter names and values in the *Parameter List Pane* to represent expressions and assignments used to set subcircuit element values. The **local variables** mode is not available from the **Library User Symbol Editor**.

Schematic Editor Pane

The *Schematic Editor Pane* is the main pane for creating and modifying circuit schematics and viewing simulation results. The operation of this pane depends on the mode of the *Edit/Analysis/Annotation Pane*. Elements are added to the *Schematic Editor Pane* by selecting a model in the *Model List Pane* and using

the pop–up button menu **paste**. When pasting elements in the *Schematic Editor Pane*, ADS replaces the arrow cursor with the symbol of the selected model. Before placement, a pop–up button menu is available for orienting the symbol. For copying, moving and deleting wires and elements use the **copy**, **cut**, **paste**, **move** and **adjust** commands.

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Terminal List Pane

The *Terminal List Pane* supports access to the circuit and subcircuit connection terminals (ports). These terminals define the connections from the local circuit (the one in the *Schematic Editor Pane*) to the next level of circuit hierarchy. Terminals from this list must be pasted into the *Symbol Editor Pane* by selecting the terminal in the *Terminal List Pane* and then using the pop–up button command **paste terminal**, paste the terminal in the *Symbol Editor Pane*. If the terminal already exists then the **paste terminal** command does not appear in the pop–up button menu.

Library Subcircuit Instance Editor

The **Library Subcircuit Instance Editor** allows the modification of a particular instance of a subcircuit within a schematic. Simulation results unique to the instance may be viewed. For convenience, schematic editing of the subcircuit definition is permitted here. Note that changes made to the subcircuit definition apply to all instances of the subcircuit.

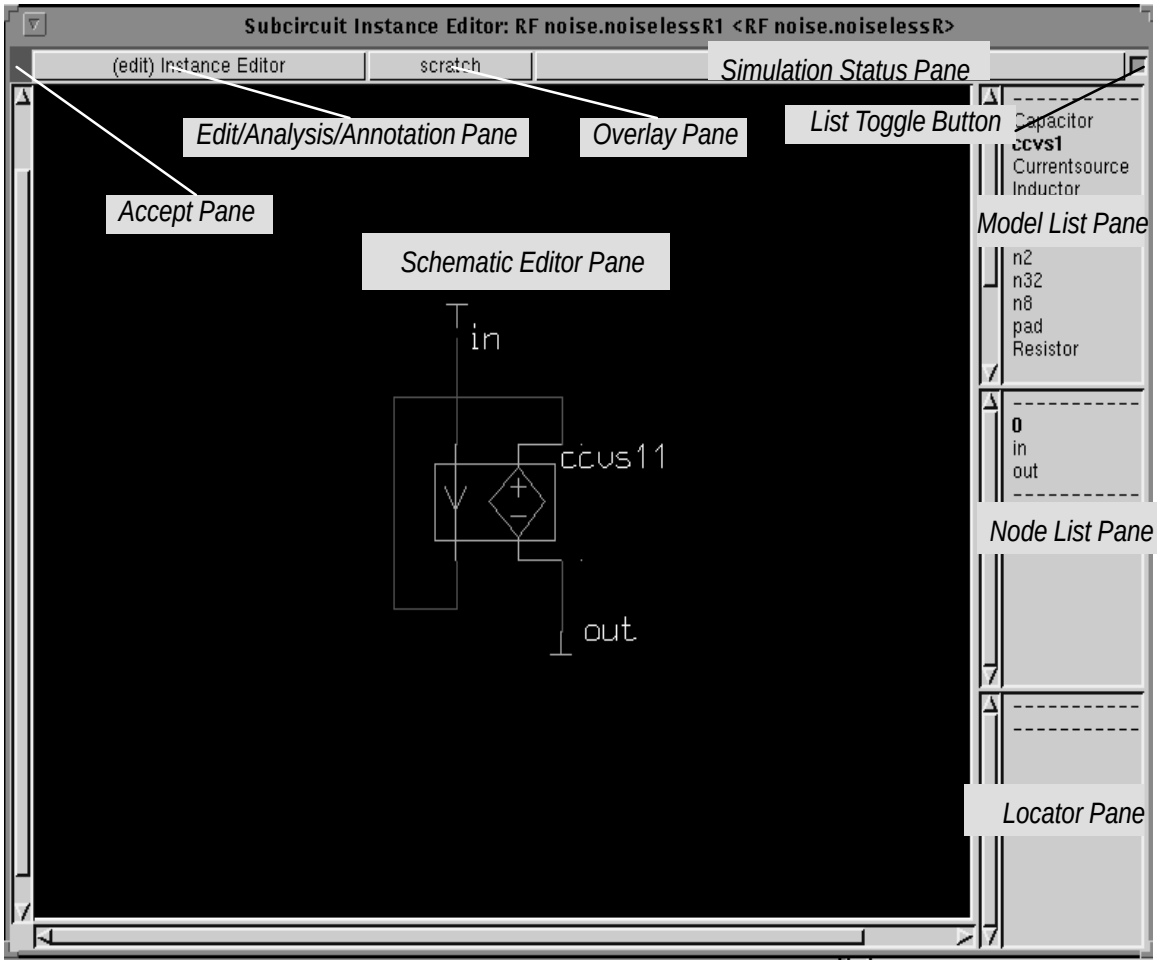


Figure 3–25: Subcircuit Instance Editor Panes

Access

Access the **Library Subcircuit Instance Editor** from the **Circuit Editor**'s *Schematic Editor Pane*. **Select** the *mode*l or *subcircuit* in the schematic and then select the pop–up button menu **expand** command.

Panes

Figure 3–25 shows the location of the panes accessed from the **Library Subcircuit Instance Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane*. If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

List Toggle Button

The *List Toggle Button* either displays or hides the contents of the *Model List Pane*, *Node List Pane* and *Locator Pane*.

Locator Pane

The *Locator Pane* provides the ability to locate circuit elements (instances), models, nets and nodes. Locating these elements models or nodes is possible in the local circuit or at other levels of hierarchy. The symbol of the selected model element or node is identified by highlighting on the circuit.

Model List Pane

The *Model List Pane* provides the list of library models available for use in the creation of a circuit schematic. Pop–up button menu commands allow adding and removing models from this list.

Node List Pane

The *Node List Pane* provides the names of local and global nodes used in the circuit. Schematic symbols representing these nodes are attached to wires or terminals in the circuit to establish the node (net) name. The local nodes listed are those only used in the schematic displayed in the *Schematic Editor Pane*. Global nodes are listed in boldface print, local nodes listed in italics print. Global nodes are constant throughout the circuit hierarchy.

Schematic Editor Pane

The *Schematic Editor Pane* is the main pane for creating and modifying circuit schematics and viewing simulation results. The operation of this pane depends on the mode of the *Edit/Analysis/Annotation Pane*. Elements are added to the *Schematic Editor Pane* by selecting a model in the *Model List Pane* and using the pop–up button menu **paste**. When pasting elements in the *Schematic Editor Pane*, ADS replaces the arrow cursor with the symbol of the selected model. Before placement, a pop–up button menu is available for orienting the symbol. For copying, moving and deleting wires and elements use the **copy**, **cut**, **paste**, **move** and **adjust** commands.

(This page intentionally left blank)

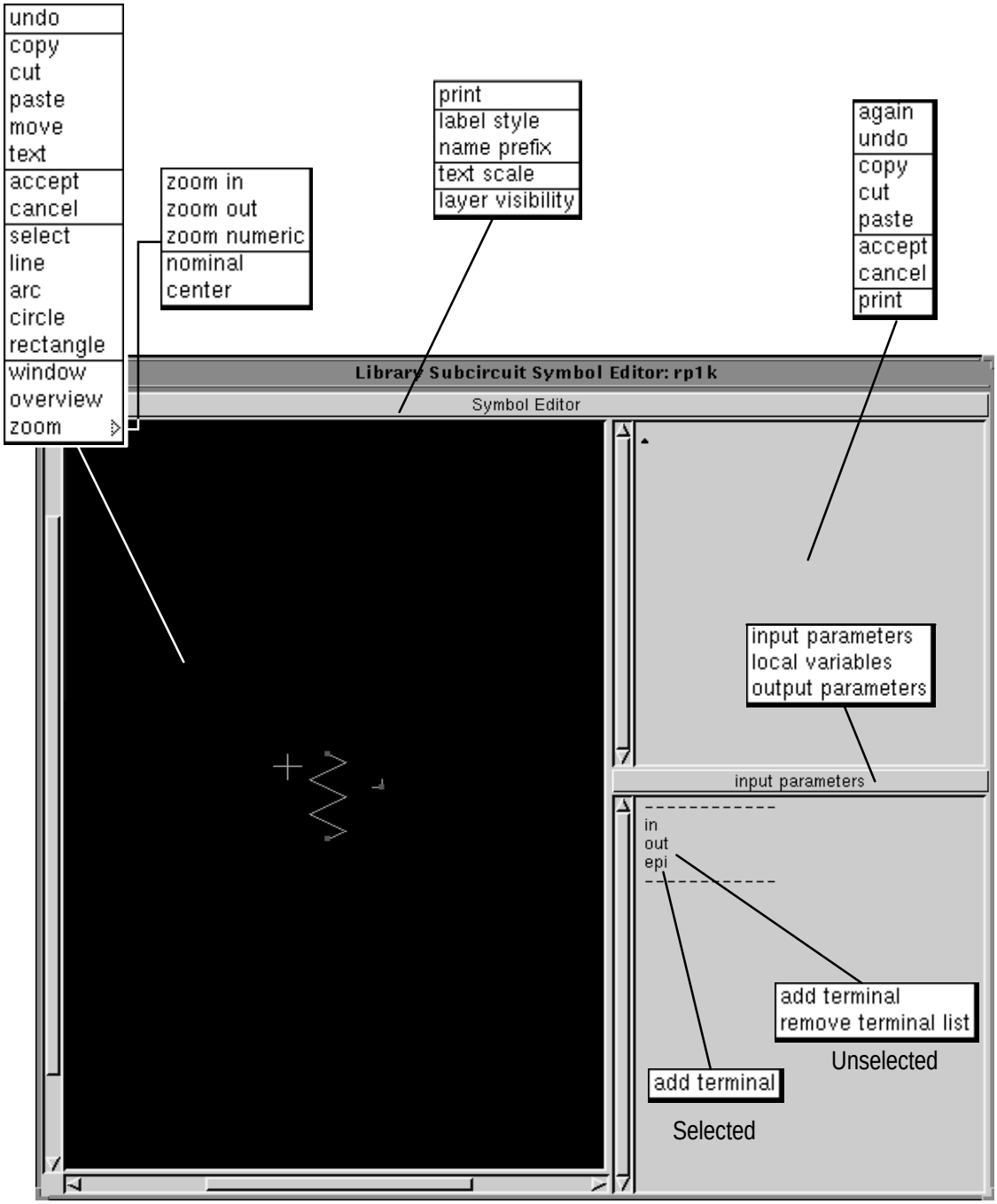


Figure 3–26: Library Subcircuit Symbol Editor Commands

Library Subcircuit Symbol Editor

The **Library Subcircuit Symbol Editor** allows the creation of symbols to represent a subcircuit in a hierarchical schematic. Along with the symbol the subcircuit parameters, terminals and terminal locations are also defined. The editing of symbols in this editor is identical to editing symbols in the Library **Subcircuit Definition Editor**, except the *Symbol Editor Pane* is larger.

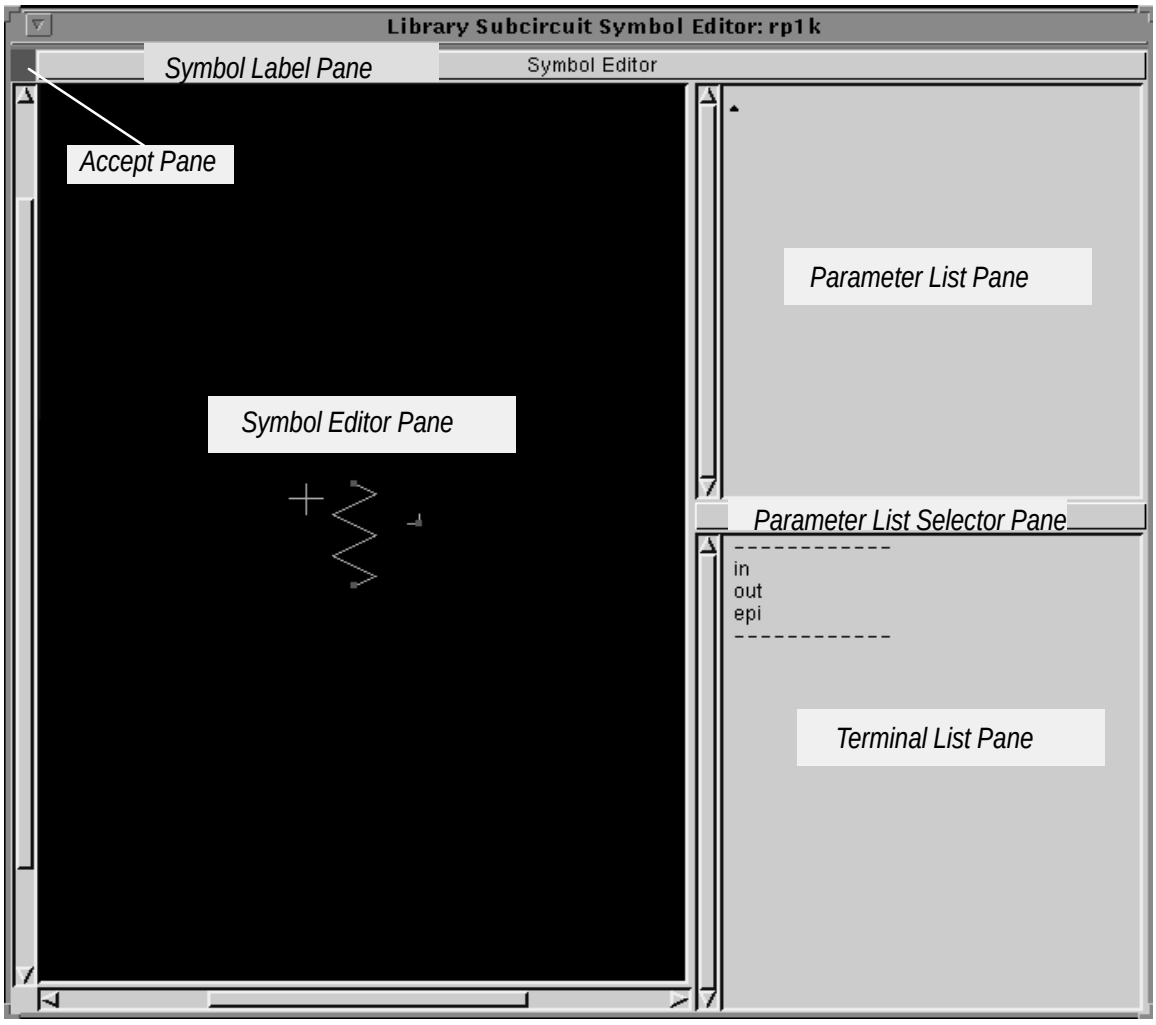


Figure 3–27: Subcircuit Symbol Editor Panes

Access

Access the **Library Subcircuit Symbol Editor** from the **Circuit Editor** or **Subcircuit Editor**'s *Model List Pane*. Highlight the library model or subcircuit name and select the pop–up button **edit symbol** command.

Panes

Figure 3–27 shows the location of the panes accessed from the **Library Subcircuit Symbol Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Parameter List Pane

The *Parameter List Pane* defines input parameters, assign values to them and defines output parameters. The displayed names are either input or output parameter names or local variable names depending on the mode set in the *Parameter List Selector Pane*. If a user symbol is being created, local variables may not be created and only value changes can be made for the input parameters.

The syntax for defining output parameters for subcircuit models is as follows:

output(element, parameter) is the value of the named **parameter** on the named **element**. Element name is relative to the definition of the subcircuit where the output function is used.

v(N) is the swept voltage on node N, where N is named relative to the definition of the subcircuit where **v()** is used.

bv(N) is the bias (operating point) voltage on node N where N is named relative to the definition of the subcircuit where the **bv()** is used.

i(element,terminal number) is the swept current through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where **i()** is used.

bi(element,terminal number) is the bias (operating point) cuirrent through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where the **bi()** is used.

The syntax for primitive and user models is as follows (where T is an integer greater than 0 and less than or equal the terminal count on the model):

primitivebv(T) is the operating point voltage at terminal T.

primitivebi(T) is the operating point current into terminal T.

primitivev(T) is the swept voltage at terminal number T of the given element.

primitiveoutput(P) is the swept value of parameter P defined in the primitive model for the given element from the swept analysis (Ac, dc, or Tran). If not defined here then it is undefined.

primitiveparameter(P) is the value of parameter P defined in the primitive model for the given element from the initial (bias, dc, ac or tran) analysis. If not defined here then it is from the 'tempAdjust' outputs. If not defined here then it is from the initial parameter values. If not defined here then it is undefined.

Parameter List Selector Pane

The *Parameter List Selector Pane* allows setting the operating mode of the *Parameter List Pane*. Setting the mode to **input parameters** causes the parameter names and value in the *Parameter List Pane* to represent subcircuit input parameters. Setting the mode to **output parameters** causes the parameter names and values in the *Parameter List Pane* to represent subcircuit output parameters. Setting the mode to **local variables** causes the parameter names and values in the *Parameter List Pane* to represent expressions and assignments used to set subcircuit element values. The **local variables** mode is not available from the **Library User Symbol Editor**.

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Terminal List Pane

The *Terminal List Pane* supports access to the circuit and subcircuit connection terminals (ports). These terminals define the connections from the local circuit (the one in the *Schematic Editor Pane*) to the next level of circuit hierarchy. Terminals from this list must be pasted into the *Symbol Editor Pane* by selecting the terminal in the *Terminal List Pane* and then using the pop–up button command **paste terminal**, paste the terminal in the *Symbol Editor Pane*. If the terminal already exists then the **paste terminal** command does not appear in the pop–up button menu.

(This page intentionally left blank)

Library User Symbol Editor

The **Library User Symbol Editor** allows the creation of user symbols which represent a library model in a schematic. A user symbol is a modified version of a basic model symbol such as a resistor. These library symbols are available from the Library Process: UTILITIES; in the Libraries: Devices_Passive, Devices_Semiconductor, and Devices_Sources. They are also available from copies of these model symbols.

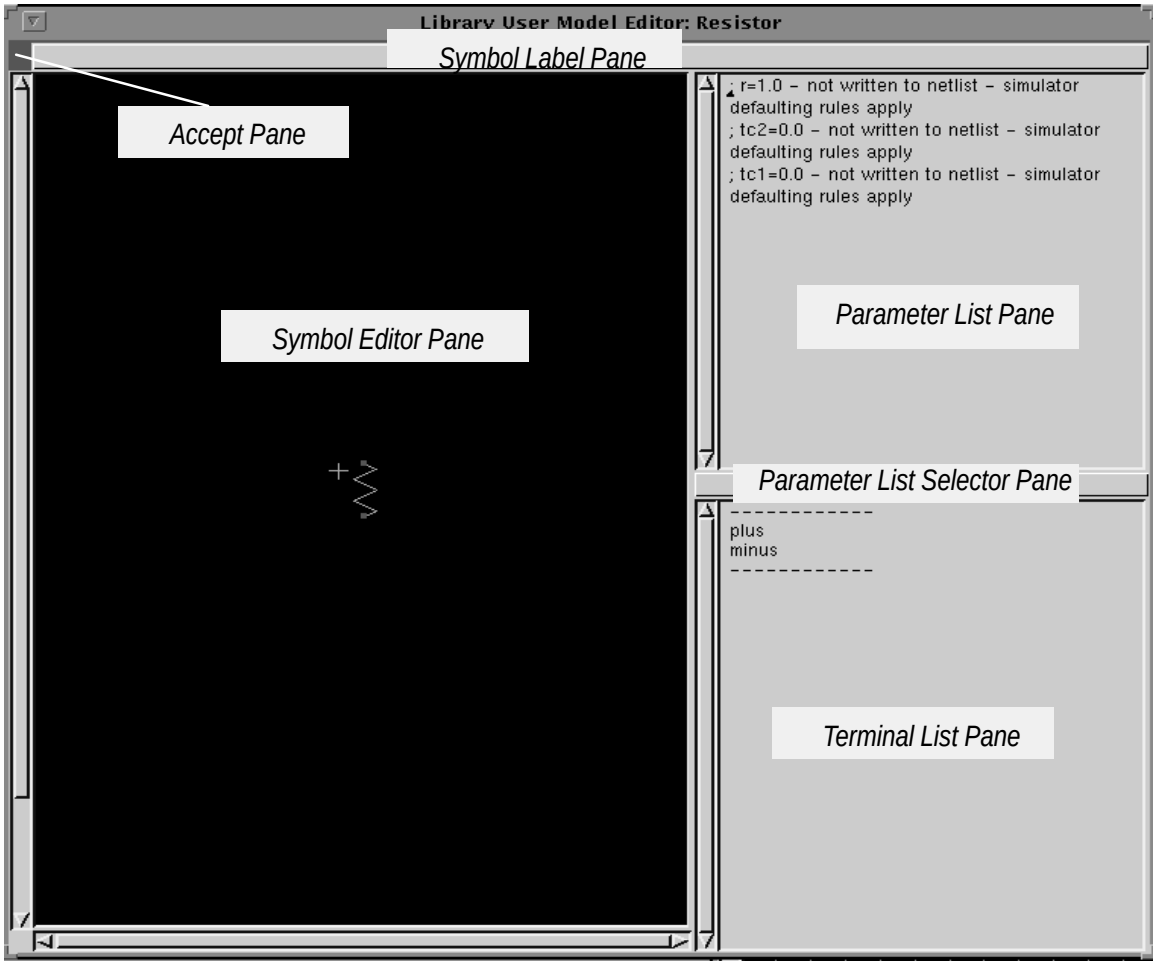


Figure 3–28: Library User Symbol Editor Panes

Access

Access the **Library User Symbol Editor** from either 1) the **Circuit Editor**'s *Schematic Editor Pane*; **select** the user *symbol* in the schematic and select the pop–up button **expand** command; or 2) the **Circuit Editor**'s *Model List Pane*; select the model name and use the pop–up button **edit symbol** command.

Panes

Figure 3–28 shows the location of the panes accessed from the **Library User Symbol Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Parameter List Pane

The *Parameter List Pane* defines input parameters, assign values to them and defines output parameters. The displayed names are either input or output parameter names or local variable names depending on the mode set in the *Parameter List Selector Pane*. If a user symbol is being created, local variables may not be created and only value changes can be made for the input parameters.

The syntax for defining output parameters for subcircuit models is as follows:

output(element, parameter) is the value of the named **parameter** on the named **element**. Element name is relative to the definition of the subcircuit where the output function is used.

v(N) is the swept voltage on node N, where N is named relative to the definition of the subcircuit where **v()** is used.

bv(N) is the bias (operating point) voltage on node N where N is named relative to the definition of the subcircuit where the **bv()** is used.

i(element,terminal number) is the swept current through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where **i()** is used.

bi(element,terminal number) is the bias (operating point) cuirrent through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where the **bi()** is used.

The syntax for primitive and user models is as follows (where T is an integer greater than 0 and less than or equal the terminal count on the model):

primitivebv(T) is the operating point voltage at terminal T.

primitivebi(T) is the operating point current into terminal T.

primitivev(T) is the swept voltage at terminal number T of the given element.

primitiveoutput(P) is the swept value of parameter P defined in the primitive model for the given element from the swept analysis (Ac, dc, or Tran). If not defined here then it is undefined.

primitiveparameter(P) is the value of parameter P defined in the primitive model for the given element from the initial (bias, dc, ac or tran) analysis. If not defined here then it is from the 'tempAdjust' outputs. If not defined here then it is from the initial parameter values. If not defined here then it is undefined.

Parameter List Selector Pane

The *Parameter List Selector Pane* allows setting the operating mode of the *Parameter List Pane*. Setting the mode to **input parameters** causes the parameter names and value in the *Parameter List Pane* to represent subcircuit input parameters. Setting the mode to **output parameters** causes the parameter names and values in the *Parameter List Pane* to represent subcircuit output parameters. Setting the mode to **local variables** causes the parameter names and values in the *Parameter List Pane* to represent expressions and assignments used to set subcircuit element values. The **local variables** mode is not available from the **Library User Symbol Editor**.

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Terminal List Pane

The *Terminal List Pane* supports access to the circuit and subcircuit connection terminals (ports). These terminals define the connections from the local circuit (the one in the *Schematic Editor Pane*) to the next level of circuit hierarchy. Terminals from this list must be pasted into the *Symbol Editor Pane* by selecting the terminal in the *Terminal List Pane* and then using the pop–up button command **paste terminal**, paste the terminal in the *Symbol Editor Pane*. If the terminal already exists then the **paste terminal** command does not appear in the pop–up button menu.

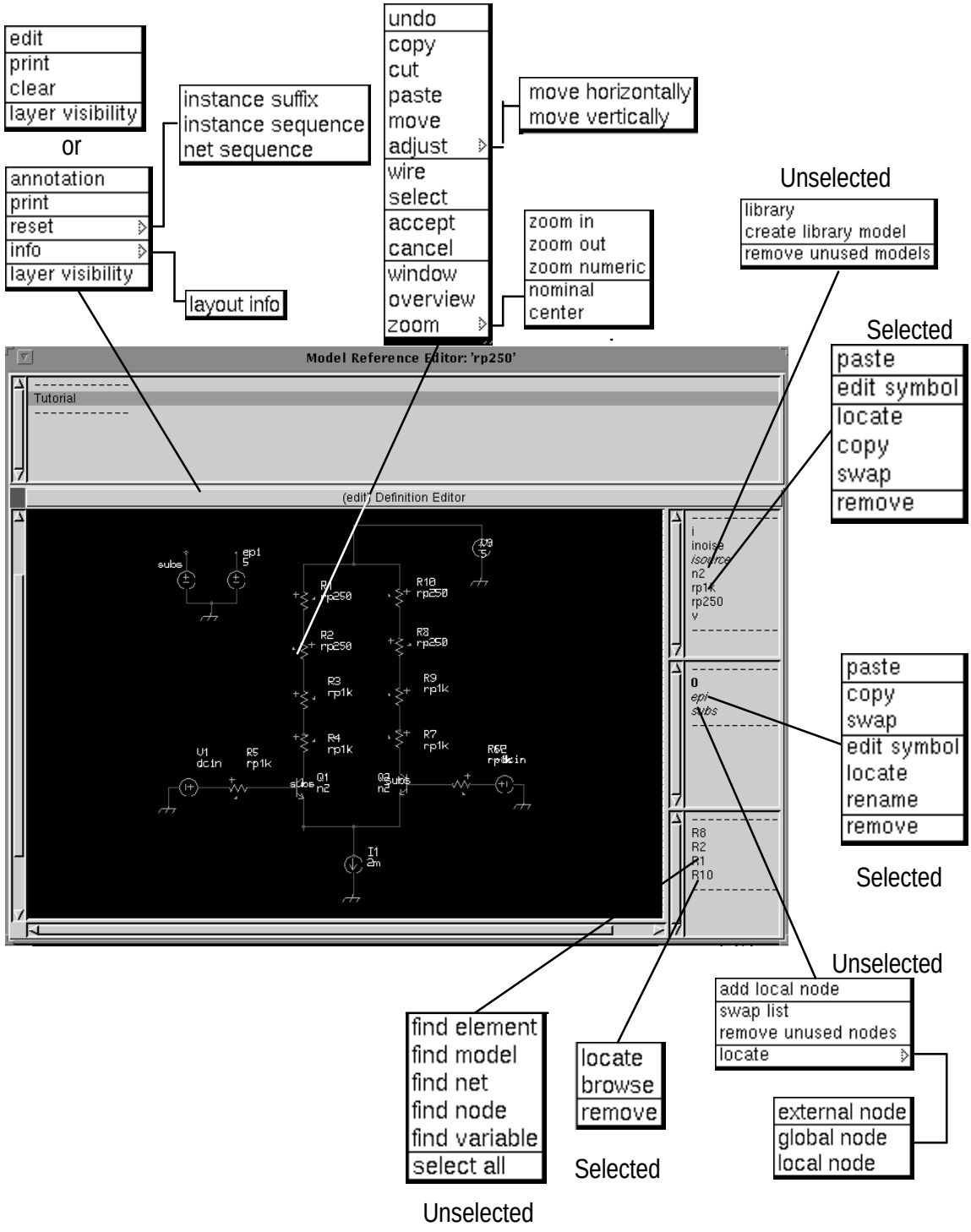


Figure 3–29: Model Reference Editor Commands

Model Reference Editor

The **Model Reference Editor** is identical to the **Circuit Editor** with the addition of a *Reference Circuit List Pane* and the elimination of a *Simulation Status Pane* on the **Model Reference Editor**.

The *Circuit List Pane* lists all locations of a selected model within the hierarchy of a circuit. The model to be located is highlighted in the *Locator Pane* and, a list of instance locations displayed in the *Circuit List Pane.* The user may select from this list and modify the schematic portion of the definition by adding or deleting primitives, models, subcircuits and wires (in the *Schematic Editor Pane*).

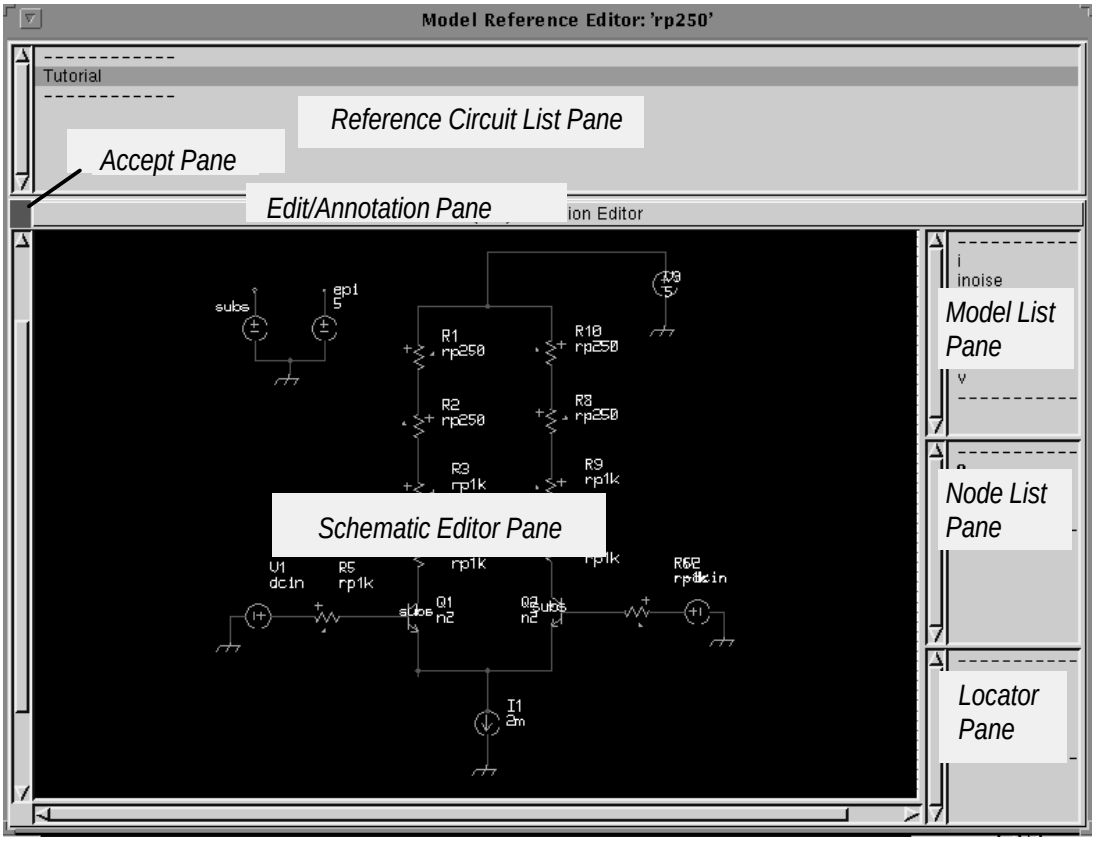


Figure 3–30: Model Reference Editor Panes

Access

Access the **Model Reference Editor** from the **Circuit Editor**'s *Locator Pane* using the **references** command in the pop–up button menu. Highlight the model in the *Model List Pane* and select the pop–up button **locate** command. The model name appears in the *Locator Pane*. Highlight this model and select the pop–up button **references** command.

Panes

Figure 3–30 shows the location of the panes in the **Model References Editor**.

NOTE. *Any edit changes made to the displayed subcircuit changes the general definition of the subcircuit and all instances of that subcircuit.*

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane*. If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Reference Circuit List Pane

The *Reference Circuit List Pane* lists the subcircuits within the circuit hierarchy containing a selected model. Selecting a subcircuit from this list displays a schematic of this subcircuit is in the *Schematic Editor Pane*. The *Model List Pane* lists the associated models; the *Node List Pane* lists the nodes. The ”located” model appears in the *Locator Pane*. Selecting this model can then locate its position in the schematic.

Edit/ Annotation Pane

The *Edit/Annotation Pane* affects the operation of the *Schematic Editor Pane*. Toggling this pane with the select button changes the displayed text between EDIT and ANNOTATION.

Locator Pane

The *Locator Pane* provides the ability to locate circuit elements (instances), models, nets and nodes. Locating these elements models or nodes is possible in the local circuit or at other levels of hierarchy. The symbol of the selected model element or node is identified by highlighting on the circuit.

Model List Pane

The *Model List Pane* provides the list of library models available for use in the creation of a circuit schematic. Pop–up button menu commands allow adding and removing models from this list.

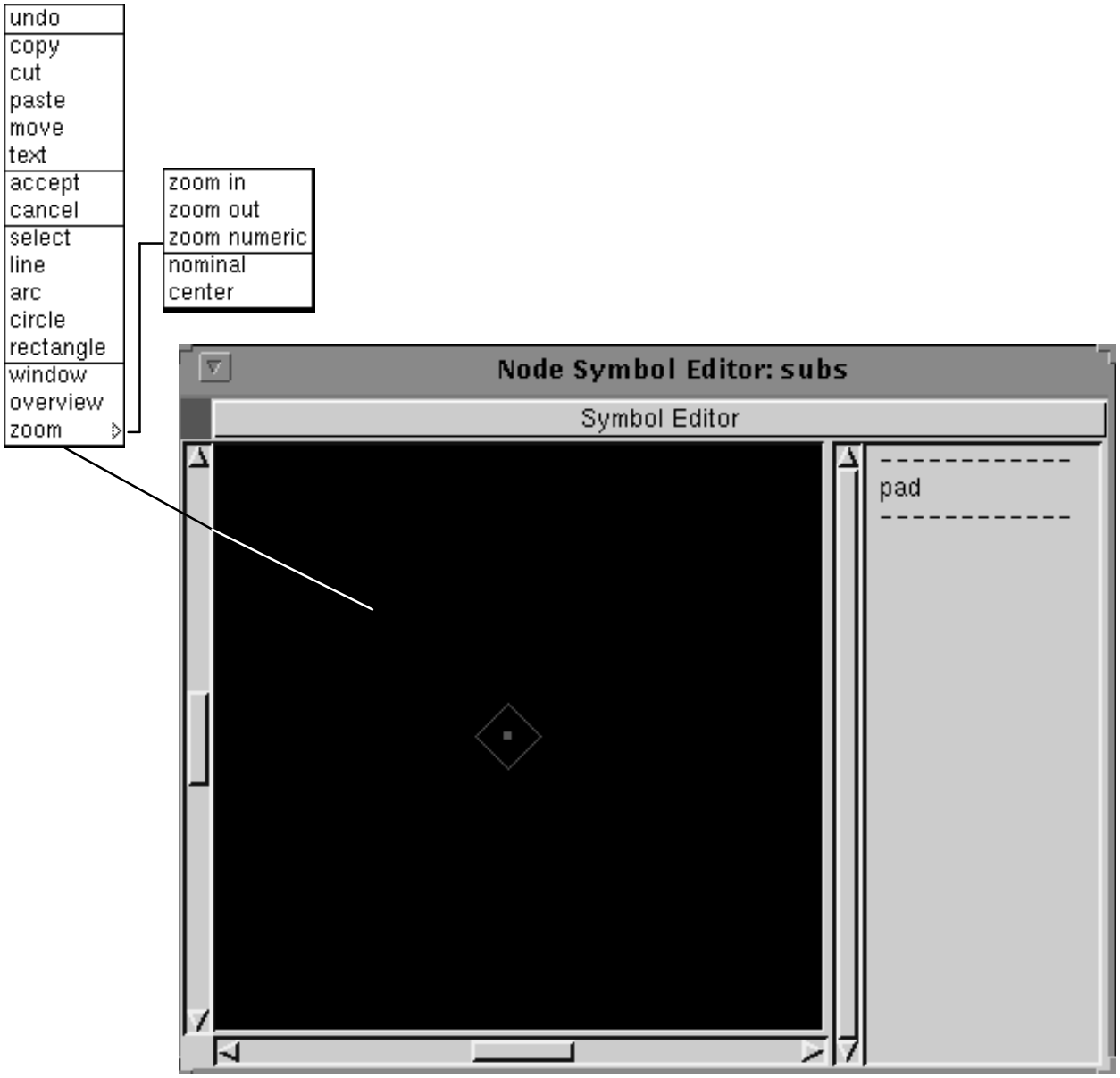
Node List Pane

The *Node List Pane* provides the names of local and global nodes used in the circuit. Schematic symbols representing these nodes are attached to wires or terminals in the circuit to establish the node (net) name. The local nodes listed are those only used in the schematic displayed in the *Schematic Editor Pane*.

Global nodes are listed in boldface print, local nodes listed in italics print. Global nodes are constant throughout the circuit hierarchy.

Schematic Editor Pane

The *Schematic Editor Pane* is the main pane for creating and modifying circuit schematics and viewing simulation results. The operation of this pane depends on the mode of the *Edit/Analysis/Annotation Pane*. Elements are added to the *Schematic Editor Pane* by selecting a model in the *Model List Pane* and using the pop–up button menu **paste**. When pasting elements in the *Schematic Editor Pane*, ADS replaces the arrow cursor with the symbol of the selected model. Before placement, a pop–up button menu is available for orienting the symbol. For copying, moving and deleting wires and elements use the **copy**, **cut**, **paste**, **move** and **adjust** commands.


undo
copy
cut
paste
move
text
accept
cancel
select
line
arc
circle
rectangle
window
overview
zoom
zoom in
zoom out
zoom numeric
nominal
center
Node Symbol Editor: subs
Symbol Editor
pad

Node Symbol Editor

The **Node Symbol Editor** allows modification of symbols used to represent nodes in a circuit schematic. Alternative node symbols can make schematics more readable. As an example, a small transistor substrate node symbol with an invisible label reduces the clutter on a schematic. Node symbols may be changed on a node by node basis, or for all nodes of a particular type (external, local or global).

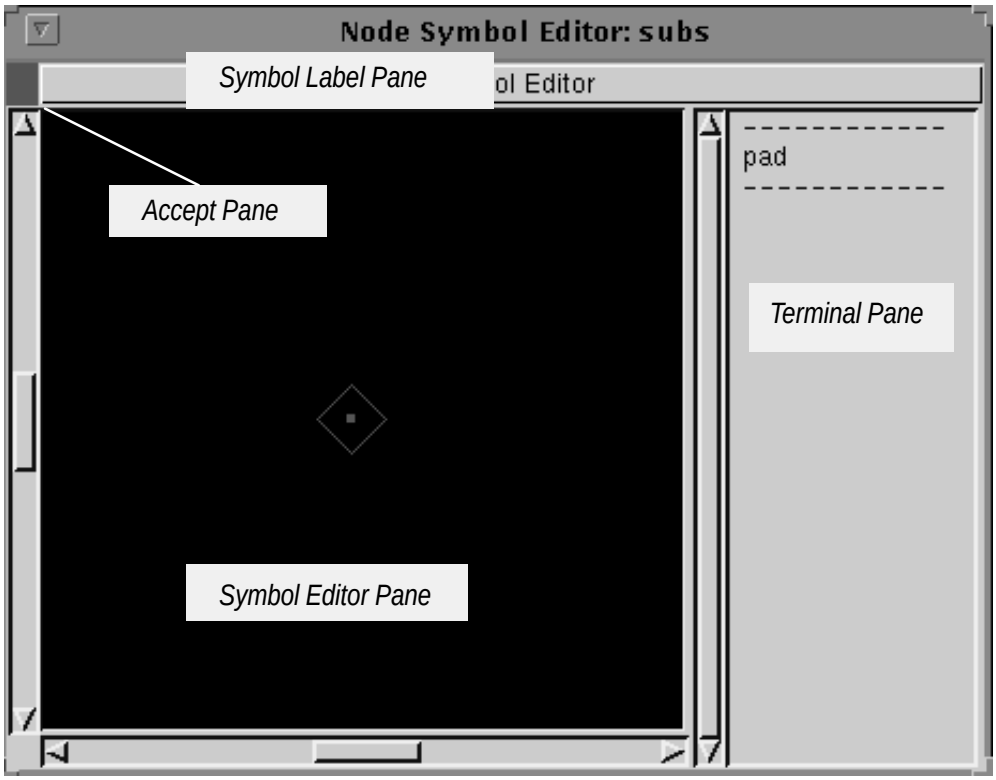


Figure 3–31: Node Symbol Editor Panes

Access

Access the **Node Symbol Editor** from the **Circuit Editor** or **Subcircuit Editor** with a node selected from the *Node List Pane* and then using the pop–up button **edit symbol** command.

Panes

Figure 3–31 shows the location of the *Symbol Editor Pane* and *Terminal List Pane* accessed from the **Library User Symbol Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Terminal Pane

The *Terminal Pane* lists the name of the terminal displayed in the *Symbol Editor Pane*.

User Symbol Editor

The User Symbol Editor allows the creation of custom element symbols that are usually modifications of existing library model symbols.

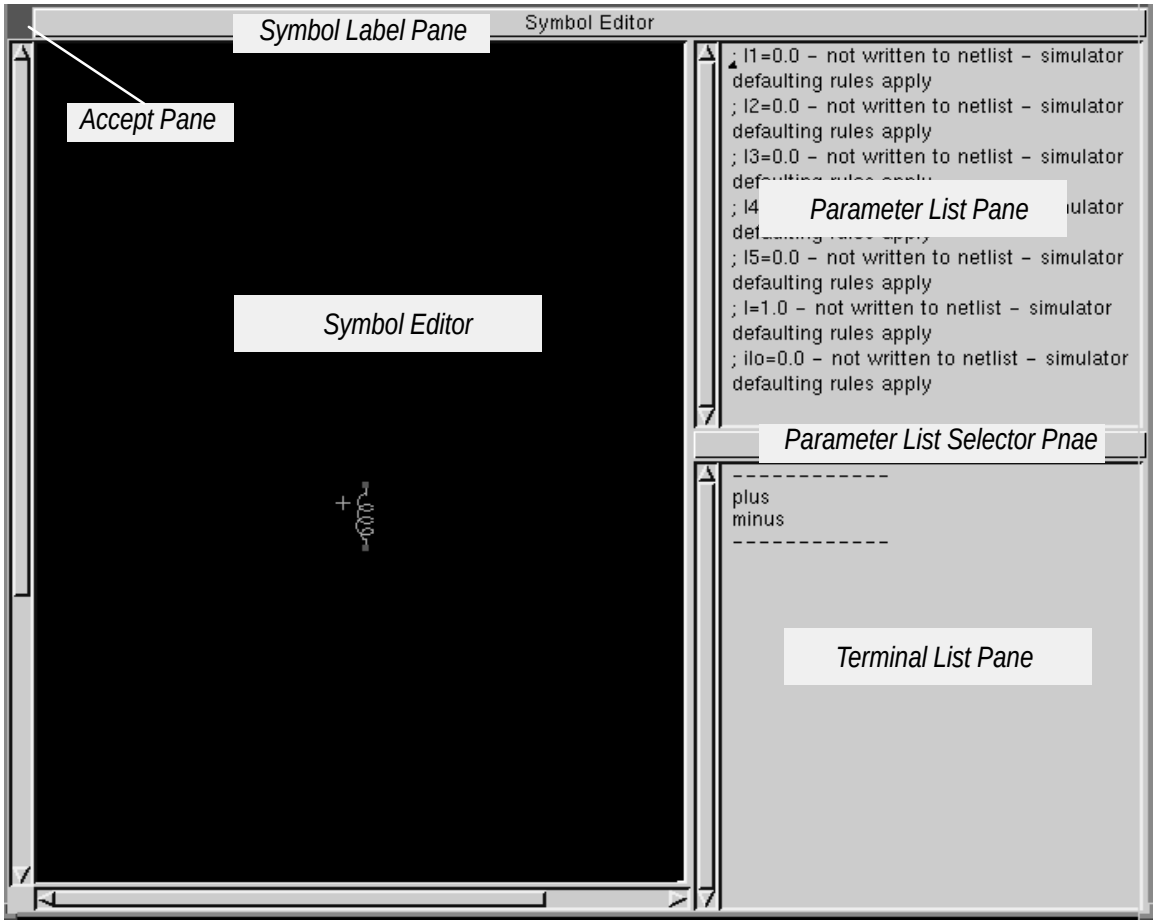


Figure 3–32: User Symbol Editor Panes

Access

Access the **User Symbol Editor** from the **Circuit Editor** or **Subcircuit Editor**'s *Schematic Editor Pane*. Highlight the library model name in the *Model List Pane* and then select the pop–up button **edit symbol** command.

Panes

Figure 3–32 the location of the panes accessed from the **User Symbol Editor**.

A description of the panes in this window follows:

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Parameter List Pane

The *Parameter List Pane* defines input parameters, assign values to them and defines output parameters. The displayed names are either input or output parameter names or local variable names depending on the mode set in the *Parameter List Selector Pane*. If a user symbol is being created, local variables may not be created and only value changes can be made for the input parameters.

The syntax for defining output parameters for subcircuit models is as follows:

output(element, parameter) is the value of the named **parameter** on the named **element**. Element name is relative to the definition of the subcircuit where the output function is used.

v(N) is the swept voltage on node N, where N is named relative to the definition of the subcircuit where **v()** is used.

bv(N) is the bias (operating point) voltage on node N where N is named relative to the definition of the subcircuit where the **bv()** is used.

i(element,terminal number) is the swept current through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where **i()** is used.

bi(element,terminal number) is the bias (operating point) cuirrent through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where the **bi()** is used.

The syntax for primitive and user models is as follows (where T is an integer greater than 0 and less than or equal the terminal count on the model):

primitivebv(T) is the operating point voltage at terminal T.

primitivebi(T) is the operating point current into terminal T.

primitivev(T) is the swept voltage at terminal number T of the given element.

primitiveoutput(P) is the swept value of parameter P defined in the primitive model for the given element from the swept analysis (Ac, dc, or Tran). If not defined here then it is undefined.

primitiveparameter(P) is the value of parameter P defined in the primitive model for the given element from the initial (bias, dc, ac or tran) analysis. If not defined here then it is from the 'tempAdjust' outputs. If not defined here then it is from the initial parameter values. If not defined here then it is undefined.

Parameter List Selector Pane

The *Parameter List Selector Pane* allows setting the operating mode of the *Parameter List Pane*. Setting the mode to **input parameters** causes the parameter names and value in the *Parameter List Pane* to represent subcircuit input parameters. Setting the mode to **output parameters** causes the parameter names and values in the *Parameter List Pane* to represent subcircuit output parameters. Setting the mode to **local variables** causes the parameter names and values in the *Parameter List Pane* to represent expressions and assignments used to set subcircuit element values. The **local variables** mode is not available from the **Library User Symbol Editor**.

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Terminal List Pane

The *Terminal List Pane* supports access to the circuit and subcircuit connection terminals (ports). These terminals define the connections from the local circuit (the one in the *Schematic Editor Pane*) to the next level of circuit hierarchy. Terminals from this list must be pasted into the *Symbol Editor Pane* by selecting the terminal in the *Terminal List Pane* and then using the pop–up button command **paste terminal**, paste the terminal in the *Symbol Editor Pane*. If the terminal already exists then the **paste terminal** command does not appear in the pop–up button menu.

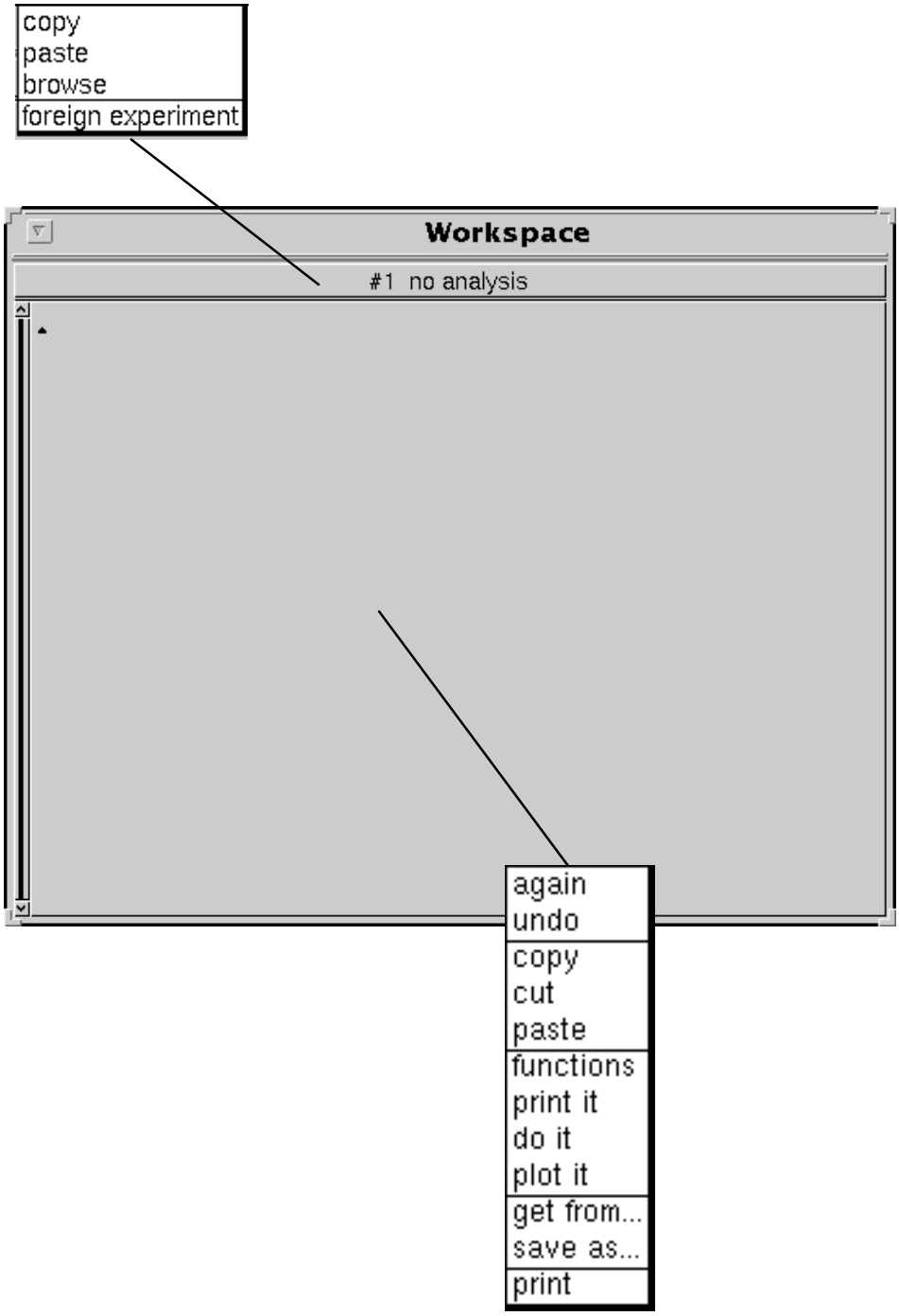


Figure 3–33: Workspace Editor Commands

Workspace Editor

The **Workspace Editor** evaluates TekSpice expressions or functions as a scientific calculator. Numeric constants are allowed. Local variables can be defined and used for subsequent calculations. The **Workspace Editor** can view results from an existing TekSpice file (external to ADS) and compare results with a simulation from ADS.

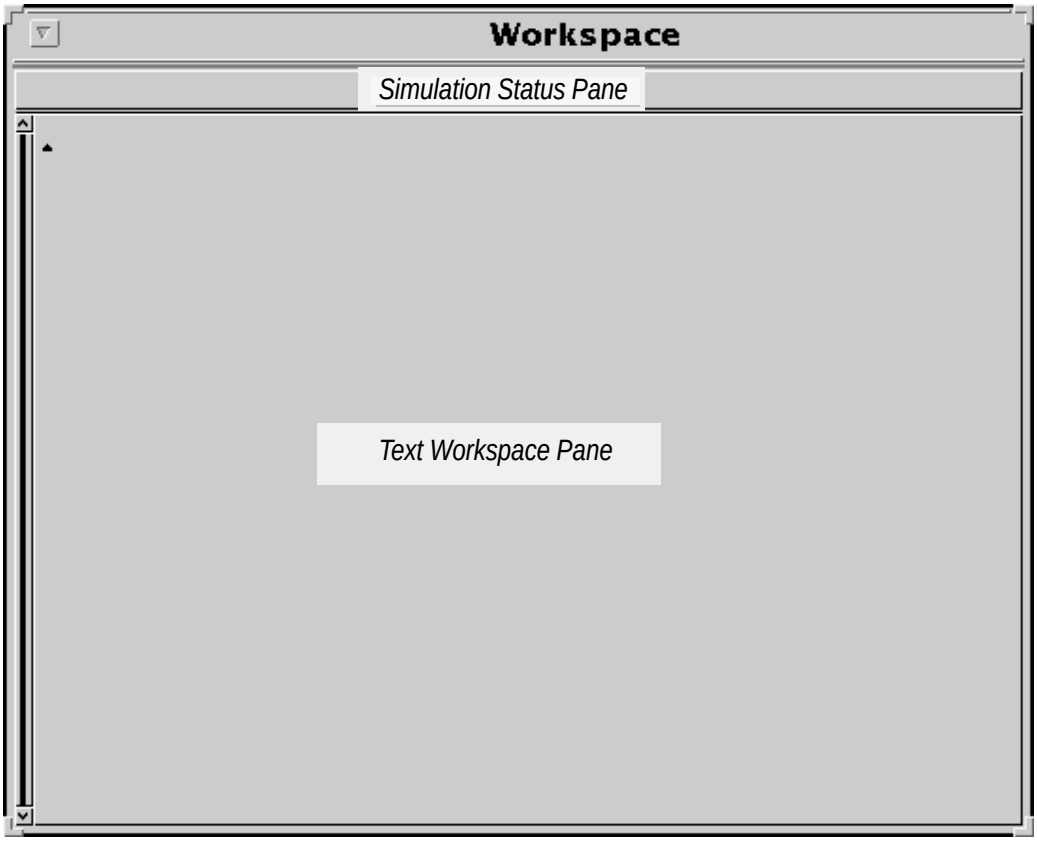


Figure 3–34: Workspace Editor Panes

Access

Access the **Workspace Editor** from the **Launcher** using the **open** > **workspace** command.

Panes

There are two panes associated with the **Workspace Editor**, the *Simulation Status Pane* and *Text Workspace Pane*. Figure 3–34 shows the location of these panes.

A description of the panes in this window follows:

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

Dialog Boxes

This reference describes the functionality of the windows within the ADS system. Although technically they are all windows, for the purposes of this manual they are divided into Launchers, Browsers, Editors and Dialog Boxes.

Dialog boxes prompt the user for information needed to complete commands. For example the **Print Options** dialog box causes ADS to prepare schematics and text for output to a printer. The purpose of this dialog box is to obtain size, orientation and file names for the resulting output. ADS automatically opens and sizes dialog boxes. After the user supplies the requested information and presses the continue button, the box is closed. It is possible to move the dialog box by dragging it to the desired location with the title bar.

A Dialog Box contains a title, a set of named panels and a row of buttons, usually **quit**, **ok** or **continue**. Quit means terminate the command that initiates the dialog box. Ok means accept the information supplied in the dialog box, close the box and continue with the command that initiates the dialog.

This section describes the Dialog Boxes alphabetically by name. Each dialog box description contains the following information:

1. A brief description of the dialog box.
2. A figure showing a snapshot of the dialog box.
3. The path to access the dialog box.
4. A brief description of the options available from the dialog box.

Listed below are the Dialog Boxes described in this section (in alphabetical order).

- Add External Library Dialog Box
- Add Internal Library Dialog Box
- Automatic Save Configuration Dialog Box
- Bug Report Dialog Box
- Change External Library Dialog Box
- Change Internal Library Dialog Box
- Color Configuration Dialog Box
- Complex Variable Configuration Dialog Box
- EDIF Dialog Boxes
- Element Browser Dialog Box
- Emergency Checker Dialog Box

- Encapsulated Edif Dialog Box (write only)
- Image Reduction Dialog Box
- Label Style Dialog Box
- Layer Visibility Dialog Box
- Netlist to Layout Analysis Dialog Box
- Parasitic Configuration Dialog Box
- Plotter Configuration Dialog Box
- Print Options Dialog Boxes
- Printer Scripts Configuration Dialog Boxes
- Prune Results Dialog Box
- Simulation Status Dialog Box
- Simulator Configuration Dialog Box
- System Configuration Dialog Box
- System Consistency Dialog Box
- System Installation Checker Dialog Box
- System Transcript Dialog Box
- User Interface Dialog Box

Add External Library Dialog Box

The **Add External Library** dialog box allows the user to add an external library to their ADS image.

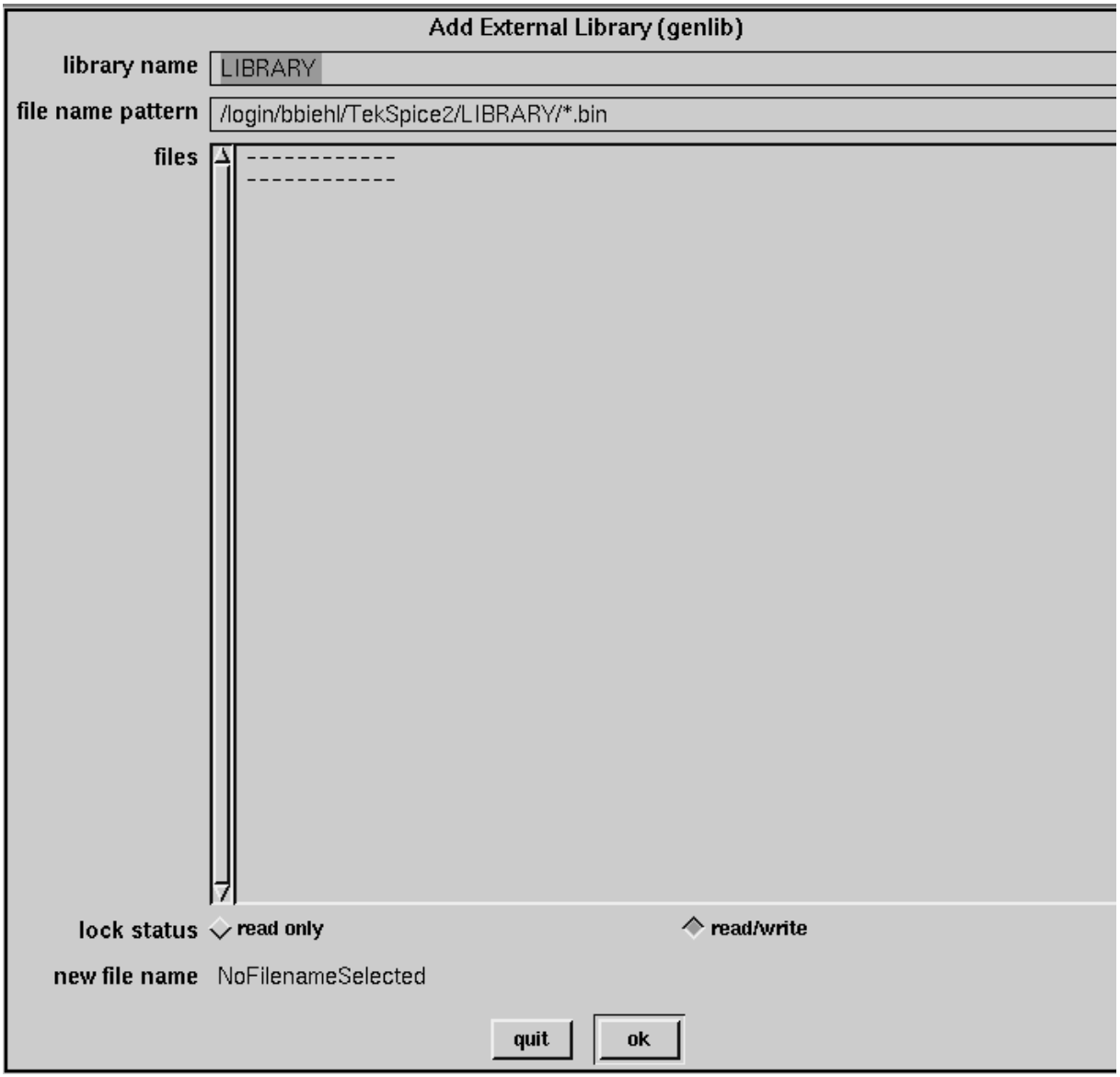


Figure 3–35: Add External Library Dialog Box

Access

Access the **Add External Library** dialog box from the **Library Browser** *library list pane* using the **add external library** command.

Options

library name. This text window allows specifying the name of the library.

file name pattern. This text window allows specifying the path to the file containing the library.

files. This text window displays the files specified by the **file name pattern**.

lock status. The library is (**read only**) or (**read/write**).

If a library is marked **read only**, then ADS prevents modification of the library. A library marked as **read only** will not be read into an image that already has that library loaded.

Add Internal Library Dialog Box

The **Add Internal Library** dialog box allows the user to add an internal library to their ADS image.

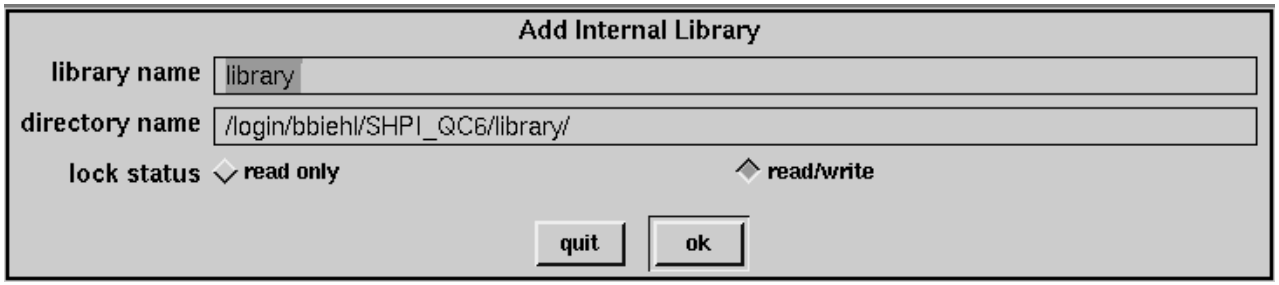


Figure 3–36: Add Internal Library Dialog Box

Access

Access the **Add Internal Library** dialog box from the **Library Browser** library list pane using the **add internal library** command.

Options

library name. This text window allows specifying the name of the library (lower case is used by convention).

directory name. This text window allows specifying the path to the directory containing the EDIF files for the library.

lock status. The library is (**read only**) or (**read/write**).

If a library is marked **read only**, then ADS prevents modification of the library. A library marked as **read only** will not be read into an image that already has that library loaded.

(This page intentionally left blank)

Automatic Save Configuration Dialog Box

The **Automatic Save Configuration** dialog box allows the user to determine if the automatic save feature is activated and if activated, set the time interval between automatic saves.

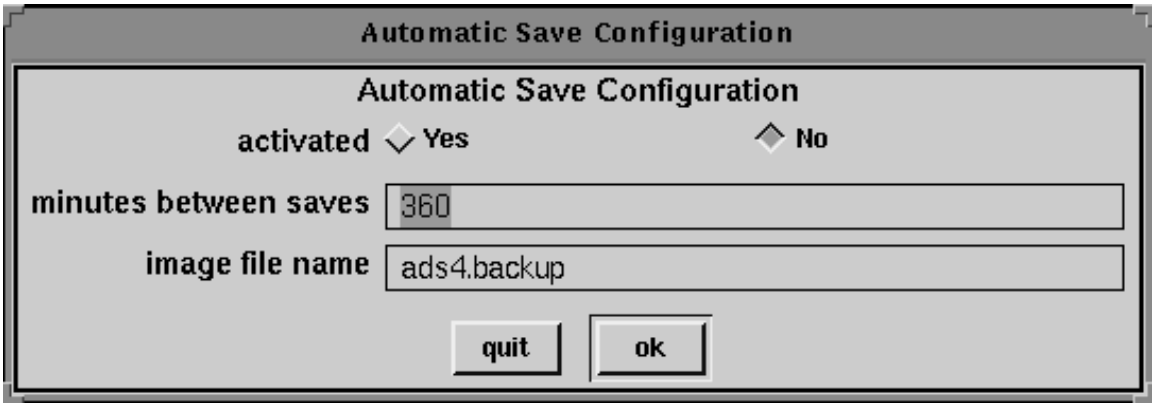


Figure 3–37: Automatic Save Configuration Dialog Box

Access

Access the **Automatic Save Configuration** dialog box from the **Launcher** using the **Configure > automatic save** command.

Options

activated. The automatic save image feature is activated (**yes**) or not activated (**no**).

minutes between saves. Specifies the minutes between automatic saves. If the simulator is running at the specified save time, a warning message appears and the save skipped.

image file name. Specifies the name of the saved image file.

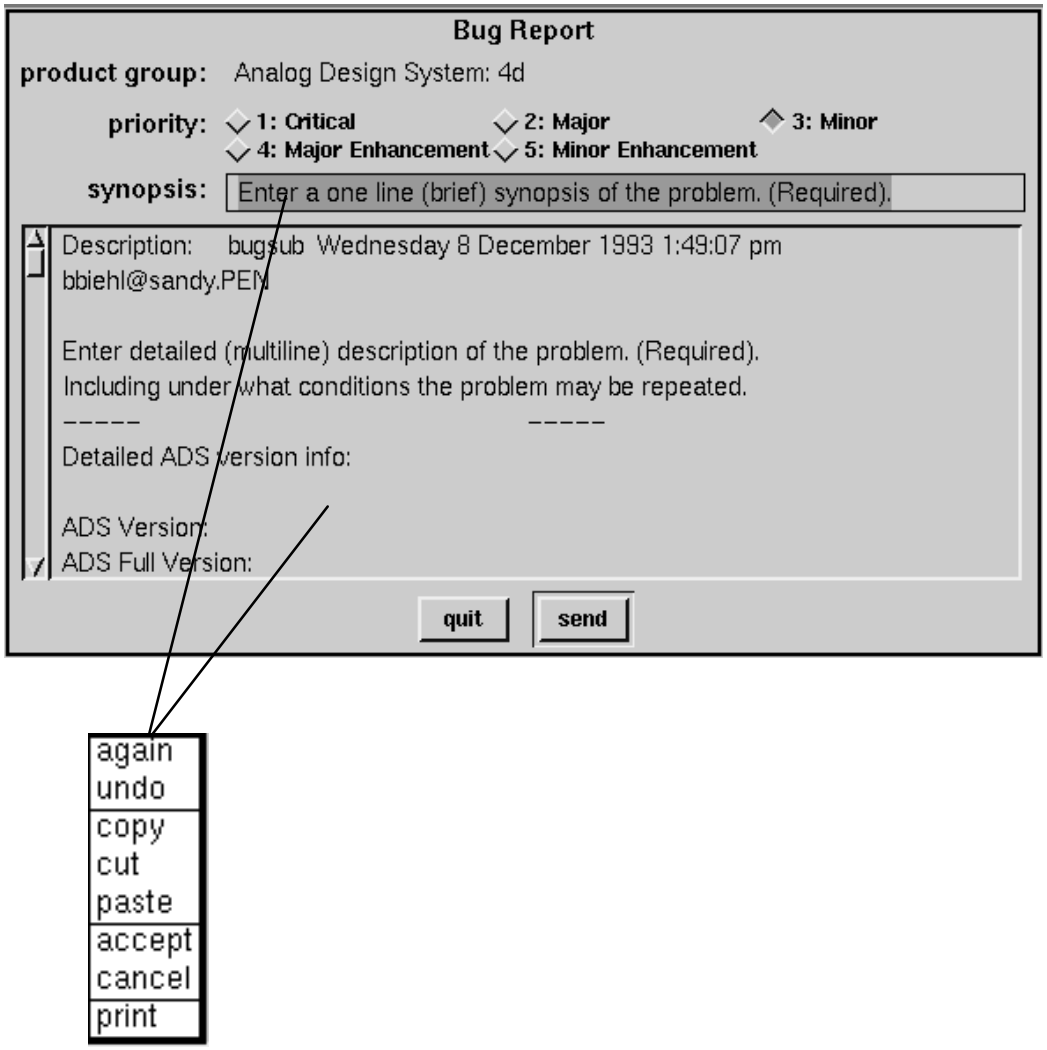


Figure 3–38: Bug Report Dialog Box Commands

Bug Report Dialog Box

The **Bug Report** Dialog Box allows submitting bugs and program enhancement requests. Figure 3–39 shows the Bug Report Dialog Box.

