Agilent 85190A
IC-CAP 2008

User's Guide

Agilent Technologies

Notices

Edition

March 2008

Printed in USA

Agilent Technologies, Inc.
5301 Stevens Creek Blvd.
Santa Clara, CA 95052 USA

Acknowledgments

UNIX ® is a registered trademark of the Open Group.

Windows ®, MS Windows ® and Windows NT ® are U.S. registered trademarks of Microsoft Corporation.

Errata

The IC-CAP product may contain references to “HP” or “HPEESOF” such as in file names and directory names. The business entity formerly known as “HP EEsof” is now part of Agilent Technologies and is known as “Agilent EEsof.” To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain “HP” or “HPEESOF” references.

Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Contents

3 An Example IC-CAP Session

5 Making Measurements

12 Creating Graphic User Interfaces

D Macros

E Transforms

F Miscellaneous

Index

1
Overview

The IC-CAP Modeling System is part of Agilent EEsof EDA's high-frequency electronic design automation (EDA) solutions. Agilent EEsof EDA offers a full array of design tools that streamline and strengthen the engineering process. These tools include powerful system and circuit simulators, 2.5-D and 3-D electromagnetic simulators, device modeling systems, and physical design tools.

The IC-CAP *User's Guide* covers basic terms and operating procedures of the IC-CAP device modeling system.

The IC-CAP Modeling System

The IC-CAP Modeling System is used to measure semiconductor device and circuit modeling characteristics and analyze the resulting data.

To use IC-CAP, you need:

- A workstation
- Instruments that perform DC, Capacitance, AC, and Time-Domain measurements
- A test fixture
- A test device
- The IC-CAP software

The following figure shows the IC-CAP global configuration.

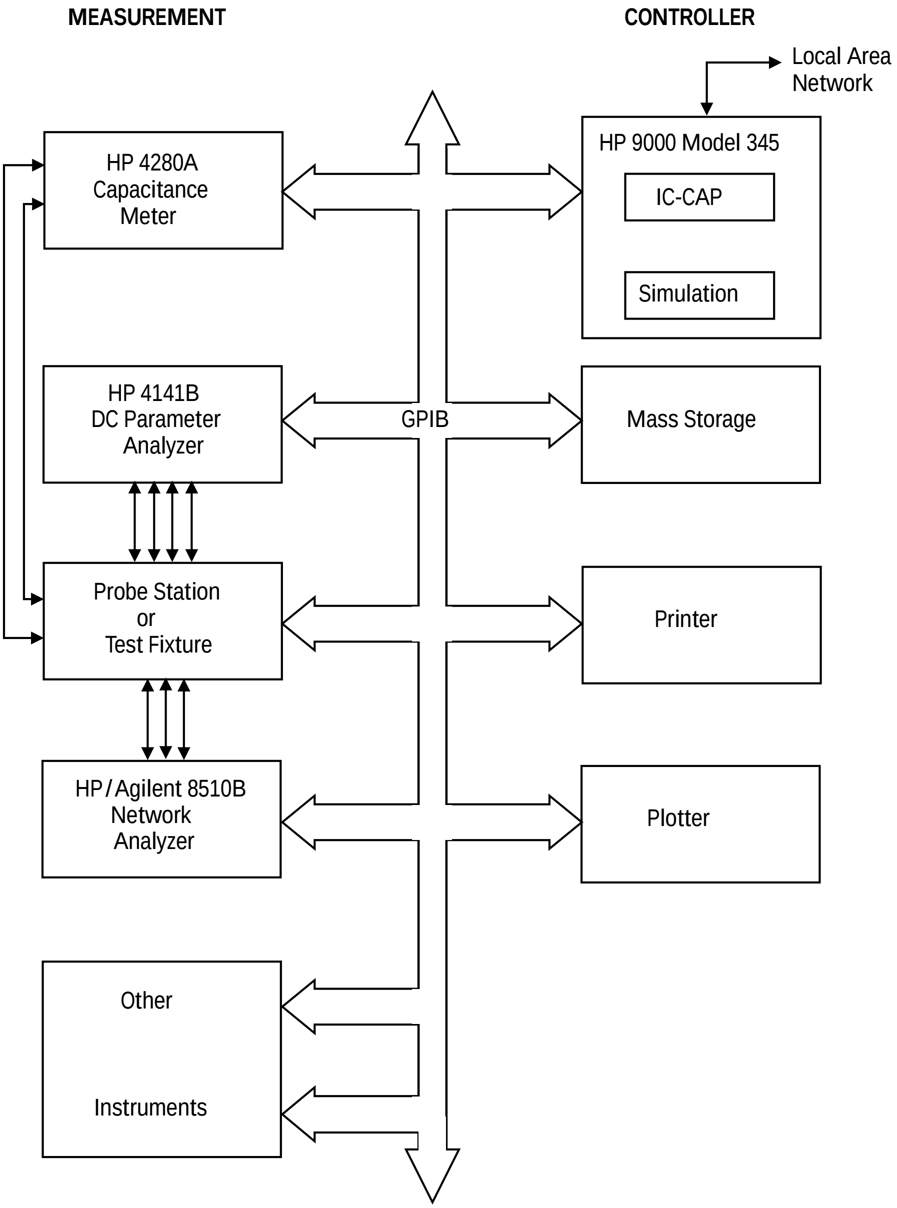

MEASUREMENT
CONTROLLER
Local Area Network
HP 4280A Capacitance Meter
HP 9000 Model 345
IC-CAP
Simulation
HP 4141B DC Parameter Analyzer
GPIB
Mass Storage
Probe Station or Test Fixture
Printer
HP / Agilent 8510B Network Analyzer
Plotter
Other Instruments

Figure 1 IC-CAP Global Configuration

The IC-CAP System Architecture

The IC-CAP system centers around a data storage area containing device and circuit characteristics. Initially, these characteristics are obtained from measurements or simulations. After gathering your data, you can perform a variety of operations on the data, including mathematical transformation, extraction, optimization, analysis, or archival to a file or a data base. The analysis features available in IC-CAP allow detailed study of the characteristics and the data can be saved for future use.

Mathematical transformations are used to extend the available set of characteristics, while extractions and optimizations provide feedback to the model in order to obtain agreement between measured and simulated data.

The following figure illustrates the IC-CAP system architecture.

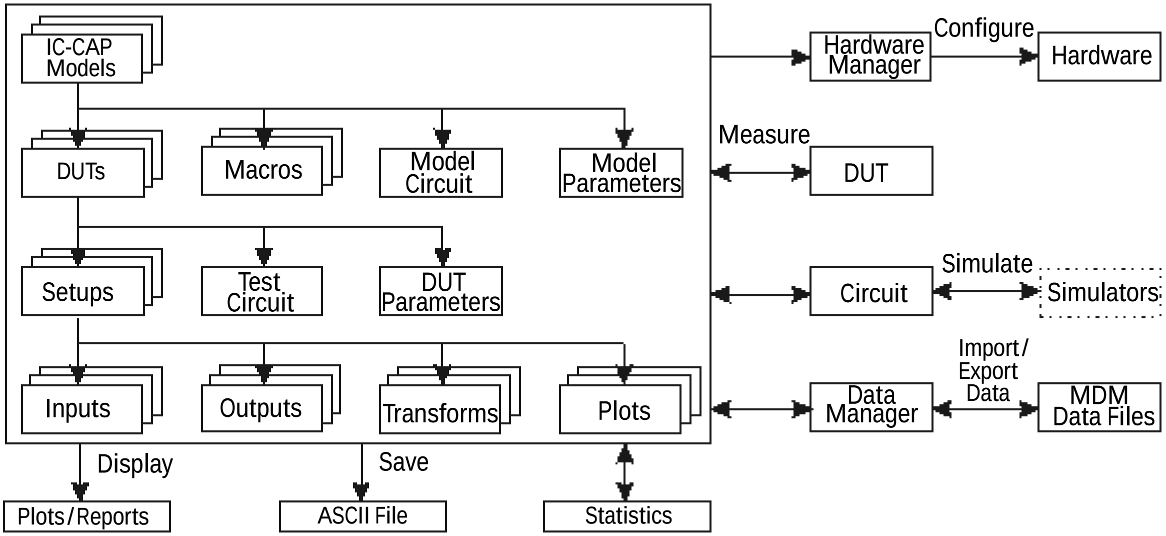


Figure 2 IC-CAP System Architecture

System Functional Areas

The four main functional areas of the IC-CAP system are:

- Modeling
- Hardware Management
- Function List
- System Utilities and Operations

The following figure shows the general organization of the IC-CAP system.

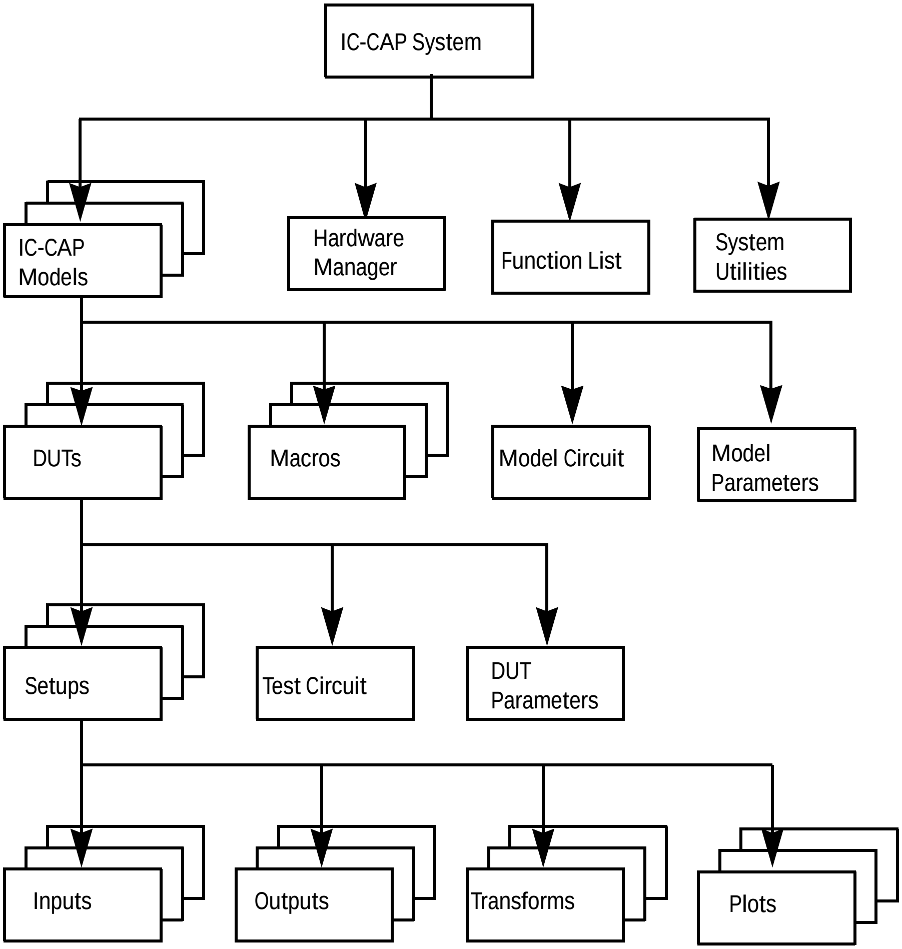


Figure 3 General Organization of the IC-CAP System

Modeling is the key functional area. You use this function to:

- Test devices and circuits using predefined models
- Modify existing models by editing
- Create new models by describing their physical and electrical characteristics
- Create macros to automate measurement and extraction processes

- Create the DUTs contained in a model by describing their physical and electrical characteristics
- Define test circuits and modify parameters for individual DUTs
- Create Setups for a particular DUT by defining:
    - The connection of a device or circuit to the test instruments
    - The inputs and outputs of the device or circuit
    - The transformations of the outputs used to calculate parameter values or additional data
    - The characteristics of the plots used to display the results of your tests

The hardware management function is used to create and modify the GPIB configuration of the test instruments. The function list is used to view the available characterization functions. These functions include mathematical transformations, model parameter extractions, and special analysis functions, such as optimization. System utilities are used to create and modify the global configuration. Operations include simulator selection and directory specification.

General Operating Procedures

IC-CAP provides a complete set of procedures for characterizing devices and circuits. Each of these procedures can be executed from easily accessed menus or programmed into IC-CAP macros. The system can be used for routine operations with a minimum of training. The general procedure for characterizing a device consists of the steps described here and shown in the following figure.

Installation Install the device in a test fixture. The instruments must be connected to the computer and IC-CAP via the GPIB bus.

Load an Existing Model or Create a New Model Load an existing model from the Model List after starting the program or create a new model by modifying an existing model or by starting with a blank template.

Measure or Create Device Characteristics Execute IC-CAP's *Measure* command to measure or create device characteristics. The program takes control of the measuring instruments and executes their functions. The instruments connected to the test fixture generate source signals at the input nodes of the test device. Then the response signals are recorded by the instruments connected to the output nodes of the device. The responses are recorded by the instruments and the measured data are loaded into the IC-CAP database.

Extract Model Parameter Data from the Measured Data Execute IC-CAP's *Extract* command to calculate the parameters that control the electrical behavior of the device from the measured output data.

Simulate Send the circuit definition and the extracted parameters to the simulator. The simulator generates a simulated data set for comparison with the measured data set.

Optimize Optimize the parameters to achieve the best possible fit between measured and simulated data.

Results Display or print results in graphic or tabular reports.

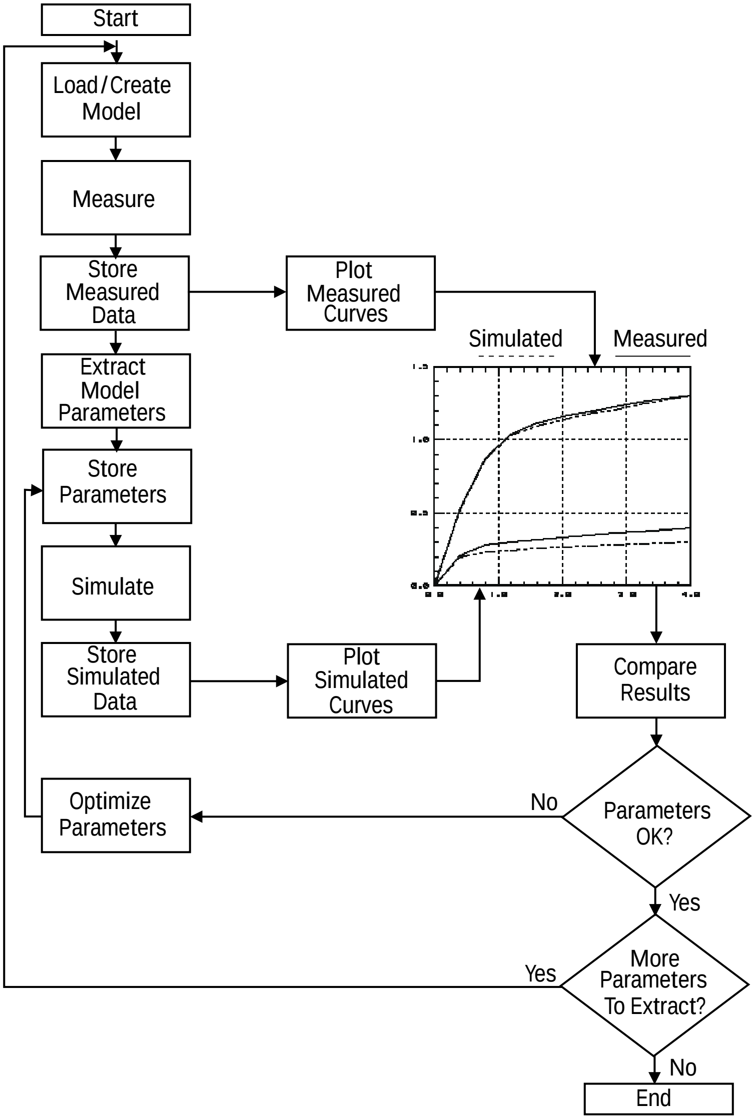


Figure 4 General Functionality of the IC-CAP System

IC-CAP Terminology

This section lists the terms used throughout the documentation to describe the basic parts of the IC-CAP system.

Model Complete instructions for characterizing a device, including a description of the device, measured and simulated data, and functions that are performed on the data.

Model Parameters Parameters that govern the behavior of the device, and whose values are common to all devices being measured for a particular characterization.

DUT (Device Under Test) A particular configuration of the device being measured and the hardware being used to perform the measurements.

Test Circuit Additional circuitry that is connected between the device being measured and the hardware being used on a particular DUT.

DUT Parameters Parameters that govern the behavior of the device, and whose values may be different for each DUT.

Circuit A description of the device, in SPICE deck format.

Setup Instructions for performing a measurement or simulation on a DUT. A setup contains definitions of stimuli, responses, mathematical transformations, extractions, and graphical or tabular reports.

Measurement The application of a set of stimuli to an actual device to obtain a set of responses that are used in the characterization process.

Input The definition of a stimulus in a measurement or simulation. Input includes the array of data that results from this definition.

Output The definition of a response in a measurement or simulation. Output includes the arrays of data that result from performing those operations.

Extraction The calculation of the parameters that control the device behavior, typically from measured data, but sometimes from simulated data or data sheet values.

Transform The definition of a function that is performed on IC-CAP data to create a new set of data or calculate parameter values. Transform includes the arrays of data that result from performing the function.

Simulation The calculation of a set of data from a mathematical model of the device.

Optimization The iterative adjustment of parameter values in order to achieve the best possible agreement between measured and simulated data.

Plot Instructions for generating and displaying IC-CAP data, either in graphical or tabular format.

Macro A group of IC-CAP functions combined into a single operation.

2
Program Basics

This chapter covers the basic concepts and skills you need in order to work successfully with the program.

To install the program, follow the instructions in the *Installation and Customization Guide*.

Documentation Conventions

The IC-CAP documentation uses consistent visual cues, standard text formats, and special terminology so that you can locate and interpret information easily. The following table lists these documentation conventions.

Table 1 Documentation Conventions

Type style	Used for
Italic	New terms, directory names, filenames.
ALL CAPITALS	Acronyms.
Initial Capitals	Names of keys on your keyboard (such as, Esc, Back Space, or Return), menu items, command names, dialog names and options.
Bold	Menu names, command names, items from a list, filenames or project names that you select when following a procedure.

Making Selections with the Mouse

The mouse is used to make selections and to open and close program windows.

- To select an item, move the mouse pointer to the item, then click (press and release) the left mouse button.
- To select a command from a menu:
    - Move the mouse pointer to the command and double-click (two clicks in rapid succession) the left mouse button.
    - Move the mouse pointer to the command and press, but do not release, the left mouse button. Drag (slide) the pointer to the command and then release.
- To cancel a menu, click anywhere outside the menu. (Or, you can press F10 or Esc.)
- To display the x- and y-axis values of a particular point on a data trace, move the mouse pointer to the point and click the middle mouse button.
- To display a list of available choices for a field, move the mouse pointer over the list pointer at the right end of the field, then click the left mouse button.

Working with Windows

A window contains an application, such as a model, or an information dialog box. The following figure shows the parts of the IC-CAP/Main window.

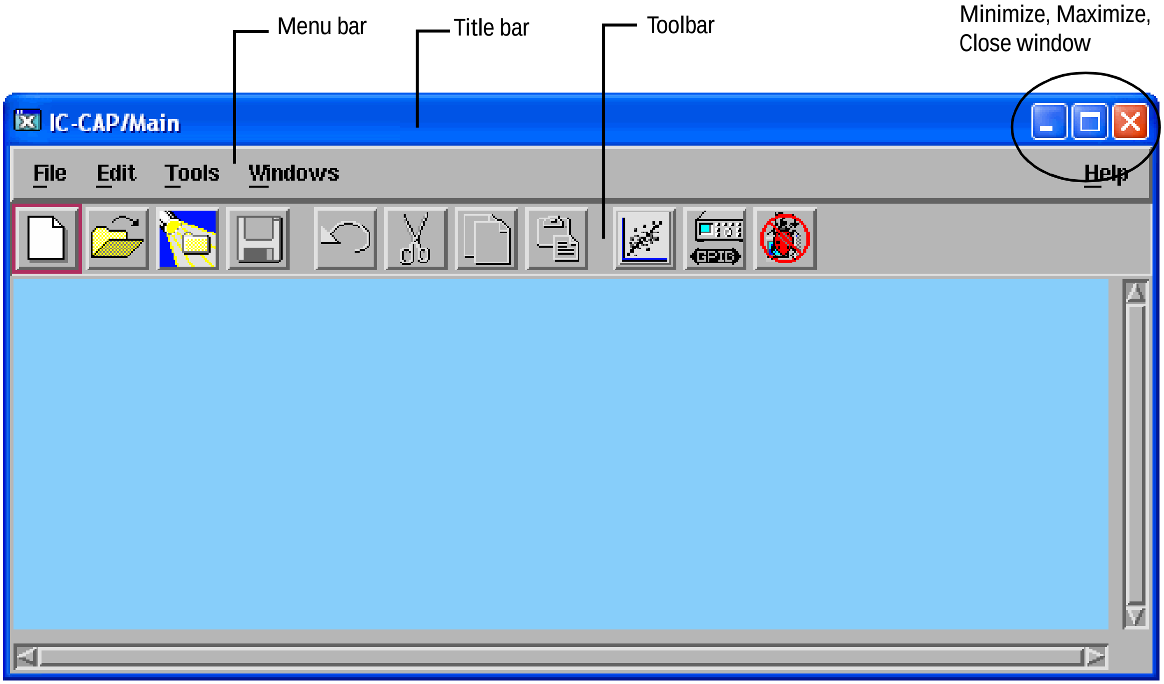


Figure 5 An Example IC-CAP Window

You can move, resize, and arrange windows for best usage of the screen. For details on window management, refer to your operating system documentation.

Working with Menus and Commands

You carry out an action by choosing a command. Program commands are listed on menus on the window menu bar. The following table lists menu conventions.

Table 2 Menu Conventions

Menu Convention	Description
Dimmed command name	The command is not available now.
Ellipsis points (...) displayed after a command name	Opens a dialog box for completing information needed to carry out the command.
A triangle (>) at the right side of the menu command	Opens a sub menu for the command.

NOTE

For information on specific commands, refer to Appendix A, “Menu Descriptions” in online help.

Using the Tearoff Menu Pull-down menus can be separated from the menu bar and placed anywhere on your desktop. To tear off a menu, click the dashed line at the top of the menu and drag the menu to a new desktop location.

Using the Toolbar Frequently-used commands are available as buttons on the toolbar in each design window. When you move the mouse pointer over a button, a balloon displays a label identifying the function of that button. The following figure shows the toolbar in the IC-CAP/Main window.

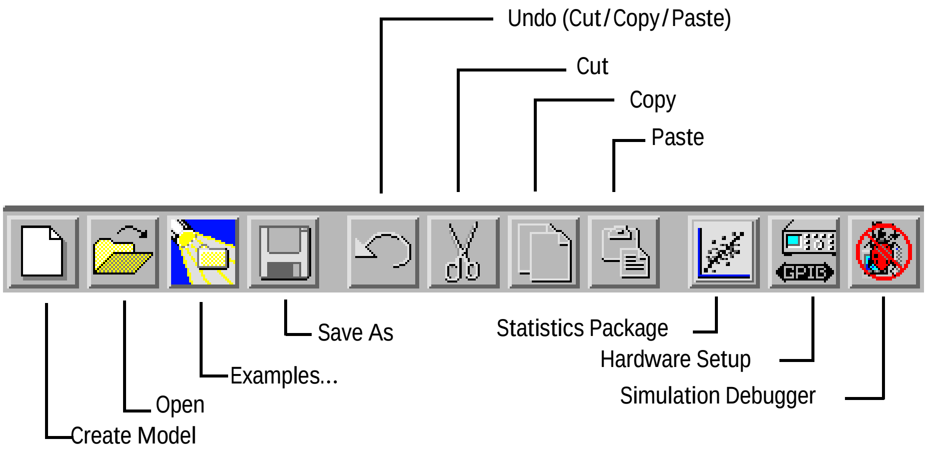
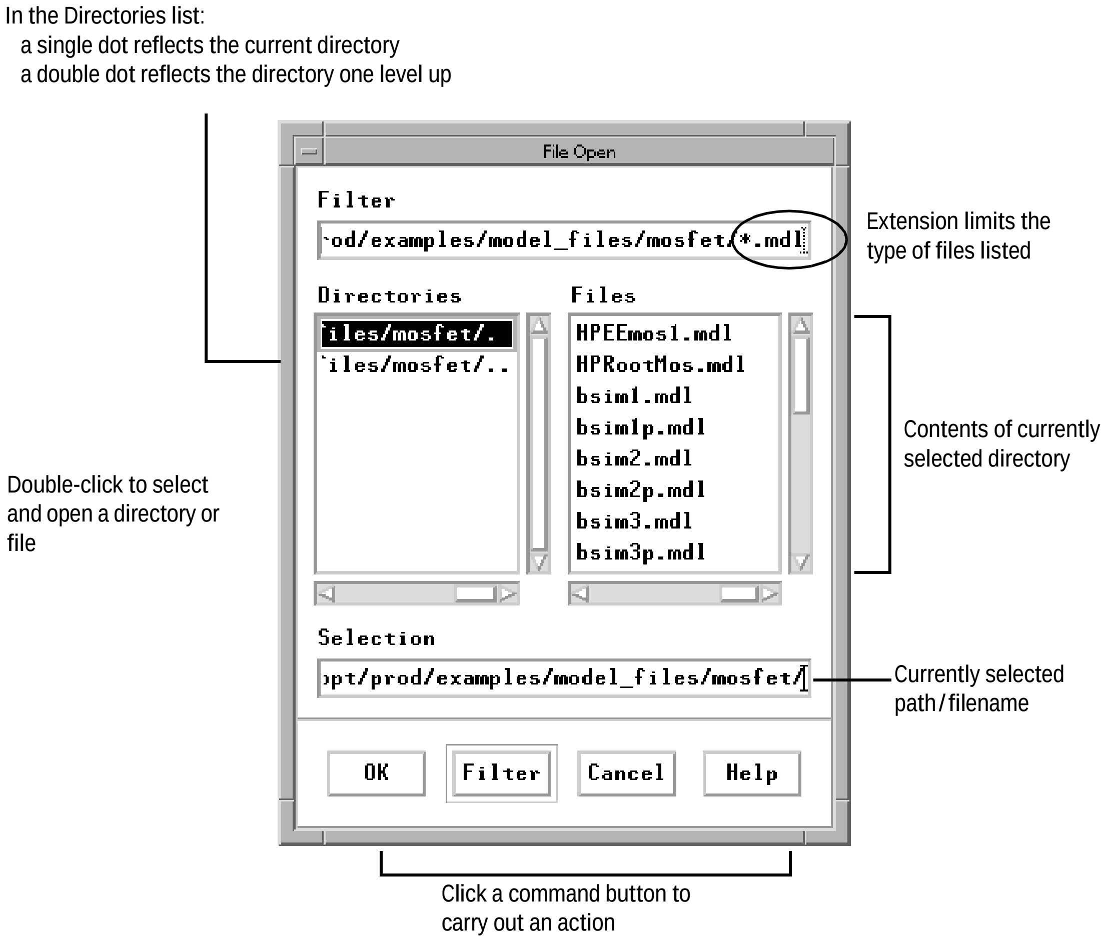


Figure 6 IC-CAP/Main Window Toolbar

Working with Dialog Boxes

The program uses a dialog box to display messages or to request information needed to carry out a command. The following figure shows an example dialog box.



In the Directories list:
a single dot reflects the current directory
a double dot reflects the directory one level up

Filter

·od/examples/model_files/mosfet/*.mdl

Directories

iles/mosfet/.
`iles/mosfet/..

Files

HPEEmos1.mdl
HPRootMos.mdl
bsim1.mdl
bsim1p.mdl
bsim2.mdl
bsim2p.mdl
bsim3.mdl
bsim3p.mdl

Selection

›pt/prod/examples/model_files/mosfet/

Extension limits the type of files listed

Contents of currently selected directory

Double-click to select and open a directory or file

Currently selected path/filename

Click a command button to carry out an action

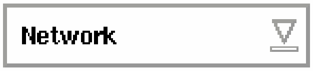
Figure 7 An Example Dialog Box

Providing Dialog Box Information

List Box Contains the available choices. If there are more choices than can fit in the list box, you can use the scroll bars to access more choices. Click the item you want to select.

Text Box Enter the required information in the fields or edit information currently entered. One field has the keyboard focus (where keystrokes appear). This field is identified by a highlighted border and a blinking input cursor that looks like an I-beam.

Drop-down List Opens after you click the triangular symbol at the end of a bar-like button. Click the item you want to select.



Click triangular symbol to view available choices

Command and Option Buttons Initiate immediate action when selected. By default, the choice that has a darker border is active. To choose the default, press **Enter** or click **OK**.

To implement the selections made in a dialog box:

- To make changes in a dialog box, provide the information required, then choose **OK** to complete the command and close the dialog box.
- To make changes, but keep the dialog box open, choose **Apply**.
- To close a dialog box without completing the command, choose **Cancel**.

Prompts A prompt dialog box requires information to carry out the current command. You must make a choice and close the dialog box before you can take any other action in the program.

Error Messages An error dialog box opens after an operation in which one or more errors have occurred. All errors resulting from a particular operation are displayed in one box. You must clear the error dialog box before you can continue with IC-CAP operations. All error messages generated during a particular session are written to a file in your home directory named *.icerrlog*. The .icerrlog file is cleared each time IC-CAP is run.

Table and Text Editors

A table editor is used to display and edit groups of information.

The *keyboard focus* field is identified by a highlighted border and a blinking input cursor. Normally the input cursor looks like an I-beam. To change the location of the keyboard focus or reposition the input cursor within a field, click in the field.

The number of visible rows in a list or table is specified by system variables. Changing a window size does not affect a list or table size. Refer to the section on *System Variables* for details on changing variables.

The following table lists keystrokes that you can use while editing fields in a table or text editor. Some keystrokes may not be available on all keyboards. Table 4 lists the mouse operations used to select and copy text. All selections described are primary selections unless noted otherwise.

Table 3 Table and Text Editor Keystrokes

Keystroke	Action
Right Arrow	Move forward one character
Left Arrow	Move back one character
Down Arrow	Move to next line or field
Up Arrow	Move to previous line or field
Home	Move to beginning of line
End	Move to end of line
Backspace	Delete previous character
Shift Backspace	Delete previous word
Delete	Delete selected text
Shift Delete Char	Delete next word
Tab	Move to next field
Return	Begin a new line

Using the Pop-up Menu as a Shortcut A pop-up menu, available in table and text editors, enables access to many common commands with a minimum of mouse movement.

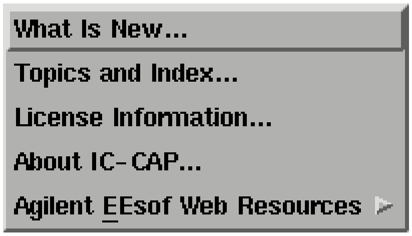
Table 4 IC-CAP Editor Select and Copy Mouse Operations

Operation	Action
Click left button	Move insertion and destination cursors
Double click left button	Select word
Triple click left button	Select line
Quad click left button	Select entire editor
Drag left button	Select to end of drag
Shift click left button	Extend selection
Shift drag left button	Extend selection to end of drag
Click middle button	Copy selection to new location
Drag middle button	Select (secondary) to end of drag and copy to destination cursor

Operations for copying and pasting text to and from terminal windows are platform-dependent. For details, refer to your workstation's documentation.

Using Help

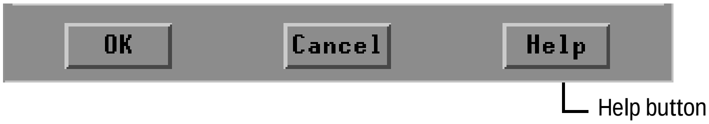
Help messages are available for each IC-CAP window. Window level help is displayed when you click *Help* in the top menu bar of a specific window.



From the Help menu, you can access this information:

What Is New...	Displays important differences between the PC and UNIX version of IC-CAP.
Topics and Index...	Displays the main contents for the online manual set. Clicking a listed chapter opens the file for that chapter.
Agilent EEsof Web Resources	Displays telephone, fax, and e-mail numbers for accessing Agilent EEsof Worldwide Customer Support.
License Information	Displays the environment/license variables and machine information, all licenses found in the license.lic file, all license servers serving Agilent EEsof licenses on your network, and the current status of all installed licenses.
About IC-CAP...	Displays version and copyright information.

You can display a Help message for a specific dialog box by clicking the Help button in the dialog box.



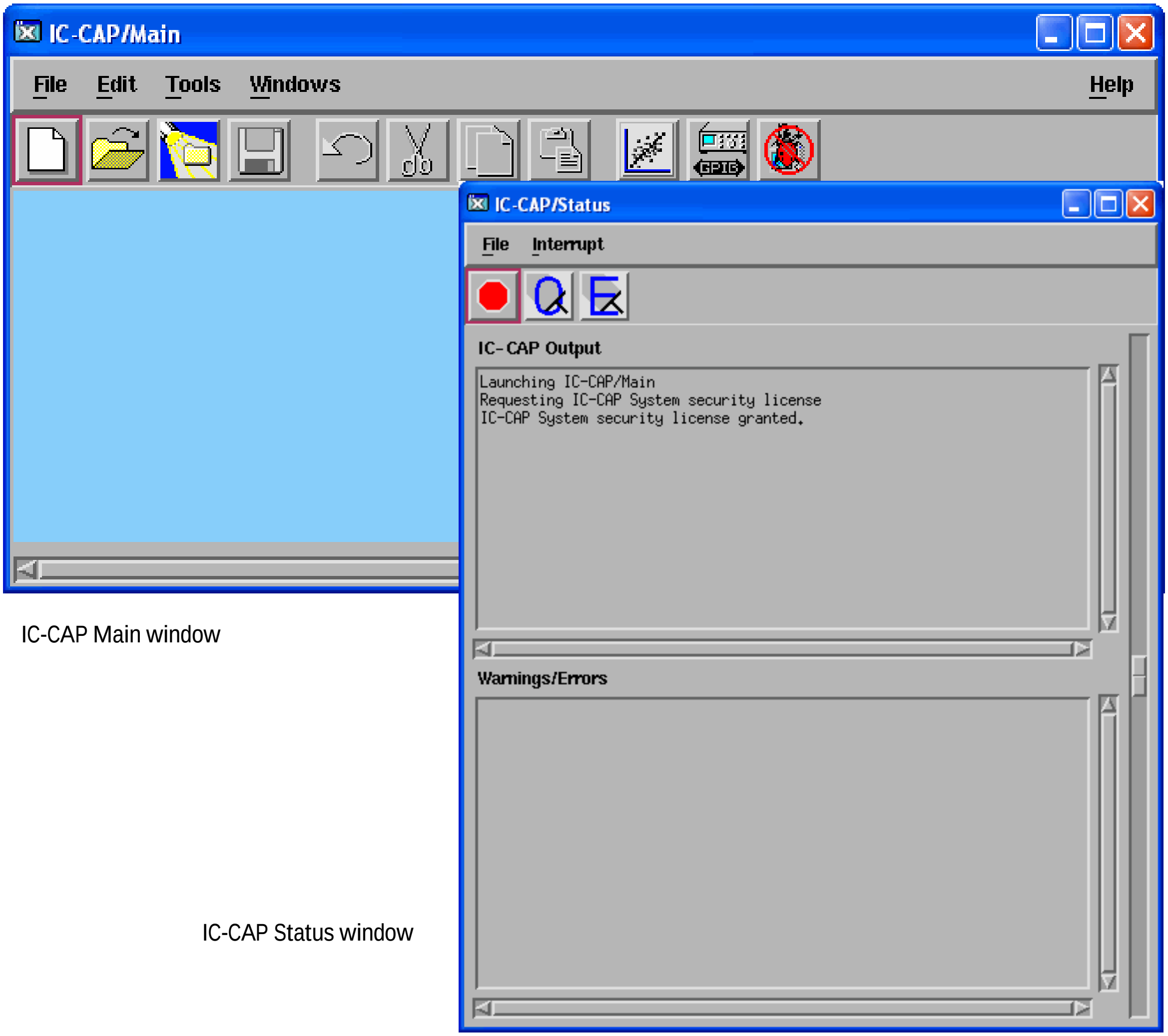
Starting the Program

You can start IC-CAP from any directory. However, after you have saved modified model files, you may find that starting the program from the directory that contains your model files simplifies archiving operations.

To start the program:

1. Change to the working directory.
2. At the system prompt, type `iccap` and press Enter.

   The IC-CAP/Main window and the IC-CAP/Status window open.



```
Launching IC-CAP/Main
Requesting IC-CAP System security license
IC-CAP System security license granted.
```

IC-CAP Main window

IC-CAP Status window

The IC-CAP/Status window displays messages about the program activity and Warnings/Errors messages. In the IC-CAP/Status window, you can open and close output and error log files, and interrupt a process. The icons allow you to interrupt a process (the red stop sign), clear the output log files (the one with the “O”), and clear the error log files (the one with the “E”). The slider along the Status window’s right side enables you to adjust the relative sizes of the Output and Warnings/Errors panes.

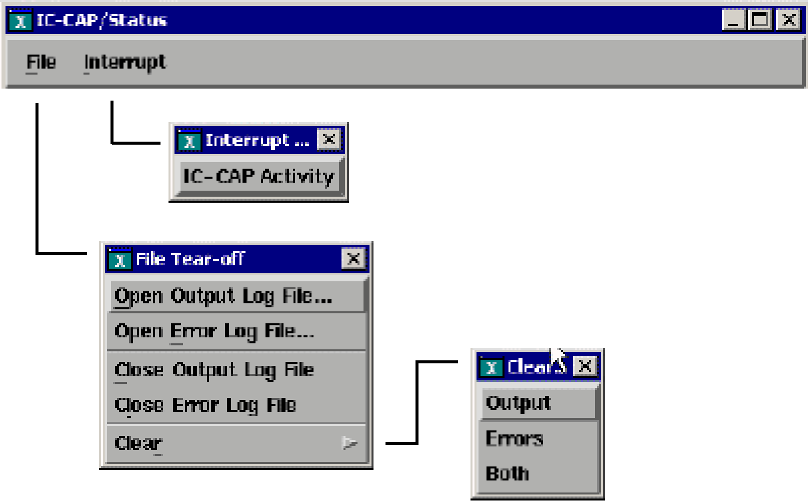


Figure 8 IC-CAP/Status window

To start the program and, at the same time, open a specific model file:

1 Change to the working directory.

2 At the system prompt, type `iccap` <*model filename*> and press Enter.

Problem: When using the hpeesofsim simulator with the IC-CAP Optimizer, the message *Not a typewriter* is reported to the status window. The message is caused by a call to *stty -tostop* in the *hpeesofsim_start* script (*/bin/bootscript.sh*). This start-up script must be accessed before starting hpeesofsim. The message appears if *hpeesofsim_start* is customized by defining a different *HPEESOF_DIR*, or if it is used for a remote simulation. This message is benign, and can be ignored, or fixed if desired.

Workaround: If the message is bothersome, it can be fixed by editing the file *HPEESOF_DIR/bin/bootscript.sh*. Change `stty -tostop` to

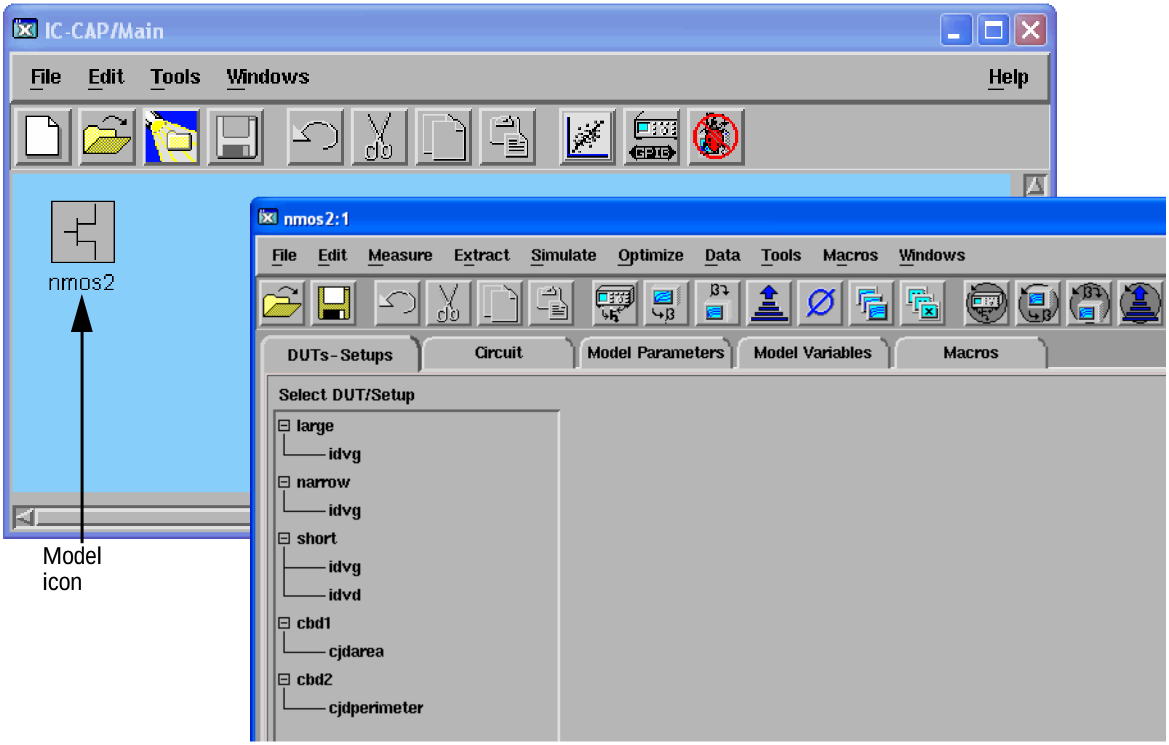
ttymsg = “tty”

if [! “$ttymsg” = “not a tty”]; then

stty -tostop

fi

The IC-CAP/Main window and the IC-CAP/Status window open. The IC-CAP/Main window displays an icon of the model and the Model window opens.



If the Model window does not open, you can open it by double-clicking on the Model icon.

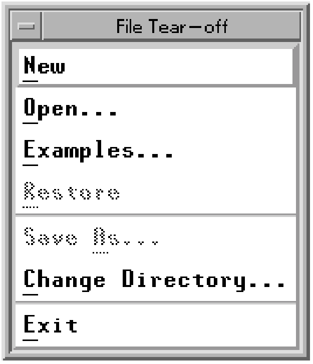
NOTE

You can load a model and start a macro when you start the program. At the system prompt, type: iccap <*model_filename*> <*macro_name*>. For details, refer to “Autostart Macros” on page 471.

Opening a Model File

If you do not load a model file when you start the program, you can open the model file from the IC-CAP/Main window.

To open a model file from the IC-CAP/Main window:



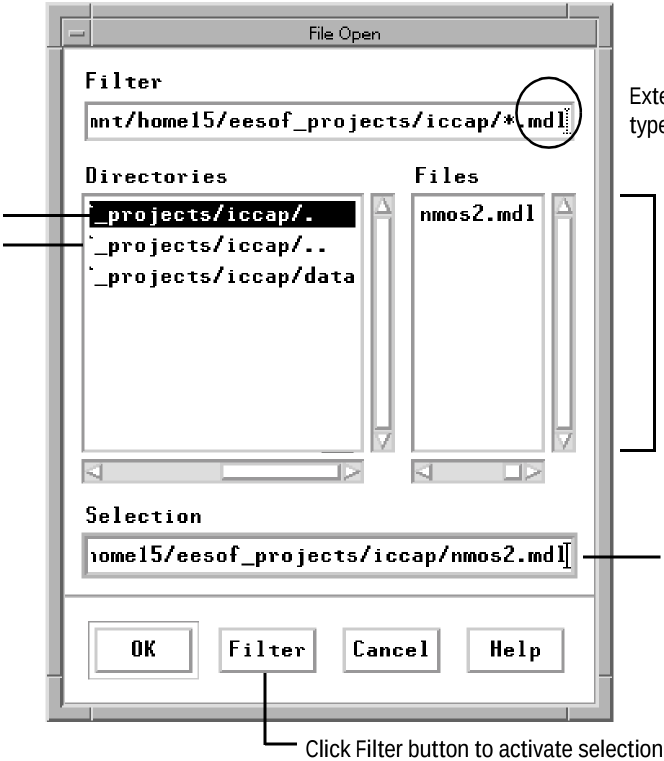
Opens an empty model template

Opens a model in the current directory

Opens a model provided with the program

- To open an empty model template, choose **File > New**.
- To open a model in the current directory, choose **File > Open**.
- To open a model provided with the program, choose **File > Examples**.

A directory dialog box is used when data is being written to or read from a file. Initially, the dialog box displays the files in the directory from which you started the session.



Filter: nnt/home15/eesof_projects/iccap/*.mdl

Directories:
- _projects/iccap/.
- _projects/iccap/..
- _projects/iccap/data

Files:
- nmos2.mdl

Selection: ıome15/eesof_projects/iccap/nmos2.mdl

Extension indicates type of files listed

Current directory

One level up

Lists files in current directory

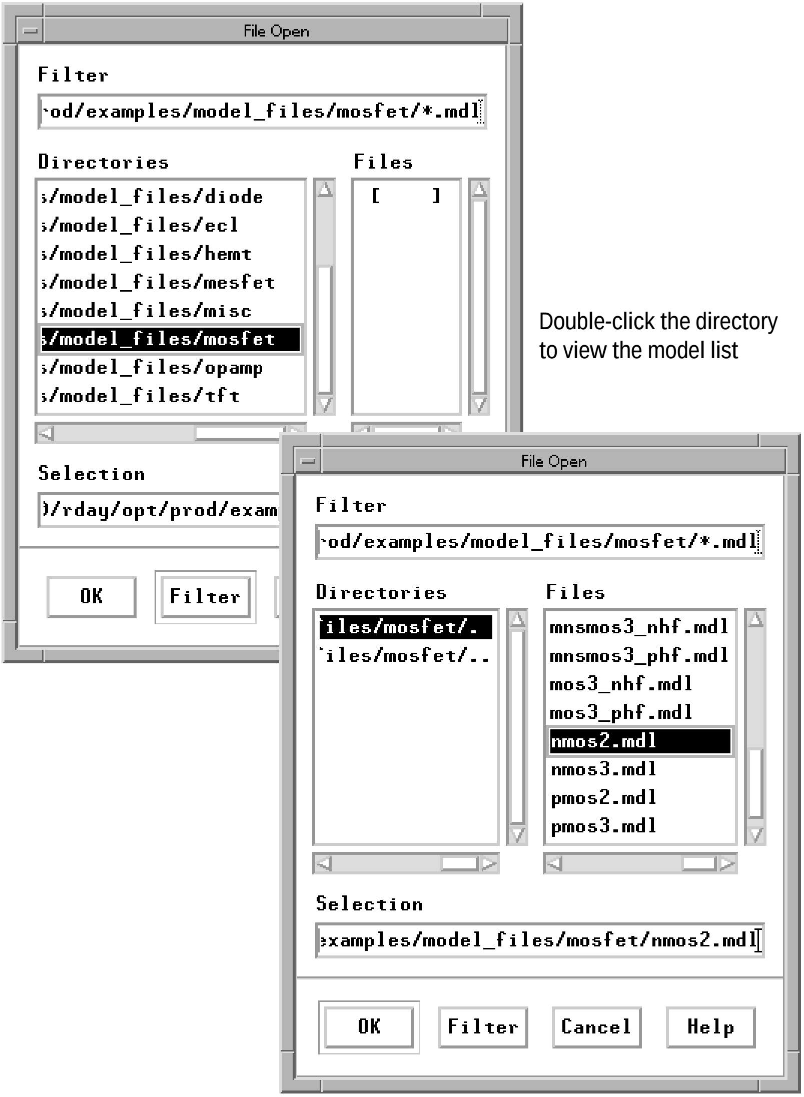
Current file selection

Click Filter button to activate selection displayed in Filter field

NOTE

On the PC, hidden and system folders do not appear in the Directory dialog box. If you can't see a directory, just change its attributes so it's not a hidden or a system file.

When you select *Examples*, double-click the **model_files** directory in the dialog box to display the directories of the models that are provided with the program (see following table). Double-click a directory to see a list of models. Double-click a model to select and open that model.



File Open

Filter: `-od/examples/model_files/mosfet/*.mdl`

Directories:
```
;/model_files/diode
;/model_files/ecl
;/model_files/hemt
;/model_files/mesfet
;/model_files/misc
;/model_files/mosfet
;/model_files/opamp
;/model_files/tft
```

Files: `[ ]`

Selection: `)/rday/opt/prod/exam`

Double-click the directory to view the model list

File Open

Filter: `-od/examples/model_files/mosfet/*.mdl`

Directories:
```
`iles/mosfet/.
`iles/mosfet/..
```

Files:
```
mnsmos3_nhf.mdl
mnsmos3_phf.mdl
mos3_nhf.mdl
mos3_phf.mdl
nmos2.mdl
nmos3.mdl
pmos2.mdl
pmos3.mdl
```

Selection: `examples/model_files/mosfet/nmos2.mdl`

Double-click to select and open a model

Table 5 Example Directory Model Files

$ICCAP_ROOT/examples/model_files/bjt:		
HPEEbjt2.mdl	bjt_npn.mdl	hpsimnpn.mdl
bjt_ft.mdl	bjt_pnp.mdl	hpsimvbic.mdl
bjt_ncehf.mdl	hpsimbjt_nsehf.mdl	mnsnpn.mdl
bjt_nhf.mdl	hpsimbjt_nhf.mdl	mxt3t_npn.mdl
spectre_ncehf.mdl	spectrenpn.mdl	vbicsdd.inc
mxt4t_npn.mdl	mxt504_npn.mdl	sabernpn.mdl
vbic_npn.mdl		
$ICCAP_ROOT/examples/model_files/diode:		
HPDiode.mdl	juncap.mdl	pn_diode.mdl
pn_test.mac		
$ICCAP_ROOT/examples/model_files/ecl:		
ECLgate.mdl	ECLornor.mdl	
$ICCAP_ROOT/examples/model_files/hemt:		
HPEEhemt1.mdl		
$ICCAP_ROOT/examples/model_files/mesfet:		
CGaas1.mdl	CGaashfax.mdl	UCBGaas.mdl
CGaas2.mdl	HPEEfet3.mdl	UGaashf.mdl
CGaashf.mdl	HPRootFet.mdl	hpsimHPEEfet3.mdl
$ICCAP_ROOT/examples/model_files/misc:		
54120.demo.mdl	lc.mdl	plot_doc.mac
TRLCAL.mdl	npnwpnp.mdl	sabercirc.mdl
hp54750.mdl	sys110_verify.mdl	sys_test.mdl
sys_testrf.mdl		

Table 5 Example Directory Model Files (continued)

$ICCAP_ROOT / examples / model_files / mosfet:		
mm9.mdl	hpmos28.mdl	mm9_demo.mdl
HPRootMos.mdl	hnmos28.mdl	pmos3.mdl
sabernmos.mdl	hnmos6.mdl	pmos2.mdl
nmos2.mdl	nmos3.mdl	
$ICCAP_ROOT / examples / model_files / mosfet / bsim3:		
BSIM3_DC_CV_Extract.mdl	BSIM3_DC_CV_Measure.mdl	BSIM3_RF_Extract.mdl
Whats_New.mdl	BSIM3_RF_Measure.mdl	
$ICCAP_ROOT / examples / model_files / mosfet / bsim3 / tutorial:		
BSIM3_AC_Noise_Tutorial.mdl	BSIM3_CV_Tutorial.mdl	BSIM3_DC_Tutorial.mdl
BSIM3_Temp_Tutorial.mdl		
$ICCAP_ROOT / examples / model_files / mosfet / bsim3 / utilities:		
BSIM3_DC_CV_Finetune.mdl		
$ICCAP_ROOT / examples / model_files / mosfet / bsim4:		
BSIM4_DC_CV_Extract.mdl	BSIM4_DC_CV_Measure.mdl	BSIM4_RF_Extract.mdl
Whats_New.mdl	BSIM4_RF_Measure.mdl	
$ICCAP_ROOT / examples / model_files / mosfet / bsim4 / tutorial:		
BSIM4_DC_CV_Tutorial.mdl		
$ICCAP_ROOT / examples / model_files / mosfet / bsim4 / utilities:		
BSIM4_DC_CV_Finetune.mdl		
$ICCAP_ROOT / examples / model_files / noise:		
noise_simu.mdl		
$ICCAP_ROOT / examples / model_files / noise / 1_f_toolkit:		
bjt_1f_noise.mdl	mos_1f_noise.mdl	
$ICCAP_ROOT / examples / model_files / opamp:		
mnsopamp.mdl	opamp.mdl	

Table 5 Example Directory Model Files (continued)

$ICCAP_ROOT / examples / model_files / pulse:		
bjt_ncehfp.mdl	opver_k46.mdl	opver_k48a.mdl
cal_8510_p.mdl	opver_k49.mdl	opver_85124P.mdl
opver_k48b.mdl		
$ICCAP_ROOT / examples / model_files / statistics / load_data:		
bsim3_1.mdl	bsim3_4.mdl	bsim3_7.mdl
bsim3_10.mdl	bsim3_40.mdl	bsim3_70.mdl
bsim3_11.mdl	bsim3_41.mdl	bsim3_71.mdl
bsim3_12.mdl	bsim3_42.mdl	bsim3_72.mdl
bsim3_13.mdl	bsim3_43.mdl	bsim3_73.mdl
bsim3_14.mdl	bsim3_44.mdl	bsim3_74.mdl
bsim3_15.mdl	bsim3_45.mdl	bsim3_75.mdl
bsim3_16.mdl	bsim3_46.mdl	bsim3_76.mdl
bsim3_17.mdl	bsim3_47.mdl	bsim3_77.mdl
bsim3_18.mdl	bsim3_48.mdl	bsim3_78.mdl
bsim3_19.mdl	bsim3_49.mdl	bsim3_79.mdl
bsim3_2.mdl	bsim3_5.mdl	bsim3_8.mdl
bsim3_20.mdl	bsim3_50.mdl	bsim3_80.mdl
bsim3_21.mdl	bsim3_51.mdl	bsim3_81.mdl
bsim3_22.mdl	bsim3_52.mdl	bsim3_82.mdl
bsim3_23.mdl	bsim3_53.mdl	bsim3_83.mdl
bsim3_24.mdl	bsim3_54.mdl	bsim3_84.mdl
bsim3_25.mdl	bsim3_55.mdl	bsim3_85.mdl
bsim3_26.mdl	bsim3_56.mdl	bsim3_86.mdl
bsim3_27.mdl	bsim3_57.mdl	bsim3_87.mdl
bsim3_30.mdl	bsim3_58.mdl	bsim3_88.mdl

Table 5 Example Directory Model Files (continued)

bsim3_31.mdl	bsim3_59.mdl	bsim3_89.mdl
bsim3_32.mdl	bsim3_6.mdl	bsim3_9.mdl
bsim3_33.mdl	bsim3_60.mdl	bsim3_90.mdl
bsim3_34.mdl	bsim3_61.mdl	bsim3_91.mdl
bsim3_35.mdl	bsim3_62.mdl	bsim3_92.mdl
bsim3_36.mdl	bsim3_63.mdl	bsim3_93.mdl
bsim3_37.mdl	bsim3_64.mdl	bsim3_94.mdl
bsim3_38.mdl	bsim3_65.mdl	bsim3_95.mdl
bsim3_39.mdl	bsim3_66.mdl	bsim3_96.mdl
bsim3_28.mdl	bsim3_67.mdl	bsim3_97.mdl
bsim3_29.mdl	bsim3_68.mdl	bsim3_98.mdl
bsim3_3.mdl	bsim3_69.mdl	load_stat_data.mdl

NOTE

Additional examples are located in *$ICCAP_ROOT/examples/demo_features* which contains examples that answer questions IC-CAP users ask most often. For information about the demos, open the *README.mdl* file in the */demo_features* directory.

IC-CAP File Extensions

Each file type has a unique file extension that is added to the filename automatically. The following table lists these file extensions.

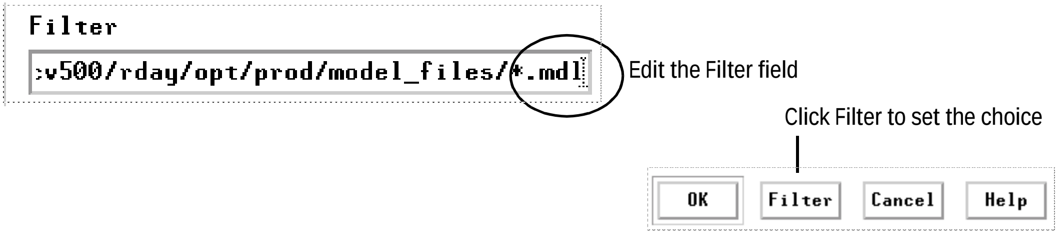
Table 6 IC-CAP File Extensions

Extension	File Type	Extension	File Type	Extension	File Type
.mdl	Model	.dut	DUT	.hdw	Hardware
.mps	Model Parameters	.set	Setup	.inp	Input
.dps	DUT Parameters	.mac	Macro	.out	Output
.vat	Variables Table	.cir	Circuit	.xfm	Transform
.iot	Instrument Options	.tci	Test Circuit	.plt	Plot
.ds	Dataset	.pop	Plot Optimizer		
.mdm	Data Manager				

To list a selected file type only:

1 In a directory dialog box, edit the Filter field by typing the file extension type.

2 Choose **Filter** to set the file extension type.



Filter

:v500/rday/opt/prod/model_files/*.mdl

Edit the Filter field

Click Filter to set the choice

Changing Directories

When you load an example model, the directory path for the model icon in the IC-CAP/Main window points to the installation directory. You can redirect the path to your working directory before saving the model data.

To change directory paths:

1 Select **File > Change Directories**.

2 Edit the selected path by typing a new name in the Selection field or by browsing the Directories list to the new directory.

NOTE

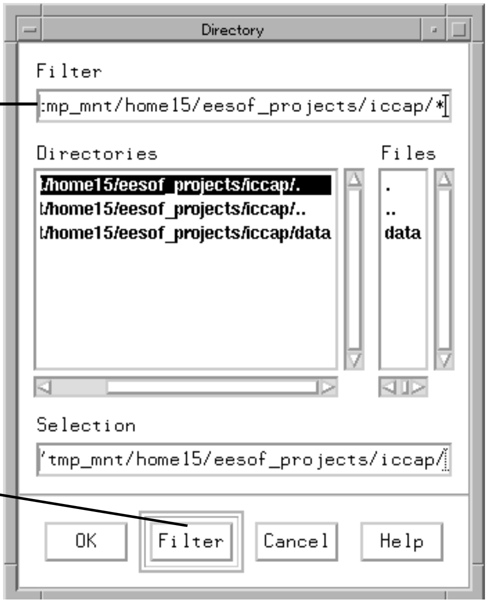
The path for IC-CAP model files cannot contain any folder names that use a space. For example, *C:\Model Files\IC-CAP 2004*. If a model file is saved in a folder name with spaces, you will not be able to open the model file. You will have to move the model file to a folder name that does not use a space.

On the PC, hidden and system folders do not appear in the Directory dialog box. If you can't see a directory, just change its attributes so it's not a hidden or a system file.

3 Choose **OK** to set the selected path.

Edit the directory path

Click Filter to set the choice



Filter

:mp_mnt/home15/eesof_projects/iccap/*

Directories

t/home15/eesof_projects/iccap/.
t/home15/eesof_projects/iccap/..
t/home15/eesof_projects/iccap/data

Files

.
..
data

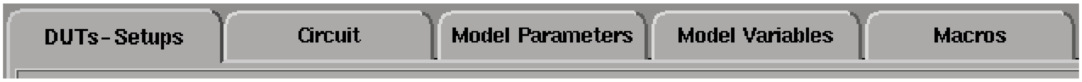
Selection

'tmp_mnt/home15/eesof_projects/iccap/

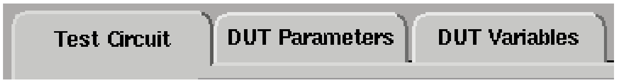
The Model Window Folders

The Model window contains sets of folders where you can view and edit the Model, DUTs, and Setups. This example describes each folder briefly. The folders are described in detail in the following chapters.

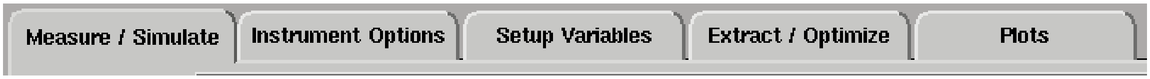
Model



DUT



Setup



Setting System Variables

Variables can be defined at the top level (system) or at the Model, DUT, and Setup levels. You can define as many variables as you need and use these variables for other functions, such as extractions.

A set of predefined variables are supplied with the program. If you want to alter behavior of IC-CAP defaults and/or define new default behavior, you can set a system variable. Also, you can set a system variable to function as a *toolbox* for miscellaneous control parameters and save the set as a file so you can reuse it for different models.

NOTE

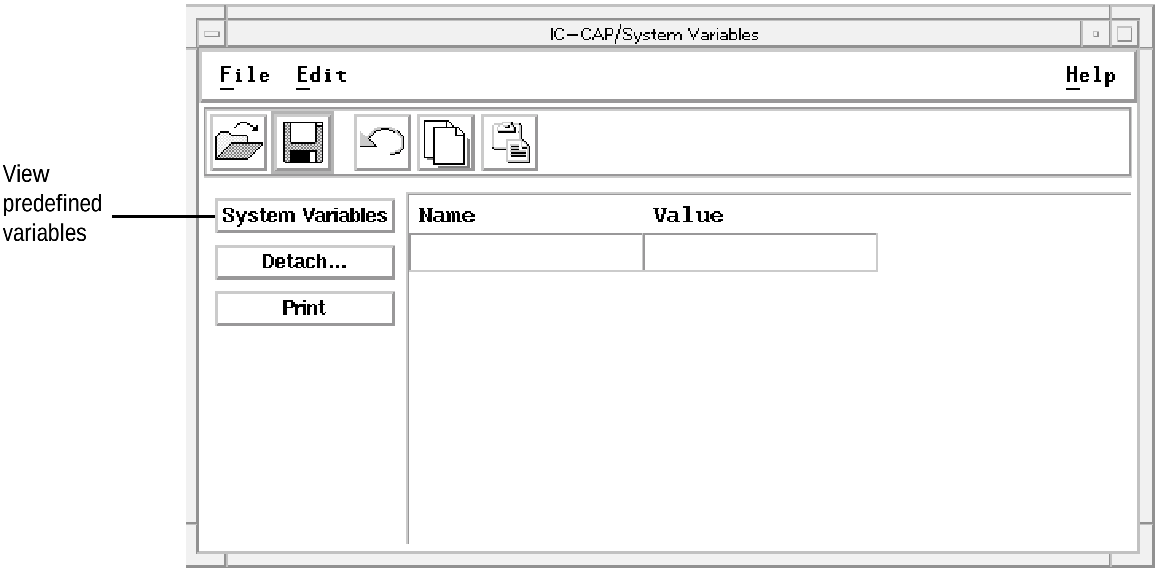
For details on predefined system variables, refer to Chapter 11, “Variables,” in the *Reference* manual.

To view system variables:

1 In the IC-CAP/Main window, select **Tools**.

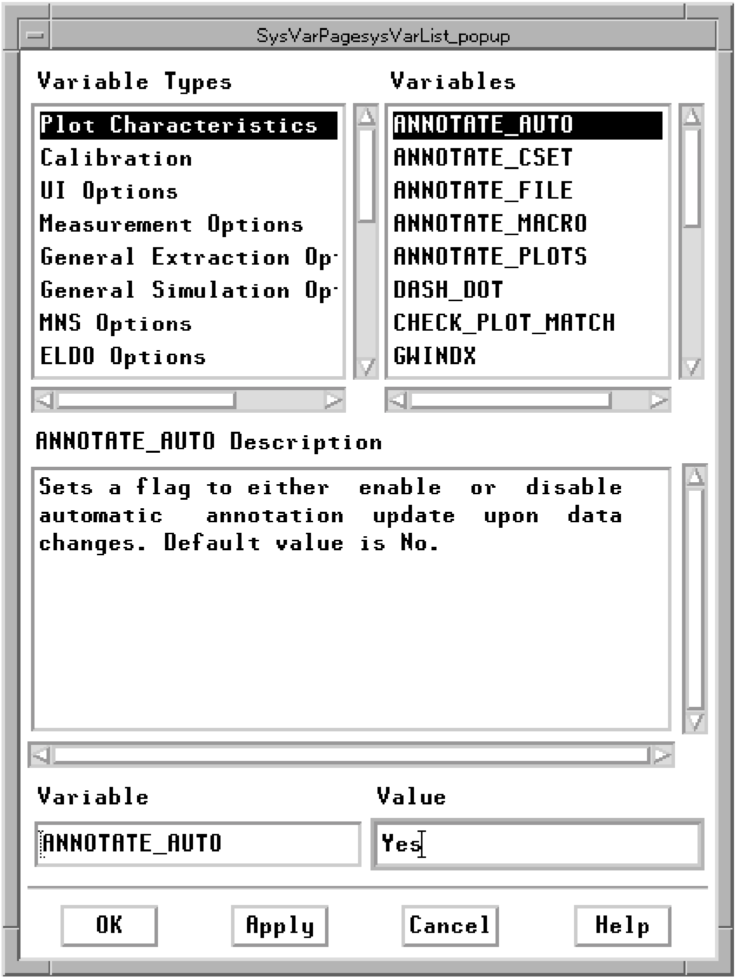
2 Select **System Variables**.