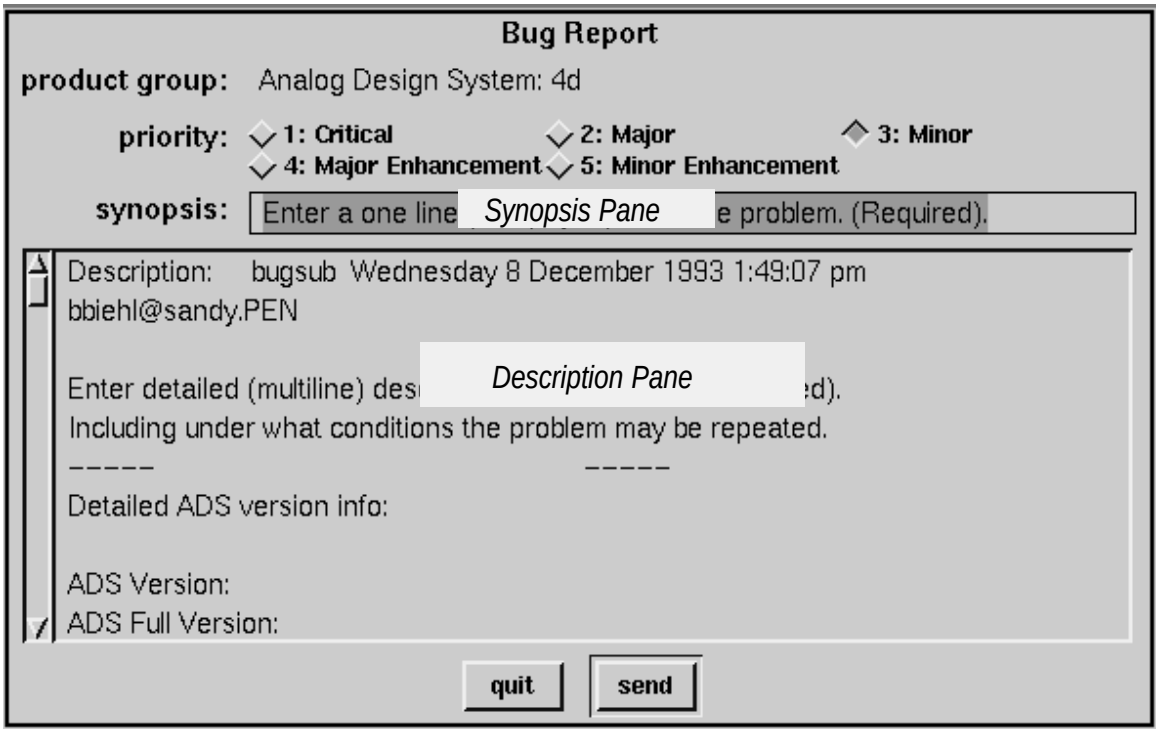


Figure 3–39: Bug Report Dialog Box Panes

Access

Access the **Bug Report** dialog box from; 1) the **Launcher** using the **Open > bug report** command; 2) from the **bug report** command on an **Error Notification** dialog box. In this case the **Error Notification** dialog box automatically appears. Submit a bug report using the pop–up button **bug report** command (See Figure 3–40). The **Bug Report** dialog box is also available from several other dialog boxes by pressing the bug report button. (See Figure 3–41.)

Panes

There are two panes in the **Bug Report** dialog box, the *Synopsis Pane* and *Description Pane*. Figure 3–39 shows the location of these panes.

Bug Priority

Indicate the priority of a bug submission based on the following criteria:

1. **Critical Bug** — the problem prevents ADS from running.
2. **Major Bug** — ADS is not working properly but work can continue.

3. **Minor Bug** — the problem has minimal effect on the use of ADS. Cosmetic problems and minor inconsistencies fall into this category.
4. **Major Enhancement** — there is a strong desire for this feature or capability.
5. **Minor Enhancement** — the new feature is desirable, but not as valuable as a Major Enhancement.

Bug Summary

Enter a one line summary of the problem or enhancement request in the *Synopsis Pane*. Make this description as brief and clear as possible. The normal text editing commands are accessible from the pop–up button.

Bug Description

Enter a full description of the bug, problem or request in the *Description Pane.* The normal text editing commands are accessible from the pop–up button.

Submitting Bug

Submit the report by selecting the “send” button at the bottom of the dialog box.

Example Errors

Figure 3–40 shows the resulting dialog box that appears when an unanticipated or more serious system error occurs. Figure 3–41 shows the resulting dialog box that appears when a system error occurs.

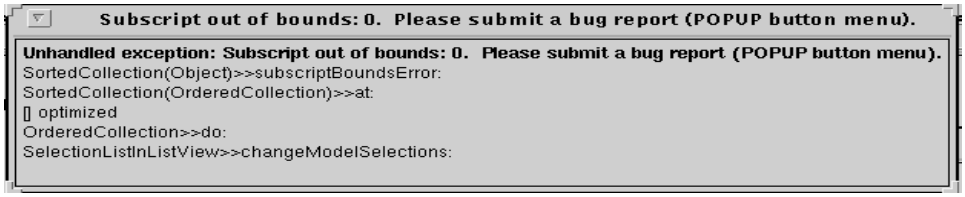


Figure 3–40: Error Notification Dialog Box

Figure 3–41: Example Error Notification Dialog Box

Change External Library Dialog Box

The **Change External Library** dialog box allows the user to rename, set the genlib file, and set the lock status of an external library. The EDIF files for the library are assumed to be in the same directory as the genlib flie.

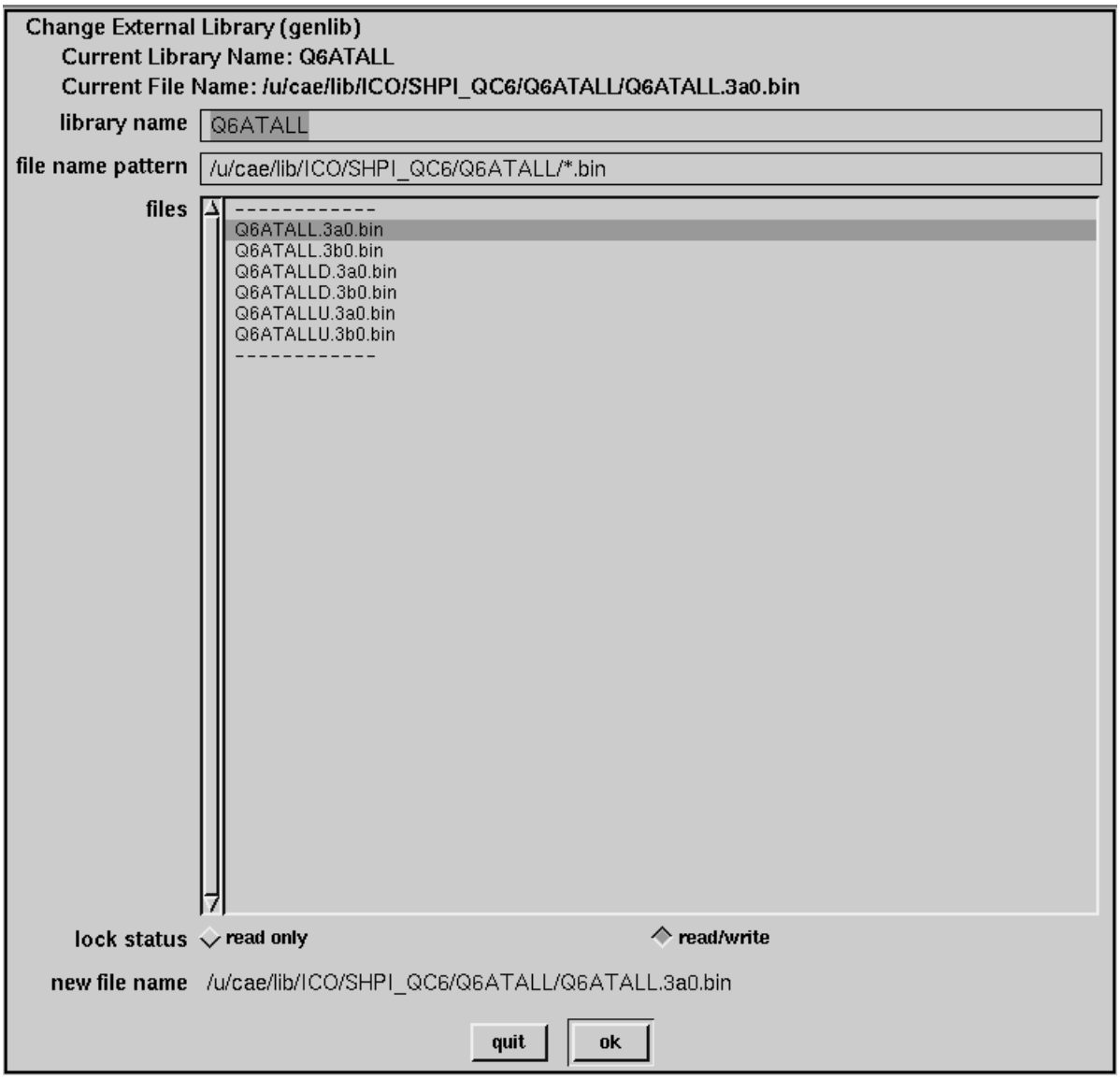


Figure 3–42: Change External Library Dialog Box

Access

Access the **Change External Library** dialog box from the **Library Browser** *library list pane* using the **change library** command.

Options

library name. The name of the external library.

file name pattern. Specifies the directory path assumed for EDIF reading and writing of models in the library as well as the directory for the genlib file.

files. Specifies the genlib file associated with this external library.

lock status. The library is (**read only**) or (**read/write**).

If a library is marked **read only**, then ADS prevents modification of the library. A library marked as **read only** will not be read into an image that already has that library loaded.

The current library name and current file name of the external library are displayed in the title at the top of the dialog box. The new file name (based on the selection from the files option) is displayed at the bottom of the dialog box.

Change Internal Library Dialog Box

The **Change Internal Library** dialog box allows the user to rename, set the default EDIF read and write directory, and set the lock status of an internal library.

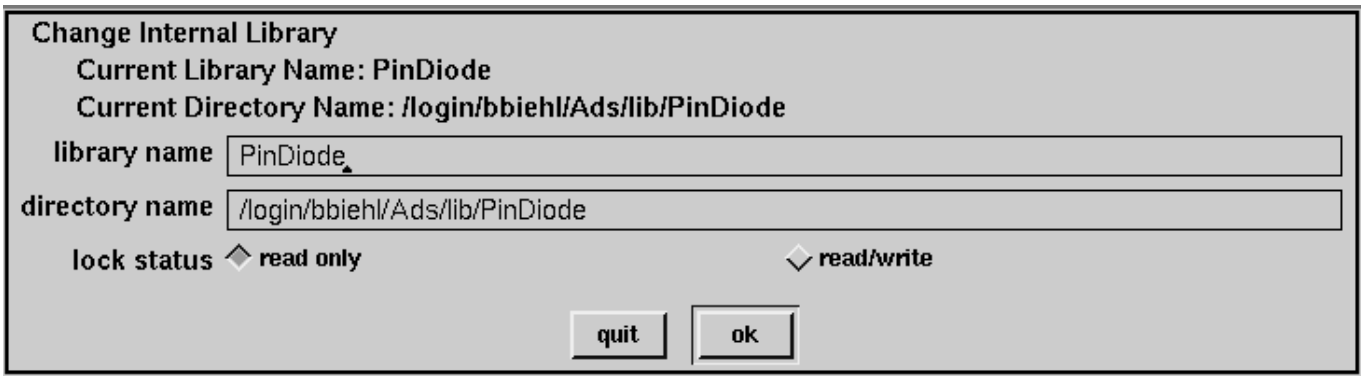


Figure 3–43: Change Internal Library Dialog Box

Access

Access the **Change Internal Library** dialog box from the **Library Browser** *library list pane* using the **change library** command.

Options

library name. This text window allows specifying the name of the library.

directory name. This text window allows specifying the path to the default EDIF read/write directory containing the library.

lock status. The library is (**read only**) or (**read/write**).

If a library is marked **read only**, then ADS prevents modification of the library. A library marked as **read only** will not be read into an image that already has that library loaded.

The current library name and current file name of the external library are displayed in the title at the top of the dialog box. The new file name (based on the selection from the files option) is displayed at the bottom of the dialog box.

(This page intentionally left blank)

Color Configuration Dialog Box

The **Color Configuration** dialog box allows the user to select graticule (grid) colors for:

Schematic Editor Pane of the **Circuit Editor** and **Subcircuit Editor**.

Waveform Pane of the **Plot Browser.**

and the text color for:

The **schematic scratch** layer of the **Circuit Editor**.

The **schematic sheet1** layer of the **Circuit Editor**.

The **schematic sheet2** layer of the **Circuit Editor**.

The **annotation layers** of the **Circuit Editor.**

Figure 3–44 shows the **Color Configuration** dialog box.

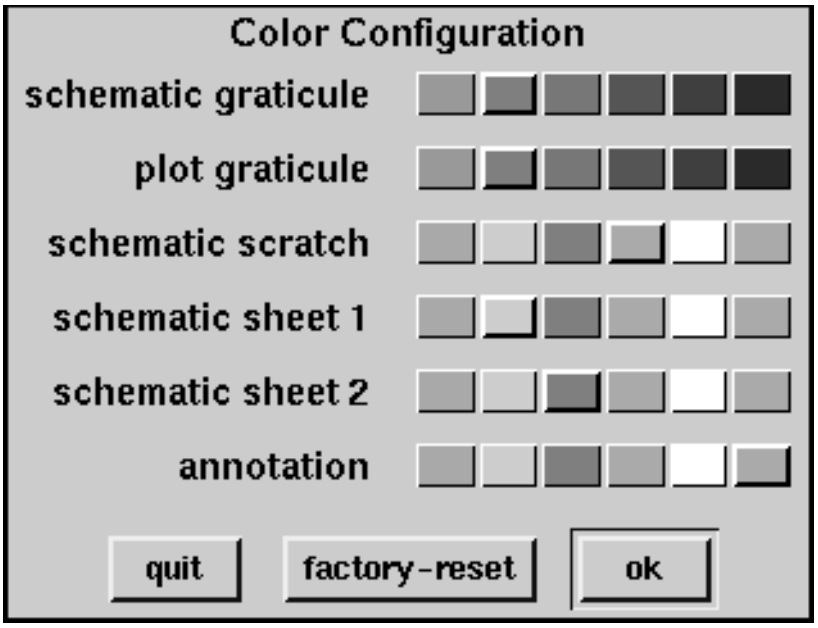


Figure 3–44: Color Configuration Dialog Box

Access Access the **Color Configuration** dialog box from the **Launcher** using the **Configure > color** command.

Options **factory–reset.** restore the colors to the original default settings.

(This page intentionally left blank)

Complex Variable Configuration Dialog Box

The **Complex Variable Configuration** dialog box affects how ADS calculates simulation results from an ac frequency response analysis and displays the resulting plots. Figure 3–45 shows the **Complex Variable Configuration** dialog box.

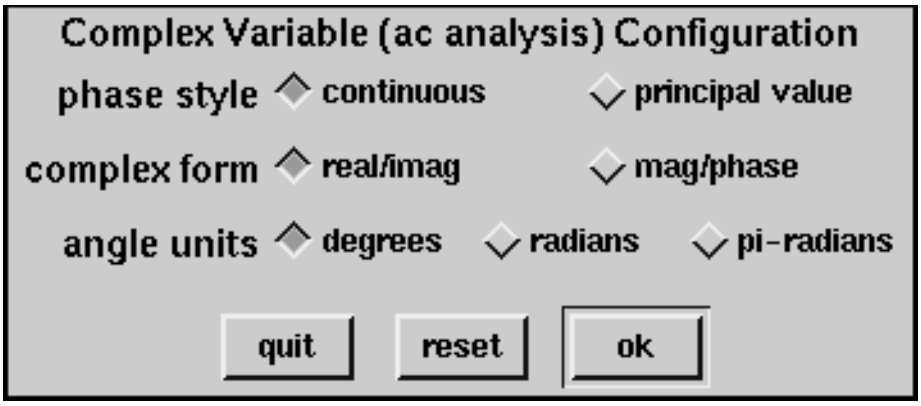


Figure 3–45: Complex Variable Configuration Dialog Box

Access

Access the **Complex Variable Configuration** dialog box from the **Launcher** dialog box using the commands **Configure > complex variables**.

Options

phase style. continuous plot or principal value (phase between –180 and +180 degrees).

complex form. The plot is two waveforms consisting of real and imaginary(**real/imag**) values or magnitude and phase values (**mag/phase**).

angle units. (for **mag/phase** only). Phase angle is in units of **degrees, radians** or **pi–radians**.

reset. Restore the settings to the values when the dialog box was first opened.

(This page intentionally left blank)

EDIF Dialog Boxes

ADS uses two methods of archiving and exchanging circuit definitions; Electronic Data Interchange Format (EDIF) and ADS images. The EDIF is the recommended and most compact method.

ENCAPSULATED EDIF WRITE

Figure 3–46 shows the **Encapsulated EDIF Dialog Box**. This is the EDIF Dialog Box used for writing EDIF circuit files. The circuits written to the EDIF file depend on the options highlighted in the **Circuit Browser**. Selecting a project causes the writing of all circuits within that project. Selecting a single circuit causes the writing of just that circuit to the file.

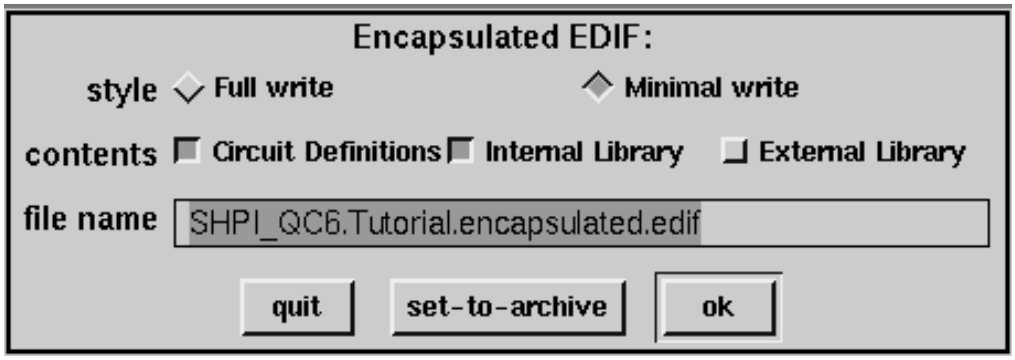


Figure 3–46: Dialog Box for Writing Encapsulated EDIF Files

Access

Access the **Encapsulated EDIF** dialog box for writing circuit files from the **Circuit Browser** with a project and/or circuit highlighted. Then select the pop–up button menu **write edif** command.

Options

The options allow for writing different types of encapsulated EDIF files. The resulting files can be used for sharing, updating, or archiving a design. When these file are restored to an ADS image (**read edif** command), the circuits and libraries are restored intact.

style. Full write causes all models, even those that are unused, to be incorporated into the EDIF file. Complete libraries named in the **Library Path Editor** are also written to the EDIF file (depending on the selection of the internal or external libraries options).

Minimal write causes only the models which are actually used in the circuit to be written to the EDIF file. This is appropriate for updating a circuit for someone who has already received a **Full write** EDIF.

contents. Select one or more options:

Circuit Definitions causes the writing of circuit definitions.

Internal Library causes the writing of internal libraries.

External Library causes the writing of external libraries.

file name. The file name that contains the resulting EDIF.

set–to–archive. Automatically selects the **contents** options: **Full write**, **Circuit Definitions**, **Internal Library** and **External Library** and sets a default file name as the concatenated project name, circuit name and the word ".encapsulated.archive.edif". This is appropriate for a complete archive of a project which may be restarted at a later date.

These options allow tailoring the EDIF file for such purposes as sharing, updating, or archiving a design, and for including or omitting standard libraries or unused portions of libraries. The following are suggestions for choosing among the options:

1. If you are sharing a circuit with another designer for the first time select; **Full write**, **Circuit Definitions**, and **Internal Libraries**. If you are sharing with someone who does not have installed the external libraries you are using, also select the **External Libraries**.
2. If you are preparing to update your circuit for someone who earlier received the EDIF from item 1 above select; **Minimal write**, **Circuit Definitions** and **Internal Libraries**.
3. If you are shelving a project for later use select; **Full write**, **Circuit Definitions**, **Internal Libraries** and **External Libraries**.

EDIF WRITE

Figure 3–47 shows the dialog box used when writing an EDIF file of a Library.

Figure 3–47: Write EDIF Library Dialog Box

Access

Access the EDIF dialog box for writing libraries from the **Library Browser** with a Library selected using the pop–up button **write edif** command.

Each model in the selected library is written to an EDIF file in the specified directory.

ADS IMAGE EDIF WRITE

Figure 3–48 shows the dialog box used for writing an EDIF file for backing up an entire ADS image.

Figure 3–48: EDIF Write Dialog Box for Backing up an ADS Image

Access

Access the dialog box in Figure 3–48 from the **Launcher** using the **System > backup to edif** command.

EDIF READ

When reading the encapsulated EDIF file, ADS automatically installs the circuit or circuits, model symbols and parameter definitions, and the GENLIB files for external libraries. It also defines the libraries and the GENLIB path in the **Library Browser**. File names are modified as necessary to avoid overwriting existing libraries. The **EDIF Dialog Boxes** shown in Figure 3–49 and Figure 3–50 are two of the EDIF dialog boxes used for reading EDIF files.

The **read edif** command preserves existing circuits and libraries in the event of naming conflicts. Existing circuits with the same name as in the EDIF file have an ”X” appended to them indicating they are old versions of the circuit. Libraries read in with the **read edif** command have the string ”encapsulated” inserted in the library name. These names will be displayed in the **Library Browse**r. If the appended name is still not unique, a number is appended to the incoming library. These numbers start with zero and increment by one. Duplicate names are probably due to multiple reads of the same encapsulated EDIF file.

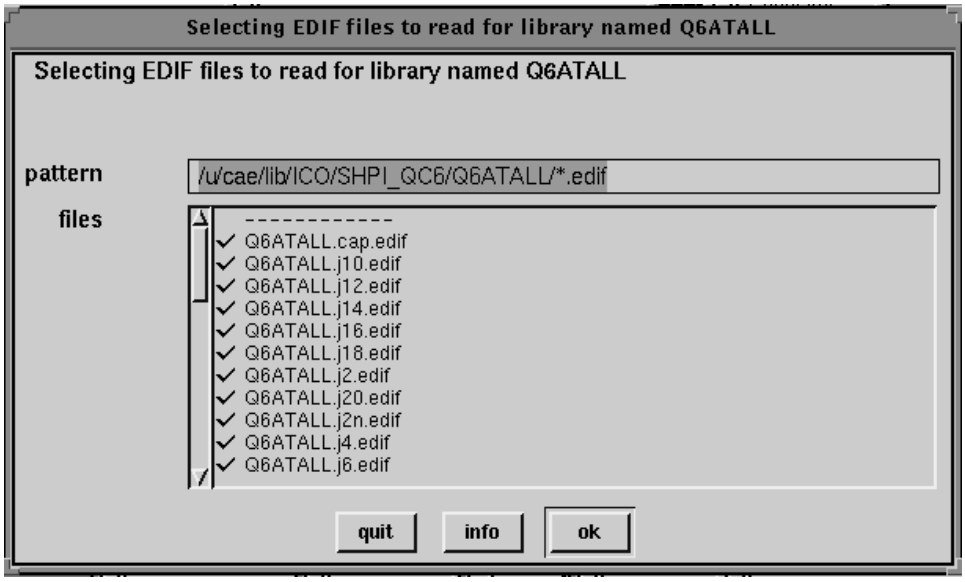


Figure 3–49: Dialog Box for Reading Library EDIF Files

Access

The EDIF Dialog Box for reading Library EDIF files is accessed from the **Library Browser**’s *Library List Pane* with a library selected. Then use the pop–up button command **read edif**.

The EDIF Dialog Box for reading EDIF Library Model files is accessed from the **Library Browser**’s *Model List Pan*e with nothing selected and use the **read edif** pop–up button command.

Figure 3–50: Dialog Box for Reading Circuit and Project EDIF Files

Access

The EDIF dialog box for reading circuits is accessed from the **Circuit Browser**'s *Project List Pane* with a project selected or *Circuit List Pane* with no circuit selected and use the **read edif** pop–up button command.

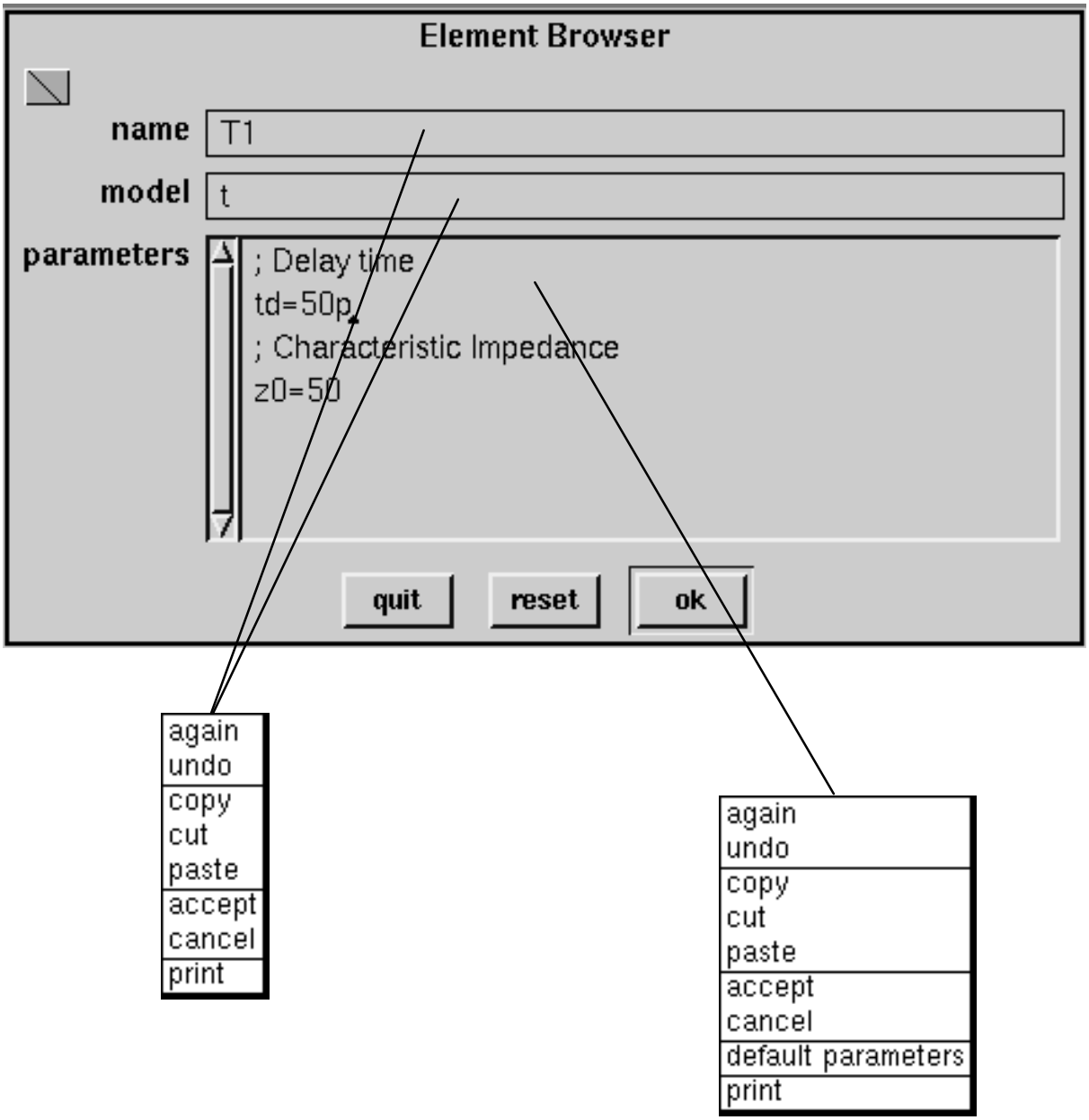


Figure 3–51: Element Browser Dialog Box Commands

Element Browser Dialog Box

The **Element Browser** dialog box allows setting model input parameter values for individual circuit elements on the schematic. ADS uses the default values if they are not set.

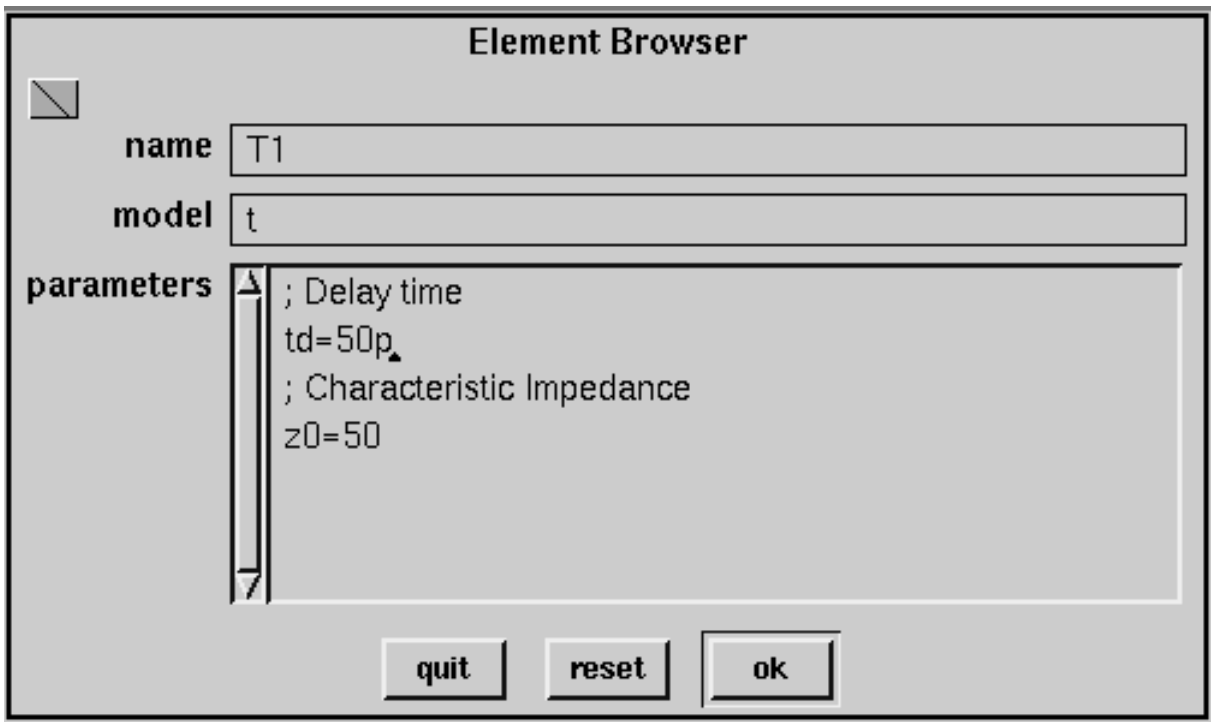


Figure 3–52: Element Browser Dialog Box

Access

Access the **Element Browser** dialog box from the **Circuit Browser's** *Schematic Editor Pane* by highlighting a device (element) and using the pop–up button **brows** command.

Options

The options for the **Element Browser** dialog box are as follows:

name. The name of the highlighted device (element).

model. The name of the library model used for the highlighted device.

(parameter list). Shows the element parameter values used to create the netlist for the TekSpice simulation. List all default parameters by using the **default**

parameters pop–up button command. Parameter values can be changed on an element by element basis.

reset. Sets the parameters, name and model to the opening values.

Emergency Checker Dialog Box

The **Emergency Checke**r dialog box is normally not required. It is for use only by support personnel.

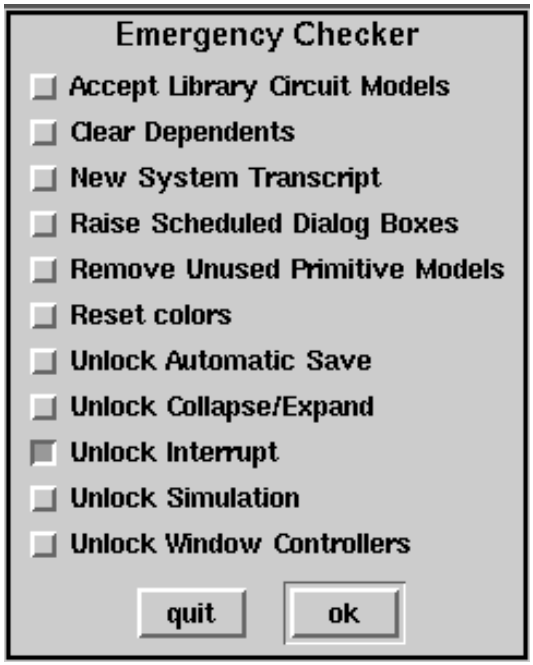


Figure 3–53: Emergency Checker Dialog Box

Access

Access the **Emergency Checker** dialog box from the **Launcher** using the **System** > **checker** > **emergency** commands.

Options

The options for the **Emergency Checker** dialog box are as follows:

Accept Library Circuit Models. Accepts all library circuit models. Only use this option if requested by ADS support personnel.

Clear Dependents. Clears all dependents and closes all ADS windows. Only use this option if requested by ADS support personnel.

New System Transcript. Closes any existing **System Transcript Browser**s and opens a new **System Transcript Browser**. Only use this option if requested by ADS support personnel.

Raise Scheduled Dialog Boxes. Causes dialog box windows that are burried under other windows to raise to the front of these windows.

Remove Unused Primitive Models. Removes unused primitive models from every subcircuit and closes all schematic windows.

Reset Colors. Resets the ADS color map. This is done after quitting another application which uses colors that conflict with those in ADS. If colors are still a problem; quit another color intensive application and use this command again.

Unlock Automatic Save. Unlocks the **Automatic Save** feature if ADS incorrectly thinks there is a process running that prevents the **Automatic Save** from working. The **Raise Scheduled Dialog Boxes** command in this menu should be run first.

Unlock Collapse/Expand. Unlocks collapse/expand when ADS refuses to let you collapse or expand ADS. That is if ADS complains about waiting to schedule a window. You waited, tried again and ADS still refused.

Unlock Interrupt. Unlocks interrupts when typing the interrupt key sequence in an ADS window. ADS wiggles the cursor and rings the keyboard bell, but a window does not open.

Unlock Simulation. Unlocks simulations. Only execute this command when no simulation are running from ADS. Be sure to close all dialog boxes. Continue only if you know a simulation is not running, but ADS thinks a simulation is running.

Unlock Window Controllers. Unlocks all window controllers and closes all dialog boxes. Use this command when a window will not respond to mouse button events.

Image Reduction Dialog Box

The **Image Reduction** dialog box allows reduction of the ADS image size by removing unused or obsolete data from the image.

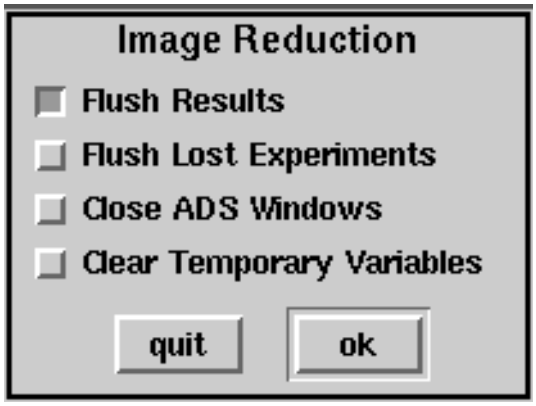


Figure 3–54: Image Reduction Dialog Box

Access

Access the **Image Reduction** dialog box from the **Launcher** using the **System > pruning > image size reduction** command.

Options

The optional settings for the **Image Reduction** dialog box are as follows:

Flush Results. Removes the stored binary results file headers from the ADS image. These stored headers make subsequent results file reading faster. Results are not removed. The file headers are re–read if needed.

Flush Lost Experiments. Removes any simulation experiment files that do not have connection to existing circuits.

Close ADS Windows. Closes and discards all open ADS editors, browsers and dialog boxes leaving only the **Launcher** and **System Transcript** dialog box. Any work in progress should be ”accepted”, otherwise it will be lost**.**

Clear Temporary Variables. Clears temporary variables in all control programs. This command has the same effect as inserting the text ”clear” in all control programs.

(This page intentionally left blank)

Label Style Dialog Box

The **Label Style** dialog box allows changing the appearance and content of device labels on specified devices within a circuit schematic.

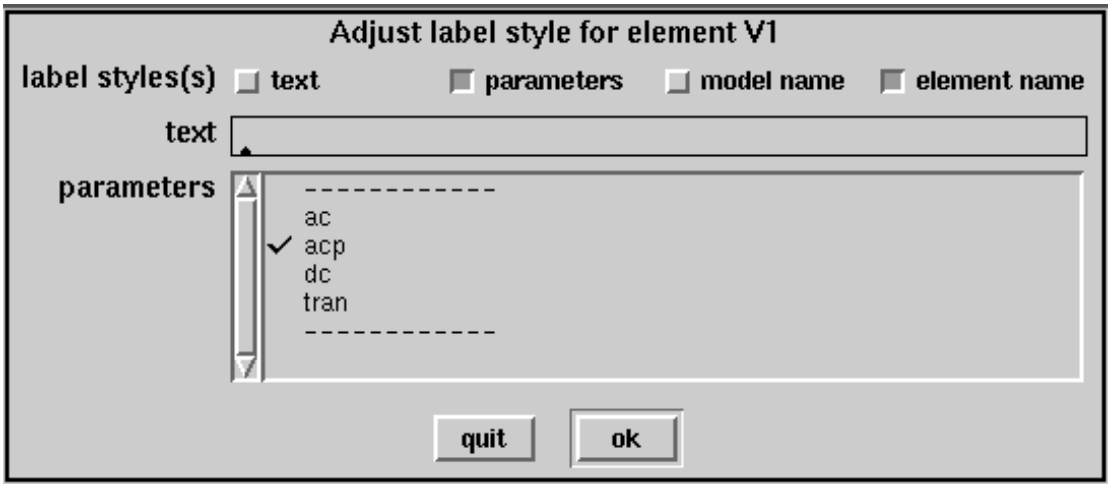


Figure 3–55: Label Style Dialog Box

Access

Access the **Label Style** dialog box from the **Circuit Editor** (edit mode) using the pop–up button menu **label style** command while over a device or subcircuit.

Options

The **Label Style(s)** dialog boxes allow selection of one or more of the following options:

text. Display the text in the text edit box with the device.

parameters. Display the model parameter(s) checked in the text window with the device. One or more of the parameters can be checked

model name. Display the model name with the device.

element name. Display the element name when checked. Default is checked.

(This page intentionally left blank)

Netlist to Layout Analysis Dialog Box

The **Netlist to Layout Analysis** dialog box allows analyzing the netlist created in the **Circuit Editor** for potential problems when laying out an IC. It also writes a netlist used for a netlist driven layout editor such as **QuicKic**. The application software used for the netlist–to–layout checking are programs such as **simtoq8** (See **Other References** in the *Preface*.)

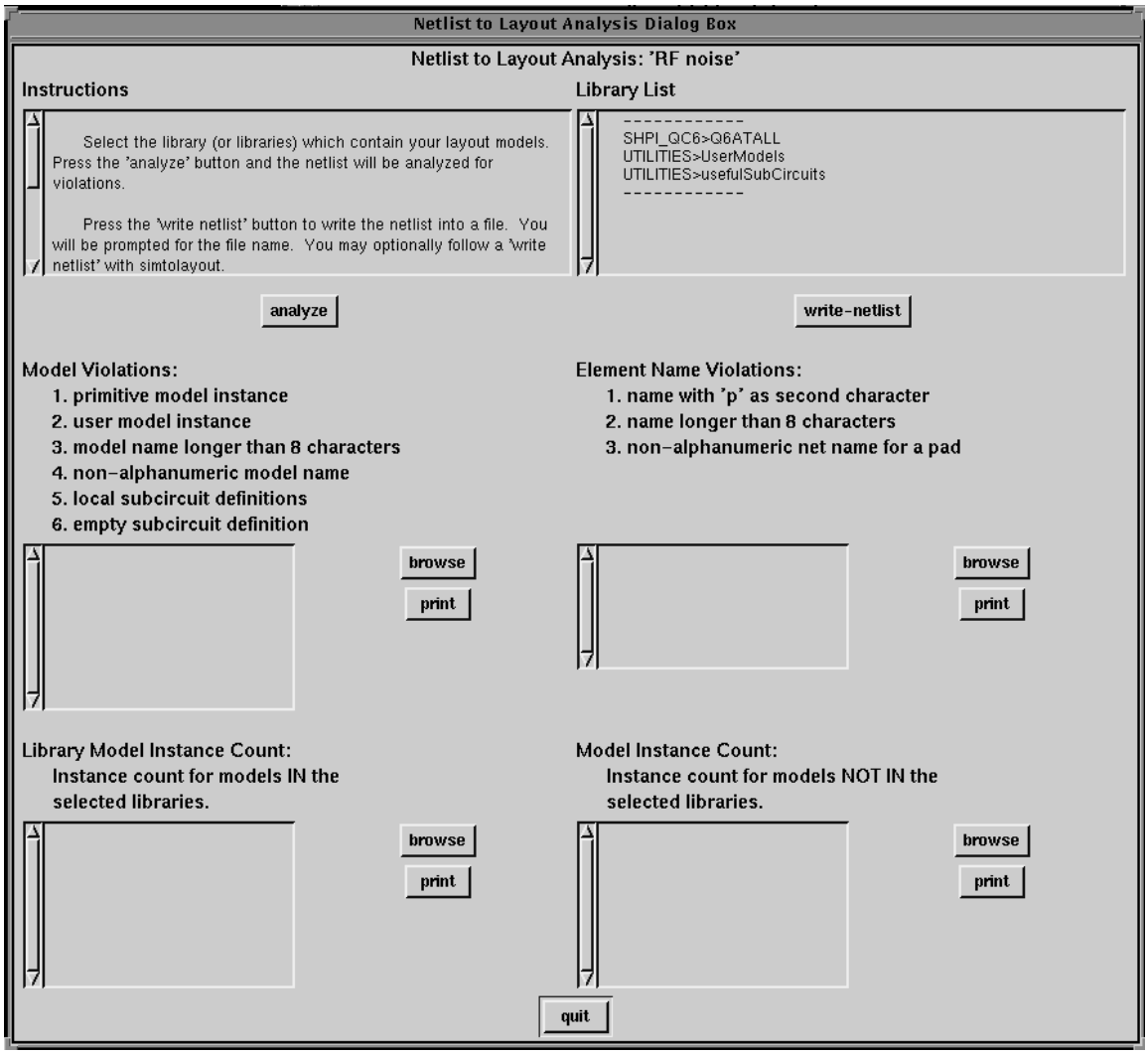


Figure 3–56: Netlist to Layout Analysis Dialog Box

Access

Access the **Netlist to Layout Analysis** dialog box from the **Circuit Editor** or **Subcircuit Editor**'s *Edit/Analysis/Annotation Pane* using the pop–up button **info > layout info** command.

Options

Select the library or libraries which contain the layout models for the circuit under analysis and choose from the following options.

analyze. Analyze the circuit for violations based on the model libraries selected.

write–netlist. Write a netlist compatible with the netlist–to–layout application software to the prompted file. You can optionally startup the netlist–to–layout application.

browse. Allows editing definitions with violations. Each violation pane has a separate **browse** button.

print. Prints a list of violations. Each violation pane has a separate **print** button.

Parasitic Configuration Dialog Box

The **Parasitic Configuration Dialog Box** allows specifying the name of a parasitic capacitance netlist file. It also allows specifying whether this file should be included as part of the circuit description during simulation.

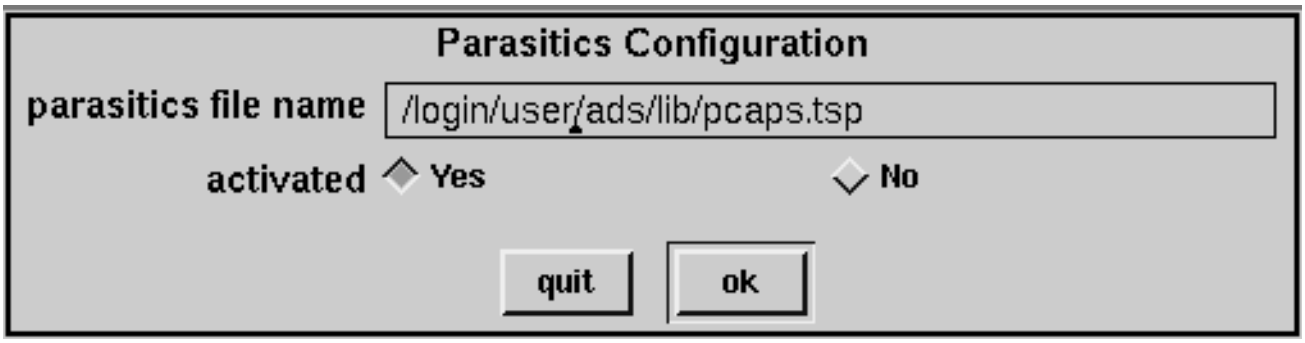


Figure 3–57: Parasitic Configuration Dialog Box

Access

Access the **Parasitic Configuration** dialog box from the **Circuit Browser** *Circuit List Pane* using the pop–up button **configure parasitics** command.

Options

parasitics file name. Allows specifying the path and filename of the parasitic netlist file.

activated. If yes is selected, the netlist file specified is appended to the circuit netlist during the TekSpice simulation. If no is selected the file is not appended to the circuit netlist.

(This page intentionally left blank)

Plotter Configuration Dialog Box

The **Plotter Configuration** dialog box specifies default sizes for the **Plot Browser** and the postage stamp plots in the *Schematic Editor Pane*.

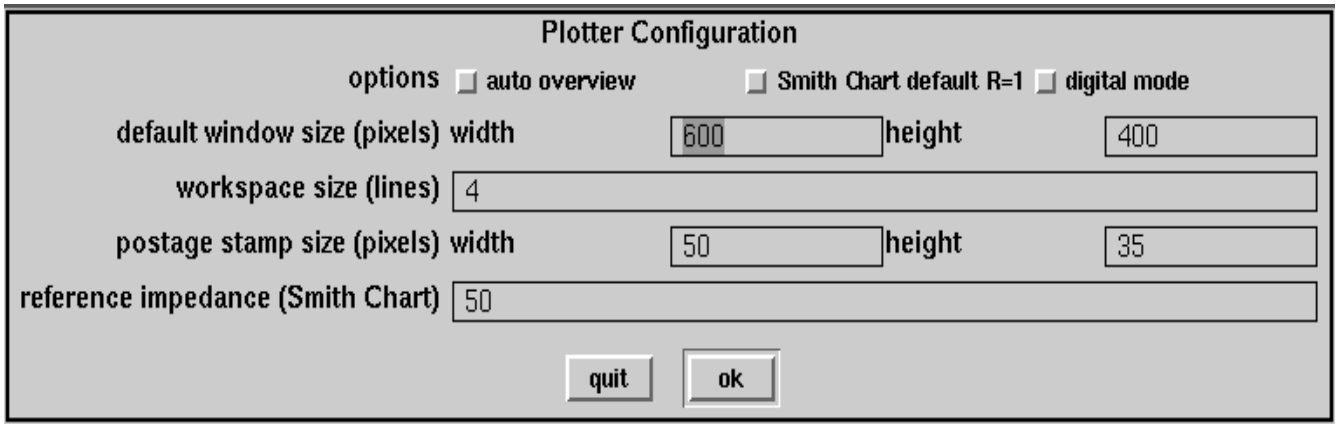


Figure 3–58: Plotter Configuration Dialog Box

Access

Access the **Plotter Configuration** dialog box from the **Launcher** using the **Configure > plotter** command.

Options

The options in the Plot Configuration window are as follows:

auto overview. Automatically adjusts the plot to fill the *Waveform Pane* when waveforms are cut or pasted to the plot.

Smith Chart default R=1. If selected, the standard Smith chart is displayed initially regardless of the scope of the data. This has a normalized radius of 1 (ohm). If not selected, the Smith chart or extended Smith chart will display all the data.

digital mode. If selected the Plot Browser operates in the digital mode. This operation is as follows:

a. Multiple Y–axes are stacked vertically.

b. The pop–up button command **paste** puts each waveform on a new Y–axis.

c. The pop–up button command **window** affects only the x (time) axis.

d. Scrolling is allowed only in the x–axis.

default window size(pixels). Sets the default pixel width and height of a **Plot Browser**

workspace size. Sets the number of lines of text allocated to the *Workspace Pane* of a **Plot Browser**.

postage stamp size(pixels). Sets the pixel width and height of a postage stamp plot.

reference impedance(Smith Chart). Specifies the reference impedance used to normalize the Smith Chart impedance calculations.

Print Options Dialog Boxes

The **Print Options** dialog box allows choosing format options for sending graphic or text output to a printer or file.

Graphic Settings

Figure 3–59 shows the dialog box for setting the graphic print options.

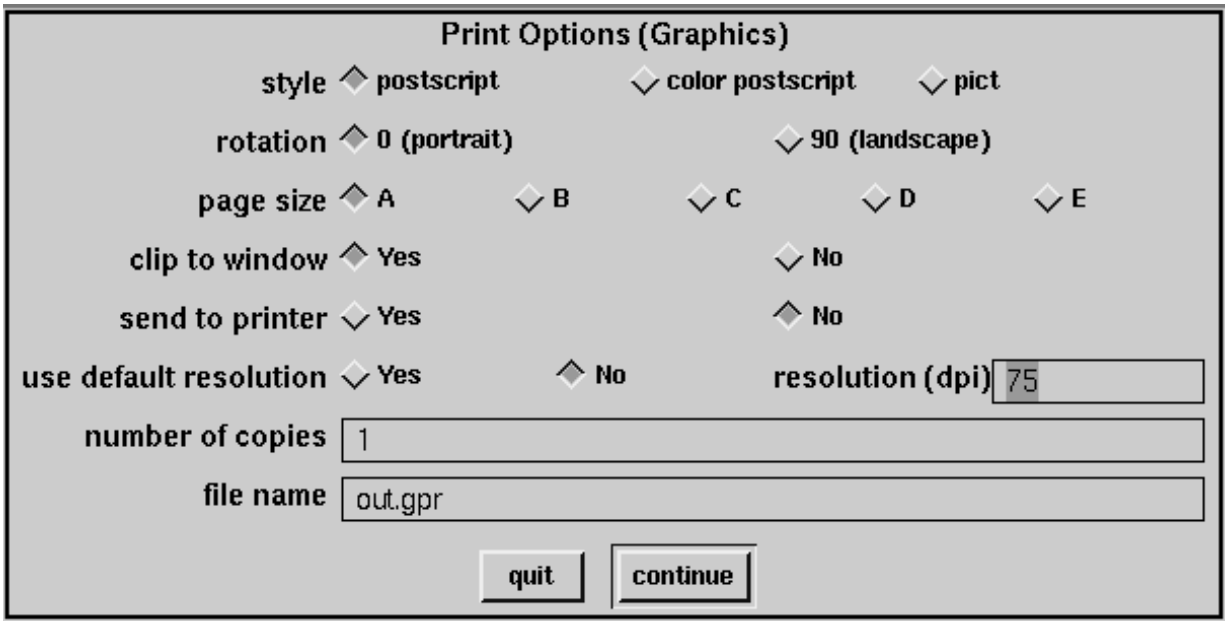


Figure 3–59: Print Options (Graphic) Dialog Box

Access

Access the **Print Options (Graphics)** dialog box from any graphic pane which lists the special button menu **print window** command or pop–up button **print** command.

Options

The choices for the **Print Options** dialog box are as follows:

style. Sets the data format for the graphic output file. Choose between **postscript**, **color postscript** or **pict**. Postscript can be level 1 or level 2 as determined by the **Printer Scripts Configuration (Graphic)** dialog box

rotation. Choose between **portrait** (long direction is vertical) or **landscape** (long direction is horizontal) rotation.

page size. Specify paper size for output; A–size is 8–1/2” x 11”; B–size is 11” x 17”; C–size is 17” x 22” and D–size is 22” x 34”. Most laser printers only print A–size.

clip to window. If yes, print the data shown in the window specified. If no, print the entire contents of the window specified.

send to printer. If **yes**, send the print **file name** directly to the printer specified in the **Printer Scripts Configuration (Graphic)** dialog box. If **no**, send the output to **file name** only.

use default resolution. If yes, use the default resolution for the printer specified in the **Printer Scripts Configuration (Graphics)** dialog box. If no, specify the resolution in the resolution text box.

resolution. Specify the number of dots per inch resolution for postscript output. Default is 300. To set resolution the **use default resolution** parameter must be set to **no**. This parameter can be used to set the line width for postscript plots. If resolution is set to 75, the line width would be 4 times wider than if set to 300.

number of copies. Specifies the number of copies to print. Default is 1.

file name. Specifies the file name where print data is sent. A path can be included.

Text Settings

Figure 3–60 shows the dialog box for setting the text print options.

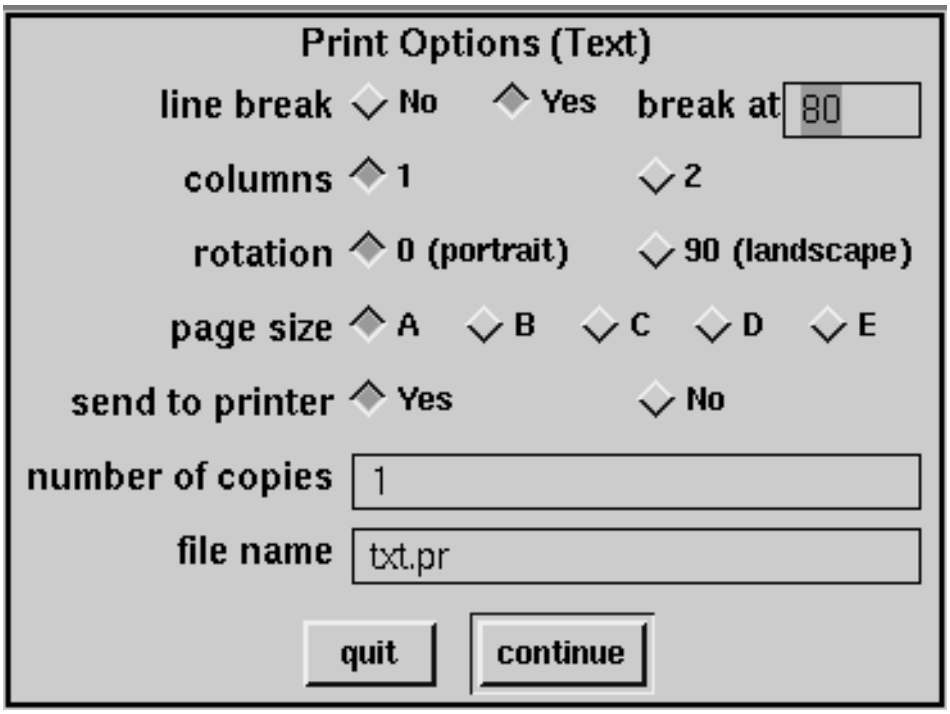


Figure 3–60: Print Options (Text) Dialog Box

Access

Access the **Print Options** (Text) dialog box from any text pane which lists the pop–up button menu **print** command.

Options

The choices for the **Print Options** dialog box are as follows:

line break. If **No**, line breaks are not inserted in the text. If **Yes,** line breaks are inserted at every *n* characters as specified by the **break at** parameter. Default is 80.

columns. Specify one (**1**) or two (**2**) columns. Default is one (**1**).

rotation. Specify print rotation as **0** degrees for portrait or **90** for landscape prints.

page size. Specify paper size for output; A–size is 8–1/2” x 11”; B–size is 11” x 17”; C–size is 17” x 22” and D–size is 22” x 34”. Most laser printers only print A–size.

send to printer. If **Yes**, send the print **file name** directly to the printer specified in the **Printer Scripts Configuration (Text)** dialog box. If **No**, send the output to **file name**.

number of copies. Specify number of copies to print. Default is 1.

file name. Specify file name where print data is sent. Path can be included.

Printer Scripts Configuration Dialog Boxes

ADS supports both graphics and text printing. There are two dialog boxes for configuring printer output: the **Printer Scripts Configuration (Graphics)** dialog box shown in Figure 3–61 and the **Printer Scripts Configuration (Text)** dialog box shown in Figure 3–62. Both of these dialog boxes allow the user to input the required information for sending data to a printer.

Printer Scripts Graphics

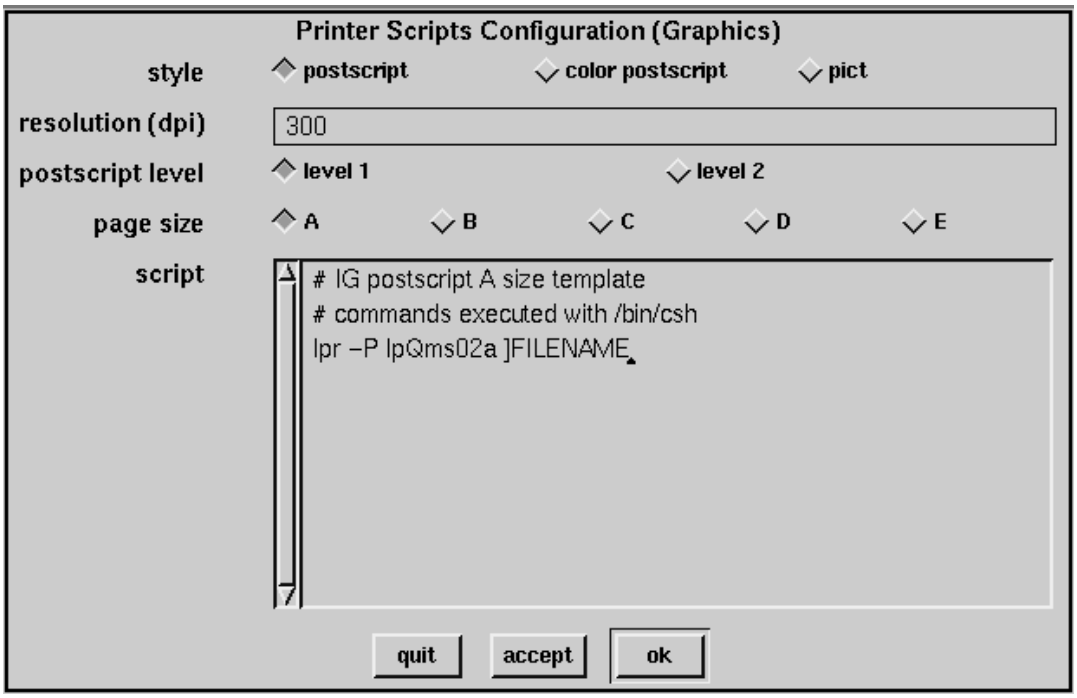


Figure 3–61: Printer Scripts Configuration (Graphics) Dialog Box

Access

Access the **Printer Scripts Configuration (Graphics)** dialog box from the **Launcher** using the **Configure > printer (graphics)** command.

Options

style. Selects output format; either postscript, color postscript or pict format.

resolution (dpi). Specifies the print resolution in dots per inch for the output file.

postscript level. Specifies the postscript level used in the output file as; level 1 (used by most laser printers) or level 2.

page size. Specifies paper size for output; A size is 8–1/2” x 11”; B size is 11” x 17”; C size is 17” x 22” and D size is 22” x 34”. Most laser printers will only print A size.

script. Specifies the script(s) for sending data to a specified printer. Each style, postscript level and page size combination requires a different script. These scripts contains the instructions sent to the operating system for printing the options selected. Many different scripts are allowed. The key word]FILENAME is replaced by the file name specified in the **Printer Options (Graphics)** dialog box.

Printer Scripts Text

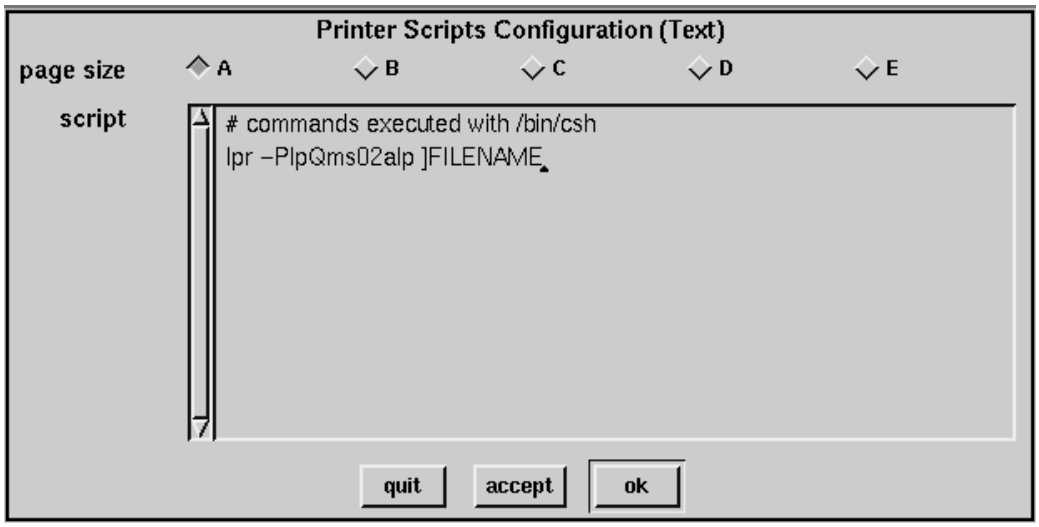


Figure 3–62: Printer Scripts Configuration (Text) Dialog Box

Access

Access the **Printer Scripts Configuration (Text)** from the **Launcher** using the **Configure > printer (text)** command.

Options

page size. Specifies paper size for output; A size is 8–1/2” x 11”; B size is 11” x 17”; C size is 17” x 22” and D size is 22” x 34”. Most laser printers will only print A size.

script. Specifies the script(s) for sending data to a specified printer. Each style, postscript level and page size combination requires a different script. These scripts contains the instructions sent to the operating system for printing the options selected. Many different scripts are allowed.

The key words:]FILENAME,]COLUMNS,]ROTATION, and]COPIES are replaced during a print operation by selections from the **Printer Options (Text)** dialog box.]FILENAME is the only required keyword.]COLUMNS is replaced by 1 or 2 depending on selection.]ROTATION is replaced by <blank> or ”r” depending on selection.]COPIES is replaced by the number of copies requested. If the optional keywords are not present in the script text, then the specifications on the **Printer Options (Text)** dialog box are ignored.

(This page intentionally left blank)

Prune Results Dialog Box

The **Prune Results** dialog box removes simulation results files that are no longer needed. As long as only one ADS image is run from the ADS directory and all TekSpice simulations are done from within ADS, then it is safe to remove all the files on this list.

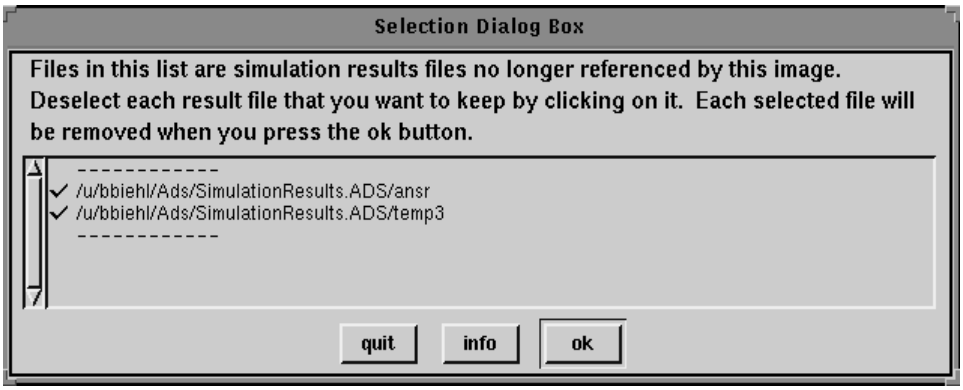


Figure 3–63: Prune Results Dialog Box

Access

Access the **Prune Results** dialog box from the **Launcher** using the **System > pruning > prune results files** command.

(This page intentionally left blank)

Simulation Status Dialog Box

The **Simulation Status Dialog Box** lists the messages from the TekSpice simulator during simulation. If the simulation completes without errors, this box automatically closes and transfers this information to the **System Transcript** Dialog Box. A snapshot of this dialog box is shown in Figure 3–64.

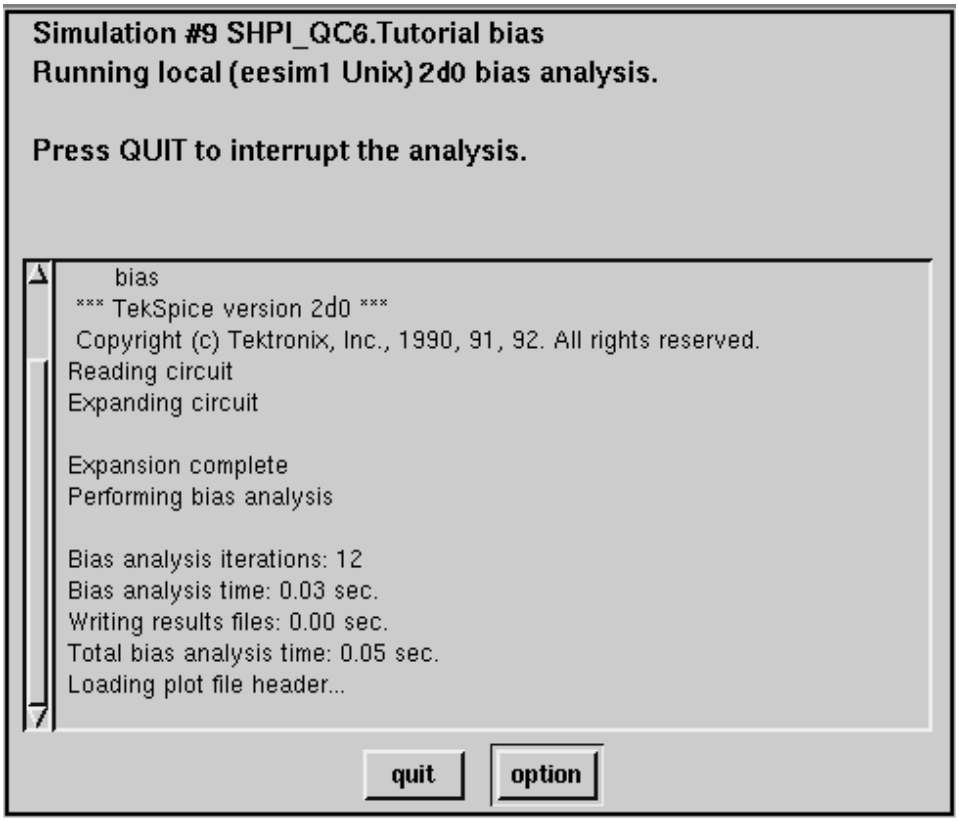


Figure 3–64: Simulation Status Dialog Box

Access

The **Simulation Status Dialog Box** automatically opens when a TekSpice simulation starts.