3 The System Variables window opens. Click **System Variables** to view predefined variables.



4 In the dialog box, select a Variable Type from the Variable Types list. Then select a Variable in the Variables list.

Select a variable type



SysVarPagesysVarList_popup

Variable Types

- Plot Characteristics
- Calibration
- UI Options
- Measurement Options
- General Extraction Op
- General Simulation Op
- MNS Options
- ELDO Options

Variables

- ANNOTATE_AUTO
- ANNOTATE_CSET
- ANNOTATE_FILE
- ANNOTATE_MACRO
- ANNOTATE_PLOTS
- DASH_DOT
- CHECK_PLOT_MATCH
- GWINDX

ANNOTATE_AUTO Description

Sets a flag to either enable or disable automatic annotation update upon data changes. Default value is No.

Variable: ANNOTATE_AUTO

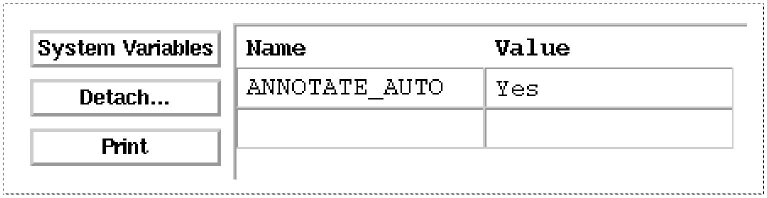
Value: Yes

Select a variable

Edit the variable value

5 The selected variable and value appears in the Variable and Value fields. If appropriate, edit the Value field.

6 Click **Apply** to add the variable to the System Variables list.



Name	Value
ANNOTATE_AUTO	Yes

7 When you have added all the system variables, click **OK** to close the dialog box.

Saving Model Data

You can save IC-CAP information at several levels. For example, in the Model window:

- The parameter set can be saved before extractions are performed and restored if the extracted values are unacceptable.
- An entire setup can be saved for future restoration in order to make temporary changes to information such as input voltages.
- An output can be saved if you want to attempt another measurement but are not sure if the results will be improved.
- A snapshot of an entire model, or of all open models, can be made and used as a general-purpose backup.

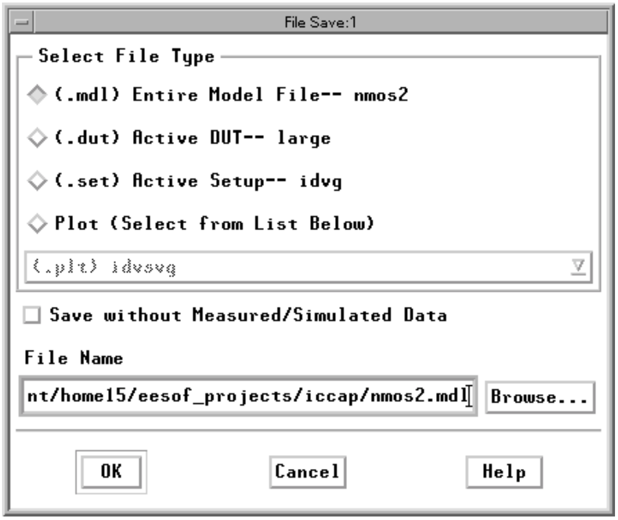
NOTE

The path for IC-CAP model files cannot contain any folder names that use a space. For example, do not save to *C:\Model Files\IC-CAP 2006*. If you include a space in a folder name, you will not be able to load the model file. You will have to move the model file to a folder name that does not use a space.

To save model data from the Model window:

1 Select **Save**. 💾

2 Choose from the save options: save the entire model file, the active DUT, the active setup, or the active plot, and save without the measured/simulated data.



Choose Save options

Enter a unique filename

File Save:1

Select File Type

◆ (.mdl) Entire Model File-- nmos2

◇ (.dut) Active DUT-- large

◇ (.set) Active Setup-- idvg

◇ Plot (Select from List Below)

(.plt) idvsvg

□ Save without Measured/Simulated Data

File Name: nt/home15/eesof_projects/iccap/nmos2.mdl

3 In the File Name field, type a unique filename and choose **OK**.

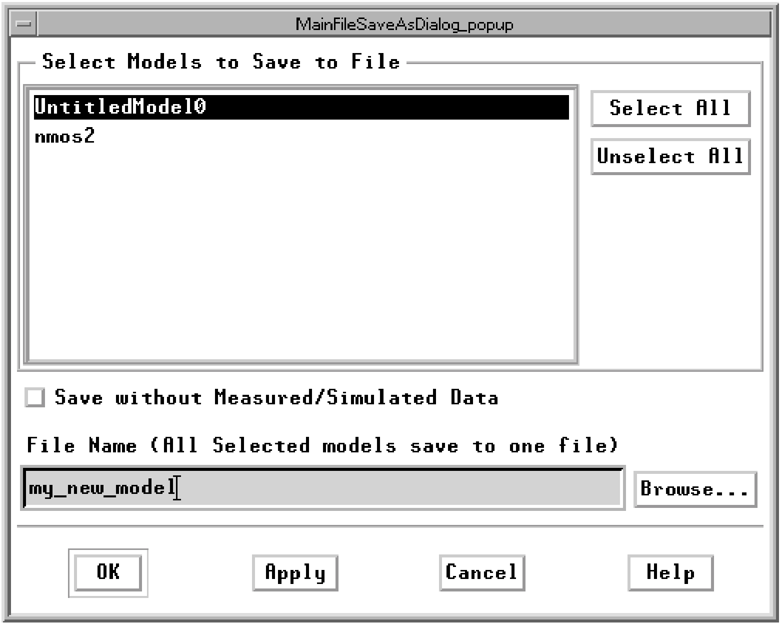
To save model data from the IC-CAP/Main window:

1 Select **Save**.

2 Select the open models you want to save.

Select files to save

Enter a unique filename



MainFileSaveAsDialog_popup

Select Models to Save to File

UntitledModel0
nmos2

☐ Save without Measured/Simulated Data

File Name (All Selected models save to one file)

my_new_model

3 In the File Name field, enter a unique filename and click **OK**.

NOTE

If a file was saved previously by that name, an information dialog box opens.

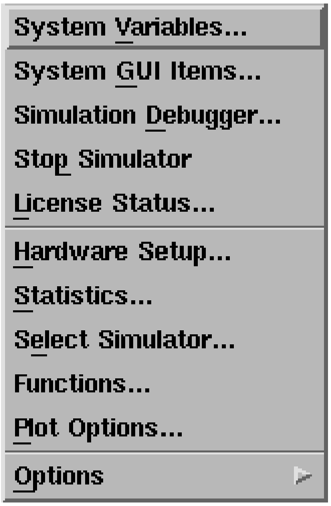
To replace the existing model by that name, choose Yes.

To avoid replacing the existing model by that name, choose No and enter a unique file name. Or use the *Save As* command to save the model to different name.

You can use the same name as the supplied file, as long as you save the model to a different directory. For details on changing directory paths, refer to “Changing Directories” on page 56.

Tools

The Tools menu in the IC-CAP/Main window provides access to a variety of features and options.



System Variables Displays a window for viewing global (system) variables. For details, refer to "Setting System Variables" on page 59 and Chapter 11, "Variables," in the *Reference* manual.

System GUI Items Displays the System GUI Items window. For details, refer to Chapter 12, "Creating Graphic User Interfaces."

Simulation Debugger Displays the Simulation Debugger window. For details, refer to "Using the Simulation Debugger" on page 232.

Stop Simulator Stops any currently running piped simulator. The command is most useful when using the ADS simulator with the Root model to ensure correct simulations can be made after changing directories to test other models, and to allow model regeneration.

License Status Displays a window with dynamically updated license information:

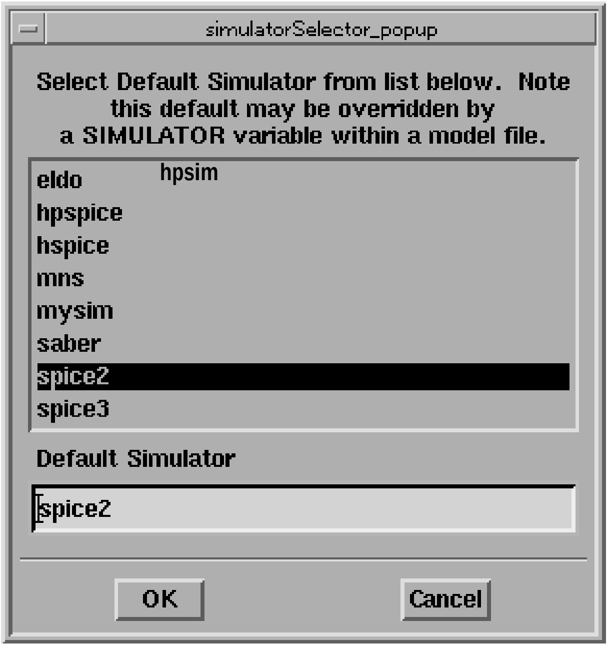
- The codewords currently in use
- The codewords currently available

To release a specific license, select it and click **Release**.

Hardware Setup Displays the Hardware Setup window. For details, refer to "Configuring the System" on page 179.

Statistics Launches the IC-CAP Statistics program, if licensed. IC-CAP Statistics provides tools for identifying and analyzing the inter-relationships between device model parameters and electrical test data. For details, refer to the *Statistics Analysis* manual.

Select Simulator Displays a dialog for changing the default simulator after startup.



simulatorSelector_popup

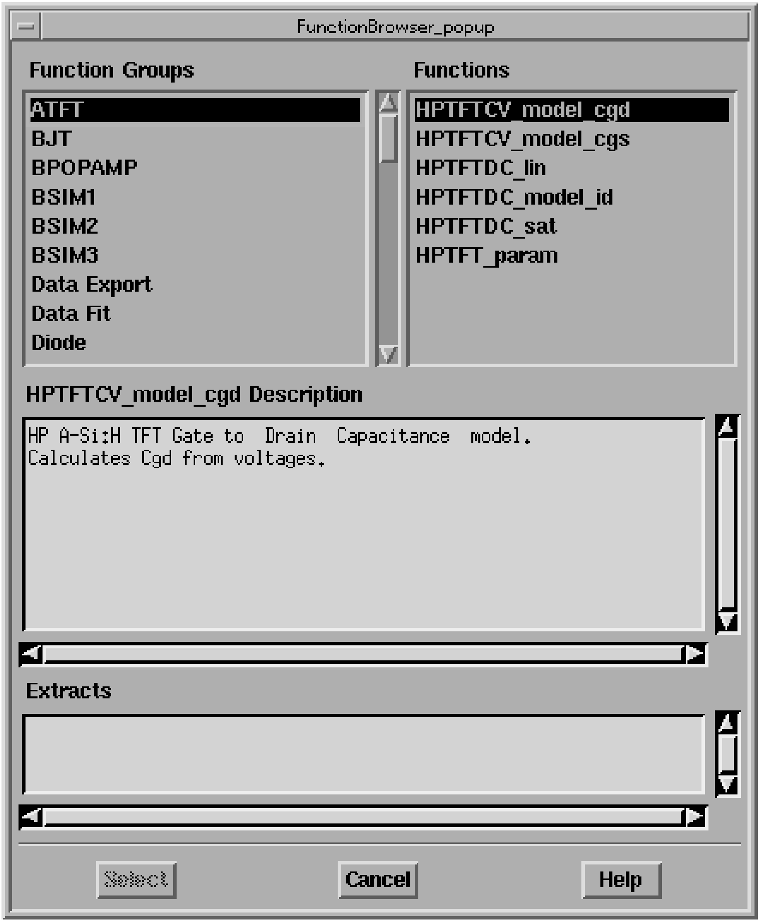
Select Default Simulator from list below. Note this default may be overridden by a SIMULATOR variable within a model file.

eldo hpsim
hpspice
hspice
mns
mysim
saber
spice2
spice3

Default Simulator: spice2

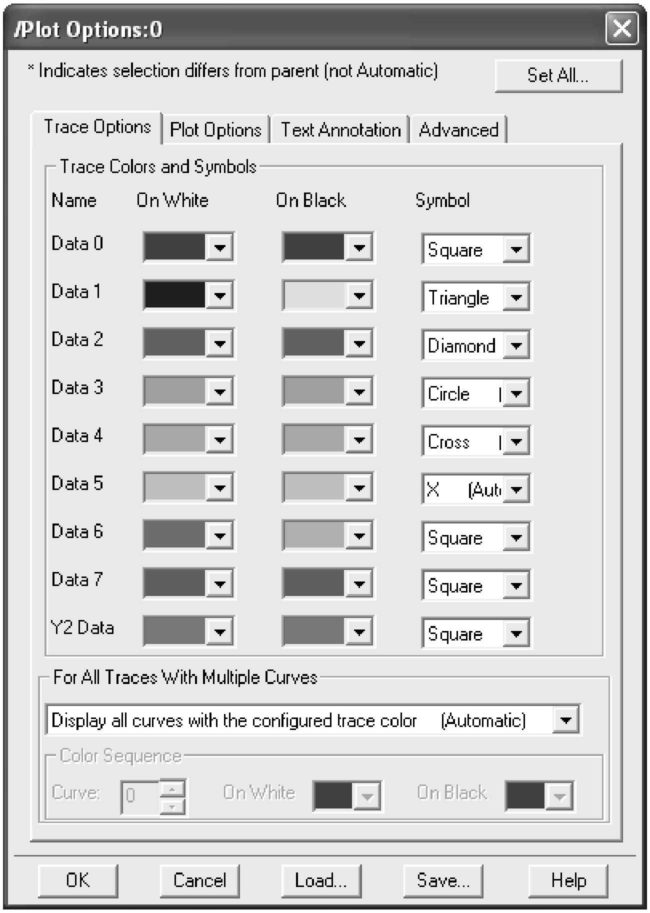
Note that this selection is overridden by a SIMULATOR variable. For details, refer to "Selecting a Simulator" on page 220.

Functions Displays the Function Browser dialog box for reviewing available functions. For details, refer to Chapter 9, "Using Transforms and Functions."



```
HP A-Si:H TFT Gate to  Drain  Capacitance  model.
Calculates Cgd from voltages.
```

Plot Options Displays the Plot Options dialog box for defining trace options, plot options, and text annotation. For details, refer to "Setting Plot Options" on page 431.



/Plot Options:0

* Indicates selection differs from parent (not Automatic)

Trace Colors and Symbols

Name	On White	On Black	Symbol
Data 0	■	■	Square
Data 1	■	□	Triangle
Data 2	■	■	Diamond
Data 3	■	■	Circle
Data 4	■	■	Cross
Data 5	■	■	X (Aut
Data 6	■	■	Square
Data 7	■	■	Square
Y2 Data	■	■	Square

For All Traces With Multiple Curves

Display all curves with the configured trace color (Automatic)

Color Sequence

Curve: 0 On White ■ On Black ■

Options Displays the following options:

- File Debug—Toggles the debugging facility on and off. When on, messages are recorded in the file .icdebug.
- Screen Debug—Toggles the debugging facility on and off. When on, messages are displayed in the IC-CAP Status window.
- View GUI Pages—Toggles displaying the GUI Items page on and off. When on, you will see a GUI Items page at the Model level, DUT level, and Setup level.

- Status Window to Top—If toggled on, the Status window pops to the front of the screen anytime new messages are displayed in it.
- Diagnostics—Executes internal diagnostics.

Aborting an Operation

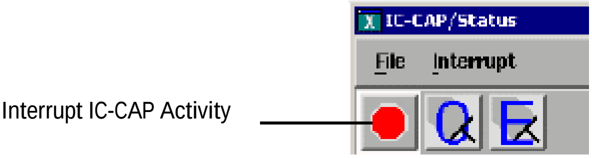
You can interrupt an IC-CAP activity from the Status window. The operations that can be aborted are: Measurement, Simulation, Macro Execution, Transform Execution, and Optimization. IC-CAP control is returned to you.

If you abort a measurement while an internal system sweep is in progress, the measurement in IC-CAP is aborted, but the instrument continues to step through its sweep values until the sweep is completed. If another IC-CAP measurement using this instrument is attempted before the sweep is completed, IC-CAP waits until the sweep is done before performing the measurement.

After an optimization is aborted, an optimization summary is printed before control is returned.

To abort an IC-CAP activity:

In the IC-CAP/Status window, click the Interrupt IC-CAP Activity button.



Alternatively, you can select the menu item, **Interrupt > IC-CAP Activity**.

Exit IC-CAP

To exit IC-CAP, select **File > Exit** in the IC-CAP/Main window. This opens the *Save As* dialog box to give you an opportunity to save changes to models before exiting the program. Click on the *Exit* button to automatically close all open windows and terminate the program.

3
An Example IC-CAP Session

Using the IC-CAP Modeling System, you can characterize a device by following this general procedure:

- Start the program and load the model file into memory.
- Measure the device characteristics.
- View the measured data.
- Perform a simulation.
- Extract the model parameters from the measured data.
- Simulate with extracted parameters.

- Compare measured data and simulated data.
- Optimize model parameters as needed.
- Measure the remaining devices.

IC-CAP supplies a variety of model extractions in model files that you can load and work with immediately. This section provides an example of a typical IC-CAP session, using a supplied model file.

NOTE

Your IC-CAP installation may not be identical to the system described. Depending on the options of your IC-CAP product, you may not be able to perform some portions of this demonstration. In such cases, an error message describes the missing codeword or license. For more information, refer to the installation procedures or consult your system administrator.

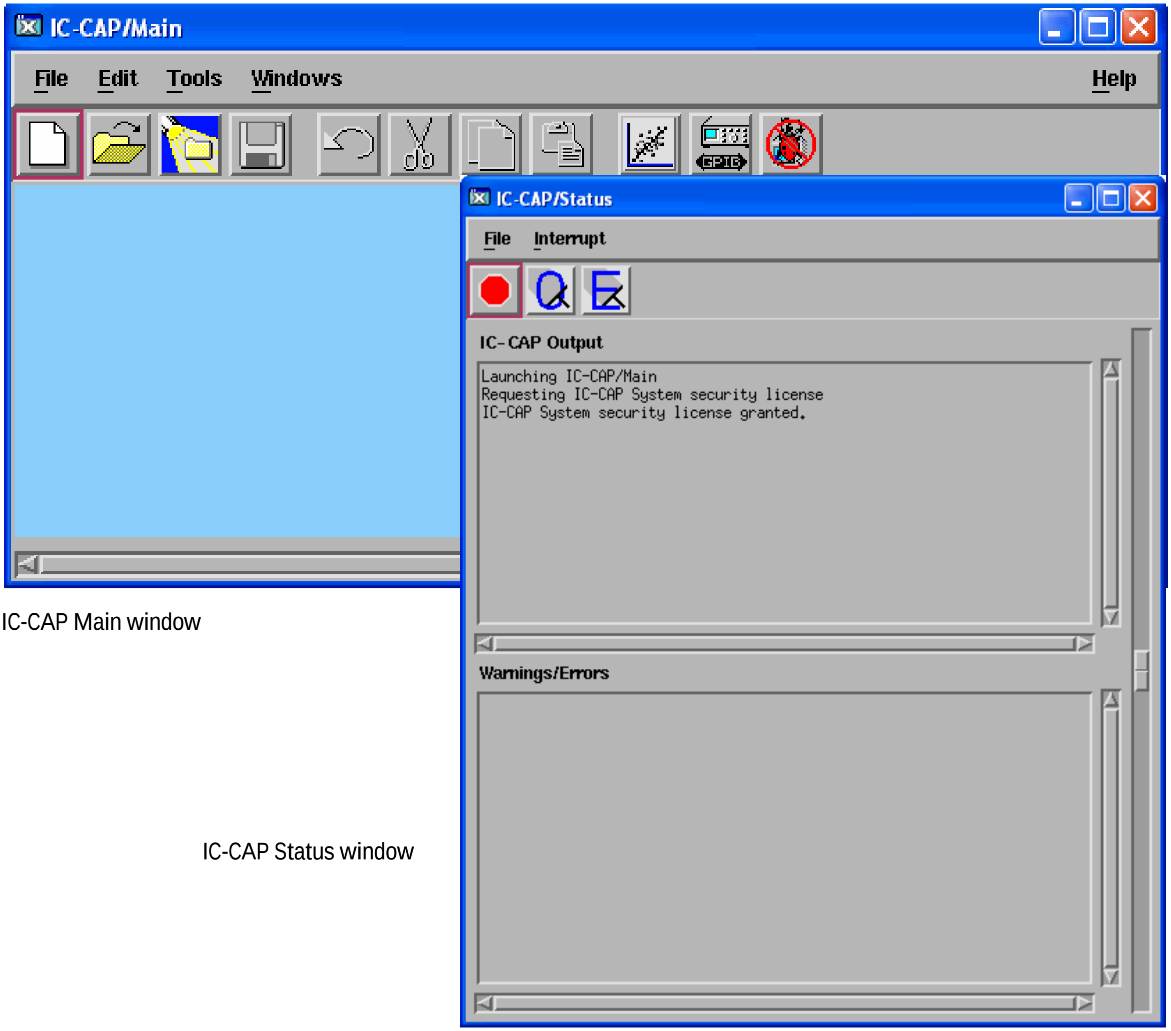
Starting the Program

To start the program for the first time:

1 Change to the working directory.

2 At the system prompt, type `iccap` and press Enter.

The IC-CAP/Main window and IC-CAP/Status window open.



```
Launching IC-CAP/Main
Requesting IC-CAP System security license
IC-CAP System security license granted.
```

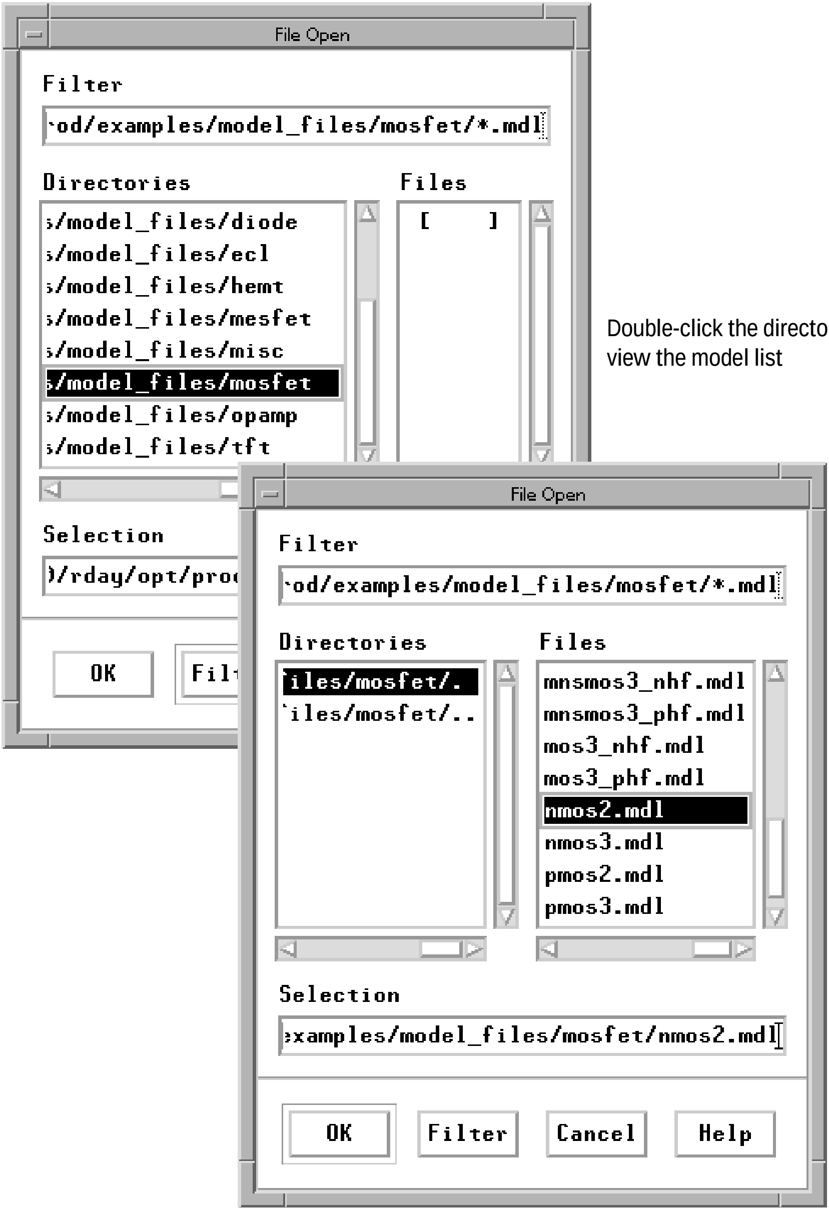
IC-CAP Main window

IC-CAP Status window

Opening a Model File

1 In the IC-CAP/Main window, click **Examples** on the toolbar.

2 The directory dialog box displays the directories of examples. Double-click **model_files** to see the models that are provided with the program. Double-click the **mosfet** directory. Then double-click **nmos2.mdl**



Filter
`od/examples/model_files/mosfet/*.mdl`

Directories
```
s/model_files/diode
s/model_files/ecl
s/model_files/hemt
s/model_files/mesfet
s/model_files/misc
s/model_files/mosfet
s/model_files/opamp
s/model_files/tft
```

Files
`[    ]`

Selection
`)/rday/opt/proc`

Double-click the directory to view the model list

File Open

Filter
`od/examples/model_files/mosfet/*.mdl`

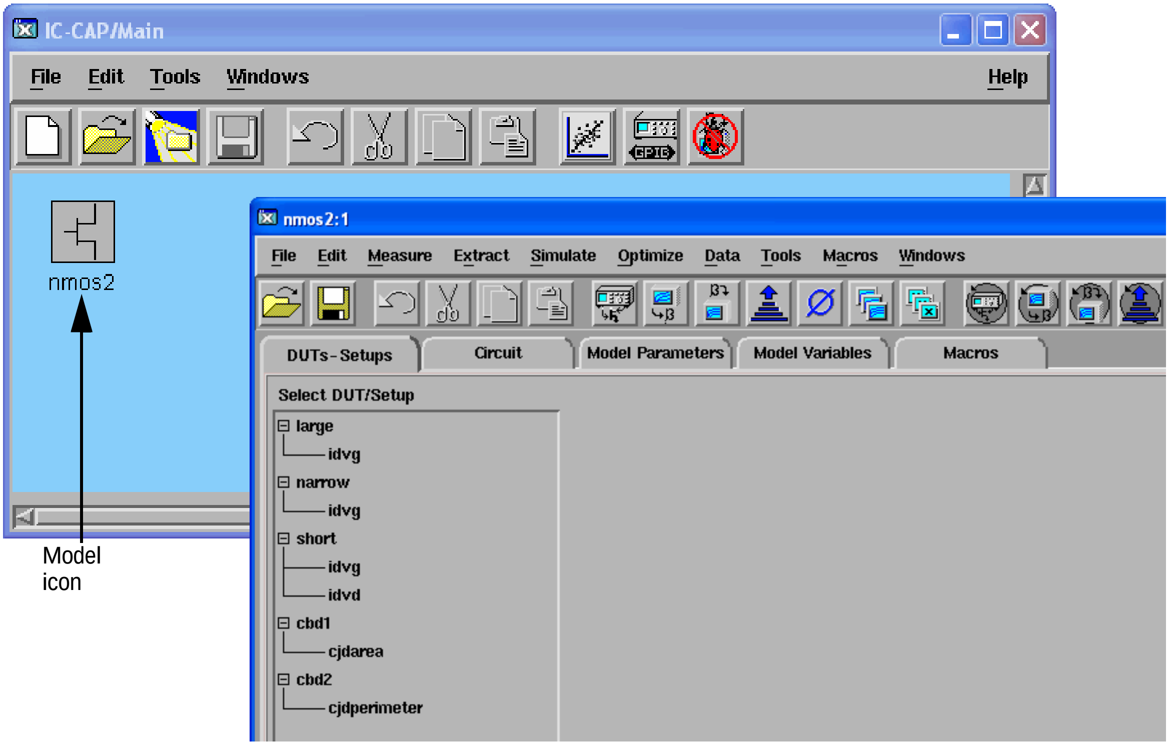
Directories
```
iles/mosfet/.
iles/mosfet/..
```

Files
```
mnsmos3_nhf.mdl
mnsmos3_phf.mdl
mos3_nhf.mdl
mos3_phf.mdl
nmos2.mdl
nmos3.mdl
pmos2.mdl
pmos3.mdl
```

Selection
`xamples/model_files/mosfet/nmos2.mdl`

Double-click to select and open a model

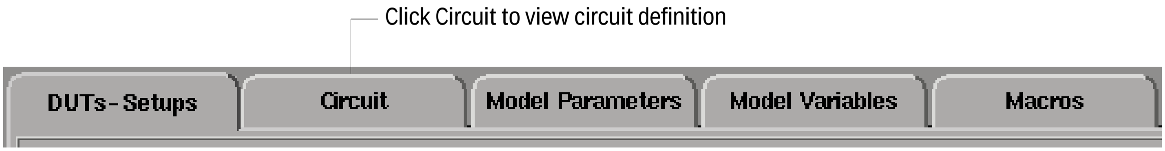
An icon of the model file displays in the IC-CAP/Main window and the Model window opens. The *nmos2.mdl* example extracts parameters for the Level 2 N-channel UCB MOSFET transistor.



Viewing the Circuit Description

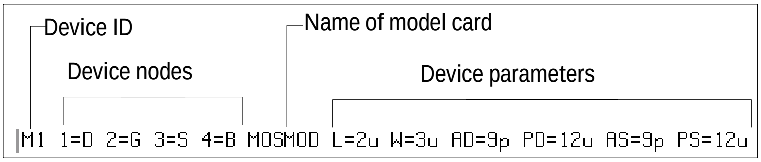
The Circuit description defines the device topology and parameters to be extracted.

To view the default values of the parameters, select **Circuit.**


Click Circuit to view circuit definition

The circuit definition begins with the Options definition (.OPTION). The definition lists the device to be measured, the device nodes, the name of the model card that contains the model parameters used by the device, and the device parameters.

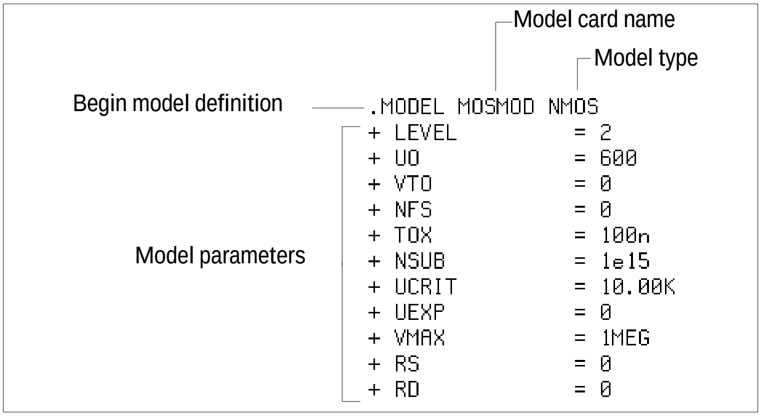
Circuit Options definition



```
M1 1=D 2=G 3=S 4=B MOSMOD L=2u W=3u AD=9p PD=12u AS=9p PS=12u
```

The next section is the model card definition (.MODEL). The definition lists the name of the model card, the model type, and the contents of the model card. The contents of the model card consists of the parameters that are to be extracted (model parameters) and their assigned initial values.