Options

The options for the **Simulation Status Dialog Box** are as follows:

quit. Aborts the TekSpice simulation and closes this dialog box. If the simulation has errors, then the **Simulation Status** Dialog Box is closed using this command.

option. Opens a menu of commands for running an analysis unattended, saving or exiting when done, etc. Select the cancel option to cancel a previously selected option. The option is exercised when the analysis completes. . The options button is inactive after the simulation completes. The selected option is displayed in the **Simulation Status Dialog Box**.

Simulator Configuration Dialog Box

The **Simulator Configuration** dialog box allows the selection of the TekSpice circuit simulator options available from within ADS.

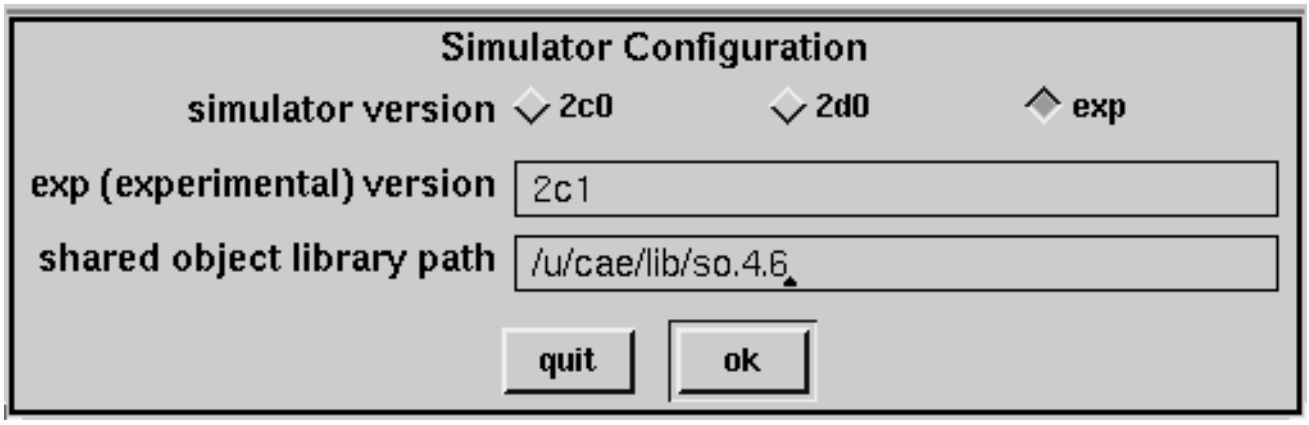


Figure 3–65: Simulator Configuration Dialog Box

Access

Access the **Simulator Configuration** dialog box from the **Launcher** dialog box using the **Configure > simulator** command.

Options

The options available from the **Simulator Configuration** dialog box are as follows:

simulator version. Select the version of TekSpice you want ADS to use.

exp version. Specify the experimental version number for the **exp** simulator if selected.

shared object library path. The shared object library path is normally blank and is used only when custom models are added to ADS/TekSpice. In this case the ADS support person can advise you of the library path, and directory name to include here.

(This page intentionally left blank)

System Configuration Dialog Box

The **System Configuration** dialog box allows specifying setup options for the ADS system.

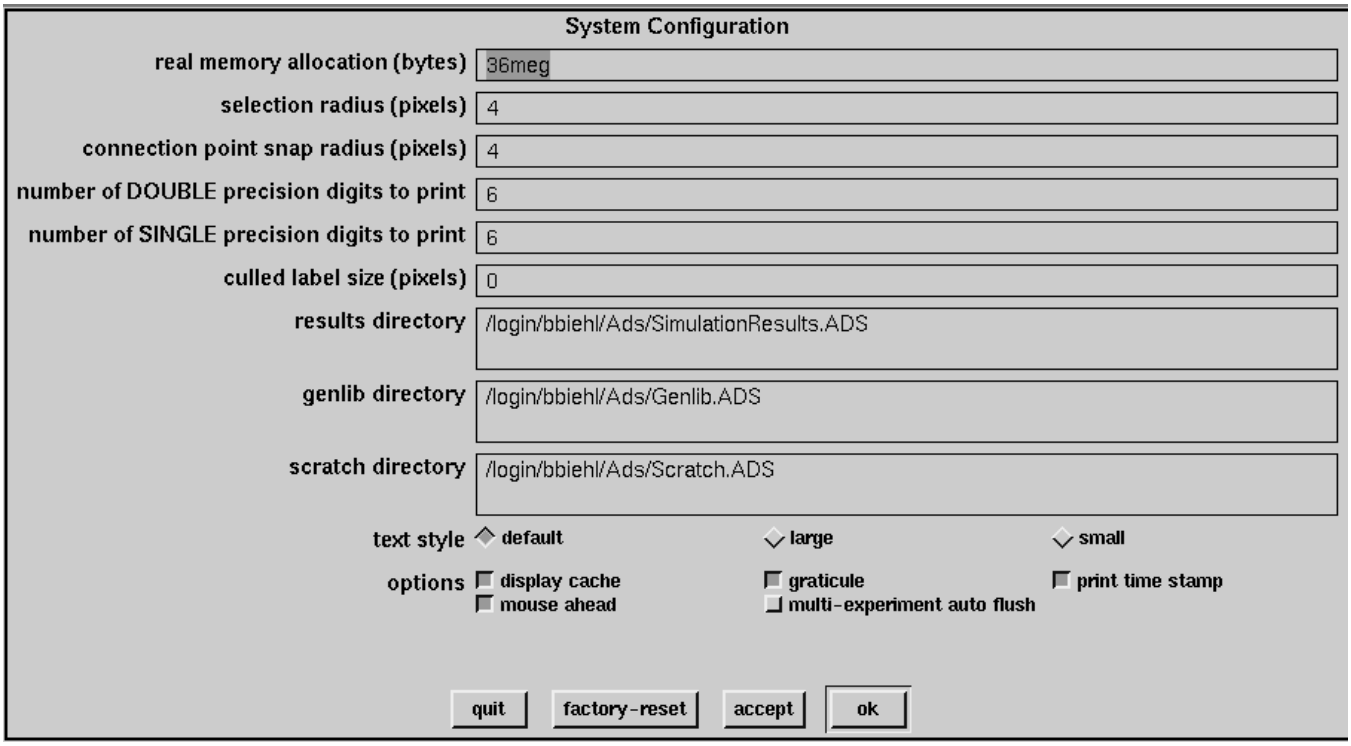


Figure 3–66: System Configuration Dialog Box

Access

Access the **System Configuration** dialog box from the **Launcher** dialog box using the **Configure > system** command.

Options

The following options are available for customizing the ADS setup (the recommended setting is ”on” for all options):

real memory allocation (bytes). Specifies the amount of real memory requested from the workstation when running ADS. See note below.

selection radius (pixels). Specifies the number of pixels around the point of the cursor considered active when selecting. The larger the radius the easier it is to select an object but harder to differentiate two objects in proximity.

connection point snap radius (pixels). Specifies the number of pixels around a connection point considered active when wiring or pasting.

number of DOUBLE precision digits to print. Specifies the number of double precision floating point digits printed in the *Workspace Pane* and on postage stamps. Default is 6.

number of SINGLE precision digits to print. Specifies the number of single precision floating point digits printed in the *Workspace Pane* and on postage stamps. Default is 6.

culled label size (pixels). Specifies the minimum size in pixels for a label. If the label will be smaller than the specified pixels, it will not be displayed. This is a dynamic operation dependant on the window scale.

results directory. Specifies the directory for saving simulation results.

genlib directory. Specifies the directory for saving genlib files for encapsulated EDIF's.

scratch directory. Specifies the directory for saving temporary scratch files.

text style. The text style used by ADS.

options. The following ADS options are toggled "on" when selected:

display cache – use memory to cache the window displays. Set to off when display memory is limited.
graticule – display graticule in schematics
print time stamp – print time and date on hard copies
mouse ahead – store mouse clicks in buffer

packages. There are several special purpose options that are not part of the standard ADS release but can be activated by ADS support personal. These options are:

advanced model creation – Allows local subcircuit models and TekSpice1 primitives. This option is used for working with old ADS images.

cell characterization – Used for bipolar digital standard cell characterization.

differential netlist – This is a prototype package used for adding differential wires to a schematic.

interconnect model synthesis – Allows the generation of coupled transmission line models.

remote simulation – Allows simulation on the cpu of another workstation on the network.

NOTE*. The* ***real memory allocation*** *value affects the frequency of garbage collections. If the ADS image grows to the size specified by* ***real memory allocation*** *ADS considers garbage collection. If an image grows beyond the size of the workstation RAM, paging begins, causing significant delays. Both paging and garbage collection slow down a workstation.*

When garbage collection begins the GC icon and the Garbage Can icon appear frequently — to the point that they appear several times during an operation such as a circuit edit.

There is a trade–off between the frequency of ”garbage collection” and ”real memory allocation”. For a workstation with 64 Meg of RAM set ***real memory allocation****, to 52Megabytes. There will then be 12 Meg of RAM remaining for X11, the window manager and the OS.*

The Sun OS operating system does not decrease the size of an executing program. Once an ADS image grows to the point where paging starts, the only way to reclaim process space is to save the image and restart ADS. Closing ADS windows only reclaims space for ADS but does not shrink the UNIX process size.

(This page intentionally left blank)

System Consistency Dialog Box

The **System Consistency Dialog Box** checks all the circuits and libraries stored in the ADS image for internal problems. Figure 3–67 shows the **System Consistency Dialog Box**.

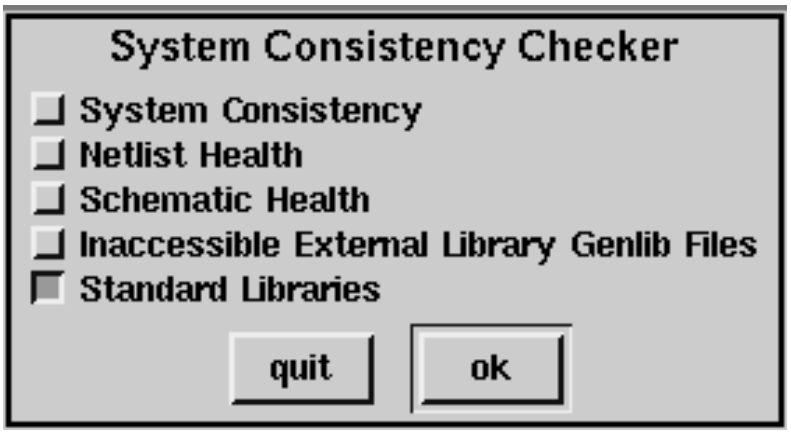


Figure 3–67: System Consistency Dialog Box

Access

Access the **System Consistency Dialog Box** from the **Launcher** using the **System > Checker > System Consistency** command.

Options

The options on the **System Consistency Dialog Box** are as follows:

System Consistency. Checks that circuits have access to all models they reference. Checks that all references to a model are consistent.

Netlist Health. Checks that the internal circuit description data structures are consistent.

Schematic Health. Checks that the schematic agrees with the circuit description.

Inaccessible External Library Genlib Files. Checks for external libraries in the ADS image that refer to an inaccessible genlib file.

Standard Libraries. Identifies ’standard” libraries in the ADS image that are not the most recent version.

(This page intentionally left blank)

System Installation Checker Dialog Box

The **System Installation Checker** dialog box allows the checking for proper installation of the selected option(s).

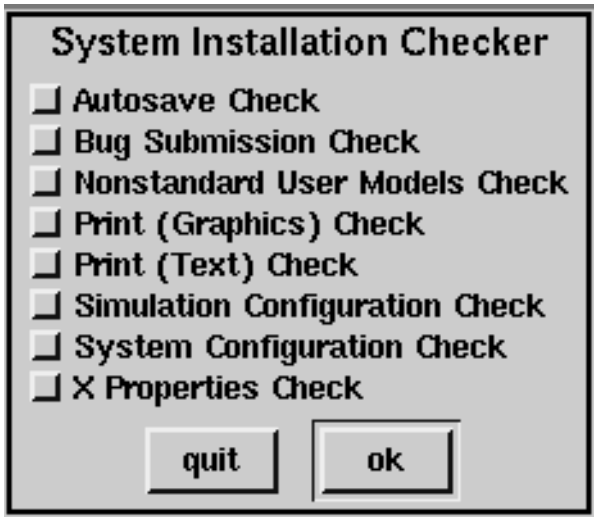


Figure 3–68: System Installation Checker Dialog Box

Access

Access the **System Installation Checker** dialog box from the **Launcher** dialog box using the **System > checker > installation** command.

Options

The following checks are available from the **System Installation** dialog box:

Autosave Check. Checks values specified in the **Automatic Save Configuration** dialog box.

Bug Submission Check. Checks values specified in the **Bug Report Configuration** dialog box.

Nonstandard User Model Check. Checks for obsolete nonstandard user models.

Print (Graphics) Check. Checks values specified in the **Print Configuration (Graphics)** dialog box.

Print (Text) Check. Checks values specified in the **Print Configuration (Text)** dialog box.

Simulation Configuration Check. Checks values specified in the **Simulation Configuration** dialog box.

System Configuration Check. Checks values specified in the **System Configuration** dialog box.

X Properties Check. Checks values specified in the .Xdefaults file.

System Transcript Dialog Box

The **System Transcript** dialog box gives feedback from ADS to the user. Error and warning messages, time and date of last save are examples of information conveyed to the user in this window. The **System Transcript** dialog box should remain open at all times. Figure 3–69 shows the **System Transcript** dialog box.

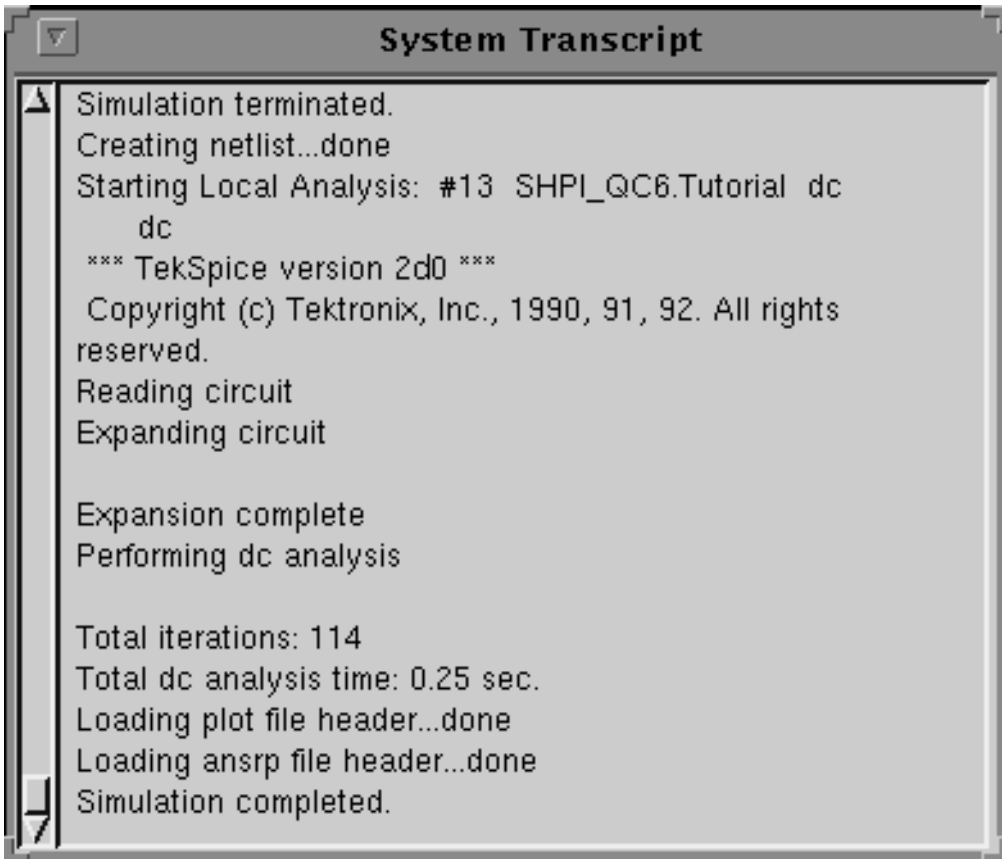


Figure 3–69: System Transcript Dialog Box

Access

Access the **System Transcript** dialog box from the **Launcher** using the **Open > system transcript** command. This dialog box should remain open at all times.

(This page intentionally left blank)

User Interface Configuration Dialog Box

The **User Interface Configuration** dialog box allows setting of ADS interface parameters. Figure 3–70 shows the **User Interface Configuration** dialog box.

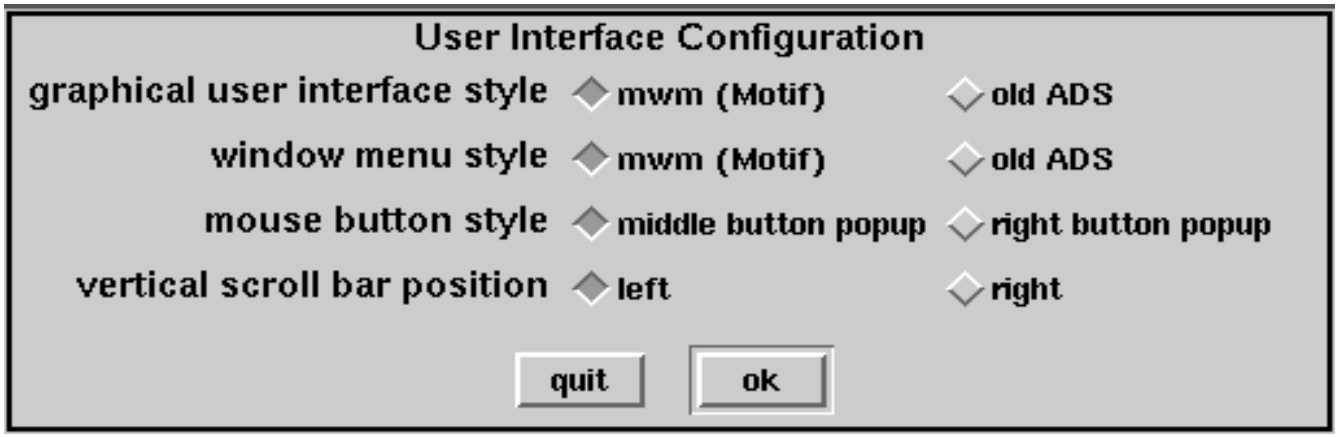


Figure 3–70: User Interface Configuration Dialog Box

Access

Access the **User Interface Configuration** dialog box from the **Launcher** using the **Configure > user interface** command.

Options

Old ADS refers to versions 3 and older. Options for the **User Interface Configuration** dialog box follow:

graphical user interface style. Allows the choice between the Motif or old ADS GUI window manager ”look and feel” styles.

window menu style. Allows the choice between the Motif or old ADS GUI window manager menu styles.

mouse button style. Allows the choice between placing the location of the pop–up button at the middle or right mouse button. The use of mouse buttons is described in more detail in the *Overview* chapter in the section: *Using the Mouse Buttons*.

selection style. Determines the function of the mouse buttons. The standard selection style deselects previous items selected, then selects the new item(s). The shift key is used for additive selection and the control key for subtractive selection. In the old ADS selection scheme, selections were additive. Deselect is done with the select pop–up menu command.

vertical scroll bar position. Choose between scroll bar on left or right side of pane.

Panes

Panes are found in all of the ADS browsers and editors and in some of the dialog boxes. The panes used in ADS are described in this section and listed below. This is the same description included with each browser, editor and dialog box.

- Accept Pane
- Circuits List Pane
- Comment Text Pane
- Control Program List Pane
- Control Program Text Pane
- Cutoff Change Pane
- Edit/Analysis/Annotation Pane
- Experiment Browser Pane
- Experiment Table Pane
- File Name Pattern Pane
- File List Pane
- File Editor Pane
- Historical Experiment Table Pane
- Library List Pane
- Library Path Pane
- Locator Pane
- Lock Status Pane
- Model List Pane
- Node List Pane
- Overlay Pane
- Parameter List Pane
- Parameter List Selector Pane
- Process List Pane

- Project List Pane
- Results Table Pane
- Schematic Editor Pane
- Simulation Control Pane
- Simulation Status Pane
- Symbol Editor Pane
- Symbol Label Pane
- Table List Pane
- Terminal List Pane
- Text List Pane
- Text Workspace Pane
- Waveform Pane

Accept Pane

The contents of the *Accept Pane* indicated the status of the information in the *Schematic Editor Pane.* If the *Accept Pane* is a **green** box; this indicates that the schematic has been ”accepted” and no modifications have been made in the *Schematic Editor Pane*. If the *Accept Pane* is a **yellow** box containing a diagonal slash; this indicates modifications were made in the *Schematic Editor Pane*. This box remains yellow until the pop–up button menu command **accept** is issued from within the *Schematic Editor Pane*. If the *Accept Pane* is a **red** box containing an X; this indicates the *Schematic Editor Pane* is obsolete. This is usually due to changes made in another window.

Circuits List Pane

The *Circuit List Pane* lists the circuit names in the present ADS image. Pop–up button menu commands accessed within this pane support adding, renaming, copying, and removing circuits. They also allow reading and writing circuit files in EDIF format as well as adding or deleting simulation libraries, global nodes and circuit variables. See **EDIF Dialog Boxes**.

Comment Text Pane

The Comment Text Pane contains text describing the item selected in the active pane. Use the standard text editing commands in this pane. See the **Getting Started** chapter.

Control Program List Pane

The *Control Program List Pane* lists the presently defined control programs.

Control Program Text Pane

Control Program Text Pane displays the contents of a selected control program. Use the standard text editing commands for modifying the control programs. The first line of all control programs contain the statement **clear**, which resets all the variable assignments to zero, before a simulation. The contents of the control program may include (order dependent): Variable assignments (if any); TekSpice simulation command blocks; and Post–simulation commands and assignments. See also **circuitmenu**, **wiremenu**, and Waveform Functions.

Cutoff Change Pane

The *Cutoff Change Pane* allows displaying only those elements whose noise sums are above a certain percentage of the circuit’s total output noise. This is helpful in a large circuit, or a circuit where many elements have noise sums of zero. The pop–up button **change cutoff** command allows changing the cutoff percentage at any time. The cutoff percentage does not take affect until “total” is selected (see *Table List Pane*).

The normalized total noise contribution (in %) for element E, $n_{TCE,}$ of any circuit element is defined as follows.

$$n_{TCE}\ (\%) = \sqrt{\frac{\int \sum n_{cE}}{\int (onoise)^2}}$$

where; n_{cE} is the noise contribution for element E; and *onoise* is the circuit's output noise. In a TekSpice **varsweep** analysis, both n_{cE} and *onoise* are families of waveforms. The maximum value for n_{TCE} across all var sweep values is used for each element.

Edit Pane

Edit/ Annotation Pane

The *Edit/Annotation Pane* affects the operation of the *Schematic Editor Pane*. Toggling this pane with the select button changes the displayed text between EDIT and ANNOTATION.

Edit/Analysis/ Annotation Pane

The *Edit/Analysis/Annotation Pane* affects the operation of the *Schematic Editor Pane*. Toggling this pane with the select button changes the displayed text between EDIT and ANALYSIS. The pop–up button allows selection of EDIT, ANALYSIS or ANNOTATION

Experiment Text Pane

The *Experiment Text Pane* shows the library path, the variable assignments, and the TekSpice control program commands used to run the simulation. Also shown are the results file names and simulation descriptions. If a simulation was not run, the commands which will be used are shown instead.

Experiment Table Pane

The *Experiment Table Pane* sets up and execute simulations for a selected control program. This pane displays one row for each simulation command block in an analyzed simulation control program. The Experiment Table shows the selected variable values, providing a method for viewing the variable assignments used in the simulation. Control programs can be executed and the Experiment Table format can be modified from this pane.

This table displays preset variables and user defined circuit variables, and local variables. Rows of experiments associate simulation results with the values for those variables. The displayed items can be selected from all available variables. The selected variable names appear above the values assigned to them. Experiments are further identified by an automatically generated number and by the analysis categories; ac, bias, dc and tran.

Variables may be assigned values in the Experiment Table to create individual control programs from the template in the *Control Program Text Pane*. In this

case for each row of the experiment table, a TekSpice command block is generated of the form:

```
variable assignments
.
.
simulation control program
```

File Name Pattern Pane

The *File Name Pattern Pane* allows typing existing or new file names or patterns for file names. Special characters * and # are permitted. ”* ” matches any sequence of none, one, or more characters. ”#” matches any one character. Issuing the pop–up button **accept** command displays the name or pattern into a list of matching directories, files, or both in the *File List Pane*.

File List Pane

The *File List Pane* shows the files selected from an **accept** command in the File Name Pattern Pane.

File Editor Pane

The *File Editor Pane* displays and allows editing the contents of files selected in the *File List Pane*. Use the standard text editing commands in this pane.

Historical Experiment Table Pane

A *Historical Experiment Table Pane* is located in the bottom pane of the **Control Program Browser**. An experiment can be copied from the working table to the *Historical Experiment Table Pane* with the pop–up button menu **move experiment to history** command. Do this to save results. The **foreign experiment** command is added to the historical table for access to simulation results from a circuit other than the active circuit in the **Control Program** browser.

Library List Pane

The *Library List Pane* displays the list of libraries available for the selected process. Libraries on this list may be internal libraries or external libraries. Selecting a library lists the available models in the *Library Model List Pane*.

Library Model List Pane

The *Library Model List Pane* provides the list of user and subcircuit models available for use in the creation of a circuit schematic. The highlighted library in the *Library List Pane* determines the names on this list. Library models can be added, modified or removed from this list. The typface font of model names indicate the model type: *italics* font indicates a local model, a regular font indicates a library model; and **bold regular** font indicates a primitive model. ADS version 4d0 and above promote using only the library model.

Library Path Pane

The *Library Path Pane* lists the libraries available to the circuit specified by the **Circuit Browser**. Pop–up button commands allow adding or deleting libraries from this list. Any library listed in the **Library Browser** can be added to this list. Selecting a displayed library(s) and using the pop–up button accept

command adds these library models to those available from the Circuit Editors' *Model List Pane* pop–up button **library** command. The order of the libraries is important. If more than one of the libraries listed includes a model definition with the same name, the first occurrence in the list is used.The order of the library list can be changed by editing.

Locator Pane

The *Locator Pane* provides the ability to locate circuit elements (instances), models, nets and nodes. Locating these elements models or nodes is possible in the local circuit or at other levels of hierarchy. The symbol of the selected model element or node is identified by highlighting on the circuit.

Lock Status Pane

The *Lock Status Pane* indicates the status of the selected library in the *Library List Pane* of the **Library Browser**. The status can be **read/write** or **read only**. This status can be changed using the popup button command **change library** in the *Library List Pane* of the **Library Browser**.

Model List Pane

The *Model List Pane* provides the list of library models available for use in the creation of a circuit schematic. Pop–up button menu commands allow adding and removing models from this list.

Model List Pane2

The Model List Pane lists the hierarchical names of circuits or subcircuits which can be located from this pane. A schematic of the selected name is displayed in the *Schematic Editor Pane*; the associated models and node connectors appear in the *Node List Pane*. The searched item appears in the *Locator Pane*. This item may be selected to locate its position in the schematic.

Node List Pane

The *Node List Pane* provides the names of local and global nodes used in the circuit. Schematic symbols representing these nodes are attached to wires or terminals in the circuit to establish the node (net) name. The local nodes listed are those only used in the schematic displayed in the *Schematic Editor Pane*. Global nodes are listed in boldface print, local nodes listed in italics print. Global nodes are constant throughout the circuit hierarchy.

Node Type List Pane

The *Node Type List Pane* provides the names of the node types used in the ADS image. The node types are: external, global, ground and local. **Node Symbol Editor** allows viewing the schematic symbols representing these nodes. An external node is used to define the ports of a subcircuit. Global nodes are defined only once throughout the circuit hierarchy. The ground node is the special case for a global node. Local nodes are present only in the displayed schematic.

Non–Cartesian Waveform Pane

The *Non–Cartesian Waveform Pane* is the output plot area for the **Smith Plot Browser**. It is divided into three areas; the Axis Menu Area; the Waveform Menu Area; and the Waveform Selector Menu Area.

Axis Menu Area. The Axis Menu Area commands allow the changing of the plot scale, plot labels and the general appearance of the axes.

Waveform Menu Area. The Waveform Menu Area commands allow for editing and operating on plot waveforms. Adding labels, re–scaling axes and cutting and pasting of waveforms are allowable. The **track** mode allows reading data values following a waveform. Clicking on a point causes printing of the x and y values on the plot. The **delta** command reports value of an offset from the last point marked on the plot. The **value** mode is similar except that it is not constrained to follow a waveform. The **clear** command removes the tic marks.

Waveform Selector Menu Area. The Waveform Selector Menu Area commands allow adding and removing plot waveforms in the Waveform Menu Area. When managing many waveforms in a window, those not of immediate interest can be made invisible and later restored with a button click. Waveforms can be cut from a plot, and the remaining can be re–sequenced for improved clarity of colors, labels, and line styles. A label for all waveforms in a family of waveforms allows the total selection for **delete/copy** and make **visible/invisible** commands.

Overlay Pane

The *Overlay Pane* shows the selected sheet names and provides for their control. Sheets may be thought of as two layers of graphics overlaying the *Schematic Editor Pane*. The user may place simulation output postage stamp plots on either or both sheets for comparing outputs of one simulation to another. A third layer, scratch, is also available. This layer is less permanent and is erased every time a simulation is run.

Parameter List Pane

The *Parameter List Pane* defines input parameters, assign values to them and defines output parameters. The displayed names are either input or output parameter names or local variable names depending on the mode set in the *Parameter List Selector Pane*. If a user symbol is being created, local variables may not be created and only value changes can be made for the input parameters.

The syntax for defining output parameters for subcircuit models is as follows:

output(element, parameter) is the value of the named **parameter** on the named **element**. Element name is relative to the definition of the subcircuit where the output function is used.

i(element,terminal number) is the swept current through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where **i()** is used.

v(N) is the swept voltage on node N, where N is named relative to the definition of the subcircuit where **v()** is used.

bi(element,terminal number) is the bias (operating point) cuirrent through **terminal number** of **element**. **Element** is named relative to the definition of the subcircuit where the **bi()** is used.

bv(N) is the bias (operating point) voltage on node N where N is named relative to the definition of the subcircuit where the **bv()** is used.

The syntax for primitive and user models is as follows (where T is an integer greater than 0 and less than or equal the terminal count on the model):

primitivebv(T) is the operating point voltage at terminal T.

primitivebi(T) is the operating point current into terminal T.

primitivev(T) is the swept voltage at terminal number T of the given element.

primitiveoutput(P) is the swept value of parameter P defined in the primitive model for the given element from the swept analysis (Ac, dc, or Tran). If not defined here then it is undefined.

primitiveparameter(P) is the value of parameter P defined in the primitive model for the given element from the initial (bias, dc, ac or tran) analysis. If not defined here then it is from the 'tempAdjust' outputs. If not defined here then it is from the initial parameter values. If not defined here then it is undefined.

Parameter List Selector Pane

The *Parameter List Selector Pane* allows setting the operating mode of the *Parameter List Pane*. Setting the mode to **input parameters** causes the parameter names and value in the *Parameter List Pane* to represent subcircuit input parameters. Setting the mode to **output parameters** causes the parameter names and values in the *Parameter List Pane* to represent subcircuit output parameters. Setting the mode to **local variables** causes the parameter names and values in the *Parameter List Pane* to represent expressions and assignments used to set subcircuit element values. The **local variables** mode is not available from the **Library User Symbol Editor**.

Process List Pane

The *Process List Pane* lists the processes available in the current ADS image. Selecting a process lists the available libraries in this process in the *Library List Pane*.

Project List Pane

The *Project List Pane* lists the project names in the present ADS image. Pop–up menu commands within this pane support adding, renaming, copying, and

removing projects. They also allow reading and writing project files in EDIF format.

Reference Circuit List Pane

The *Reference Circuit List Pane* lists the subcircuits within the circuit hierarchy containing a selected model. Selecting a subcircuit from this list displays a schematic of this subcircuit is in the *Schematic Editor Pane*. The *Model List Pane* lists the associated models; the *Node List Pane* lists the nodes. The ”located” model appears in the *Locator Pane*. Selecting this model can then locate its position in the schematic.

Results Table Pane

The *Results Table Pane* contains data from the selected table. If there is no data for the rows, labels appear instead of numbers. The pop–up button menu commands change depending on the selection of a row label, a column label, a single cell or nothing.

In the **Noise Table Browser** the *Results Table Pane* is a set of postage stamp plots of noise versus frequency. The particular noise parameters are identified by the column headings. The contents of the *Results Table Pane* are controlled by the *Table List Pane* and the *Cutoff Change Pane*. At the top of each column heading are the units on the vertical axis of those postage stamps, for example V*V/Hz is volts squared per hertz. TekSpice only provides voltage noise outputs; input noise may be a current, so amps per square root of hertz is shown as one of two possible units on the inoise column.

Each row in the *Results Table Pane* represents one circuit element, except when “onoise and inoise” is selected in the *Table List Pane.* Selecting “onoise & inoise”, displays input and output noise for the entire circuit. For more information on the meaning of inoise, onoise, and the noise contributions, see the description of the TekSpice NOISE command.

Selecting the pop–up button command **total** in the *Table List Pane*, causes additional model names to appear on the list and the *Results Table Pane* to display total noise and n_{TCE} for each element. Selecting a model name from this list, displays circuit elements of that model type; all noise contributions; the sum of their noise contributions; and n_{TCE}.

Elements in the *Results Table Pane* are sorted from highest to lowest n_{TCD}. This is the same noise contribution number used to determine the cutoff, as described under *Cutoff Change Pane*.

Schematic Editor Pane

The *Schematic Editor Pane* is the main pane for creating and modifying circuit schematics and viewing simulation results. The operation of this pane depends on the mode of the *Edit/Analysis/Annotation Pane*. Elements are added to the *Schematic Editor Pane* by selecting a model in the *Model List Pane* and using the pop–up button menu **paste**. When pasting elements in the *Schematic Editor Pane*, ADS replaces the arrow cursor with the symbol of the selected model.

Before placement, a pop–up button menu is available for orienting the symbol. For copying, moving and deleting wires and elements use the **copy**, **cut**, **paste**, **move** and **adjust** commands.

Edit Mode. The EDIT mode allows editing the circuit schematic in the *Schematic Editor pane*.

Analysis Mode. The Analysis mode allows displaying the results of a simulation in the *Schematic Editor Pane*.

Annotation Mode. The Annotation mode allows adding text to a schematic in the *Schematic Editor Pane*. This text is on a separate ”layer” and can be erased or made invisible. Editing circuit graphics is not possible in the Annotation mode. Editing annotation text is not possible in the Edit or Analysis mode.

Text is added by selecting the **text** command from the pop–up button menu while in the *Schematic Editor Pane*. The cursor switches to an I–Beam. Click the cursor at the point where the text should start and begin typing. The right margin is defined by pressing the **Enter** key on the numeric keypad and continue typing. Clicking on existing text, opens an editor for that text; the **resize** command in the text menu allows re–sizing the text; the **select** command accepts the current text and goes to select mode. The **accept** command accepts the text and closes the editor on the current text item, but stays in the text mode. The bounding box of each line of text is separately selected for editing.

Simulation Status Pane

The *Simulation Status* pane is used to indicate the analysis status of the selected experiment. The *Simulation Status Pane* indicates when the analysis was done, and the version.

The *Simulation Status Pane* also shows the status of the last experiment, indicated by one of the following statements:

- no analysis (no simulation has been done)
- analyzing (simulation running)
- aborted (user quit the simulation)
- analyzed
- simulation error(s); results file(s) unusable
- results file(s) unusable
- simulation error(s) (simulator exited abnormally)
- analyzed; final point invalid
- simulation error(s); final point invalid
- analyzed; only partial results
- simulation error(s); only partial results

Symbol Editor Pane

The *Symbol Editor Pane* allows the creation of symbols. These symbols represents the subcircuit or model when placing an instance of that subcircuit or model in the schematic.

Symbol Label Pane

The *Symbol Label Pane* allows access to the **Visibility Panel** using the pop–up button **layer visibility** command. This panel supports access to label style (selecting visibility of labels), text scaling (size of placed text), layer visibility (graticule display, dots, snap–to–grid), and symbol printing.

Table List Pane

The *Results Table Pane* displays the table highlighted in the *Table List Pane*. Only data for the specified output parameters are computed.

For the **Noise Table Browser** the *Table List Pane* initially has the "onoise & inoise" table selected. In order to display noise contributions of individual elements, select "total" in the *Table List Pane*. This displays total noise for all elements above the cutoff (see *Cutoff Change Pane*).

Selecting "total" for the first time in a **Noise Table Browser** causes loading the individual elements' noise contributions into ADS. This step brings up a quit–button dialog box, since there may be a delay while loading results. A higher cutoff value (see *Cutoff Change Pane*) may give a shorter delay.

After loading the individual elements' noise contributions, the *Table List Pane* lists the model names for all elements passing the cutoff. Selecting a model name gives detailed information on all elements of that model.

Terminal Pane

The *Terminal Pane* lists the name of the terminal displayed in the *Symbol Editor Pane*.

Terminal List Pane

The *Terminal List Pane* supports access to the circuit and subcircuit connection terminals (ports). These terminals define the connections from the local circuit (the one in the *Schematic Editor Pane*) to the next level of circuit hierarchy. Terminals from this list must be pasted into the *Symbol Editor Pane* by selecting the terminal in the *Terminal List Pane* and then using the pop–up button command **paste terminal**, paste the terminal in the *Symbol Editor Pane*. If the terminal already exists then the **paste terminal** command does not appear in the pop–up button menu.

Text Workspace Pane

The *Text Workspace Pane* creates and edits TekSpice expressions for evaluation. These expressions are based on circuits and experiments identified in the *Simulation Status Pane*. The results of these evaluations can be printed or plotted. See also Waveform Functions and Waveform Commands.

Waveform Pane

The *Waveform Pane* is the output plot area for the **Plot Browser**. It is divided into three areas: the Axis Menu Area; the Waveform Menu Area; and the Waveform Selector Menu Area. Pop–up button menu commands depend on the location of the cursor.

Axis Menu Area. The Axis Menu Area commands allow the changing of the plot scale, plot labels and the general appearance of the axes. If the digital mode option is selcected, there can be multiple Y–axes, each of which can have a different scale, label, min/max, etc.

Waveform Menu Area. The Waveform Menu Area commands allow for editing and operating on plot waveforms. Adding labels, re–scaling axes and cutting and pasting of waveforms are allowable. The **track** mode allows reading data values following a waveform. Clicking on a point causes printing of the x and y values on the plot. The **delta** command reports value of an offset from the last point marked on the plot. The **value** mode is similar except that it is not constrained to follow a waveform. The **clear** command removes the tic marks.

Waveform Selector Menu Area. The Waveform Selector Menu Area commands allows adding and removing plot waveforms in the Waveform menu Area. When managing many waveforms in a window, those not of immediate interest can be invisible and later restored with a button click. Waveforms can be cut from a plot, and the remaining waveforms can be re–sequenced for improved clarity of colors, labels, and line styles. A label for all waveforms in a family of waveforms allows the total selection for **cut, copy** and make **invisible** commands.

Postage stamp plots on the schematic editor can be edited with the **cut, copy,** and **paste** commands. They can be pasted into any plot window. Minor tics are not longer for single decade log plots.

For a list of waveforms available for plot, type **list** and then select the **do it** command in the *Workspace Pane* of the **Plot Browser**.

Pop–up Button Commands

This section gives a brief description of the pop–up button commands found in the launchers, browsers, editors and dialog boxes. They are listed alphabetically by command. These are listed by the first level command; second level commands are listed alphabetically under the first level command.

A

accept Makes changes made within the editor known to the rest of ADS. Until an accept is issued, changes are local to the editor being used, and can be cancelled back to the last accept with the cancel command.

active **scratch**, **sheet 1**, **sheet 2**– Identifies which analysis results layer will receive new pasted results.

add ac Adds the default ac control program to the **Control Program Browser**.

add bias Adds the default bias control program to the **Control Program Browser**.

add circuit Adds a circuit with the specified name into the *Circuit List Pane* of the **Circuit Browser**. This command is available when no circuit is selected.

add circuit variable Allows adding a circuit variable to a selected circuit in the **Circuit Browser**. This command is only available when a circuit is selected.

add dc Adds the default dc control program to the **Control Program Browser**.

add external library Adds an external library with the specified name into the selected Library Process of the **Library Browser**. Specify the path to the GENLIB TekSpice library with the **change library** command.

add global node Allows adding global nodes to a selected circuit in the **Circuit Browser**. This command is only available when a circuit is selected.

add internal library Adds an internal library with the specified name into the **Library Browser.**

add libraries at end Adds the selected library(s) name to the end of the library list in the **Library Path Editor**.

add libraries before selected library Adds the selected library name before the selected library name in the **Library Path Editor**.

add local node Adds a local node with the specified name into the *Node List Pane* in a **Circuit Editor** or a **Subcircuit Editor**.

add new Adds a new control program to the **Control Program Browser**. A name is specified and a blank control program with that name is inserted.

add process Adds a new Library process with the specified name into the *Process List Pane* of the **Library Browser**.

add program Adds the specified control program (**ac**, **bias**, **dc**, **tran**, **querc**, **new**) to the **Control Program** browser, from the **Circuit Editor**.

add project Adds a project with the specified name into the **Circuit Browser**.

add querc Adds access to quERC (Electrical Rules Checker) by adding a querc analysis control program to the **Control Program Browser**. This control program defines variables and incorporates a bias analysis that produces simulation results suitable as quERC input. QuERC is automatically invoked by ADS.

add subcircuit Adds a subcircuit with the specified name into the *Library Model List Pane* of the **Library Browser**.

add terminal Adds one or more terminals to the definition of a subcircuit in a *Terminal List Pane*. If more than one is added, they are specified by names delimited by a ”>”.

add tran Adds the default tran control program to the **Control Program Browser**.

adjust **move horizontally.** Moves the selected objects, constraining the movement to be horizontal.

move vertically. Moves the selected objects, constraining the movement to be vertical.

again Matches the internal contents of the text buffer at their next occurrence in the text following the caret (∧) location. Highlights the matched item is and executes any changes previously made to the original buffer contents.

analysis Switches the mode to analysis in a **Circuit** or **Subcircuit Editor**. In this mode, simulation results can be probed.

annotation Switches the mode to annotation in a **Circuit** or **Subcircuit Editor**. In this mode, text and graphics can be added to the circuit.

ansr1 Puts the highlighted value into a variable called **ansr1**, which can be evaluated in an expression in the *Workspace Pane*.

ansr2 Puts the highlighted value into a variable called **ansr2**, which can be evaluated in an expression in the *Workspace Pane*.

arc Allows drawing an arc with three specified points. The first two points are endpoints of the arc and the third point is a point on the arc.

auto scale Automatically scales the minimum and maximum x and y axis values in the **Plot Browser** based on the waveform data. This is the default behavior within the **Plot Browser**.

axis Set the axis to a **linear** or **log** scale.

B

browse If the mouse cursor is on an element instance in the *Schematic Editor Pane* an **Element Browser** is opened. Here the parameters and name of the element instance can be set. If the mouse cursor is on postage stamp plot or in a *Simulation Status Pane*, an **Experiment Browser** is opened.

browse experiment Open an **Experiment Browser** from the **Circuit** or **Subcircuit Editor.** It is used to see the libraries and variables used to obtain those results.

browse waveform Allows changing the color or displayed name for a waveform.

C

cancel

Returns pane contents to the state they were in at the last issue of the accept command.

change default ac

Allows the changing of the default for the ac control program. This changes the default to be used for all new circuits throughout the image. It does not change any existing control programs named ac.

change default bias

Allows the changing of the default for the bias control program. This changes the default to be used for all new circuits throughout the image. It does not change any existing control programs named bias.

change default dc

Allows the changing of the default for the dc control program. This changes the default to be used for all new circuits throughout the image. It does not change any existing control programs named dc.

change default tran

Allows the changing of the default for the tran control program. This changes the default to be used for all new circuits throughout the image. It does not change any existing control programs named tran.

change default querc

Allows the changing of the default for the querc control program. This changes the default used for all new circuits throughout the image. It does not change any existing control programs named querc.

change library

Used to change the name of a library. If it is an external library, the path to the TekSpice file and the file name are also changeable.

circle

Allows drawing a circle with two specified points. The first mouse click defines the location of the origin, the second defines the radius.

circuitmenu

The **circuitmenu** command is not a pop–up button command. However this command, when present in the *Control Program Text Pane*, allows the creation of user defined pop–up menu commands. During the analysis mode these commands are accessible from a *Schematic Editor Pane* using the **functions >** sub menu pop–up button command while not pointing at a wire or element. The **circuitmenu** command arguments can reference information using specified nets

or elements. Waveform functions may appear in the “circuitmenu statements” with the arguments to those functions set to the reserved variables @, @1,... , @n for any integer n. The **wiremenu** command is a subset of the **circuitmenu** command and deals with only net related information such as dc bias voltages, ac and transient voltage waveforms.

The reserved variables @, @1, @2, etc., receive their value from elements or nets specified by mouse selection. The mouse cursor becomes a cross–hair which selects elements until each variable @1, @2, etc. has received a value. The expression is evaluated and the result presented as a postage stamp plot or value tag which are placed with the cursor.

clear Clears information from windows.

all. Removes all analysis results pasted into the *Schematic Editor Pane*.

scratch. Removes analysis results from the scratch overlay layer (sheet).

sheet 1. Removes analysis results from the **sheet1** overlay layer.

sheet 2. Removes analysis results from the **sheet2** overlay layer.

configure In the **Plot** browser this is issued to configure the **Plot** browser’s workspace lines, etc.

configure parasitics Opens a **Parasitic Configuration Dialog Box**. Here the name of the parasitic file can be specified and whether the file should be included as part of the circuit description during simulation.

copy Copies highlighted items to a graphics or text buffer. If the mouse cursor is in a *Simulation Status Pane*, that pane’s active experiment is copied to the text buffer. Items in the graphics or text buffer are available for a paste operation. See **paste**.

copy circuit Copies the selected circuit to a re–named circuit in the same project.

copy experiment Copies the selected experiment to an experiment buffer. The copied experiment may be pasted in a *Simulator Status Pane* of a **Plot** browser or **Workspace** browser to change the context of the window to the copied experiment.

copy library Create a new library in the selected process with a specified name and copy the models in the selected library to the new library.

copy process Create a new process with a specified name and copy the libraries in the selected process to the new process.

copy project Create a new project with a specified name and copy the circuits in the selected project to the new project.

copy to library Copy the selected model into the specified library.

copy to process Copy the selected library into the specified process.

copy to project Copies a circuit to a specified existing project.

create library model Creates a new library model name in the *Model List Pane*.

cut The highlighted item is copied into the text or graphic buffer and deleted from the displayed text or graphics.

D

default parameters When selected, adds the default parameters to the **Element Browser** dialog box text window. The default parameters are predefined for each model.

do it If the highlighted text represents an ADS command, then that command is executed.

E

edit When in the *Edit/Analysis/Annotation Pane*, switches to edit mode, otherwise allows editing the definition of an internal library subcircuit, including its schematic representation. The resulting **Subcircuit Definition Editor** can be used to modify the subcircuit symbol, parameters, or schematic definition. This Editor must be opened from the **Circuit** or **Subcircuit Editor** from the *Model List Pane*.

edit circuit Opens the **Circuit Editor** on the selected circuit**.**

edit circuit variables Allows editing circuit variables from a selected circuit in the **Circuit Browser**. If you edit the right hand sides of assignments to circuit variables, this changes the value of the circuit variable for new control programs only. A variable cannot be defined in terms of another variable. This command is only available when a circuit is selected.

edit control programs Opens the **Control Program Browser** for the selected circuit.

edit parent Opens (or finds) a schematic editor on the parent subcircuit of the active subcircuit. If the active circuit is at the top level, then **edit parent** displays a **User Error Dialog Box**.

edit symbol Allows editing the symbol and associated parameter definitions of the specified library subcircuit This symbol editor can be opened from the **Circuit** or **Subcircuit Editor** from the *Model List Pane*, or from the **Library Browser**.

expand Pushes down one level into the hierarchy of a circuit. When in the **Circuit Editor**, opens a library editor for the apropriate element or subcircuit at the mouse cursor. This element or subcircuit can then be browsed, modified, etc. Changes will affect all circuit instances of this element or subcircuit.

F

file–in Read the selected file as new ADS application code. Use this command to upgrade an ADS image to a new version.

find element Used to locate the specified element (instance) on the schematic. The command causes that element to appear in the *Locate Pane*. Selecting that name causes the element to be selected. The **locate** command can then be used to warp the cursor to that element.

find model Used to find instances or locations of the specified model. In the **Library Browser** all libraries with the given model name are listed. In the **Circuit Editor** *Model List Pane*, the command causes the model to appear in the *Locate Pane*. The **locate** command then causes all instances of that model to appear in the *Locate Pane*, and each can then be selected individually.

find net Used to locate the specified net on the schematic. The command causes that net to appear in the *Locate Pane*. Selecting that name causes the net to be selected. The **locate** command can then be used to warp the cursor to that net.

find node Used to locate the specified node on the schematic. The command causes the node to appear in the *Locate Pane*. The **locate** command then causes all instances of that node to appear in the *Locate Pane*, and each can then be selected individually.

find variable Used to locate the specified circuit variable. The command causes the variable name to appear in the *Locate Pane*. The r**eferences** command then causes all references to that variable to be displayed. If there are no references, the **remove** command will remove the variable.

foreign experiment Used to connect the simulation results files of any simulation to a **Plot Browser** or **Circuit Editor** or **Workspace Browser** – allowing display and manipulation of the results.

functions List a menu of user defined functions as specified in the **Control Program Browser**.

functions i,v Lists a menu of the current and voltage related query functions in the **Circuit Editor**.

functions other Lists a menu of user defined functions specified in the **Control Program Browser**.

G

genlib library Creates a GENLIB (binary) version of the TekSpice file.

get from Allows reading the contents of a file.

global nodes **add global node.** Adds a global node of the specified name to the selected circuit while in the **Circuit Browser**.

remove global node. Opens up a list of all global nodes for the selected circuit, allowing selection for removal while in the **Circuit Browser**. The nodes will only be removed if they are not used in the circuit.

I

info **layout info.** From the **Circuit Editor**, *Edit Pane*, allows viewing possible netlist problems prior to layout, count the number of instances by model name, and

writes a modified netlist with invocation of the program **simtoq8** to prepare the netlist for layout.

library model map. Opens a **Workspace Editor** showing the models used in the circuit shown in the *Schematic Editor Pane* and their associated librarys.

install standard process — Used to install libraries in an ADS image from a list of standard process libraries. This command is accessed from the **Library Browser** *Process List Pane* with no process selected. The resulting dialog boxes allow the selection of a library process and its library(s). The library(s) and models are then read in from their EDIF representation.

invisible

all. Makes all overlay sheets invisible

scratch. Makes the SCRATCH overlay sheet invisible.

sheet 1. Makes the SHEET1 overlay sheet invisible.

sheet 2. Makes the SHEET2 overlay sheet invisible.

L

label — Allows the placement of specified text into the *Waveform Pane* of the **Plot Browser.**

label precision — Used to specify the number of digits to display on the plot axis.

label style — Used to select what text to display on the schematic for a particular model. This can be specified at the model definition level and at the model instance level. Options include no text, model name, selected string, selected parameter(s) value.

layer visibility — Used to specify what layers to display on the schematic.

less tics — When selected in the **Plot Browser**, Approximately half the the number tics are displayed on the plot axis. Operates on the plot axis under the cursor at the time of the command.

library Brings up a dialog box allowing selection of models to be added to the *Model List Pane*.

line Allows creation of a line segment(s) using the mouse select button. Line creation is terminated with the mouse special button.

line style **solid.** Makes the selected waveforms visible.

invisible. Makes the selected waveforms invisible.

locate *Locator Pane*–Expand the model or node to include all instances of the model or node. Subsequent selection of an instance will select it on the schematic.

Model List Pane–Insert the selected model in the *Locator Pane*.

Node List Pane– the following options are available:

external node. Causes all external nodes to be listed in the *Locator Pane*.

global node. Causes all global nodes to be listed in the *Locator Pane*.

local node. Causes all local nodes to be listed in the *Locator Pane*.

M

markers **diamond**, **plus**, **square**, **none** – Places symbols (as specified by diamond, plus, square, none) on the selected waveform at each data point of the independent axis.

minor tics ON(OFF) Indicates to turn ON (OFF) the subdivision of the plot axis into 5 tics between each labeled tic mark. Operates on the plot axis under the cursor at the time of the command.

more tics When selected in the **Plot Browser**, approximately doubles the number of displayed tics on the plot axis. Operates on the plot axis under the cursor at the time of the command.

move experiment to history From the **Control Program Browser**, allows saving the present simulation results (experiment) to a history buffer. Otherwise the results are written over by the next simulation results.

move label Allows moving an elements label to another position. Move label should only be used for short moves, primarily to bring a value label back on the screen that would ordinarily be clipped. Labels should be removed and recreated if they are to be moved long distances.

move to library Allows moving the selected model to a specified library. This model will no longer be in the existing library.

move to process Moves the selected library to a new process and changes all library path circuit references to point to this new process.

move to project Moves a circuit to the prompted project when in the *Circuit List Pane* of the **Circuit Browser**. All states of the circuit and control programs are preserved exactly.

N

name prefix Used to specify the prefix to use to construct the instance name of a model. This prefix has a unique number appended to it for each instance placed.

new axis **x**, **y**, **x and y** – Requests the construction of a new axis in the plot. Selected waveforms change to this new axis. Does not create a new axis if no waveforms are selected.

new experiment Removes the selected experiment and all results files from the ADS image.

O

open **operating point table.** Opens an operating point table showing operating points of the TekSpice primitive models in the circuit.

user operating point table. Opens a user operating point table showing operating points of user models in the circuit.

operating point text. Opens a window showing the results of the operating point analysis for the circuit

smith chart browser. Opens a **Smith Chart Plot Browser**.

overview Increase or decrease the scale in the editor as large as possible and still keep all data visible. Applies in schematic, symbol and plot panes.

P

paste Insert the contents of the text or graphic buffer at the cursor location. The contents of the text buffer replaces the highlighted item. Buffers are filled as a result of a **copy** or **cut** command. If the mouse cursor is in a *Simulation Status Pane*, the last copied experiment becomes the current experiment in that pane. The paste command in the **Smith Chart Browser** operates as follows:

S–parameter. Paste the data in the copy buffer into the **Smith Chart Browser** as an S parameter (S_{11} or S_{22}). This data must be normalized by Z_0 (50 ohm default) and should be in the form Re + j Im (not magnitude and phase).

Y–admittance. Paste the data in the copy buffer into the **Smith Chart Browser** as admittance. This data should be an admittance in the form Re + j Im (not magnitude and phase).

Z–impedance. Paste the data in the copy buffer into the **Smith Chart Browser** as impedance. This data should be an impedance in the form Re + j Im (not magnitude and phase).

plot Opens a **Plot Browser** and insert the waveform of the net under the cursor at the time of the command.

plot it If the highlighted expression evaluates to a waveform generated by simulation, a **Plot Browser** opens and displays the waveform.

pop Go back one window magnification on a waveform or set of waveforms.

print Prints the graphic or textual contents of a pane to a file and then sents it to a printer as specified in the **Print Configuration (Graphics)** dialog box or **Print Configuration (Text)** dialog box.

print all Prints all models for the selected circuit in the *Circuit List Pan*e.

print it When in a **Workspace Editor**, causes evaluation of the highlighted expression and displaying results as highlighted text following the original selected expression.

R

read edif Reads the specified EDIF file into the ADS image. Requires a file name if the file is a circuit or model. Lists the available model names if the file is a library.

rectangle Allows creating a rectangle by specifying the two diagonal points with the mouse select button.

references In the *Locator Pane*, determine what circuits reference the selected model, node or library. In the **Library Browser** *Process List Pane* it brings up a window listing all circuits that reference any library in the selected process.

remove Remove the selected model or node. ADS indicates if the model or node has references in other circuits.

remove circuit Allows removing (deleting) the selected circuit.

remove circuit variables Allows removing circuit variables from a selected circuit in the **Circuit Browser**. This command is only available when a circuit is selected.

remove experiment Allows removing the selected experiment and all the associated results files.

remove global node Allows removing global nodes from a selected circuit in the **Circuit Browser**. This command is only available when a circuit is selected.

remove library Removes the selected library, after verifying the command. Issues a warning if circuits reference this library.

remove process Remove the selected library process and all of its libraries. Issues a warning if circuits reference this process

remove program Remove the selected program and all of its circuits.

remove unused models Removes unused models from the *Model List Pane* (Those not pasted into the schematic). Does not remove models from the library – just the *Model List Pane* of that particular circuit.

remove project Allows removing the selected project.

remove terminal list Allows removal of specified terminals from the *Terminal List Pane*.

remove unused models Allows removing unused models from the *Model List Pane* (Those not pasted into the *Schematic Editor Pane*).

remove unused nodes Allows removing unused nodes from the *Node List Pane* (Those not pasted into the *Schematic Editor Pane*).

rename Changes the name of the selected list item. This item could be a model, internal node, control program or library model.

rename process In the **Library Browser** *Process List Pane*, changes the name of a library process.

rename project In the **Circuit Browse**r *Project List Pane*, changes the name of a circuit project.

resequence

colors. Reassigns the colors for the waveform labels to their default colors from left to right.

names. Reassigns the names for the waveform labels alphabetically from left to right.

reset

instance suffix. Sets the instance suffix value for the specified instance prefix to a new number. All subsequent instances for that instance prefix begin at the new number and increments from there.

instance sequence. Renumbers instance suffixes, by model name, starting at a suffix of 1 and building up from there, by model name. Works for the entire schematic, at one level, with nothing selected. Works on selected instances if any are selected.

net sequence. Renumbers net numbers, starting at 1 and increments from there.

resize Changes the display box size of a table cell.

S

save as.. Saves text in a workspace to the specified file.

select	Returns to the select mode from th wire mode. This command also de–selects all selected items.
select all	Highlights in the *Schematic Editor Pane* all elements and nodes in the *Locator List Pane*.
select control program	Allows the selection of the specified control program from the **Circuit Editor**.
select experiment	Allows the experiment number desired for results display to be specified from the **Circuit editor**.
set axis label	Specifies a new text string label for for the specified axis.
set libraries	Sets the allowed libraries for the selected circuit.
set min/max	In the **Plot Browser** allows specification of a minimum and maximum value for the selected plot axis.
show/hide	Allows control over what columns appear in a table.
show/hide variables	Allows control over what variables will show and not show in a table.
show library text	Opens a window showing the textual representation of all of the models in the GENLIB (binary) TekSpice file for the selected library.
show model text	Opens a window showing the textual representation of the selected model in the GENLIB (binary) TekSpice file for the selected library.
show name	shows the node name of the node pointed to with the mouse. The node text can be placed by pressing the select mouse button.
show net list	Opens a window containing the netlist text for the selected experiment.
simulate	Runs TekSpice simulator on the selected circuit using the selected control program.
sort rows	Rearranges the rows in tables into their default (initial) row order.

swap Swaps the selected model with the specified model. The models must have the same input parameters and symbol port locations.

swap list Allows swapping of node names by name specification – oldNode > newNode

T

text Creates text in a window. Define the upper left corner of the text box using the mouse select button, then type in the text. The enter key defines the right edge of the text box. the return key leaves the right edge of the text box unchanged.

text scale Sets the scale of text in the symbol editor. This is a relative scale.

track From the **Plot Browser** or **Smith Chart Plot Browse**r, puts the mouse cursor into track mode. At this point the cursor is restricted to moving along the displayed waveform and displays the data at the (x,y) location of the cursor.

U

undo Returns the pane contents to the state they were in before the last issued command.

user model Specifies a new model based on a TekSpice primitive model, with the same input and output parameters as the original model. Default parameters values can be the different than the original model.

V

value Returns the location of the mouse cursor while in the *Waveform Pane*. This value is relative to the scale of the data on the plot. For x–y plots the values of x and y are returned. For the Smith Chart plot values are in resistance and impedance multiplied by the normalized impedance.

visible **all.** Makes all overlay sheets visible

scratch. Makes the scratch overlay sheet visible.

sheet 1. Makes the sheet1 overlay sheet visible.

sheet 2. Makes the sheet2 overlay sheet visible.

variables

add circuit variable. Adds a circuit variable by name to the selected circuit.

remove circuit variable. Causes a list of all circuit variables defined for the selected circuit to be displayed and selectively removed. Removal is not allowed if the variable is in use.

W

window

Allow changing the scale of a schematic or plot indicating the area of interest.

wire

Changes the cursor to a crosshair and causes the select button to lay down points connected by wire segments. The wire is terminated by the select button. To exit wire mode, click on the pop–up button menu **select** command.

wiremenu

The **wiremenu** command is not a pop–up button command. However this command, when present in the *Control Program Text Pane*, allows the creation of user defined pop–up menu commands. During the analysis mode these commands are accessible from a *Schematic Editor Pane* using the **functions >** sub menu pop–up button command while pointing at a wire. The **wiremenu** command arguments can reference information using specified nets. The **circuitmenu** command is a superset of the **wiremenu** command and deals with both net and element related information. Waveform functions may appear in the "wiremenu statements" with the arguments to those functions set to the reserved variables @, @1,... , @n for any integer n.

The reserved variable @ receives its voltage value from the wire under the mouse cursor when the pop–up button menu command @ is executed. If additional arguments @1, @2, etc. appear in the selected expression, then the cursor becomes a cross–hair and selects wires until each variable @1, @2, etc. receives a value. The expression is evaluated and the results presented as a postage stamp plot or value tag which are placed on the schematic with the cursor.

write edif

Writes the selected circuit, project or library model to an EDIF file.

write netlist

Writes the netlist of the circuit or subcircuit to a specified file. For a library, write each library model to an EDIF file.

Z

zoom

zoom in. Makes the size of the schematic or plot approximately 2 times larger.

zoom out. Makes the size of the schematic or plot approximately 2 times smaller.

zoom numeric. Makes the size of the schematic or plot the size of the specified number.

nominal. Makes the scale factor equal 1.

nominal elements. Makes the scale factor 1 and make each cell large enough to display the entire cell contents.

center. Pans the schematic to the specified center point.

ADS–TekSpice Reference

Syntax

The syntax for the operations available in the ADS workspace windows are given in this section.

Math Operators

The math operators available from within ADS in Workspace Panes and TekSpice commands are shown in Table 4–1.

Table 4–1: ADS/TekSpice Math Operators

Operator	Description
(	left parenthesis
)	right parenthesis
^	power
*	multiply
/	divide
+	addition
–	subtraction
>	greater than
>=	greater than or equal
<	less than
<=	less than or equal
==	equal
!=	not equal
|	or
&	and
#	exclusive-or
!	unary not

Scale Factors

The syntax used in the ADS windows is given here. The Permitted ADS/TekSpice scale factors are shown in Table 4–2.

Table 4–2: ADS/TekSpice Scale Factors[1]

Name	Abbreviation	Value
general exponent	exx, Exx	10^{xx}
tera	t, T	10^{12}
giga	g, G	10^{9}
mega	meg, Meg, mEg, meG, MEg, MeG, mEG, MEG	10^{6}
kilo	k, K	10^{3}
mili	m	10^{-3}
micro	u, U	10^{-6}
nano	n, N	10^{-9}
pico	p, P	10^{-12}
femto	f, F	10^{-15}
atto	a, A	10^{-18}

1 **Scale factors are case insensitive, except for mili. M is often used as a synonym for MEG. Here it would be comfused with milli.**

Math Functions

The math functions available from within ADS in Workspace Panes and TekSpice commands are shown in Table 4–3. These math functions operate on scalar, array and waveform data.

The definition and the returned values for the math functions are similar to the standard ”C” or FORTRAN functions.

Table 4–3: ADS Math Functions

Function	Input Type[1]	Definition
abs(A)	R or C	Absolute value of A
acos(A)	R or C	Arc cosine of A
acosh(A)	R or C	Arc hyperbolic cosine of A
asin(A)	R or C	Arc sine of A
asinh(A)	R or C	Arc hyperbolic sine of A
atan(A)	R or C	Arc tangent of A
atanh(A)	R or C	Arc hyperbolic tangent of A
atan2(A1,A2)	R or C	Arc tangent of A1/A2
cmplx(A,B)	Real	Complex result A+jB
conjg(A)	R or C	Conjugate of A
cos(A)	R or C	Cosine of A
cosh(A)	R or C	Hyperbolic cosine of A
db(A)	R or C	$20*\log_{10}$(magnitude(A))
equal(A,B)	R or C	One if A=B; zero if $A \neq B$
exp(A)	R or C	e^A
gt(A,B)	Real	One if A>B; zero if $A \leq B$
gte(A,B)	Real	One if $A \geq B$; zero if A<B
imag(A)	R or C	Imaginary part of A
isinf(A)	R or C	One if $A = \pm \infty$
isnan(A)	R or C	One if A=NaN (Not a Number)
log(A)	R or C	Log_e(A)
log10(A)	R or C	Log_{10}(A)
lt(A,B)	Real	One if A<B; zero if $A \geq B$
lte(A,B)	Real	One if $A \leq B$; zero if A>B
mag(A)	R or C	Magnitude of A
max(A1,A2)	Real	Largest value of A1, A2
min(A1,A2)	Real	Smallest value of A1, A2

Table 4–3: ADS Math Functions (Cont.)

Function	Input Type[1]	Definition
mod(A,B)[2]	Real	Remainder of A/B[2]
noteql(A,B)	R or C	One if $A \neq B$; zero if A=B
nrand( )	Real	Generates a random number with normal distribution, zero mean, unity variance
pcmplx(A,B)	Real	Complex results with magnitude A and phase B degrees
phase(A)	R or C	Phase of complex number A
ranset(A)	Real	Sets random number seed
real(A)	R or C	Real part of complex number A
sign(A1,A2)	A1: R or C A2: Real	Transfer sign of A2 to |A1|
sin(A)	R or C	Sine of A
sinh(A)	R or C	Hyperbolic sine of A
sqrt(A)	R or C	$A^{1/2}$ if A>0; NaN if $A \leq 0$
tan(A)	R or C	Tangent of A
tanh(A)	R or C	Hyperbolic tangent of A
trunc(A)	Real	Truncate A to integer
urand( )	Real	Generates a random number with a uniform distribution between zero and one.

1 **Functions noted as R or C accept real or complex inputs.**

2 **MOD(A,B) returns A-X*B where X is the largest integer that does not exceed the magnitude of A/B with sign the same as A. B must be non-zero.**

Array Functions

The built–in array functions available from ADS can handle:

- Declaration of array constants, e.g. {2,3}.
- Array assignments, eg. a={2,3}.
- Declare array dimensions, eg. array a[2] [3].
- Array point access and storage, eg. a[0] [1] =a[1] [2]+1.
- Array arithmetic, eg. sqrt(a), a+1.
- Transpose and dot product matrix arithmetic.
- Set array parameters.

Array indexing is zero based, similar to the C language, in that **a[0]** is the first entry in an array **a** with dimension 1. Higher dimensions are also possible and will be shown in some of the examples.

Array expressions can only be used in the workspace panes of ADS windows. If they were used in analysis blocks or within the circuit block they would be passed to TekSpice2. The present version of TekSpice2 does not handle array expressions.

Array Function Definitions

The array functions available from ADS are defined here.

Table 4–4: Array Functions

Function[1]	Input Type[2]	Definition
append(<*array1*>,<*array2*>) or append(<*array1*>,<*array2*>,<*dimension*>)	R or C array	append to an array
buildarray(<*array1*>,<*array2*>,,,)	R or C array	adds dimensions to array
dependentData(A)	R or C waveform	return data as an array
dot(<*array1*>,<*array2*>)	R or C array	compute dot product of arrays
extractPoint(A,<*string*>)	R or C waveform	select point and return value
independentData(A)	R or C waveform	return data as an array
linearSequence(<*integer*>)	real int	construct array of n integer values
sizes(<*array*>)	R or C array	determine size of an array

Table 4–4: Array Functions (Cont.)