Circuit Model definition

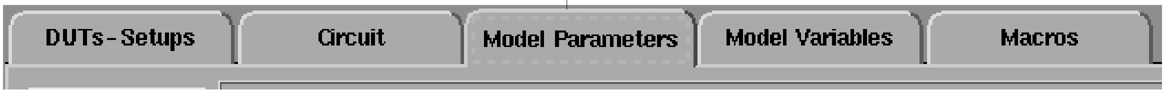


```
.MODEL MOSMOD NMOS
+ LEVEL          = 2
+ UO             = 600
+ VTO            = 0
+ NFS            = 0
+ TOX            = 100n
+ NSUB           = 1e15
+ UCRIT          = 10.00K
+ UEXP           = 0
+ VMAX           = 1MEG
+ RS             = 0
+ RD             = 0
```

Defining Model Parameters

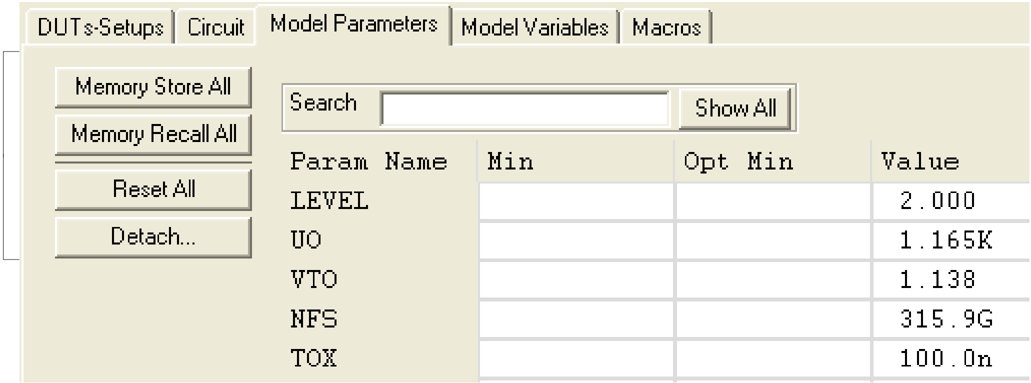
The Model Parameters folder displays the current parameter values. These parameters are common to all DUTs for that model and are used in a simulation. When a new model is read, these values are set to the defaults defined in the circuit. The values change after extractions and optimization are performed.

To define the model parameters, select **Model Parameters.**



Click Model Parameters
to view current parameter values

The Parameters table contains the same names as those specified in the *MODEL* definition of the Circuit. However, by editing the table, the values can be set independently of those in the circuit definition and can be specified as a real number or an equation. In addition, you can set limits by entering minimum and maximum values for each parameter. Parameter values that are outside their limits are clamped to their minimum or maximum values.



Param Name	Min	Opt Min	Value
LEVEL			2.000
UO			1.165K
VTO			1.138
NFS			315.9G
TOX			100.0n

Control value
set selection

After editing the defaults, you can store the current set of parameter values in a temporary buffer (*Memory Store All*) and recall these values at any time during the IC-CAP session (*Memory Recall All*). If you prefer, you can restore the default values (*Reset All*).

NOTE The values stored using Memory Store All are not saved with the model file and are lost when the IC-CAP session ends.

Defining Model Variables

Variables can be defined at the top, or system, level or at the Model, DUT and Setup levels. You can define as many variables as you need and use these variables for other functions, such as extractions.

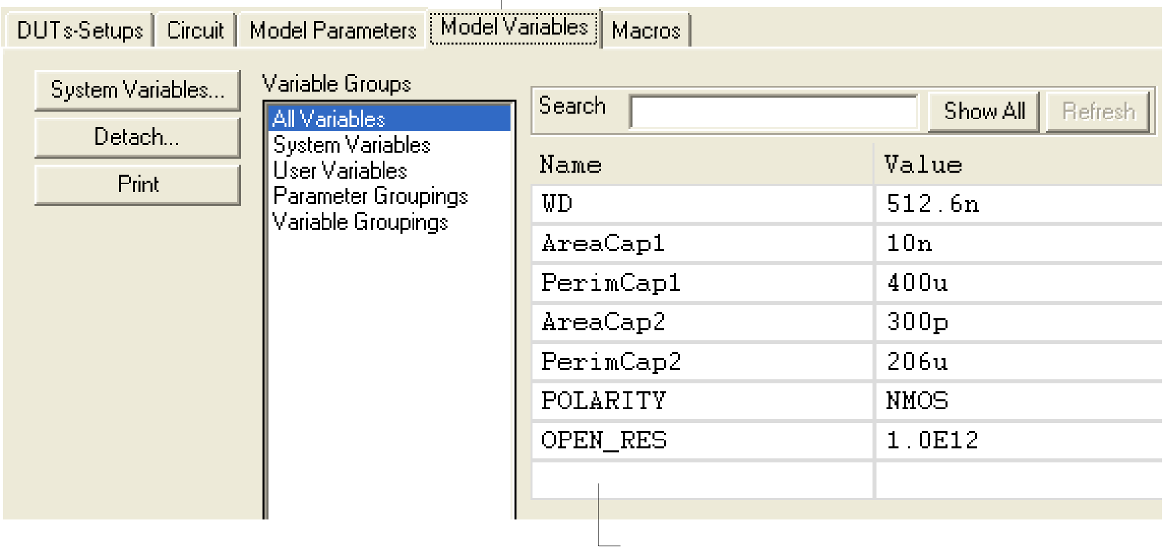
Model variables can be accessed by any of the components of a model, including the DUTs and setups, while DUT and Setup variables (such as inputs or outputs) can be accessed only by the components of that setup.

A useful example of a variable defined at the model level is the compliance limit for measurements. Typically, compliance values are defined for each input in a setup. You can refer all of the compliance values in a model to one variable defined at the model level by creating a compliance variable for the model. This compliance variable can be modified to change the compliance values for all inputs in the model.

The Model Variables folder contains the names and values of all variables that are global to the model.

To view the model variables, select **Model Variables**.

Click Model Variables to view current values



Name	Value
WD	512.6n
AreaCap1	10n
PerimCap1	400u
AreaCap2	300p
PerimCap2	206u
POLARITY	NMOS
OPEN_RES	1.0E12

Add new variable in empty field

NOTE To view system defaults from the Model Variables folder, select the System Variables button.

To define a new variable:

1 Type the variable name in the first empty *Name* field row.

2 Press Tab to move the cursor to the *Value* field. Type a value for the variable.

3 Press Enter to accept the value and open a new row.

NOTE Do not attempt to enter an expression when editing model variables. Expressions in variables are not evaluated when the variables are referenced. If a variable is used in a numeric expression, enter a single number only.

Do not enter the name of a variable table variable into the value field. The value of the variable is not evaluated when the variable is referenced.

Defining DUT Options

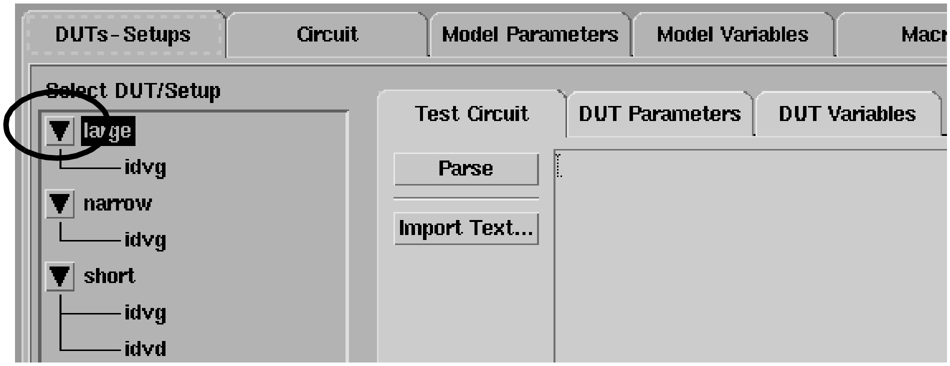
IC-CAP models usually contain several DUTs. DUTs contain groups of setups that have a similar physical connection to the device. Each DUT contains its own DUT parameter set and test circuit. If two setups require differences in either of these areas, you must define a different DUT for each.

Before you can measure a device, you must make any needed changes to the DUT options. The DUTs-Setups folder displays the DUTs and setups in the model. Three folders are available for defining DUT options.

To view the DUT options and make a device active, select **DUTs-Setups**. Then select the DUT. If necessary, display the list by clicking the arrow button.

Click DUTs-Setups to select the DUT

Click the arrow to see the setups



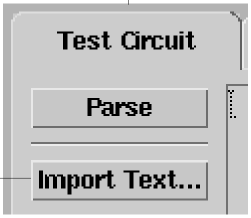
Accelerator Pop-Up Menus

In the DUTs/Setups tree structure, you can view pop-up menus that make it easier for you to perform actions on DUTs and Setups. To view the menus, right-click with your mouse pointer over a DUT or Setup name. The menus for a DUT and a Setup are different. The commands available in the menus enable you to perform selected actions that are allowed for the DUT/Setup. These are the same actions that are available in the Model window's main menu bar and folders. The pop-up menus save time by enabling you to perform an action on a DUT or Setup while avoiding the time required to display the folder contents in the work area.

As you work through this example, try also using these pop-up menus to perform the actions. The contents displayed in the Model window's work area depends on the DUT/Setup folder currently displayed in the work area, and which pop-up menu item you right-click on. When you right-click on a DUT or Setup, it is selected just as if you left-clicked on it. For a DUT or Setup folder displayed in the work area, if you perform an action from the pop-up menu normally associated with a different folder, the associated folder is not displayed in the work area. If a folder is currently shown, and you right-click on a different DUT or Setup, the current folder closes so there is no inconsistency between the selected DUT or Setup and the work area.

Defining a DUT Test Circuit

You can use the Test Circuit folder to attach additional circuitry between the device or circuit being modeled and the measurement hardware. For example, an OpAmp may need to be connected in different configurations for different measurements.



Click Test Circuit to attach additional circuitry

Click to reuse a circuit text file

NOTE

For complete information on defining Test Circuits, refer to Chapter 6, "Simulating."

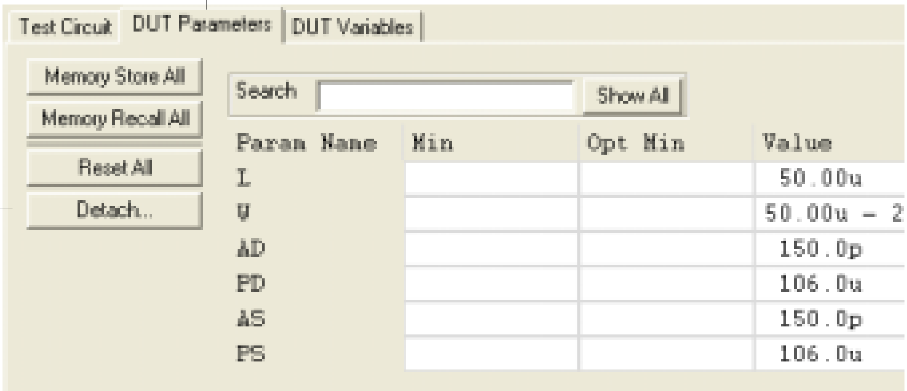
Defining DUT Parameters

You can edit the values of parameters that change from one DUT to another (such as, channel length and width for a MOSFET). In addition, you can set limits by defining minimum and maximum values for each parameter. If a parameter value is outside its limits, it will be clamped to its minimum or maximum value.

After editing the values, you can temporarily store the current set of parameter values (*Memory Store All*), recall stored parameter values (*Memory Recall All*), apply the changes to the circuit definition (*Update Circuit*), reset the default values specified in the circuit definition (*Reset All*).

To view the DUT parameters, select **DUT Parameters.**

Click DUT Parameters to view current values



Param Name	Min	Opt Min	Value
L			50.00u
W			50.00u - 2
AD			150.0p
PD			106.0u
AS			150.0p
PS			106.0u

For frequent editing, detach table

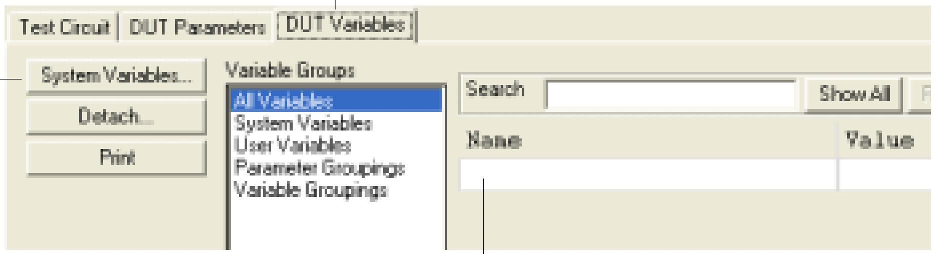
Defining DUT Variables

The DUT Variables folder contains the names and values of all variables that are global to the DUT.

To view the DUT variables, select **DUT Variables**.

Click DUT Variables to view current values

Click to view system variables



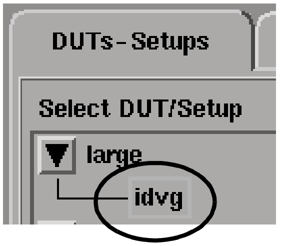
Add new variable in empty field

NOTE To view system defaults from the DUT Variables folder, select the System Variables button.

Defining Setup Options

You can define setup options from the DUTs-Setups panel. A setup contains definitions for inputs, outputs, instrument options, measuring and simulating, and results.

To view and edit a setup, select a setup from the DUTs-Setups panel.

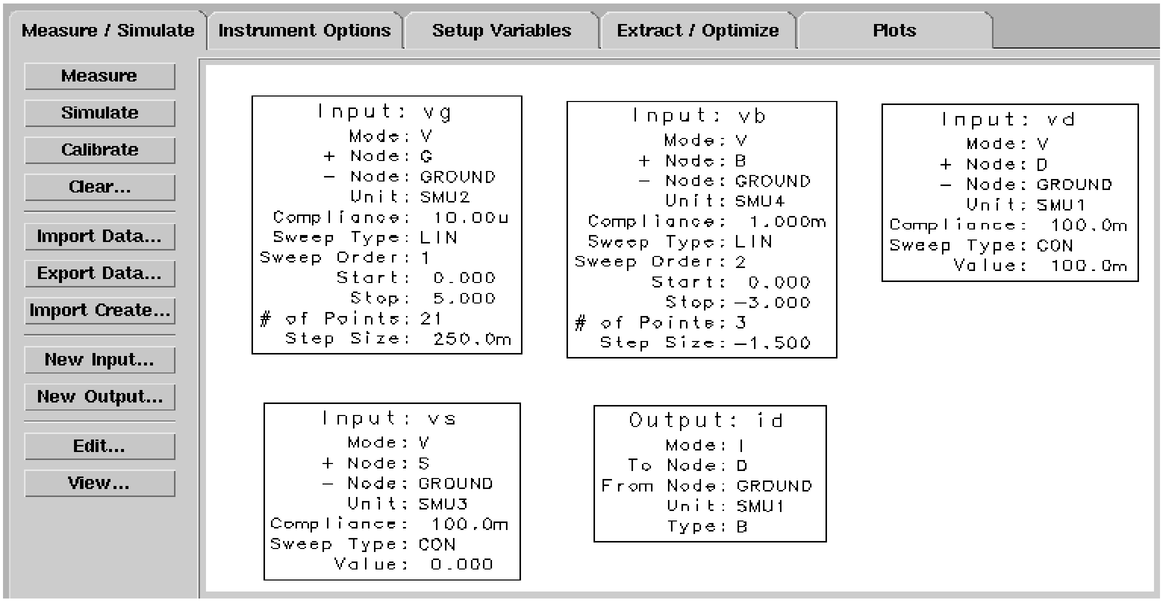


Select setup

NOTE

To detach the setup window for convenient editing, click on *Detach* below the DUT / Setup tree.

The Measure/Simulate folder opens so you can define the inputs and outputs for a particular setup.



```
       Input: vg
         Mode: V
       + Node: G
       - Node: GROUND
         Unit: SMU2
   Compliance:  10.00u
   Sweep Type: LIN
  Sweep Order: 1
        Start:  0.000
         Stop:  5.000
  # of Points: 21
    Step Size:  250.0m
```

```
       Input: vb
         Mode: V
       + Node: B
       - Node: GROUND
         Unit: SMU4
   Compliance:  1.000m
   Sweep Type: LIN
  Sweep Order: 2
        Start:  0.000
         Stop: -3.000
  # of Points: 3
    Step Size: -1.500
```

```
       Input: vd
         Mode: V
       + Node: D
       - Node: GROUND
         Unit: SMU1
   Compliance:  100.0m
   Sweep Type: CON
        Value:  100.0m
```

```
       Input: vs
         Mode: V
       + Node: S
       - Node: GROUND
         Unit: SMU3
   Compliance:  100.0m
   Sweep Type: CON
        Value:  0.000
```

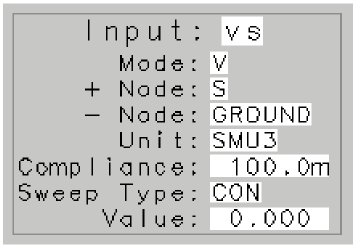
```
     Output: id
        Mode: I
     To Node: D
   From Node: GROUND
        Unit: SMU1
        Type: B
```

Viewing or Editing Input or Output

Setup input and output data are listed in a table (sometimes called a *tile*). You can edit input and output in one of two ways:

- Edit directly in the table. Click or double-click a field and type the new data.

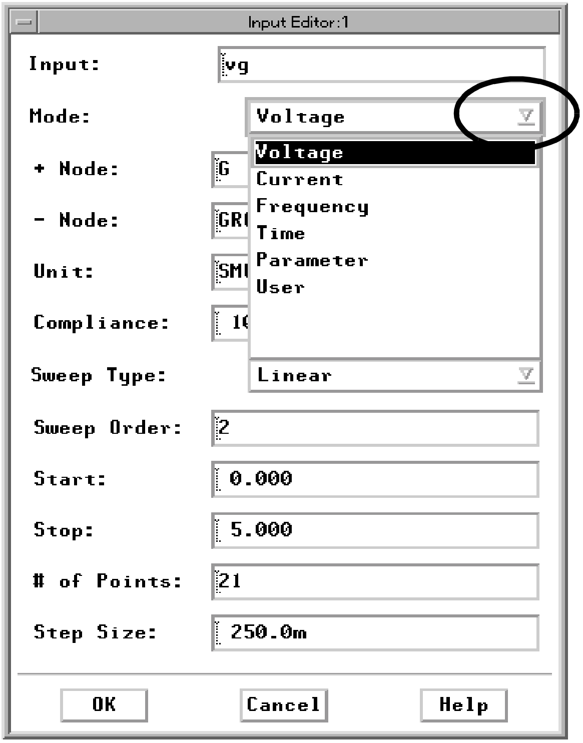
Edit directly in the table



```
       Input: vs
         Mode: V
       + Node: S
       - Node: GROUND
         Unit: SMU3
   Compliance:  100.0m
   Sweep Type: CON
        Value:  0.000
```

- Use the Input or Output Editor. To open the Input or Output Editor, first select the table and then click **Edit**. Edit the fields and click **OK.**

Click to open the Input or Output Editor



Input: vg

Mode: Voltage

+ Node: G

- Node: GR(

Unit: SMI

Compliance: 10

Sweep Type: Linear

Sweep Order: 2

Start: 0.000

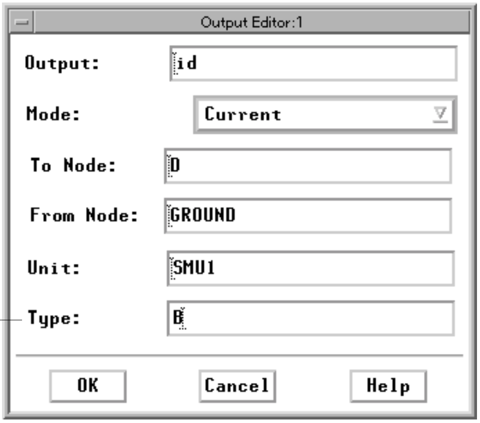
Stop: 5.000

# of Points: 21

Step Size: 250.0m

Click arrow for list of values

Output parameters can be derived from Measured data, Simulated data, or Both.



Output Editor:1

Output: id

Mode: Current

To Node: D

From Node: GROUND

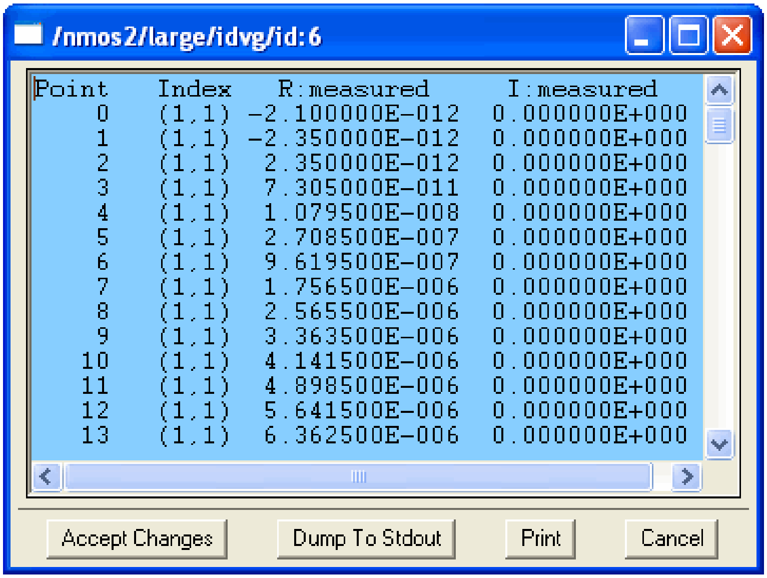
Unit: SMU1

Type: B

Type field indicates output parameter type: Measured, Simulated, or Both

To view the raw data for an input or output, select one or more tables then click **View**.

Click to view raw data.



/nmos2/large/idvg/id:6

```
Point     Index    R:measured      I:measured
    0     (1,1) -2.100000E-012  0.000000E+000
    1     (1,1) -2.350000E-012  0.000000E+000
    2     (1,1)  2.350000E-012  0.000000E+000
    3     (1,1)  7.305000E-011  0.000000E+000
    4     (1,1)  1.079500E-008  0.000000E+000
    5     (1,1)  2.708500E-007  0.000000E+000
    6     (1,1)  9.619500E-007  0.000000E+000
    7     (1,1)  1.756500E-006  0.000000E+000
    8     (1,1)  2.565500E-006  0.000000E+000
    9     (1,1)  3.363500E-006  0.000000E+000
   10     (1,1)  4.141500E-006  0.000000E+000
   11     (1,1)  4.898500E-006  0.000000E+000
   12     (1,1)  5.641500E-006  0.000000E+000
   13     (1,1)  6.362500E-006  0.000000E+000
```

NOTE

You can enable the Input/Output Finder mode in the Measure/Simulate folder. The Input/Output Finder mode adds an area that lists the Setup's inputs and outputs. When enabled, you can quickly bring data into view by selecting one or more of the inputs and outputs in the list.

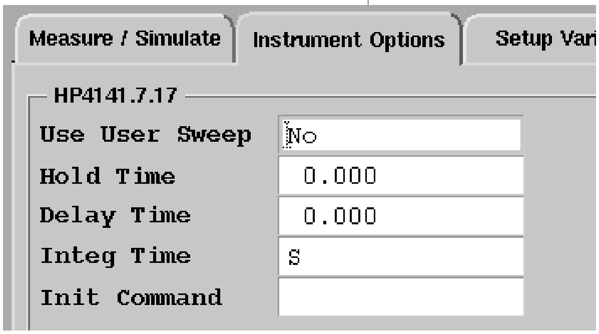
To enable the mode for a Setup, open the Setup Variables folder. Enter the variable name SHOW_INPUT_OUTPUT_FINDER in the Name field, then enter yes, true, or a non-zero integer in the Value field. If you change the variable's setting after opening the Model, you must close (File > Close), then reopen, the Model window for the new setting to take effect.

Viewing or Editing Instrument Options

The Instrument Options table is available after the hardware has been set up.

To edit instrument options, select the **Instrument Options** folder.

Click to view available instrument options



HP4141.7.17

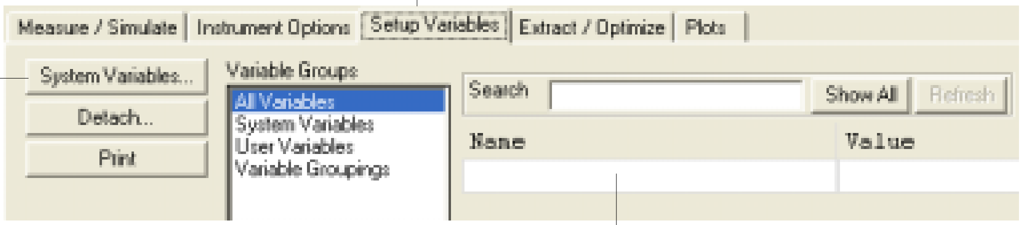
Use User Sweep	No
Hold Time	0.000
Delay Time	0.000
Integ Time	S
Init Command	

For details on Hardware Setup, refer to Chapter 5, “Making Measurements.”

Defining Setup Variables

The Setup Variables folder contains the names and values of all variables that are global to the Setup.

To view the Setup variables, select **Setup Variables**.



Click Setup Variables to view current values

Click to view system variables

Add new variable in empty field

Measuring the Device

The measurement on the devices provides sufficient information for extracting a full set of model parameters. To avoid errors, measuring devices, and extracting parameters must be performed in a specified order.

NOTE

For detailed procedures on measurement order, refer to the tutorial for your model.

- To initiate the measurement using the active setup, select the setup and click **Measure Setup**.
- To initiate the measurement using all setups in the active DUT, select the DUT and click **Measure DUT**.

Performing a Simulation

You can determine the accuracy of current parameter values by performing a simulation and comparing the results to the measured data.

- To perform a simulation on the active setup, select the setup and click **Simulate Setup**.
- To perform a simulation on all setups in the active DUT, select the DUT and click **Simulate DUT**.

Extracting Model Parameters

Agreement between two sets of data is not always good prior to extraction. IC-CAP modules provide standard device extractions, as well as macros for performing an extraction. If the model you are using does not have macros available, you can set the criteria in the Transforms folder.

Viewing or Editing Transforms

You can choose from a large variety of functions to use for extracting model parameters in a Transform.

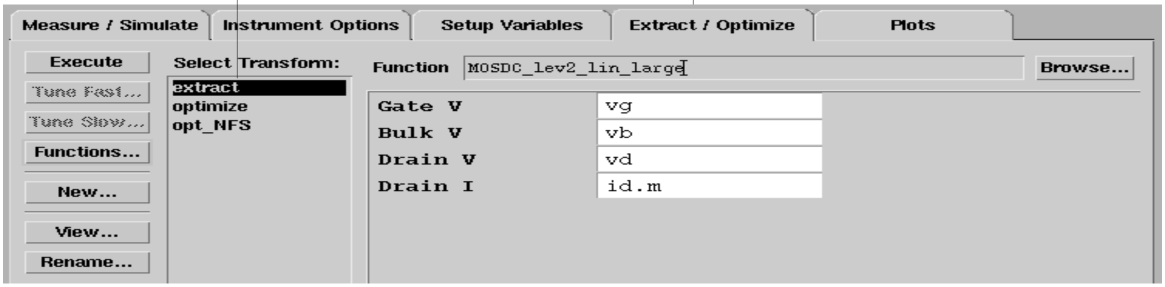
To view or edit transforms:

1 Select a setup.

2 In the Extract/Optimize folder, choose the **extract** transform.

Select transform

Click Extract/Optimize to view transforms

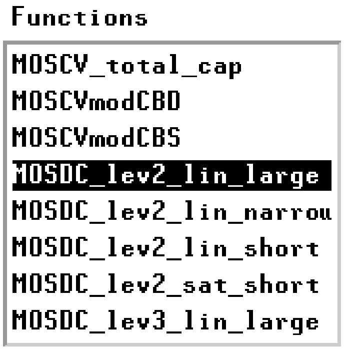


Function MOSDC_lev2_lin_large

Gate V	vg
Bulk V	vb
Drain V	vd
Drain I	id.m

3 Click **Browse** to view the Function list. Browse...

4 Select a Function from the list.



Functions

MOSCV_total_cap
MOSCVmodCBD
MOSCVmodCBS
MOSDC_lev2_lin_large
MOSDC_lev2_lin_narrow
MOSDC_lev2_lin_short
MOSDC_lev2_sat_short
MOSDC_lev3_lin_large

Double-click to select the function

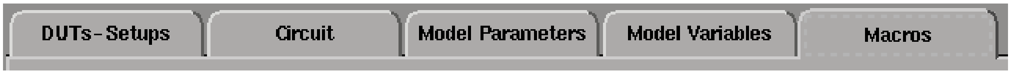
Setting Initial Values Using a Macro

The initial values of the process-dependent parameters for the extraction are entered using a macro.

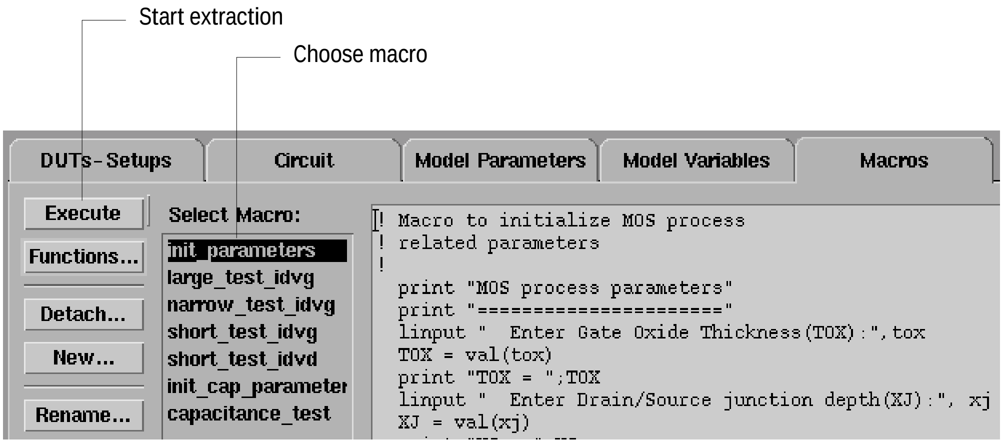
To run a macro:

1 In the Model folder, select **Macros**.

Click Macros to set initial values



2 In the Select Macro list, choose **init_parameters** and click **Execute**.



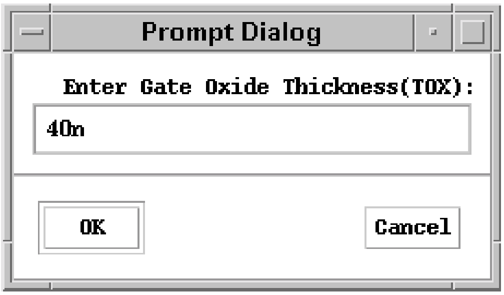
```
! Macro to initialize MOS process
! related parameters
!
  print "MOS process parameters"
  print "======================"
  linput "  Enter Gate Oxide Thickness(TOX):",tox
  TOX = val(tox)
  print "TOX = ";TOX
  linput "  Enter Drain/Source junction depth(XJ):", xj
  XJ = val(xj)
```

NOTE

You can also execute macros from the Model window’s main menu. Select **Macros > Execute**, then select the macro you want to run.

3 In the series of prompts, enter the initial values.

In the example, the initial values are:
TOX = 40n
XJ = 0.2u
LD = 0.4U
RS = RD = 10



Enter Gate Oxide Thickness(TOX):

40n

Performing an Extraction

You can extract all extraction transforms in the active setup or extract all setups in the active DUT.

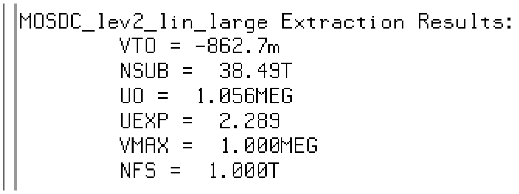
- To perform an extraction on all extraction transforms in the active setup, select the setup and click **Extract Setup**.
- To perform an extraction on all extraction transforms in all setups in the active DUT, select the DUT and click **Extract DUT**.