Function[1]	Input Type[2]	Definition
waveform(<*dependentArray*>, <*string*>, <*independentArray.*>..)	real array	construct a waveform from arrays
transpose(<*array*>)	real array	compute transpose of array

1 **A is a waveform or expression, array is an array enclosed by { }, and string is a string enclosed in " ".**

2 **R or C is Real or Complex.**

Array Management

Array Arithmetic

Arrays can be used in place of variables in many of the math functions and waveform functions.

```
a={2,3}
B=sqrt(a)
B=B+1
```

Array Assignment

Arrays can be assigned to variable names:

```
a={2,3}
```

Array Constants

Arrays can be given constant values:

```
{2,3,4}
{{2,3,4},{3,2,1}}
```

Array Declaration

The dimensions of arrays can be declared:

```
array a[3]
array b[2][4]
```

Array Point Access and Store

```
a[0]
b[1][3]
b[0][0]=a[1]
```

APPEND – append two arrays

Syntax

append(<*array1*>,<*array2*>)
or
append(<*array1*>,<*array2*>,<*dimension*>)

Examples

```
;Example1: matrix1 has dimensions [2][2]
; matrix2 has dimensions [2][2]
; resulting array has dimensions [2][4]
matrix1={{1,2},{3,4}}
matrix2={{5,6},{7,8}}
matrix3=append(matrix1,matrix2)
;matrix3={{1,2,5,6},{3,4,7,8}}
;
```

$$\text{matrix1} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\text{matrix2} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$\text{matrix3} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

```
;Example 2:
;matrix4 has dimensions [2][3]
;matrix5 has dimensions [2][1]
matrix4={{1,2,3},{4,5,6}}
matrix5={{7},{8}}
; <dimension> = 1 is default. Here it is the outermost dimension.
matrix6=append(matrix4,matrix5)
;matrix6={{1,2,3,7},{4,5,6,8}}
```

$$\text{matrix4} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\text{matrix5} = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

$$\text{matrix6} = \begin{bmatrix} 1 & 2 & 3 & 7 \\ 4 & 5 & 6 & 8 \end{bmatrix}$$

```
;Example 3: same as previous example but <dimension>=0
;matrix4 has dimensions [2][3]
;matrix7 has dimensions [1][3]
matrix4={{1,2,3},{4,5,6}}
matrix7={{7,8,9}}
```

```
matrix7=append(matrix4,matrix7,0)
;matrix8={{1,2,3},{4,5,6},{7,8,9}}
```

$$\text{matrix4} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$\text{matrix7} = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$

$$\text{matrix8} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}$$

```
;Example 3: Create a pulse by appending n/4 zeroes to the
; beginning and end of a pulse of width n/2 and value of 1.
n=100
; initialize arrays to 0
array ones[n/2] zeros[n/4]
ones=ones+1
pulseArray1=append(zeros, ones)
pulseArray2=append(pulseArray1, zeros)
```

Description

Append appends array <*array2*> to <*array1*> along the given dimension (innermost dimension is the default).

<*array1*> is an n–dimensional array defined as in the standard C language notation. That is an array A[2][3][4] is a 3–dimensional array with two rows (first dimenson) and three columns (second dimension). The third dimension, [4] represents a four high stack of 2x3 arrays. Here the innermost dimension is [4] and the outermost dimension is [2].

<*array2*> is an n–dimensional array which must match the dimension of <*array1*> in the direction of <*dimension*>.

<*dimension*> is an integer that refers to the direction to append <*array2*>. The value of <*dimension*> for the outermost dimension is 0. Default is the innermost dimension. See the examples above.

BUILDARRAY – add one or more dimension to existing arrays

Syntax

buildarray(<*array1*>,<*array2*>,[<*array3*> ...])

Examples

Build a 2X3X3 array from two 3X3 arrays as follows:

```
A=buildarray({{1,0,0},{0,1,0},{0,0,1}},{{2,1,0},{1,2,1},{0,1,2}})
```

Giving the following results:

```
{{1,0,0},{0,1,0},{0,0,1}},{{2,1,0},{1,2,1},{0,1,2}}
```

$$\text{Array1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{Array2} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$\text{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Description

Buildarray has two or more arguments, all of which must be arrays of the same structure. A new array is created which has one more dimension than all the argument arrays with the new dimension outermost in this new array. There is no need to use this to create constant arrays. The examples above uses constant arrays to show the usage.

<array1>, <array2>, <arrayN> are n–dimensional arrays defined as in the standard C language notation. That is an array A[2][3][4] is a 3–dimensional array with two rows (first dimenson) and three columns (second dimension). The third dimension, [4] represents a four high stack of 2x3 arrays. Here the innermost dimension is [4] and the outermost dimension is [2].

DEPENDENTDATA – change the specified data to an array

Syntax

dependentdata(<*argument*>)

Examples

```
; Create a waveform which represents a line with unity slope and
; then extract the dependent data to an array.
n=50
line=waveform(linearsequence(n), "time", linearsequence(n))
yValues=dependentdata(line)

; create a 2-dimensional waveform
Circuit
V1 1 0 dc=dcin
R1 1 2 r:r=1 tc1=.02
R2 2 0 r:r=1 tc1=0
EndC
var sweep TEMP: 27,127,100
dc analysis dcin: 0 10 1
enddc
endvar
; from a workspace window:
OutputV=dependentdata(v(2))
InputV=independentdata(v(2),1)
InputT=independentdata(v(2),0)
; Results:
;OutputV={0,.5,1,1.5,2,2.5,3,3.5,4,4.5,5}
;InputV={0,1,2,3,4,5,6,7,8,9,10}
;InputT={27,127}
```

Description

Return the dependent data of a waveform as an array.

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

DOT – determine the dot product of two arrays

Syntax

dot(<*array1*>,<*array2*>)

Examples

```
A={{4.5,6.2},{143.1,6.1}}
B={{5,2},{8,3}}
dotProduct=dot(A,B)
```

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \cdot \begin{vmatrix} e & f \\ g & h \end{vmatrix} = \begin{vmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{vmatrix}$$

```
; Determine the dot product of A and B
n=50
A=cos(linearsequence(n))
B=sin(linearsequence(n))
dotProduct=dot(A,B)
```

Description

Compute the dot product of <*array1*> and <*array2*>.

<*array1*> is an n–dimensional array defined as in the standard C language notation. That is an array A[2][3][4] is a 3–dimensional array with two rows (first dimenson) and three columns (second dimension). The third dimension, [4] represents a four high stack of 2x3 arrays. Here the innermost dimension is [4] and the outermost dimension is [2].

<*array2*> is an n–dimensional array which must match the dimension of <*array1*>.

EXTRACTPOINT – extract a point from a waveform

Syntax

extractpoint(<*argument*>,”<*string*>”)

Examples

```
probe waveform=waveform
point=extractpoint(waveform,"Select initial tdr corner.")
x=point[0]
y=point[1]
```

Description

Extractpoint extracts an array or point from a waveform. It also provides support for ADS plot interaction for array selection from an ADS script.

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

<*string*> is a text array which contain a group of alpha–numeric characters enclosed by double quotes (”). This string appears in the plot as instructions to the user.

INDEPENDENTDATA – return data as an array

Syntax

independentdata(<*argument*>)
or
independentdata(<*argument*>,<*dimension*>)

Examples

```
; Create a waveform which represents a line with unity slope.
; Then extract the independent data to an array
n=50
line=waveform(linearsequence(n),"time", linearsequence(n))
xValues=independentdata(line,0) ; Note dimensions start at 0

; create a 2-dimensional waveform
Circuit
V1 1 0 dc=dcin
R1 1 2 r:r=1 tc1=.02
R2 2 0 r:r=1 tc1=0
EndC
var sweep TEMP: 27,127,100
dc analysis dcin: 0 10 1
enddc
endvar
; from a workspace window:
OutputV=dependentdata(v(2))
InputV=independentdata(v(2),1)
InputT=independentdata(v(2),0)
; Results:
;OutputV={0,.5,1,1.5,2,2.5,3,3.5,4,4.5,5}
;InputV={0,1,2,3,4,5,6,7,8,9,10}
;InputT={27,127}
```

Description

Return the independent values of <*expression*> as an array. For multi–dimensional waveforms, <*dimension*> determines which axis is used to form the array. See also **dependentdata.**

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

<*dimension*> is an integer that refers to the direction of the array. The value of <*dimension*> for the outermost dimension is 0. Default is the innermost dimension. See the examples above.

LINEARSEQUENCE – construct array of n integer values

Syntax

linearSequence(<*n*>)

Examples

```
n=5
A=linearsequence(n)
; results:
; A={0,1,2,3,4}
```

```
; Generate a sine wave from -2pi to 2pi.
n=100        ; set resolution of waveform
p=n/(4*pi)   ; scale factor
a=append(-n/(2*p)+(1/p)*linearsequence(n/2),
      (1/p)*linearsequence(n/2))
b=sin(a)
sinWave=waveform(b,"radians", a)
```

Description

Linearsequence creates an array of <*n* > integer values starting at 0 and ending with <n>–1. Arithmetic progressions are generated by scaling and shifting. Geometric progression are generated using the arithmetic progression as an exponent.

<*n*> is a positive valued integer specifying the length of the linear sequence.

SIZES – determine the size and dimension of an array

Syntax

sizes(<*array*>)

Examples

```
A={{3,5},{6.2,9.71},{100,187}}
X=sizes(A)
;results: X={3,2}
Y=sizes(A[0])
;results: Y={3}

;Create family of curves using sizes and multidimensional
; arrays.
n=100        ; set precision of generated waveform
p=n/(4*pi)   ; determine scale factor
a=append(-n/(2*p)+(1/p)*linearsequence(n/2),
  (1/p)*linearsequence(n/2))
b=sin(a)     ; create sine wave
c=sin(a+pi/2)+2 ; shift sine up 2 units and delay by pi/2
Size=sizes(a)
firstSize=Size[0]
array dFamily[2][firstSize]
dFamily[0]=b
dFamily[1]=c
family=waveform(dFamily,"time",{-8, 8},"frequency", a)
; For results see Figure 4-1.
```

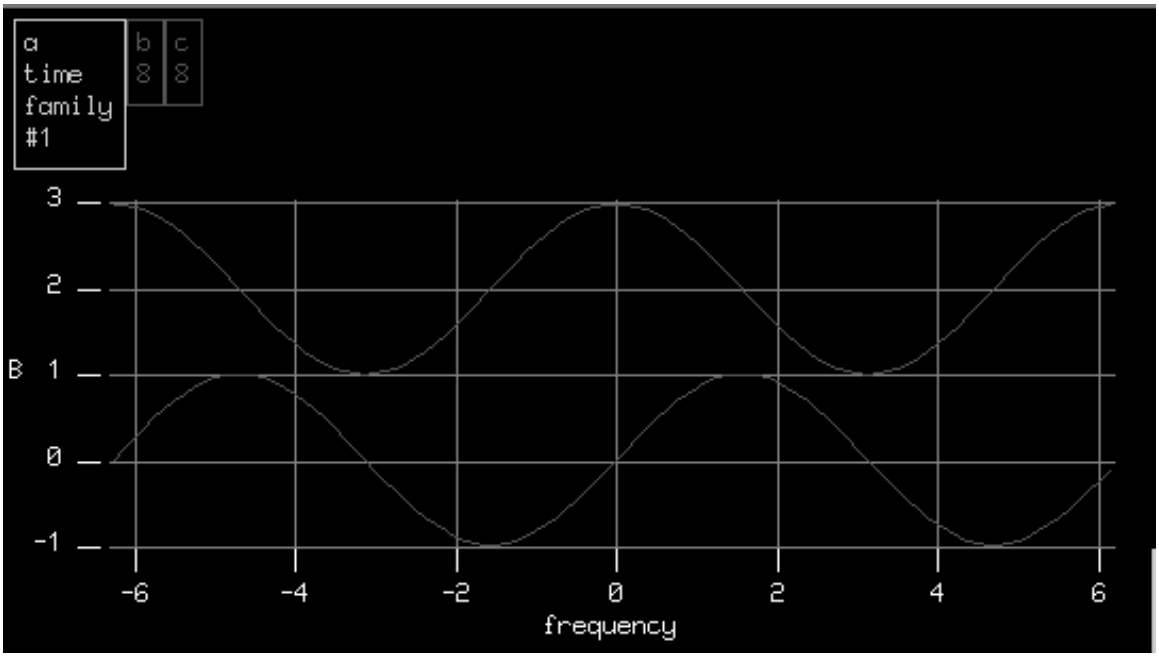


Figure 4–1: Example using the Sizes Function

Description

Returns an array containing the size of each dimension of <array>.

<*array*> is an n–dimensional array defined as in the standard ”C” notation. For example an array A[2][3][4] is a 3–dimensional array with two rows (first dimenson) and three columns (second dimension). The third dimension, [4] represents a four high stack of 2x3 arrays. Here the innermost dimension is [4] and the outermost dimension is [2].

TRANSPOSE – transpose an array

Syntax

transpose(<*array*>)

Examples

```
A={{1,2,3,4},{5,6,7,8}}
B=transpose(A)
; results: B={{1,5},{2,6},{3,7},{4,8}}
```

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix}$$

Description

Create an array which has the rows and columns of <array> transposed (That is: an m x n array becomes an n x m array.)

<*array*> is an n–dimensional array defined as in the standard C programming language notation. For example an array A[2][3][4] is a 3–dimensional array with two rows (first dimenson) and three columns (second dimension). The third dimension, [4] represents a four high stack of 2x3 arrays. Here the innermost dimension is [4] and the outermost dimension is [2].

WAVEFORM – construct a waveform from two arrays

Syntax

waveform(<*dependentArray*>,"<*string1*>",<*independentArray1*> [,"<*string2*>",<*independentArray2*> ...])

Examples

```
indep={0,3,6,9}
dep={0,2,4,6}
plot waveform(dep,"input",indep)
```

Description

Waveform constructs a waveform from a specified dependent data array <*dependentArray*> and an independent data array <*independentArray*>. See also **independentdata** and **dependentdata**.

<*dependentArray*> is an n–dimensional array of dependent data defined as in the standard C language notation. For example an array A[1][3] is a 2–dimensional array with one row (first dimenson) and three columns (second dimension). Here the innermost dimension is [3] and the outermost dimension is [1]. <*dependentArray*> must be of the same size as <*independentArray*>.

<*string*> is a string of alpha numeric characters enclosed in double quotes ("") that will appear as the horizontal axis label of a plot.

<*independentArray*> is an n–dimensional array of independent data. IndependentArray must be of the same size as <*dependentArray*>.

Waveform Functions

The Waveform functions defined in this section are for processing waveforms (trace variable data) produced by sweep analysis in TekSpice or data generated from external sources. Included are functions to:

- Process analysis results, including time– and frequency–domain transformations
- Measure characteristics of waveforms
- Perform matrix operations
- Generate random numbers

The inputs to these functions are waveforms (trace variables). An example waveform is a time varying node voltage produced by a transient analysis. The **probe** command allows using waveform data generated by other simulators, measurment equipment, or computed waveforms saved in a file.

Waveforms contain X–Y data pairs: when generated by sweep analysis, the Y values are the dependent variable representing a node voltage or branch current computed point–by–point during a sweep analysis, and the X values are the independent variable, DCIN, FREQ, or TIME.

Functions can appear in expressions for plotting, printing or assigning to other variables as the right–hand side of assignment statements. Examples are:

```
print tmax(v(1)) ; prints largest value of voltage at node 1
x1=cross(i0,2,3,1);second location where waveform i0 equals 3
```

Expressions which include waveforms are allowed wherever waveforms are allowed as input to a function. These expressions can also include constants, operators, other functions, and other variables. Nesting functions within functions are also allowed as shown in this example:

```
print peak(mag(fft(v(11),per))) ;print amplitude and
frequency of largest signal
```

For more discussion of expressions, operators, variables, etc., see the *Syntax Details* in the TekSpice Reference Manual.

Dimension of Function Inputs and Outputs

The waveform functions operate on waveforms (trace variables) of dimension one to n, operating on individual waveforms within a family of waveforms. Waveform measurement functions that return a single value from a waveform,

such as **overs** (overshoot) or **mean**, reduce the dimension of the output by one compared to the input. This is the same effect as that of the **reduce** function. Other functions, such as **int** (integrate) and **impls** (impulse response) return waveforms with the same dimension as the input.

If a waveform computation cannot be done, NaN (not a number) is returned.

Nesting ac, dc, or transient analysis inside two variable sweeps produces a family of curves of dimension three. A waveform function that returns one value per waveform returns a two–dimensional matrix of *m* waveforms of *n* points when applied to a three–dimensional waveform, which is a series of *m* x *n* matrices containing *m* waveforms of *n* points. Plotting the output of this function displays a family of curves.

For example, the analysis commands shown here operate on a filter circuit to produce a three–dimensional family of curves; plotting **overs** produces a two–dimensional family of curves:

```
var sweep cap : 1p 16p 5p
var sweep lvar : 5n 45n 10n

tran analysis time : 0 15n .1n limpts=3100
endtran

endvar
endvar
autoprobe=on
plot overs(v(6))
```

A two–dimensional family of curves like the one produced by the **overs** function in the example above are generated directly by: 1) a sweep analysis (ac, dc, or transient) nested within a variable sweep or 2) a bias analysis nested within two variable sweeps. Such a waveform is a *m* x *n* matrix that is plotted as *m* waveforms of *n* points. When applied to such a waveform, a waveform measurement function returns either a one–dimensional matrix *n* or a single waveform with *m* points. Taking a single family of curves for a given value of the variable ”cap” and applying the overshoot function produces a single waveform:

```
cap=11p
v6_11p=reduce(v(6),cap)
plot overs(v6_11p)
```

A single analysis sweep produces one dimensional waveforms. The waveform measurement functions return scalar values (single points) when applied to waveform produced by ac, dc, or transient analysis or bias analysis nested within a variable sweep. For example, taking a single waveform from the two–dimensional family of curves produced by the reduce function example above and applying the overshoot function produces a scalar:

```
lvar=25n
v6_11p_25n=reduce(v6_11p,lvar)
plot v6_11p_25n
scalar = overs(v6_11p_25n)
```

Fourier Transform Functions

The **fft**, **ift**, **conv**, **step**, **impulse**, and **correl** functions are based on the Split–radix **fft** algorithm. The algorithm is a “fast” method of computing the Discrete Fourier Transform, and is subject to the same constraints. The user is referred to the following references:

[1] E. Oran Brigham, **The Fast Fourier Transform**, Prentice–Hall, Inc., Englewood Cliffs, New Jersey; 1974.

[2] Ronald N. Bracewell, **The Fourier Transform and its Applications**, McGraw–Hill Book Company, New York, New York; 1978.

[3] Fredrick J. Harris, **On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform**, Proceedings of the IEEE, Vol. 66, No. 1, pp. 51–83; Jan. 1978.

[4] The **ift**, **step**, and **impls** functions should be applied only to stable systems for valid results. A function for small–signal stability testing, **minphs**, allows stability of the circuit to be determined from the ac frequency response.

Pulse Characteristics

Terms for pulse characteristics that appear in measurement function descriptions are defined by the following standards:

IEEE Standard Pulse Terms and Definitions, IEEE Std. 194–1977, Institute of Electrical and Electronics Engineers, Inc., 1977, New York, N.Y.

IEEE Standard on Pulse Measurement and Analysis, IEEE Std. 181–1977, Institute of Electrical and Electronics Engineers, Inc., 1977, New York, N.Y.

Several pulse functions take their names from standard terms identified in Figure 4–2.

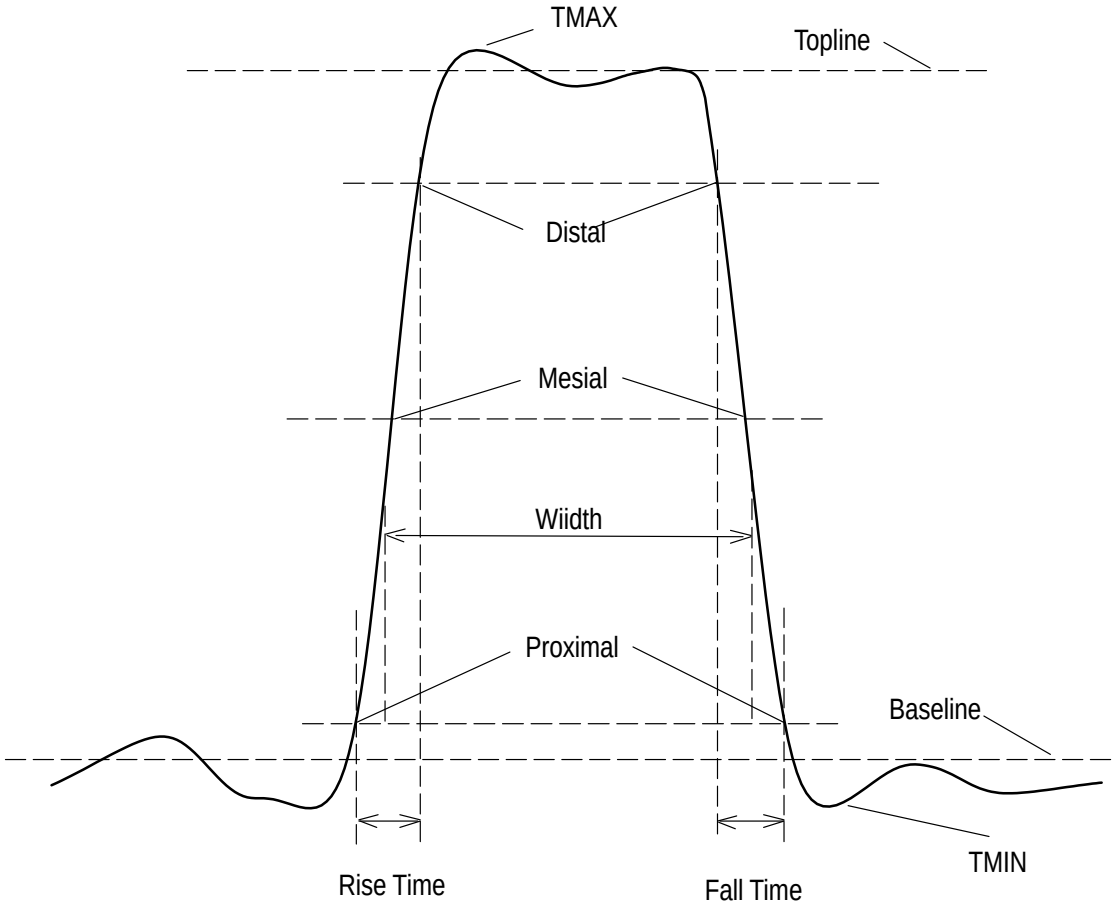


Figure 4–2: Pulse Characteristics as Used in TekSpice Functions.

Frequency–Count Histogram

Some waveform functions compute a frequency–count histogram. This histogram is based on bins that are evenly spaced between the minimum and maximum value of the input expression (waveform). As the input expression is evaluated point–by–point, a count of the number of data points in each bin is maintained. The result is a waveform with X (independent variable) values determined by the number of bins and Y (dependent variable) values determined by the number of points in each bin.

Waveforms that depart radically from that shown in Figure 4–2 may not produce a valid histogram. Check the value of baseline, topline, etc to verify.

Waveform Function Definitions

The waveform functions defined here all require waveforms as inputs. The waveforms are produced from analyses (ac, dc or transient analysis). Some of these functions also accept complex variables as input such as produced by an ac analysis or **fft** operations. Complex waveforms can be used with real–valued functions by first applying the **mag**, **phase**, **real**, or **imag** functions to obtain a real–valued waveform. See also the *Math Function*s section earlier in this chapter. Many of the Math Functions also work on waveform functions.

Table 4–5: Waveform Functions

Function[1]	Input Type[2]	Definition
bandpass(A)	R or C[3]	Width of passband of A
baseline(A)	Real	Baseline value of A
bmatrix(*m*,*n*,*v*)[4]	R or C	Creates *m*x*n* matrix
conv(A,B)	R or C	Convolution of A with B
correl(A,B)	R or C	Correlation of A with B
cross(A)	Real	X–value of A at crossing
dbo(A)	R or C	Db per octave of A
dif(A)	R or C	Derivative of A
distal(A)	Real	Distal value of A
dmatrix(A)[4]	Real	Determinant of *n*x*n* matrix A
duty(A)	Real	Percent duty cycle of A
edgedelay(A)	Real	Delay between specified edges from A
extract(A,start,stop)	R or C	Extract points from A
fall(A)	Real	Fall time of A
fft(A,type)	R or C	Fast–Fourier Transform of A
fmatrix(A,*m*,*v*)[4]	R or C	Fill row *m* of matrix A with waveform
fmatrix(A,*m*,*n*,*v*)[4]	R or C	Fill position *m*,*n* of matrix A with scalar
fold(A)	R or C	Fold A to be asymmetric
freq(A)	Real	Frequency (reciprocal of one period) of A
highpass(A)	R or C[3]	Highpass (crossing on rising edge) of A
histogm(A)	Real	Histogram of A
ift(A,type)	R or C	Inverse FFT of A
imatrix(A)[4]	R or C	Inverse of *n*x*n* matrix A
impls(A)	R or C	Impulse response of A
int(A)	R or C	Integral of A

Table 4–5: Waveform Functions (Cont.)

Function[1]	Input Type[2]	Definition
interp(A,num)	R or C	Linear interpolation of A
logterp(A,num)	R or C	Log to linear interpolation of A
lowpass(A)	R or C[3]	Lowpass (crossing on falling edge) of A
mean(A)	R or C	Mean value of A
merge(A,B)	Real	M erge A and B
mesial(A)	Real	Mesial value of A
minphs(A)	Complex	Minimum–phase shift function of A
mmatrix(A,B)[4]	R or C	Matrix multiplication of A and B
nrand(A)	R or C	Generate random number(s), normal distribution
overs(A)	Real	Overshoot of A in percent
peak(A)	R or C[3]	Location (X) and magnitude (Y) of peak as complex number
period(A)	Real	Period of A measured between appropriate edges
proximal(A)	Real	Proximal value of A
reduce(A,*v*)	R or C	Row or column of matrix A, reducing by *v*
reduceInterp(A,*v*)	R or C	Row or column of matrix A, reducing by *v* with interploation
repeat(A,*n*)	R or C	Repeat A *n* times
rise(A)	Real	Rise time of A
rms(A)	R or C	Root–mean–square of A
scalar(A,x)	R or C	Value of A at x
settle(A)	Real	Settling time of A
slew(A)	Real	Slew rate of A
smatrix(A,b)[4]	R or C	Ax=b or xA=b
smooth(A,n)	R or C	Smooth A, n times
spline(A,*n*)	Real	Spline interpolation of A
step(A)	R or C	Step response of A
stopband(A)	R or C[3]	Width of stopband of A
sum(A)	R or C	Running sum of A
tmatrix(A)[4]	R or C	Transpose matrix A
tmax(A)	Real	Maximum value of A
tmin(A)	Real	Minimum value of A

Table 4–5: Waveform Functions (Cont.)

Function[1]	Input Type[2]	Definition
topline(A)	Real	Topline value of A
unders(A)	Real	Undershoot of A in percent
unfold(A)	R or C	Unfold A to be quasi–symmetric
urand(A)	R or C	Generate random number(s), uniform distribution.
versus(A,B)	Real	Creates waveform of B(x–axis) vs. A(y–axis)
window(A,type,num)	R or C	Window A
xref(A,s)	R or C	Shift A on X axis

1 Form shown is minimum required input, defaults are not shown. A and B are waveforms or matrices and can be expressions.

2 Functions noted as "R or C" accept real or complex inputs.

3 Function operates on the magnitude of the input.

4 Obsolete Function. To be replaced at a later date with Array Functions.

(This page intentionally left blank)

BANDPASS – determine the bandpass of a waveform

Syntax

bandpass (<*expression*> [,<*down*> [,<*type*> [,<*reference*>]]])

Examples

```
print bandpass (v(11)) ; bandpass determined by
;crossing 3 dB below topline
```

```
plot bandpass (v(1),6,0,tmax(mag(v(1)))); use reference as top of
;waveform, bandpass is determined by crossing 6dB below reference
```

Description

The **bandpass** function determines the width (X2 – X1) between the initial rising edge and the final falling edge of the magnitude for <*expression*>. See Figure 4–3. The width is measured between the cross points where the magnitude of <*expression*> matches the cross level established by <*down*> and <*reference*>. The cross points are interpolated linearly if they fall between the data points of <*expression*>. See also **stopband**, **lowpass** and **highpass**.

<*expression*> is an expression based on a real– or complex–valued one dimensional (single waveform) or of two or more dimensional (family of curves) waveform. An error is reported if the magnitude of the first or last data point of the expression is above the cross level; this applies to each waveform in a family of curves.

<*down*> determines the cross level according to <*type*> referred to <*reference*>. If both <*down*> and <*type*> are omitted, the cross level defaults to 3 dB below <*reference*>.

<*type*> determines how <*down*> and <*reference*> are evaluated. If <*type*> is zero, they are evaluated as dB below <*reference*>. If type is 1, then they are evaluated as units below <*reference*>. Default is 0.

<*reference*> defaults to topline, but can be entered as a number or expression. The value of topline is computed as in the **topline** function with a histogram of 1001 bins.

Requires an input pulse with similar characteristics to that shown in Figure 4–2

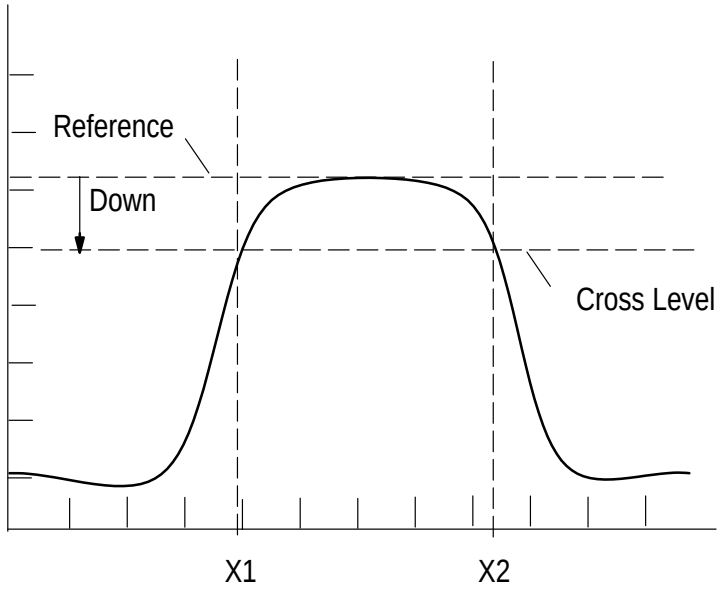


Figure 4–3: BANDPASS function Showing Cross Level

BASELINE – determine the baseline of a waveform

Syntax

baseline (<*expression*>[,<*#bins*>])

Examples

```
print baseline (v(1)-v(2));voltage at nodes 1 & 2 are real
; valued

stopfloor=baseline(mag(acv)); acv is ac response of passband
; filter

baseout=baseline(pulseo,100); pulseo is transient pulse response
```

Description

The **baseline** function returns the value of the baseline for <*expression*>.

<*expression*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of curves) waveform.

Baseline is determined by constructing a histogram with evenly spaced bins between <*expression*>'s minimum and maximum independent variable values. The number of points landing in each bin from <*expression*> are counted. The bin with the largest count for bins below the midpoint of <*expression*> is returned as **baseline**. In case of a tie (more than one bin below the midpoint contains the largest count), the lower bin is returned.

<*#bins*> specifies the number of bins (resolution) used in determining **baseline**. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

BMATRIX – build a matrix with analyses and analysis points (obsolete)*

Syntax

bmatrix (<*R*>,<*C*>,<*v1*> [,<*v2*> [,<*v3*> [, ...]]])

Examples

```
m1=bmatrix (3,4,0,1,2,3,4,5,6,7,8) ;3 X 4 real matrix
; initialized to:
 ; 0 1 2 3
 ; 4 5 6 7
 ; 8 0 0 0

x3=bmatrix (5,1,v1); one-column matrix filled from v1,which
; contains five values

n1=bmatrix (1,65,cmplx(2,3)) ; 65-point complex waveform,first
; point set to (2,3), all other points set to zero
```

Description

The **bmatrix** function builds a one or two–dimensional matrix that represents waveform. Row values can be defined point–by–point or as existing waveforms. Column values are units, 1 through <*C*>.

<*R*> is the number of rows in **bmatrix**; it corresponds to the number of analyses (ac, dc, or transient) performed in a variable sweep analysis. <*R*> must be an integer greater than zero.

<*C*> is the number of columns in **bmatrix**; it corresponds to the number of analysis points in an ac, dc, or transient analysis. <*C*> must be entered as a positive integer.

<*v1*>, <*v2*>, <*v3*>, ... are numbers, variables, or expressions that are scalars; one–dimensional (single waveforms); or two–dimensional (family of curves) waveforms. The matrix is filled with values row–by–row, left–to–right across each row. At least one value is required; no more than <*R*> x <*C*> values are allowed. Matrix locations not specified are set to zero.

Bmatrix is complex if any values are complex or real if all values are real (real values are stored as complex with an imaginary part of zero).

* **To be replaced at a later date with Array Functions.**

CONV – compute the convolution of two waveforms

Syntax

conv(<*expression1*>,<*expression2*>)

Examples

```
plot conv(v(1),v(2))
```

```
result=conv(vd1,v(2))
```

Description

The **conv** function returns a waveform that is the convolution of two waveforms.

<*expression1*> and <*expression2*> are aperiodic expressions based on a real–valued or complex–valued n–dimensional waveform. The independent variable values must be uniformly spaced, either from a linear sweep or from application of the **logterp** function. For multi–waveforms (families of curves), these constraints apply to the inner sweep. Also, for multi–waveforms, the number of points of the inner sweep must be even; and the waveforms must be of the same dimension (same number of sweeps as well as same number of points per sweep) or one must be a single sweep.

CORREL – compute the correlation of two waveforms

Syntax

correl(<*expression1*>,<*expression2*>)

Examples

```
plot correl(v(1),v(2))
```

```
cross=correl(v(1),v(2))
```

Description

The function **correl** returns a waveform that is the correlation of two waveforms. Cross correlation is performed if the two input variables are different; auto–correlation is performed if the two input variables are identical.

<*expression1*> and <*expression2*> are expressions based on a real–valued or complex–valued n–dimensional waveform. Use of **correl** is subject to the same constraints as for the **conv** function, but the number of points for each waveform must be the same.

CROSS – determine intercept of a waveform and a cross point

Syntax

cross(<*expression*> [,<*cross#*> [,<*m*> [,<*type*> [,<*#bins*>]]]] *)*

Examples

```
print cross(v(2),1); first crossing through mesial

x=cross(v(5),1,0,1); first crossing through zero
```

Description

The **cross** function returns a value determined by the intercept of the independent waveform (e.g. time) and the specified cross level <*m*>. Linear interpolation is used if the crossing occurs between two analysis points. Cross equires an input waveform with characteristics similar to that in Figure 4–2.

<*expression*> an expression based on a real–valued n–dimensional waveform.

<*cross#*> – an integer which specifies the crossing number of the specified cross level <m>. **Cross** scans the waveform from left to right, returning the value at the first cross point if <*cross#*> is one, the value at the second cross point if <*cross#*> is two, etc. If <*cross#*> is zero the last cross point is returned. The default for cross# is zero.

<*m*> determines the cross level according to <*type*>. Default is 50% which is halfway between topline and baseline.

<type> a flag which determines how the cross level, <m> is evaluated. If <*type*> is zero, the cross level is <*m*> percent as given by the equation:

cross level = baseline + (topline – baseline)*<*m*>/100.

If <*type*>=1, then cross level is an actual value. Default is 0.

<*#bins*> used only if <*type*> is 1. Selects the number of bins for the histogram used to find topline and baseline. The default is 1001.

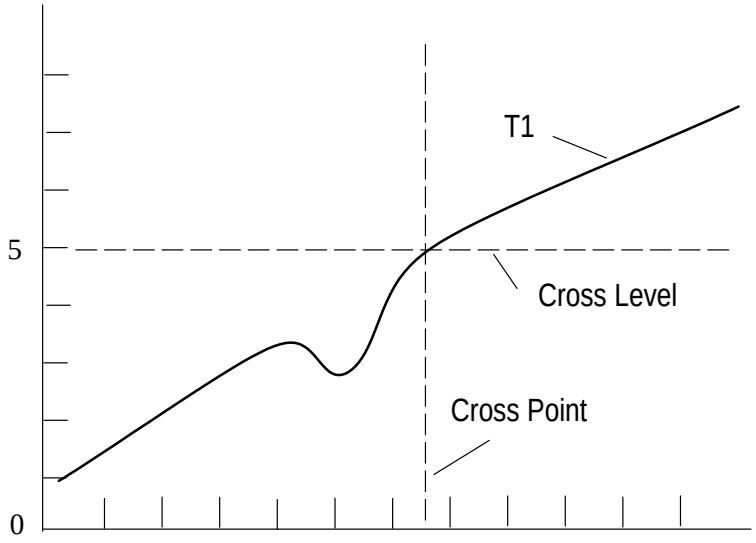


Figure 4–4: Example of CROSS function Showing CROSS(T1,1,5,1)

DBO – compute the decibels per octave of a waveform

Syntax **dbo**(<*expression*>)

Examples

```
plot dbo(v(1))
```

Description The function **dbo** returns a waveform that is the decibels per octave of the input waveform or <expression>.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

DELAY – determine difference in crosspoints of two waveforms

Syntax

delay*(<expression1>,<expression2>*[,*<m1>*[,*<m2>*[,*<type>*[,*<#bins>*]]]])*

Examples

```
print delay(v(5),v(1))
```

Description

The **delay** function returns a value determined by the difference between the cross points of two independent waveforms. In the case of a transient analysis this would be delay time. Delay is measured from the first cross point of <expression1>. From this point, **delay** scans <expression2> from left–to–right for the first cross point . If the scan fails to find a cross point, then the second waveform is searched right–to–left from the same starting point, and a negative value is returned. See Figure 4–5.

Delay requires an input waveform with characteristics similar to that in Figure 4–2.

Reference levels can be set independently for the two waveforms. Cross points are linearly interpolated if a reference level falls between analysis points.

<expression1> is an expression based on a real–valued n–dimensional waveform. *<Expression1>* is the reference waveform for the start of the delay measurement.

<expression2> is an expression based on a real–valued waveform. It must have the same number of dimensions as <expression1>, but is not required to have the same start, stop, or number of analysis points.

<m1> is the reference cross level on *<expression1>* dependent on the value of *<type>*. The default is 50 (percent), halfway from the baseline to topline.

<m2> is the reference cross point on *<expression2>* dependent on the value of *<type>*. The default is the same as *<m1>*.

<type> determines how the cross levels <m1> and <m2> are evaluated. If *<type>* is zero, the default, the cross level is set by *<m1>or <m2>* as given by the equation:

```
cross#level = baseline + (topline - baseline)*<m>/100.
```

If *<type>*=1, then they are evaluated as actual values. The default is for type is zero.

<#bins> sets the number of histogram bins used by **delay** to determine the topline and baseline. The default is 1001.

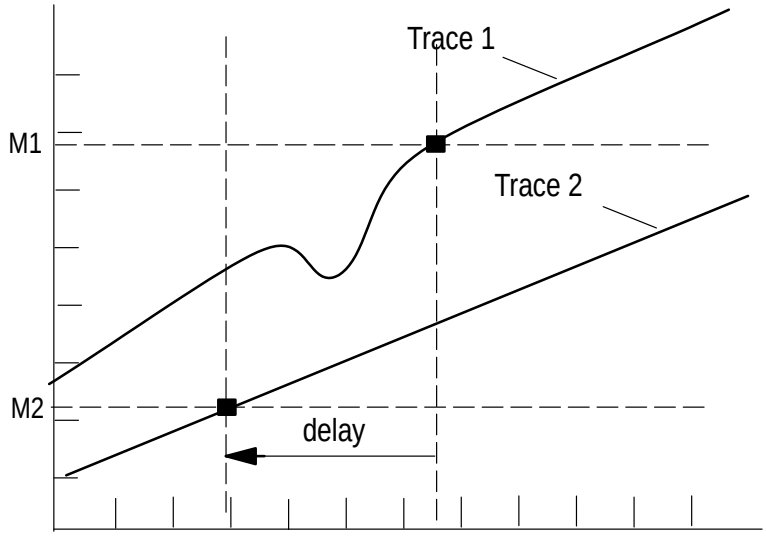


Figure 4–5: Delay Between Reference level Crossings of Two Waveforms.

DIF – differentiate a waveform

Syntax

dif (<*expression*>)

Examples

```
plot dif(v(1))
```

```
slope=dif(v(2))
```

Description

The **dif** function performs differentiation on <*expression*>. The derivative is approximated by second order central differencing for all points except the first and last, which are determined by second order right and left differences respectively.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

DISTAL – determine the distal point of a waveform

Syntax

distal (<*expression*> [,<*d*> [,<*#bins*>]])

Examples

```
print distal(v(3)); first crossing at 90%
```

Description

The **distal** function computes the dependant value of the waveform defined by <*expression*> as:

```
baseline + (topline - baseline)*<d>/100
```

The topline and baseline are found by a frequency–count histogram as described for the **topline** and **baseline** functions. See also the **proximal** function.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*d*> sets the desired distal level in percent. The default is 90 percent. Values less than zero and greater than 100 are permitted; large undershoot or overshoot may make this desirable.

<*#bins*> selects the number of bins **distal** uses for the histogram. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

DMATRIX – compute the determinant of a matrix (obsolete)*

Syntax **dmatrix** (<*expression*>)

Examples

```
print dmatrix(zmax)
```

Description The **dmatrix** function returns the determinant of a matrix. The result is a scalar value.

<*expression*> is a non–singular n *x* n matrix.

* To be replaced at a later date with Array Functions.

DUTY – determine percentage of points above a reference level

Syntax

duty (<*expression*> [,<m> [,<type> [,<#bins>]]])

Examples

```
print duty(v(11))
```

```
apow=vpk^2/rvari*duty(extract(v(1), cross(v(1),1),cross(v(1),3)))
; adjust peak power for duty cycle of single period
```

Description

The **duty** function returns a value determined by the percent of analysis points at or above the reference point. Other functions can be used to apply the **duty** function to an exact period as shown in the example.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*m*> sets the reference level in percent. The default is 50 (percent), halfway between the topline and baseline.

<*type*> determines how <*m*> is evaluated. If <*type*> = 0, <*m*> is evaluated as percent of topline – baseline. If <*type*> = 1, then <*m*> is the actual reference level. Default is zero.

<*#bins*> selects the number of histogram bins used by **duty** to determine topline and baseline The default is 1001.

EDGEDELAY – computes delay between specified edges of two waveforms

Syntax

edgedelay(*<expression1>,<expression2>,<edge1>, <edge2>[, m1 [, m2 [, type [,bins]]]]*)

Examples

```
edgedelay(v(11),v(5),1,1)
edgedelay(v(21),v(2),2,3,25,75,0,251)
```

Description

Edgedelay returns a value determined by the difference in x–axis values between the *<edge1>* edge of *<expression1>* and the *<edge2>* edge of *<expression2>*. For a transient analysis, this difference is delay time. See Figure 4–6.

Both rising and falling edges are counted in the value of *<edge1>* or *<edge2>*. *<expression1>* and *<expression2>* cannot be scalars, they must be waveforms, but can have any number of dimensions. If the waveforms are more than one–dimensional, *<expression1>* and *<expression2>* must have the same number of curves in each family. The results of edgedelay can be negative.

The value of **edgedelay** from two one–dimensional waveforms is a single number, from two two–dimensional waveforms the result is a one dimensional waveform, and so on.

<expression1> is an expression based on a real–valued n–dimensional waveform. *<Expression1>* is the reference waveform for the start of the delay measurement.

<expression2> is an expression based on a real–valued waveform. It must have the same number of dimensions as <expression1>, but is not required to have the same start, stop, or number of analysis points.

<edge1> is an integer specifying the number of rising and falling edges in <expression1> at which the *<m1>* reference level is applied. The direction for determining edge, goes from the first data point to the last data point of *<expression1>*.

<edge2> is an integer specifying the number of rising and falling edges in *<expression2>* at which the *<m2>* reference level is applied. The direction for determining edge, goes from the first data point to the last data point of *<expression2>*.

<m1> is the reference cross point for *<expression1>* dependent on the value of *<type>*. The default is 50 (percent), halfway from the **baseline** to **topline**.

<m2> is the reference cross point for *<expression2>* dependent on the value of *<type>*. The default is 50 (percent), halfway from the **baseline** to **topline**.

<*type*> determines how the cross levels <m1> and <m2> are evaluated. If <*type*> is zero, the default, the cross level is set above the baseline by <*m1*>*or* <*m2*> percent of topline minus baseline**:**

```
cross#level = baseline + (topline - baseline)*<m>/100.
```

If <*type*>=1, then they are evaluated as actual values. Default is 0.

<*#bins*> sets the number of histogram bins used by **edgedelay** to determine the topline and baseline. The default is 1001.

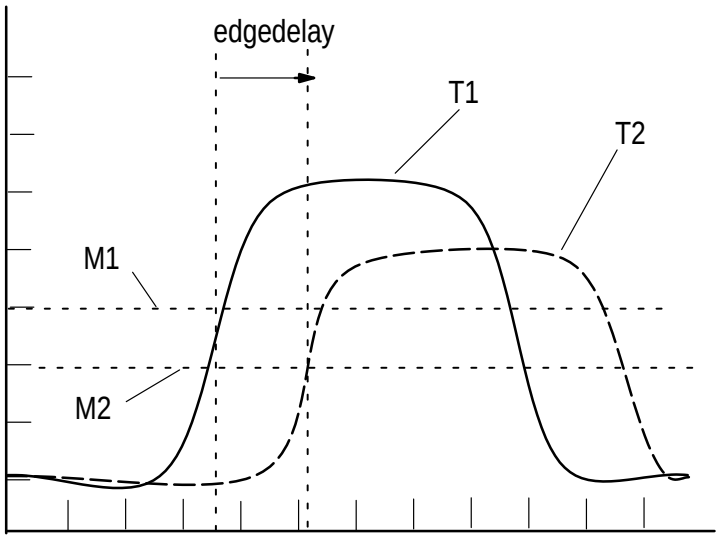


Figure 4–6: Example Showing EDGEDELAY(T1,T2,1,1,M1,M2,1)

EXTRACT – create a subset of the original waveform

Syntax

extract (<*expression*>,<*xfrom*>,<*xto*>)

Examples

```
tran analysis time:0 100n 1n
endtran

; v(1) x-axis limits are 0 to 100n
newv1=extract(v(1),50n,100n);newv1 x-axis limits are 50n to 100n
```

Description

The **extract** function creates a waveform that is a subset of the original waveform. The new waveform is identical to the original waveform between points <*xfrom*> and <*xto*>. If <expression> contains multiple waveform (n–dimensional), the extraction is performed with respect to the innermost independent variable (x–axis). The use of extract is subject to the following constraints:

1. The independent variable values of the original waveform must be evenly spaced, either from a linear sweep or from application of the **logterp** function.
2. <*xfrom*> and <*xto*> are required to be within the domain of the original waveform.
3. <*xfrom*> must be less than <*xto*>.
4. For a family of curves, these constraints apply to the inner sweep.

FALL – determine the fall time of a waveform

Syntax

fall(<*expression*> [,<*fedge#*> [,<*p*> [,<*d*> [,<*type*> [,<*#bins*>]]]]])

Examples

```
print fall(v(out)); fall time of voltage at node 'out'

a=fall(v(5),2)); a holds value of fall time of 2nd falling edge
```

Description

The **fall** function returns a value determined by the difference in X values of the first proximal point minus the last distal point of the desired falling edge. These points are found as described for the **proximal** and **distal** functions. Points located between analysis points are interpolated linearly.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*fedge#*> is the number of the falling edge. If <*fedge#*> is zero, the function returns the fall time of the last falling edge. Default is zero.

<*p*> is the proximal value; the default is 10 (see <*type*>).

<*d*> is the distal value; the default is 90 (see <*type*>).

The value of <*p*> must be less than the value of <*d*>.

<*type*> determines how <*p*> and <*d*>are evaluated. If <*type*>=0, they are evaluated as percent. If <*type*>=1, then they are evaluated as actual values. Default is 0.

<*#bins*> sets the number of bins used for the histogram employed by **fall**. The default is 1001.

The proximal, mesial, and distal values of the input are computed using a frequency–count histogram, described more fully near the beginning of this section. To support the edge–finding algorithm, the same tests that are specified for **period** must be satisfied.

FFT – compute the Fast Fourier Transform of a waveform

Syntax

fft(<*expression*>,<*type*>)

Examples

```
;Example 1. In the control program browser place the command
plot db(fft(window(v(1),bh,0),per))

Example 2:
fdomain=fft(window(v(2),bh,0),aper)

;Example 3. Add the fft(@) statement to the wiremenu statment in
; the Control Program Browser:
wiremenu @-@1 rise(@) rise(@-@1) fft(@)
; After a transient simulation, point to the
; wire of interest on the schematic, use the command:
functions>fft(@)
; The resulting postage stamp can be pasted in a Plot Browser.

;Example 4. With a time domain plot window open, in the workspace
; pane at the bottom of the window type:
fft(a)
; where "a" is the name of the waveform. Highlight fft(a) and
;issue the pop-up "do it" command. This will put the frequency
;domain waveform in the paste buffer. Open another plot window
;and issue the pop-up button "paste" command. The result is a
;mag/phase or real/imag waveform as a function of frequency.
```

Additional fft examples can be found in Appendix B page B–30.

Description

The **fft** function computes the Fast Fourier Transform of a waveform.

Because the **fft** function is most often used to transform a time–domain waveform to a frequency–domain waveform, the innermost independent variable of the result is named freq. In general, however, the input waveform could be other than time–varying and the output could be based on some other variable than frequency. The input can be real, complex, positive, or negative with x–coordinates beginning at negative time, positive time, or time zero. The resultant frequency–domain output has phase that is shifted in accordance with the x–axis values of the input variable.

<*expression*> is an expression based on an n–dimensional waveform.

The argument <*type*> must be either ”per” or ”aper”. For ”per” the resultant frequency–domain waveform is normalized for a ”periodic” input variable. For ”aper” the resultant frequency–domain waveform is normalized for an ”aperiodic” input variable. The resultant waveform from an **fft** operation is quasi–symmetrical about the vertical axis.

The example PLOT command would display the value in db of the **fft** of V(1), a periodic steady–state signal. The example assignment statement would set a variable called FDOMAIN equal to the result of an **fft** of waveform V(2), data taken from a single–shot (aperiodic) event.

Use of the **fft** function is subject to these constraints (see References at the beginning of this section.):

1) The x values of the input variable must be evenly spaced (from a linear sweep).

2) The number of points of the input waveform must be even.

For a family of curves, these constraints apply to the inner sweep.

FMATRIX – insert value(s) into matrix location(s) (obsolete)*

Syntax

fmatrix(<*expression*>,<*R*>,<*C*>,<*val*>)
or
fmatrix(<*expression*>,<*R*>,<*val*>)

Examples

```
mtrx1=fmatrix(bz00,3,sqrt(2));set 1st value in 3rd row to
;sqrt(2)

nmatrix=fmatrix(mmatrix,2,2,4); row 2,col 2 element set to 4

b1=bmatrix(1,10,1.21,2,3.1,.2,-2.2,-1.78,0.2,8.99,9.1,0)
; both are one-dimensional waveforms with same number of points

v(1)=fmatrix(v(1),1,b1) ; existing waveform v(1) substituted
; with values in b1

pmatrix=fmatrix(pmatrix,3,b1); row 3 set equal to matrix b1
```

Description

The **fmatrix** function inserts a value into the indicated element of a matrix, or a series of values into a row in the input matrix.

<*expression*> is the name of an n–dimensional matrix.

<*R*> is the row number, an integer greater than or equal to one. If <*C*> is not supplied, then row <*R*> is filled beginning with the first column. If <*val*> is a scalar, then only the first column in row <*R*> is filled. If <*val*> is a waveform, then as many columns as are supplied are filled unless the number of columns in <val> exceeds the numbers of columns in row <*R*>, in which case an error is reported. As shown in an example above, this is a way to insert the Y values of a one–dimensional matrix created by **bmatrix** into a waveform such that the result has the desired X values, rather than integer column numbers created by **bmatrix**.

<*C*> is the column number of the element where <*val*>, a scalar value, is to be inserted. <*C*> must be an integer greater than zero.

<*val*> can be a number, or expression, including one containing a variable. If a one–dimensional matrix, it must fit within row <*R*> of <*expression*>.

* **To be replaced at a later date with Array Functions.**

FOLD – convert a quasi–symmetric waveform to symmetric

Syntax

fold(<*expression*>)

Examples

```
qsym=ift(w1,aper) ; qsym holds quasi-symmetrical result

asym=0.36*fold(qsym); make result domain from time=0 to time=t
 ; 0.36 equalizes coherent gain of
 ; previously applied Blackman-Harris WINDOW.
```

Description

The **fold** function returns an asymmetric waveform converted from a quasi–symmetrical waveform. Quasi–symmetrical refers to data points ranging from –x to +x*(N–2)/N), where N is the number of points in the argument <*expression*>. Asymmetrical applies to a waveform with data points ranging from zero to x. The **fold** command is of particular use to convert time–domain data created by the **ift** function, whose output is quasi–symmetrical about the x=0 axis, to a familiar format (from time=0 to time=t). See Figures 4–7 and 4–8.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

The **fold** function is subject to the following constraints:

1) The x values of the input variable must be evenly spaced (from a linear sweep).

2) The number of points in the waveform must be even.

3) The domain of x–values must be quasi–symmetric as described above.

For a family of curves, the above constraints apply to the inner sweep independent variable.

See the description of **ift** for a larger example using **fold**.

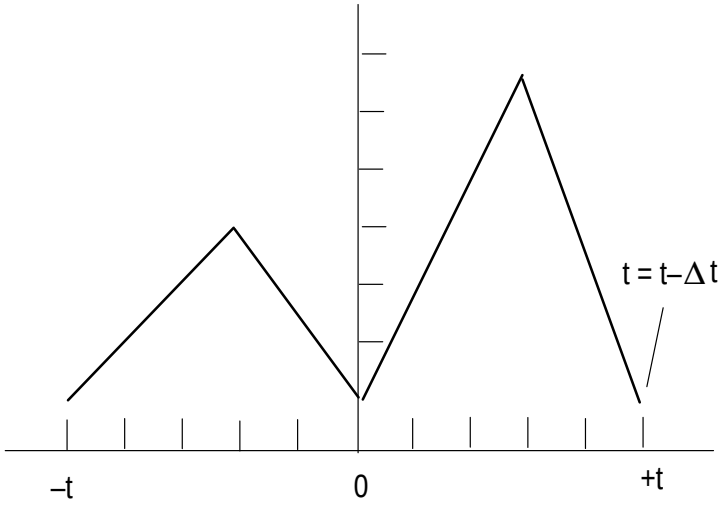


Figure 4–7: Input Waveform for Fold Example.

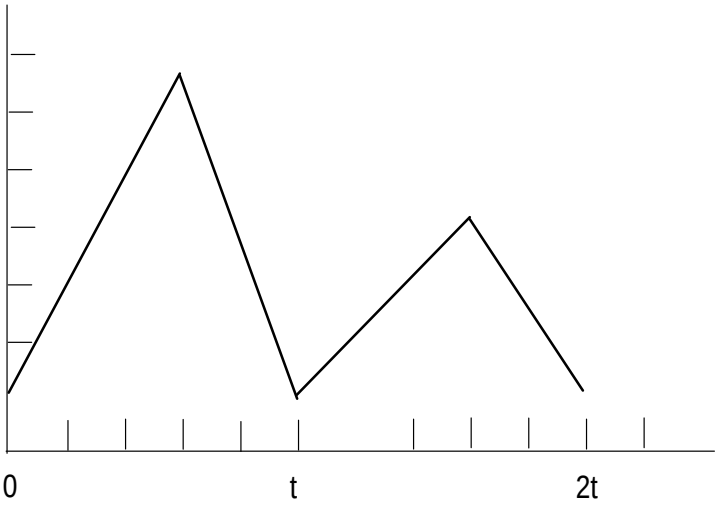


Figure 4–8: Output of Fold Example.

FREQ – determine the frequency of a waveform

Syntax

freq(<*expression*>[,<*period#*>[,<*p*>[,<*m*>[,<*d*>[,<*type*>[,<*#bins*>]]]]]])

Examples

```
print freq(v(2))
```

Description

The **freq** function returns a value equal to the frequency of a waveform. The period in which the frequency is calculated can be specified. The period is measured as described for the **period** function.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

<*period#*> selects the edge to be taken as the beginning of the period. <*period#*> plus two then becomes the edge to be taken as the end of the period. The default is zero, which selects the last period of <*expression*>.

<*p*> is the proximal value for testing for an edge. The default is 10 (see <*type*>).

<*m*> is mesial value that determines both the beginning and end of the period. The default is 50 (see <*type*>).

<*d*> is the distal value for testing for an edge. The default is 90 (see <*type*>).

<*type*> is a switch to select the meaning of <*p*>, <*m*>, and <*d*>: actual when type=1 or percent when type=0 (the default).

<*#bins*> selects the number of histogram bins used by **freq**. The default is 1001.

The proximal, mesial, and distal values of the input are computed using a frequency–count histogram. To support the edge–finding algorithm, the same tests must be satisfied as for the **period** function.

Requires an input waveform with characteristics similar to that in Figure 4–2.

HIGHPASS – determine the highpass point of a waveform

Syntax

highpass(<*expression*> [,<*down*> [,<*type*> [,<*reference*>]]])

Examples

```
print highpass(v(11)) ;node 11 is output of highpass filter

plot highpass(v(1),1,1,1) ;cross point is zero volts
```

Description

The **highpass** function returns an X–value determined by the magnitude of <expression> at the final cross point. An error is reported if the cross level determined by <*down*> and <*reference*> is not both above the first Y value and below the last Y value. See also **lowpass**, **bandpass** and **stopband**.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

<*down*> determines the cross level according to <*type*> referred to <*reference*>. If both <*down*> and <*type*> are omitted, the cross level defaults to 3 dB below <*reference*>.

<*type*> determines how <corner> and <*reference*> are evaluated. If <*type*> is zero, they are evaluated as dB below <*reference*>. If type is 1, then they are evaluated as units below <*reference*>. Default is 0.

<*reference*> is the last Y value in <*expression*> by default, but can be entered as a value (number or expression).

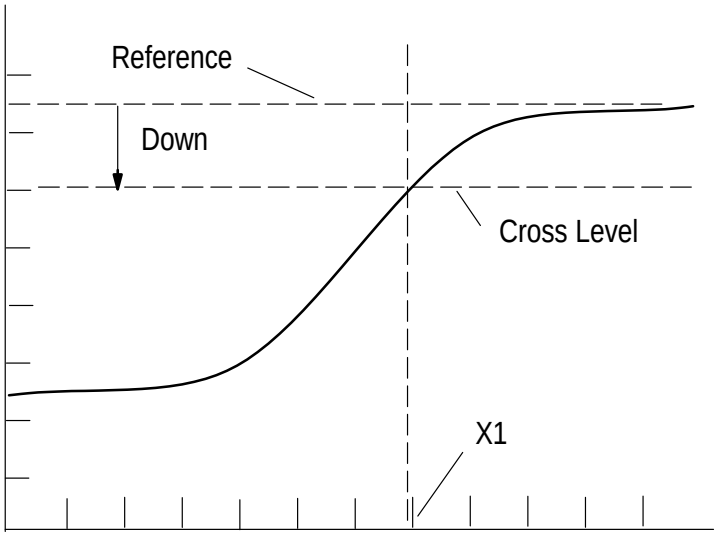


Figure 4–9: HIGHPASS Function Applied to a Waveform.

HISTOGM – determine the frequency histogram of a waveform

Syntax

histogm(<*expression*>,[,*#bins*])

Examples

```
plot histogm(v(7))
```

Description

The **histogm** function returns a waveform that is the frequency–count histogram of <*expression*>. The X values are the bin levels uniformly spaced between the minimum and maximum of <*expression*>. Each Y value of the histogram is the count of Y values of <expression> corresponding to the bin level, i.e., Y values with values enclosed by the interval surrounding the bin level.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*#bins*> selects the number of histogram bins used by **histogm**. The default is 1001, but a smaller number is recommended.

This function can be used as check on the histogram incorporated in many of the waveform measurement functions (**rise**, **overs**, **lowpass**, etc.). The function also has applications to sort results of other processing, such as the waveform of rise times produced by processing a family of waveforms with the **rise** functions.

Requires an input waveform with characteristics similar to that in Figure 4–2.

IFT – determine inverse Fast Fourier Transform of a waveform

Syntax

ift(<*expression*>,<*type*>)

Examples

```
ac analysis freq:0 100meg 100meg/128;129 point analysis fits
endac ; unfold's 1+2**N points-format

probe v(1)
u1=unfold(v(1));Make quasi-symmetrical waveform about 0 frequency
w1=window(u1,kb,2) ; Apply Kaiser-Bessel window
impulse=ift(w1,aper) ; result is quasi-symmetrical;

asym=0.49*fold(impulse) ; make result from time=0 to time=t
 ; and adjust for coherent gain of window.

asym=real(asym) ; Take real part of complex waveform
or:
asym=0.49*real(fold(ift(window(unfold(v(1)),kb,2),aper)))
```

Description

The **ift** function calculates the inverse Fast Fourier Transform of the input waveform computed for a stable system. Because in most cases it is used to transform from frequency–domain to time domain, the **ift** operation makes time the name for the innermost independent variable of the resultant waveform.

<*expression*> is an expression based on a real–valued n–dimensional waveform. The X axis must have an even number of evenly spaced points.

<*type*> must be entered as either ”per” or ”aper”. The type ”per” causes the **ift** function to treat the input variable as periodic or ”steady–state” and normalize accordingly. The type ”aper” causes the **ift** function to treat the input variable as a single–shot or aperiodic event. For example, if the frequency–domain input is the result of a swept ac analysis, then the input variable represents the ”impulse response” of the circuit.

For periodic waveforms in a quasi–symmetrical format, half of the energy is in the negative frequency domain. After applying **ift** and **fold**, the time domain signal has twice the frequency density, but half the amplitude.

In general, **fft–ift** operation pairs work with any variable quantity, not just frequency–time pairs. The **fold** command allows users of time–frequency pairs to view the waveform resulting from the **ift** function (a time–domain waveform) in a familiar format (from time=0 to time=t). The **unfold** and **fold** functions are necessary when the **window** function is to be applied to frequency–domain data; the **window** operation is performed on the quasi–symmetrical (**unfold**ed) waveform. If no windowing is needed, the **impls** or **step** functions should be used instead of **ift** because of their simplicity. See the references at the beginning of this section.

IMATRIX – determine the inverse of a matrix (obsolete)*

Syntax **imatrix**(<*expression*>)

Examples

```
A=bmatrix(2,2,1,2,3,4)
B=imatrix(A)
```

Description The **imatrix** function returns the inverse of a matrix.

<*expression*> must be, or evaluate to, a non–singular n*x*n matrix.

For the example above:

$$A=\begin{bmatrix} 1.0 & 2.0 \\ 3.0 & 4.0 \end{bmatrix}$$

$$B=\begin{bmatrix} -2.0 & 1.0 \\ 1.5 & -0.5 \end{bmatrix}$$

* **To be replaced at a later date with Array Functions.**

IMPLS – convert frequency domain waveform to impulse response

Syntax

impls(<*expression*>)

Examples

```
ac analysis freq:0 100meg 100meg/128 ; 129 point analysis
endac
probe v(1)
impulse=impls(v(1)) ; calculate time response
```

Description

The **impls** function computes a time–domain impulse response based on a frequency–domain waveform of a stable system.

<*expression*> is assumed to be based on a waveform produced by ac analysis. The input waveform is subject to the following constraints (for a family of waveforms these constraints apply to the inner sweep):

1. The independent variable domain must be from 0 to some positive value.
2. The x values of the input variable must be evenly spaced, either from a linear sweep or from application of the **logterp** function.
3. The variable must contain n points where n is odd, with a mimimum value of 17 . The resultant waveform contains 2*(N+1) points.

The **impls** function is used when windowing is not necessary before an **ift** is performed. If the frequency–domain response has achieved its asymptotic value at the highest analysis frequency, then there is no need to apply the **window** operation, hence no need to perform the **unfold** and **fold** operations, and the impulse response is requested directly using the **impls** function. See the References at the beginning of this section.

INDATT – create matrix from dependent and independent variables (obsolete)*

Syntax

indatt(<*expression1*>,<*expression2*>)

Examples

```
var sweep tvar : 0 3 1 ; makes matrix with 4 waveforms (rows)
dc analysis dcin:0 3 1 f=dcfile1 ; each waveform has 4 points
enddc
endvar

bldvec=bmatrix(4,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4)
 ; creates 4x4 matrix with independent variables
 ; called "row" and "column", each row with with ; values 1, 2,
3, and 4

probe v(1)=vec:f=dcfile1 ; loads dc analysis waveform called
 ; "vec" with independent variables called "tvar
 ; and "dcin"

y=indatt(bldvec,vec) ; outputs independent variable values
 ; and names from "vec" with the dependent variable
 ; values from "bldvec" into new 4x4 matrix

print y ; output of print command follows:

 ; tvar dcin y
 ;
 ; 0 0 1
 ; 1 2
 ; 2 3
 ; 3 4
 ; 1 0 1
 ; 1 2
 ; 2 3
 ; 3 4
 ; 2 0 1
 ; 1 2
 ; 2 3
 ; 3 4
 ; 3 0 1
 ; 1 2
 ; 2 3
 ; 3 4
```

* **To be replaced at a later date with Array Functions.**

Description

The **indatt** function returns a result that has the dependent data attributes of <*expression1*> and the independent data attributes of <*expression2*>. The dependent data attributes are the name, type, number of points, and values of the dependent data of <*expression1*>. The independent data attributes are the name, type, number of points, ”from” value, ”to” value, ”by” value, and independent variable values for each dimension of <*expresssion2*>.

<*expression1*> and <*expression2*> must, in general, have the same dimensions. In dimension one, the expressions must have the same number of columns (points on the waveform containing dependent values), in dimension two, the same number of rows (waveforms), and in dimension three, the same number of families of waveforms. For example, a matrix with a single row and a waveform can both be arguments if they have the same number of points.

INT – compute the integral of a waveform

Syntax

int(<*expression*>)

Examples

```
plot int(v(1))
trap=int(v(1))
```

Description

The **int** function calculates a trapezoidal approximation of the integral of a waveform described by <*expression*>. The calculation is performed using the value of the first point as the lower integration bound and the value of the last point as the upper integration bound. For a family of curves, the operation is performed one waveform at a time.

INTERP – interpolate additional points for a waveform

Syntax

interp(<*expression*>,<*npts*>)

Examples

```
tran analysis time:0 10n .1n ;101 point tran analysis
endtran

probe v(1)
newvec=interp(v(1),256) ; interpolate to 256 points
```

Description

The **interp** function performs a linear interpolation of the input waveform described by <*expression*>. The output contains the number of points specified by <*npts*>. A common use of **interp** is to change the number of points to be acceptable for an **fft**–related function. In the example, a 101–point waveform is expanded for input to an **fft** operation, which works only on a waveform with an even number of points. **interp** should be used when the input waveform is rapidly varying (a step or a pulse). Smooth functions should be interpolated by the **spline** function, for logarithmically spaced points, use the **logterp** function.

LOGTERP – create a linear waveform from a logarithmic waveform

Syntax

logterp(<*expression*>,<*npts*>)

Examples

```
ac analysis freq:100 100meg 20 type=dec ; logarithmic steps
endac

linvar=logterp(v(1),257); Change to linear steps from 0 to
 ; 100meg.
 ; Points from zero to 100 Hz are assigned
 ; the value of v(1) at 100 Hz.
```

Description

The **logterp** function causes a linear interpolation on the waveform of logarithmically spaced points described by <*expression*>. The result is a waveform with the number of equally space points specified by <*npts*> and with a domain of zero to the maximum x value of the input variable. The input variable should be based on a logarithmic analysis sweep (type=oct or type=dec). For a family of curves, only the inner sweep need be logarithmic.

The **logterp** function converts a waveform to a format compatible with the **ift**, **step**, and **impls** functions, which require a linear sweep starting at zero for the innermost independent variable.

At least one point will be generated beyond the left end of the waveform. This point (at zero) and any others needed to accomplish equal spacing between zero and the end point of the input variable are assigned the y value of the end point.