When you extract all setups in the active DUT, the extractions are performed in the left-to-right order listed in the setup. This order is usually critical to proper extraction performance.

NOTE

For detailed procedures on extraction order, refer to the tutorial for your model.

Typically, extractions are completed instantly. The newly extracted model parameter values are listed in the IC-CAP Status window and are placed in Model Parameters.



```
MOSDC_lev2_lin_large Extraction Results:
        VTO = -862.7m
        NSUB =  38.49T
        UO =  1.056MEG
        UEXP =  2.289
        VMAX =  1.000MEG
        NFS =  1.000T
```

Simulating with Extracted Parameters

After extracting the model parameters, you can perform a simulation with the extracted parameters and compare the results to the measured data.

- To perform a simulation on the active setup, select the setup and click **Simulate Setup**.
- To perform a simulation on all setups in the active DUT, select the DUT and click **Simulate DUT**.

Repeat the simulation procedure for each DUT in the model. You can observe the differences in output that result from changes to various model parameters. For example, change the value of one or more parameters. Then run a simulation and view the changes in the plot.

Optimizing the Model Parameters

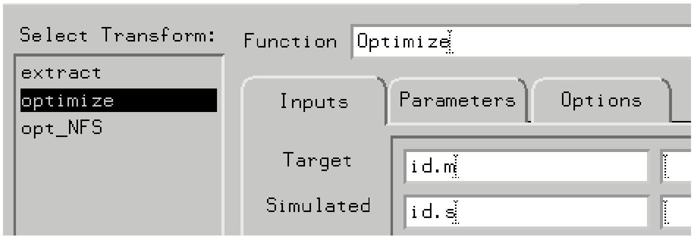
To achieve greater accuracy, you can optimize the model parameters. This optional step obtains the best possible fit between measured and simulated data by altering the parameter values iteratively until the difference between the data sets is minimized. Since repeated simulations are required, optimization is more time-consuming than an extraction.

To perform an optimization:

1 Select a setup.

2 In the Extract/Optimize folder, select the **optimize** transform.

3 Define the Inputs, Parameters, and Options for the selected transform. For further details, refer to Chapter 9, “Using Transforms and Functions.”



Select Transform:

extract
optimize
opt_NFS

Function Optimize

Target id.m
Simulated id.s

4 Execute the optimization:

- To perform an optimization on the active setup, select the setup and click **Optimize Setup**.
- To perform an optimization on all setups in the active DUT, select the DUT and click **Optimize DUT**.

Measuring the Remaining Devices

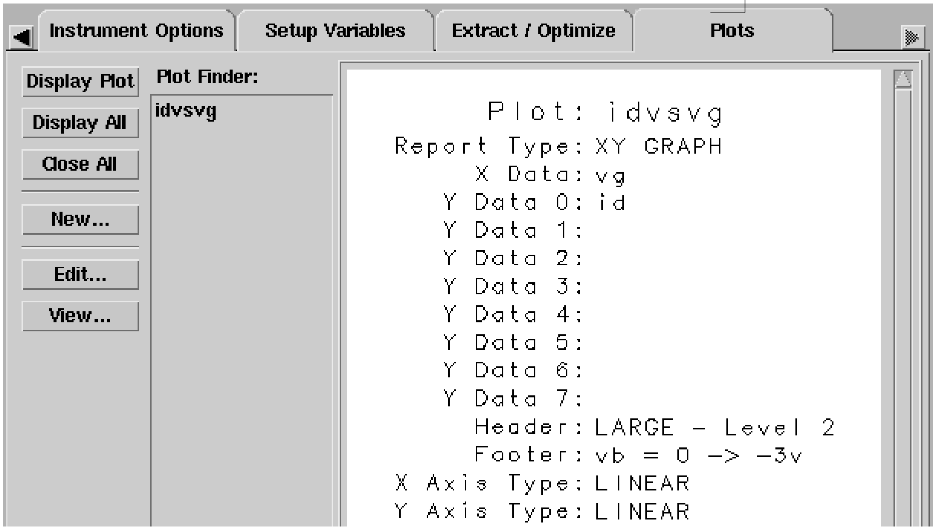
To complete the characterization, you must measure the remaining devices by repeating these steps for the other DUTs contained in the model.

To measure all setups in a DUT with multiple setups, click **Measure DUT**.

Viewing the Results

You can view the results of the measured data for a setup by selecting the Plots folder in the Setup window. The Plot Finder list contains all of the plots available for a Setup. If there are multiple plots, select the one you want to view. You can view multiple plots by pressing the Ctrl key while selecting the plot names in the list.

Click Plots to display results



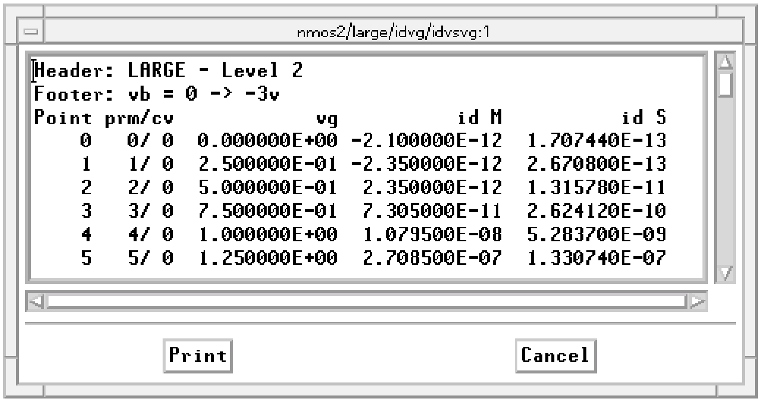
Plot Finder:

idvsvg

```
       Plot: idvsvg
Report Type:XY GRAPH
     X Data:vg
   Y Data 0:id
   Y Data 1:
   Y Data 2:
   Y Data 3:
   Y Data 4:
   Y Data 5:
   Y Data 6:
   Y Data 7:
     Header:LARGE - Level 2
     Footer:vb = 0 -> -3v
X Axis Type:LINEAR
Y Axis Type:LINEAR
```

To see the measured data, click **View.**

View... Click to view measured data results

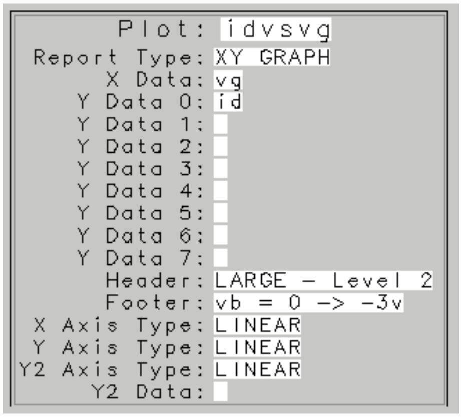


```
Header: LARGE - Level 2
Footer: vb = 0 -> -3v
Point prm/cv          vg          id M          id S
    0   0/ 0  0.000000E+00 -2.100000E-12  1.707440E-13
    1   1/ 0  2.500000E-01 -2.350000E-12  2.670800E-13
    2   2/ 0  5.000000E-01  2.350000E-12  1.315780E-11
    3   3/ 0  7.500000E-01  7.305000E-11  2.624120E-10
    4   4/ 0  1.000000E+00  1.079500E-08  5.283700E-09
    5   5/ 0  1.250000E+00  2.708500E-07  1.330740E-07
```

You can edit the setup for the data table in one of two ways:

- Edit directly in the table. Click or double-click a field and type the new data

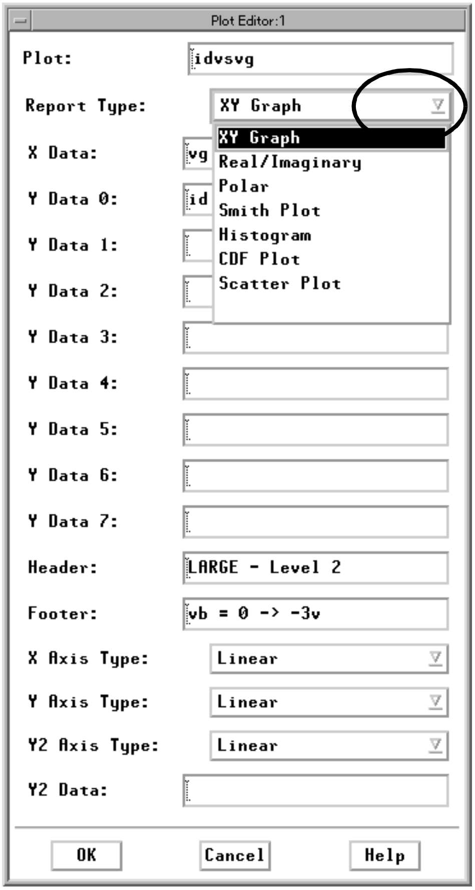
Edit directly in the table



```
        Plot: idvsvg
 Report Type: XY GRAPH
      X Data: vg
    Y Data 0: id
    Y Data 1:
    Y Data 2:
    Y Data 3:
    Y Data 4:
    Y Data 5:
    Y Data 6:
    Y Data 7:
      Header: LARGE - Level 2
      Footer: vb = 0 -> -3v
 X Axis Type: LINEAR
 Y Axis Type: LINEAR
Y2 Axis Type: LINEAR
     Y2 Data:
```

- Use the Plot Editor. To open the Plot Editor, first select the table and then click **Edit** (or double-click on the table). Edit the fields and click **OK.**

Click to open the Plot Editor



Plot: idvsvg

Report Type: XY Graph

X Data: vg

Y Data 0: id

Y Data 1:

Y Data 2:

Y Data 3:

Y Data 4:

Y Data 5:

Y Data 6:

Y Data 7:

Header: LARGE - Level 2

Footer: vb = 0 -> -3v

X Axis Type: Linear

Y Axis Type: Linear

Y2 Axis Type: Linear

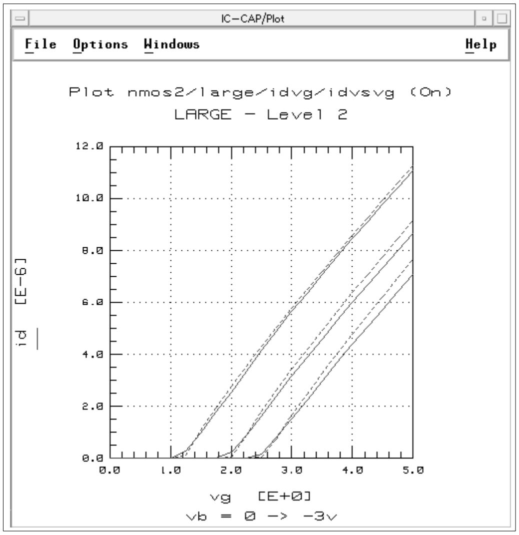
Y2 Data:

Click arrow for list of values

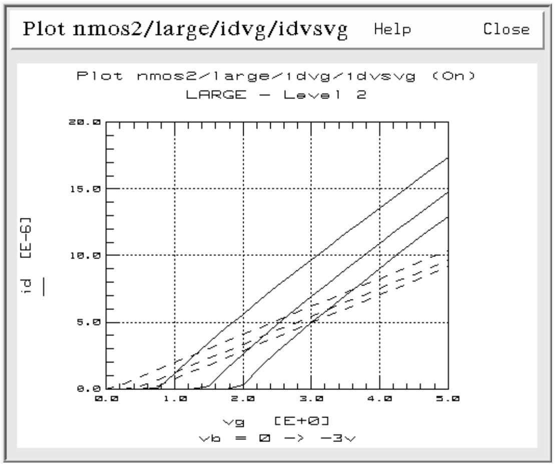
For a complete description of the editor fields, refer to Chapter 10, “Printing and Plotting.”

To see the plot, click **Display Plot** or **Display All**. The IC-CAP/Plot window opens. The following figure shows examples of results.

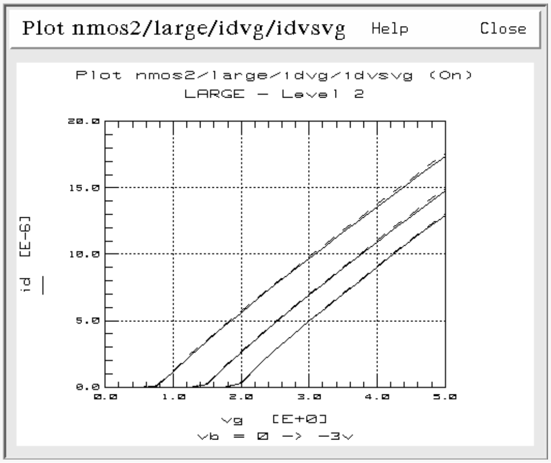
Measured Data



Measured and Simulated Data



Measured and Simulated Data
After Extraction



Measured and Simulated Data
After Optimization

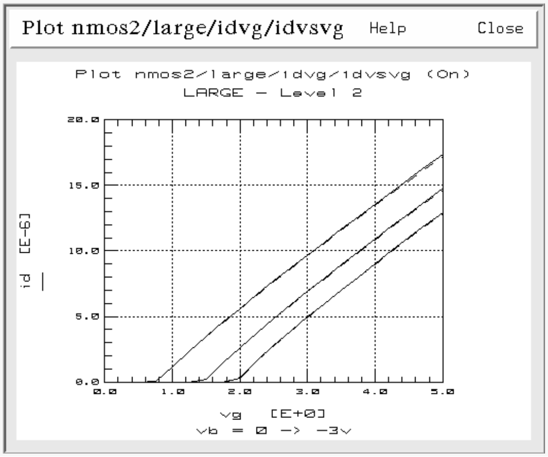
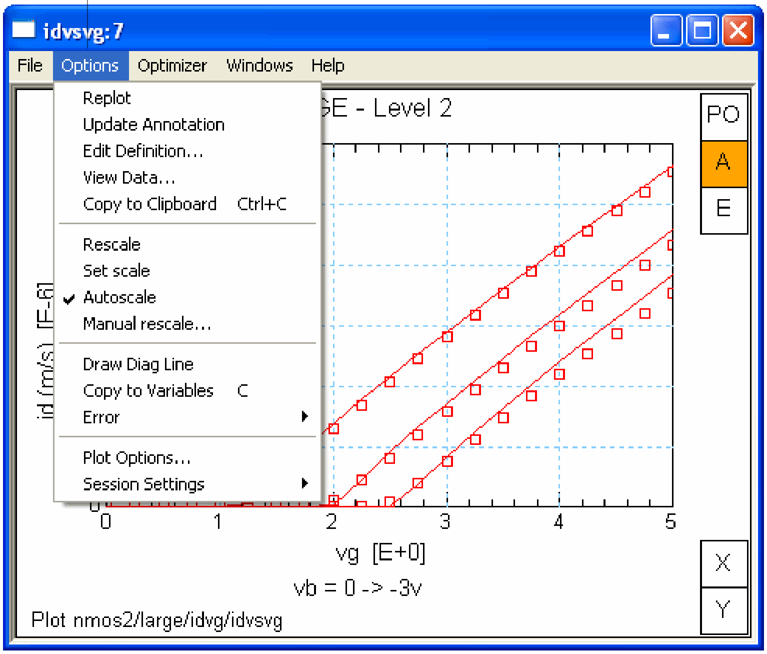


Figure 9 Examples of Displayed Results

You can define display options by selecting *Options* in the plot window. For details, refer to Chapter 10, “Printing and Plotting.”

Select Options then choose from the menu



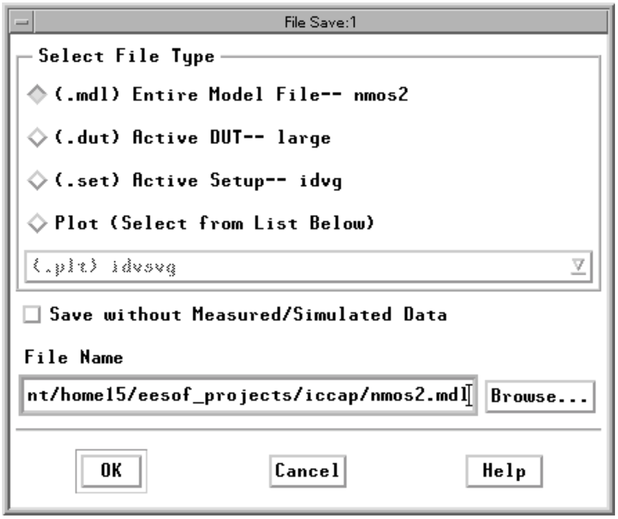
Saving the Model

When you have completed the characterization process, you can save the model to a file. You have several options for saving.

To save the model to a file:

1 Select **Save**.

2 Choose saving options.



File Save:1

Select File Type

◆ (.mdl) Entire Model File-- nmos2

◇ (.dut) Active DUT-- large

◇ (.set) Active Setup-- idvg

◇ Plot (Select from List Below)

(.plt) idvsvg

□ Save without Measured/Simulated Data

File Name

nt/home15/eesof_projects/iccap/nmos2.mdl

3 In the File Name field, type a unique name (such as, *mybjt_npn.mdl*).

4 Click **OK**.

Exiting IC-CAP

To exit IC-CAP, select **File > Exit** in the IC-CAP/Main window. This opens the Save As dialog box to give you an opportunity to save changes to models before exiting the program. Click on the **Exit** button to automatically close all open windows and terminate the program.

4
Creating and Modifying Models

In IC-CAP, all of the information required to characterize a particular device or circuit is contained in a model. With a variety of model configurations available, you can easily create new models by copying and/or modifying existing models. The changes may be structural (such as the addition or removal of setups) or may involve modification of table values only.

NOTE

Before starting this chapter, you must configure the necessary source and measuring instruments for the IC-CAP system. For information on configuring the system, refer to the *Installation Guide*.

Model Components

The IC-CAP model components are shown in the following figure. A model consists of these components:

- **Circuit** Description of the circuit used to represent the actual device.
- **Model Parameters** Values used in simulating the circuit.
- **Model Variables** User-defined variables that are global to a model.
- **DUTs** Representations of different physical devices or measurement configurations.
- **Macros** Combinations of the available menu functions used to create complex operations.

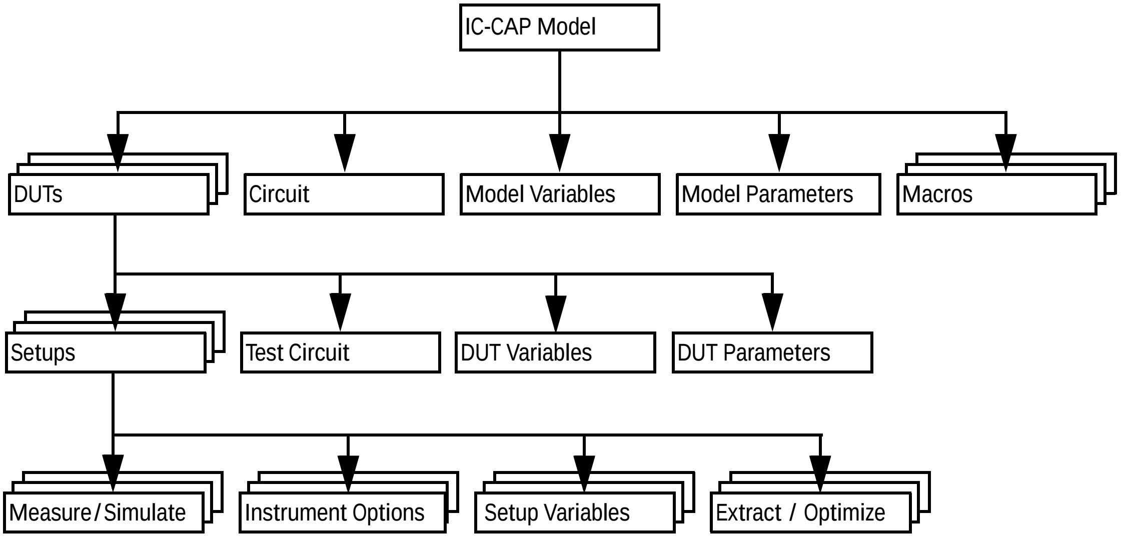


Figure 10 IC-CAP Model Components

DUTs and Setups

Each DUT contains:

Test Circuit Description of the circuit used in a particular DUT.

DUT Parameters Values for parameters specific to a DUT.

DUT Variables User-defined variables that apply only to a DUT.

Setups Define measurements and simulations for a DUT.

Each setup contains:

Inputs A definition of the input stimuli.

Outputs A definition of the types of responses to monitor.

Transforms Specifications for different data manipulations, such as mathematical functions and extractions.

Plots Data display, in a variety of formats.

Setup Variables User-defined variables that apply only to a setup.

Instrument Options Instrument options control the state of the hardware for a setup.

Defining Model Components

When you create an IC-CAP model, you must define the components. You can create a new model and then describe its physical and electrical characteristics or you can load and modify an existing model. Some components, such as user-defined variables and test circuitry, are optional.

The general procedure for creating a model is:

- Add a model to the work area.
- Define variables that are global to the model.
- Define the device or circuit to be characterized.
- View and modify the parameter values to be used in the simulations.
- Add new (or modify existing) DUTs.
  - If necessary, define a test circuit.
  - Edit device parameters specific to the DUT
  - Define variables that are global to the DUT.
  - Add new (or modify existing) setups.
- For each setup, define the following:
  - Inputs to the device or circuit.
  - Outputs to monitor.
  - Variables global to the setup.
  - Values of instrument options for all instruments used in a setup.
  - Transforms, including mathematical functions and extractions.
  - Plots that specify graphical and tabular formats for the data.

NOTE

For an example, see “Example: Creating a New Model” on page 162.

Adding a Model to the Work Area

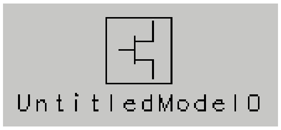
You can add a model to the work area in one of two ways:

- Open an empty template
- Open an existing model

To open an empty template:

1 In the IC-CAP/Main window, click **Create Model** on the toolbar.

An icon representing a new untitled model appears in the IC-CAP/Main window.



2 To open the Model window, double-click the model icon.

To open an existing model:

1 In the IC-CAP/Main window:
   - Click **Open** to open a model in the current directory
   - Click **Examples** to open a model provided with the program

2 In the dialog box, double-click **model_files** to see the list of model directories, or double-click **demo_features** to see the list of demonstration directories.

3 Double-click on a directory name, then click on a model name in the Files list to select a model file.

4 Choose **OK** to accept the selection and close the dialog box.

An icon of the model file displays in the IC-CAP/Main window and the Model window opens. If the model displays as an icon only, double-click the icon to open the Model window.

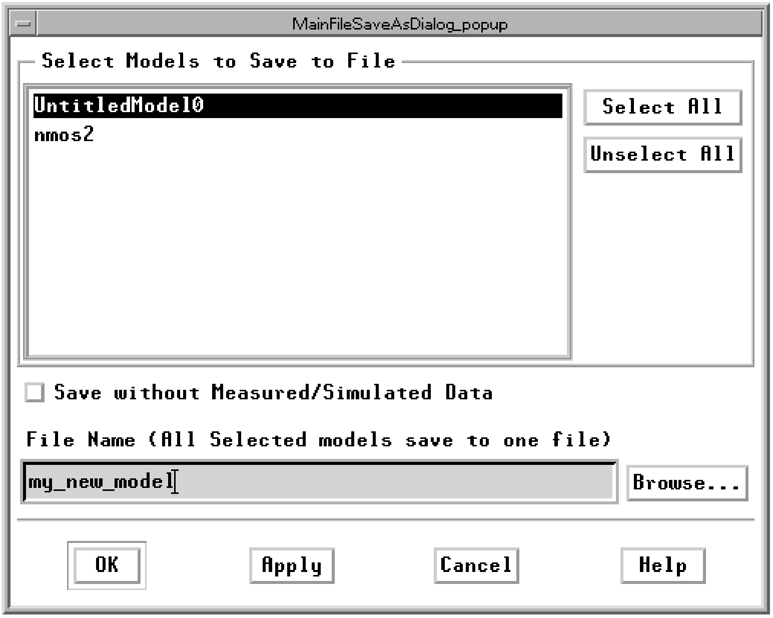
Saving Model Files

Before modifying a new model, you can save the model file to different name.

To save model data from the IC-CAP/Main window:

1 Select a model icon in the IC-CAP/Main window, then select **Save As**.

2 Select the open models you want to save.



MainFileSaveAsDialog_popup

Select Models to Save to File

UntitledModel0
nmos2

□ Save without Measured/Simulated Data

File Name (All Selected models save to one file)

my_new_model

Select files to save

Enter a unique filename

3 In the File Name field, enter a unique filename and click **OK**. The new model name appears in the directory list.

If a file was saved previously by that name, an information dialog box opens.

- To replace the existing model by that name, choose **Yes**.
- To avoid replacing the existing model by that name, choose **No** and enter a new file name to save the model to different name.

You can use the same name as the supplied file, as long as you save the model to a different directory. For details on changing directory paths, refer to “Changing Directories” on page 56.

Copying Parts of a Model

You can create a set of models, or parts of a model, that are similar to each other by using the Copy command. You can copy any part of the model that you can select. For example:

- To copy the circuit definition, parameter set, or variable table, open the Circuit, Model Parameters, or Model Variables tab.
- To copy an entire DUT, select the DUT.
- To copy an entire setup, select the Setup.
- To copy an input, output, transform, or other table in a setup, select the table (by clicking on the outside ruling) or tab.

After copying, you can place the copied parts into the new model and modify the copied parts.

To copy parts of an existing model:

1. Open the model and select the parts you want to copy.
2. Select **Edit > Copy [part type]** or click **Copy** on the toolbar.

To place the copied parts in the new model:

1. Place the cursor in the folder where you want to put the copied parts.
2. Select **Edit > Paste [part type]** or click **Paste** on the toolbar.

NOTE

To copy an entire DUT or Setup in a model, right-click on the DUT or Setup you want to copy to view the pop-up menu, then select **Edit > Copy**. To paste a Setup into another DUT, right-click on the DUT, then select **Edit > Paste**. To paste a copied DUT into another model use the directions in the procedure above.

Specifying the Circuit Definition

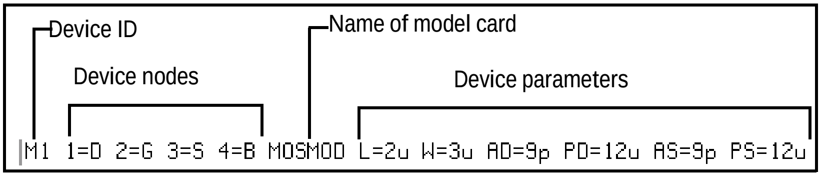
The circuit definition specifies the parameters to be extracted and the representation of the model in the simulator. The parameter values act as defaults in IC-CAP. In the supplied model files, these values are usually set to SPICE defaults. However, you may want to specify values for the devices or circuits that you plan to characterize, since extractions and optimizations run faster if starting parameter values are closer to the final values. For a complete description of the types of circuits that can be defined, refer to Chapter 6, “Simulating.”

Example The example circuit is a supplied model, *nmos2.mdl*, that extracts parameters for the Level 2 N-channel UCB MOSFET transistor.

```
.OPTIONS ucb
M1 1=D 2=G 3=S 4=B MOSMOD L=2u W=3u AD=9p PD=12u AS=9p PS=12u
.MODEL MOSMOD NMOS
+ LEVEL = 2
+ UO = 600
+ VTO = 0
+ NFS = 0
+ TOX = 100n
+ NSUB = 1e15
+ UCRIT = 10.00K
+ UEXP = 0
+ VMAX = 1MEG
+ RS = 0
+ RD = 0
+ XJ = 0
+ LD = 0
+ DELTA = 0
+ NEFF = 1.00
+ NSS = 0
+ CGSO = 0
+ CGDO = 0
+ CGBO = 0
+ CBD = 0
+ CBS = 0
+ CJ = 0
+ MJ = 0.5
+ CJSW = 0
+ MJSW = 0.33
+ IS = 1.0E-14
+ PB = 0.8
+ FC = 0.5
```

The circuit definition begins with the Options definition (.OPTION). The definition lists the device to be measured, the device nodes, the name of the model card that contains the model parameters used by the device, and the device parameters. In the example

Circuit Options definition

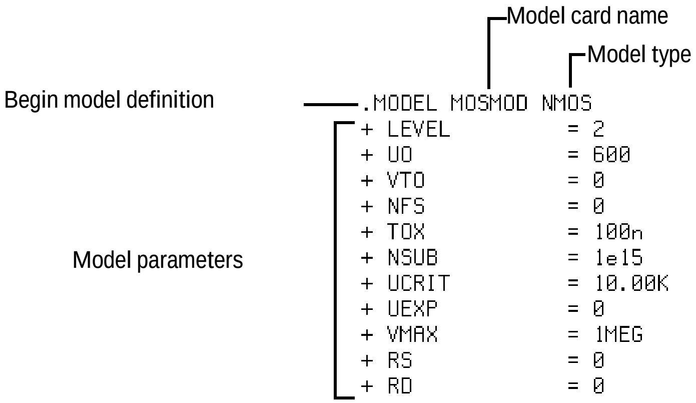


```
M1 1=D 2=G 3=S 4=B MOSMOD L=2u W=3u AD=9p PD=12u AS=9p PS=12u
```

- The first item identifies the device to be measured, *M1*. The character *M* indicates that the device is a MOSFET.
- Next, the device nodes are identified (1 = drain, 2 = gate, 3 = source, 4 = bulk (substrate).
- The next item, *MOSMOD,* is the name of the model card that contains the model parameters used by the *M1* device.
- The remaining items in the Options definition specify device parameters. For real devices, refer to the design specification for the values of device parameters.

The next section is the model card definition (.MODEL). The definition lists the name of the model card, the model type, and the contents of the model card. The contents of the model card consists of the parameters that are to be extracted (model parameters) and their assigned initial values. In the example

Circuit Model definition

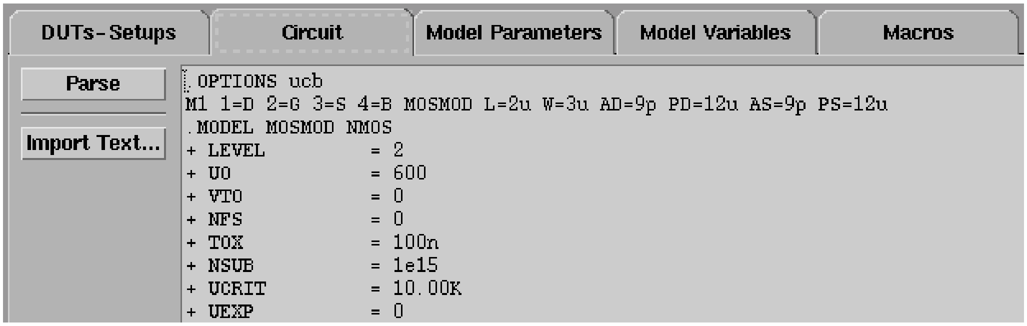


```
.MODEL MOSMOD NMOS
+ LEVEL          = 2
+ UO             = 600
+ VTO            = 0
+ NFS            = 0
+ TOX            = 100n
+ NSUB           = 1e15
+ UCRIT          = 10.00K
+ UEXP           = 0
+ VMAX           = 1MEG
+ RS             = 0
+ RD             = 0
```

- The name of the model card is identified (MOSMOD), then the model type N-channel (NMOS).
- The remaining lines are the contents of the model card. These consist of the parameters that are to be extracted (model parameters) and their assigned initial values. The leading + indicates that each parameter follows the preceding just as if all the parameters had been entered on a single line.

To specify the circuit definition:

1 In the Model window, click **Circuit** to open the folder.



```
.OPTIONS ucb
M1 1=D 2=G 3=S 4=B MOSMOD L=2u W=3u AD=9p PD=12u AS=9p PS=12u
.MODEL MOSMOD NMOS
+ LEVEL          = 2
+ UO             = 600
+ VTO            = 0
+ NFS            = 0
+ TOX            = 100n
+ NSUB           = 1e15
+ UCRIT          = 10.00K
+ UEXP           = 0
```

2 Edit the existing values by selecting the value and typing over it.

3 When you move the mouse pointer out of the window, the new circuit description is parsed automatically. This parse only occurs after a change has been made to the contents of the circuit description.

Syntax errors are reported in the IC-CAP/Status window and must be corrected. For example, if the wrong type of simulator is selected and syntax errors result from the parse, you must parse the circuit after the correct simulator is specified.

To parse the circuit definition manually, click **Parse**.