LOWPASS – determine the lowpass point of a waveform

Syntax

lowpass(<*expression*> [,<*down*> [,<*type*> [,<*reference*>]]])

Examples

```
print lowpass(v(x1)) ;node x1 is output of lowpass filter
plot lowpass(v(1),3,0,2);cross point is .707*2
```

Description

The **lowpass** function takes the magnitude of the input expression, searches for the final cross point determined by <*down*> and <*reference*> and returns its X value. An error is reported if the magnitude of the first Y value is below the cross level or the magnitude of the last Y value is above the cross level. See also **highpass**, **bandpass**, and **stopband**.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

<*down*> determines the cross level according to <*type*> referred to <*reference*>. If both <*down*> and <*type*> are omitted, the cross level defaults to 3 dB below <*reference*>.

<*type*> determines how <corner> and <*reference*> are evaluated. If <*type*> is zero, they are evaluated as dB below <*reference*>. If type is 1, then they are evaluated as units below <*reference*>. Default is 0.

<*reference*> is the first Y value in <*expression*> by default, but can be entered as a value (number or expression).

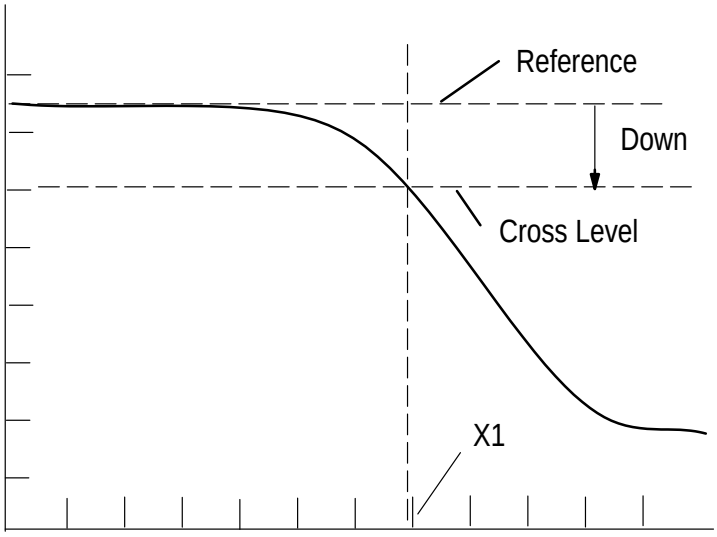


Figure 4–10: LOWPASS Function Applied to a Waveform.

MEAN – compute the time–varying mean value of a waveform

Syntax

mean(<*expression*>)

Examples

```
avgval=mean(v(1))
plot mean(v(2))
```

Description

The **mean** function computes the time–varying mean value of the input waveform, <*expression*>, which must be based on a linear sweep. The result of the **mean** operation is a new waveform whose value is the ”instantaneous” mean value of the input waveform. The **mean** value of a waveform approaches a steady–state value equal to a constant as the waveform approaches a steady–state condition.

MERGE – Merge two waveforms along the independent axis

Syntax **merge**(<*expression1*>, <*expression2*>)

Examples `merge(a,b)`

Description The **merge** function merges two waveforms along the independent axis. If two points fall on the same independent value, use value from the first waveform.

<*expression1*>*and* ,<*expression2*> are expressions based on real–valued one dimensional waveform.

MESIAL – determine the mesial value of a waveform

Syntax

mesial(<*expression*> [,<*m*> [,<*#bins*>]])

Examples

```
print mesial(v(4)) ; 50% level on waveform
print mesial(v(4),40) ; 0.4*(topline(v(4))-baseline(v(4)))
```

Description

The **mesial** function returns the value of the input waveform defined by <*expression*> according to the value entered as <*m*>:

```
baseline + (topline - baseline)*m/100
```

The topline and baseline are found by a frequency–count histogram as described for the **topline** and **baseline** functions.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*m*> is the desired mesial value in percent and need not be between zero and 100. The default is 50 percent.

<*#bins*> selects the number of histogram bins used by this **mesial**. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

MINPHS – create a minimum phase–shift waveform

Syntax

minphs(<*expression*>)

Examples

```
ac analysis freq: .1 10g 10 type=dec
endac
```

```
probe v(6)=t
plot mag(t) mag(1+t) mag(1–2*t)
plot phase(1+t)/180; 1+t meets conditions judging by display
```

```
plot phase(1+t)/180 phase(minphs(1+t))/180; check for
 ; differences
```

Description

The **minphs** function is used for stability testing. The input <*expression*> should be a complex–valued waveform created by swept ac analysis (or derived from such a variable). The **minphs** function creates a waveform that is the minimum phase–shift function associated with the real part of the input waveform.

Given the proper conditions, stability is determined using the **minphs** function. The stability test requires a properly chosen waveform derived from ac analysis. Stability is determined by comparing the waveform and the **minphs** function of the variable. The circuit is stable if, and only if, the output of **minphs** matches the input. The example follows this procedure. A testable function is identified from plots of several derived from a waveform created by ac analysis. The one selected is compared to the **minphs** function output.

MMATRIX – multiply two compatible matrices (obsolete)*

Syntax

mmatrix(<*expression1*>,<*expression2*>)

Examples

```
; circuit
; circuit description
; endc
tran analysis 0 100n 1n
endtran

probe v(3)=x;x is one-dimensional waveform or 1X101 matrix
y=tmatrix(x) ; y is transpose of x, a 101X1 matrix

dot=mmatrix(x,y) ; produces dot product, a scalar
prod=mmatrix(y,x) ; prod is 101X101 matrix
```

Description

The **mmatrix** function multiplies two compatible matrices and returns the result, a matrix of dimension n *x* m given <*expression1*> of n x k and <*expression2*> of k x m.

<*expression1*> and <*expression2*> are matrices and must be compatible in dimension, i.e., the number of columns of <*expression1*> equals the number of rows of <*expression2*>.

* **To be replaced at a later date with Array Functions.**

NRAND – generate random number from a Gaussian distribution

Syntax

nrand()

Examples

```
ab=bmatrix(1,50,0) ; makes 50-point waveform
rand=nrand() ; puts random numbers in ab
```

Description

The **nrand** function returns a random number chosen from a normal (Gaussian) distribution with zero mean and unity variance. **Nrand** employs the same generator as the **urand** function to avoid sequential correlation in a sequence of numbers. See also: **randset.**

OVERS – determine the percent overshoot of a waveform

Syntax

overs(<*expression*>[,<*reference*>[,<*#bins*>]])

Examples

```
print overs(pulseo) ; pulseo is transient pulse response
```

Description

The **overs** function returns the overshoot in percent of the defined by <expression>: (**tmax**(<*expression*>)–<*reference*>)/(topline – baseline)x100

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*reference*> is the reference level. By default, it is the topline as computed by the **topline** function.

<*#bins*> selects the number of histogram bins used by **overs**. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

PEAK – determine the frequency and magnitude of a peak

Syntax

peak(<*expression*>[,<*peak#*>[,<*srchbw*>[,<*floor*>[,<*type*>]]]])

Examples

```
print peak(v(111)) ; complex number
fpk=real(peak(v(111))) ; frequency of peak value
vpk=imag(peak(v(111))) ; magnitude of peak value
```

Description

The **peak** function returns the frequency and magnitude of the peak of a real or complex waveform. The function can find the peak that occurs first, second, or third, etc., above a specified level (floor). What is returned by **peak** is a complex number: the real part is the frequency of the peak; the imaginary part is the magnitude of the peak.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform. The **peak** operates on the magnitude of the input expression.

<*peak#*> an integer greater than or equal to zero. Selects the peak to be returned, searching from left–to–right. Returns the maximum peak and the lowest frequency at which it occurred. The default is zero.

<*srchbw*> sets the search resolution as +_<*srchbw*> percent of the last X value of <*expression*>. A peak is a value greater than any value within this search resolution. The default is four percent.

The search is restricted to peaks greater than the level set by <*floor*>. The default is 80 dB below the maximum of the input expression, but another level can be entered as a number or expression in units or dB (see <*type*>). If entered as units, peaks must exceed **tmax**(<*expression*>)–<*floor*>. If entered as dB, peaks must be greater than a level <*floor*> dB down from **tmax**(<*expression*>).

<*type*> is a switch to interpret <*floor*>: as units when type=1 or dB when type=0. Default is 0.

PERIOD – determine the period of a waveform

Syntax

period(<*expression*>[,<*period#*>[,<*p*>[,<*m*>[,<*d*>[,<*type*>[,<*#bins*>]]]]]])

Examples

```
print period(v(2))
```

Description

The **period** function returns a period selected from the input expression. Period is the distance between the X value at the mesial of the first edge of the desired period and the mesial of the third edge. Proximal, mesial, and distal values can be entered to control how <expression> is searched for a period.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*period#*> selects the edge to be taken as the beginning of the period. <*period#*> plus two then becomes the edge to be taken as the end of the period. The default is zero, which selects the last period of <*expression*>.

<*p*> is the proximal value for testing for an edge. The default is 10 (see <*type*>).

<*m*> is mesial value that determines both the beginning and end of the period. The default is 50 (see <*type*>).

<*d*> is the distal value for testing for an edge. The default is 90 (see <*type*>).

<*type*> is a switch to select the meaning of <*p*>, <*m*>, and <*d*>: actual when type=1 or percent when type=0 (the default).

<*#bins*> selects the number of bins for the histogram employed by **period**. The default is 1001.

The proximal, mesial, and distal values of the input are computed using a frequency–count histogram. To support the edge–finding algorithm, the tests listed below must be satisfied or an error is reported:

a. **tmin**(<*expression*>)<**proximal**(<*expression*>)

b. **proximal**(<*expression*>)<**mesial**(<*expression*>)

c. **mesial**(<*expression*>)<**distal**(<*expression*>)

d. **distal**(<*expression*>)<**tmax**(<*expression*>)

Requires an input waveform with characteristics similar to that in Figure 4–2.

PROXIMAL – determine the proximal point of a waveform

Syntax

proximal(<expression> [,<p> [,<#bins>]])

Examples

```
print proximal(v(3)) ; first crossing at 10%
```

Description

The **proximal** function computes dependant value of the waveform defined by <*expression*> as:

baseline + (topline – baseline)*<p>/100

The topline and baseline are found by a frequency–count histogram as described for the **topline** and **baseline** functions. See also the **distal** function.

<*expression*> is an expression based on a real– valued or complex–valued n–dimensional waveform.

<*p*> sets the desired proximal level in percent. The default is 10 percent. Values less than zero and greater than 100 are permitted particularly for waveforms with undershoot or overshoot.

<*#bins*> selects the number of bins **proximal** uses for the histogram. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

RANSET – set seed for random number generators

Syntax

ranset(<expression>)

Examples

```
Example 1:
samestartingpoint=9.9
var sweep temp: 25 125 25
  ;each temperature sweep use same sequence of cktvar values
  ranset(samestartingpoint)
  var sweep dummy: 1,15,1
     cktvar=(100+nrand())*50m
    tran analysis time: 0,1u,1n
    endtran
    endvar
endvar
```

```
Example 2:
array arr[44]
i=43
ranset(12.0d-44)
while (i>(-1)) {
  arr[i]=nrand()
  i=i-1
}
```

Description

The argument to **ranset** is a new seed for the random number generator. The random number generator underlies the **nrand()** and **urand()** functions. Identical random number sequences can be obtained by calling ranset with the same argument at the start of each sequence. The return value from ranset has no meaning and should be ignored.

NOTE*. The* ***ranset*** *function can only be used inside a var sweep block. In some cases it may be desireable to add an outer var sweep, solely to allow the ranset function to be seen by TekSpice.*

REDUCE – reduce a multi–dimensional waveform by one

Syntax

reduce(<*expression*>,<*independent variable*>)
or
reduce(<*expression*>,<*independent variable*>, <*independent value*>)

Examples

```
var sweep cval:1 3 1 ; outer sweep
tran analysis time:0 10n .1n ; inner sweep
endtran
endvar

time=5n ; Assign value to inner indep. variable
v4cap=reduce(v(1),time) ; v4cap contains voltages taken at 5n
; intervals from each curve. cval is an independent variable

cval=1 ; Assign value to outer indep. variable
waveform1=reduce(v(1),cval) ; Get 1st waveform from data

reduce(waveform1,time); Get scalar value of 1st waveform at t=5n
```

Description

The **reduce** function extracts values from a waveform reduced by one in dimension from the input waveform. This function is identical to the **reduceinterp** function except that interpolation is not used. For a three–dimensional waveform as created by ac, dc, or transient analysis nested within two variable sweeps, **reduce** obtains a two–dimensional family of waveforms. See also **reduceinterp**.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform. For a family of curves, the result is a waveform; for a single waveform, the result is a scalar value. If the value requested lies between points of the waveform, the **reduce** function returns the value of the waveform nearest the point requested.

<*independent variable*> is an independent variable of <*expression*>, such as an analysis variable (FREQ, TIME, DCIN) or the independent variable in a variable sweep. With this independent variable fixed at its current value, the function returns all dependent values corresponding to the remaining independent variable(s) of <*expression*>. For instance, the argument selects the row or column of a two–dimensional waveform. Naming an analysis variable (FREQ, TIME, or DCIN) returns a waveform that is one point from each curve taken at the same point of the inner sweep. This can be thought of as a column from a data matrix. Again for a two dimensional variable, naming the variable of a variable sweep causes one curve to be returned from the family of curves, a row from the data matrix.

<*independent value*> if present, the waveform is reduced at this value.

REDUCEINTERP – reduce a multi–dimensional waveform by one with interpolation

Syntax

reduceinterp(<*expression*>,<*independent variable*>)
or
reduceinterp(<*expression*>,<*independent variable*>, <*independent value*>)

Examples

```
var sweep cval:1 3 1 ; outer sweep
tran analysis time:0 10n .5n ; inner sweep
endtran
endvar

time=5.3n ; Assign value to inner indep. variable
v4cap=reduceinterp(v(1),time);v4cap contains voltages taken at
;          5.3n from each curve, cval is independent variable

cval=1 ; Assign value to outer indep. variable
waveform1=reduceinterp(v(1),cval) ; Extract first waveform
;                                    from family of curves

reduceinterp(waveform1,time); Returns scalar value of first
;                              waveform at time=5n
```

Description

The **reduceinterp** function extracts values from a waveform reduced by one in dimension from the input waveform. This function is identical to the **reduce** function except that interpolation is used if the reduction point is not a data point. For a three–dimensional waveform as created by ac, dc, or transient analysis nested within two variable sweeps, **reduceinterp** obtains a two–dimensional family of waveforms. See also **reduce**.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform. For a family of curves, the result is a waveform; for a single waveform, the result is a scalar value. If the value requested lies between points of the waveform, the **reduceinterp** function interpolates the value of the waveform based on the two ajacent points.

<*independent variable*> is an independent variable of <*expression*>, such as an analysis variable (FREQ, TIME, DCIN) or the independent variable in a variable sweep. With this independent variable fixed at its current value, the function returns all dependent values corresponding to the remaining independent variable(s) of <*expression*>. For instance, the argument selects the row or column of a two–dimensional waveform. Naming an analysis variable (FREQ, TIME, or DCIN) returns a waveform that is one point from each curve taken at the same point of the inner sweep. This can be thought of as a column from a data matrix. Again for a two dimensional variable, naming the variable of a

variable sweep causes one curve to be returned from the family of curves, a row from the data matrix.

<*independent value*> if present, the waveform is reduced at this value.

REPEAT – concatenate points to extend a waveform

Syntax

repeat(<*expression*>,<*n*>)

Examples

```
tran analysis time:0 10n 10n/63 ; 64 point analysis
endtran
```

```
probe v(1)
bunch=repeat(v(1),4) ; make 256-point variable
```

Description

The **repeat** function concatenates <*n*> copies of the waveform evaluated from <*expression*>. The input is repeated on a [ ) basis; that is, the first point of the copy is appended one point to the right of the right–most point of the original with the same spacing as the original points. This is in keeping with the literature regarding fft–related functions.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*n*> is an integer, greater than or equal one, that specifies how many times the input variable is to be repeated.

An application of **repeat** is to extend the number of periods of a steady–state periodic time–domain variable, saving the expense of extending the transient analysis. For example, a transient analysis of a high–Q circuit need only include one or two cycles of steady–state operation of the circuit at the end of the analysis output. For further processing of the large–signal steady state data, the **extract** function could be applied to recover those cycles of interest, and the **repeat** function could be applied to create as many cycles as desired for **fft** analysis. (The more cycles in the input time–domain variable, the better the resolution in the frequency–domain after using the **fft** function.)

RISE – determine risetime of a waveform

Syntax

rise(<*expression*> [,<*redge#*> [,<*p*> [,<*d*> [,<*type*> [,<*#bins*>]]]]])

Examples

```
print rise(v(out)) ; risetime of voltage at node 'out'
a=rise(v(5),2)) ; a holds risetime of 2nd rising edge
```

Description

The **rise** function returns the difference in X values of the first distal point minus the last proximal point of the selected rising edge. These points are found as described for the **proximal** and **distal** functions. Points located between analysis points are interpolated linearly.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*redge#*> is the number of the desired rising edge. For zero, the function returns the last rise time. Default is 0.

<*p*> is the proximal value (in percent); the default is 10 (see <*type*>).

<*d*> is the distal value (in percent); the default is 90 (see <*type*>).

<*type*> is a switch which determines how <*p*> and <*d*> are evaluated. If type = 0, <*p*> and <*d*> are in percent. If type = 1 then they are the actual value. Default is 0.

<*#bins*> selects the number of bins **rise** uses for the histogram. The default is 1001.

The **rise** requires that the value of <*p*> be less than the value of <*d*>.

Requires an input waveform with characteristics similar to that in Figure 4–2.

RMS – compute the root–mean–square value of a waveform

Syntax

rms(<*expression*>)

Examples

```
rmsval=rms(v(1))
plot rms(v(2))
```

Description

The **rms** function computes the time–varying root–mean–square value of the input. The output is the ”instantaneous” **rms** value of the input waveform. The output approaches a steady–state value as the input waveform approaches a steady–state condition.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

Use of the **rms** is subject to the following constraints:

1. The waveform must be based on a linear sweep. For a family of curves, this applies to the inner sweep.
2. Whether the waveform is real or complex, the output waveform is of type *real*.

SCALAR – extract a scalar value from a waveform

Syntax

scalar(<*expression*>,<*x1*>[,<*x2*>,...,<*xn*>])

Examples

```
; extracting a scalar value from a family of curves
var sweep cval:1u 3u 1u ;outermost ind. var is cval
ac analysis freq:1 100meg 30 type=dec;innermost ind. var is freq
endac
endvar

; v(2) is waveform with outer sweep values of 1u,2u,and 3u
; Inner sweep is freq from 1 to 100meg with 30 points/decade
eval scalar(phase(v(2)),2.5u,70meg)
; returns approximate phase at cval=2.5u and freq=70meg
```

Description

The **scalar** function returns the value of the input at the point specified by <*x1*>, <*x2*>, ...,<*xn*>. Brackets in the format above indicate <*x2*>, <*x3*>, etc., are optional. Linear interpolation is used to determine the value if the point lies between points on the waveform. If the point requested is outside the domain of the variable, the value returned is that of the nearest point.

<*expression*> is an expression based on a real–valued n–dimensional waveform. The value returned by the **scalar** operator is of the same type (real or complex) as that of the input.

The number of arguments <*x1*>, <*x2*>, ..., <*xn*>, must match the number of dimensions of the variable. The order is related to the sweep analysis that created the variable: outermost–to–innermost. For a single waveform, only <*x1*> is specified. For a two–dimensional family of curves, <*x1*> identifies the value of the variable swept, and <*x2*> identifies the value of the inner sweep variable (TIME, FREQ, or DCIN). The arguments <*x1*>, <*x2*>, ..., <*xn*> represent the value of the independent variable, not the number of the point. They should be real scalar values and can be expressions including variables.

A special application of the **scalar** function allows ”circuit partitioning” for dc or transient analysis, simulation of a large circuit by serially simulating portions. The source for a succeeding sweep analysis is specified proportional to a previously calculated waveform. This makes available the output of one analysis as the input for another analysis.

SETTLE – determine the settling time of a waveform

Syntax

settle(<*expression*> [,<*srchband*> [,<*type*> [,<*#bins*>]]])

Examples

```
print settle(v(11))
```

Description

The **settle** function determines the settling time of the input expression. **Settle** searches backwards from the end of the waveform to the first point outside the search band. It returns the X value of the following point. See Figure 4–11.

<*expression*> is an expression based on a real–valued one dimensional waveform or two or more dimensional family of waveforms.

The search band extends +<*srchband*> centered on either the topline or baseline of <*expression*>, depending on whether the last data value falls within the top search band or the bottom search band. If neither, the function reports an error. The default is one percent of the difference between topline and baseline as defined by the **topline** and **baseline** functions.

<*type*> determines how <*srchband*> is evaluated. If <type> = 0, <*srchband*> is evaluated as percent of topline – baseline. If <*type*> = 1, then <*srchband*> is the actual level. Default is zero.

<*#bins*> selects the number of bins **settle** uses for the histogram. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

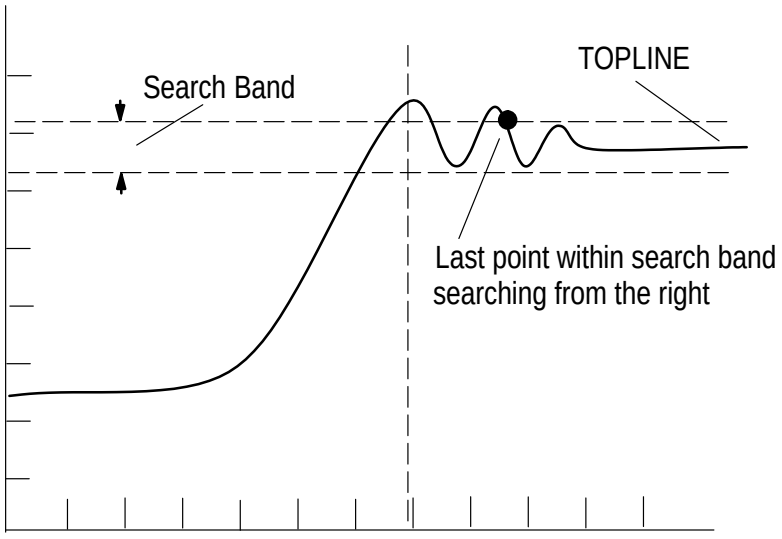


Figure 4–11: SETTLE Function Applied to a Waveform.

SLEW – determine the slew rate of a waveform

Syntax

slew(<*expression*> [,<*edge#*> [,<*p*> [,<*d*> [,<*type*> [,<*#bins*>]]]]])

Examples

```
print slew(v(out)) ; rise time of voltage at node 'out'
a=slew(v(5),2)) ; a holds rise time of second rising edge
```

Description

The **slew** function returns the rate of change between the distal and proximal values for the selected edge. The last distal point and the first proximal point of the are chosen for a falling edge and the first distal point and last proximal point are chosen for a rising edge. Points located between analysis points are interpolated linearly. The slew rate for a falling edge is negative.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<edge#> is the number of the desired edge. For zero the function operates on the last edge. Default is zero.

<*p*> is the proximal value; the default is 10 (see <*type*>).

<*d*> is the distal value; the default is 90 (see <*type*>).

<*type*> determines how <*p*> and <*d*> are evaluated. If <*type*> = 0, <*p*>and <*d*> are evaluated as percent of topline – baseline. If <*type*> = 1, then they are the actual reference level. Default is zero.

<*#bins*> selects the number of histogram bins used by **slew** to determine topline and baseline The default is 1001.

The proximal, mesial, and distal values of the input are computed using a frequency–count histogram as described for the **proximal** and **distal** functions. To support the edge–finding algorithm, the tests specified for the **period** function must be satisfied.

SMATRIX – solve set of linear equations represented by a matrix (obsolete)*

Syntax

smatrix(<*expression1*>,<*expression2*>)

Examples

```
A=bmatrix(2,2,1,2,3,4)
b=bmatrix(2,1,0,1)
x=smatrix(A,b)
c=bmatrix(1,2,4,6)
y=smatrix(A,c)
```

Description

The **smatrix** function solves a system of linear equations represented by the input expressions. The function applies LU decomposition with partial pivoting. It then makes one iteration improvement of the solution.

<*expression1*> is, or is based on, an n x n non–singular matrix.

<*expression2*> is either an n x 1 or 1 x n data structure.

Given A=<*expression1*> and b=<*expression2*>, then for the assignment statement

```
x=smatrix(A,b)
```

1. The function solves Ax=b and returns x with a data structure of n x 1 if b is an n x 1 data structure.
2. The function solves xA=b and returns x with a data structure of 1 x n if b is an 1 x n data structure.

* To be replaced at a later date with Array Functions.

SMOOTH – smooth waveform

Syntax **smooth**(<*expression*>, *times*)

Examples

```
smooth(a,5)
```

Description Smooth waveform by repeated application of dif(int(<expression>)). The number of repeated integration then differentiation cycles is specified by <*n*>, an integer greater than zero.

<*expression*> is an expression based on a real–valued or complex–valued n–dimensional waveform.

SPLINE – add points to waveform using a cubic spline function

Syntax

spline(<*expression*>,<*n*>)

Examples

```
tran analysis time:0 100n 100n/99;perform 100 point analysis
endtran

newpts=spline(v(1),128) ; change v(1) to 128 points
```

Description

The **spline** function performs a clamped, cubic–spline interpolation of the input waveform. The result comprises the number of points specified by <*n*>, an integer greater than zero.

<*expression*> is an expression based on a real–valued or complex valued n–dimensional waveform. <*expression*> must be based on a linear sweep; this applies to the inner sweep of a family of curves.

Spline can be used to change the point count of a waveform, although the result of spline may not maintain the same accuracy as the original waveform. For example **FFT** requires an even number of points in its input waveform. Spline can be used to convert a waveform with an odd number of points to an equivalent waveform with an even number of points.

A common use of **spline** is to change the number of points to be acceptable in an **FFT**–related function.

The **spline** function should be used when the input waveform is a smoothly varying function with no sharp discontinuities. Because the cubic spine computation requires the input waveform be third order differentiable, its use on sharply discontinuous variables can cause ”ringing”. The **interp** function, rather than **spline**, should be used if the waveform contains rapidly varying functions.

STEP – compute time domain step response of ac waveform

Syntax

step(<*expression*>)

Examples

```
ac analysis freq:0 100meg 100meg/128 ; do 129 point analysis
endac

tdomain=step(v(1)) ; calculate time response
```

Description

The **step** function computes the time–domain step response of a stable frequency–domain waveform. The input for **step** is assumed to be an output from ac analysis.

<*expression*> is an expression based on a real–valued n–dimensional waveform subject to the following constraints:

1. Its domain must be ≥ 0.
2. It must be based on a linear sweep.
3. The independent variable must comprise n points where n is an integer with an odd number of points.

For a family of curves, the constraints apply to the inner sweep. The number of points of the waveform produced by **step** is m=2^N+1.

The **step** function is preferred when windowing need not be applied to the frequency–domain input waveform before an **ift** is performed. Windowing is unnecessary if the frequency–domain response achieves its asymptotic value at the highest analysis frequency. Use of **step** saves the user from calling the **unfold**, **window**, **ift**, **fold**, and **int** operations to calculate step response. See the References at the beginning of this section.

STOPBAND – determine the stopband of a waveform

Syntax

stopband(<expression> [,<down> [,<type> [,<reference>]]])

Examples

```
print stopband(v(11)) ; node 11 is stopband filter output

plot stopband(v(1),6) ; v(1) is a family of curves
 ; function returns a waveform where x is outer variable
 ; of family of curves and y is bandpass of each
```

Description

The **stopband** function measures the width between the initial falling edge and the final rising edge of the magnitude of the input expression. The width is measured between the cross points where the magnitude of <*expression*> matches the cross level established by <down> and <*reference*>. The cross points are interpolated if either falls between the data of <expression>. See also **bandpass, lowpass** and **highpass.**

<*expression*> is an expression based on a real–valued one dimensional waveform or two or more dimensional waveforms. An error is reported if the magnitude of the first or last data point of the expression is above the cross level; this applies to each waveform.

<*down*> determines the cross level according to <*type*> referred to <*reference_level*>. If both <*down*> and <*type*> are omitted, the cross level defaults to 3 dB below <*reference_level*>.

<*type*> determines how <corner> and <*reference_level*> are evaluated. If <*type*> is zero, they are evaluated as dB below <*reference_level*>. If type is 1, then they are evaluated as units below <*reference_level*>. Default is 0.

<*reference*> is the topline by default, but can be entered as a value (number or expression). The default topline is computed as for the **topline** function with a histogram of 1001 bins.

Requires an input waveform with characteristics similar to that in Figure 4–2.

SUM – compute the running sum of a waveform amplitude*

Syntax sum(<*expression*>)

Examples

```
A=sum({3,2,1,2,3,4,5,6})
; results= {3,5,6,8,11,15,20,26}
```

Description The **sum** functions returns the sum of the y values of waveform <expression>.

***** **To be replaced at a later date with Array Functions.**

TMATRIX – compute the transpose of a matrix (obsolete)*

Syntax

tmatrix(<*expression*>)

Examples

```
A=bmatrix(3,2,1,2,3,4,5,6)
B=tmatrix(A)
```

Description

The **tmatrix** functions returns the transpose of the input matrix following the rule ”the i^{th} row becomes the i^{th} column.” Given a dimension of n x m for the input matrix, the output would be m x n.

<*expression*> must be, or evaluate to, a non–singular matrix.

In the example,

```
     1 2
A =  3 4
     5 6

B =  1 3 5
     2 4 6
```

* To be replaced at a later date with Array Functions.

TMAX – compute the maximum value of a waveform

Syntax **tmax**(<*expression*>)

Examples

```
print tmax(V(1))
```

Description The **tmax** function returns the maximum value for the input expression.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

TMIN – compute the minimum value of a waveform

Syntax **tmin**(<*expression*>)

Examples

```
print tmin(V(1))
```

Description The **tmin** function returns the minimum value for the input expression.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

TOPLINE – determine the topline of a waveform

Syntax

topline(<expression>[,<#bins>])

Examples

```
print topline(v(1)-v(2)); voltage at nodes 1 and 2 are
; real-valued
baseout=topline(pulseo,100); pulseo is transient pulse
response=topline(mag(acv)) ; acv is ac response of passband
; filter
; returns apparent magnitude of signal within passband
```

Description

The **topline** function returns the value of the topline of the input expression.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

Topline is determined by constructing a histogram with evenly spaced bins between <*expression*>'s minimum and maximum independent variable values. The number of points landing in each bin from <*expression*> are counted. The bin with the largest count for bins above the midpoint of <*expression*> is returned as **topline**. In case of a tie (more than one bin below the midpoint contains the largest count), the lower bin is returned.

<*#bins*> specifies the number of bins (resolution) used in determining **topline**. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

UNDERS – compute the percent undershoot of a waveform

Syntax

unders(<*expression*>[,<*reference*>[,<*#bins*>]])

Examples

```
print unders(pulseo) ; pulseo is transient pulse response
```

Description

The **unders** function returns the undershoot in percent of the waveform defined by <*expression*>*:*

(**tmin**(<*expression*>)–<*reference*>)/(topline – baseline)*100

Topline and baseline are computed as described for the **topline** and **baseline** functions.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

<*reference*> is the reference level. By default, this is the baseline.

<*#bins*> specifies the number of bins (resolution) used in determining **unders**. The default is 1001.

Requires an input waveform with characteristics similar to that in Figure 4–2.

UNFOLD – convert from an asymmetric to symmetric waveform

Syntax

unfold(<expression>)

Examples

```
u1=unfold(v(1)); if v(1) is 129 points from 0 to 128 MHZ, u1
 ; will be 256 points from -128 MHZ to 127 MHZ
```

Description

The **unfold** function converts from an asymmetrical data format to a quasi–symmetrical format. When applied to an input variable (generally complex) of N points with a domain of zero to +x, the function returns a variable with domain –x to x*(N–2)/(N–1) for 2N–2 points). The real part of the resultant has even symmetry about the y axis, that is, y(x)=y(–x). The imaginary part of the resultant has odd symmetry about the y=0 axis, that is, y(x)=–y(–x).

<*expression*> is an expression based on a real–valued n–dimensional waveform.

The **unfold** operation is used before applying the **ift** function when a swept ac (frequency–domain) analysis is to be converted to a time–domain impulse response. An example is shown with the **ift** function description. As explained there, **unfold** and **ift** are used when windowing is to be applied to the frequency–domain waveform before the inverse Fourier transformation. Otherwise, the **impls** or **step** functions are to be preferred for their ease of use.

The following constraints apply to the use of **unfold**:

1. The domain of x values of the input waveform must be from 0 to some positive number (asymmetric about the y axis).
2. The variable must contain an odd number of points
3. The input waveform must be from a linear sweep (points equally spaced).

If the waveform is a family of curves, these constraints apply to the inner sweep.

URAND – generate a random number from a uniform distribution

Syntax **urand**( )

Examples

```
rand=urand()
```

Description The **urand** function returns a random number chosen from a uniform distribution between zero and one. If the argument is a waveform rather than a scalar, the function fills the variable, point–by–point. This function performs additional shuffling of the output of the system–supplied **randf** function to avoid sequential correlation. **Urand** employs the same generator as the **nrand** function to avoid sequential correlation in a sequence of numbers. See also: **randset.**

VERSUS – make dependent data the independent values on plot

Syntax

versus(<*expression1*>, <*expression2*>)

Examples

```
versus(ocvt,bvt)
```

If ocvt is output current versus temperature and bvt is bandwidth versus temperature, execution of this function would result in a plot of output current versus bandwidth. See Figure 4–12 and Figure 4–13.

Description

The **versus** function makes dependent data the independent values of the plot of a waveform. The versus function is entered in an expression; the first argument is a waveform whose dependent values will supply the dependent (Y) values on the plot and the second argument is a waveform whose dependent values will supply the independent (X) values on the plot. The resulting waveform is more easily displayed in a new or blank plot window

<*expression1*> must be a waveform.

<*expression2*> must be a real valued waveform.

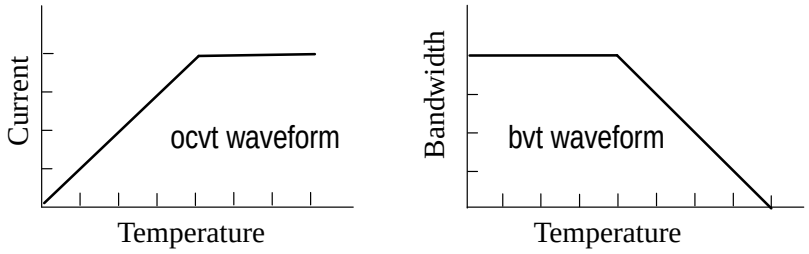


Figure 4–12: VERSUS Input Waveforms

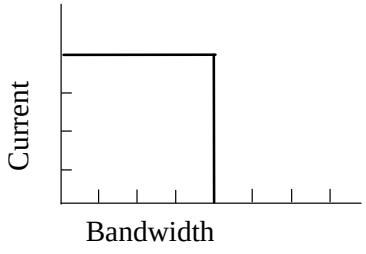


Figure 4–13: Output Waveform for VERSUS(ocvt, bvt)

WINDOW – perform a windowing operation on a waveform

Syntax

window(<*expression*>,<*type*>,<*val*>)

Examples

```
; u1 is frequency-domain waveform
u1=unfold(u1)
w1=window(u1,bh,0); blackman-harris window
tdomain=0.36*int(real(fold(ift(w1,aper))))
; normalize step response
```

or:

```
tran analysis time:0 10n 10n/127 ; do 128 point analysis
endtran

; v(1) is 128-point time-domain waveform
w1=window(v(1),kb,2); Kaiser-Bessel window with parameter=2.
fdomain=fft(w1,per) ; no "coherent gain" factor needed
```

Description

Window performs a windowing operation on the input waveform. The term ”window” refers to sampling windows as described in digital signal–processing literature. **Window** is usually applied to a time–domain waveform to insure that the extreme left and right edges of the variable are zero–valued and have continuous derivatives. Performing this operation on a time domain variable, then performing an **fft** on the result, yields a variable in the frequency domain that has characteristics similar to those of a finite–bandwidth spectrum analyzer. Performing a window operation also insures accurate vertical–scale frequency–domain responses alleviating ”leakage” as referred to in the literature. The **window** function can be used in either the time or frequency domains to insure that no sharp discontinuities are present in the periodicity that the Discrete Fourier Transform algorithm assumes.

<*expression*> is an expression based on a real–valued n–dimensional waveform. The input must have equally–spaced points (linear sweep); for a family of curves, this restriction applies to the inner sweep.

<*type*> determines which type of window is used. There are two choices; ”bh” and ”kb”. If <*type*> is ”bh” (for Blackman–Harris, then the argument <*val*> must be ”0”. If <*type*> is ”kb” for Kaiser–Bessel then, <*val*> must be specified as 2, 2.5, 3, or 3.5. The characteristics of the two types of windows are shown in Table 4–6.

Table 4–6: Windowing Characteristics

<type>	<val>	Sidelobe Level	6db Bandwidth[1]	Coherent Gain
kb	2.0	–46 dBc	1.99*[(n–1)/n*(xmax–xmin)]	0.49
kb	2.5	–57 dBc	2.20*[(n–1)/n*(xmax–xmin)]	0.44
kb	3.0	–69 dBc	2.39*[(n–1)/n*(xmax–xmin)]	0.40
kb	3.5	–82 dBc	2.57*[(n–1)/n*(xmax–xmin)]	0.37
bh	0.0	–92 dBc	2.72*[(n–1)/n*(xmax–xmin)]	0.36

1 xmax = last value of sweep variable
xmin = first value of sweep variable
n = 2^N number of points, 4<N<15 (16,32,64,...16384)

As shown in the table, the Blackman–Harris window (<*type*>=”bh”) offers a very low sidelobe level but relatively wide bandwidth. The Kaiser–Bessel window, on the other hand, offers narrower bandwidths; the cost is higher sidelobe level. A low sidelobe level is necessary when trying to compare two or more signals of much different amplitude. Lower sidelobe levels also mean that the noise floor of the spectra will be low, thus resulting in a high dynamic range. A narrow bandwidth is required to be able to discern signals that are close together in frequency.

The factors shown in the table as ”Coherent Gain” are used to equalize the response when **window** is applied to frequency–domain data. This occurs as part of the **window**, **ift**, and **fold** sequence to transform a variable from the frequency domain to the time domain. An example is shown with the **ift** description. Coherent gain is inherent in windowing, reducing the energy of the result. This effect is automatically equalized by the program so that when windowing is applied in the time domain, no further action is required by the user: the program automatically equalizes the energy loss by multiplying the result by a scale factor. The value of the scale factor is matched to the type of window utilized. However, when windowing is applied in the frequency–domain, the action of the program, appropriate in the time domain, inflates the result and requires correction. Values for this correction are what are called Coherent Gain in the table and are applied before **ift** of a variable that has been operated upon by the **window** function.

XREF – shift the x=0 reference for a waveform

Syntax

xref(<*expression*>,<*s*>)

Examples

```
; v(1) is time-domain variable with domain 0 to 10 p

newv1=xref(v(1),2.5p) ; domain of newv1 is -2.5p to 7.5p
```

Description

The **xref** function redefines the variable x–axis values. The new x=0 reference point is shifted to the position of the old x–value <*s*>, a point of the innermost independent variable of the input variable.

<*expression*> is an expression based on a real–valued n–dimensional waveform.

These following constraints apply to the use of **xref**:

1. The argument <*val*> must be within the original domain of the input variable.
2. The input variable must be based on a linear sweep.
3. For a family of curves, the above constraints apply to the inner sweep.

The **xref** is generally used to shift a time–domain waveform so that its phase is set to match some condition. An example is the phase reference plane for a time domain reflectometer. An **fft** of a time–domain variable whose phase has been shifted using the **xref** command will show a commensurate phase shift in the frequency–domain.

Commands

This section contains descriptions for commands used for analysis output. The commands are described in alphabetic order.

Each command is executed as it is read, statement-by-statement, except plot blocks. Execution of a plot block is deferred until the statement that signals end-of-block, the **endplotb** statement is encountered. Plot blocks are discussed under **plotb** in this section. The commands are given in Table 4–7.

Table 4–7: ADS Commands

Command	Description
!	Shell escape
error( )	Execution termination
finalvoltages	Print final swept voltages to a file
four	Compute discrete fourier analysis of waveform
fourcos	Compute discrete fourier analysis of waveform
if <expression>	Conditional execution
list	Print variable from plot file
nv	Print specified operating points from file
nvall	Print all operating points from file
overridelibrary	Over ride library with specified library
plotb	Plot block
plot	Plot waveforms
print	Print waveform
probe	Load variables from output files into plotter
tfprintf	write string to a file
tprintf	write text to a window
while <expression>	Repetitive execution

Command Format Conventions

Labels that indicate what the user is to supply (names, expressions, etc.) are printed inside angle brackets (<>). Square brackets ([ ]) surround *optional* entries. Do not enter the brackets or labels when typing a command, only the desired keywords and user-supplied names or expressions. Some command format specifications are too long to be printed on a single line and use multiple lines. This does not imply the user could enter the command on multiple lines

without using the continuation character — the statement would be printed on one line if it would fit.

! – Shell Escape Command

Syntax !<*operating system command*>

Examples

```
!ls /login/user/*
```

Description The **!** (shell escape) command transmits the remainder of the line to the operating system for execution. Semicolon and comments should not be added after the system command because the entire line is passed to the operating system. At the present time this command opens a ”c shell” process before passing the command line.

ERROR – Execution termination

Syntax Error(”<*string*>”,<*value*>)

Examples

```
error("%d is not a legal value",n)
```

Description The **error** command terminates a TekSpice simulation after execution. The syntax is the same as for the **tprintf** command. The **error** command brings up a notifier dialog box with the message <*string*> and terminates TekSpice.

FINALVOLTAGES – Print final swept voltages to a file

Syntax

finalvoltages [: f=<*file*> [format=initdc or inittr]]

Examples

```
finalvoltages : f=allnv format=initdc
```

Description

A **finalvoltages** subcommand causes ADS to print the last circuit node voltage of a swept analysis (Such as a transient analysis) to an ascii file <*file*>. These values are extracted from the binary plot file generated by the swept analysis.

The output of the **finalvoltages** command must be directed to a named file using the **f** parameter.

The **format** parameter causes the output to be printed in the form of **initdc** or **inittr** statements.

A **finalvoltages** command enables saving the final node voltages of a circuit and later using them as initial conditions for other analyses.

If the **finalvoltages** command is used in a var sweep command the values at the end of the last analysis are saved.

The example causes all node voltages to be saved as **initdc** statements. The form of node names is specified in the *Syntax* section of the *TekSpice Reference Manual*.

FOUR – Compute Discrete Fourier Analysis on Waveform

Syntax

four <*output1*> [<*output2*> ...] : fund=<*value*> [f=<*outfile*>]

Examples

```
trans analysis time:0 4u 1u/101 ;makes step size 1/101 of
; fundamental period
endtran
four v(out):fund=1/1u f=fourand ; period of fundamental is 1 uS
```

Description

The **four** command causes ADS to perform a discrete Fourier analysis on the specified waveforms. **Four** is identical to the **fourcos** command except that zero degrees phase is a sine waveform. The dc component and first ten harmonics of the fundamental are computed by evaluating the Fourier-series coefficients for the input waveform over the last full period of the waveform (determined by the **fund** parameter). A reported phase of zero degrees (at a particular harmonic) implies a pure sine wave with no delay.

<*output1*>, <*output2*>, etc., are node voltages or branch currents produced by a transient analysis.

Fund determines the fundamental frequency at which the analysis is performed and the time interval over which the analysis is performed. This interval is the reciprocal of the value of **fund** and ends at the last point of the waveform. The **freq** command can be used to compute the fundamental, however this computation is sensitive to the precision of **fund**. Providing an exact value will yield a more accurate result.

For best results, input waveforms should contain 101 data points per period of the fundamental. Otherwise, **four** will use interpolation to get the required 101 points. This can introduce harmonic distortion. A convenient way to specify the correct step size is shown in the example above. Here the transient time step is expressed as the reciprocal of the value of **fund** divided by 101. The value of **fund** and the analysis step size can be chosen such that an integer multiple of 101 analysis points (202, 303, ...) fall within the time interval analyzed by **four** without introducing distortion.

The higher frequency content of the input waveform should be considered when specifying **fund**: frequencies 50 times greater than the value of **fund** may result in aliasing to lower frequencies, affecting the harmonic energies that are computed.

The stop time of the transient analysis should be set to allow for at least one more point than the number of analysis points within the period of the fundamental. If you want to produce 101 points in the time interval analyzed by **four**, the analysis should include at least 102 points, so only one end point of the period is analyzed by the **four** co**mmand.**

<*outfile*> is the filename where the output of **four** is written. If <*outfile*> is not specified, the output of **list** goes to a new text window.

FOURCOS – Compute Discrete Fourier Analysis on Waveform

Syntax

fourcos <*output1*> [<*output2*> ...] : fund=<*value*> [f=<*outfile*>]

Examples

```
trans analysis time:0 4u 1u/101 ;makes step size 1/101 of
; fundamental period
endtran
fourcos v(out):fund=1/1u f=fourand ; period of fund. is 1 uS
```

Description

The **fourcos** command causes ADS to perform a discrete Fourier analysis on the specified waveforms. **Fourcos** is identical to the **four** command except that zero degrees phase is a cosine waveform. The dc component and first ten harmonics of the fundamental are computed by evaluating the Fourier-series coefficients for the input waveform over the last full period of the waveform (determined by the **fund** parameter). A reported phase of zero degrees (at a particular harmonic) implies a pure cosine wave with no delay.

<*output1*>, <*output2*>, etc., are node voltages or branch currents produced by a transient analysis.

Fund determines the fundamental frequency at which the analysis is performed and the time interval over which the analysis is performed. This interval is the reciprocal of the value of **fund** and ends at the last point of the waveform. The **freq** command can be used to compute the fundamental, however this computation is sensitive to the precision of **fund**. Providing an exact value will yield a more accurate result.

For best results, input waveforms should contain 101 data points per period of the fundamental. Otherwise, **fourcos** will use interpolation to get the required 101 points. This can introduce harmonic distortion. A convenient way to specify the correct step size is shown in the example above. Here the transient time step is expressed as the reciprocal of the value of **fund** divided by 101. The value of **fund** and the analysis step size can be chosen such that an integer multiple of 101 analysis points (202, 303, ...) fall within the time interval analyzed by **fourcos** without introducing distortion.

The higher frequency content of the input waveform should be considered when specifying **fund**: frequencies 50 times greater than the value of **fund** may result in aliasing to lower frequencies, affecting the harmonic energies that are computed.

The stop time of the transient analysis should be set to allow for at least one more point than the number of analysis points within the period of the fundamental. If you want to produce 101 points in the time interval analyzed by **fourcos**, the analysis should include at least 102 points, so only one end point of the period is analyzed by the **fourcos** command.

<*outfile*> is the filename where the output of **fourcos** is written. If <*outfile*> is not specified, the output of **list** goes to a new text window.

IF – Conditional execution

Syntax

if(<*expression*>){
<*statements*>
}

or

if(<*expression*>){
<*statements*>
}else{
<*statements*>
}

Examples

```
i=1
if(i==1) {
 tprintf("true\n")
}

i=0
if(i==1) {
 tprintf("true\n")
} else {
 tprintf("false\n")
}
```

Description

The if statement allows defining conditional execution statements. The braces are part of the statements and must be placed as shown since the language terminates statements with an end of line (eol). The statements following the ’’if’’ condition are executed if the condition evaluates to a nonzero.

LIST – Print waveforms from Plot File

Syntax

list [<*plotfile*>] [: f=<*outfile*>]

Examples

```
list anlplt:f=traclist
```

Description

The **list** command causes ADS to print a summary of the waveforms (binary data kept during a sweep analysis) that are available in a binary plot file. The summary also contains the date of analysis and analysis title.

<*plotfile*> defaults to the binary data file (*bifile*, *dcfile*, *acfile*, or *trfile*) generated by the last analysis.

In the example above, waveform names produced by **list** are the names used in a **probe** statement to load data for printing or plotting.

<*outfile*> is the filename where the list is written. If <*outfile*> is not specified, the output of **list** goes to a new text window.

NV – Print Specified Operating Points from File

Syntax

nv <*node1*> [<*node2*> ...] [: f=<*outfile*> [format=initdc or inittr]]

Examples

```
nv sumjunc 5 27 OUTER.inner.52
```

Description

An **nv** command causes ADS to print the dc operating point voltages of nodes <*node1*>, <*node2*>, etc., contained in the binary plot file generated from an analysis.

By default the output of the **nv** command is directed to a new window. It can also be directed to a disk file <*file*> using the **f=** parameter.

The **format** parameter causes the output to be printed in the form of **initdc** or **inittr** statements.

The example above causes the top level circuit (OUTER) node voltages to be printed for nodes: sumjunc, 5, and 27. The voltages for node 52 in subcircuit inner are also printed. The hierarchial form for node names is specified in the *Syntax* section of the *TekSpice Reference Manual*.

<*node1*>, <*node2*> specifies the nodes at which the operating points are to be printed.

<*outfile*> is the filename where the output of **nv** is written. If <*outfile*> is not specified, the output of **nv** goes to a new text window.

NVALL – Print Operating Points from File

Syntax

nvall [: f=<*outfile*> [format=initdc or inittr]]

Examples

```
nvall : f=allnv format=initdc
```

Description

An **nvall** command causes ADS to print all the dc operating point node voltages contained in the binary plot file generated by the list analysis.

The **format** parameter causes the output to be printed in the form of **initdc** or **inittr** statements.

The example above causes all node voltages to be saved as **initdc** statements. The hierarchial form for node names is specified in the *Syntax* section of the *TekSpice Reference Manual*.

<*outfile*> is the filename where the operating point data is stored. If <*outfile*> is not specified, the output of **nvall** goes to a new text window.

OVERRIDELIBRARY – substitute a different file for external library

Syntax

overrideLibrary <*process–name*> <*library–name*> <*filename*>

Examples

```
; Example 1. First simulation with original library for
; this circuit, MY_LIB
tran analysis time : 0 4.2n 4.2n/511
endtran
probe v(ref)=refstd

; Override one library before 2nd simulation, use
; non-standard library
overrideLibrary MY_PROCESS MY_LIB ~/spcl/DEVICES.nonstd.bin
tran analysis time : 0 4.2n 4.2n/511
endtran
probe v(ref)=refnonstd
plot refnonstd refstd

; Example 2. First simulation with original library for
; this circuit, GSATALL
tran analysis time : 0 4.2n 4.2n/511
endtran
probe v(38)=v38nominal

; Override one library before 2nd simulation, use GSATALLU
overrideLibrary GST_CUSTOM GSATALL[1]
 ~cae/lib/ICO/GST_CUSTOM/GSATALL/GSATALLU.3c2.bin
tran analysis time : 0 4.2n 4.2n/511
endtran
probe v(38)=v38up

; Another override, before 3rd simulation, use GSATALLD
overrideLibrary GST_CUSTOM GSATALL[1]
 ~cae/lib/ICO/GST_CUSTOM/GSATALL/GSATALLD.3c2.bin
tran analysis time : 0 4.2n 4.2n/511
endtran
probe v(38)=v38down
v38=waveform(buildArray(dependentData(v38down),
                        dependentData(v38nom),
                        dependentData(v38up)),
                        "lib",
                        {-1,0,1},
                        "time",
                        independentData(v38nom))
plot v38
```

[1] **No carriage return allowed here.**

Description

The **overridelibrary** command overrides the file associated with an external library. Each external library has exactly one binary ”device library” file associated with it. The name of that device library file can be seen by bringing up a **Change Library** Dialog Box for the library.

Overridelibrary will temporarily change the name of the device library file associated with the given library. The library is identified by its process name followed by the library name. The filename specified on the command should be an absolute path, that is, it should start with either the ”~” or the ”/” character.

The change of file association lasts only until the next overridelibrary command applying to the same library, or until the end of the control program. Every simulation starts with the file name originally specified for the library. That original file name is the one shown in the **Change Library Dialog Box**. The override does not carry over from one simulation to another.

In Example 2, three simulations are run. When it simulated from the **Control Program Browser,** three experiments are created. One voltage waveform is probed from each of the three experiments in that example. A family of three waveforms is then plotted using the **waveform** and **buildarray** functions. Each waveform comes from a simulation with a different library.

To view additional waveforms after the family of curves is plotted, the correct experiment must be selected in the *Experiment Table Pane*. The waveforms can then be plotted for that single experiment. To compare additional waveforms between experiments, the three individual waveforms must be obtained by selected the three experiments; one at a time.

<*filename*> is the filename where the library is stored.

See Also

Library Browser Dialog Box.

PLOTB – Plot Block

Syntax

plotb [: [title=<*title*>] [xaxis=lin or log] [xmax=<*xmax*>] [xmin=<*xmin*>]]
; plot commands go here to force multiple waveforms onto the same
; plot window

endplotb

Examples

```
plotb
plot v(b)-v(e) vref ; appropriate vertical scales used for
                    ; each plot statement, all waveforms on
                    ; screen at the same time

endplotb
```

Description

A **plotb** command is the beginning statement for a plot block. This block contains one or more plot statements. These statements display multiple waveforms in a single ADS plot window. The **plotb** statement is terminated by an **endplotb** statement.

<*title*> is the title which appears on the plot. This is the default title which will appear on all plots in the **plotb** block. The title string must be surrounded with quotes (“.....”) if spaces appear in the string

<*xmin*>, <*xmax*> determine the minimum and maximum scale values on the x axis.

PLOT – Plot Waveform(s) in ADS window

Syntax

plot <*expression*> [<*expression*> ...] [vs <*expression*>]: [title=<*title*>] [xaxis=lin or log] [xmax=<*xmax*>] [xmin=<*xmin*>] [yaxis=lin or log] [ymax=<*ymax*>] [ymin=<*ymin*>]]

Examples

```
; Example 1. Set x and y axis max and min.
plot v(3): xmin=1n xmax=10n ymin=.2 ymax=.5

; Example 2. Plot magnitude of v(3)
plot mag(v(3))

; Example 3. Plot phase of v(3)
plot phase(v(3))

; Example 4. Plot db and phase of v(3)
plot db(v(3))
plot phase(v(3))

; Example 5. Start plot block and set x axis and title
; for all plots in block.
plotb:xaxis=log xmax=1.1 title="m999 test"
plot v(b)-v(e) vref ; appropriate y axis scale used for
plot phase(v(out))   ; each plot statement
endplotb

; Example 6. Plot log-log Transistor dc beta vs IC
plot versus(IC/IB,IC):xaxis=log yaxis=log

; Example 7. Plot operating point voltage and Q1 collector
;current
plot bv(1) bi(i(Q1,1))

; Example 8. Test circuit for 2N3904, sweep eb current and
ce voltage.
libpath translib
circuit ibeval
Q1 c b e 2N3904 model in 'translib'
ibeele e b i: dc=ibeval
vceele c e v: dc=dcin
endc
var sweep ibeval: 2u 8u 0.5u
dc analysis dcin: 0 4 0.04 enddc
endvar
probe i(vceele,1)=ic
plot ic ; plots family of curves
```

```
cvar=4 ; picks value of an outer sweep variable of 3-d sweep
pvar1=reduce(v(24),cvar) ; plots 2-dimensional family of curves
plot v(24)

var sweep temp : 27 107 30 ; same circuit as previous var sweep
var sweep ibeval: 2u 8u 0.5u

dc analysis dcin: 0 4 0.04
enddc

endvar
 endvar
temp=27;fix value of outer sweep to reduce 3D to ;2D for plotting
ic27=reduce(i(vceele,1),temp) ; result is family of curves
plot ic27
```

Description

The **plot** command plots waveforms <*expression1*>,<*expression2*>,etc., generated by swept analyses to an ADS plot browser.[2] The first example above includes an expression for the ratio of voltages at node 1 and node 2. The third example includes an expression for the difference between voltages at nodes, b and node e.[3]

Waveforms can be plotted, from analysis results files stored on disk, from SPICE **plotdt** files or files created by other applications. Waveforms can also be plotted that were created by assignment of an expression based on one or more waveforms.[4] These waveforms are loaded into ADS using the probe command. In this case names used in the **plot** command must match those used in the **probe** command.

The horizontal axis is different for each analysis type: for dc analysis, the axis is “dcin”, for ac analysis, “freq”, and for transient analysis, “time”. Additional x-axis variables (such as another waveform) can be specified by the **versus** function. In the example above, tramsistor collector current ,’ic’ and base current ’ib’ are plotted. This example shows how the **versus** function is used to plot one waveform versus another. The waveform named after **versus** must be one-dimensional and real.

<*expression*> can be a waveform, or a combination of waveforms and scalar variables, constants, operators, or functions.

<*title*> is the title which appears on the plot ; the default title is the current title from the **title** command. The title string must be surrounded with quotes (“.....”) if spaces appear in the string

2. **Expressions that evaluate to a scalar can not be plotted; instead use *print*.**
3. **Forms used in SPICE, such as V(1,2) for the difference between nodes 1 and 2, are not accepted on probe or plot statements; see probe in this section for the syntax for node voltages and branch currents on probe and plot statements.**
4. **Functions used in expressions are listed in the Syntax section. Included are functions for complex and real numbers, absolute value, square root, trigonometric functions, etc.**

xaxis controls the scale type for the x-axis as either linear or log scale. The default is the type of sweep used for the analysis.

<*xmin*>, <*xmax*> determine the minimum and maximum scale values on the x axis.

yaxis controls the scale type for the y axis as either linear or log scale (the default is linear).

<*ymin*>, <*ymax*> determine the minimum and maximum scale values on the y–axis.

PRINT– Print real values of waveforms

Syntax

print <*expression1*> [<*expression2*> ...] [: [f=<*outfile*>] [format=<*plotdt* or *free* or *pwl* or *sweep* or *tek11k* or *tektds* or *tekspice* or *workspace*] [numsigd=<*number*>] [xmax=<*xmax*>] [xmin=<*xmin*>]]

Examples

```
print v(1) v(2)/v(1)
print db(v(2)/v(1))
print phase(v(2)/v(1)): f=phasev2overv1 format=sweep
```

Description

The **print** commands are used to print <*expression1*>, <*expression2*>, etc., which are waveforms kept during analysis. The operation of the print command is similar to the **plo**t command.

<*expression1*>,<*expression2*> are waveforms, or a combination of waveforms and scalar variables, constants, operators, or functions.

<*outfile*> is the filename where the output of **print** is written. If <*outfile*> is not specified, the output of **print** goes to a new text window.

See the **probe** command in this section for a description of the **plotdt** and **free** formats.

pwl format produces a piece wise linear expression suitable for inclusion in a TekSpice expression assignment.

sweep format prints the waveform values separated by tabs as follows:

```
v(1)x1      v(1)y1      v(2)y1 ...   v(n)y1
v(1)x2      v(1)y2      v(2)y2 ...   v(n)y2
  .           .           .            .
v(1)xm      v(1)ym      v(2)ym  ...  v(n)ym
```

The waveforms must have the same independent axis.

tek11k format produces an expression of the same form as created from a Tektronix 11000 family oscilloscope by dumping **WFMpre** and curve data in ASCII format in verbose mode.

tektds format produces an expression of the same form as created from a Tektronix tdr family oscilloscope by dumping **WFMpre** and curve data in ASCII format in verbose mode.

tekspice format produces an expression in the form of a **waveform** TekSpice expression

waveform produces an expression of the form:

`a=((x1@y1) (x2@y2) ... (xn@yn))`

<*number*> is the number of significant digits printed in the output.

<*xmin*>, <*xmax*> determine the minimum and maximum scale values on the x axis.

PROBE – Load Variables from Output Files into Plotter

Syntax

probe <*par1*>[=<*newname*> <*par2*> ...] [: f=<*file*> [format=<*plotdt* or *free* or *sweep* or *tek11k* or *tektds* or *tekspice*>]]

Examples

```
probe I(VIN,2) V(12) V3_7 BV(N1) BI(X1,6) ONOISE:f=firstrun

probe V(OUTER.INNER.3)=IN3 I(OUTER.INNER.VSS,1)=ISS

probe V(1) VM(2)=VM2:format=plotdt f=plotdt

probe a=k: f=datafile format=sweep; read data from external file
             that is in 2 column format

probe DCIN=DCSTEPS : f=dcfile ; values of independent variable
probe FREQ=F : f=acfile ; take the real part of FREQ

fsteps=real(f) ; to extract frequency steps of ac analysis

probe S(V(101),R1.R)=SR1 ; ac sensitivity of V(101) to
; resistance R1
```

Description

The **probe** statement loads output variables from analysis output files (sometimes called “binary plot files”) making the waveforms available for printing or plotting.

Output variables are either waveforms (sets of x-y values that can be plotted) or scalar variables (single valued).

Waveforms are generated by either 1) a sweep analysis (ac, dc, or transient analysis) or 2) by operating point analysis performed within a variable sweep block. Other waveforms can be probed after a distortion, noise, or an ac sensitivity analysis. These outputs are described in the **noise** and **acsens** commands in the *TekSpice Reference Manual*.

Scalar variables are generated by operating point analysis that is performed once (outside a variable sweep block), either as **bias** analysis, or as the first point in a dc sweep, as the operating point for an ac sweep, or as the initial transient solution.

<*par1*>**,** <*par2*> take any of several forms: waveforms, bias voltages or branch currents, independent variables, ac sensitivities, or variables names. If renamed by <*par*>=<*newname*>, the new name is used in printing, plotting, or assignment commands.

<*file*> is the file where the input for **probe** is stored.

sweep format reads values separated by tabs as follows:

```
v(1)x1      v(1)y1      v(2)y1 ...   v(n)y1
v(1)x2      v(1)y2      v(2)y2 ...   v(n)y2
  .           .           .            .
v(1)xm      v(1)ym      v(2)ym  ...  v(n)ym
```

The values must have the same independent axis.

tek11k format reads the preamble and curve data from the file <*file*> and creates the waveform in variable <y>. It is assumed that the file <*file*> was created from a Tektronix 11000 family oscilloscope by dumping **WFMpre** and curve data in ASCII format in verbose mode.

Waveforms created by a variable sweep can be multi-dimensional, either a family of curves or a series of families of curves. The **probe** statement automatically dimensions the variables loaded to include all waveforms in a multi-dimensional variable.

Waveforms

Waveforms are voltages and currents computed by a swept analysis, either ac, dc, or transient.

V(<*node*>) is a waveform of voltage at a node calculated at each step during ac, dc, or transient analysis. In the first example above, V(12) is the voltage at node 12. V(OUTER.INNER.3) names the voltage at node 3 inside the subcircuit definition for the subcircuit instance INNER, which is nested in the definition called by instance OUTER.[12] The name is shortened by assigning the name IN3, which is used for output (PRINT or PLOT) or assignment statements.

I(<*element or subcircuit instance*>,<*term#*>) is a current in a branch of the circuit calculated during a sweep analysis. In the first example above, I(VIN,2) is the current entering the second terminal of element VIN. The positive direction is into the element. I(OUTER.INNER.VSS,1)=ISS is the current entering terminal one of an element called VSS inside two levels of subcircuit instances, INNER and OUTER. The waveform is renamed to ISS. If VSS were a subcircuit instance instead of an element, ISS would be the current entering the terminal from the first node in the node list for the instance.

Bias Voltages and Branch Currents

Operating point voltages and currents are produced by **bias** analysis, as the first step in a dc swept analysis, as the operating point for an ac analysis, or as the initial transient solution. If produced within a variable sweep, these voltages and currents can be plotted.

BV(<*node*>) is an operating point voltage at a node.[5] In the first example, BV(N1) is the operating point voltage at node N1.

5. Compound names for nodes and elements within subcircuits are further described under Node Names in the TekSpice Reference Manual.

BI(<*element or subcircuit instance*>,<*term#*>) is a branch current at the operating point. In the first example, BI(X1,6) is the operating point current entering the sixth node in the node list of subcircuit instance x1.

Analysis Variables

<*ind var*>=<*var*> loads the analysis steps as a waveform by probing the independent variable. The independent variable must be renamed on the **probe** line. Two examples are shown for the ac and dc analysis steps; TIME, the independent variable in transient analysis, and the independent variables named for variable sweeps could be probed in the same manner. FREQ is returned as a complex number: the real part is frequency and the imaginary part is 2πf.

Variable Names

<*name*> is any legal variable name assigned on a **keep** or **keepi** statement or waveform name produced by distortion analysis; V3_7 and **onoise** the first example are examples of both.

Parameters

The file named on the **f** parameter names the source of the data. The default is the file produced by the last sweep analysis, ac, dc, or transient. The default file is the last analysis output file produced during the simulation session. It is listed in the **list** command output.

Format can be assigned the values **plotdt**, **free**, **sweep**, **tek11k**, **tektds** or **tekspice**. If not specified, the file is assumed to be analysis files output by a TekSpice2 analysis in a binary format. **Plotdt** files are ASCII text files in a fixed format output by TekSpice and plotted by the program **plot**. Files containing the statement **format=free** are similar to **plotdt** files, but in a less restrictive format which allows comments and number fields which are not fixed. This format facilitates loading measured data for plotting. The **sweep** format is a format in which one to n–columns of data are separated by tabs similar to that output by spreadsheets. The **tek11k** format [6]is the format output by the Tektronix 11000 family of oscilloscopes.

FORMAT=FREE Example

A file in free format is listed here. It contains data from four waveforms. Comments, which start with a semicolon, explain the meaning of the lines. Add or delete lines that relate to multiple waveforms to fit other needs. The format starts with a line for the title, a line identifying the number of waveforms, and lines for the waveform labels and the x-axis label, then y and x values in a free format. Lines with particular elements are required to delimit the labels and y and x values.

```
Ft data for Vce = 4, 2, 1, .5 ; title
4 27 ; # of waveforms, analysis temperature (printed on plot)
1 ; this pair (this and next line) are repeated for each waveform
V4 ; y-axis label for first waveform
1
```

6. See the appropriate 11000 or TDS Family *Programmer Manual*.

```
V2 ; y-axis label for second waveform
1 V1 ; y-axis label for third waveform
halfV ; y-axis label for fourth waveform
Ic ; x axis label
10 ; # points/waveform must be same for each waveform & for X's
1 1 ; any two numbers--not used; y values for first waveform
follow:
1.14E9 2.06E9 3.43E9 4.39E9 ; integers, floating point,
5.65E9 ; or sci. notation ok
6.95E9 ; white space ok and can be spread over multiple lines
7.67E9 7.77E9 7.43E9 6.11E9
2 2 ; beginning of second waveform values
1.14E9 1.96E9 3.27E9 4.12E9 5.2E9 6.27E9 6.79E9 6.78E9 6.30E9
4.57E9
3 3 ; third waveform follows
1.11E9 1.86E9 3.03E9 3.75E9 4.63E9 5.41E9 5.65E9
5.47E9 3.98E9 1.29E9
4 4 ; start of fourth waveform
1.06E9 1.80E9 2.77E9
3.35E9 3.97E9 4.43E9 3.97E9 2.13E9
0.62E9 0.45E9
0.5M 1.0M 2.0M 3.0M 5.0M 10M 20M 30M 50M 70M ; X axis values for
all waveforms
```

Waveforms in a free-format file are loaded and plotted by **probe** and **plot** commands. ADS shifts all upper–case characters to lower case and appear as lower–case in the **plot** line.

```
probe V4 V2 V1 halfV : f=mdata format=free
plot v4 v2 v1 halfv
```

TFPRINTF – write a string to a file

Syntax

tfprintf(”<*filename*>”,”<*string*>”, <*argument1*>[,<*argument2*>...])

Examples

tfprintf(”file1”,”hello\n”)
tfprintf(”file1”,”there\n”)
tfprintf(”file1”,”%d %e %f \n”, 1, 1.0, 1.0)’

Description

Write a formatted string,<*string*>, to a file.

<*filename*> is the name of the disk file, enclosed in double quotes, where <*argument1*>,<*argument2*>, ... are written.

<*string*> is a text array containing a group of alpha–numeric characters enclosed by double quotes (”). <*String*> can also include C language style print statements as shown in Table 4–8. The \n representing the new line character and \t representing the tab character are also allowed.