NOTE

You can copy the circuit definition from an existing model. Open the model you want to copy. Open the Circuit folder. Select **Edit > Copy Circuit**. In the new model, select **Edit > Paste Circuit**.

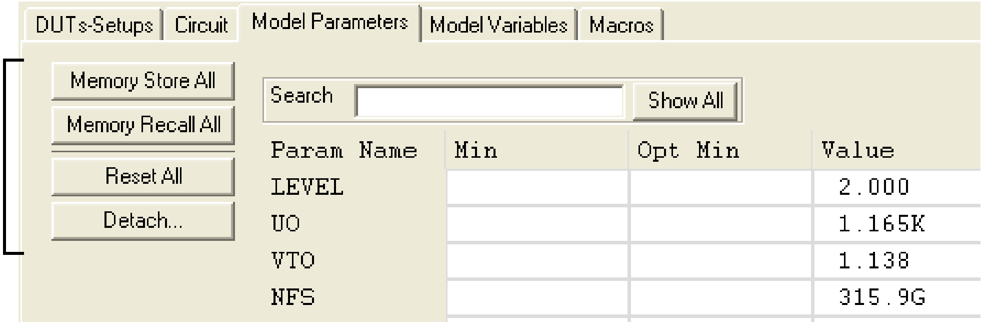
Implementing Macro Models

Macro modeling provides a powerful method for defining and simulating new models in IC-CAP. The approach is particularly useful when a complex topology is required. There are several ways to create macromodels. For example, enter the model in the form of a circuit definition and include any model component that is supported by SPICE. Another method is to enter new model equations using the dependent voltage and current sources, although the equations for these sources are limited to polynomial functions.

Specifying Model Parameters

The Model Parameters folder displays the current parameter values. These parameters are common to all DUTs for that model and are used in a simulation. When a new model is read, these values are set to the defaults defined in the circuit. The values change after extractions and optimization are performed.

To view the model parameters, select the **Model Parameters** folder.



Control value set selection

Param Name	Min	Opt Min	Value
LEVEL			2.000
UO			1.165K
VTO			1.138
NFS			315.9G

The Parameters table contains the same parameters as those specified in the *MODEL* definition of the Circuit. However, by editing the table, the parameter values can be set independently and can be specified as a real number or an equation. In addition, you can set limits by entering minimum and maximum values for each parameter. Parameter values that are outside their limits are clamped to their minimum or maximum values.

NOTE

The value fields can be set either to an actual number or to functions of other data available in IC-CAP. For example, if you enter RS as the value for the RD parameter, this parameter is always equal to the RS parameter.

To edit the model parameters:

1 In the Model window, select **Model Parameters**.

2 Edit the parameter value fields and select an action from the following:

  - To temporarily store the current set of parameter values, click **Memory Store All**.
  - To recall stored parameter values, click **Memory Recall All**.
  - To reset the default values specified in the circuit definition, click **Reset All**.

NOTE

You can copy a current set of parameter values from an existing model. Open the model you want to copy. Select the **Model Parameters** folder. Select **Edit > Copy Parameter Set**. In the new model, select **Edit > Paste Parameter Set**.

Setting Model Variables

Variables can be defined at the IC-CAP system level or at the model, DUT and setup levels. You can define as many variables as you need and use these variables for other functions, such as extractions.

The Model Variables folder contains the names and values of all variables that are global to the model.

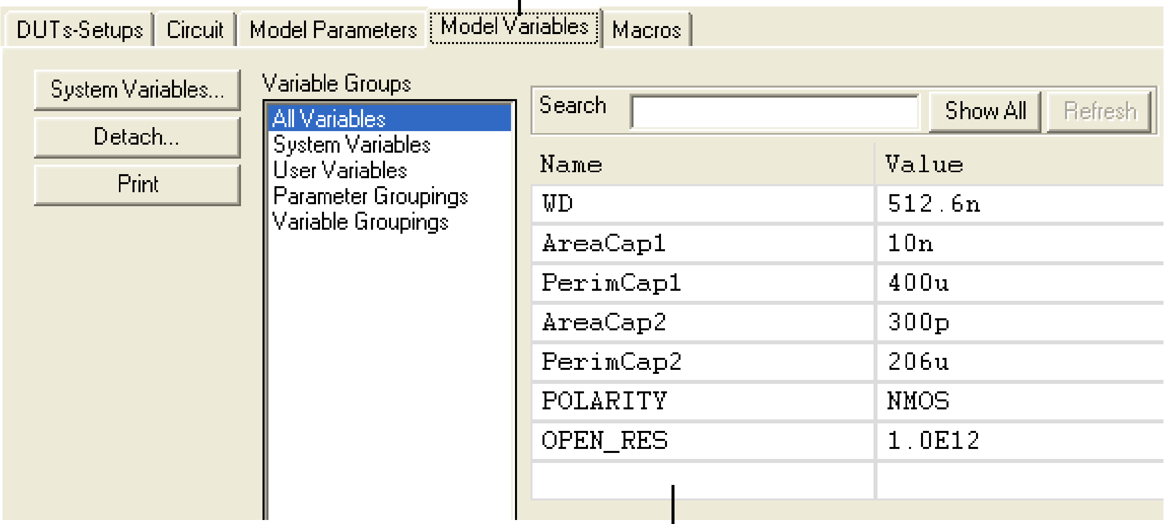
NOTE

Do not attempt to enter an expression when editing model variables. Expressions in variables are not evaluated when the variables are referenced. If a variable is used in a numeric expression, enter a single number only.

Do not enter the name of a system variable into the value field. The value of the system variable is not evaluated when the variable is referenced.

To view or edit the model variables:

1 In the Model window, select **Model Variables**.



Name	Value
WD	512.6n
AreaCap1	10n
PerimCap1	400u
AreaCap2	300p
PerimCap2	206u
POLARITY	NMOS
OPEN_RES	1.0E12

Click Model Variables to view current values

Add new variable in empty field

2 Edit the model variable fields by selecting the value and typing over it.

NOTE To view system defaults from the Model Variables folder, select the System Variables button.

To specify new model variables:

1 In the Model window, select **Model Variables**.

2 Enter the names and values for all model variables:

- Click the first empty row in the Name field and type a name for the variable.
- Press Tab to move to the Value field. Type a value for the variable.

- Press Enter to accept the value and open another row.

NOTE

You can copy the model variables from an existing model. Open the model you want to copy. Select the **Model Variables** folder. Select **Edit > Copy Variable Table**. In the new model, select **Edit > Paste Variable Table**.

Finding Model Parameters or Variables

Use the search box on the *Model Parameters* and *Model Variables* folders to find subgroups of parameters and variables. In addition, you can save any subgroup to a name for easy retrieval using the search box.

Using the Search Rules

In the search box, special characters are: *, <space>, and <comma>.

Handling rules:

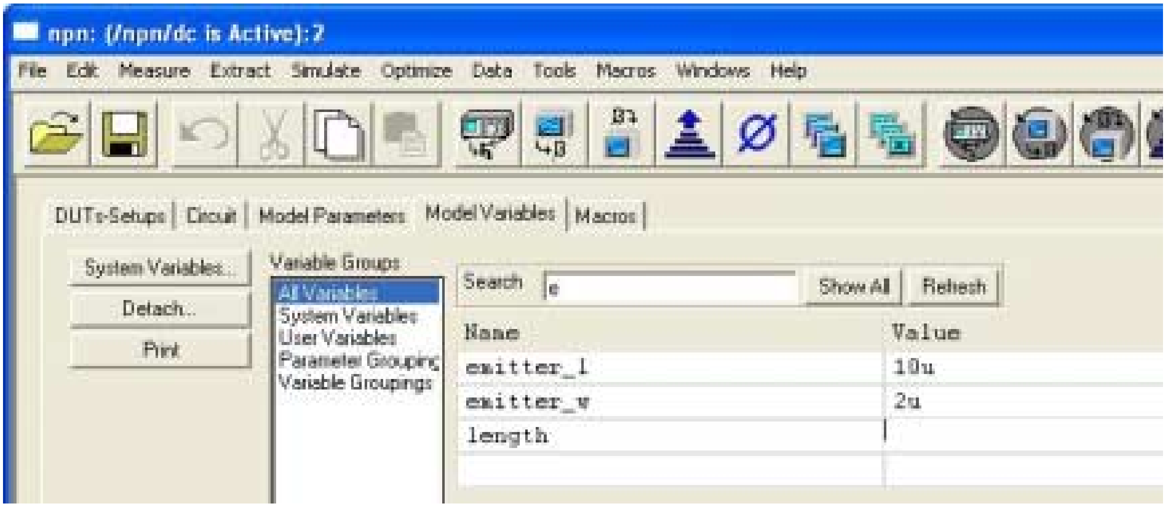
1. * means zero or more characters. An implied * will apply to the last cluster of letters.
2. <space> delimits clusters of letters but the implied * remains
3. ',' delimits clusters and terminates the implied *
4. [ ] is used for searching all elements in a ICCAP_ARRAY

Example	Description	Heading Rule
ABC	ABC* (implied trailing *)	1
ABC (trailing space)	ABC* (implied trailing *)	2
ABC DEF	ABC* and DEF* (implied * for both)	1,2
ABC* DEF	ABC* and DEF* (implied * on DEF)	1
ABC,	ABC (no more implied *)	3
ABC,DEF	ABC and DEF* (implied * on DEF)	1,3
ABC*, DEF	ABC* and DEF* (implied * on DEF)	1
*AB	*AB* (implied trailing *)	1
AB[]	all elements in ICCAP_ARRAY AB	4
AB[1][]	all elements in ICCAP_ARRAY AB[1]	4

Refresh button in variable page:

If a new variable is added to the current search results, the newly added variable will remain in current search results until the *Refresh* button is selected.

For example, insert a variable not begin with character *e*. In this example, *length* remains visible until the *Refresh* button is selected.



Search e

Name	Value
emitter_l	10u
emitter_w	2u
length	

Creating Parameter Groups

Use parameter groups to select parameters.

MDL file specific groups

Standard Groups:

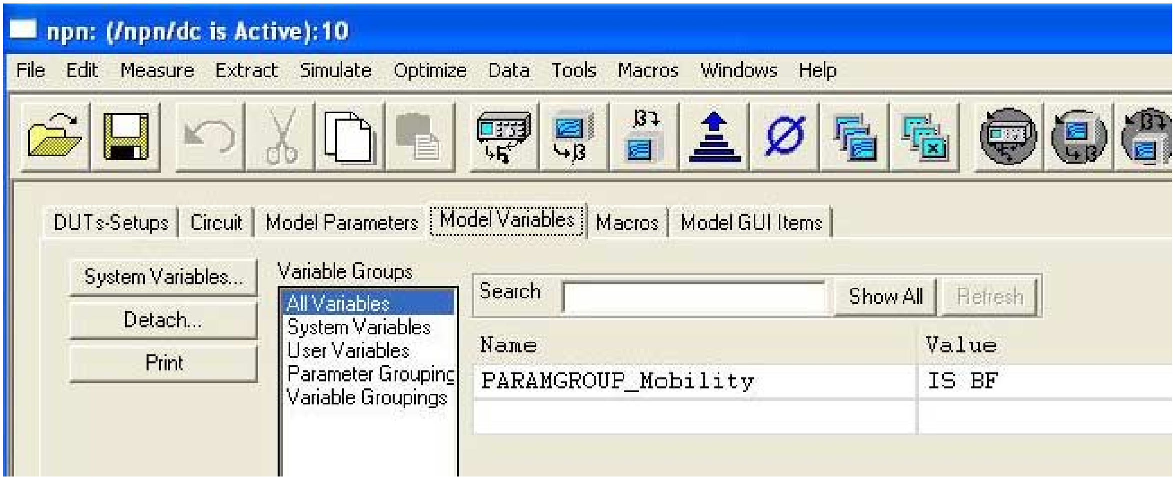
All Parameters includes all parameters in the current level.

Ungrouped includes parameters in *All Parameters* that are not included in other groups.

User Defined Groups:

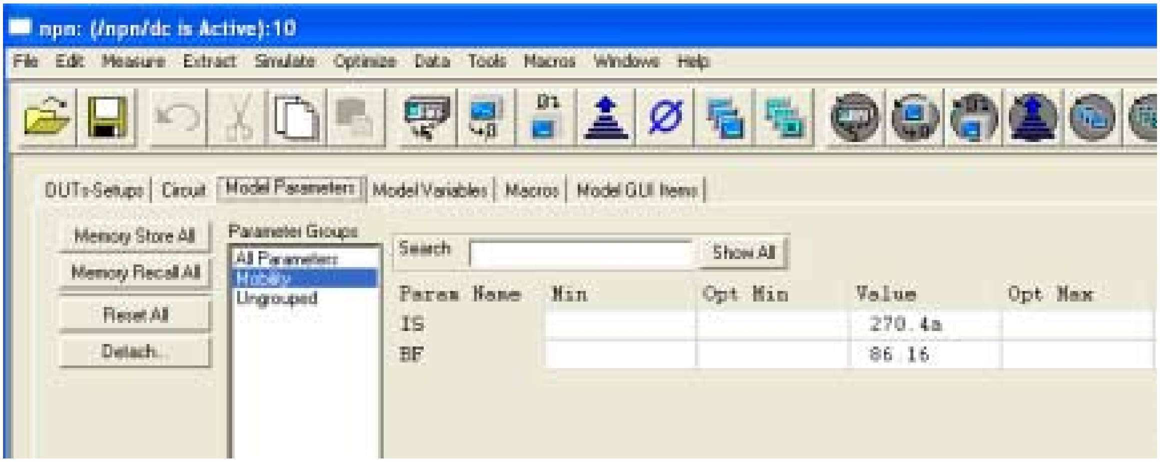
Use the keyword *PARAMGROUP_* to define a parameter group. Specify the parameters in the value column, using <space> as the delimiter. Group name can be nested in the value column. If there are no user defined groups, the list will not be displayed in the *Model Parameters* folder.

For example, define a variable that begins with the keyword *PARAMGROUP_* as shown below.



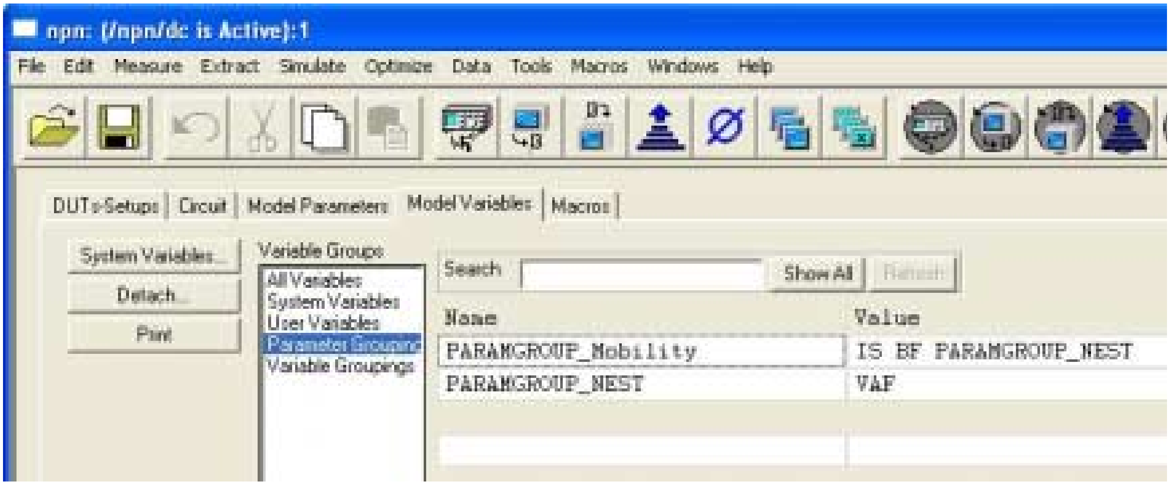
Name	Value
PARAMGROUP_Mobility	IS BF

Then the parameters included in the *Value* column are displayed together in the *Model Parameters* folder as shown below.



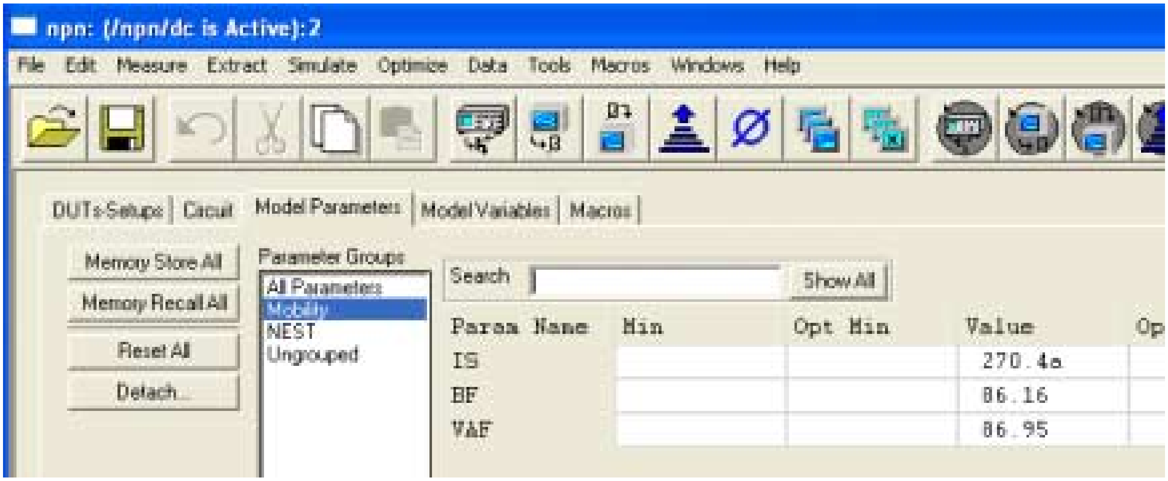
Param Name	Min	Opt Min	Value	Opt Max
IS			270.4a	
BF			86.16	

If you specify a group name in *Value* column.



Name	Value
PARAMGROUP_Mobility	IS BF PARAMGROUP_NEST
PARAMGROUP_NEST	VAF

It is displayed it in the *Model Parameters* folder as shown below.


npn: (/npn/dc is Active):2
File Edit Measure Extract Simulate Optimize Data Tools Macros Windows Help
DUTs-Setups Circuit Model Parameters Model Variables Macros
Memory Store All
Memory Recall All
Reset All
Detach...
Parameter Groups
All Parameters
Mobility
NEST
Ungrouped
Search
Show All

Param Name	Min	Opt Min	Value	Op
IS			270.4a	
BF			86.16	
VAF			86.95	

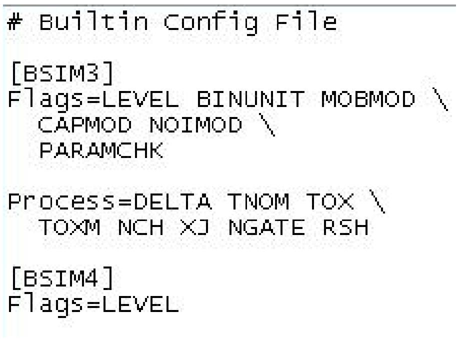
Builtin Groups

The user defined groups for parameters can also be defined in a configuration file.

Format of the file:

- [ ] is used to specify technology name.
- # at the beginning means this line is a comment.
- \ at the end of a line means the following line is also included in current group.
- **device**. means the group is defined for the device level, no prefix means for the model level.
- A empty line will be ignored.

For example,



```
# Builtin Config File

[BSIM3]
Flags=LEVEL BINUNIT MOBMOD \
  CAPMOD NOIMOD \
  PARAMCHK

Process=DELTA TNOM TOX \
  TOXM NCH XJ NGATE RSH

[BSIM4]
Flags=LEVEL
```

Steps for configuration:

1 Specifying file location in *iccap.cfg*.
For example:
`ICCAP_BUILTIN_GROUPS_TEXT=`
`{$ICCAP_ROOT}/config/builtin_groups.txt`

2 Select the technology name in the variable table.
For example:
`BUILTIN_TECHNOLOGY_NAME=BSIM3`

Creating Variable Groups

Variable groups are used for selecting variables.

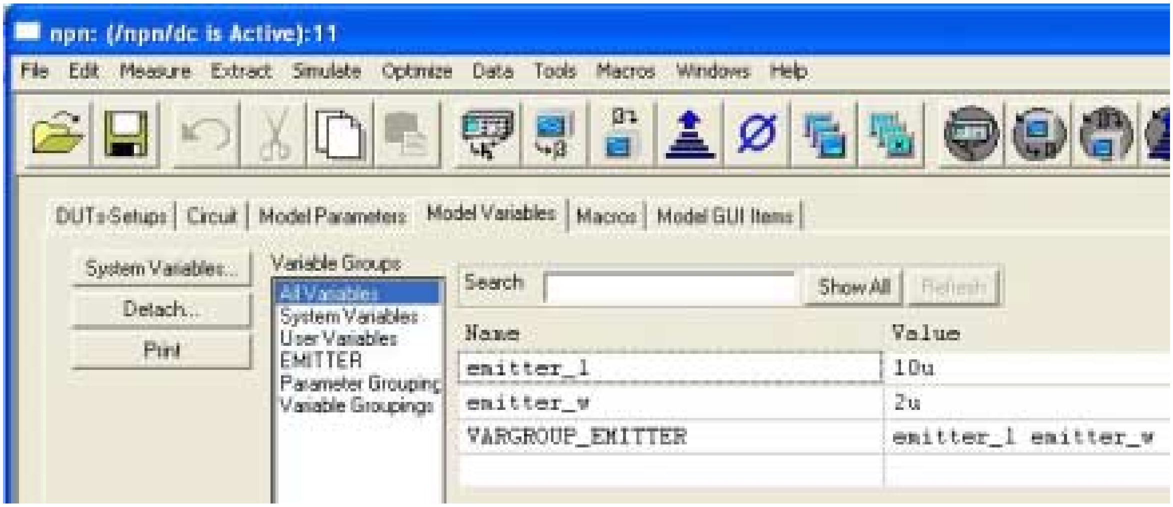
Standard Groups:

- *All Variables* includes all variables in current level.
- *System Variables* includes all system variables defined in the current level.
- *User Variables* includes variables in *All Variables* that are not included in other groups.
- *Parameter Groupings* includes variables that begin with keyword *PARAMGROUP_*.
- *Variable Groupings* includes variables that begin with keyword *VARGROUP_*.

User Defined Groups:

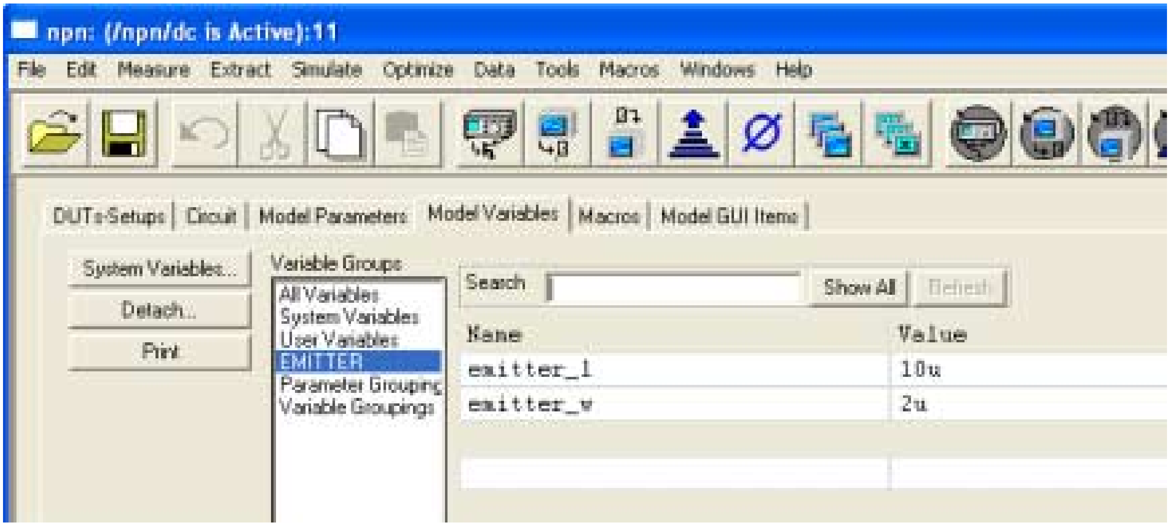
A variable group can be defined in the *Model Variables* folder with the keyword *VARGROUP_*. Specify the variables in the *Value* column, using <space> as delimiter. Group name can be nested in the *Value* column.

For example, define a variable that begins with the keyword *VARGROUP_* as shown below.



Name	Value
emitter_l	10u
emitter_w	2u
VARGROUP_EMITTER	emitter_l emitter_w

Then the variables included in the *Value* column are displayed together in the *Model Variables* folder.



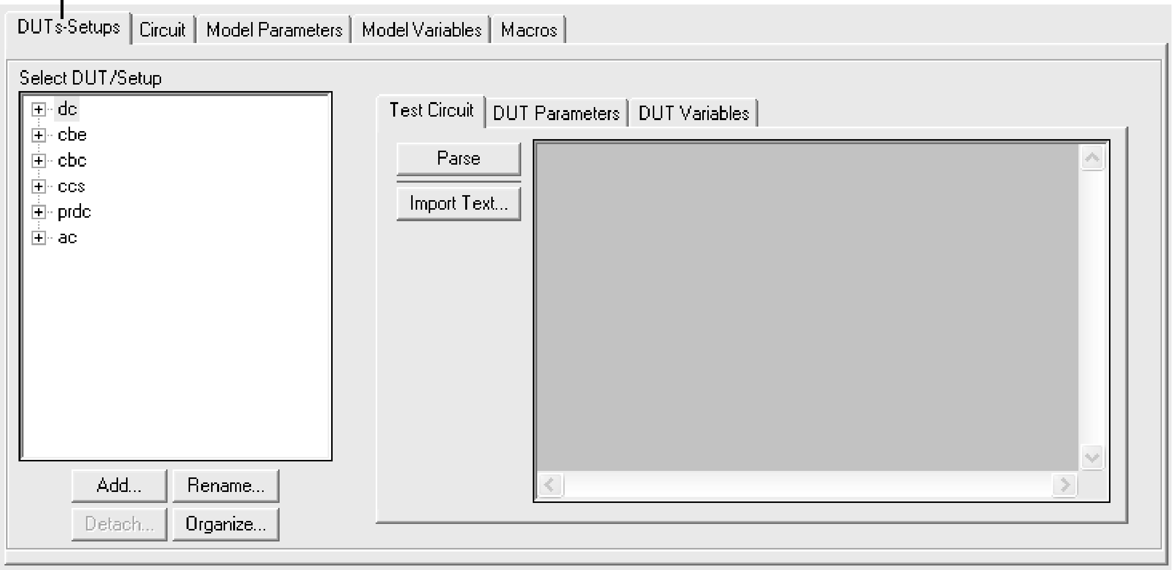
Name	Value
emitter_l	10u
emitter_w	2u

Defining DUT Options

IC-CAP models usually contain several DUTs. DUTs contain groups of setups that have a similar physical connection to the device. In addition to setups, each DUT contains its own variables, parameters and test circuit. If two setups require differences in either of these areas, you must define different DUTs for each.

Before you can measure a device, you must make any needed changes to the DUT options.

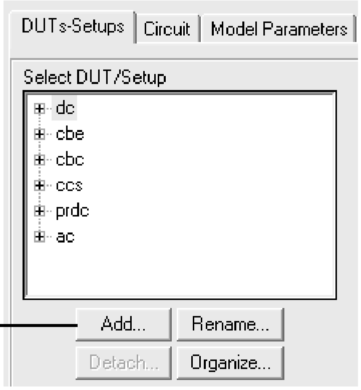
To view the DUT options and make a device active, select the **DUTs-Setups** folder. Then select the DUT.


Click DUTs-Setups to select the DUT

Adding a DUT

To add a DUT:

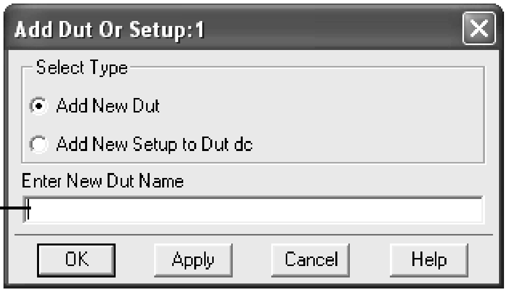
1 In the DUTs-Setups folder, click **Add**.



Add a new DUT

2 In the dialog box that opens, select **Add New DUT**.

3 In the Enter New DUT Name field, type a name for the new DUT and click **OK**.



Enter new DUT name

See also: Organizing DUTs

Deleting a DUT

To delete a DUT:

1 In the DUTs-Setups folder, select the DUT you want to delete.

2 Select **Edit > Cut DUT**, or click **Cut** on the toolbar.

NOTE

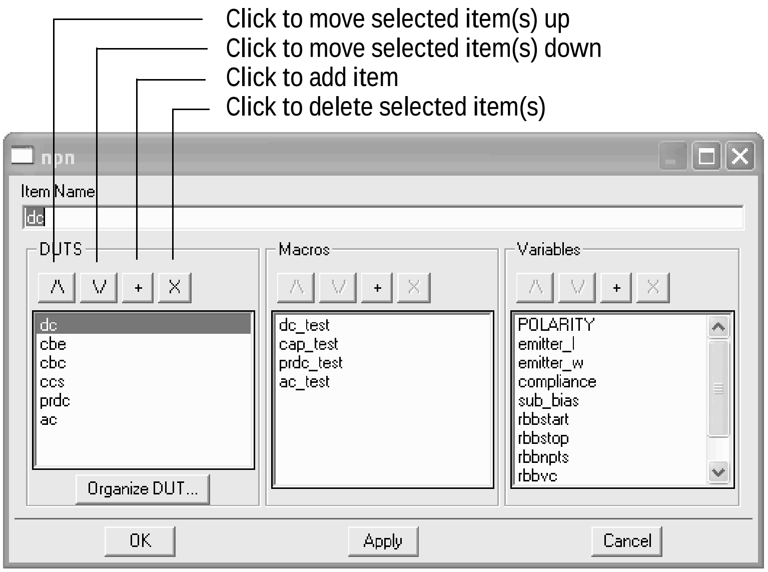
You can also delete a DUT by right-clicking on the DUT to view its pop-up menu. Then select **Edit > Cut**.

See also: Organizing DUTs

Organizing DUTs

To move a DUT in the list:

1 Select **Tools** > **Organize Model** from the Model window to display the following dialog box:



Click to move selected item(s) up
Click to move selected item(s) down
Click to add item
Click to delete selected item(s)

Item Name: dc

DUTS: dc, cbe, cbc, ccs, prdc, ac

Macros: dc_test, cap_test, prdc_test, ac_test

Variables: POLARITY, emitter_l, emitter_w, compliance, sub_bias, rbbstart, rbbstop, rbbnpts, rbbvc

2 From this dialog box, select one or more DUTs.

3 Above the selected DUT(s), click the move up Λ icon to move the selected DUT(s) up in the list, or click the move down V icon to move the selected DUT(s) down in the list. Each click moves the selected DUT(s) one position.

To add a DUT:

1 Choose where you want to add the DUT in the list:

- Select a DUT if you want to add a DUT after that position.

- Do not select a DUT if you want to add a DUT to the first position.

2 Click the plus + icon.

3 A new DUT named *Untitled* is added. Type the desired name in the *Item Name* field then select the Enter key.

To delete DUTs:

1 Select one or more DUTs.

2 Click the delete × icon.

To apply you changes and leave the dialog box open, select the **Apply** button.

To apply your changes and close the dialog box, select the **OK** button.

To close the dialog box without making changes, click the **Cancel** button.

See also: “Organizing DUTs and Setups” on page 636

Specifying a Test Circuit

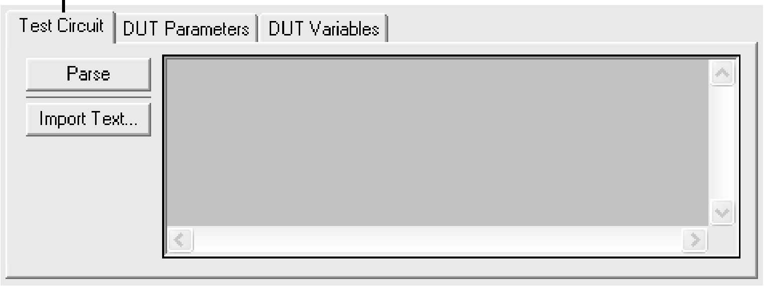
Typically, the test circuit is empty. However, you can use the Test Circuit folder to attach additional circuitry between the device or circuit being modeled and the measurement hardware. For example, an OpAmp model may need to be connected in different configurations for different measurements.