Table 4–8: Allowable Format Statements for TPRINTF

Format	Example	Desccription
%<n>d	%2d\n	prints an n digit decimal integer
%<n>e	%6e	prints a folating point of double precision number in the format [–]m.nnnnnnE±xx. N defaults to 6.
%<n>s	%2s	prints a string of characters with length determined by n.
%<n>f	%6.2f	prints a floating point or double precision number in the format [–]mmm.nnnnn. N defaults to 6.

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

TPRINTF – write formatted data to to a workspace window

Syntax

tprintf(”<*format string*>”, <*argument*1>[,<*argument2*>...])

Examples

```
tprintf(”hello\n”)
tprintf(”there\n”)
x=3.5
tprintf(”%d %e %f \n”, 1, x, 1.0)
```

Description

Tprintf is a formatted print function which opens and writes to an ADS text window. The format statements are similar to those in the C programming language. This includes **\n** for printing a carriage return.

<*format string*> is enclosed by double quotes and define the format used to write <*argument*>. See Table 4–9. The \n represents the newline character.

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

Table 4–9: Allowable Format Statements for TPRINTF

Format	Example	Description
%<n>d	%2d\n	prints an n digit decimal integer
%<n>e	%6e	prints a floating point or double precision number in the format [–]m.nnnnnnE±xx. N defaults to 6.
%<n>s	%2s	prints a string of characters with length determined by n.
%<n>f	%6.2f	prints a floating point or double precision number in the format [–]mmm.nnnnn. N defaults to 6.

<*argument*> is an expression based on a real–valued one dimensional (single waveform) or two or more dimensional (family of waveforms) waveforms.

WHILE – Repetitive execution

Syntax

while(<*condition–expression*>){
<*statements*>
}

Examples

```
n=0
while (n<5) {
  tprintf("n=%d\n",n)
  n=n+1
}
```

Description

The **while** statement specifies that a sequence of <*statements*> is to be executed repeatedly zero or more times. The <*condition–expression*> is evaluated and tested before each execution of <*statements*>. If <*condition–expression*> evaluates to non–zero, then <*statements*> is executed and the **while** is re–executed. If <*condition–expression*> evaluates to zero, then the **while** statements execution is terminated.

Cursors

There are several different mouse cursors used by ADS to give the user feedback about the current mouse position, ADS's activities, the memory managers activities, as well as a prompt for user input.

Cursors Showing ADS Activity

Normal – This is the default cursor. The point of selection (or *hot spot*) is the tip of the arrow.

Bulls Eye – ADS is requesting a point selection during a zoom operation.

Drag – ADS is dragging a visual object such as a wire. This operation is usually part of a paste operation or creation of a visual object.

Execute — ADS is working on a task initiated by the user. During this time, mouse inputs are ignored and keyboard inputs are deferred until the task is finished.

IBeam – ADS is editing text in a graphics window.

Read – Data is being read from an external file. During this time ADS does not respond to keyboard or mouse inputs.

Wait – ADS is carrying out an operation, such as saving the image. During this time ADS does not respond to keyboard or mouse inputs.

Write – Data is being written to an external file. During this time ADS does not respond to keyboard or mouse inputs.

Cursors Showing Memory Manager Activity

Compacting garbage collector – A full compacting garbage collection is taking place. This occurs only when the ADS process size is close to the memory allocation limit. During this time ADS does not respond to keyboard or mouse inputs. See the ***Note*** on ***real memory allocation*** in the **System Configuration Dialog Box** writeup in this chapter.

Garbage collector – Non–compacting garbage collection is taking place. This occurs only when the ADS process size is close to the memory allocation limit. During this time ADS does not respond to keyboard or mouse inputs.

Memory compactor – Memory is being compacted. This occurs when the free memory space is heavily fragmented. During this time ADS does not respond to keyboard or mouse inputs.

Procedures

Adding Parasitic Devices in ADS

Parasitic devices can be added to ADS using two methods:

1. Adding them directly to the schematic.
2. Linking a netlist of the parasitic devices as a file (shunt devices only).

Adding Parasitic Devices to Schematic

The first method requires adding the parasitic devices to the schematic as if they were any other device. If the resulting circuit is an integrated circuit, and LVS software such as QuicKic will be used for layout, the netlist is first passed to SIMTOQ8 either directly or from ADS using the **Netlist to Layout Dialog Box**. SIMTOQ8 removes the parasitic devices from the netlist if they are specified as follows:

1. The following TekSpice primitive devices models can be used as parasitic elements; r, l, c, v, i.
2. The second character of the device name must be a ”p” (e.g. rp23 1 4 r: r=100). Upper case P can also be used.

In the netlist all Rp* and Lp* devices are shorted and Cp* devices removed. Subcircuits with ”p” as a second character are not removed from the netlist.

Adding a Parasitic Device File

Shunt parasitic devices can be added to the circuit netlist as an attached file without the need to include them in the schematic. In TekSpice versions 2D or later this is done as follows:

1. Select a circuit in the *Circuits List Pane* of the **Circuit Browser**.
2. Select the ’configure parasitics’ menu item. (This menu item is after ’set libraries’.)

This produces a dialog box with the prompt ’Enter parasitics file name’.

3. Enter the name of the file containing the parasitic elements and press ’ok’.

The following are required of the parasitic circuit file:

- **a.** The parasitics file must contain only circuit elements which reference nodes from the specified hierarchical circuit and primitives.
- **b.** If a relative file name is given then it is assumed to be relative to the simulation–results directory.
- **c.** The file must be readable by the machine running TekSpice2.

d. The element names must all be unique including the element names from the top level of the hierarchical circuit.

See Also

QuicKic 8 User Manual for details on how to extract parasitic capacitors from a QuicKic layout.

ADS PICT Print Format Conversion to Interleaf

This section outlines a technique for including high resolution PICT format graphics, taken from ADS into Interleaf documents. This applies to ADS versions 5a0 and higher.

ADS Configuration Requirements

The procedure for configuring ADS for printing PICT graphics is as follows:

1. From the Launcher window select the commands, **configure > printer(graphics)**
2. Select the **Style button: PICT** and **page size button A.** The other options do not effect the output.
3. Enter the following text in the script pane(approximately):

```
# pict template
# commands executed with /bin/csh
/interleaf/ileaf5/bin/macdrfilt -desktop -overwrite ]FILENAME
```

The last line (the only executable line in the script) sends the PICT output from ADS through the MACDRAW filter and places the converted document on the Interleaf desktop.

Creating the Document

Select the ADS print menu in the normal manner. Be sure to select the 'send to printer' button. This causes execution of the print script (above) . The PICT format is then converted to Interleaf format, and moved to the Interleaf *desktop* directory as a filename specified from your print panel.

Interleaf Operations

At this point, the document is on the Interleaf desktop as a file (out.gpr by default). This file contains a single frame containing all the graphic data. The data scale probably exceeds the size of the standard Interleaf frame. It is normal for the entire document to appear white when first viewed. The following list of Interleaf steps will get the document in a viewable form:

1. Select the converted document and use the pop–up button menu command **copy > normal** and copy to the Interleaf clipboard.
2. Move to the target document and use the pop–up menu command **Paste**.
3. Select the frame (this requires two clicks of the select mouse button). A broad grey line appears.
4. Select the pop–up button command **Select > All**.
5. Select the pop–up button command **Size > ToFrame > Diagona**l. This scales the data to fit inside the frame without introducing any distortion.

6. Select the pop–up button command **Select > All**.

7. Select the pop–up button command **Edit > Edge > Weight.** Select the thinnest line available (the topmost line on the pop–up menu).This converts the thick lines produced by PICT to a line more suited to the laser printer. Heavier weight lines can be selected according to personal preference. The original line widths are typically too bold to read the text.

8. Resize and position the frame. The document is now included in Interleaf. Refer to the appropriate Interleaf manual for additional instructions.

ASIC Design Guidelines

The following are guidelines for ASIC designs with ADS/TekSpice:

- All subcircuits should be LIBRARY models, as opposed to LOCALLY DEFINED subcircuits. That is, no subcircuits with italicized names should exist in the *Model List Pane* of a **Circuit Editor** or **Subcircuit Editor**.
- The ASIC should be a subcircuit with a model name commensurate with the M–number of the IC (i.e., ”M777B”) or equivalent vendor–dependent nomenclature. This subcircuit is referenced from a top–level circuit definition that must also include; the circuit stimulus, power supplies, loading and control circuitry.
- The use of GLOBAL NODES including ground are not recommended within the ASIC subcircuit.
- The ASIC SUBSTRATE connection is always an External Node (terminal) of the ASIC subcircuit.
- Primitive names should not exist in any *Model List Pane.* Primitive instances should not exist in any circuit or subcircuit. Primitives are indicated by a bold font (i.e., **gp1, vccs, c**) in the *Model List Pane*.
- TekSpice primitive models (r, l, c, d, v, etc.) should not be used within the ASIC subcircuit except for parasitic devices. Parasitic device names have the character ”p” as the second character in the name.
- Finished ASIC designs should not be shipped using encapsulated libraries. If subcircuits are brought into the design through encapsulated EDIF files, they should be moved from the encapsulated libraries into personal libraries.
- Circuit variables can not be present in the final ASIC subcircuit design. If used in the design process, they should begin with the string ”cv_” to reduce confusion between circuit variables, local variables, and reserved variables.
- Variable assignments should be made in the *Control Program Text Pane* of the **Control Program Browser**. The *Experiment Table Pane* can be used for variable assignments, but these assignments are not retained with EDIF file transfers, or during copies of the control program in the *Control Program List Pane*.

ADS Library Model Guidelines

The following are recommended naming guidelines for ADS library models:

Model Names

Model names should be short, lower case font and unique. Duplicates should not exist unless they deliver the same functionality, have the same pin out and input/output parameters.

Instance Name Prefixes

Instance name prefixes should be in UPPER CASE font. Use the full model name unless a standard replacement is known (such as Q for transistor). If the model name ends with a number, the name prefix should have the underscore (_) character appended.

Model Label Style

The model name should not be displayed if the full model name is used in the name prefix. If a replacement name is used for the name prefix, the label style(s) **model name** option should be selected in the Label Style Dialog Box. In ADS versions 4e and later, both the model name and parameters can be displayed.

Model Parameters

Displaying model parameters clutters up the schematic. Select only the important parameters to display.

Input Parameters, Local Variables, Output Parameters

For the input parameters, local variables and output parameters (*Parameter List Selector Pane*) use lower case font unless both cases are needed. For example V and v.

Pad (Port) Count and Location

The number of pads (ports) and the graphics location of the pad connections should match models of a similar type. This is important if these models will be interchanged. An example would be several different four port transistor models.

Dimension Units

The fundamental units used in all model libraries should be in the base MKS system as:

Meters
Kilograms
Seconds
Volts
Amps
Hertz
Ohms
Henries
Farads

(This page intentionally left blank)

Using QuERC from ADS or Stand–alone

Querc is a rule–based electrical rule checker. It reads a TekSpice netlist and the resulting simulation answer file containing the dc bias and operating point information. Voltage and current rule checks are performed on all devices and subcircuits specified in the rule file. A rule can be written using algebraic expressions consisting of terminal (port) voltages, transistor currents, resistor and capacitor values, etc. An error report may be written to an output file. Reserved variable names cannot be used except as specified.

Using Querc from ADS

Querc can be invoked from ADS by including a querc analysis block in the **Control Program Browser**'s *Control Program List Pane.* This can be done automatically from the **Circuit Editor** by selecting the popup button command **add program > querc** from the *Simulation Status Pane* (upper right). The resulting **Control Program Browser** will appear as shown in Figure 4–14.

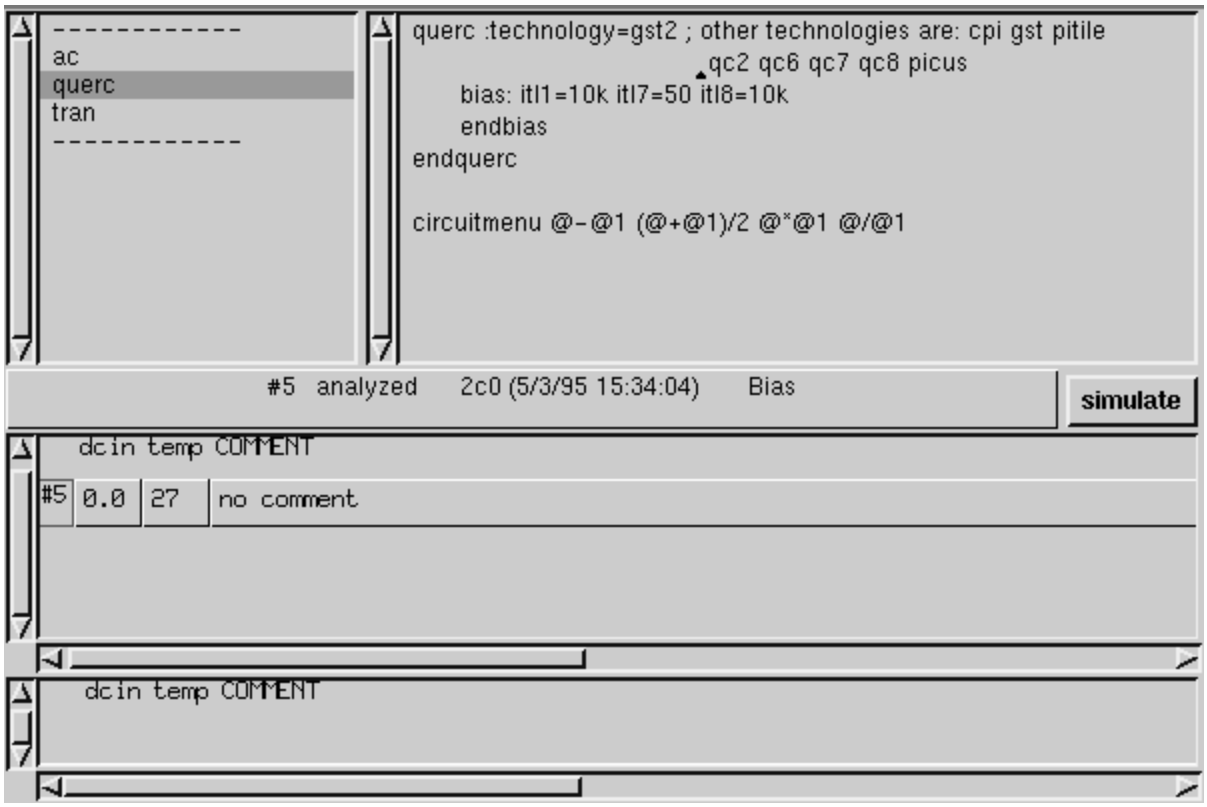


Figure 4–14: Control Program Browser Showing the Querc Analysis Block

With querc highlighted (as shown in Figure 4–14) press the simulate button. Results from the querc analysis are viewed in a text window using the popup button command **open > querc** text from the *Schematic Pane* of the **Circuit Editor** when in the Analysis Mode.

Using Querc Stand–Alone

Querc can be invoked without using ADS by including the appropriate parameters in the command line at a Unix prompt. The following description is copied from the **QuicKic Layout with Verification User Manual** published by Maxim.

The Syntax is as follows:

```
querc –<technology> –s <Netlist_Filename> –b <Bias_Filename> [–o
<Output_Filename>] [–r <Rule_Filename>] [–pspice] [–spice] [–v]
```

where:

–<*technology*> – where *technology* is gst, gst2, custom, etc.

–*s* <*Netlist_Filename*> – where *Netlist_Filename* is the name of the input TekSpice simulation netlist file. If a Spice netlist is used to generate a bias file, the netlist must be converted to TekSpice format using the program **spitek**.

–*b* <*Bias_Filename*> – where *Bias_Filename* is the name of the dc bias file containing the bias and operating point solut*ions.*

–*o* <*Output_Filename*> – where *Output_Filename* is the name of a file containing the resulting electrical rule violations. If this option is not specified, the violations are written to standard I/O.

–*r* <*Rule_Filename*> – where *Rule_Filename* is the name of the file containing the electrical rules to be checked. If this file is not specified, the default rule file is used.

The path to the default rule file for the gst technology is:

```
~tekcad/quickic8/prod/qkiclib/gst/QUERC.RC
```

–*pspice* – (flag) if present, indicates the dc bias file is in Pspice format.

–*spice* – (flag) if present, indicates the dc bias file is in Spice 2G format.

–*v* – (flag) if present, displays the version of **querc** to standard I/O.

Using Non–Interactive ADS

A non–interactive (no windows) version of ADS, adsni, can be used to process tekspice ADS workspace statements in a batch mode. For example, a file named extract.tsp containing:

```
probe v(1)=wave : f:/login/user/ads/SimulationResults.ADS/trfile66
print wave: f=filename format=sweep
```

can be evaluated using the following statement:

```
adsni –input extract.tsp
```

The non–interactive image will be invoked by a statement of the form:

adsni [–input <*tekspice–input–file*>] [–errorOutput <*error–file name*>] [–output <*output–file name*>] [–voe <*version #*>] [–vim <*image version #*>]

where:

–input <*tekspice–input–file name*> is the input file containing ADS workspace statements. The default input file is *adsni.tsp*. Use dash ”–” instead of filename to take input from standard input.

–errorOutput <*error–file name*> is the error file that will contain errors output by TekSpice. The default error file name is *adsni.err*

–output <*output–file name*> is the output file. The default output is *standard out*.

–voe <*version #*> is the version number appended to VW2.5ads to form Object Engine# name, replacing the default OE name.

–vim <image version #> the image version number is part of an image file name: /u/cae/adsniIVN.im where IVN is replaced by given image version number. This will replace the default image.

Examples:

```
adsni
adsni -input - -vim 5a1c < file
adsni -input file.tws -errorOutput file.err -output -| myProc
> myOutput
```

Using Differential Nodes in Circuit Editor

CAUTION. *The differential nodes capability is not recommended for general use. The existing differential nodes capability has known bugs which will be fixed at a later date.*

ADS has an optional capability of having pairs of signals (such as differential pairs of nets) represented as single wires. This option is not a full feature set and as a result there are some idiosyncrasies which will be covered in this document. It should be kept in mind that this capability is primarily driven by the need to derive correct netlists while having the ability to use the shorthand representation of signal pairs.

To use the differential capability in ADS, the differential netlist package must be enabled. To enable the differential netlist open the **System Configuration** dialog box from the **Launche**r window by executing the **Configure > system** command. In the packages section (near bottom of window) there should be a selection box entitled **differential netlist**. If so, select this package and exit window by selecting ”ok”. If not, quit this window and copy the following line into the **System Transcript** dialog box,

```
TspiceUserStyle installDifferentialPackage
```

highlight this line and use the pop–up button command **do it**. Reopen the **System Configuration** dialog box and enable **differential netlist** as previously described.

Differential wires in ADS schematics do not have any physical attribute to differentiate them from single net wires. The fact that a wire represents a single net or a pair of nets is totally determined by the characteristics of the ports of subcircuits to which the wire is connected. Users can facilitate visual differentiation of single net wires from net pair wires by using naming conventions for net naming (such as naming single nets ”netH”, ”netL”, or ”Vnet” and naming net pairs as ”net”). The primary impact of differential wiring is in the **Circuit Editor** and **Subcircuit Editor**. Since the behavior in the **Circuit Editor** is a subset of that in the **Subcircuit Editor**, only the operation in the **Subcircuit Editor** window will be described.

In the **Subcircuit Editor** (or the **Symbol Editor**), the *Port Editor Pane* (top middle pane of **Subcircuit Editor** and lower right pane of **Symbol Editor**) has additional capability to handle differential ports. If no ports are highlighted, the selection **add decl terminal** results in a differential port being added to the port list. **Add secl terminal** results in a single–ended port being added. Either port type is pasted on the symbol exactly as a standard port. Again labeling of ports on the symbol can be used as a visual indicator of the port type (differential vs. single–ended.). Note that the *Port List Pane* differentiates differential ports by appending the string ”(#DECL)” to the port name. Once a new port has been

added (but before it is pasted on the symbol) and highlighted in the port list pane, the user can ”convert to DECL” or ”convert to SECL” using the port list menu choices.

In the **Subcircuit Editor**, the node list pane (right middle pane) is modified to handle differential nodes. All differential ports appear in the node list pane just as single net ports. Local differential node names can be added to the node list. Entries for adding a differential local node can be a single node name (the default assignment for a differential node called ”net” is to produce two nets in a netlist called ”netH” and ”netL”). One idiosyncrasy is if a ”local” node is added to the node list pane and is pasted onto a wire that is a differential (as defined by the port of a subckt to which it is connected), then the ”local” node acts like a differential node. It produces the default net names (”netH” and ”netL”) when a netlist is generated. If the component node names (”netH” and ”netL”) for either a differential port or local differential node are needed in the schematic pane, then they must be added to the node list pane using the selection ”add local node”.

In the **Subcircuit Editor** *Schematic Pane*, differential connections between subcircuits is done with wires identically to normal single net connections. Likewise, labels can be pasted on wires in the normal manner. The actual nets generated when a differential net pair is netlisted consists of the names ”netH” followed by ”netL”. One question that often arises is how one makes connections between a differential wire and the two component wires. The answer is by node naming. For example, if a differential wire named ”net” needs to be connected to the bases of two transistors, the wires connected to these bases should be labeled ”netH” and ”netL” (assuming the default differential to component net mapping). Thus the netlister will make the connection via the net naming and there is no actual physical wire connection in the schematic between the differential wire and either component net wire.

Before closing a **Subcircuit Editor** that uses differential ports, it is recommended that all differential port labels be pasted into the circuit. If this is not done, you will get a warning when accepting the subcircuit that all un–placed ports have not been pasted into the subcircuit. Many times a subcircuit will not contain a differential ported subcircuit that would connect a differential port. Instead the component nets (”netH” and ”netL”) will be pasted into the subcircuit. Upon acceptance, the ADS checker logic does not recognize that these two component port names are equivalent to the differential port ”net”, hence the previously mentioned warning. Note that for nets (other than ports) which must transition between a differential wire and two component single–ended wires, an ”accept” of the circuit (or subcircuit) will result in a Dialog Box which will say ”Warning: some net(s) have less that two connections.” The lower righthand pane will contain a list of all such nodes. Again the ADS checker does not recognize the differential node as being composed of the two component nets, hence the message. The actual netlist will show the proper connections.

Differential connections on schematics are done with wires as are single– ended wires. A special model exists that allows a differential wire net to be inverted by reversing the connections of the component nets. This special model ”bubble” has two differential ports and is installed just like any other model with one major requirement : one port must be connected to a wire and the other port must be directly connected to the differential port of another model. Note that the ”bubble” model does not generate an instance or model description when the netlist is generated. Any model description associated with ”bubble” will be ignored. The ”bubble” can be thought of as a plumbing structure whose sole function is to reverse the net order. Thus if component nets ”aH” and ”aL” are on the wire side of a ”bubble”, then ”aL” and ”aH” respectively will be connected to the differential port of the other model connected to the other ”bubble” port. Note that the ”bubble” structure is completely symmetric and either port can be an input.

Wiremenu’s for differential wires differ from those of single–ended wires. The wiremenu (available after an analysis has been run) for a differential wire contains 6 choices : bv(H), bv(L), v(H), v(L), bv(H)–bv(L), or v(H)–v(L). These correspond to the Bias Voltages of the two component nets (referred to as H and L in the menu), the transient voltages of the component nets, the differences between the bias voltages of the two nets, and the differences between the transient voltages of the two nets.

One confusing thing can happen when the single–ended port device is connected to a differential port. ADS will not issue an warning to indicate this has occurred. The netlist generated will have the single– ended port connected to the net name (without any H or L suffix). The differential device port will show two component nets (typically netH and netL) connected. The resulting simulation results may be confusing since the single–ended port will NOT actually be connected to the differential port.

Back Annotation of Schematics in ADS

The back annotation feature is used for changing element parameters of an ADS circuit as specified by a ”back annotation” file. (See example2 below). The topology, net names, element names or model names of a circuit cannot be changed using the **back annotate** command.

To use the back annotation capability in ADS, the back annotation package must be enabled. To check if the back annotation package is enabled, open the **System Configuration** dialog box from the **Launche**r window by executing the **Configure > system** command. In the packages section (near bottom of window) there should be a selection box entitled **back annotation**. If it is listed then it is enabled. Select this package and exit window by selecting ”ok”.

If it is not listed it must be enabled. It is enabled by quiting this window and copying the following line into the **System Transcript** dialog box (**Open > System Transcript** from the **Launcher**),

```
TspiceUserStyle installBackAnnotationPackage
```

highlight this line and select the pop–up button command **do it**. Reopen the **System Configuration** dialog box and enable **back annotation** as previously described.

The **back annotate** menu can be accessed from 1) The **Circuit Browser** *Circuit List Pane* with a circuit selected and 2) any circuit editor *Edit/Analysis/Annotation Pane.*

The **back annotate** command prompts the user for a back annotation file, <*file.ann*>. It uses the element statements in the back annotate file to modify element parameter values. When an element parameter is changed, the old parameter value is appended to the statement as a comment using a semicolon, ”;”. See example 1 below.

An undo file and an error file are created during the back annotation process. The undo file, <*file*>*_undo.ann*, is in the back annotation format and can be used to get back to a previous state. The error file, <*file*>*_error.ann* lists lines of the <*file.ann*> file that were not processed due to errors.

CAUTION. *The ADS* ***undo*** *command does not work for the* ***back annotate*** *command. The user should write an EDIF of the circuit before using the* ***back annotate*** *command. (This will make a copy of the libraries.)*

Procedures and processes for creating the back annotation file external to ADS are under development.

Example 1

This example of a parameter line (as seen in an **Element Browser**) shows the original parameter line and then the new parameter line which includes the original parameter value separated by a semicolon.

```
r=19.8k                    (original value)

r=1909.6 ; 19.8k           (new value)
```

where r is a parameter in the resistor model.

Example 2

This is an example of a back annotation file that was extracted at the top level of the circuit. The top level contains two resistors R1 and R2 and a subcircuit, circuit1 which contains two resistors R11 and R22):

```
R1 1 2 r: r=201;
R2 3 4 r: r=301;
circuit1.R11 in 4 r: r=2000;
circuit1.R22 4 out r: r=3000;
```

NOTE*. Semicolons are required at the end of lines.*

For this example, to use this file in ADS, the user must be at the top level of the circuit when executing the **back annotate** command. (The same level of hierarchy at which the file was created.)

Libraries

ADS Models

Models are available to the circuit through libraries. Library models are global to the ADS image.

The following four items are required for an ADS model:

1. A graphic schematic symbol with a list of terminals.
2. A list of input parameters with default values.
3. A list of output parameters.
4. A method of describing the model to TekSpice. Internal libraries use a schematic, extermal libraries use a TekSpice netlist.

The **Library Browser** lists all libraries in the ADS image. Libraries are linked to a circuit using the **set libraries** command found in the **Circuit Browser**. Models available to be pasted into the circuit must come from one of these libraries.

The ADS Library Structure

Figure 4–15 shows the ADS library structure. This structure shows the hierarchy in the **Library Browser**. These libraries are organized into three levels by Process, Library and Models. Libraries may only be added to the selected (highlighted) Process. Models may only be added to the selected Library.

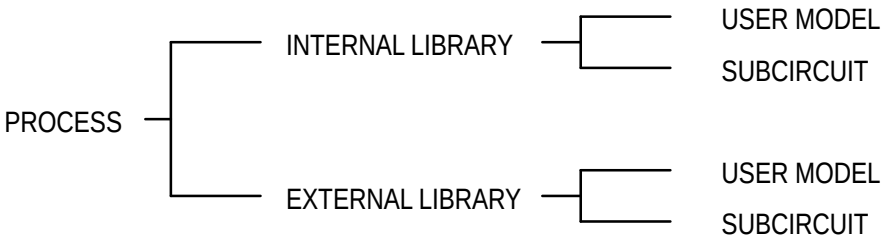


Figure 4–15: ADS Library Structure

Figure 4–16 shows the **Library Browser** and the the location of the panes within the **Library Browser**; the *Process List Pane*; the *Library List Pane*; the *Library Model List Pane* and the *Comment Text Pane.*

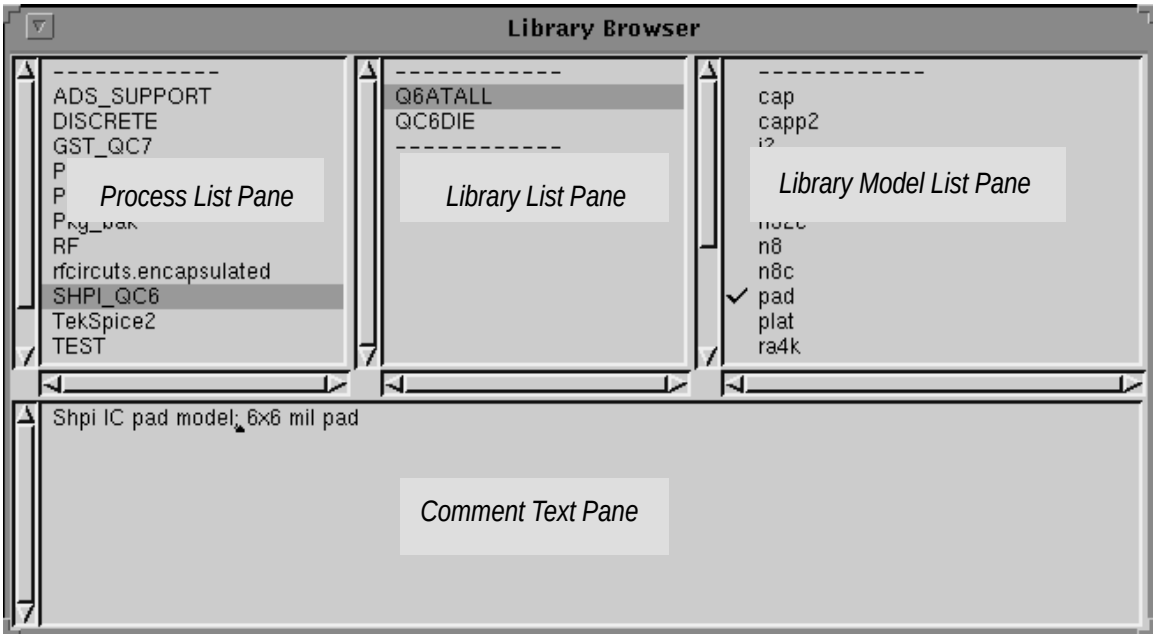


Figure 4–16: Library Browser Panes

There are two types of libraries used by ADS; Internal Libraries and External Libraries.

Internal ADS Libraries

An Internal Library is a library generated within the ADS system. Internal models contain symbols that represents the model and a circuit schematic that represents the equivalent circuit of the symbol. When the model is referenced, in another circuit, that circuit will require access to those libraries referenced by that model. These libraries would be made available to the circuit using the **set libraries** command in the **Circuit Browser**.

External ADS Libraries

An External Library is a library that exists as a file in Tekspice netlist format. This file must be generated using the ”**genlib**” command. (See the end of this section for the ”genlib” command). The External Library name is set using the **Library Browser**’s *Library Model List Pane* **add external library** or **change library** commands. The netlist text for this library can be viewed through the pop–up button **show model text** command.

External Libraries have a path, which specifies the location of the library file. This path can be changed to specify alternative versions of the device library in

the same directory. This is often done, for example, to switch between ”nominal”, ”up” and ”down” libraries.

Example of a Model Found in an External Library

The **Library Subcircuit Symbol Editor** in Figure 4–17 shows the *rax2k* model defined in the external library *Q6RTVER*. See the *Window Description* chapter for the windows used for editing these symbols and model definitions. There is no schematic equivalent for the *rax2k* model since it is described in the External Library as a netlist.

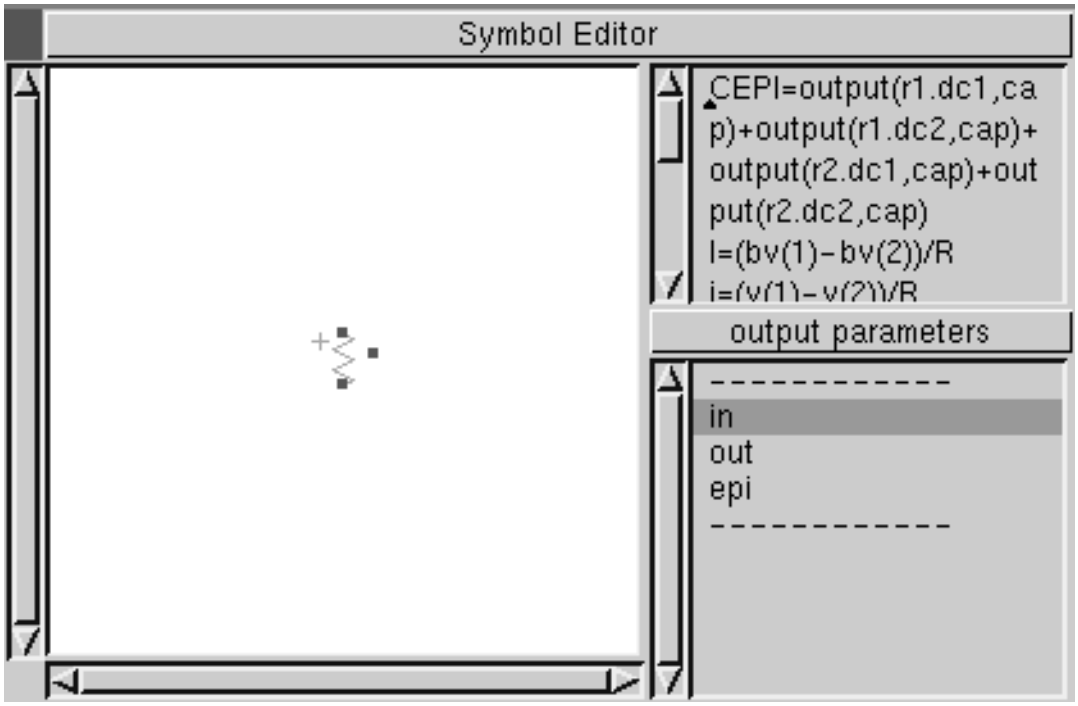


Figure 4–17: ’rax2k’ Model from the Q6RTVER External Library

Adding an External Library to ADS

Add an External Library to your ADS image as follows:

1. Convert the library to TekSpice subcircuit format (if necessary). This is done with the **genlib** command.
2. From the *Library List Pane* select the pop–up button command **add external library**. See Figure 4–18. This opens the **Add External Library** dialog box. See Figure 4–19

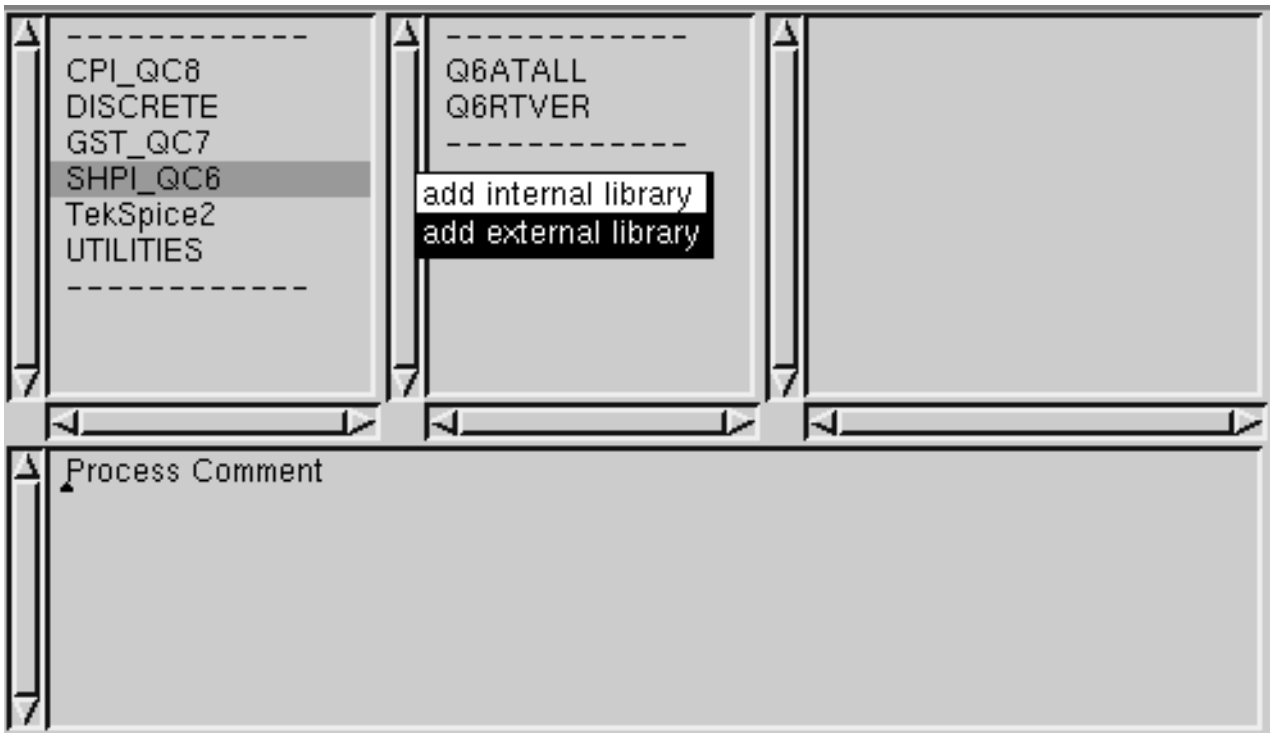


Figure 4–18: Adding an External Library to the ADS Image

3. In the **Add External Library** dialog box enter the the library name and file name pattern for the genlib file (usually *.bin) in the apropriate text boxes. If the genlib file does not exist add the *.bin anyhow.

4. If the genlib file does not exist press the ok button. If it does exist go to step 6. The genlib file can also be generated manually using the /u/cae/bin/genlib command with the text file as input.

5. A dialog window appears asking for the location of the TekSpice text file. Enter the path and file name in the text box. The path should already be correct from step 3. Hit continue.

6. Highlight the file name pattern and select the pop–up button command **accept**. This causes all files matching the pattern to appear in the files list.

7. Select the apropriate file and press the **ok** button.

8. The library name should appear in the *Library List Pane*. From the Library *Model List Pane* use the pop–up command **add subcircuit** and create the model names from the library. Then use the **edit symbol** command to generate the model symbols for each of the models on the list. If the symbols have already been generated but are in edif format then use the **read edif** command to read them into the *Library Model List Pane*.

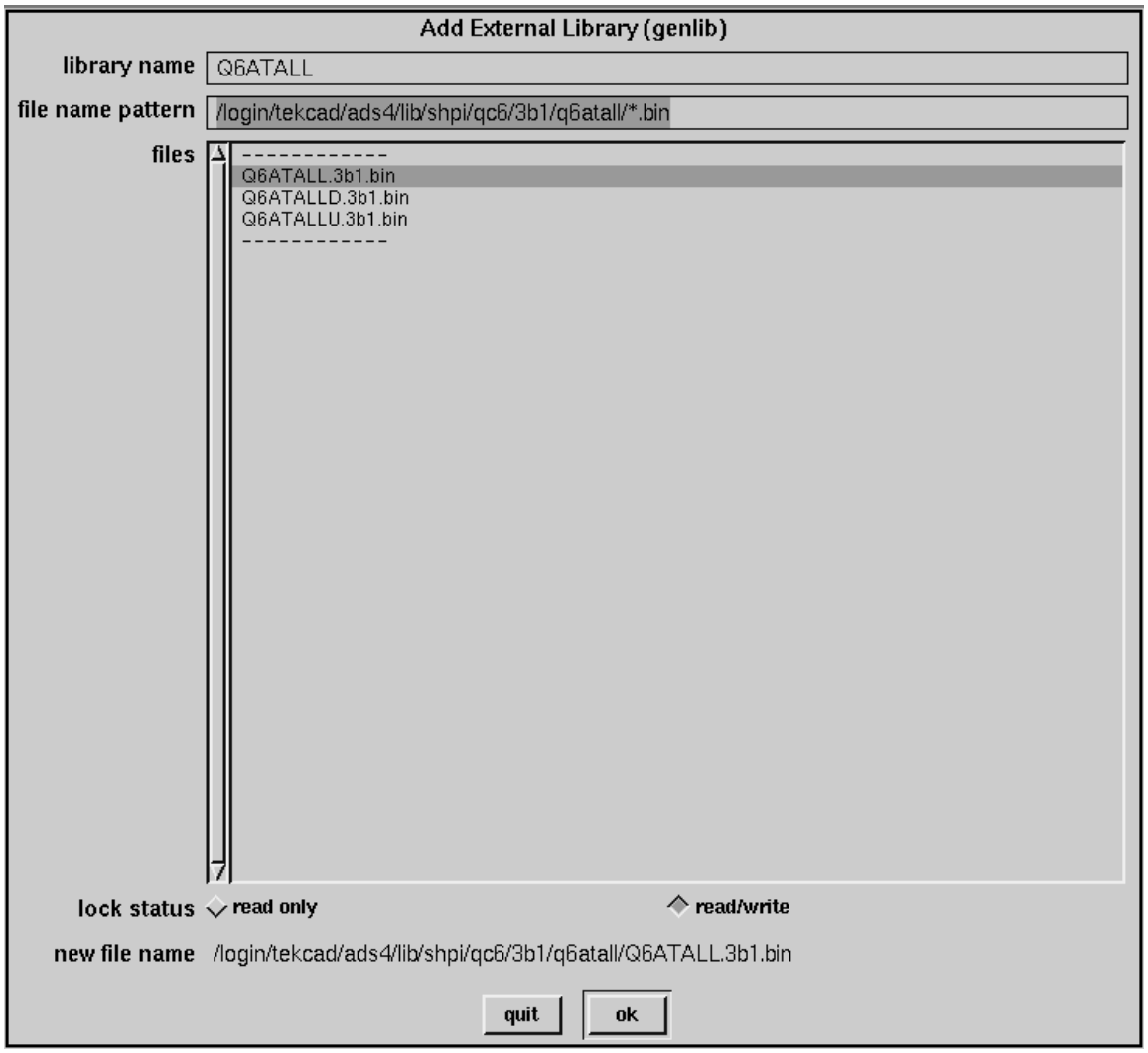


Figure 4–19: Add External Library Dialog Box

TekSpice Primitive Device Models

In ADS, a primitive model refers to a model which maps to a model in TekSpice or to an optional shared object library. The ADS primitive models do not always correspond directly to the primitive TekSpice models, for example, two models in ADS map to the gp2 model in TekSpice: gp2_npn, and gp2_pnp. The two models are necessary in ADS, since they each require a different symbol, and in some cases a different number of terminals.

These standard models are available through four libraries. In most cases these libraries are user models that reference the primitive models. They are in the UTILITIES process of the **Library Browser**. The library names and model

(This page intentionally left blank)contents are as follows: (optional devices and libraries are indicated with asterisks)

Table 4–10: UTILITIES>Devices_Passive

Model	Description
c	capacitor
cpl	coupled lossless transmission line (symmetric)
cplc	coupled lossless transmission line (non–symmetric)
l	inductor
reporter	reporter element
r	resistor
t	lossless split reference transmission line
t1	lossless floating reference transmission line
tf2	transformer; 2–terminal
tf3	transformer; 3–terminal
tf4	transformer; 4–terminal
tf5	transformer; 5–terminal
tf6	transformer; 6–terminal
tf7	transformer; 7–terminal
tf8	transformer; 8–terminal
tf9	transformer; 9–terminal
tlse*	lossy transmission line

Table 4–11: UTILITIES>Devices_Semiconductors

Model	Description
d	diode
gp2_npn	npn Gummel Poon bipolar transistor: level 2
gp2_pnp	pnp Gummel Poon bipolar transistor: level 2
jfet_n	junction field effect transistor: n–type
jfet_p	junction field effect transistor: p–type
mfet2_n_dep	metal oxide field effect transistor: n–type depletion mode
mfet2_n_enh	metal oxide field effect transistor: n–type enhancement mode

Table 4–11: UTILITIES>Devices_Semiconductors (Cont.)

Model	Description
mfet2_p_dep	metal oxide field effect transistor: p–type depletion mode
mfet2_p_enh	metal oxide field effect transistor: p–type enhancement mode

Table 4–12: UTILITIES>Devices_Sources

Model	Description
cccs1	current controlled current source: one control
cccs2	current controlled current source: two controls
cccs3	current controlled current source: three controls
cccs4	current controlled current source: four controls
ccvs1	current controlled voltage source: one control
ccvs2	current controlled voltage source: two controls
ccvs3	current controlled voltage source: three controls
ccvs4	current controlled voltage source: four controls
i	current source
v	voltage source
vccs1	voltage controlled current source: one control
vccs2	voltage controlled current source: two controls
vccs3	voltage controlled current source: three controls
vccs4	voltage controlled current source: four controls
vcvs1	voltage controlled voltage source: one control
vcvs2	voltage controlled voltage source: two controls
vcvs3	voltage controlled voltage source: three controls
vcvs4	voltage controlled voltage source: four controls

Table 4–13: UTILITIES>Devices_Functional*

Model	Description
abs	absolute value
accumulator	Clocked accumulator
acos	arccosine value
acosh	arccosh value
adc	Analog–to–digital converter
add1b	one bit adder
and	logical and gate
asin	arcsine value
asinh	arcsinh value
atan	arctan value
atanh	arctanh value
biquad	bi–quadratic filter(2–pole, 2–zero)
bmax	Maximum of inputs
bmean	Average of inputs
bmin	Minimum of inputs
bpulse	binary pulse sequence generator
ccomp	clocked comparator
ckdelay	clock–controlled delay
clock	digital clock
combflt	digital comb filter
comp	comparator
cos	cosine value
cosh	cosh value
counterf	frequency counter
ctb	composite triple beat noise generator
dac	digital–to–analog converter
ddt	derivative with respect to time
delay	time delay
dflipflop	d–type flip flop with reset
div	divider
exp	exponential value
fadc	analog–to–integer converter
fdac	integer–to–analog converter
firflt	finite impulse response filter
gain	gain device

Table 4–13: UTILITIES>Devices_Functional* (Cont.)

Model	Description
gpwl	piecewise linear gain device
gspline3	cubic spline gain device
gtanh	hyperbolic tangent gain device
inoise	noise current generator
integ	integrator
inv	inverter
latch	digital latch
ln	mathematical function: natural logarithm value
log	mathematical function: logarithm base 10 value
lpf	low pass filter
mod	modulo value
monostable	monostable multivibrator device
mult	multiplier
mux221	two–to–one multiplexer
nand	logic nand gate
nor	logic nor gate
or	logic or gate
pdnstate	n–state abstract phase detector
pdsine	sine abstract phase detector
pdtriangle	triangle abstract phase detector
prbs	pseudo random bit sequence generator
prbsp	pseudo random bit sequence phase generator
pulse	PWL pulse generator
rms	root mean square of the input signal
sampleh	sample and hold device
schmitt	schmitt trigger
sin	sine value
sinh	sinh value
spreadctrl	high level spread spectrum controller model
sqrt	square root value
sum	summer
switchn	n–type mos switch
switchp	p–type mos switch
tan	tangent value
tanh	hyperbolic tangent value

Table 4–13: UTILITIES>Devices_Functional* (Cont.)

Model	Description
tmax	maximum value of trace
tmean	mean of the input signal
tmin	minimum value of trace
trackh	track and hold
vco	voltage controlled oscillator
vnoise	noise voltage generator
waveshape	waveshaping of rise and fall time
xnor	exclusive nor gate
xor	exclusive or gate

*** optional– available through shared object libraries.**

Editing a Library Model

Library models are edited in one of several ways, depending on the type of change required.

The model symbols are edited from the *Symbol Editor Pane* of the **Library Subcircuit Definition Editor,** or the **Library Subcircuit Symbol Editor**. Here symbol graphics, port ordering and terminal count are changed. If port ordering or terminal count changes then existing model instances must be cut and pasted back into the schematic with the new symbol.

The model schematics are edited from the **Library Subcircuit Definition Editor** or the **Library Subcircuit Instance Editor.** Use the **Library Subcircuit Definition Editor** for initially defining a model. Use the **Library Subcircuit Instance Editor** (accessed from the **Circuit Editor**) during the simulation process.

It is important to remember that library models are global to the ADS image. This means any model changes affect all circuits.

An example of this is copying a circuit to a different circuit name, to experiment with design variations. The copied circuit still references the same libraries as the original circuit. Library model changes in the copied circuit change the models globally. If you do not want the models to change, change the library model name, library name or path.

GENLIB – Convert Model Definitions into Library File

Syntax

genlib [<textfile1> [<textfile2> ...]] [–o <outputfile>]

Description

genlib converts one or more files of model and subcircuit definitions into a device library file. If no input file names are specified, **genlib** reads standard input. If no output file is specified (the ”–o” option is not used), **genlib** writes its output to a file named ”genlib.bin”. If a file already exists with the selected output name, it is first moved to a file named <outputfile>~.

Device library files created by **genlib** are read into an ADS image using the ”add external library” command in the **Library Browser**, or are used on a ”libpath” statement in a TekSpice input file.

Options

–o outputfile

Name the new device library output file. If a file already exists with that name, first move it to a file named <outputfile>~.

Examples

In this example:

```
genlib mod1.text mod2.text –o models.bin
```

the model definitions in the files ”mod1.text” and ”mod2.text” are concatenated into a device library named ”models.bin”.

To read the source from standard input, type either:

```
genlib
or
genlib –o myModels.bin
```

and then begin typing the model definition[s]. Type `control–d` when finished.

The following command sequences also read from standard input:

```
genlib –o myModels.bin < myModels.text
cat models1.text models2.text models3.text | genlib
```

See Also

UNGENLIB

WARNING.

*The file created by **genlib** is a combination of a text and binary file. Do not attempt to edit this file in any way. Use **ungenlib** to look at the file. If the file is mistakenly edited, use **genlib** to regenerate it from the original source files.*

UNGENLIB – Convert Library File(s) into Text

Syntax

ungenlib [<deviceLibrary1> [<deviceLibrary2> ...]]

Description

ungenlib converts one or more device library files into text, which is written to standard output.

Examples

In this example:

```
ungenlib mod1.bin mod2.bin > models.text
```

the model definitions in the device library files ”mod1.bin” and ”mod2.bin” are concatenated into a text file named ”models.text”. The text file is modified with a text editor, then **genlib** is used to create a new device library from the results.

See Also

GENLIB

TS2TEXT – Convert TekSpice2 ansr file format to ASCII text

Syntax

ts2text [–q] <*ansrp*> <*plotfile*>

Description

Ts2text converts a TekSpice2 ansrp file (binary) and a TekSpice2 plotfile into an ASCII text file. The output is to stdout.

–q a flag indicating the output should be in QuERC input file format. If –q is omitted, the output format is similar to the TekSpice1 file format.

<*ansrp*> is the name of the TekSpice2 ansrp file to be converted. E.g. acansrp15, dcansrp93, transrp5, biansrp99.

<*plotfile*> is the name of the TekSpice2 plot file to be converted. E.g. acfile15, dcfile14, trfile5, bifile99

Examples

ts2text –q bifile99

Application Models

RF Microstrip Models

The RF microstrip model library is an optional library available for use with ADS. These models are designed for use with ac and transient analyses. These libraries are located in the **Library Browser**: RF > Microstrip. A list of the models available are shown in Table 5–1.

Table 5–1: RF Microstrip Models

Model Name	Description
M_BEND90	Right angle bend
M_BEND90M	Right angle mitered bend
M_BEND90OPT	Right angle optimal mitered bend
M_GAP	Symmetrical Gap in line
M_NOTCH	Notch in line
M_OPEN	Open circuit line
M_SHORT	Short circuit line
M_STEP	Abrupt symmetric step in line width
M_TRL	Lossless transmission line

Microstrip Dimensions

The dimensions for a buried microstrip line are shown in Figure 5–1.

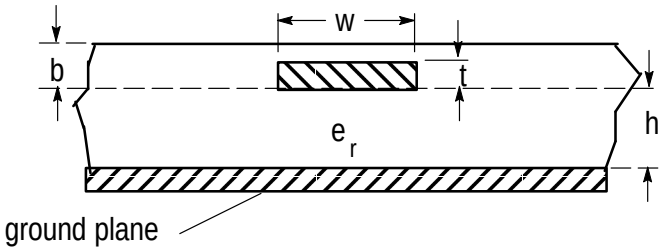


Figure 5–1: Buried Microstrip Line showing dimensions

Relative Dielectric Constant, e_{reff}

The relative dielectric constant e_r, of the substrate below the microstrip is changed due to an air or other dielectric layer above the microstrip. This change is a function of line thickness t, line width w, and height of the line above the ground plane h. The thickness of a the dielectric layer above the microstrip is given by parameter b. The relative dielectric constant of this layer is assumed to be the same as that of the substrate. See Figure 5–1. The equations used to compute the relative dielectric constant (not including the effect of soldermask thickness, b) is given as:

$$e_{re} = \begin{cases} \frac{(e_r + 1)}{2} + \frac{(e_r - 1)}{2}\left[\left(1 + 12\frac{h}{w}\right)^{-0.5} + 0.04\left(1 - \frac{w}{h}\right)^2\right] - \frac{(e_r - 1)}{4.6}\frac{t}{\sqrt{wh}} & \frac{w}{h} < 1 \\ \frac{(e_r + 1)}{2} + \frac{(e_r - 1)}{2}\left(1 + 12\frac{h}{w}\right)^{-0.5} - \frac{(e_r - 1)}{4.6}\frac{t}{\sqrt{wh}} & \frac{w}{h} > 1 \end{cases}$$

Including the effect of soldermask thickness, b, we have:

$$e_{reff} = e_{re}e^{-2b/h} + e_r\left(1 - e^{-2b/h}\right) \qquad \frac{b}{h} < 1$$

Effective Line Width, w_{eff}

For lines with thickness t, the line width w, is modified as given in the equations below:

$$w_{eff} = \begin{cases} w + \frac{1.25}{\pi} t\left(1 + \log_e\left(\frac{4\pi w}{t}\right)\right) & \frac{w}{h} < \frac{1}{2\pi} \\ w + \frac{1.25}{\pi} t\left(1 + \log_e\left(\frac{2h}{t}\right)\right) & \frac{w}{h} > \frac{1}{2\pi} \end{cases}$$

Characteristic Impedance, Z_0

The characteristic impedance and time delay of a microstrip line is determined by the following equations:

$$Z_0 = \begin{cases} \frac{60}{\sqrt{e_{trff}}} \log_e\left(8\frac{h}{w_{eff}} + 0.25\frac{w_{eff}}{h}\right) & w/h < 1 \\ \frac{120\pi}{\sqrt{e_{reff}}\left(\frac{1}{w_{eff}/h + 1.393 + .667\log_e(w_{eff}/h + 1.444)}\right)} & w/h > 1 \end{cases}$$

$$t_d = \frac{pl\sqrt{e_{reff}}}{300x10^6}$$

Dispersion

The frequency dependence of the dielectric constant is not included in these models. An output parameter, fLimit, has been included in the appropriate models to give an indication of the lower frequency at which the dielectric frequency dependence becomes important [3].

$$fLimit[gHz] = 0.03024\sqrt{\frac{Z_0}{h\sqrt{e_r - 1}}}$$

Ground

The microstrip models in this section are not grounded internally. Each model has a terminal located on the perimeter of the symbol which should be connected to ground.

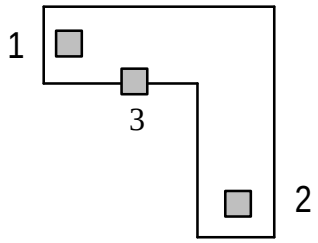
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits,* Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_BEND90 – Right Angle Bend

Symbol



Syntax

<name> <n1> <n2> <n3> m_bend90 [: <parameter> = <value> ...]

Examples

bend1 1 2 0 m_bend90: w=3.1m, h=.9m, er=2.5

Description

The right angle bend model is represented by the LC tee circuit shown in Figure 5–2.

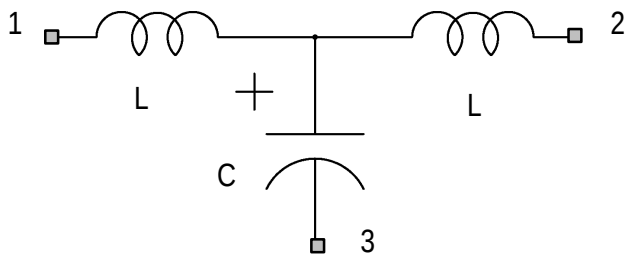


Figure 5–2: Equivalent Circuit for Right Angle Bend

The m_bend90 model include only the square portion of the line at the corner of the bend. The equations used to compute L and C are given below:

$$\frac{C}{w}[pF/m] = \begin{cases} \dfrac{(14e_r + 12.5)w_{eff}/h - (1,83e_r - 2.25)}{\sqrt{w_{eff}/h}} + \dfrac{0.2e_r}{w_{eff}/h} & w/h < 1 \\ (9.5e_r + 1/25)w_{eff}/h + 5.2e_r + 7.0 & w/h > 1 \end{cases}$$

$$L[nH] = \begin{cases} \cdot 50\left(4\sqrt{w_{eff}/h} - 4.21\right)h & w/h > 1 \\ \cdot 220h\left[1 - 1.35e^{\left[-0.18(w/h)^{1.39}\right]}\right] & w/h < 1 \end{cases}$$

$$Z_0 = \sqrt{L/C}$$

$$t_d = L/Z_0$$

The parameter w_{eff} is defined on page 5–2 in this section. See also Figure 5–1.

Parameters

Table 5–2: M_BEND90 Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–3: M_BEND90 Model Output Parameters

Parameter	Description	Units
C	Capacitance of equivalent circuit	Farads
ere	Relative effective dielectric constant	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
L	Inductance of equivalent circuit	Henries
td	time delay of line	seconds
z0	characteristic impedance of line	Ohms

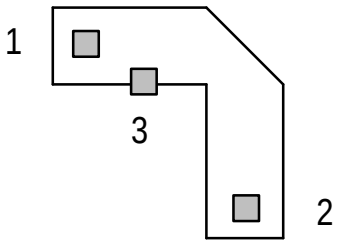
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_BEND90M – Right Angle Mitered Bend

Symbol



Syntax

<*name*> <*n1*> <*n2*> <*n3*> m_bend90m [: <*parameter*> = <*value*> ...]

Examples

bend1 1 2 0 m_bend90m: w=3.1m, h=.9m, er=2.5 coff=50

Description

The right angle mitered bend model is represented by the LC tee circuit shown in Figure 5–3. The miter or chamfer for this model is fixed at 45°with a length of w times the square root of 2

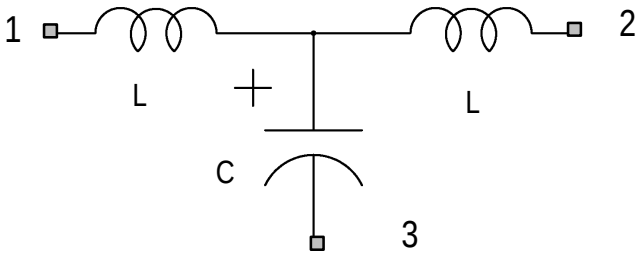


Figure 5–3: Equivalent Circuit of Right Angle Mitered Bend

The equations used to compute L and C are given below:

$$L[nH] = 0.44h\left(1-1.062e^{\left[-0.177(w/h)\right]^{0.9477}}\right)$$

$$C[pF] = h\left[(3.93e_r + 0.62)(w/h)^2 + (7.6e_r + 3.8)(w/h)\right]$$

Parameters

Table 5–4: M_BEND90M Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–5: M_BEND90M Model Output Parameters

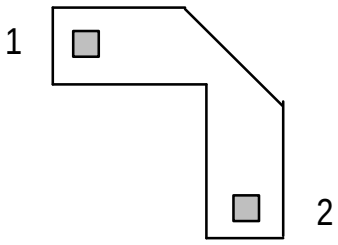
Parameter	Description	Units
C	Capacitance of equivalent circuit	Farads
ereff	Relative effective dielectric constant	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
L	Inductance of equivalent circuit	Henries
td	Time delay of line	seconds
z0	Characteristic impedance of line	Ohms

References

Microwave Transmission Lines and Their Physical Realizations, Steven L. March, Besser Associates, Inc.

M_BEND90OPT – Right Angle Optimal Miter Bend

Symbol



Syntax

<*name*> <*n1*> <*n2*> m_bend90m [: <*parameter*> = <*value*> ...]

Examples

bend1 1 2 m_bend90m: w=3.1m, h=.9m, er=2.5 coff=50

Description

The right angle optimal miter bend model is represented by the series inductor shown in Figure 5–3.

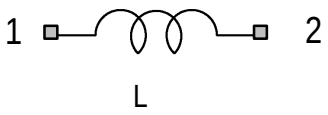


Figure 5–4: Equivalent Circuit of Right Angle Optimal Miter Bend

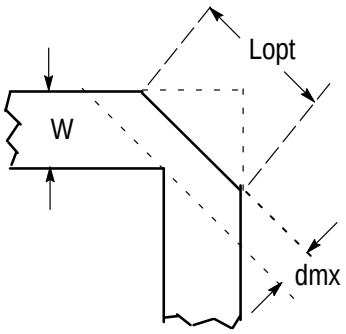


Figure 5–5: Physical Dimensions of Optimal Bend

The inductance L, and the length of the miter, L_{opt} for this model are given by the following equations:

$$\cdot L[nH] = \begin{cases} h\left[0.5\left(\frac{w_{eff}}{h}\right)^{1.08} - \frac{0.45}{\sqrt{e_{reff}}} - 0.12\right] & L_{opt} \leqq w\sqrt{2} \\ \cdot h\left[0.5\left(\frac{w_{eff}}{h}\right)^{1.08} - \frac{0.45}{\sqrt{e_{reff}}} - 0.12\right] - 2w + L_{opt}\sqrt{2} & L_{opt} > w\sqrt{2} \end{cases}$$

$$L_{opt} = w\sqrt{2}\left[1.04 + 1.3e^{(-1.35w/h)}\right] \qquad \begin{array}{l} w/h \geqq 0.25 \\ e_r \leqq 25 \end{array}$$

$$dmx = w\sqrt{2} - L_{opt}/2$$

Parameters

Table 5–6: M_BEND90OPT Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–7: M_BEND90OPT Model Output Parameters

Parameter	Description	Units
dmx	Distance of miter from inside corner	meters
ereff	Relative effective dielectric constant	–
L	Inductance of equivalent circuit	Henries
Lopt	Length of miter	meters

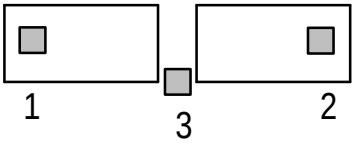
References

Steven L. March, ”Microwave Transmission Lines and Their Physical Realizations”, Besser Associates, Inc.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_GAP – Symmetrical Gap in Line

Symbol



Syntax

<*name*> <*n1*> <*n2*> <*n3*> m_gap [: <*parameter*> = <*value*> ...]

Examples

gap1 1 2 0 m_gap: w=3.1m, h=.9m, er=2.5, gap=5m

Description

The gap model is represented by a modified LC tee.

Parameters

Table 5–8: M_GAP Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
gap	Gap width	meters	.001
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–9: M_GAP Model Output Parameters

Parameter	Description	Units
C	Capacitance of gap	Farads
ereff	Relative effective dielectric constant	–
z0	Characteristic impedance of line	Ohms

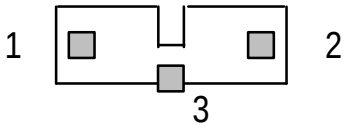
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_NOTCH – Notch in Line

Symbol



Syntax <*name*> <*n1*> <*n2*> <*n3*> m_notch [: <*parameter*> = <*value*> ...]

Examples step1 1 2 0 m_notch: w=2.31m, wnd=1.31m, h=.9m, er=2.5

Description The microstrip notch is represented by the circuit shown in Figure 5–6.

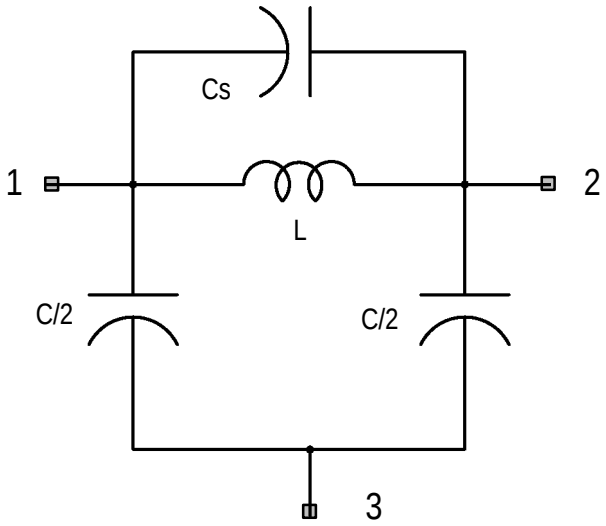


Figure 5–6: Notch Equivalent Circuit

The section of line included in the model is indicated by ports 1 and 2 in Figure 5–7 . The notch width a, must be less than or equal the height of the line above the ground plane.

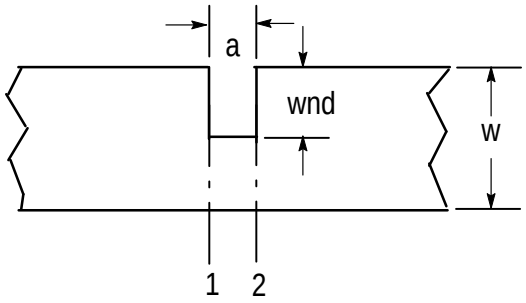


Figure 5–7: Top view of Notch Showing Ports and Parameters

The inductance L is determined by:

$$L[uH] = ah\frac{\pi^2}{5}\left[1-\frac{Z_{0w}}{Z_{0n}}\sqrt{\frac{e_{reffw}}{e_{reffn}}}\right]^2$$

$$C[uF] = \frac{a\sqrt{e_{reffn}}}{2\ Z_{on}\ 3x10^8}$$

The parameters Z_o and e_{reff} are defined on page 5–2 in this section. See also Figure 5–1.

Parameters

Table 5–10: M_NOTCH Model Input Parameters

Parameter	Description	Units	Default
a	Width of notch	meters	.001
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
wnd	Depth of notch	meters	.001
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–11: M_NOTCH Model Output Parameters

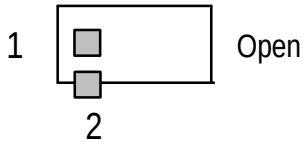
Parameter	Description	Units
ereffn	Relative effective dielectric constant of notch	–
ereffw	Relative effective dielectric constant of line	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
L	Inductance of notch	Henries
z0n	Characteristic impedance of notch	ohms
z0w	Characteristic impedance of line	ohms

References

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991, pp 342–343.

M_OPEN – Open Circuit Line

Symbol



Syntax

<*name*> <*n1*> <*n2*> m_open [: <*parameter*> = <*value*> ...]

Examples

open1 out1 0 m_open: w=3.1m, h=.9m, er=2.5

Description

The open circuit line model uses the UTILITIES>Devices_Passive transmission line model t, to represent the open circuit. The values for Z_0 and t_d are computed from the parameters listed in Table 5–12 using the following equations:

$$Z_o = 60\log_e(8h/w_{eff} + 0.25w_{eff}/h)$$

$$Z_o = \frac{120\pi}{\sqrt{e_{reff}}}\left(\frac{1}{w_{eff}/h + 1.393 + .667\log_e(w_{eff}/h + 1.444)}\right)$$

$$pl = \frac{1}{4}\frac{3x10^6}{f\sqrt{e_{reff}}} - \Delta pl$$

$$\Delta pl = 0.412h\frac{(e_{reff} + 0.3)\ (w_{eff}/h + 0.262)}{(e_{reff} - 0.258)\ (w_{eff}/h + 0.813)}$$

The physical length pl, of the line is reduced due to the capacitive end effects.

The parameters w_{eff} and e_{reff} are defined on page 5–2 in this section. See also Figure 5–1.

Parameters

Table 5–12: M_OPEN Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
f	Frequency at which line is open	Hz	1g
h	Height of line above ground plane	meters	.001

Table 5–12: M_OPEN Model Input Parameters (Cont.)

Parameter	Description	Units	Default
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–13: M_OPEN Model Output Parameters

Parameter	Description	Units
ereff	Relative effective dielectric constant	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
pl	Physical length of line at frequency f	meters
td	Time delay of line	seconds
z0	Characteristic impedance of line	ohms

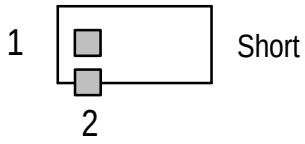
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_SHORT – Short Circuit Line

Symbol



Syntax

<*name*> <*n1*> <*n2*> m_short [: <*parameter*> = <*value*> ...]

Examples

short1 out1 0 m_open: w=3.1m, h=.9m, er=2.5

Description

The short circuit line model uses the UTILITIES>Devices_Passive transmission line model t, to represent the open circuit.. The values shown in Table 5–15 are computed from the parameters listed in Table 5–14 using the following equations:

$$Z_o = 60\log_e(8h/w_{eff} + 0.25w_{eff}/h)$$

$$Z_o = \frac{120\pi}{\sqrt{e_{reff}}}\left(\frac{1}{w_{eff}/h + 1.393 + .667\log_e(w_{eff}/h + 1.444)}\right)$$

$$pl = \frac{1}{2}\frac{3x10^6}{f\sqrt{e_{reff}}} - \Delta pl$$

$$\Delta pl = 0.412h\frac{(e_{reff} + 0.3)\ (w_{eff}/h + 0.262)}{(e_{reff} - 0.258)\ (w_{eff}/h + 0.813)}$$

The physical length pl, of the line is reduced due to the capacitive end effects.

The parameters w_{eff} and e_{reff} are defined on page 5–2 in this section. See also Figure 5–1.

Parameters

Table 5–14: M_SHORT Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
f	Frequency at which line is open	Hz	1g
h	Height of line above ground plane	meters	.001

Table 5–14: M_SHORT Model Input Parameters (Cont.)

Parameter	Description	Units	Default
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.002

Table 5–15: M_SHORT Model Output Parameters

Parameter	Description	Units
ereff	Relative effective dielectric constant	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
pl	Physical length of line at frequency f	meters
td	Time delay of line	seconds
z0	Characteristic impedance of line	ohms

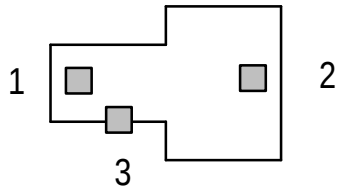
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_STEP – Abrupt Symmetric Step in Line Width

Symbol



Syntax

<*name*> <*n1*> <*n2*> <*n3*> m_step [: <*parameter*> = <*value*> ...]

Examples

step1 1 2 0 m_step: w1=3.1m, w2=10m, h=.9m, er=2.5

Description

The abrupt symmetric step in line model uses the modified LC tee equivalent circuit shown in Figure 5–8. There is no line length included in the model.

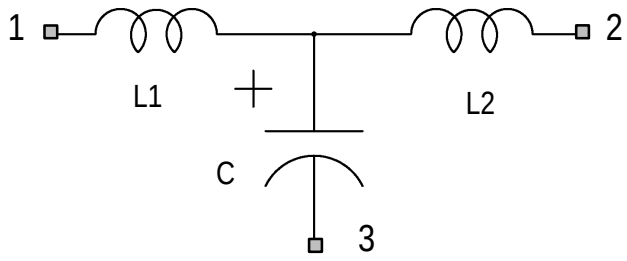


Figure 5–8: Equivlent Circuit of Abrupt Step in Line Width

where:

$$L[nH/m] = 40.5\left[\frac{W_{eff2}}{W_{eff1}} - 1.0\right] - 32.57\log_e\left(\frac{W_{eff2}}{W_{eff1}}\right) + 0.2\left[\frac{W_{eff2}}{W_{eff1}} - 1.0\right]^2$$

$$L_1 = L\left[\frac{L_{m1}}{L_{m1} + L_{m2}}\right] \qquad L_2 = L\left[\frac{L_{m2}}{L_{m1} + L_{m2}}\right]$$

where:

$$L_{m1}[H/m] = \frac{Z_0\sqrt{e_{reff1}}}{3x10^8} \qquad L_{m2}[H/m] = \frac{Z_0\sqrt{e_{reff2}}}{3x10^8}$$

$$C[pF/m] = \begin{cases} \sqrt{w_{eff1} w_{eff2}}\,[4.386 \log_e(e_r) + 2.33]\,\frac{W_{eff2}}{W_{eff1}} - 5.472 \log_e(e_r) - 3.17 & e_r \leqq 10 \\ \sqrt{w_{eff1} w_{eff2}}\,56.46 \log_e\left(\frac{W_{eff2}}{W_{eff1}}\right) - 44 & e_r = 9.6 \end{cases}$$

The parameters w_{eff} and e_{reff} are defined on page 5–2 in this section. See also Figure 5–1.

Parameters

Table 5–16: M_STEP Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w1	Width of narrow side of line	meters	.002
w2	Width of wide side of line	meters	.004

Table 5–17: M_STEP Model Output Parameters

Parameter	Description	Units
C	Capacitance of C	Farads
ereff1	Relative effective dielectric constant of wide side of line	–
ereff2	Relative effective dielectric constant of narrow side of line	–
L1	Inductance of L1	Henry
L2	Inductance of L2	Henry
z01	Characteristic impedance of narrow side of line	ohms
z02	Characteristic impedance of wide side of line	ohms
weff1	Effective width of port1	meters
weff2	Effective width of port2	meters

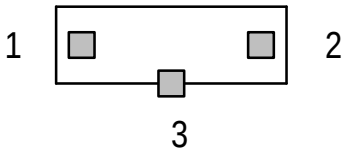
References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

M_TRL – Lossless Transmission Line (Physical)

Symbol



Syntax

<*name*> <*n1*> <*n2*> <*n3*> m_trl [: <*parameter*> = <*value*> ...]

Examples

trl1 11 12 0 m_trl: w=3.1, h=.9, er=2.5

Description

The physical transmission line model is based on the lossless transmission line model, t. The physical representation of the transmission line is shown in Figure 5–1. The input parameters for this model are listed in Table 5–18.

The equations for Z_o and t_d are given on page 5–3.

Parameters

Table 5–18: M_TRL Model Input Parameters

Parameter	Description	Units	Default
b	Bury depth of line by solder mask	meters	0
er	Relative dielectric constant	–	1
pl	Physical length of line	meters	.01
h	Height of line above ground plane	meters	.001
t	Thickness of line	meters	1n
w	Width of microstrip line	meters	.001

Table 5–19: M_TRL Model Output Parameters

Parameter	Description	Units
Cpl	Capacitance per unit length	Farads
ereff	Relative effective dielectric constant	–
fLimit	Frequency below which dispersion effects can be neglected	Hertz
Lpl	Inductance per unit length	Henry
td	Time delay of line	seconds

Table 5–19: M_TRL Model Output Parameters (Cont.)

Parameter	Description	Units
weff	Effective width of line	meters
z0	Characteristic impedance of line	Ohms

References

K.C. Gupta, R. C. Garg, I. J. Bahl, *Computer Aided Design of Microstrip Circuits*, Artech House, Inc., Dedham, Massachusetts, 1981.

Brian C. Wadell, *Transmission Line Design Handbook*, Artech House, Boston, 1991.

Microstrip Example Circuits

Microstrip Chebychev Filter Circuit

A four pole Chebyshev low pass filter circuit is given in this example. This circuit has 0.1dB ripple at a cutoff frequency of 1GHz and is designed for a 10 mil alumina substrate (h= 254um, er=9.6). Line thickness is ignored (t=0). The schematic diagram of this circuit is shown in Figure NO TAG.

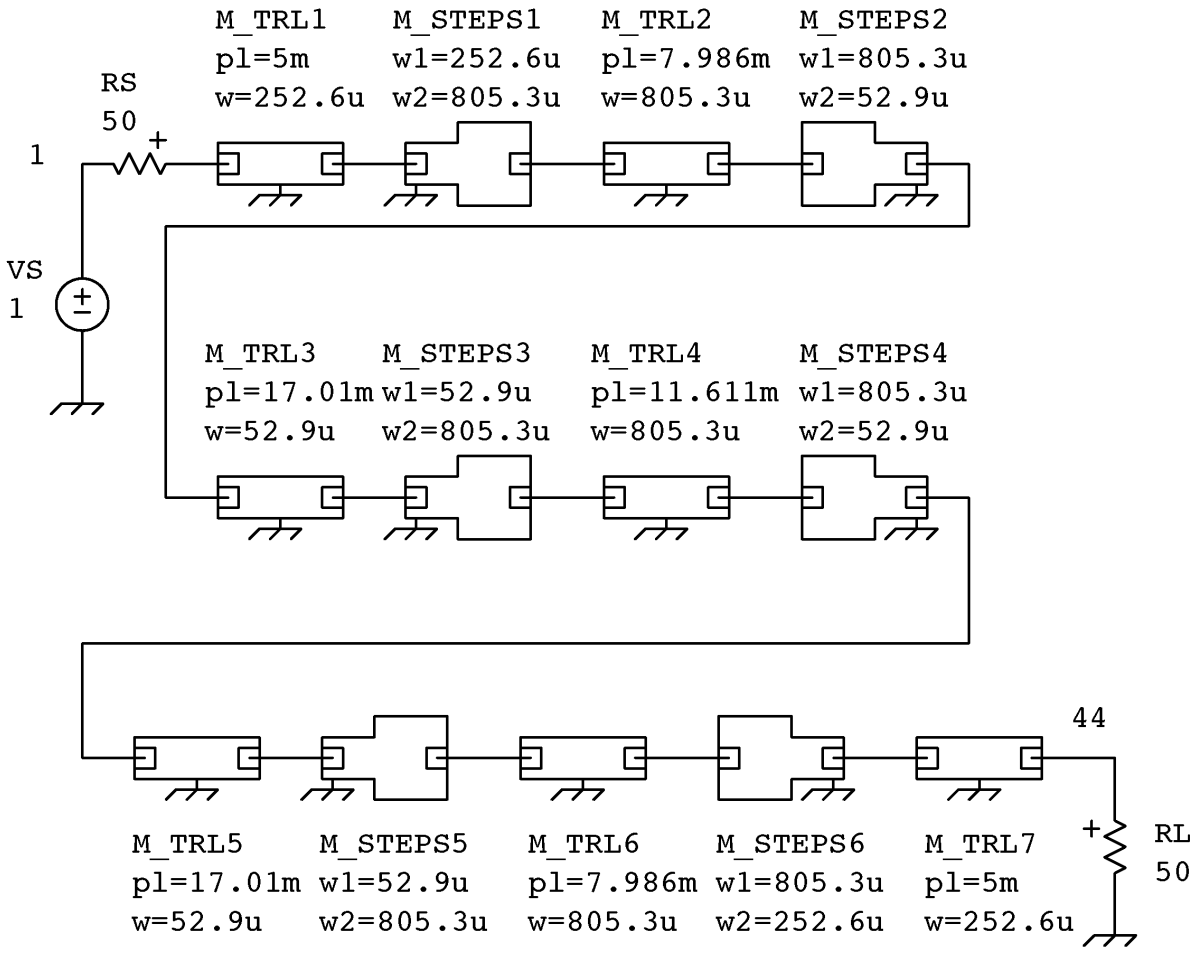


Figure 5–9: Four Pole Chebychev Filter Schematic Diagram

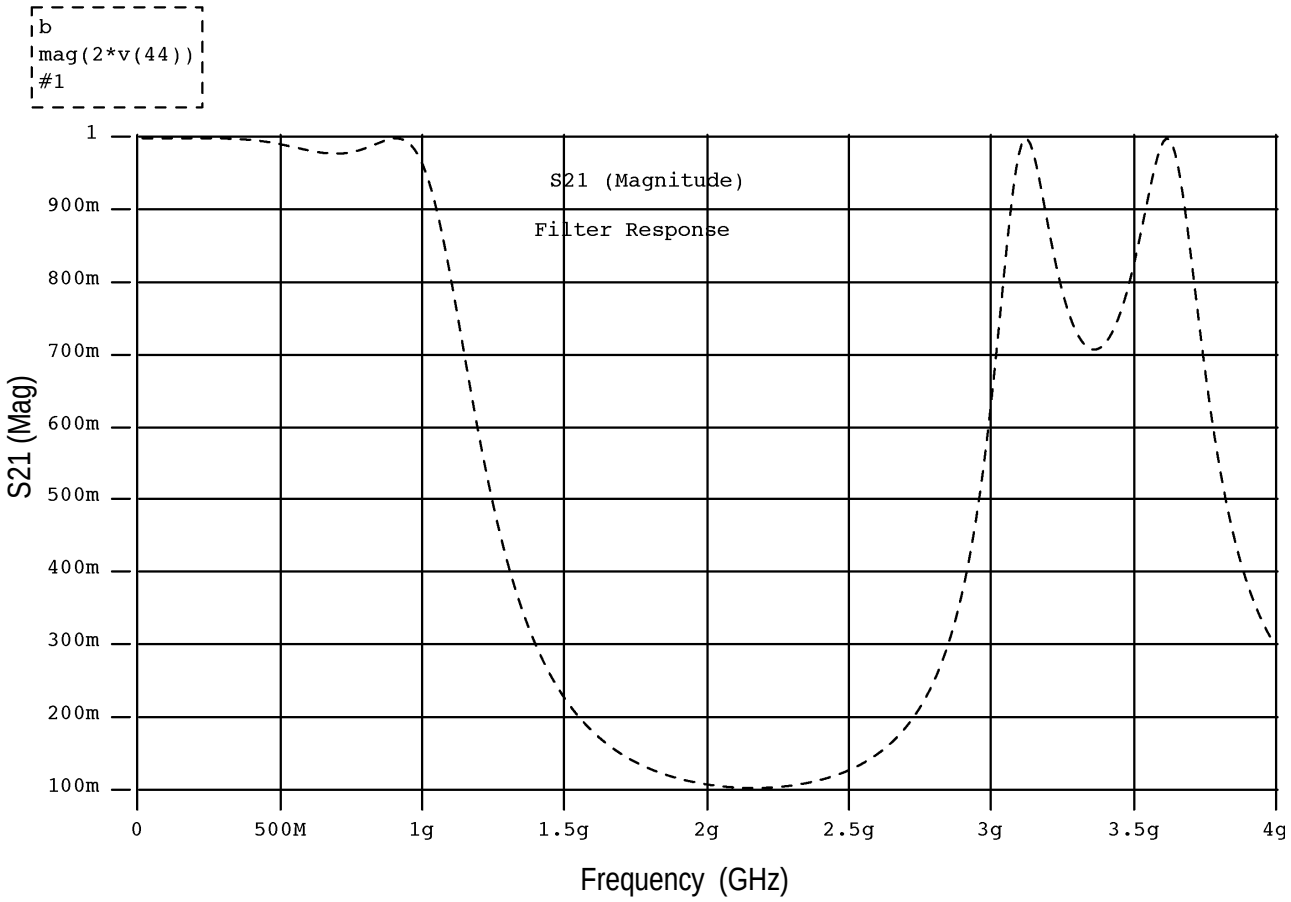


Figure 5–10: S21 Output vs. Frequency

Appendices

Appendix A: BJT Example Circuits

This section uses ADS to test the built–in BJT models for the SHPi IC and GST–1 IC processes. The features covered are:

- Defining a project and adding libraries
- Using Var Sweeps and plot windows to generate different plots
- Using the ADS Workspace and command window to imbed calculated values and plotting commands

It is assumed that the user is familiar with the material in the ADS Tutorial chapter and is able to start ADS and create circuit schematics. Earlier sections of this manual contain additional information about the windows and commands introduced in this appendix. All circuits used in the BJT Example Circuits should be available from the Circuit Browser under the project *Examples_BJT*. If not then they can be loaded from the EDIF format file *~cae/lib/ICO/UTILITIES/ExampleCircuits*.

DC Characteristics

This example investigates some ways of finding DC operating characteristics of a SHPi n2. For this exercise you will plot and simulate the following:

1. I_C, I_B vs. V_{BE} vs. V_{CB} (Forward Gummel Plots)
2. DC Beta vs. I_C vs. Temperature
3. Create a new project in the **Circuit Browser**, by choosing the pop–up button **add project** command and naming the project **BJT dc testing**.
4. Create a test circuit by copying the SHPi–3 template to the new project. Use the pop–up button **copy>circuit to project** command. This copies the library path for SHPi–3 libraries as well as the circuit template to the new project.
5. Figure A–1 shows the modified template circuit. The local **epi** node is removed and two sources are added; VBE is set to dcin and a VCE is set to VCB + dcin. This allows a sweep of both V_{BE} and V_{CE} for different values of V_{CB}. The default value of V_{CB} is set to 0 volts.

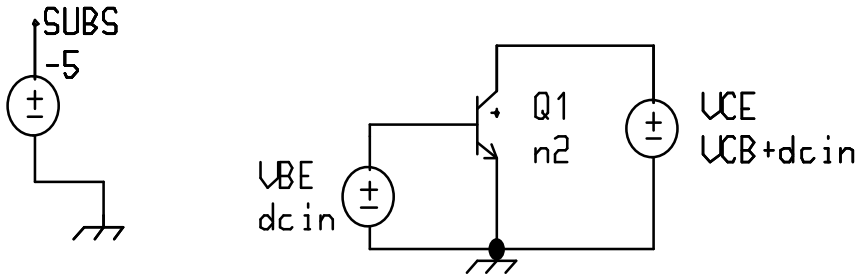


Figure A–1: BJT DC Test Circuit 1

6. Next set up the simulation control program, with a Var Sweep in a dc analysis. The completed control program follows:

```
var sweep temp : -50 130 20
 dc analysis dcin : 0.4 1.0 0.6/100
 enddc
endvar
ic=output(q1,ic)
ib=output(q1,ib)
beta=ic/ib
```

NOTE*. Three variables are defined to make the calculations simpler –* ***ic****,****ib****, and,* ***beta****. These assignments could be made in a plot window after the simulation but since these values are used in several different ways, defining them in the control program is simpler.*

7. After **accepting** the control program execute the simulation. Once the simulation is complete some of the plot functions are explored.

Plotting Simulation Results with the Plot Browser

This section looks at some applications for plot windows; to manipulate the simulation results, to change plotting scales, and to add labels to the finished plots.

1. Starting from the **Circuit Editor** in analysis mode with a simple plot from the simulation which requires no post–processing, create a postage stamp plot of ic and ib using the **function** menu for Q1. The plots of **ic** and **ib** are the same as those defined in the control program, but the **betadc** in the function menu is a single value result and is not a swept value as with the beta defined in the control program.

2. Move the postage stamp plots to a plot window using **copy** and **paste,** then edit the plot. Figure A–2 shows the resulting plot. This plot is too complex

for our purposes. Simplifiy by cutting all the ic and ib traces except for 30° C and 130° C. This is still not in the standard form for a Forward Gummel Plot; the Y axis needs changed to a log scale. Move the cursor to the scale area of the Y–axis and select the pop–up button **axis > log** command. While modifying the axis, change the Y–axis label to **IC, IB** using the pop–up button **setAxisLabel** command. Move the cursor to the X–axis and change the label to **VBE (VCB = 0 v)**. This almost completes the plot, only labels need to be added to the traces in the plot field. Move the cursor to the center of the plot and select the pop–up button **label** command. Enter a label for each trace, the final plot should look like Figure A–3.

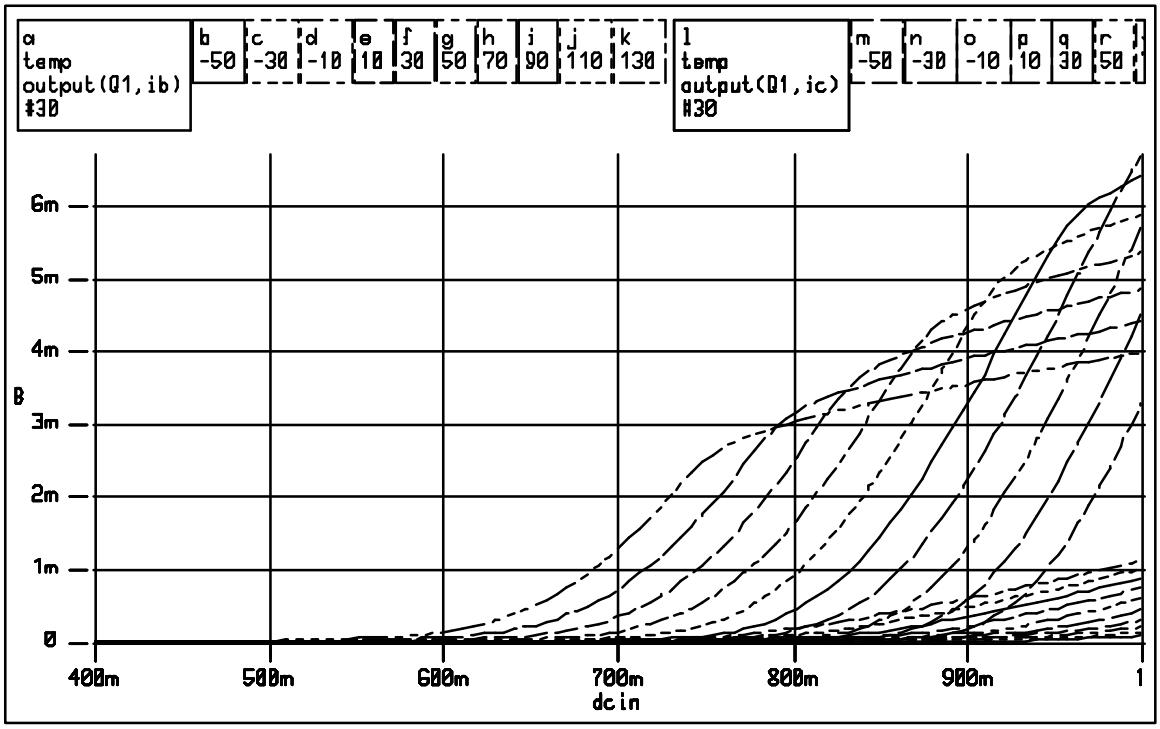


Figure A–2: Forward Gummel Plots

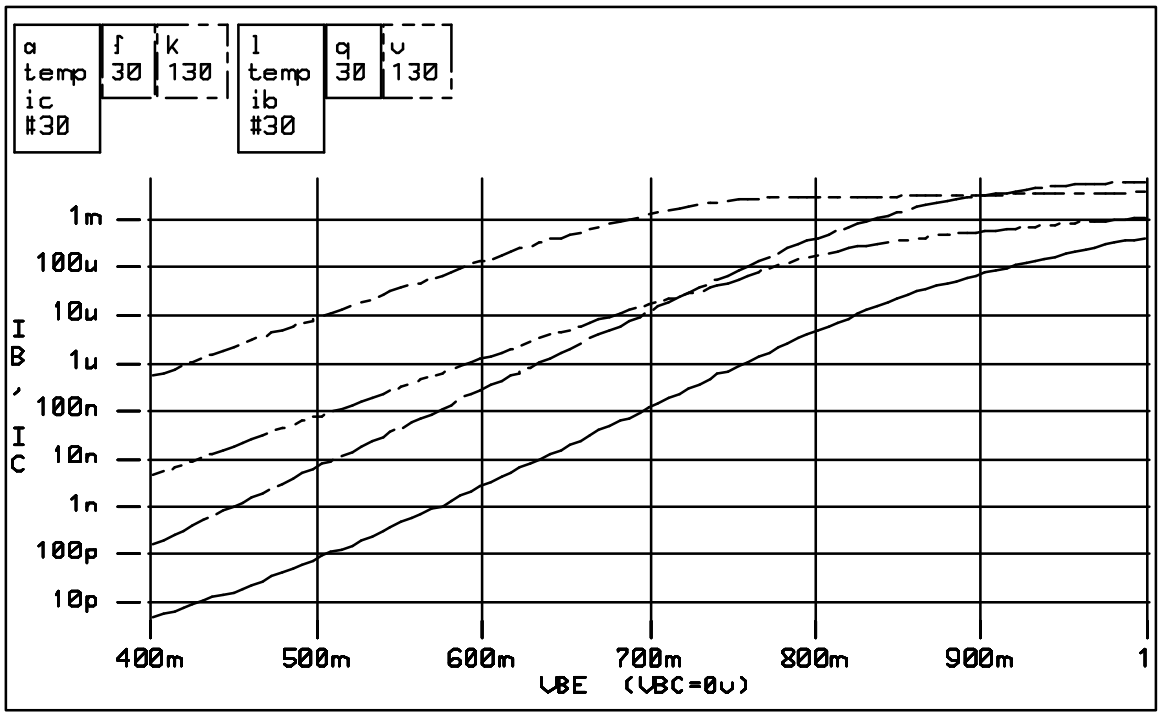


Figure A–3: Final Forward Gummel Plots I_C, I_B vs. V_{BE}

Multi–Variable Simulation Sweeps

You have used some editing features of the **Plot Browser** to create a custom plot, but plots can also be created from more complex simulation sweeps. Start with the previous sweep but this time sweep VCB as well as sweeping Temp.

1. Return to the **Control Program Browser** and add a Var sweep block for **Vcb**. The modified control program is as follows:

```
var sweep Vcb : 0 5 1
 var sweep temp : –50 130 20
 dc analysis dcin: 0.4 1.0 0.6/100
 enddc
 endvar
endvar
ic=output(Q1,ic)
ib=output(Q1,ib)
beta=ic/ib
```

   accept and **simulate** the new control program.

2. The simulation contains too much information for a single plot and requires trimming the results into manageable blocks. First, open a **Workspace**

Editor from the **Launcher**. Within the **Workspace Editor** TekSpice expressions and commands can be created which then can be used for a number of different simulations or circuits without retyping the commands.

3. The last simulation resulted in a multi–dimensional variable which is not directly plottable by ADS . Use the **reduce** command, to create new variables which are plottable. In the **Workspace Editor** enter the following:

```
temp=30
ib30=reduce(ib,temp)
ic30=reduce(ic,temp)
beta30=reduce(beta,temp)
vcb=3
ic30_3=reduce(ic30,vcb)
plot beta30 vs ic30_3 : xaxis=log
```

   This TekSpice command block generates I_B and I_C traces for different values of V_{CB} at 30° C and the DC Beta for the same conditions. The last line of the command block plots DC Beta vs. IC for different values of V_{CB}. When using the **versus** command the second variable must be of a single dimension. This was done with the statement, `ic30_3=reduce(ic30, vcb)`.

4. Open a **Plot Browser** and paste the command block in the command pane of the plot window. Highlighting the command block in the **Plot Browser** and selecting the pop–up button **do it** command produces the plot in Figure A–4.

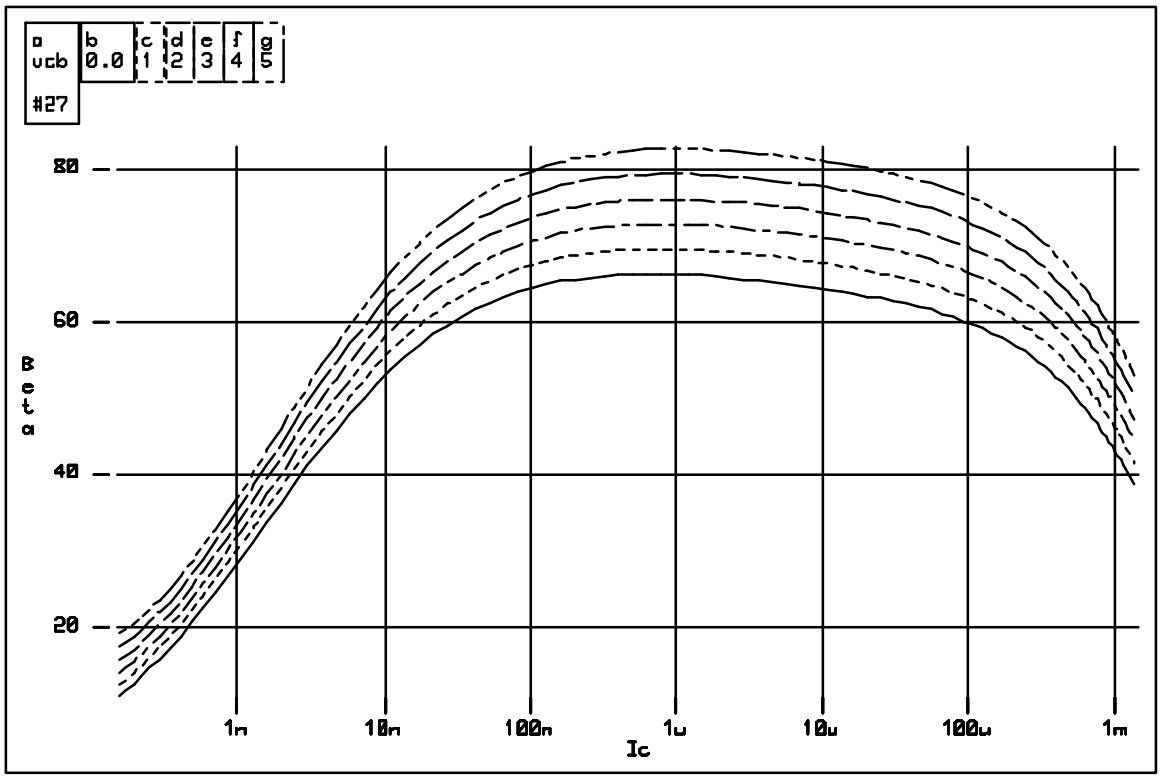


Figure A–4: DC Beta vs. I_C for different values of V_{CB}

Adding Libraries and Comparing Results

Now add the GST–1 process to the project and compare the DC characterstics of a SHPi n2 to a GST–1 G11M02.

1. In the **Circuit Browser** window select the BJT Test Circuit and then select the pop–up button **set libraries** command. From the **Library Path Editor**, select the test circuit and the pop–up button menu **add libraries at end** command. From here you can see all the installed libraries. Select the **GST_QC7>Q7ATALL** library. Remember to **accept** the new library settings before leaving the window.

2. Now change the circuit by exchanging the SHPi n2 model with a GST–1 g11m02 model. After adding the g11m02 model to the parts list pane, select the n2 model in the *Model List Pane* and use the pop–up button **swap** command to modify the circuit. Then **accept** the new circuit.

3. Using the **Workspace Editor** change the control program such that all the earlier sweeps and plots are preformed in a single simulation. The new control program is:

```
var sweep vcb: 0 5 1
 var sweep temp : -50 130 20
 dc analysis dcin: 0.4 1.0 0.6/100
 enddc
 endvar
endvar

; Some definitions
ic=output(Q1,ic)
ib=output(Q1,ib)
beta=ic/ib
temp=-50
ibm50=reduce(ib,temp)
icm50=reduce(ic,temp)
betam50=reduce(beta,temp)
temp=30
ib30=reduce(ib,temp)
ic30=reduce(ic,temp)
beta30=reduce(beta,temp)
temp=80
ib80=reduce(ib,temp)
ic80=reduce(ic,temp)
beta80=reduce(beta,temp)
temp=130
ib130=reduce(ib,temp)
ic130=reduce(ic,temp)
beta130=reduce(beta,temp)
vcb=0
ib30_0=reduce(ib30,vcb)
ic30_0=reduce(ic30,vcb)
ib130_0=reduce(ib130,vcb)
ic130_0=reduce(ic130,vcb)
vcb=3
ib30_3=reduce(ib30,vcb)
ic30_3=reduce(ic30,vcb)
ib130_3=reduce(ib130,vcb)
ic130_3=reduce(ic130,vcb)

; Plotting commands
plot ib30_0 ic30_0 ib130_0 ic130_0 : yaxis=log
plot beta30 vs ic30_3 : xaxis=log
wiremenu @-@1 dif(@-@1)
circuitmenu @-@1 dif(@)
```

4. After completing the simulation compare the DC characteristics of a SHPi n2 and a GST–1 g11m02. Open a new plot window and **cut** and **paste** the desired curves from the n2 and g11m02 simulation. Adding labels results in the plots shown in Figure A–5 and Figure A–6.

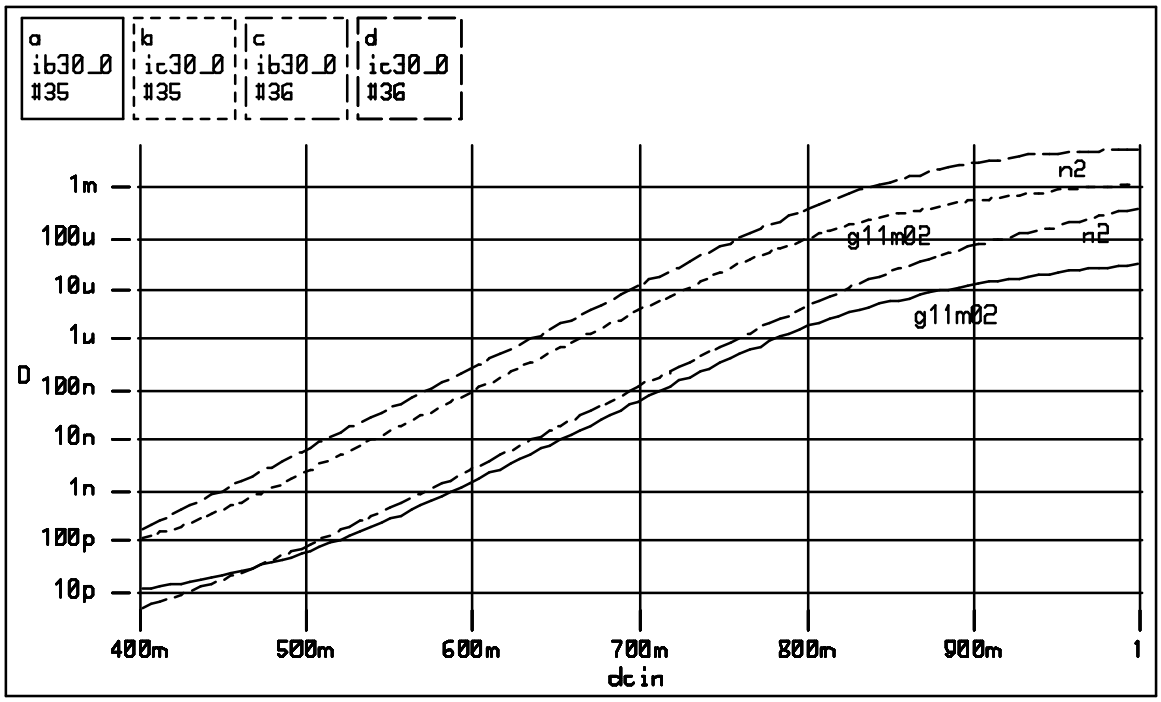


Figure A–5: Gummel Plot of a SHPi n2 and a GST–1 g11m02

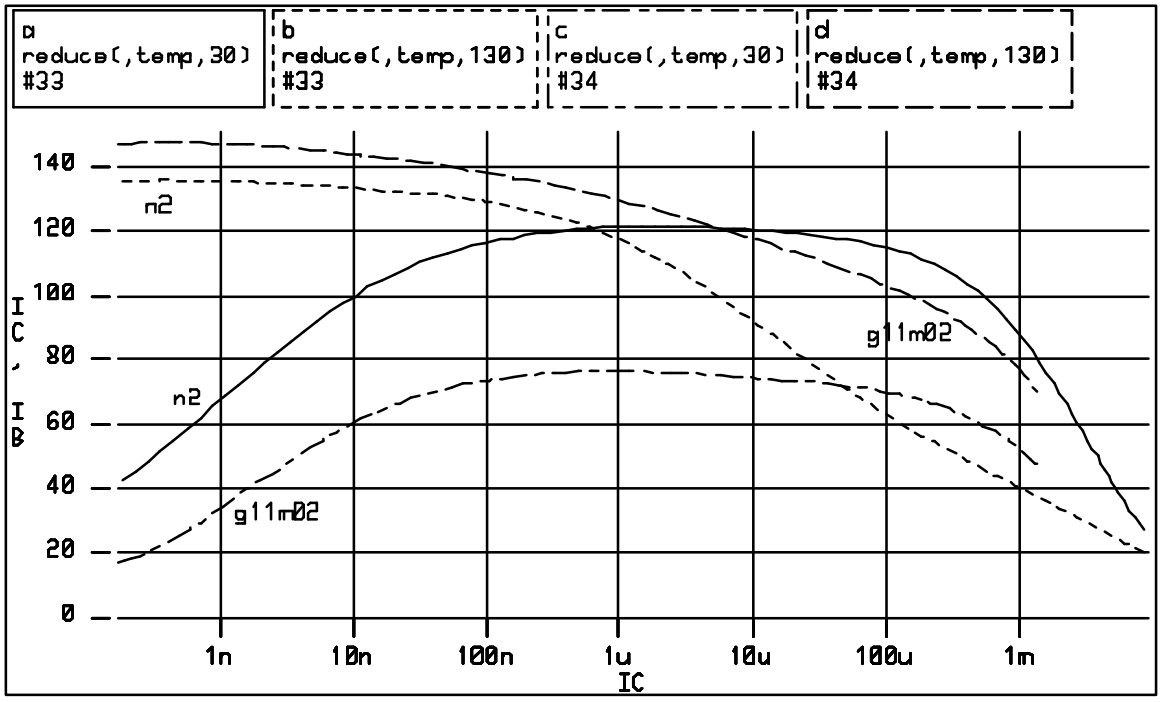


Figure A–6: DC Beta vs. Ic of a SHPi n2 and a GST–1 g11m02

DC Transfer Characteristics

This example presents a test circuit and control program for finding the DC transfer characteristics of a SHPi n2 transistor. Figure A–7 shows the test circuit.

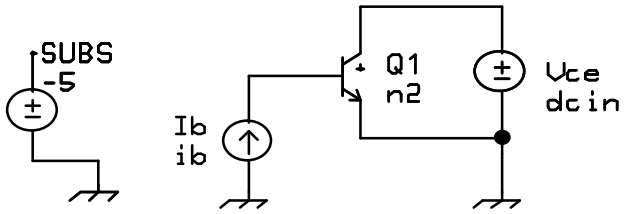


Figure A–7: DC Transfer Circuit

The control program is,

```
var sweep ib: 0u 12.5u 2.5u
dc analysis dcin: 0 5 5/100
enddc
endvar
plot output(q1,ic)
wiremenu @-@1 dif(@-@1)
circuitmenu @-@1 dif(@)
```

Figure A–8 shows the plot resulting from this simulation.

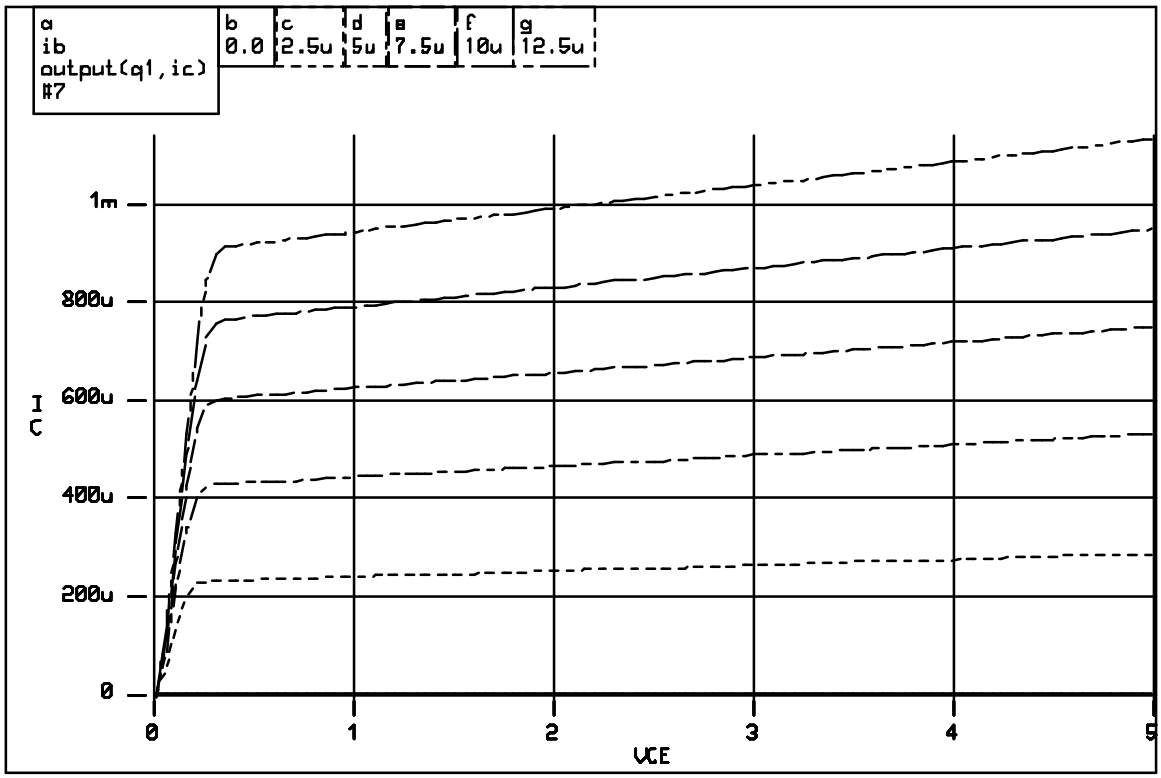


Figure A–8: DC Transfer Characteristics of I_c vs. V_{ce}

AC Characteristics

This section looks at some ways of finding the AC response characteristics of a GST–1 g11m02 transistor. Figure A–9 shows the circuit used for finding these characteristics. This circuit is a differential pair which uses ideal sources to preform AC sweeps. The current controlled current source is set up to self bias the circuit as I_C swept .

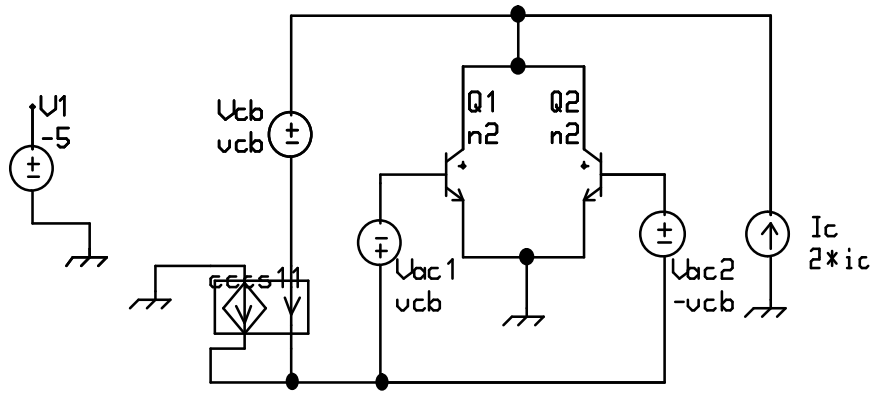


Figure A–9: AC Test Circuit

First it is necessary to find the ac beta, to calculate f_T as a function of both I_C and Temperature. While f_T is available from the **function** menu of the transistor this value of f_T is calculated from device parameters and bias conditions and is not a result of the ac simulation. Figure A–10, Figure A–11, Figure A–12 and Figure A–13 show these results. The control program command block follows:

```
var sweep temp: -33 127 20
var sweep vcb: 0.0 2.5 0.5
;sweep step needed for Vce 0.7- 5.7
 var sweep ic: 1e-5 1e-2 5 type=dec
 ac analysis freq: 1 100g 5 type=dec
 endac
 endvar
endvar
endvar

; Useful Variable Definitions
ic=mag(output(q1,ic))
;we only need the magnitude here
ib=mag(output(q1,ib))
beta=ic/ib ;AC beta

; Create beta for different Temp
temp=-33
betam33=reduce(beta,temp)
temp=27
beta27=reduce(beta,temp)
temp=127
beta127=reduce(beta,temp)

; Create beta for different Temp and Vcb
vcb=0
 beta27_0=reduce(beta27,vcb)
```

```
 beta_0=reduce(beta,vcb)
 vcb=1.5
 beta27_3=reduce(beta27,vcb)
 beta_3=reduce(beta,vcb)
 vcb=2.5
 beta27_5=reduce(beta27,vcb)
 beta_5=reduce(beta,vcb)

; Find ft from beta results for different Vcb and fixed temp
 freq=1g
;we have to choose a freq on the rolloff slope
 ft_vs_ic_0=reduce(beta27_0,freq)*freq
 ft_vs_ic_3=reduce(beta27_3,freq)*freq
 ;here we find when the ac beta roll-off reaches 1
 ft_vs_ic_5=reduce(beta27_5,freq)*freq

; Find peak ft vs temp for different Vcb
 ft_ic_0=reduce(beta_0,freq)*freq
 peakft_0=imag(peak(ft_ic_0));peak returns value and position
 ft_ic_3=reduce(beta_3,freq)*freq ;imag' reduces this to a
;value only
 peakft_3=imag(peak(ft_ic_3))
 ft_ic_5=reduce(beta_5,freq)*freq
 peakft_5=imag(peak(ft_ic_5))

; Plotting Commands
 plot beta27_3 : xaxis=log yaxis=log
 plot ft_vs_ic_0 ft_vs_ic_3 ft_vs_ic_5 : xaxis=log
 plot ft_ic_3 : xaxis=log
 plot peakft_0 peakft_3 peakft_5

wiremenu @-@1 db(@) db(@-@1) phase(@) phase(@-@1) mag(@)
mag(@-@1) lowpass(@) lowpass(@-@1)
circuitmenu @-@1 db(@) phase(@) mag(@) lowpass(@)
```

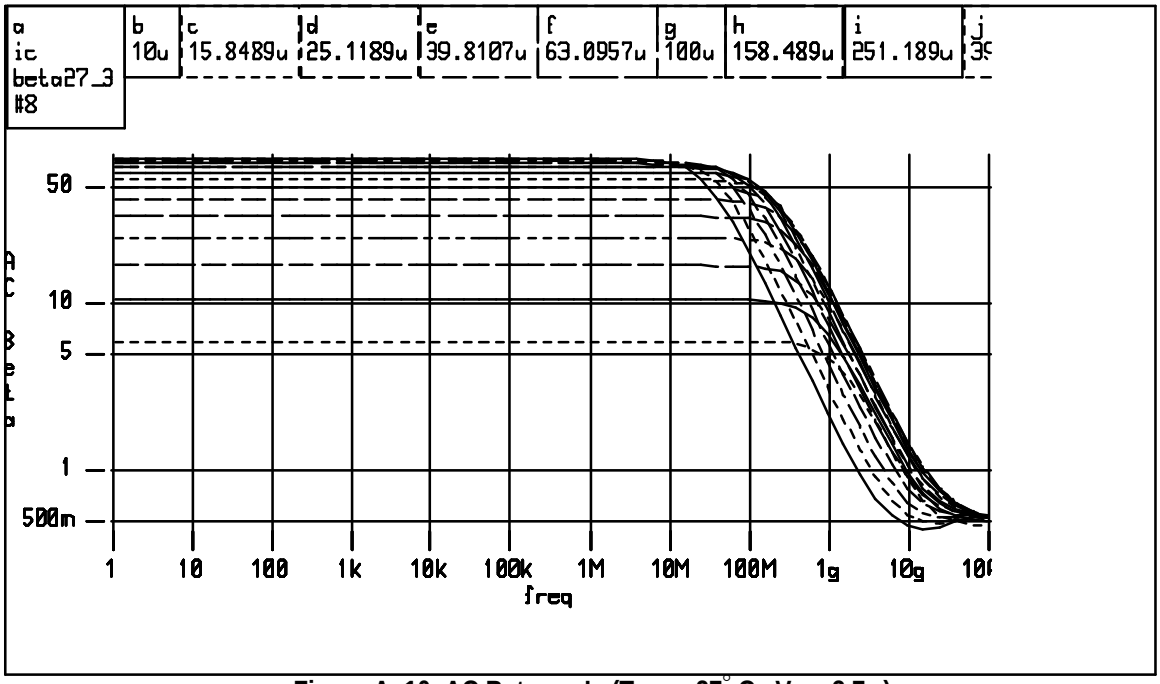


Figure A–10: AC Beta vs. I_C (Temp=27° C., V_{CE}=3.7v.)

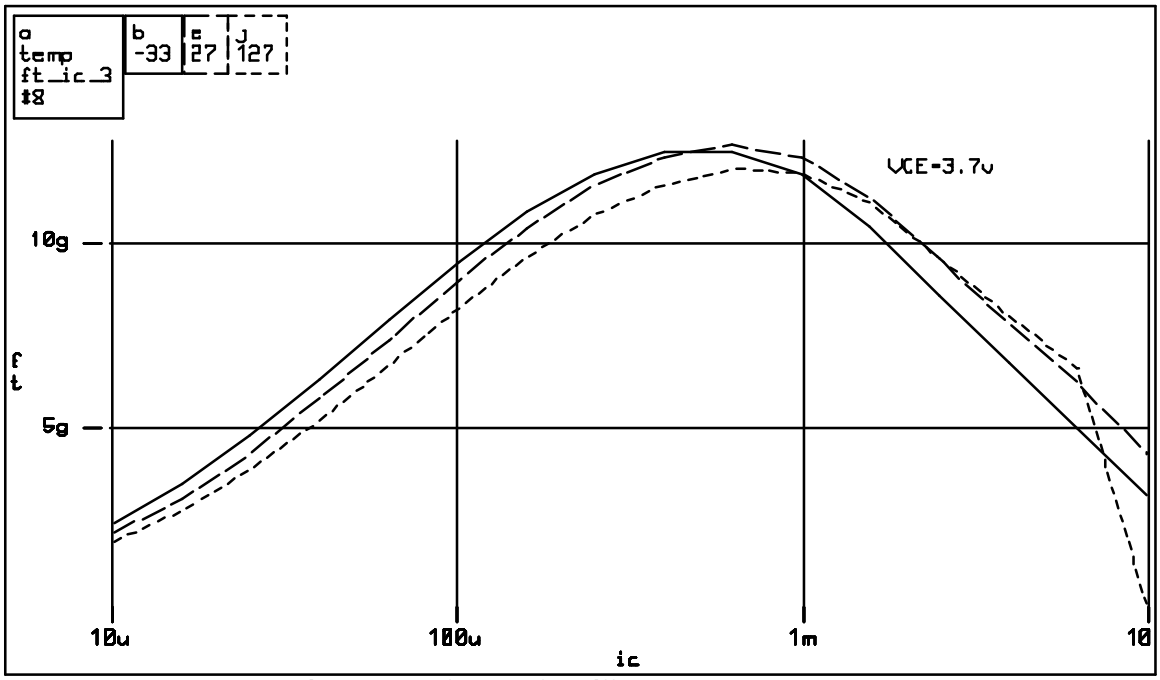


Figure A–11: f_T vs. I_C for Different Temperatures

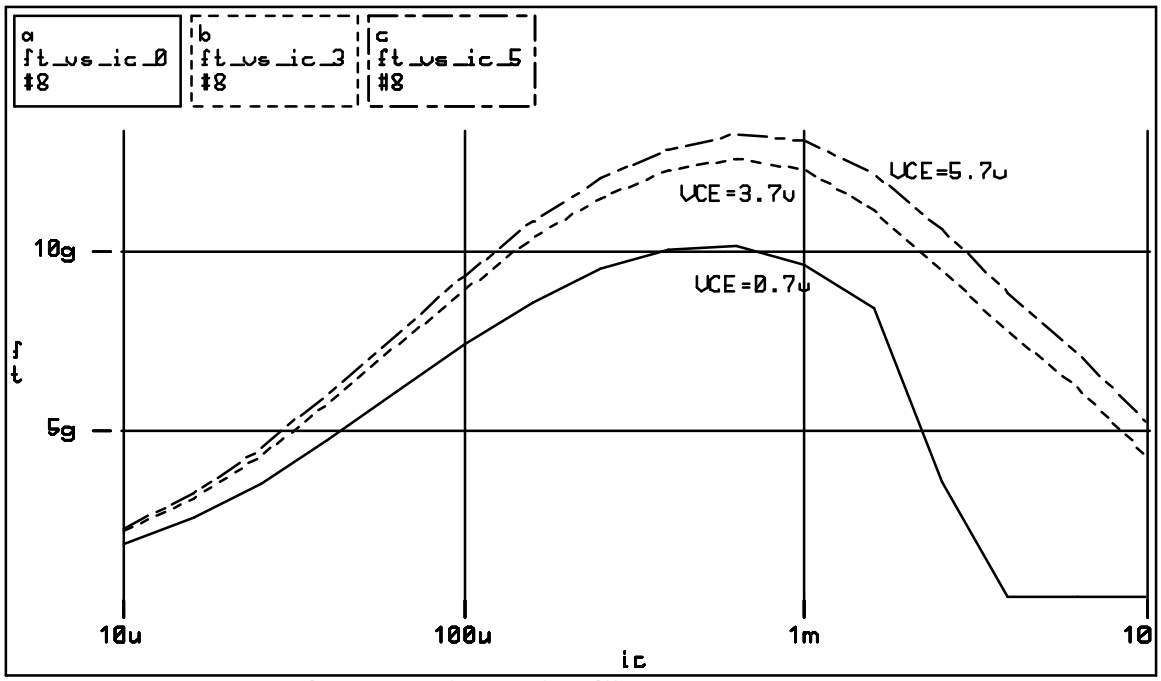


Figure A–12: f_T vs. I_C for Different V_{CE}

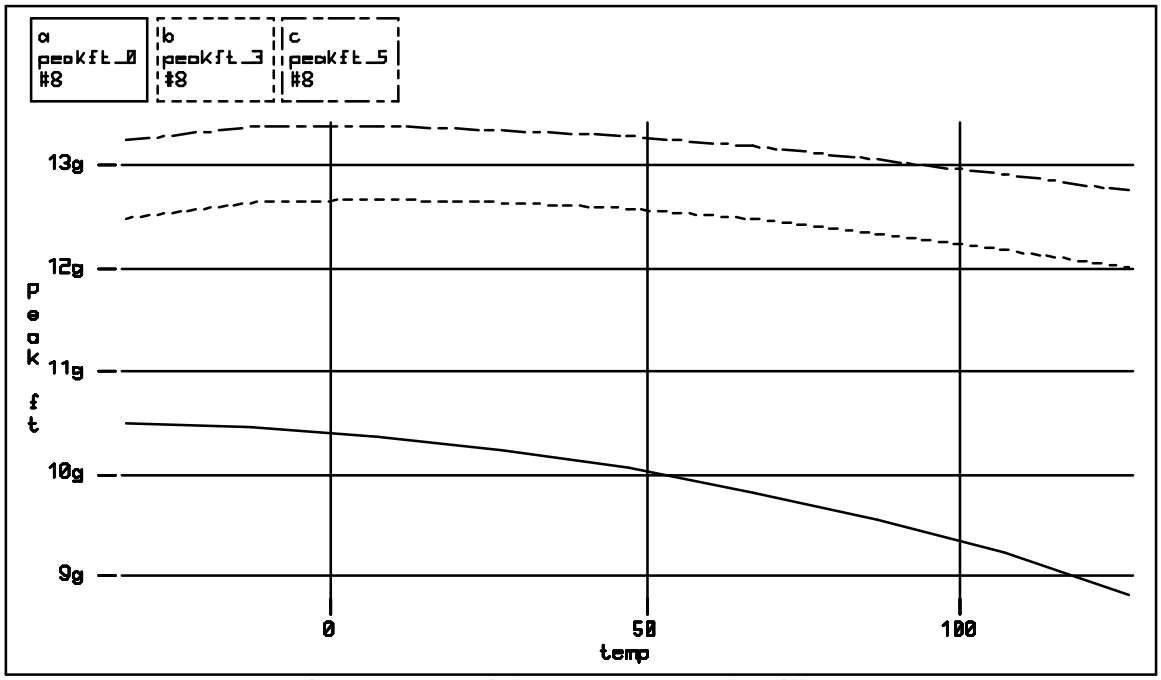


Figure A–13: Peak f_T vs. Temperature for Different V_{CE}

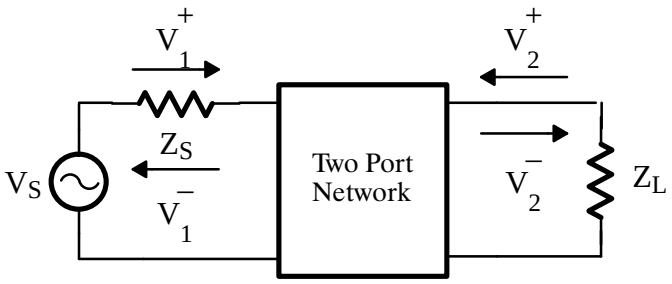
Appendix B: RF Example Circuits

This section shows the capabilities of ADS and TekSpice to perform simulations of RF circuits. All circuits used in the RF Example Circuits should be available from the Circuit Browser under the project *Examples_RF*. If not then they can be found in EDIF format in the file ”~cae/lib/ICO/UTILITIES/ExampleCircuits”. The RF circuit measurements examined here are as follows:

- Voltage Standing Wave Ratio (VSWR) and Power Gain
- Noise Figure Measurement (NF)
- 3rd Order Intercept (IP_3)
- 1 db Compression

Introduction

In RF systems the characteristics of a circuit are often defined in terms of incident and reflected traveling waves. Figure B–1 shows a two port network with incident waves, V_1^+ and V_2^+, and reflected waves, V_1^- and V_2^-. The S–parameter relationships are also shown defined in terms of these waves.

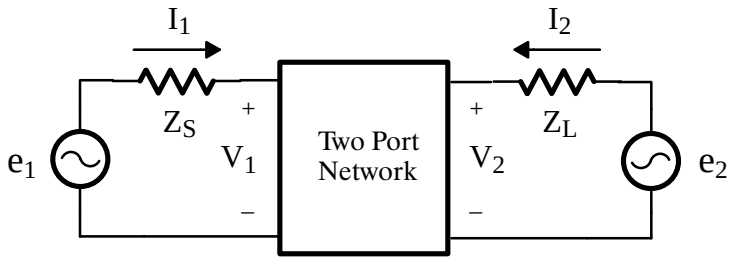


$$V_1^- = S_{11}V_1^+ + S_{12}V_2^+$$

$$V_2^- = S_{21}V_1^+ + S_{22}V_2^+$$

Figure B–1: Two Port Network

With ADS it is necessary to define the S–parameter relationships in terms of simulation voltages and currents. Figure B–2 shows the modified two port network with both input and output sources and the definitions of the traveling waves in terms of the network’s voltages and currents. Note that positive current flows into the network.



$$V_1^+ = V_1 + Z_S I_1\,, \quad V_2^+ = V_2 + Z_L I_2$$

$$V_1^- = V_1 - Z_S I_1, \quad V_2^- = V_2 - Z_L I_2$$

Figure B–2: Modified Two Port Network

The S–parameter relationship in terms of voltages and currents is rewritten as,

$$V_1 - Z_S I_1 = S_{11}\left(V_1 + Z_S I_1\right) + S_{12}\left(V_2 + Z_L I_2\right)$$

$$V_2 - Z_L I_2 = S_{21}\left(V_1 + Z_S I_1\right) + S_{22}\left(V_2 + Z_L I_2\right).$$

If the network is terminated in a matched impedance where $Z_S = Z_L$,it can be seen from Figure B–2 that the input and output test sources can be defined as $e_1 = V_1 + Z I_1$ and $e_2 = V_2 + Z I_2$ and the following relationships result.

If $e_1 = 1$ and $e_2 = 0$ then

and $\boldsymbol{S_{11} = 2V_1 - 1}$ $\boldsymbol{S_{21} = 2V_2}$

If $e_1 = 0$ and $e_2 = 1$ then

$\boldsymbol{S_{22} = 2V_2 - 1}$ and $\boldsymbol{S_{12} = 2V_1}$

These last four equations define how the S–parameters are calculated from the ADS simulation results.

At this point it is helpful define the reflection coefficient Γ and its relationship to the S–parameters, S_{11} and S_{22}. The reflection coefficient Γ can be defined as,

$$\Gamma_{in} = \frac{V_1^-}{V_1^+} = \frac{V_1 - Z_S I_1}{V_1 + Z_S I_1} \ .$$

Applying the earlier test source relationships gives the following:

If $e_1 = 1$ and $e_2 = 0$ then

$$\boldsymbol{\Gamma_{in} = 2V_1 - 1 = S_{11}}$$

If $e_1 = 0$ and $e_2 = 1$ then

$$\boldsymbol{\Gamma_{out} = 2V_2 - 1 = S_{22}} \ .$$

It is important to note the difference between the reflection coefficient Γ and S–parameters S_{11} and S_{22}. S–parameters are based on condition that that all ports of the network are terminated in reference loads. With Γ there are no constraints on the input or output impedances, therefore, Γ changes based on the terminations of the network. In this example Γ is calculated for an 50Ω reference network. If the network's termination resistors change the value of Γ also changes.

Voltage Standing Wave Ratio (VSWR) and Power Gain

The voltage standing wave ratio or VSWR is another common RF specification. The VSWR for the test environment is calculated from Γ as shown below.

$$\boldsymbol{VSWR = \frac{1 + mag(\Gamma)}{1 - mag(\Gamma)}}$$

S–parameters also simplify calculating the power gain, G. For a matched network the forward power gain is given by the equation below. The power gain is generally plotted in dB, using the **db** function. The power gain can be plotted from the S–parameter expression as follows,

$$\boldsymbol{forgain = dB(S_{21}) = 20\log(|S_{21}|) = 10\log\left(|S_{21}|^2\right).}$$

The circuit used for these measurements is a low noise amplifier (LNA01A) that was designed and fabricated using the Tektronix SHPi process. A schematic of this circuit is shown in Figure B–3.

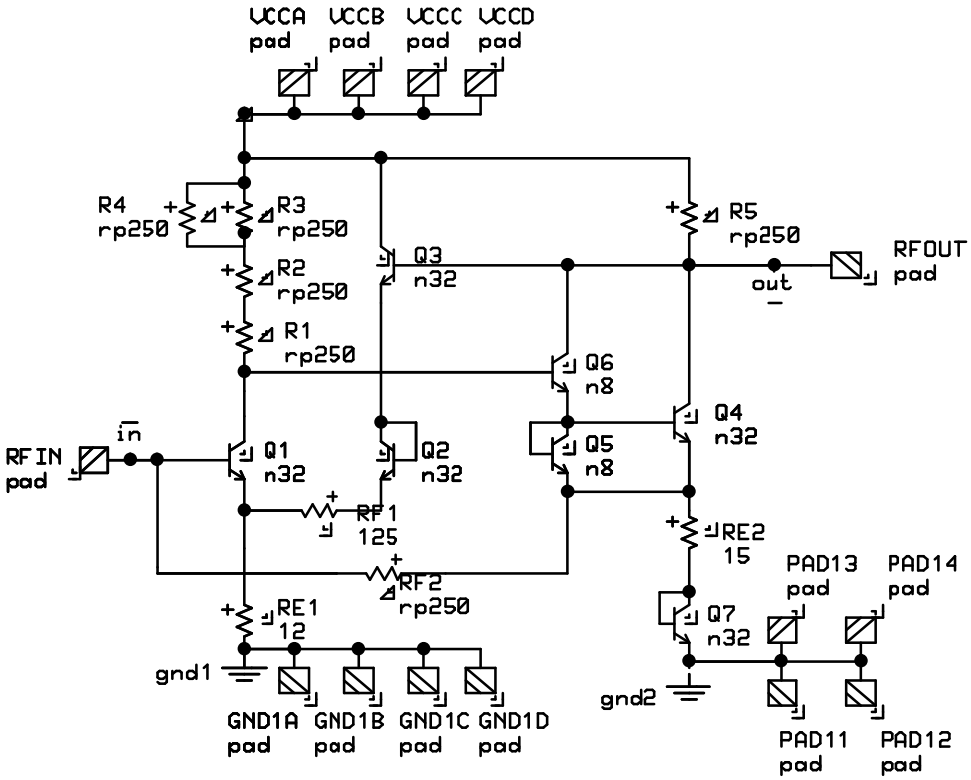


Figure B–3: LNA01A Subcircuit

The test circuit shown in Figure B–4 is used to find the S–parameters for the LNA01A amplifier.

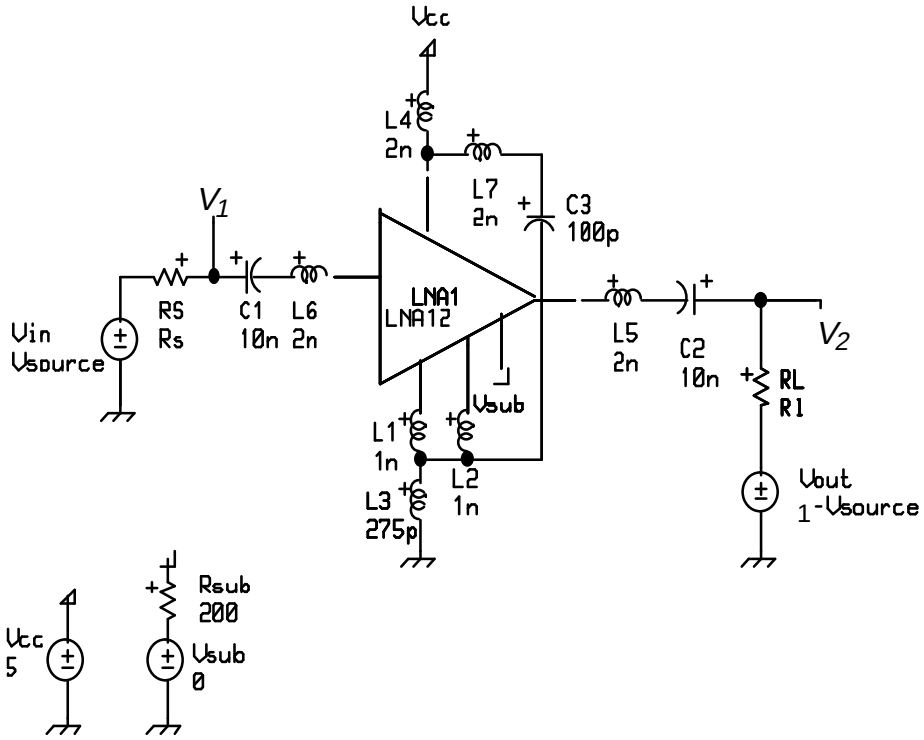


Figure B–4: S–Parameter Test Circuit

In the test circuit shown in Figure B–4 the bond wire inductances have been modeled with L3, L4, L5, and L6. DC de–coupling has also been added with C1, C2, and C3. Note the second voltage source at the output of the LNA01A, this source is set up such that when Vin is 1 v, Vout is zero and it does not affect the circuit. When Vin is 0, Vout applies 1 v at the output of the LNA01A. Both the input and output S–parameters are calculated in one simulation.

The following ADS control program extracts the S–parameter data from the test circuit in Figure B–4. The S–parameter expressions and the associated relationships for Γ, VSWR, and G are as follows:

```
Rs=50          ; source resistance
Rl=50          ; load resistance
dcin=0         ; DC value of Vin source
Vsource=1      ; AC value of Vin source
Vsph=0         ; AC phase value of Vin source

var sweep Vsource : 0 1 1
ac analysis freq: 10e6 10e9 20 type=dec
endac
endvar
```

```
; extract data for signal applied to output
Vsource=0
s12=reduce(2*v(V1),Vsource)
s22=reduce(2*v(V2)-1,Vsource)

; extract data for signal applied to input
Vsource=1
s11=reduce(2*v(V1)-1,Vsource)
s21=reduce(2*v(V2),Vsource)

; Gain Calculations
forgain=db(s21)
revgain=db(s12)

; Input Voltage Standing Wave Ratio
InVSWR=(1+mag(s11))/(1-mag(s11)) ; input VSWR

; Output Voltage Standing Wave Ratio
OutVSWR=(1+mag(s22))/(1-mag(s22)); output VSWR

; Plotting commands
plot forgain
plot revgain
plot InVSWR
plot OutVSWR
```

Plots of simulation results from the above control program are shown in Figure B–5, Figure B–6, Figure B–7, and Figure B–8.