When a test circuit is defined for a DUT, the parameters in the test circuit appear in the DUT parameters folder. For complete information on defining Test Circuits, refer to Chapter 6, “Simulating.”

To specify a test circuit:

1 In the DUTs-Setups folder, select the DUT. Then select **Test Circuit**.

Click Test Circuit to view or edit current values



2 Edit the existing test circuit by typing over the existing values or enter new test circuit data in the window.

NOTE

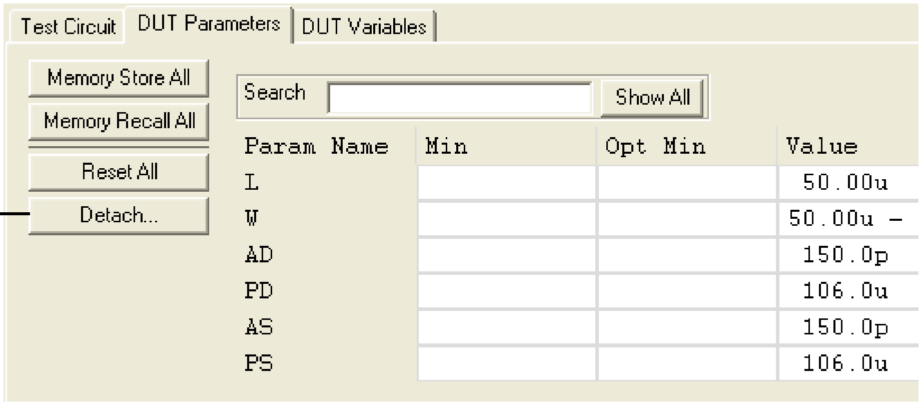
You can copy a test circuit from an existing model. Open the model and select the DUT you want to copy. Select the **Test Circuit** folder. Select **Edit > Copy DUT / Test Circuit**. In the selection dialog box, choose Selected **Test Circuit**. Then select **Edit > Copy**. In the new Model window, select the **Test Circuit** folder. Select **Edit > Paste Test Circuit**.

Defining DUT Parameters

The DUT parameters may take on different values in each DUT. For example, the DC characterization of the MOSFET requires measurements on three DUTs (*large*, *narrow*, and *short*). Each of these DUTs have different specifications for channel length (L) and width (W). You can specify these different values in the DUT Parameters folder. In addition, you can set limits by defining minimum and maximum values for each parameter. If a parameter value is outside its limits, it will be clamped to its minimum or maximum value.

To specify DUT parameters:

1 In the DUTs-Setups folder, select the DUT. Then select **DUT Parameters**.



Param Name	Min	Opt Min	Value
L			50.00u
W			50.00u –
AD			150.0p
PD			106.0u
AS			150.0p
PS			106.0u

Click DUT Parameters to view current values

For frequent editing, detach table

2 Select the **Param Name Value** field and edit the DUT parameters for the device being measured.

3 Optionally, select the **Param Name Min** field and set minimum values for the DUT parameters.

4 Optionally, select the **Param Name Max** field and set maximum values for the DUT parameters.

5 Select an action from the following:

- To temporarily store the current set of parameter values, click **Memory Store All**.
- To recall stored parameter values, click **Memory Recall All**.
- To reset the default values specified in the circuit definition, click **Reset All**.

NOTE

You can copy DUT parameters from an existing model. Open the model and select the DUT you want to copy. Select the **DUT Parameters** folder. Select **Edit > Copy DUT / Device Parameter Set**. In the selection dialog box, choose **Selected Device Parameter Set**. In the new Model window, select the **DUT Parameters** folder. Select **Edit > Paste Device Parameter Set**.

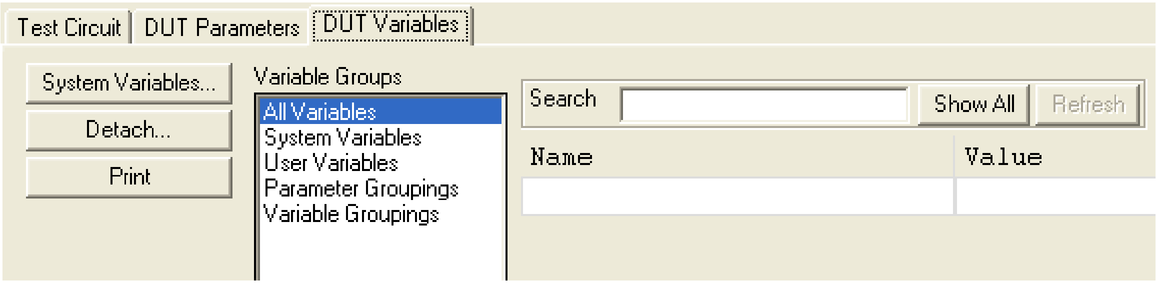
Defining DUT Variables

You can define variables that are accessible only to the DUT or its components in the DUT Variables folder. After you have defined the variables, you can reuse these definitions by applying the *Import* and *Export* commands.

To specify new DUT variables:

1 In the DUTs-Setups folder, select the DUT. Then select **DUT Variables**.

Click DUT Variables to view current values



2 Enter the names and values for all DUT variables:

- Click the first empty row in the Name field and type a name for the variable.
- Press Tab to move to the Value field. Type a value for the variable.
- Press Enter to accept the value and open another row.

NOTE

To view system defaults from the DUT Variables folder, select the System Variables button.

NOTE

You can copy DUT variables from an existing model. Open the model and select the DUT you want to copy. Select the **DUT Variables** folder. Select **Edit > Copy DUT / Variable Table**. In the selection dialog box, choose **Selected Variable Table**. In the new Model window, select the **DUT Variables** folder. Select **Edit > Paste Variable Table**.

Defining Setup Options

You can define setup options from the DUTs-Setups folder. A setup contains folders for defining measurement and simulation options, instrument options, setup variables, transforms, and results.

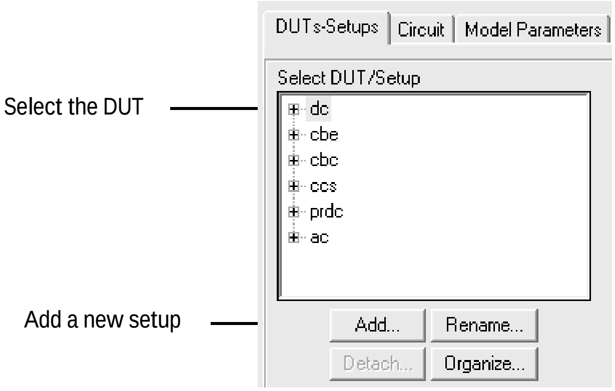
The combination of setup options you specify depends on the following:

- Measurement or simulation to perform. For example, to perform the measurement and simulation, you can specify four input voltages and one output current.
- Extractions or data manipulations required for characterization. For example, to extract values for specific model parameters and optimize the parameter values to achieve the best possible fit between measured and simulated data, you can add a transform that performs the extraction and a transform that performs the optimization.
- Desired data presentation. For example, to view the data, and compare the measured and simulated results, you can add a plot to the setup.

Adding a New Setup

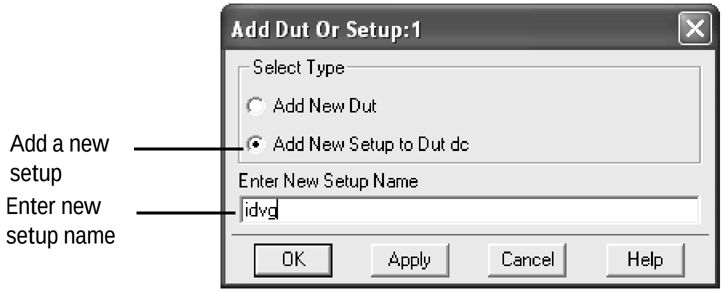
To add a new setup to a DUT:

1 In the DUTs-Setups folder, select the DUT then click **Add**.



2 In the dialog box:

- Select **Add New Setup to DUT [selected]**.
- In the Enter New Setup Name field, type a name for the new setup.

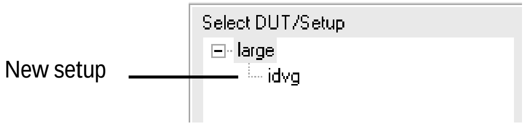


○ Add New Dut

● Add New Setup to Dut dc

Enter New Setup Name: idvg

3 Click **OK**. The new setup is added to the DUT.

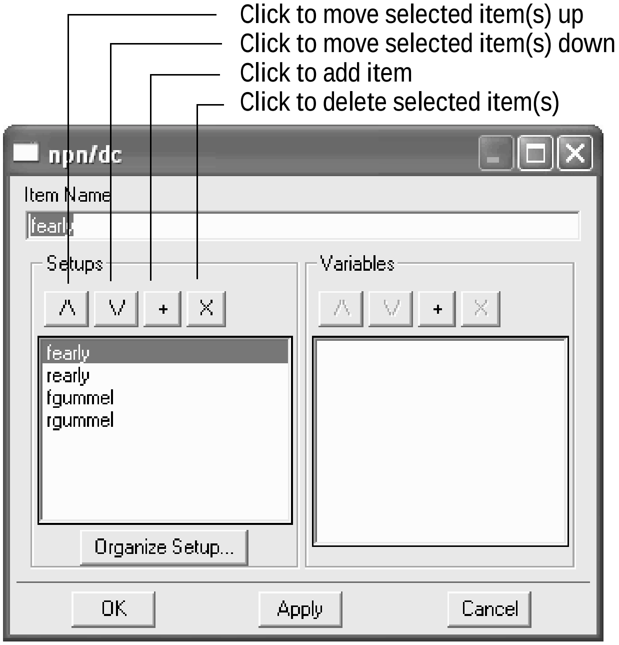


See Also: Organizing Setups

Organizing Setups

To move a Setup:

1 In the DUTs-Setups folder, select the DUT then click **Organize** to display the following dialog box:



NOTE

You can also access this dialog box by first selecting *Tools* > *Organize Model* from the Model window, then select the *Organize DUT* button.

2 In the dialog box, select one or more Setups.

3 Above the selected Setup(s), click the move up Λ icon to move the selected Setup(s) up in the list, or click the move down V icon to move the selected Setup(s) down in the list. Each click moves the selected Setup(s) one position.

To add a Setup:

1 Choose where you want to add the Setup in the list:

- Select a Setup if you want to add a Setup after that position.
- Do not select a Setup if you want to add a Setup to the first position.

2 Click the plus + icon.

3 A new Setup named *Untitled* is added. Type the desired name in the *Item Name* field then select the Enter key.

To delete a Setup:

1 Select one or more Setups.

2 Click the delete ✕ icon.

To apply you changes and leave the dialog box open, select the **Apply** button.

To apply your changes and close the dialog box, select the **OK** button.

To close the dialog box without making changes, click the **Cancel** button.

See also: "Organizing DUTs and Setups" on page 636

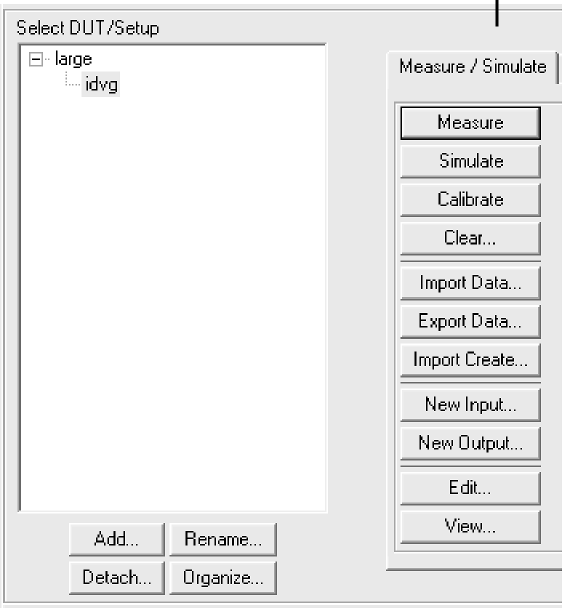
Specifying Measurement and Simulation Options

You can specify measurement and simulation options either by adding new inputs and outputs or by modifying existing values.

To add inputs:

1 In the DUTs-Setups folder, select the setup. Then select **Measure / Simulate**.

Click Measure / Simulate to view current values

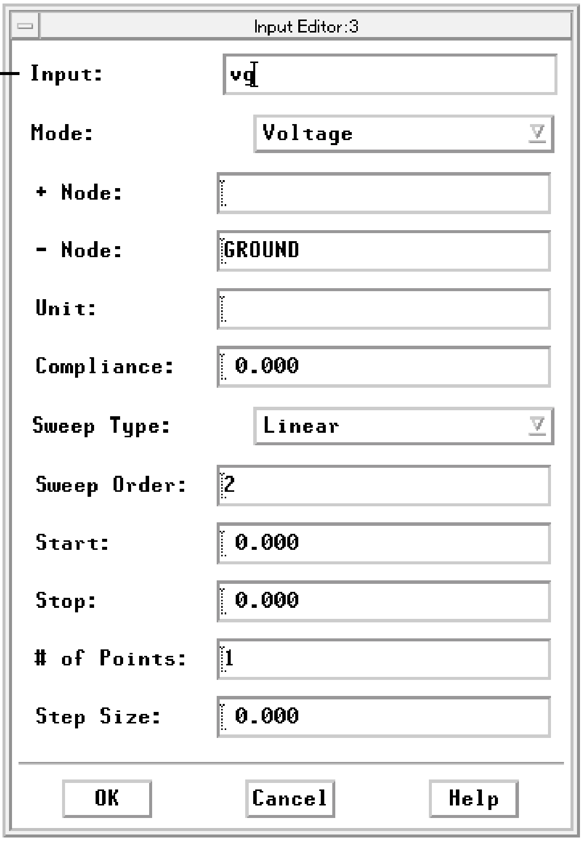


2 Click **New Input** to open the Input Editor.

Add input

3 In the Input field, enter a name for the new input.

Enter name



Input: vg

Mode: Voltage

+ Node:

- Node: GROUND

Unit:

Compliance: 0.000

Sweep Type: Linear

Sweep Order: 2

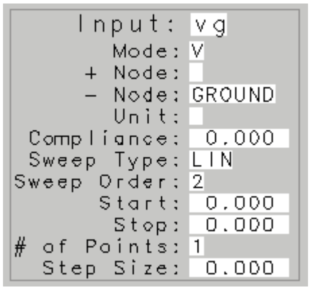
Start: 0.000

Stop: 0.000

# of Points: 1

Step Size: 0.000

4 Click **OK** to add the input to the Measure/Simulate folder.



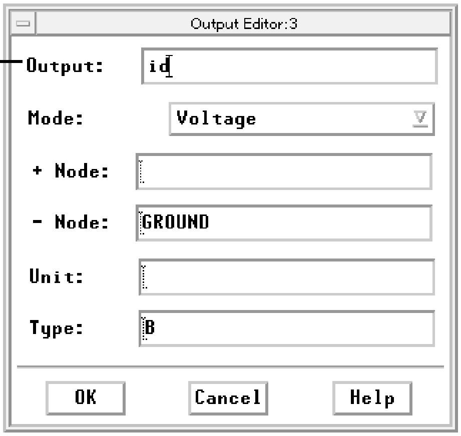
```
       Input: vg
        Mode: V
      + Node:
      - Node: GROUND
        Unit:
  Compliance:  0.000
  Sweep Type: LIN
 Sweep Order: 2
       Start:  0.000
        Stop:  0.000
 # of Points: 1
   Step Size:  0.000
```

To add outputs:

1 Click **New Output** to open the Output Editor.

Add output

2 In the Output field, enter a name for the new output.



Enter name

Output: id

Mode: Voltage

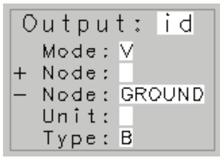
+ Node:

- Node: GROUND

Unit:

Type: B

3 Click **OK** to add the output to the Measure/Simulate folder.



Output: id

Mode: V

+ Node:

– Node: GROUND

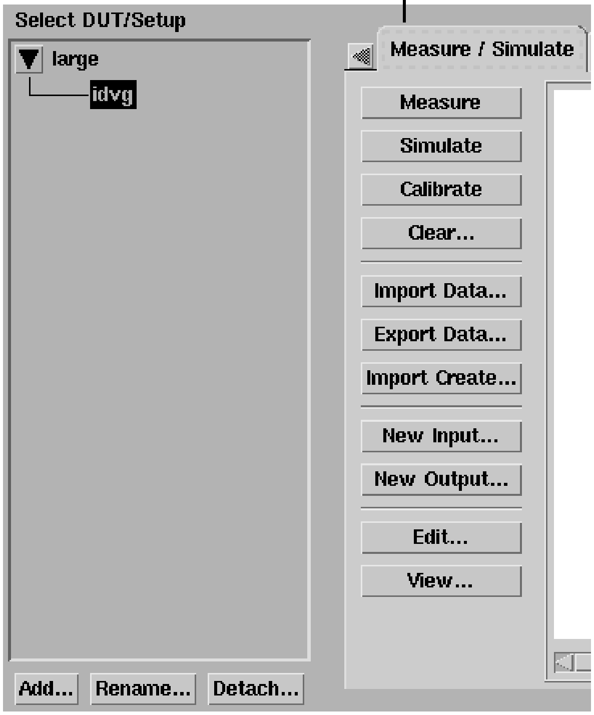
Unit:

Type: B

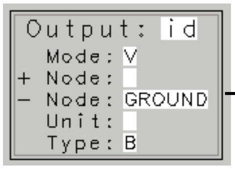
To edit inputs and outputs:

1 In the DUTs-Setups folder, select the setup. Then select **Measure / Simulate**.

Click Measure / Simulate to view current values

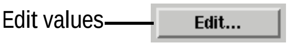


2 In the Measure/Simulate folder, select the Input or Output you want to modify.

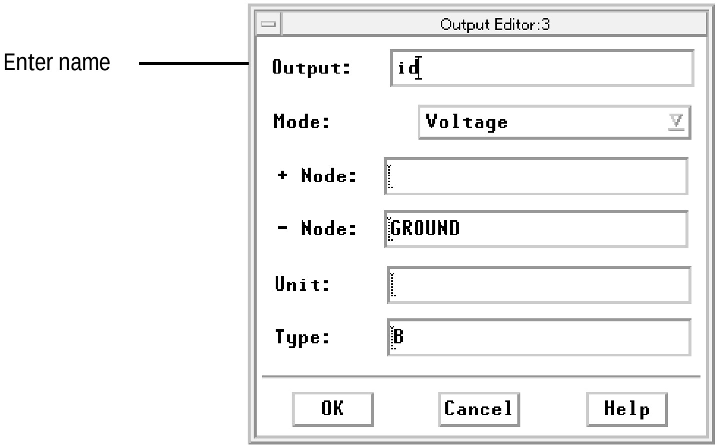

Output: id
Mode: V
+ Node:
− Node: GROUND
Unit:
Type: B

To select table, click outside boundary

3 Click **Edit** to open the Input or Output Editor.



4 Select the field you want to modify and edit the existing data.



Output: id

Mode: Voltage

+ Node:

- Node: GROUND

Unit:

Type: B

5 Click **OK**.

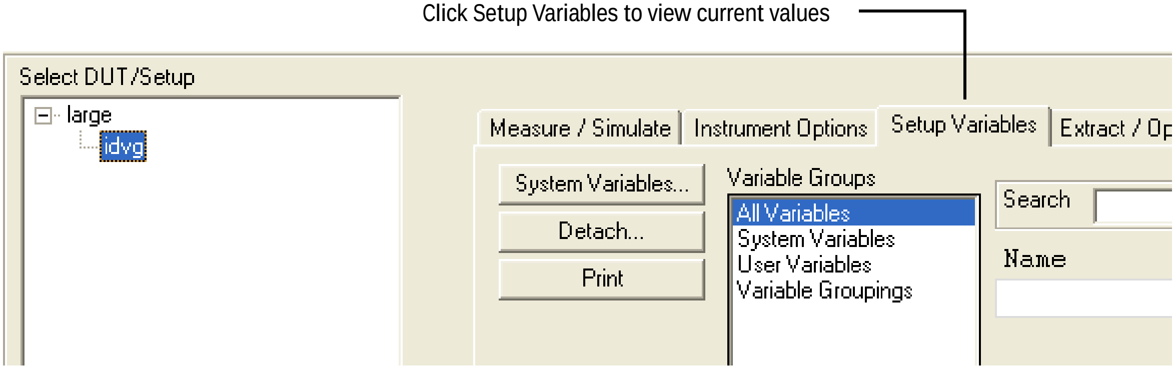
NOTE Alternatively, you can edit directly in the table by selecting the field and typing the new values.

Defining Setup Variables

You can define any variables that are globally available to the setup.

To add or modify setup variables:

1 In the DUTs-Setups folder, select the setup. Then select **Setup Variables**.


Click Setup Variables to view current values

2 Enter the names and values for all setup variables:

- Click the first empty row in the Name field and type a name for the variable.
- Press Tab to move to the Value field. Type a value for the variable.
- Press Enter to accept the value and open another row.

NOTE

To view system defaults from the Setup Variables folder, select the System Variables button.

NOTE

You can copy setup variables from an existing model. Open the model and select the setup you want to copy. Select the **Setup Variables** folder. Select **Edit > Copy Setup / Variable Table**. In the selection dialog box, choose **Selected Variable Table**. In the new Model window, select the **Setup Variables** folder. Select **Edit > Paste Variable Table**.

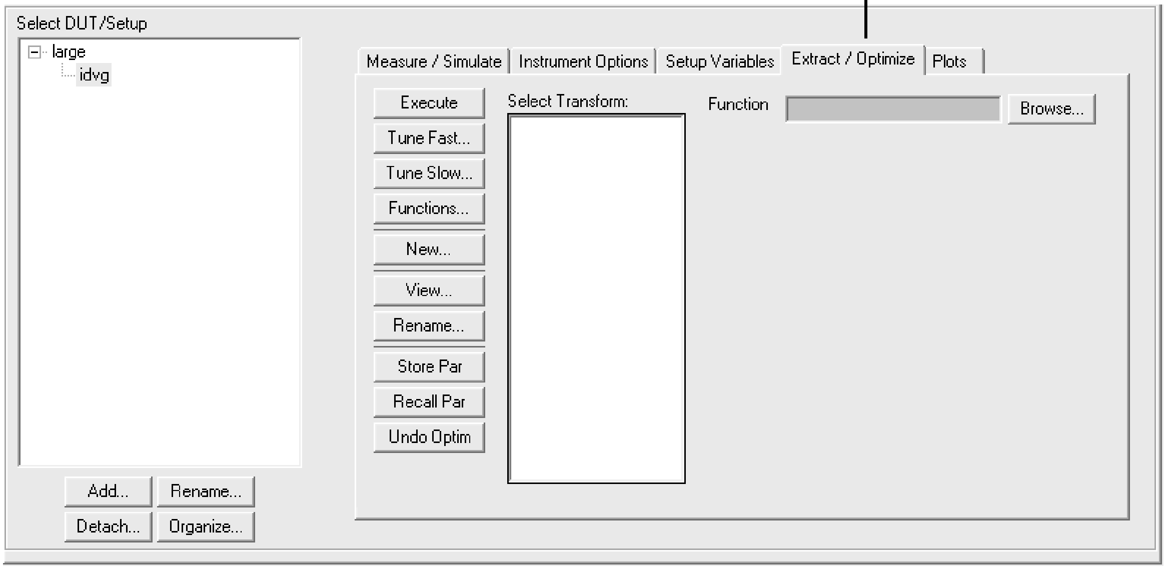
Defining Transform Options

You can define transform options by adding new functions or by modifying existing functions.

To add functions to a new setup:

1 In the DUTs-Setups folder, select the setup. Then select **Extract / Optimize**.

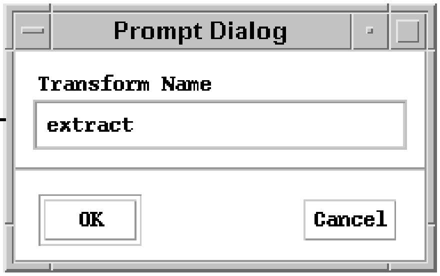
Click Extract / Optimize to view current values



2 Click **New...**

Add transform

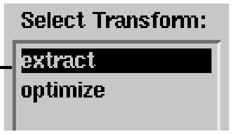
3 In the dialog box, enter a name for the new transform in the Transform Name field.



Transform Name: extract

Enter name

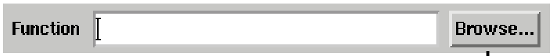
4 Click **OK** to add the transform to the list.



Select Transform:
extract
optimize

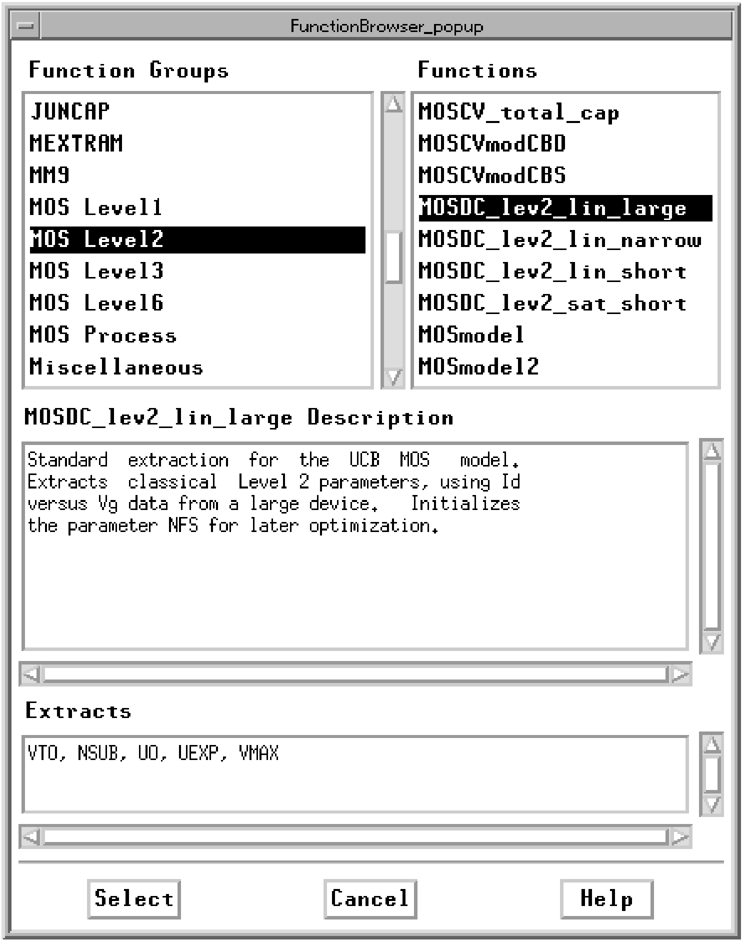
Add transform

5 Select **Browse** to see the transforms provided with the program.



View available functions

6 Choose a Function Group and a Function from the lists.



MOSDC_lev2_lin_large Description

Standard extraction for the UCB MOS model. Extracts classical Level 2 parameters, using Id versus Vg data from a large device. Initializes the parameter NFS for later optimization.

Extracts

VTO, NSUB, UO, UEXP, VMAX

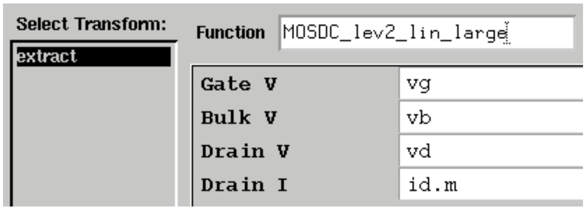
Select a group and a function

7 Choose **Select** to assign the selected function to the transform.

To edit functions:

1 In the DUTs-Setups folder, select the setup. Then select **Extract / Optimize**.

2 Select the function you want to modify. Edit the values in the table by selecting the field and typing the new value.



Select Transform: extract

Function MOSDC_lev2_lin_large

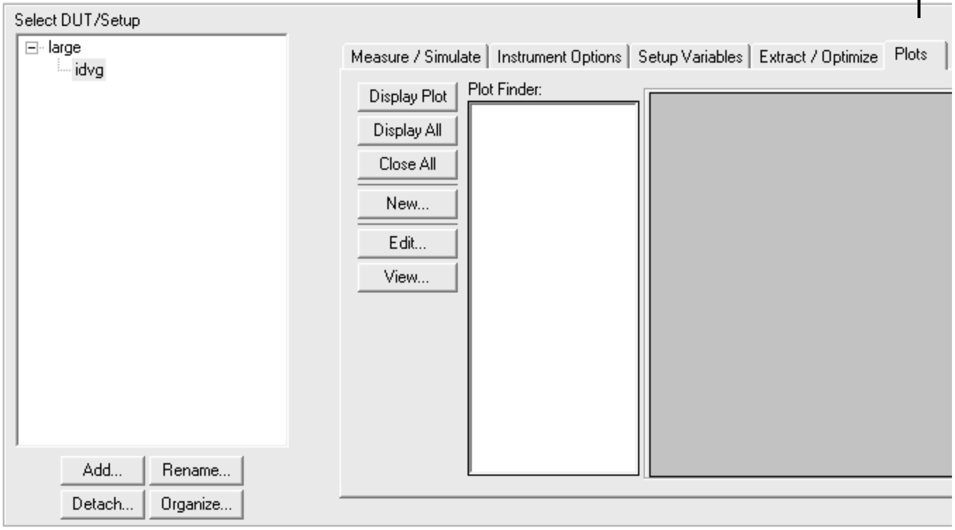
Gate V	vg
Bulk V	vb
Drain V	vd
Drain I	id.m

Edit fields

To add results to a new setup:

1 In the DUTs-Setups folder, select the setup. Then select **Plots**.

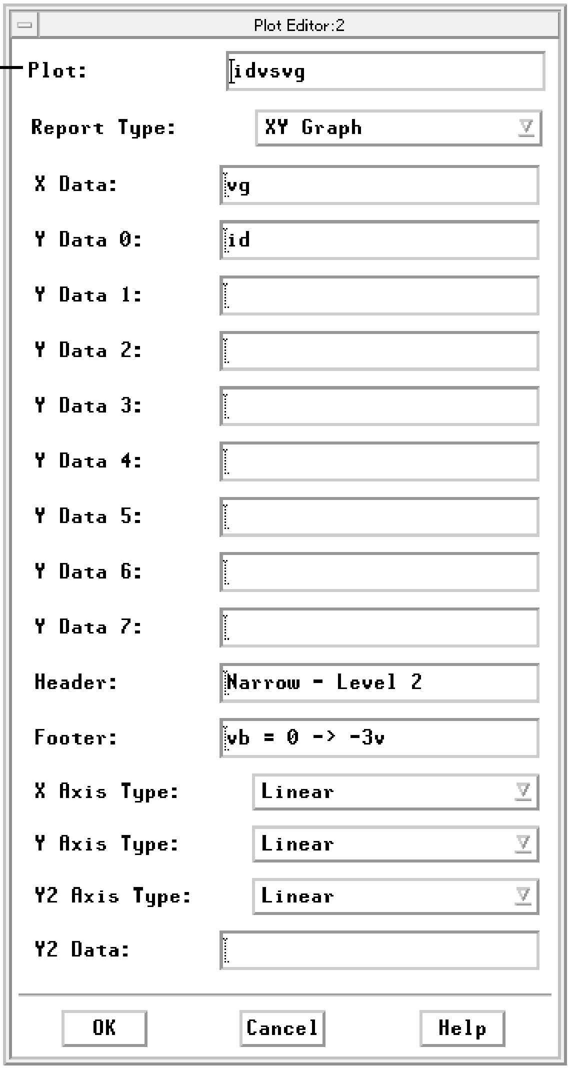
Click Plots to view current values



2 Click **New** to open the Plots Editor.

Add plot

3 In the Plot field, enter a name for the new plot and complete the values as required.



Enter name

Enter plot values

Plot Editor:2

Plot: idvsvg

Report Type: XY Graph

X Data: vg

Y Data 0: id

Y Data 1:

Y Data 2:

Y Data 3:

Y Data 4:

Y Data 5:

Y Data 6:

Y Data 7:

Header: Narrow - Level 2

Footer: vb = 0 -> -3v

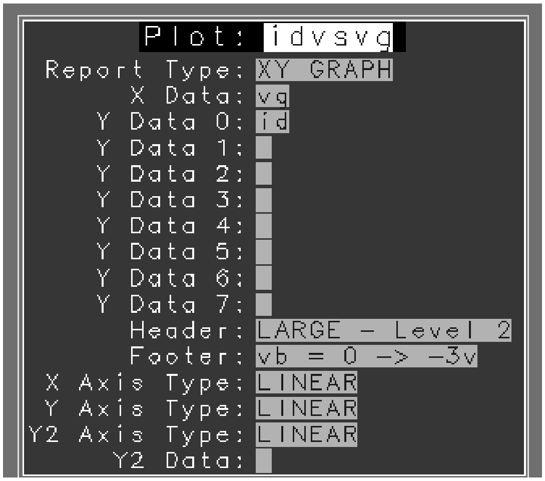
X Axis Type: Linear

Y Axis Type: Linear

Y2 Axis Type: Linear

Y2 Data:

4 Click **OK** to add the plot to the setup.



```
          Plot: idvsvg
 Report Type: XY GRAPH
      X Data: vg
    Y Data 0: id
    Y Data 1:
    Y Data 2:
    Y Data 3:
    Y Data 4:
    Y Data 5:
    Y Data 6:
    Y Data 7:
      Header: LARGE - Level 2
      Footer: vb = 0 -> -3v
 X Axis Type: LINEAR
 Y Axis Type: LINEAR
Y2 Axis Type: LINEAR
     Y2 Data:
```

To edit plots:

1 In the DUTs-Setups folder, select the setup. Then select **Plots**.

2 Select the plot you want to modify.

3 Click **Edit** to open the Plots Editor.

4 Edit the values you want to change and click **OK**.

NOTE Alternatively, you can edit directly in the table by selecting the field and typing the new values.