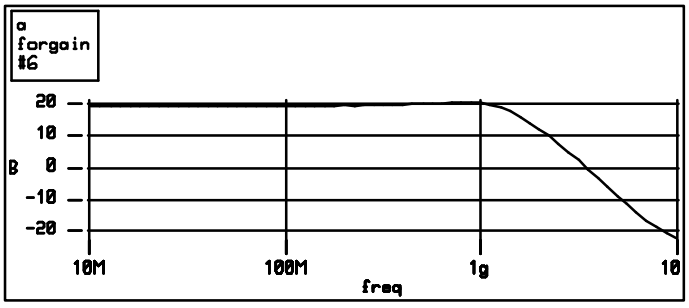


Figure B–5: Forward Gain

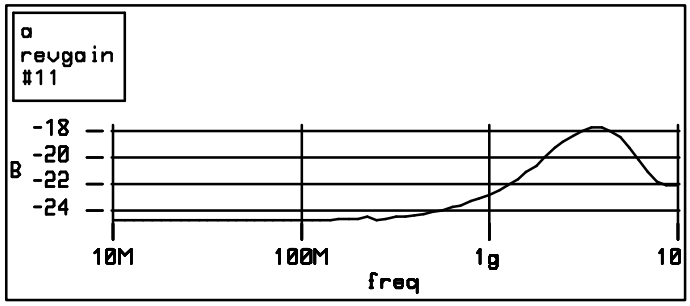


Figure B–6: Reverse Gain

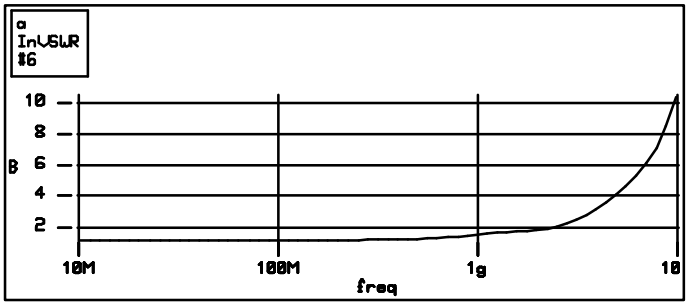


Figure B–7: Input VSWR

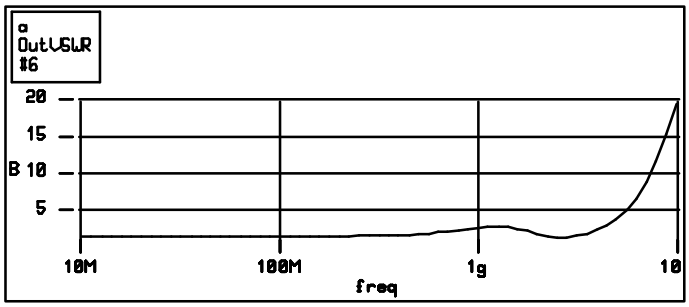


Figure B–8: Output VSWR

Noise Figure Measurement

ADS can be used to find the noise figure of the LNA01A. The noise figure is a measure of the degradation of the signal power to noise power ratio from the input to the output of a device or subcircuit. The SHPi models and User Models include noise sources. These sources simulate the noise added to the signal by components of the LNA01A. The noise factor is defined at the input signal to noise ratio divided by the output signal to noise ratio,

$$F = \frac{S_i/N_i}{S_o/N_o} \geq 1.$$

The noise figure is the $10\log_{10}$ plot of the noise factor.

As with most RF parameters the noise factor is a power measurement. It is necessary to translate noise factor power terms into the simulation voltages and currents. The output noise power, N_0 can be defined in terms of the noise analysis variable *onoise* as follows,

$$N_o = \frac{(onoise)^2}{R_L} .$$

In addition to *onoise* ADS produces a variable *inoise* which calculates the equivalent noise voltage at the input source. In defining *onoise* in terms of the S–parameter input divider network must included. Therefore, *inoise* is related to *onoise* by the following,

$$onoise = \left|\frac{S_{21}}{2}\right| inoise .$$

The input noise power, N_i is defined as the thermal noise generated by a matched resistor,

$$N_i = 4kT , \quad \text{where} \quad T = temp + 273.15$$

k = Boltzmans constant

The following is obtained using the above definitions and substitutions:

$$F = \left(\left|\frac{2}{S_{21}}\right|^2\right)\left(\frac{onoise^2}{N_i}\right)$$ This is the noise figure definition

$$F = \frac{|inoise|^2}{4kTR_L}$$ After substitution

$$NF = 10\log\left[\frac{|inoise|^2}{4kTR_L}\right]$$ This is our final result, the *Noise Figure*, NF

The input noise element is defined as the noise generated from a matched resistor at *temp = 25° C,* this definition is met by the source resistor used in the previous test circuit. However, the noise added by the output termination of the LNA01A should not be included. In order to eliminate the noise from the load resistor noise a noiseless resistor is simulated. The noiseless resistor is simply a current controlled voltage source (CCVS) with its transfer function set to a variable R_L. This is shown in Figure B–9.

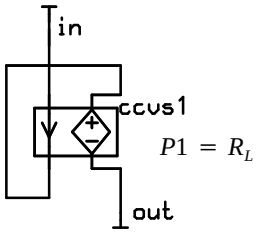


Figure B–9: Noiseless Resistor Subcircuit

Figure B–10 shows the noise figure test circuit for this example. The noise sources are summed at the node **out** using the input source V_{in} as a reference. The results of the noise analysis returned is the RMS sum of the individual noise sources at each frequency of the AC simulation.

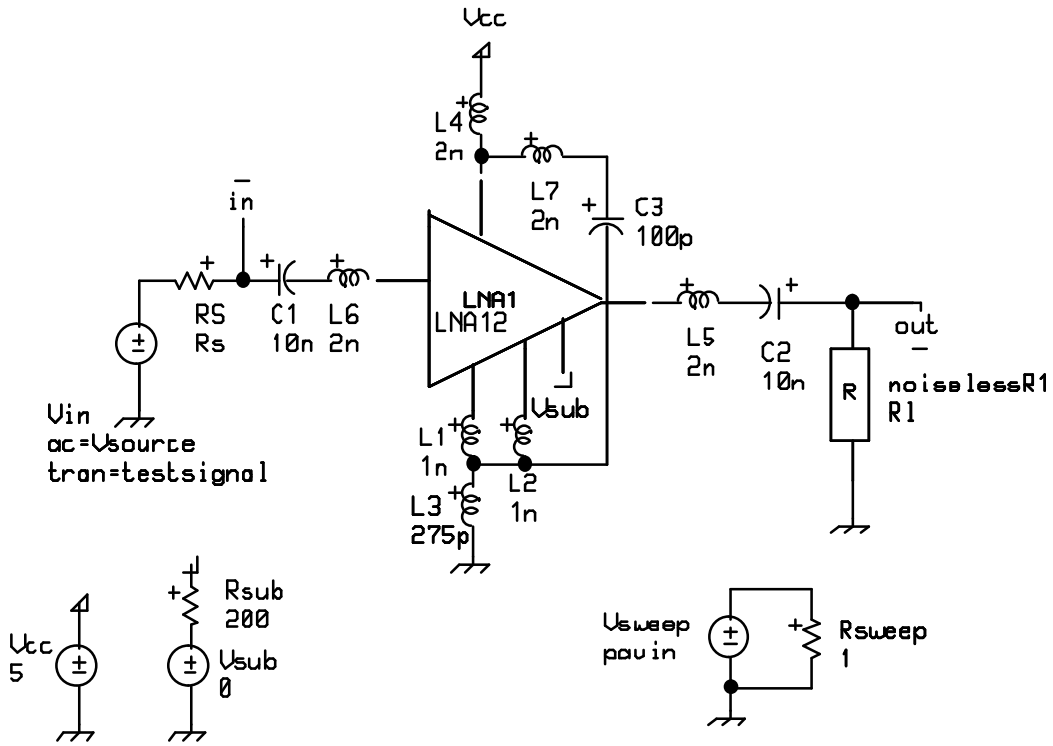


Figure B–10: Noise Figure Test Circuit

Translating the previous noise figure relationships results in the following control program:

```
Rs=50        ; Source Resistance
Rl=50        ; Load Resistance
dcin=0       ; DC level of input Source
pavin=-30    ; Power variable for transient input source
Vsource=1    ; AC value of input and output source
Vsph=0       ; Phase of input and output source
T=(temp+273.15)
k=1.380e-23; k is Boltzmann's constant

ac analysis freq: 10e6 10e9 20 type=dec
noise v(out) Vin
endac

; Noise figure Calculation
probe inoise=inoise
F=mag(inoise)^2/(4*k*T*Rl)
NF=10*log10(F)

; Plotting commands
plot NF
```

Figure B–11 shows the results of the above noise figure simulation.

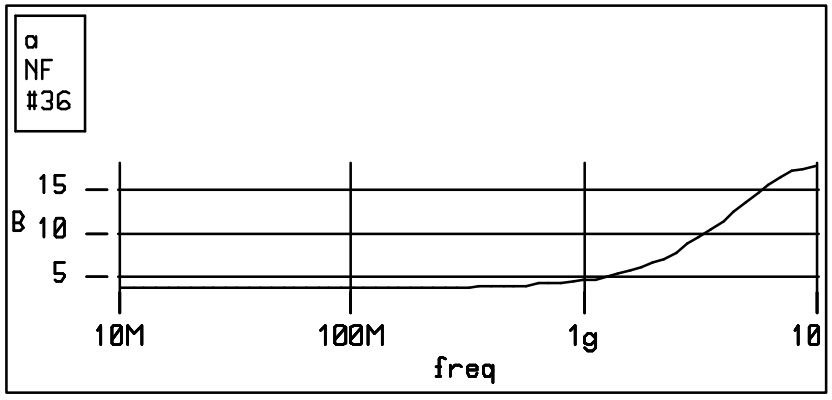


Figure B–11: NF vs. Frequency

The plot of the NF versus frequency shows the noise performance of the LNA01A but from a design standpoint it would be helpful to have information about the individual noise sources. In addition to calculating and plotting the noise figure, ADS saves information on the noise generated by each component of the LNA01A. These noise results are easily accessed through the **Circuit Editor** in the analysis mode by opening the **Noise Table Browser** from the pop–up button **open > noise table** command. The noise table makes available noise data on; total noise for each component, onoise & inoise, and the noise from the device model subcircuits. Figure B–12 shows the noise table with plots of total noise for each component of the LNA01A in units of RMS voltage and $\frac{(Volts)^2}{Hertz}$.

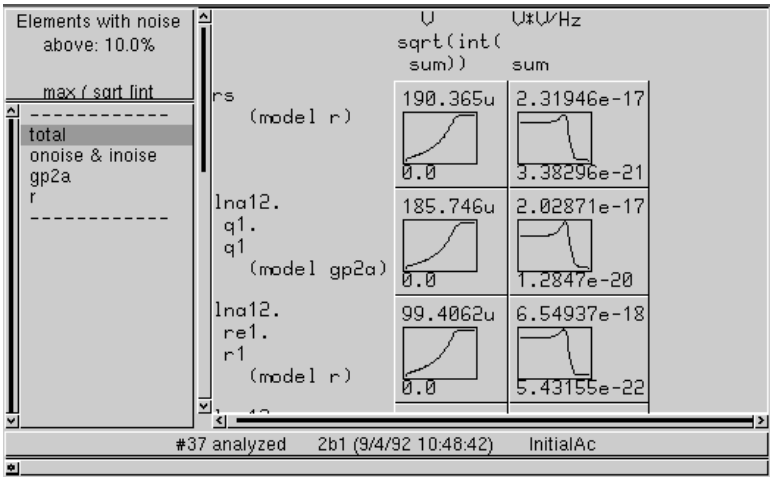


Figure B–12: ADS Noise Table

3rd Order Intercept (IP_3)

A detailed discussion of distortion and the techniques required to calculate and measure these effects are beyond the scope of this discussion. A brief presentation is given to help define variables and clarify this example. For a full and detailed description of the material presented a list of references is provided at the end of this section.

The IP_3 is a measure of the distortion due to non–linear components of a device or subcircuit. The output harmonics are classified by their order with the first order being the desired fundamental and all other higher orders being harmonic distortions. These effects become most noticeable when the input is a ”two tone” signal, made up of two relatively closely spaced frequencies. The major problem arises with the third order harmonic terms,

$$V_{in} = \sin \omega_1 t + \sin \omega_2 t$$

$$V_{out} = a_0 + a_1 V_{in} + a_2 V_{in}^2 + a_3 V_{in}^3 + \cdots$$

The 3rd products are:

$$3\omega_1,\ 3\omega_2,\ 2\omega_1 + \omega_2,\ 2\omega_2 + \omega_1,\ 2\omega_1 - \omega_2,\ and\ 2\omega_2 - \omega_1$$

In general, all but the last two terms are spaced far enough from the fundamental to be easily filtered. The output power of the last two terms,
$2\omega_1 - \omega_2,\ and\ 2\omega_2 - \omega_1$, is what is of interest for the IP_3 measurement.

A plot of output power versus input power in dB for an amplifier has a slope of one for small signal levels, but as the input power is increased the output signal starts to distort by diverting part of the input power to the higher order harmonics. See Figure B–13.

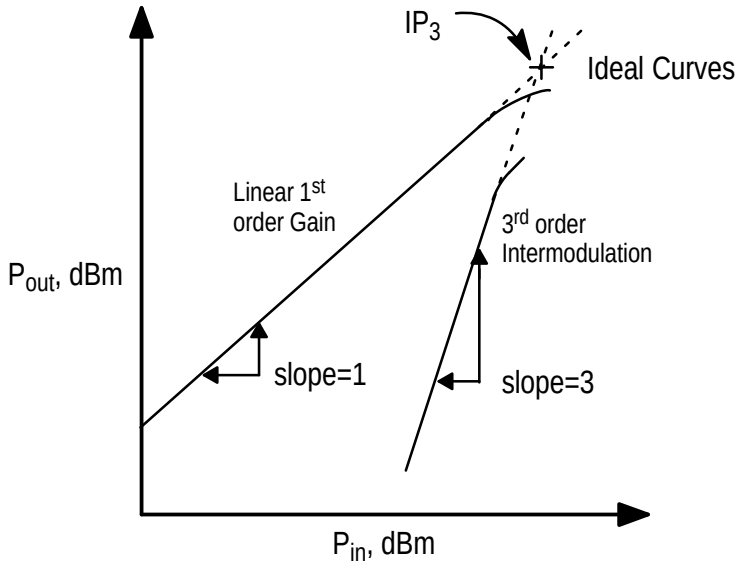


Figure B–13: Input Power vs. Output Power

If the linear part of the small signal gain curve is extended it intercepts the third order power gain curve at a point, IP_3.

IP_3 is calculated from the output power levels of frequency components as follows:

$$IP_3 = P_{out} + \frac{IMR_3}{2}$$

Where

P_{out} = output power of the fundamental in dBm

IMR_3 = the difference between the output power of the fundamental and the third order term in dBm

A transient analysis is required to perform the necessary simulation for the IP_3 measurement. The time domain output of the transient analysis must be transformed into the frequency domain for these calculations, which is done by using the ADS FFT waveform function.

This example presents some general guidelines to aid in calculation to the FFT, but a detailed discussion of the FFT (Fast Fourier Transform) is outside the scope of this example. For a complete presentation of the FFT see the *Waveform Processing* chapter in this manual.

The FFT algorithm has two constraints

1. The x values of the input variable must be evenly spaced (from a linear sweep) and,

2. The number of points of the input trace must be even.

In order to simplify setting up the FFT the ”two tone” input signal should be made up of commensurate frequencies. If V_{in} is made of two signals, $\omega_1 = 100\ \text{MHz}$ and $\omega_2 = 105\ \text{MHz}$ then the greatest common denominator is 5 MHz. This means there is a frequency component at the output every 5 MHz, this relationship is used to set the FFT the time window. The time window must be an integral number of periods of the minimum resolution frequency which for this example is, $\frac{k}{5\ MHz}$. If *k*=2 then the FFT extracts frequency data every 2.5 MHz and the transient time window is 400 ns. This resolves the 5 MHz information plus some the extra points that help show the noise floor of the simulator.

Next the number of points in time window must be selected to reduce aliasing in the frequencies of interest and insure 2^n simulation points. To eliminate aliasing the sampling rate must be greater than 2 times the highest frequency of interest,

$$Sampling\ Rate = \frac{1}{Sampling\ Interval} \geq 2(Nyquist\ Frequency).$$

Choosing a Nyquist frequency of 1.25GHz the number of points is calculated as follows,

$$Time\ Step\ = \frac{1}{2\left(Nyquist\ frequency\right)} = \frac{1}{2.5\ GHz} = 0.4\ ns$$

$$Number\ of\ Points = \frac{Time\ Window}{Time\ Step} = \frac{400ns}{0.4\ ns} = 1000\ pts$$

rounding up to the nearest 2^n

$$n = 10 \quad 2^{10} = 1024.$$

By choosing n=10 for this example the sampling rate is 2.56 Gs/s which means that any frequency component above the Nyquist frequency of 1.28 GHz is reflected about the Nyquist frequency into the simulation results.

Another problem which must be accounted for in the FFT is leakage. By choosing a sample window which is integral number of periods of the commensurate input frequencies leakage should be eliminated. But if an FFT were performed on a sample larger than 1024 the result would also show aliasing effects. Looking at this example, 1024 points generates 1025 intervals this means it is necessary to remove one time interval in the extract statement

The level of the noise floor is set by the TekSpice integrator accuracy. This accuracy is controlled by the simulator parameters, reltol and chgtol. The more accurate the integration the better the noise floor. For this example reltol and chgtol have been set in the transient statement. For more information on the controlling the FFT accuracy see the *TekSpice Reference Manual*.

The test circuit used to simulate the IP_3 is the shown in Figure B–10, this circuit was used to simulate the noise figure in a earlier example. Because the IP_3 is a power measurement it is useful to define the transient sources in terms of input power rather than input voltage. In writing the input transient voltage sources in terms of RMS power and accounting for the divider effect of $R_{s,}$ the following is obtained:

```
testsignal=sint(0,sqrt(8m*Rs*10^(pavin/10)),100meg)
 +sint(0,sqrt(8m*Rs*10^(pavin/10)),105meg)
```

The transient value of our input source is defined as a variable, testsignal. This variable is then defined within the transient analysis block. This allows the same test circuit to be used in a number of different transient analyses without having to alter the test circuit.

Another consideration is initial start up effects, non–periodic transients can cause errors in the simulation results. To illustrate these effects a control program with four sample windows, each with the defined sample rates, was created. Start up effects sure be clearly seen by comparing the results for each window; 0–400ns, 400ns–800ns, 800ns–1200ns, and 1200ns–1600ns.

Compiling the above information generates the following ADS control program:

```
; Circuit Variables
Rs=50        ; Source Resistance
Rl=50        ; Load Resistance
dcin=0       ; DC level of input Source
pavin=-30    ; Power variable for transient input source
Vsource=1    ; AC value of input and output source
Vsph=0       ; Phase of input source

; FFT Analysis Variables
freqres=5meg ; The min freq to be resolved
k=2          ; The over sample rate
res=k/freqres; The time domain sampling window
n=10         ; The power to rise two to
pts=2^n      ; The number of pts in the fft
inc=res/pts  ; The necessary time step
tstop=4*(res); Trans analysis stop time
tran analysis time: 0,tstop,inc reltol=1e-7 chgtol=1e-16
limpts=10e6
testsignal=sint(0,sqrt(8m*Rs*10^(pavin/10)),100meg)
 +sint(0,sqrt(8m*Rs*10^(pavin/10)),105meg)
endtran

; Process the Simulation Resultsvout1=extract(v(out), 0,
400n-inc)
vout2=extract(v(out), 400n, 800n-inc)
vout3=extract(v(out), 800n, 1200n-inc)
vout4=extract(v(out), 1200n, 1600n-inc)
fftout1=db(fft(vout1,per))
fftout2=db(fft(vout2,per))
fftout3=db(fft(vout3,per))
fftout4=db(fft(vout4,per))

; Plotting commands
plot fftout1
plot fftout2
plot fftout3
plot fftout4;
```

The output plots resulting from this simulation is shown in Figure B–14.

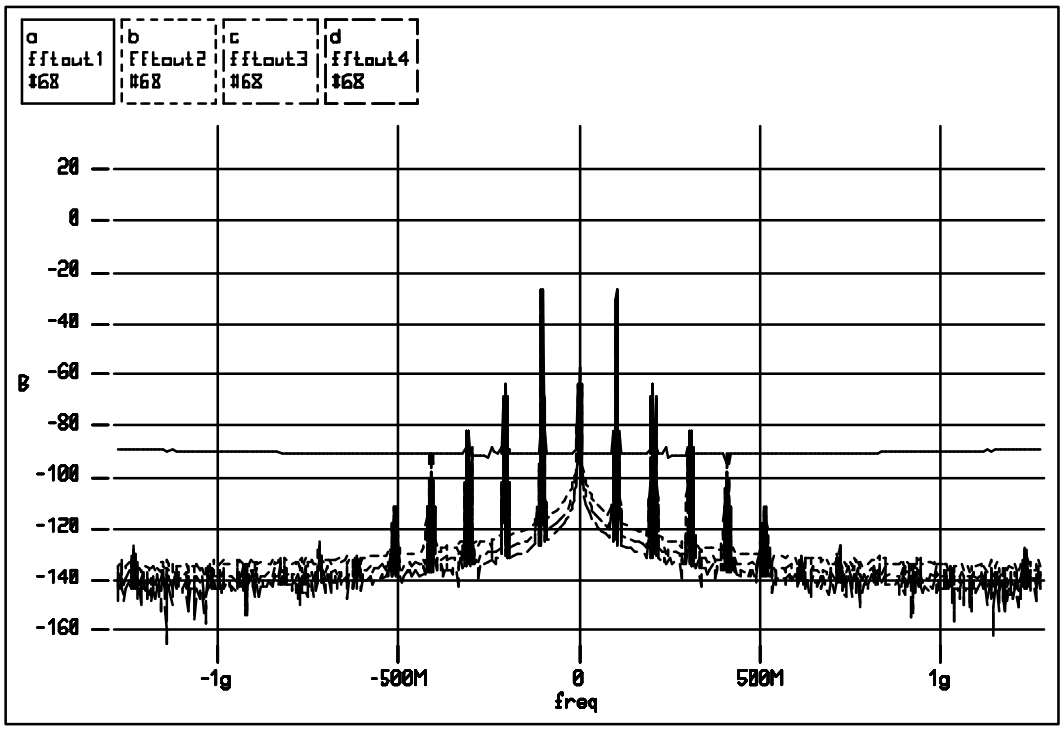


Figure B–14: Start–up Transient FFT Effects

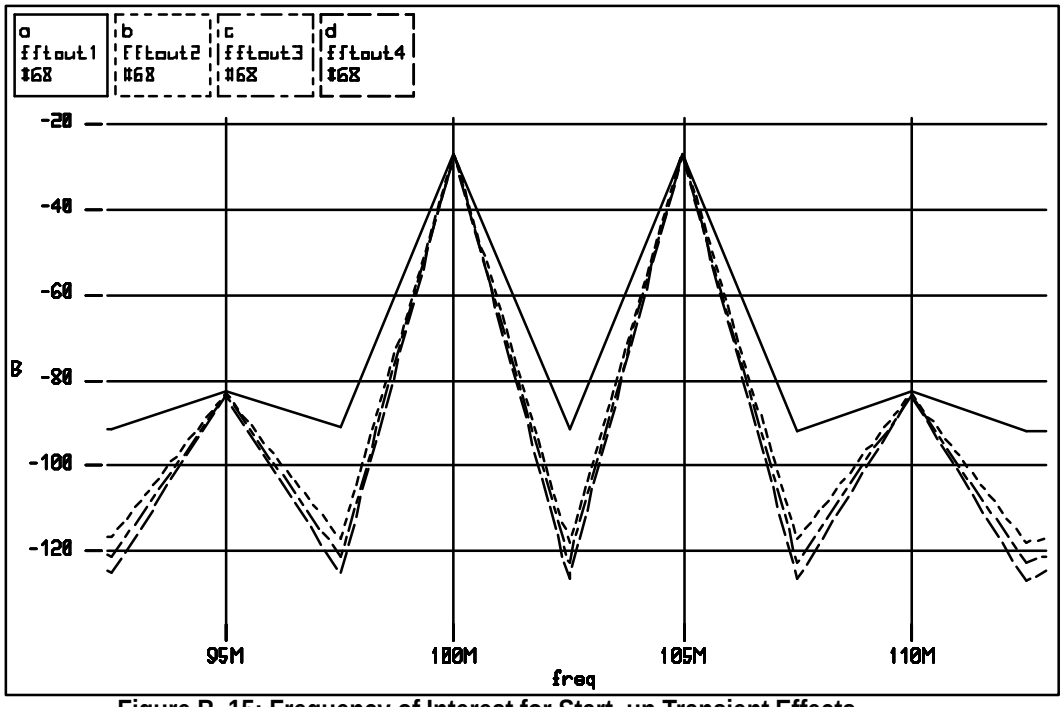


Figure B–15: Frequency of Interest for Start–up Transient Effects

The start up effects can be seen in the plot in Figure B–15, but is is clear that the peak values of the both the fundamental and the third order harmonics of interest are constant. This permits using these values for the IP_3 calculations with confidence that the start–up transient effects are inconsequential.

Of special note, because the FFT divides energies into positive and negative frequencies it is necessary to add 6 dB to **pout** to account for the mirror image of the output spectrum which lies along the negative frequency axis. Another 10 dB is added to convert the output results into units of dBm. The following ADS control script extracts the necessary information:

```
specout=extract(fftout1, 90Meg, 110Meg)
pout=specout+16
plot specout
```

The results are shown in Figure B–16.

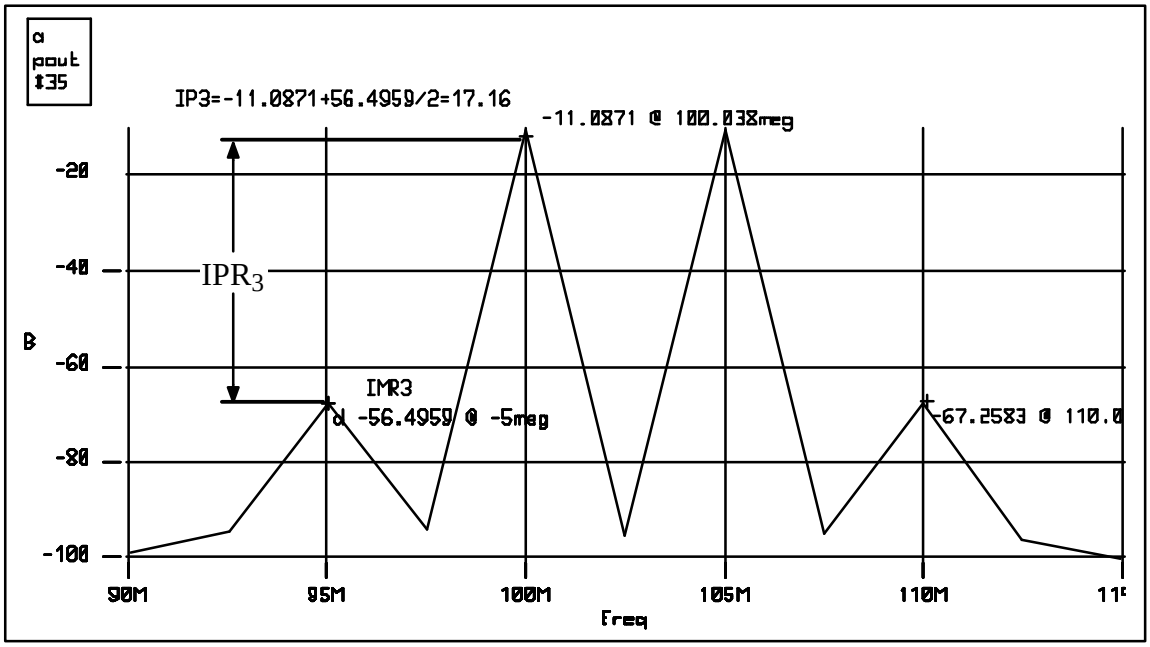


Figure B–16: Output FFT for Calculating IP_3 at 100 MHz

1 dB Compression Test

With an ideal RF amplifier, a linear relationship between the input power and output power is expected for a sinusoidal but with a real amplifier there is a limited dynamic range. As the amplifier reaches the upper end of its dynamic range the output begins to saturate causing the output to no longer increase linearly with the input power. The 1 dB compression point can be defined in

terms of input or output power, for this example it is defined as the input power for which the output power has dropped to 1 dB below that of the ideal amplifier. See Figure B–17. The decrease in the output power is caused by non–linearities of the components which makeup the LNA01A. Because the 1 dB point marks the transition of the amplifier for small signal and large single behavior it is necessary to perform a transient analysis for this simulation.

As with the IP_3 example it is convenient to define the input voltage source in terms of RMS power and the source resistance, R_s. In this example the 1dB compression point is found for an input signal frequency of 100 MHz. Since the transient value of the input source was defined as a variable in the previous example it is not necessary to modify the test circuit used for IP_3 simulation. It is only necessary to create a new transient analysis experiment by renaming the IP_3 experiment and modifying the transient analysis control program.

This simulation also requires the use of the FFT waveform function to extract the desired information at the test frequency. For this simulation a 100 ns window was used with 128 points in the time sample window. The following control program performs the necessary simulation and extracts 1dB compression data:

```
; Circuit Variables
Rs=50        ; Source Resistance
Rl=50        ; Load Resistance
dcin=0       ; DC level of input
Source pavin=-30; Power variable for transient input source

; FFT Analysis Variables
n=7          ; This give the number of pts
freqres=10meg; The min freq to be resolved
k=1          ; This define the freq resolution
tstart=5n    ; Trans analysis start time
pts=2^n      ; The number of pts in the fft
res=k/freqres; The rate to extract freq data
inc=res/pts  ; This calculates the time step
tstop=tstart+res ; This calculates the stop time

var sweep pavin : -40 -2 2
tran analysis time: 0 tstop inc reltol=1e-7 chgtol=1e-16
testsignal=sint(0,sqrt(8m*Rs*10^(pavin/10)),100meg)
endtran
endvar

; extract the necessary number of point for the FFT (128)
vout=extract(v(out),tstart, tstop-inc)

; extract the frequencies of interest for the FFT
specout=extract(fft(vout,per),90meg,110meg)
```

```
; extract the voltage at the fundamental frequency
vout_db=db(imag(peak(specout)))

; correction for FFT (+6) and conversion to power (+10)
pout=vout_db+16

; Extract the swept variable Pvin from the dummy source
vp=extract(output(Rsweep,v),tstart,tstop-inc)
time=40n
vp40=reduce(vp,time)

; Plotting Commands
plot pout vp40
```

A ”dummy” voltage source was included in the test circuit to aid in extracting the swept input power variable, pavin. Because the simulation is a transient analysis the results are a function of time but for the 1 dB test it is necessary to plot the ideal response in terms of power out versus power in. The dummy source along with some extract and reduce operations creates the plot Pavin vs pavin, by added constant equal to the gain to this variable the necessary ideal response line can be plotted.

The plot resulting from this control program is shown in Figure B–17.

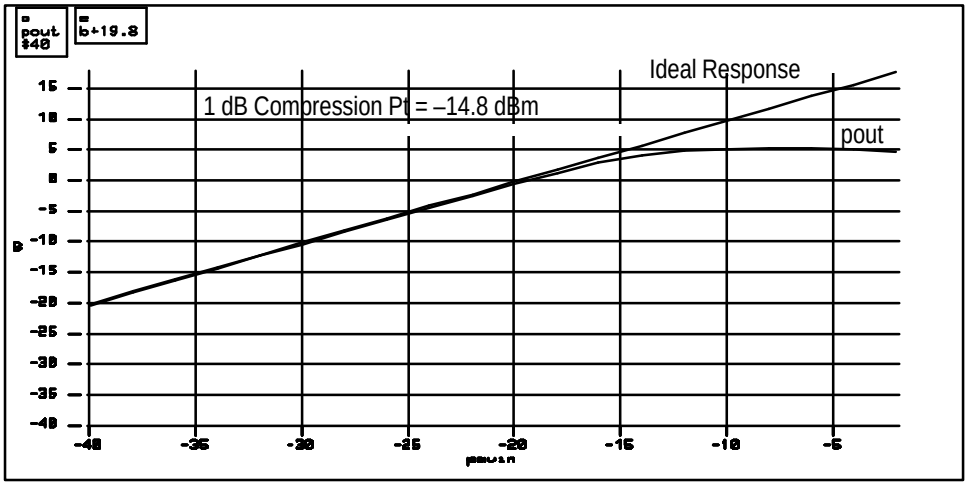


Figure B–17: 1 dB Compression Point

References

[1] E. Oran Brigham, *The Fast Fourier Transform*”, Prentice–Hall, Inc., Englewood Cliffs, New Jersey, 1974

[2] Ronald N. Bracewell, *The Fourier Transform and it Applications*, McGraw–Hill Book Company, New York, New York, 1978

[3] Paul R. Gray and Robert G. Meyer, *Analog Integrated Circuits*, 2nd ed., John Wiley and Sons, Inc., New York, 1984

[4] Stephen A. Maas, *Nonlinear Microwave Circuits*, Artech House, Inc., Norwood, Massachusetts, 1988

[5] David M. Pozar, *Microwave Engineering*, Addison–Wesley Publishing Company, Inc., Reading, Massachusetts, 1990

[6] Robert A. White, ”Distortion Measurements Using a Spectrum Analyzer,” *RF Design*, Vol.15, No. 9, September 1992, p.75

Appendix C: Discrete Example Circuits

This section looks at a high voltage video display driver. The driver uses discrete components and is simulated using ADS model libraries. All circuits used in this should be available from the **Circuit Browser** under the project *Examples_Discrete*. If not, then they are in EDIF format in the file"~cae/lib/ICO/UTILITIES/ExampleCircuits". The topics covered in this example are:

- Simulation of Discrete Components
- Transient Analysis
- Var Sweeps with Computed Component Values

Creating a Discrete Video Driver Circuit

The circuit schematic of the video driver is shown in Figure C–1. This circuit uses an Analog Devices AD811, high performance current–feedback op amp, to drive a pair of 2N3498 transistors in a discrete cascode output stage. Note that many of the circuit component values are defined as variables. The default values are set to zero and then defined at the start of the control program. This gives greater flexibility within the control program to modify the circuit parameters.

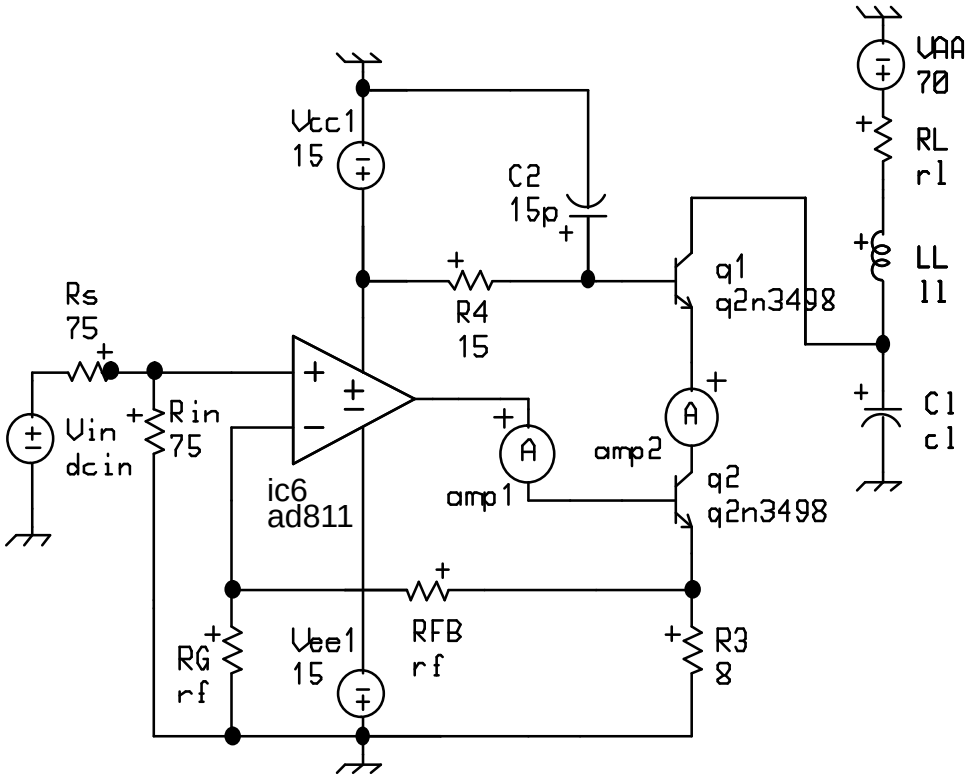


Figure C–1: Video Driver Circuit

Adding Discrete Device Models

First it is necessary to look at getting the discrete device models into the new circuit. This is done through the **Library Browser**.

1. Open the **Library Browser** from the **Launcher**, then open the DISCRETE library. This shows all the discrete libraries currently linked to your ADS image.

2. Select the ADI library in the center pane, in the right pane are the installed device subcircuits. These installed subcircuits are linked to symbols with the correct number of connections for the different devices in the library. When you select one of the installed models, the *Comment Text Pane* gives specific information about the symbol and its associated subcircuits.

3. In order to use the device model subcircuits in the ADI library, the subcircuit needs copied to a symbol. Select the five–terminal device, ad645, and then select the pop–up button **copy** command. The copy command prompts for a subcircuit name, enter ad811. Once the copy is completed the ad811 appears in the right pane.

4. Repeat this process in the ACOTECH library for the 2N3498 using a 2N3904 for the **copy**. When finished exit the **Library Browser**.

5. Now create the circuit in Figure C–1. The new device model subcircuits can be used in the same manor as the other devices used in earlier examples.

This circuit has two ammeters, amp1 and amp2, inserted into the the base and collector of the transistor q2. These meters are used to measure both the swept and transient currents of q2. Unlike the SHPi and GST–1 device models the discrete device models do not provide these values as a built–in function of the subcircuit. These ammeters are in the **UTILITIES>CommonSubcircuits** library.

Control Program and Source Definitions

The Vin source is set as follows:

```
tran=pwl(0,50m,30n,50m,30.1n,dcin,92n,dcin,92.1n,50m,120n,50m)
ac=1
dc=dcin
```

Note that the peak level of the PWL waveform is defined as dcin, this allows the user to adjust the level of the pulse from within the control program. The control program contains all the circuit variable assignments plus some useful result definitions. The complete control program is as follows:

```
; Circuit Variables
 dcin=2
rl=200; load resistances
rf=1k; op-amp feedback resistors
cl=6p; load resistance
ll=0n; series peaking element
var sweep cl : 2p 12p 1p
tran analysis time: 0 150n 150n/300
endtran
endvar
; Useful Definitions
vin=v(vin)
vout=v(vout)
rise=rise(vout)
fall=fall(vout)
pdq1=output(q1,vce)*output(amp2,i) ; power dissipation q1
pdq2=output(q2,vce)*output(amp2,i) ; power dissipation q2
; Plotting Commands
plot vout
plot rise fall
plot pdq1 pdq2
```

```
wiremenu @-@1 rise(@) rise(@-@1)
circuitmenu @-@1 rise(@)
```

Figure C–2, Figure C–3 and Figure C–4 shows the resulting plots from this control program.

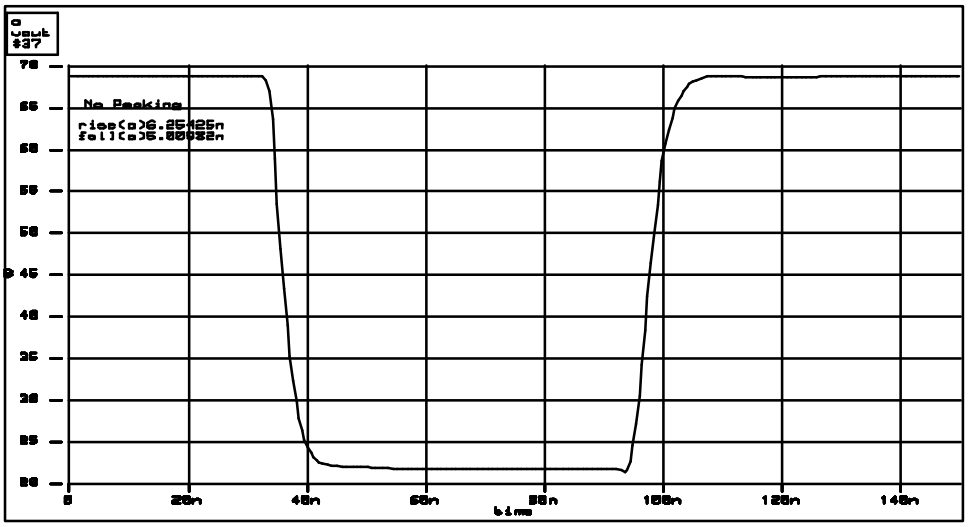


Figure C–2: V_{out} of Video Display Driver without Peaking

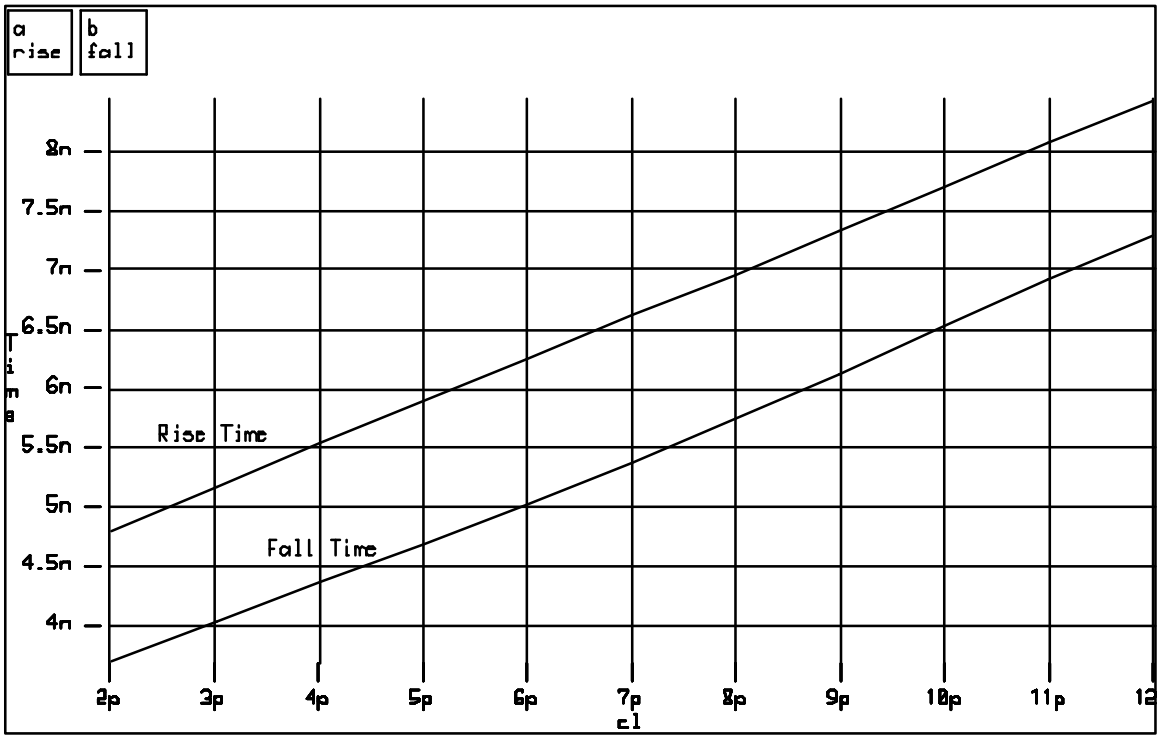


Figure C–3: Rise and Fall Times vs. Load Capacitance C_L

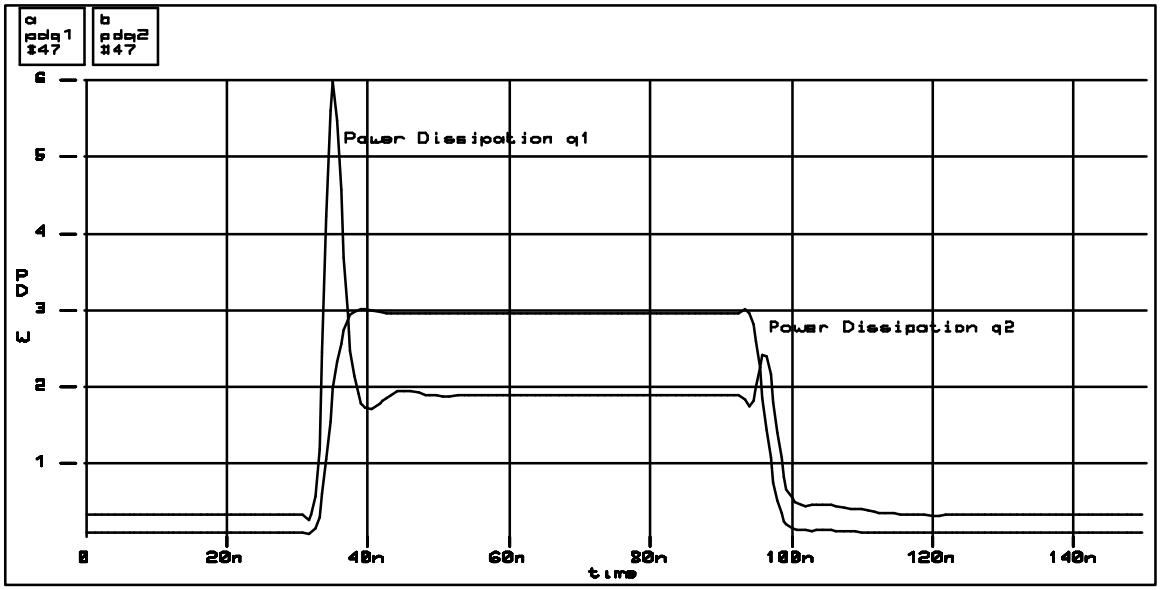


Figure C–4: Power Dissipation of Transistors q1 and q2

Calculated Component Values and Series Peaking

This section looks at ways of improving the performance of the video driver. Starting with simple series peaking, the relationship for a MFED (Maximum Flat Envelope Delay) response is given by:

$$L = \frac{R_L^2 C_L}{K} \quad \text{Where } K = 3$$

In this example the load capacitor is swept while at the same time changing the value of the series peaking inductor. By including the equation inside of the varsweep block, the series peaking inductor, L, changes as a function of the load capacitor, C_L. The modified control program is:

```
; Circuit Variables
 dcin=2
rl=200       ; load resistances
rf=1k        ; op-amp feedback resistors
cl=6p        ; load resistance
ll=0n        ; series peaking element
K=3.0        ; Damping Constant

var sweep cl : 2p 12p 1p
ll=(rl^2*cl)/K
tran analysis time: 0 150n 150n/300
endtran
endvar

; Useful Definitions
vin=v(vin)
vout=v(vout)
rise=rise(vout)
fall=fall(vout)

; Plotting Commands
plot vout
plot rise fall

wiremenu @-@1 rise(@) rise(@-@1)
circuitmenu @-@1 rise(@)
```

With the results from the example without a peaking element a plot is created which shows the improvement due to the peaking network. This plot is shown in Figure C–5.

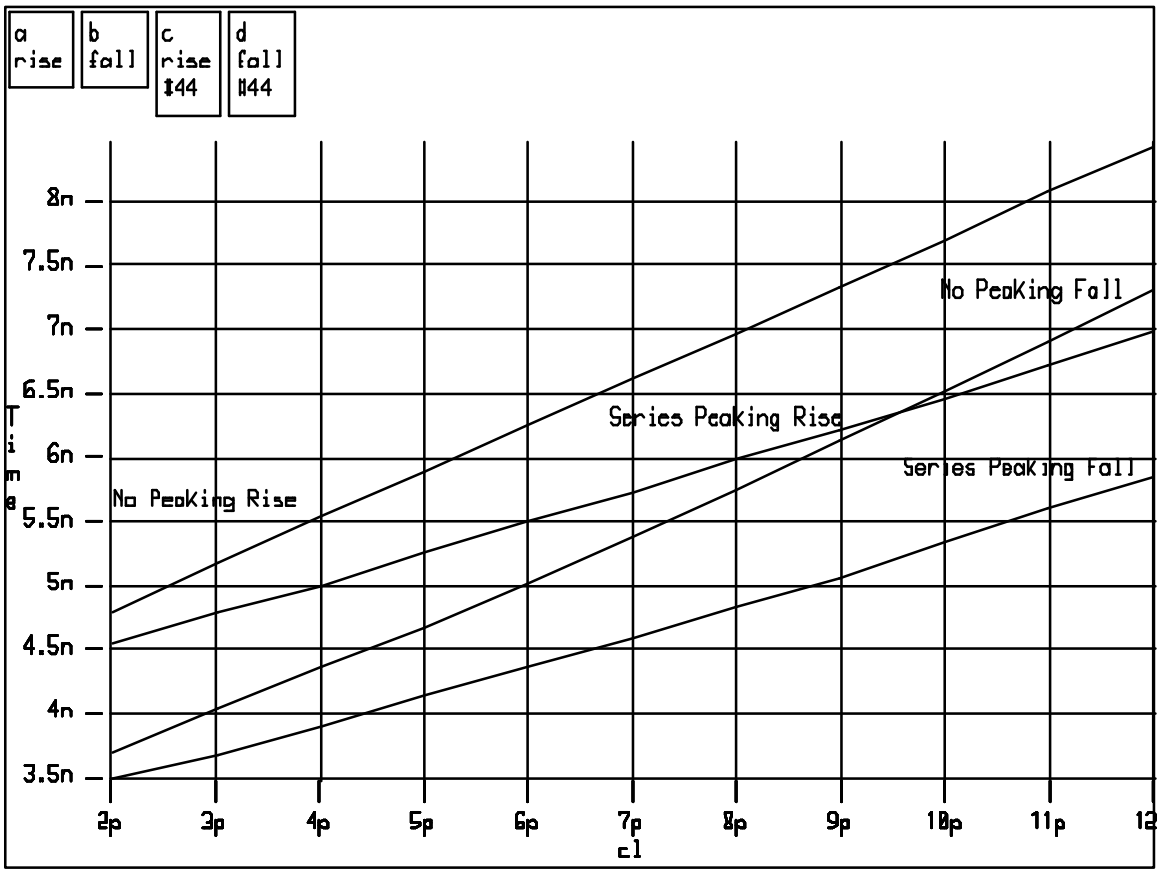


Figure C–5: Rise and Fall Times vs C_L

T–Coil Peaking

Having looked at series peaking and its simple relationship to R_L and C_L, it is clear that this simple analysis can be applied to a more complex peaking network, the T–Coil. The modified video display driver circuit is shown in Figure C–6.

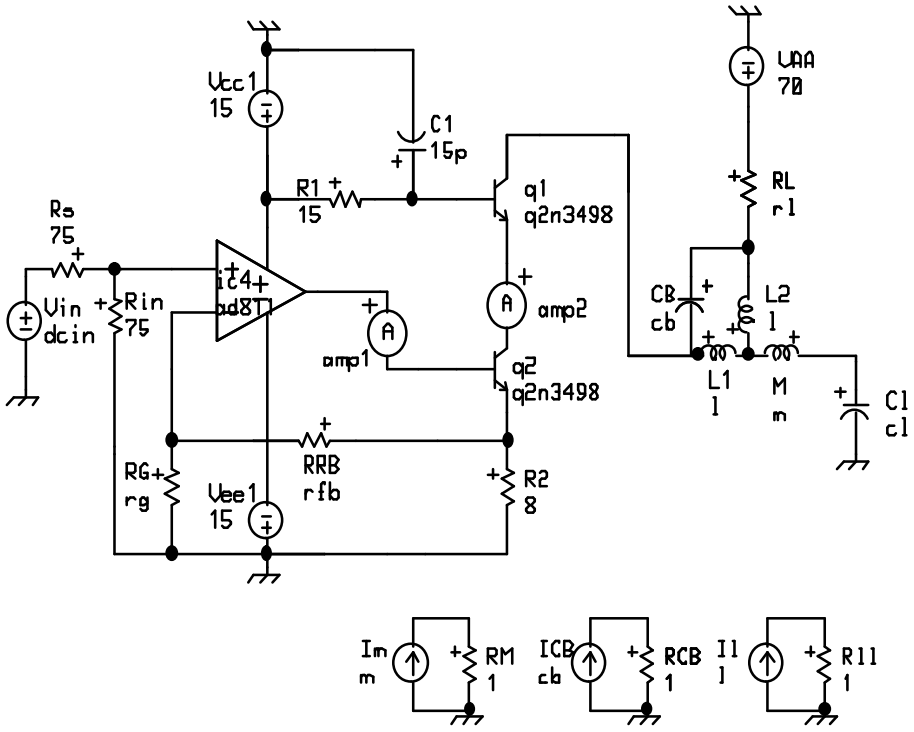
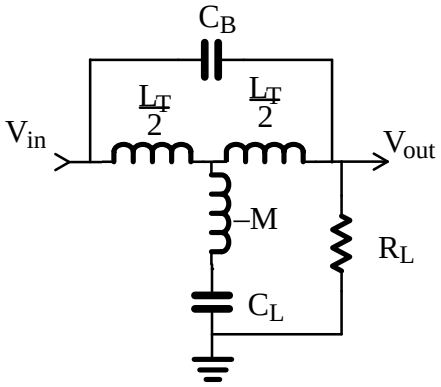


Figure C–6: Modified Video Driver with T–Coil Peaking Network

The T–coil network is made up of L_1,L_1, C_B and M. The inductor M is a mutual coupling element which has a negative inductance value for simulation purposes. The T–coil network could be simulated as a transformer made up by L_1, L_2, and M with a bypass capacitor C_B but the equivalent configuration shown allows greater flexibility for our analysis. The three current source and resistor circuits in the bottom right corner of Figure C–6 are used to plot the values of L_1, L_2, M, and C_B as C_L is swept. By setting the DC value for the current source equal to the circuit component value, the voltage across the 1 ohm resistor is the value of the component as a function of the swept parameter.

The equations for the T–Coil network are shown below in Figure C–7.



$$L_T = R_L^2 C_L$$

$$M = \frac{1 - K}{4} R^2 C_L$$

$$C_B = \frac{K}{4} C_L$$

K can range $1 \rightarrow \frac{1}{4}$

For the MFED case K = 3

Figure C–7: T–Coil Peaking Network

Translating the T–Coil circuit and equations to an ADS control program gives:

```
; Circuit Variables
dcin=2
rl=200       ; load resistor
rfb=1.0k     ; feedback resistor
rg=1.0k      ; gain resistor
cl=6p        ; load capacitor
K=0.33       ; coupling coefficient

var sweep cl : 2p 12p 1p
;var sweep K : 1/4 1 (1-1/4)/5
;var sweep Lt : 200n 400n 20n
Lt=(rl^2)*cl
l=Lt/2
m=-((1-K)/4)*Lt
cb=(K/4)*cl
tran analysis time: 0 150n 150n/300
endtran
endvar

; Useful Definitions
vout=v(vout)
rise=rise(vout)
fall=fall(vout)
under=unders(vout)
over=overs(vout)

; Plotting Commands
plot vout
plot fall rise
plot under over
```

```
wiremenu @-@1 rise(@) rise(@-@1)
circuitmenu @-@1 rise(@)
```

Sweep the total inductance L_t, to check if the ideal equations are correct for the simulation model. The design goal is the fastest rise and fall times with the least amount of under/over shoot. The results of this sweep are shown in Figure C–8 and Figure C–9. It can be seen that the predicted value of L_t = 240 nH is about right, with the undershoot increasing sharply after about 250 nH.

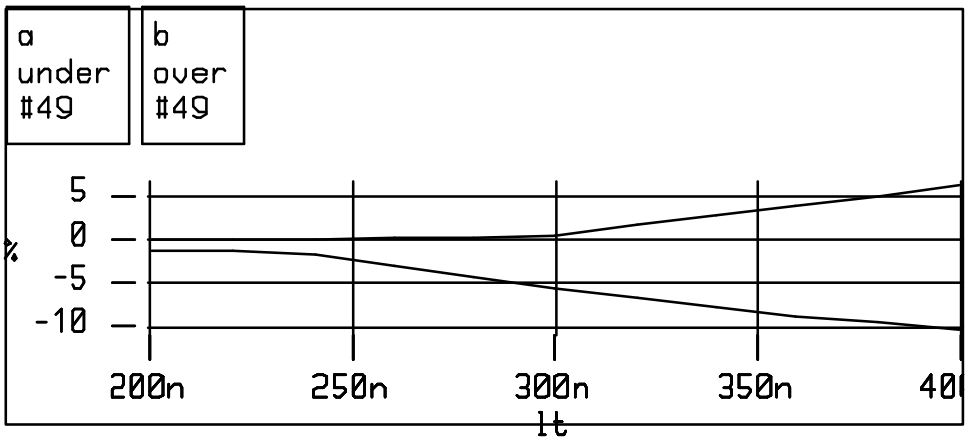


Figure C–8: Percent Over/Undershoot vs Total Inductance

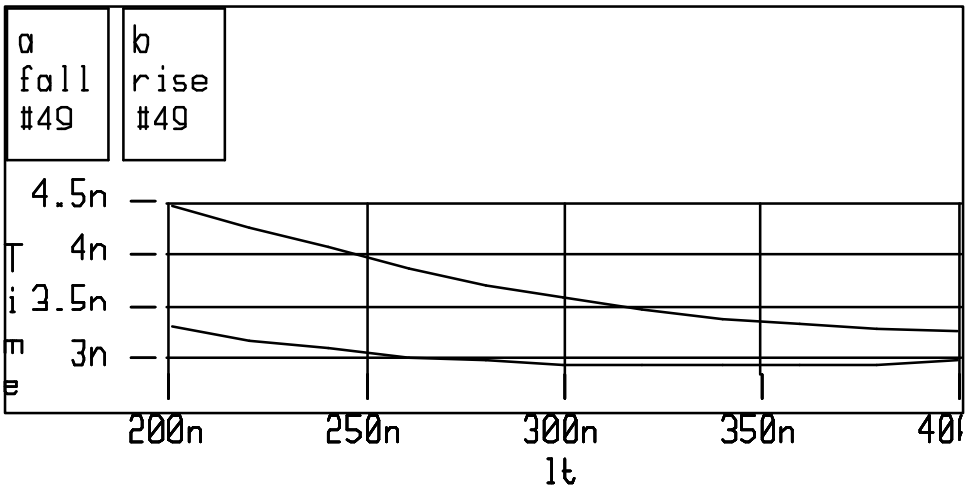


Figure C–9: Rise/Fall Time vs. Total Inductance

Now sweep the load capacitance and compare the rise/fall time performance to the other simulations. The results are shown in Figure C–10.

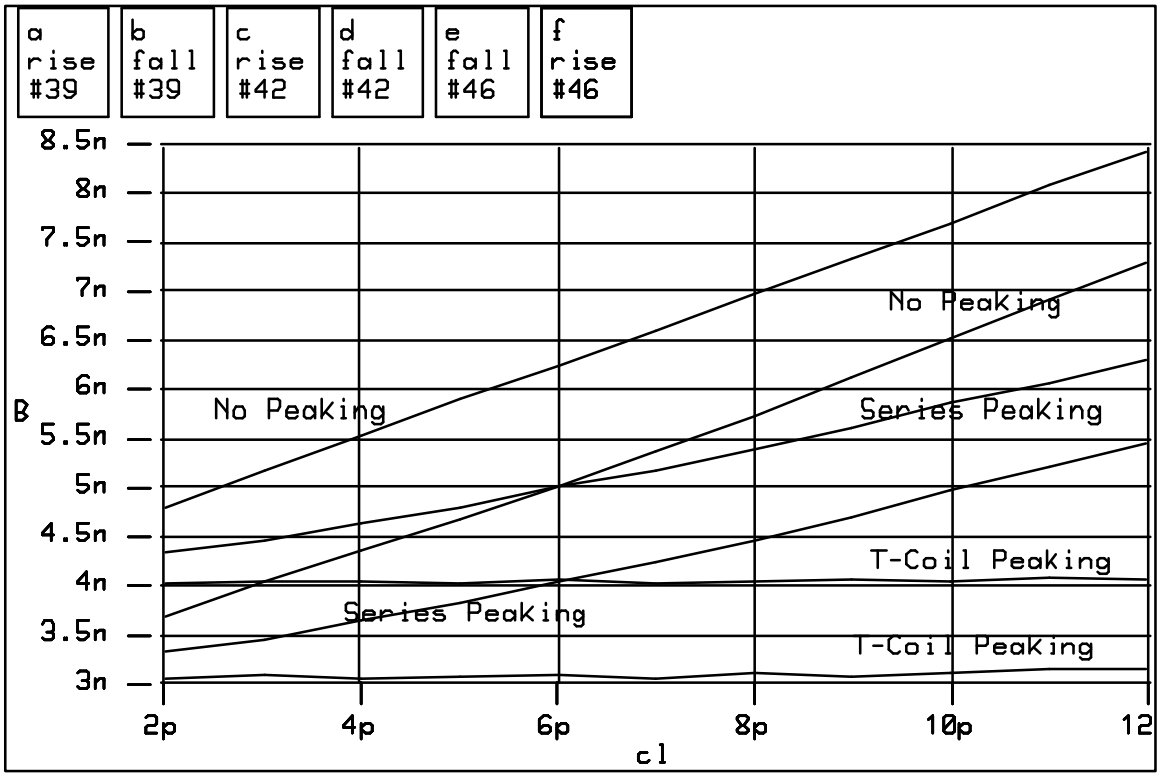


Figure C–10: Rise and Fall Time for Different Peaking Networks

Index

Glossary

Glossary

ansr file

A file containing node voltages and device operating point information resulting from a TekSpice simulation.

assignment

An assignment is performed when a variable name on the left–hand–side of an ”**=**” sign is assigned a value equal to an expression on the right–hand–side of the ”=” sign. For example, **rval=rval0+temp*0.01** is an assignment statement whereby the variable **rval** is assigned the value that results from evaluating the expression **rval0+temp*0.01**. For this assignment to work without error, the values of the variables **rval0** and **temp** must have been assigned prior to the evaluation.

The variables used in assignments are of two types: scalar variables and trace variables (of which either may be of type real or complex). Scalar variables are one (real or complex) value and may be thought of as a single point. Trace variables are sets of real or complex values, usually ”traces” created by a swept analysis; these traces may have one or more independent variables, each independent variable adds one to the dimension of the trace. The variable on the left hand side of an assignment statement assumes the dimensionality and type (either real or complex) that results from the evaluation of the right–hand side.

browser

A window for viewing and modifying information.

buffer

A buffer may be thought of as a temporary data–storage area. Data put into a buffer is stored until the buffer is overwritten with new data. Whenever text, simple graphics, or graphics that represents more information (such as circuitry from schematic diagrams) are manipulated with the **cut** or **copy** commands, the data is stored in a paste buffer. Once data is in the buffer it may be placed multiple times, each time with the **paste** command.

circuit

The term circuit when used within the context of an ADS circuit, usually means the hierarchical representation that is directly below the project level. Within each circuit may exist one or more devices and/or subcircuits. Each subcircuit may contain one or more devices and/or subcircuits. This may continue down through all desired levels of circuit hierarchy. Simulation is always done at the circuit level.

circuit variable

Equivalent to a variable that appears on the **circuit** statement in a TekSpice netlist.

Defined with the **variables > add circuit variables** command in the *Circuit List Pane* of the **Circuit Editor**. When an undefined variable is detected the user is asked to declare it as either a circuit variable, a local variable, or leave it as undeclared. If it is declared as a circuit variable, a default value is then specified. Circuit variables are global and may be used at any level of the circuit hierarchy.

click The term ”click” refers to the pressing and releasing of a button on the mouse.

control program A control program is a set of TekSpice control commands. The control program resides within the *Control Program Text Pane* in the **Control Program Browser**. A control program may consist of as little as a single analysis block or may consist of expressions, assignments, various pre– and post–processing commands, and multiple analysis blocks (multiple simulations).

device A model representing a TekSpice primitive model. Devices are available through the libraries: Devices_Passive, Devices_Semiconductor, and Devices_Source within the UTILITIES process.

dialog box A special purpose window used to input information needed to complete a command. The dialog box has text lines and push–button locations to make this input. In these cases, click on **quit** to abort the entry of information with no new input, and **ok** to signal that the data has been correctly entered and to exit from the dialog box.

EDIF Electronic Design Interchange Format. EDIF is used to store library and circuit data in a standard ASCII format. It can be used for design sharing, permanent archiving of data and transitions between major version changes in ADS.

element An instance of a device, subcircuit or model (See instance)

environment A binding of a set of values to a set of names.

expression A mathematical equation using +,–, * and / as well as the mathematical functions of ADS. Expressions are numerically evaluated and these values assigned to a variable, an input parameter or output parameter.

external library An ADS library which uses a binary (via the **genlib** command) model library to describe the models. The output parameters and symbols for these models are read into the ADS image via EDIF files.

external node A node connector representing a terminal of a subcircuit. A node name imposes a name on its net.

garbage collection An automatic operation that frees up system memory being occupied by objects that are no longer needed by the system. During this operation one of the three garbage collection cursors is visible. See the Cursor section of the *Reference* chapter.

global node A node connector that sets the potential of all nets where it is connected, whether in the top level circuit, or inside subcircuits at any level of hierarchy. Global node 0 (zero) representing circuit ground, is an example of a global node. A node name imposes a name on its net.

hierarchy A relational order of sets of the form:

$$A > B > C > D$$

where ”>” denotes that the set appearing to the left of the symbol contains that appearing to the right. An element c of C may be denoted A.B.C.c to indicate the property of the set inclusion appearing above.

image The complete state of an ADS session is called the ”ADS image” or just ”image”. An image can be stored as a binary disk file not running on the workstation.

independent variable An independent variable serves as an indexing variable(s) for traces or vectors. Plots of traces are exhibited versus an independent variable. Independent variable values are generated by swept analysis.

instance When a model is selected and pasted into a *Schematic Editor Pane*, the result is an instance of the model. The instance inherits the parameter values from the model definition. Those parameters can be changed on an instance by instance basis. Other modifications to the instance, such as to the model topology, affect the model definition, and hence other instances of the same model.

internal library A library whose models were generated within the ADS system. These models are composed of a symbol defined for the model, (for example a right–pointing triangle with two input lines and one output line to represent an op–amp) as well as the schematic diagram, input/output parameters and their default values.

library A collection of models accessible by circuits for building schematics for a TekSpice simulation.

local node A node connection which sets the potential for all nets connected to that node. It only connects nodes within the subcircuit or circuit where it is defined. A node name imposes a name on its net.

local variable A user–defined variable used for temporary storage of expression evaluation at the subcircuit level.

model There are three types of models in ADS: device models, user models, and subcircuit models. Device models are fundamental models available from the simulator without creating subcircuits. User models are customized device models created by the user. Subcircuit models are a combination of device models, user models and other subcircuits. They are used to create hierarchy within a circuit.

net The graphical or pictorial representation of an equi–potential set of wires and/or terminals.

node A symbol which can be placed into a schematic one or more times to create an equi–potential within the circuits. When a node is placed on a wire or terminal, the net name is set to the node name.

notifier A window that is automatically invoked whenever a non–resolvable program error condition is encountered. The appropriate response to this is to select the bug report option presented on the pop–up button menu.

pane A sub–region of a window. Panes most often display data and provide pop–up button menus from which data–related commands may be issued.

parameter A variable associated with a model. A parameter is one of two types, input parameter or output parameter. Input parameters are used by the simulator to produce outputs. Output parameters represent constructs from simulation results or small–signal values obtained from the mathematical evaluation of the device equations.

plotfile A binary file containing all node voltage and voltage source current traces and their indexing independent variables generated by the simulator. Operating point node voltages and source currents are also available as variables prefixed by ”bv” or ”bi”

preset variable A variable with a reserved ADS or TekSpice name, used for program and analysis control. An example is the reserved name ”pi” representing 3.14159...

primitive See Device

project A set of circuits and associated control programs and simulation results.

scalar A variable of one real dimension and no independent variable.

subcircuit The most general model type which can contain element models, user models and other subcircuits. It can be used to reduce circuit complexity by creating circuit hierarchy.

symbol A graphical representation of a device, element, model or subcircuit.

swept analysis A simulation performed over a set of values for time or another independent variable.

terminal A node of a subcircuit, user model or device, which may be connected to other circuit elements, excluding global nodes. Terminals must appear on the graphical schematic symbol.

trace See waveform.

trace variable A set of values indexed by an independent variable, generated through simulation.

user model A label, graphic symbol, set of parameter values and output definitions that supplant these blocks for a TekSpice primitive model.

variable A named value or array of values that may be changed by assignment or simulation.

waveform The viewable display of a trace variable from TekSpice.

window A frameable and movable region of the screen in which ADS panes and menus are available.

wire A line segment, or collection of contiguous line segments, in a circuit schematic that join one or more elements together, such as resistors and transistors.

workspace An environment where expressions can be executed.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z