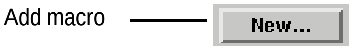
Adding a Macro to the Model

After defining a model, you can define several macros to help organize the characterization procedure. Macros allow any of the available IC-CAP menu operations to be combined and executed as a single operation. IC-CAP features a powerful macro language which is described in detail in Chapter 11, “Creating and Running Macros.”

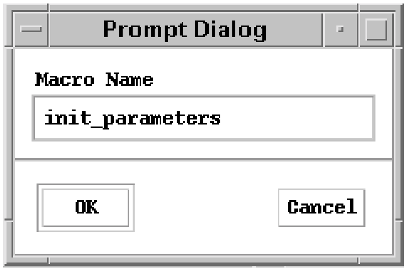
To define a macro:

1 In the Model window, select **Macros**.

2 Click **New**.


Add macro

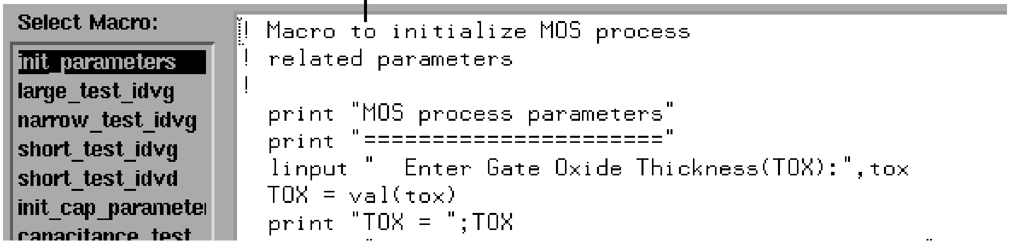
3 At the prompt, type a macro name and choose **OK**.



Macro Name: init_parameters

The new name appears in the macro list.

4 To define the macro, type the program directly into the window.



Define macro

```
! Macro to initialize MOS process
! related parameters
!
  print "MOS process parameters"
  print "======================"
  linput "   Enter Gate Oxide Thickness(TOX):",tox
  TOX = val(tox)
  print "TOX = ";TOX
```

Example: Creating a New Model

The example in this section describes the procedure for defining a portion of the MOSFET model. The MOSFET model supplied with IC-CAP contains a variety of DUTs and Setups for characterizing the DC and Capacitance behavior of the device over a range of geometries. The model is described in Chapter 6, "UCB MOS Level 2 and 3 Characterization," in the *Nonlinear Device Models, Volume 1* manual. Several alterations have been made to the supplied model in order to illustrate advanced features, such as user-defined variables and test circuitry.

NOTE

These instructions assume that the necessary source and measuring instruments are already configured to the IC-CAP system. If this is not the case, refer to the *Installation Guide*.

Add a New Model to the Work Area

1 In the IC-CAP/Main window, select **File > New**. An icon representing a new untitled model appears in the IC-CAP/Main window.

2 Double-click the model icon to open the Model window.

Save the Model File

1 Select **File > Save As**.

2 In the dialog box, enter the design name, **nmos2**, and choose **OK**.

Define Model Variables

1 In the Model window, select **Model Variables**.

2 Click the first empty row in the Name field and type the variable name, `Compliance`.

3 Press Tab to move to the Value field. Type the variable value, `20m`.

Specify the Circuit Definition

1 In the Model window, select **Circuit**.

2 Enter the circuit description. For the *nmos2* model, the circuit is:

```
.OPTIONS ucb
M1 1=D 2=G 3=S 4=B MOSMOD L=2u W=3u
+ AD=9p PD=12u AS=9p PS=12u
.MODEL MOSCAP NMOS
+ LEVEL = 2.000
+ UO = 600
+ VTO = 0
+ NFS = 0
+ TOX = 100n
+ NSUB = 1E15
+ UCRIT = 10.0K
+ UEXP = 0
+ VMAX = 0
+ RS = 0
+ RD = 0
+ XJ = 0
+ LD = 0
+ DELTA = 0
+ NEFF = 1.00
+ NSS = 0
+ CGSO = 0
+ CGDO = 0
+ CGBO = 0
+ CBD = 0
+ CBS = 0
+ CJ = 0
+ MJ = 0.5
+ CJSW = 0
+ MJSW = 0.33
+ IS = 1.0e-14
+ PB = 0.8
+ FC = 0.5
```

The new circuit definition is parsed automatically when you click outside the window.

View Model Parameters When a new model is read, parameter values are set to the defaults defined in the *MODEL* definition of the Circuit. To view the model parameters, select the Model Parameters folder. For this example, no changes are made to the model parameters.

Add a DUT The MOSFET model contains three DUTs for DC characterization and two DUTs for Capacitance characterization. The procedure creates the DC DUT *large*.

1 In the Model window, select **DUTs-Setups**.

2 Click **Add**.

3 In the dialog box, click **Add New DUT**.

4 In the Enter New Dut Name field, type the DUT name, `large`.

5 Click **OK**.

View DUT Variables To view the model parameters, select the DUT name **large**, then select the **DUT Variables** folder. For this example, no changes are made to the DUT Variables.

Specify DUT Parameters The DC characterization of the MOSFET requires measurements on three DUTs. Each of these DUTs have different specifications for *L* and *W*. For the example, specify geometry parameters for the large device.

1 Select **DUT Parameters**.

2 Select the **Param Name Value** field and edit the DUT parameters for the device being measured. The DUT parameters are:

```
L - 50.00u
W - 50.00u
AD - 150.0p
PD - 106.0u
AS - 150.0p
PS - 106.0u
```

Specify a Test Circuit A test circuit is usually not used with a MOSFET, but this example defines one. You can use a test circuit if you have a large amount of probe resistance that you do not want included in the model parameters. In this case, you can define a test circuit that has an additional resistor connected between each of the internal device nodes and the external nodes that are connected to the sources.

The example test circuit definition contains the following information:

- Line 1 contains the mandatory term *.SUBCKT,* which is required if a circuit contains more than one device (in this case, a transistor and four resistors). Line 1 also includes the arbitrary name of the test circuit and the identities of the four device nodes. The four device nodes are numbered as in the Model Circuit Definition (1 = C, 2 = B, 3 = E, 4 = S); however, alternate names can be used. If you do not assign a name to each node number, the system assigns the node number as the node name.

- Line 2 identifies the device (Q1), the internal nodes of the transistor (11, 12, 13, and 14), the name of the model (*npn*), and the *AREA* parameter. The model name must be entered precisely as it appears in the model icon.
- Lines 3 through 6 specify the four resistors that represent the additional resistance at each node. Each resistor is connected from an external Test Circuit node to the corresponding internal device node.
- Line 7 indicates the end of the *.SUBCKT* definition.

To define a test circuit:

1 In the DUT window, select **Test Circuit**.

2 In the test circuit window, type the test circuit data:

```
.SUBCKT large 1=D 2=G 3=S 4=B
M1 11 12 13 14 nmos2 L=50u W=50u
RDX 11 1 5.0
RGX 12 2 5.0
RSX 13 3 5.0
RBX 14 4 5.0
.ENDS
```

When a Test Circuit is defined for a DUT, the parameters in the Test Circuit appear in the DUT parameters folder and are editable.

Add a Setup

1 In the Model window, select **DUTs-Setups**.

2 Click **Add**.

3 In the dialog box, click **Add New Setup to Dut large**.

4 In the Enter New Setup Name field, type the setup name, `idvg`.

5 Click **OK**.

Specify Setup Components To perform the measurement and simulation, four input voltages and one output current are specified. In the case of the *idvg* setup for the *large* DUT of the MOSFET, the drain current, *id*, of the device, is monitored as a function of the gate voltage, *vg,* and the substrate (bulk) voltage, *vb.* The drain voltage, *vd,* and the source voltage, *vs,* are held constant.

For each of the components added to the setup, fill in a table of information that defines the desired behavior. This example presents only a small subset of the available capabilities possible in a setup. To edit a setup:

1 In the DUT-Setups window, select **large / idvg**.

2 Select **Setup Variables** and define any variables you want to apply to the setup.

3 Select **Measure / Simulate**.

4 Click **New Input** to open the Input Editor.

5 In the Input field, enter the new input name, vg.

6 Complete the value entries as shown below:

Mode	Voltage
+ Node	G
- Node	GROUND
Unit	SMU2
Compliance	10.00u
Sweep Type	Linear
Sweep Order	1
Start	0.000
Stop	5.000
# of Points	21
Step Size	250.0m

The value in the *Compliance* field is a reference to the global compliance variable that was defined for the setup variables.

7 Click **OK** to add the input to the Measure/Simulate folder.

8 Add the input vb. Complete the value entries as shown below:

Mode	Voltage
+ Node	B
- Node	GROUND

Mode	Voltage
Unit	SMU4
Compliance	1.000m
Sweep Type	Linear
Sweep Order	2
Start	0.000
Stop	-3.000
# of Points	3
Step Size	-1.500

The value in the *Compliance* field is a reference to the global compliance variable that was defined for the setup variables.

9 Add the inputs that remain constant, `vd` and `vs`, by completing the value entries as shown below:

	Input vd	Input vs
Mode	V	V
+ Node	D	S
- Node	GROUND	GROUND
Unit	SMU1	SMU3
Compliance	100.0m	100.0m
Sweep Type	CON	CON
Value	100.0m	0.000

When you change the *Sweep Type* from *LIN* to *CON* the fields at the bottom of the table used to specify a linear sweep are replaced by a single *Value* field. Similarly, a change in the *Mode* alters the structure of the upper portion of the table.

10 Add the output `id` by completing the value entries as shown below:

Mode	I

To Node	D
From Node	GROUND
Unit	SMU1
Type	B

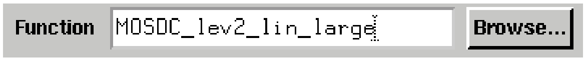
When you change the *Mode* to *I,* the names of the node fields of the table change accordingly.

Add Transforms When the measurement has been made, values for the model parameters that control the *large* DUT behavior are extracted, and, optionally, the parameter values are optimized to achieve the best possible fit between measured and simulated data.

To accomplish this, you add a transform that performs the extraction and a transform that performs the optimization.

1 In the setup window, select **Extract / Optimize**.

In the Function field, click **Browse**. Select the **Function Group MOS Level2** and the Function **MOSDC_lev2_lin_large**.



Function MOSDC_lev2_lin_large

In the list of transforms, select **extract**. Complete the values as shown below:

Gate V	vg
Bulk V	vb
Drain V	vd
Drain I	id.m

2 In the list of transforms, select **optimize**.

For Inputs, complete the values as shown below:

Target	id.m
Simulated	id.s

Weight	1.000	1.000

For Parameters, complete the values as shown below

Name	Min	Max
VTO	0.000	10.00
NSUB	1.000T	1.000E+18
UO	100.0	1.000K
UEXP	0.000	1.000
	1.000f	1.000MEG

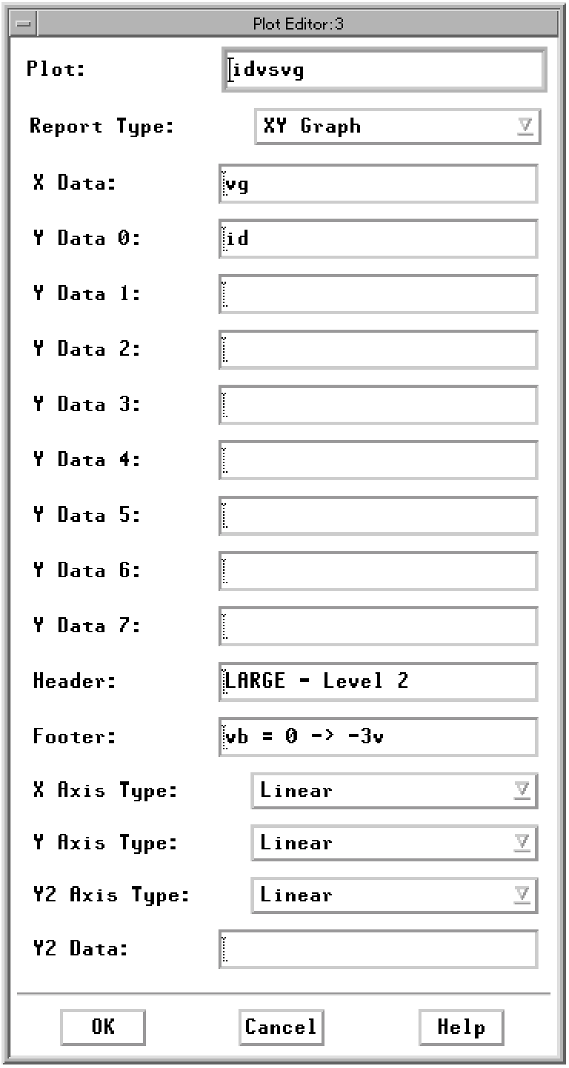
For Options, complete the values as shown below:

Y Lower Bnd	0.000	Rank 1 Flag	Yes
Y Upper Bnd	0.000	Param Delta	1.000m
X Lower Bnd	0.000	Param Print	A
X Upper Bnd	0.000	Error Print	A
Comb Filter	1	Sens Print	1
RMS error	10.00m	Normal Sens	Yes
Max error	10.00m	Optim Mode	L
Abs Err Flag	Yes	Rand Std Dev	300.0m
Param Tol	1.000m	Rand Reward	500.0m
Function Tol	1.000m	Rand Penalty	500.0m
Max Evals	50	Rand Seed	0
Extract Flag	No	Rand Iters	25

The tables adjust depending on the functions selected.

Finally, in order to view the data and compare the measured and simulated results, add a plot to the setup.

1 In the setup window, select **Plots**. Define the *idvsvg* plot as shown below:



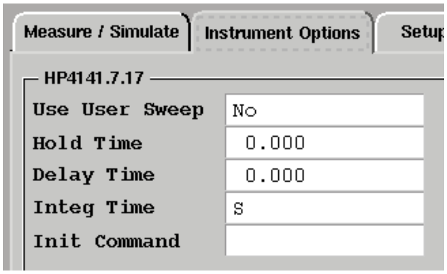
Plot Editor:3

Plot:	idvsvg
Report Type:	XY Graph
X Data:	vg
Y Data 0:	id
Y Data 1:	
Y Data 2:	
Y Data 3:	
Y Data 4:	
Y Data 5:	
Y Data 6:	
Y Data 7:	
Header:	LARGE - Level 2
Footer:	vb = 0 -> -3v
X Axis Type:	Linear
Y Axis Type:	Linear
Y2 Axis Type:	Linear
Y2 Data:	

The plot table adjusts according to the value of the *Report Type* field.

2 Select **Instrument Options** to enter the instrument options values. Instruments used in each setup are determined from the instrument unit names specified in the Unit fields of the Input and Output tables.

Usually only one DC measurement instrument is used in this setup. When using the HP 4141, the window looks similar to the following figure.



HP4141.7.17

Use User Sweep	No
Hold Time	0.000
Delay Time	0.000
Integ Time	S
Init Command	

Figure 11 Instrument Options for HP 4141

Add a Macro You can add a macro to simplify simulations. The example for the MOSFET defines a macro that performs all of the DC simulations together. This macro calls the *Simulate* function for the *large, narrow,* and *short* DUTs.

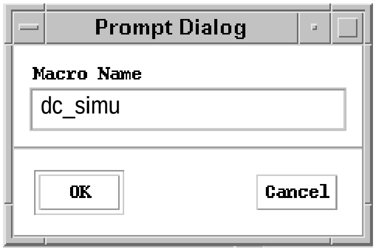
To define a macro:

1 In the Model window, select **Macros**.

2 Click **New**.

Add macro

3 At the prompt, type the macro name, `dc_simu`.



Macro Name
dc_simu

4 Click **OK**. The new name appears in the macro list.

5 To define the macro, type the program directly into the window.

```
menu_func("large", "Simulate")
menu_func("narrow", "Simulate")
menu_func("lshort", "Simulate")
```

To execute the macro, click **Execute**. Execute

When you execute this macro, all three DC DUTs (large, narrow, and short) are simulated.

NOTE

You can also execute macros from the Model window's Macro menu. Click on **Macros > Execute**, the click on the macro you want to execute. This is useful if you want to execute a macro, and you do not have the Macro folder open.

Create Other DUTs and Setups You can create other DUTs and setups by repeating the example procedure. Or, you can build a complete model by copying portions of the supplied model and then modifying any parts of the new setup that needs to be changed to achieve the desired behavior. For the MOSFET model, you can copy the entire *large* DUT to two new DUTs named *narrow* and *short*. In the *short* DUT, the *idvg* setup can be copied to a new Setup named *idvd.* After making the necessary modifications, you have the complete set of DUTs and setups necessary for DC MOSFET characterization. For details, refer to "Copying Parts of a Model" on page 118.

5
Making Measurements

This chapter describes the procedure for connecting the hardware, configuring the hardware within the program, and performing measurements.

Measurement requires the physical connection of a test fixture and necessary instruments to the IC-CAP system. A typical IC-CAP test setup consists of a test fixture (such as the HP 16058A) on which the device under test (DUT) is mounted, along with any additional test circuitry. Measurement instruments are connected to the fixture terminals using triaxial or coaxial cables and to the GPIB bus on the computer using an GPIB connector and cable.

Hardware connections vary depending on the test fixture being used. For information on available test fixtures, refer to the operating manual for the specific instrument.

The general procedure for performing measurements is:

- Physically connect the hardware.
- Specify an interface file for each GPIB card.
- Build an active instrument list.
- Assign unit names.
- Use unit names in setups.
- Specify instrument options.
- Perform calibration.
- Measure.

NOTE

For a list of supported instruments and configuration information, refer to Chapter 1, "Supported Instruments," in the *Reference* manual.

Hardware Connections

Connecting the hardware consists of making the appropriate GPIB connections between the instruments and the computer. Connections must also be made between the individual units of the instruments and the test fixture. The test device should be inserted into the test fixture with the appropriate unit to device connections made.

DC Connections DC connections usually require connecting the SMUs and VS/VMUs of a DC Analyzer to the test fixture in which the DUT is mounted. The test fixture may be the HP 16058A Test Fixture, a probe station, or a switching matrix.

CV Connections CV measurements are taken using either an internal DC bias or an external DC bias. Usually, a DC analyzer is used to provide the external DC bias. When using internal or external biasing sources take note of the hardware connection differences.

When using the DC bias internal to the CV meter, you only need to connect the capacitance meter high and low units to the fixture in which the DUT is mounted. The internal DC bias is automatically available through this connection.

When using an external DC bias, this external source is used by making a connection to the rear panel of the CV meter.

AC Connections AC connections require connecting the 2-port units of a network analyzer to the fixture in which the DUT is mounted, as well as connecting a DC analyzer to provide a DC source. Connect the external DC source to the rear panel of the network analyzer.

Time Domain Connections Time domain connections may involve cabling between a generator, an oscilloscope, and the DUT. Another cable may be used to pass a trigger pulse from a generator to the oscilloscope's trigger input. For detailed information on these connections, refer to the text file, 54120.help, provided in the lib/iccap directory.

In the example procedure, the DC Analyzer contains two SMUs (Source/Measurement Units), each capable of sourcing and measuring voltage and current. For each SMU, a single triaxial cable carries the input signal to the device and the output signal from the device.

To connect the DC Analyzer and the test fixture:

1 Connect a cable from each of the connectors (marked SMU1 - 2) on the back of the DC Analyzer to each of the connectors (marked SMU1 - 2) on the back of the test fixture.

2 Insert the device under test (DUT) into the test fixture and make the appropriate SMU to device lead connections.

3 Connect an GPIB cable from the GPIB connector on the back of the DC Analyzer to the GPIB bus connector on the computer.

4 Make sure the GPIB address is set to a value that does not conflict with other GPIB addresses on the bus.

5 Turn on the DC Analyzer.

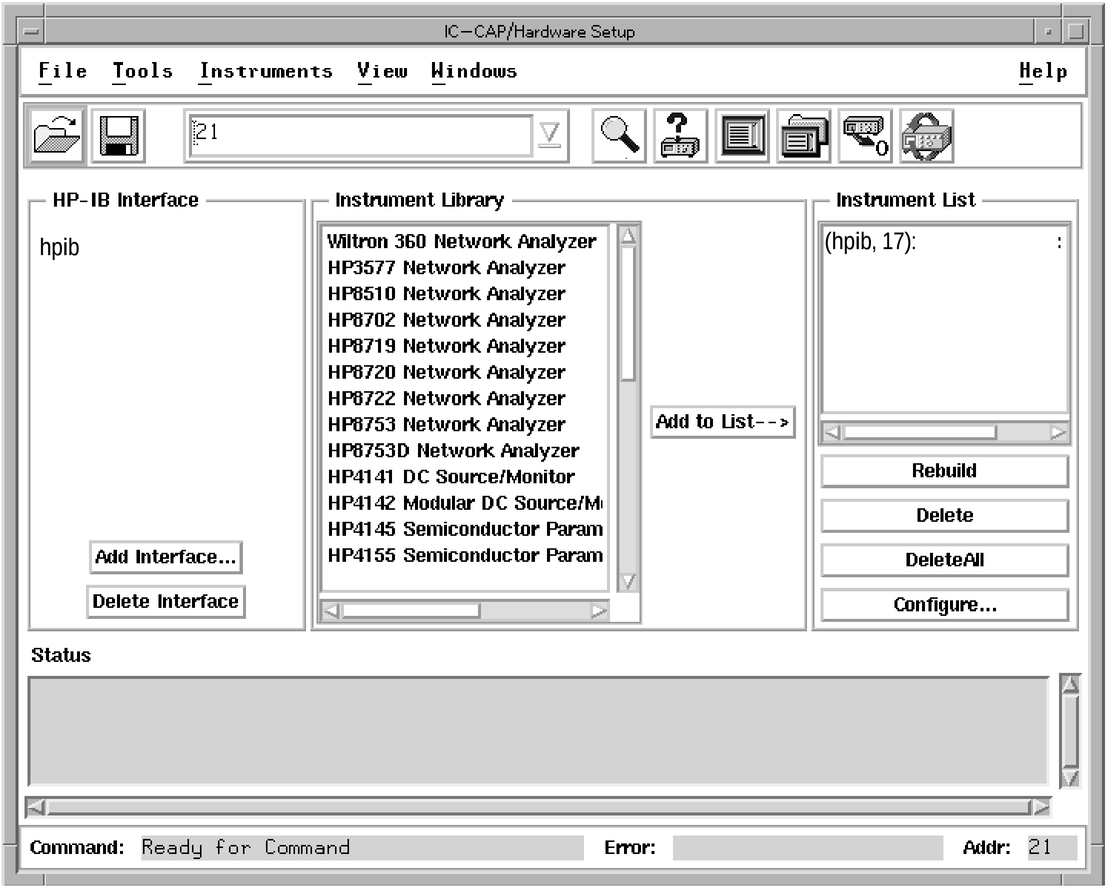
Configuring the System

You must configure the program so it can recognize the system instruments on the GPIB and the individual source/monitor units (SMUs) in the measurement instruments. The complete configuration procedure is done after initial system installation or any time the system hardware is changed. Any time you change device types, such as changing from FET to BJT, you must rename the SMUs in the configuration.

A configuration file containing hardware information and system level variables is generated during the installation procedure and read when the program is started. When you exit IC-CAP, the current configuration is saved in the *.icconfig* file in your home directory. For details, refer to the *Installation Guide*.

To view the hardware setup:

- In the IC-CAP/Main window, select **Tools > Hardware Setup** or, click **Hardware Setup** on the toolbar.



21

hpib

Wiltron 360 Network Analyzer
HP3577 Network Analyzer
HP8510 Network Analyzer
HP8702 Network Analyzer
HP8719 Network Analyzer
HP8720 Network Analyzer
HP8722 Network Analyzer
HP8753 Network Analyzer
HP8753D Network Analyzer
HP4141 DC Source/Monitor
HP4142 Modular DC Source/M
HP4145 Semiconductor Param
HP4155 Semiconductor Param

(hpib, 17): :

Command: Ready for Command
Error:
Addr: 21

Viewing the Instrument Library

The Instrument Library lists all instruments for which IC-CAP drivers are provided. The Instrument Library cannot be edited. Refer to Chapter 1, “Supported Instruments,” in the *Reference* manual for a list of supported instruments and configuration information.

Specifying Interface Files

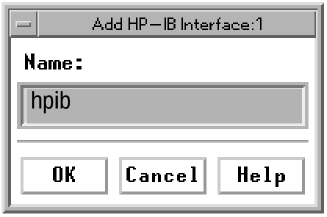
You must specify an interface file for each GPIB card being used for the measurement. See the IC-CAP *Installation and Customization Guide* for information about configuring your interface. If your computer does not have access to an interface file, and you need to specify instrument options in a Model file, you can add a dummy interface. When you enter the name of an interface, begin the name with *dummy*, such as *dummy_gpib*, and IC-CAP will interpret it as a dummy interface.

To add an interface file:

1 In the IC-CAP/Main window, select **Tools > Hardware Setup** or click **Hardware Setup**.

2 Click **Add Interface**.

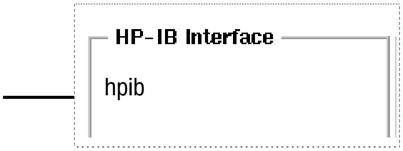
3 In the prompt dialog, type the name of the interface.



Name:
hpib

4 Choose **OK**.

The GPIB interface name is added to the HB-IB Interface list and made the currently selected interface.



hpib

Interface added

Building an Active Instrument List

The IC-CAP program must be able to recognize the instruments that are physically connected to the setup. When an instrument is added to the active list, the program identifies the instrument by instrument name, interface name, and GPIB address.

For example, an HP 4141 added to the active instrument list as:

HP4141(gpib0, 23)

where:

HP4141 = instrument name

gpib0 = gpib interface symbolic name

23 = HP 4141 gpib address

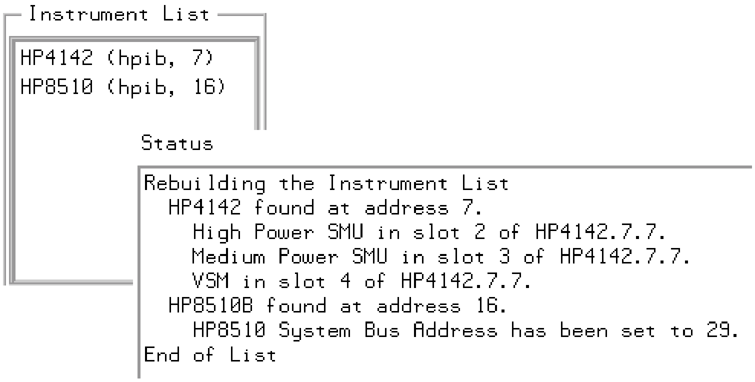
If an instrument is powered up and connected to the GPIB bus, you can have the program add it to the active instruments list automatically.

To add an instrument automatically:

1 In the IC-CAP/Main window, click **Hardware Setup** on the toolbar. .

2 Click **Rebuild**.

All active instruments, with their respective addresses and interface name, are added to the list. The status of the setup is displayed in the Status window.



```
HP4142 (hpib, 7)
HP8510 (hpib, 16)
```

```
Rebuilding the Instrument List
  HP4142 found at address 7.
    High Power SMU in slot 2 of HP4142.7.7.
    Medium Power SMU in slot 3 of HP4142.7.7.
    VSM in slot 4 of HP4142.7.7.
  HP8510B found at address 16.
    HP8510 System Bus Address has been set to 29.
End of List
```

NOTE The hardware displayed in the Instrument List may not reflect the physical instruments actually connected. See adding instruments manually below.

Alternatively, you can add an instrument to the active instruments list manually, whether the instrument is physically connected to the system or not.

To add an instrument manually:

1 In the IC-CAP/Main window, click **Hardware Setup** on the toolbar. .

2 Select the instrument in the Instrument Library.

3 Click **Add to List**. Add to List-->

NOTE You can not manually add an HP/Agilent 4142 or 4155/6 to the list of the Active Instruments because the units of these instruments are configurable. The program finds what units these instruments have when you execute *Rebuild Active List*.

Certain IC-CAP error messages include the internal instrument/GPIB identifier. It is helpful for you to understand the address syntax. The format of this internal id is:

`INSTR_TYPE SELECTCODE ADDR or INSTR_TYPE.SELECTCODE.ADDR` where:

INSTR_TYPE is the instrument model number exactly as it is listed in the Instrument Library.

SELECTCODE is the gpib interface's Logical Unit Number or Board number of the GPIB interface.

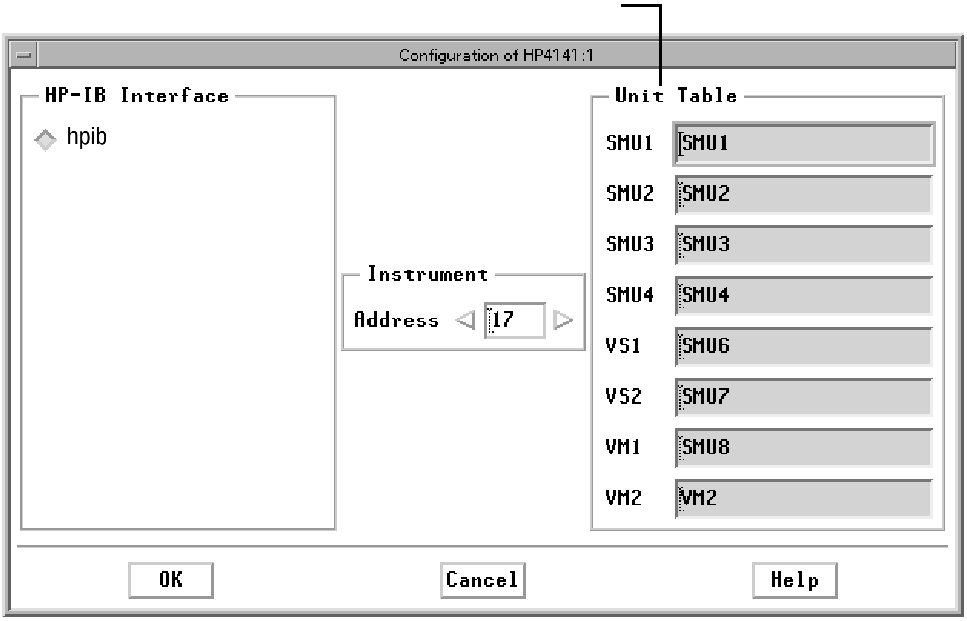
ADDR is the address (in decimal notation) of instrument, as set on the instrument's address selector switch.

NOTE

On Sun SPARC workstations, the selectcode is the GPIB board number and defaults to 1 for the first GPIB SBus or PCI GPIB board and 2 for the second. On Linux workstations, the selectcode is the Logical Unit Number assigned to the GPIB interface. By default this is 7. See your GPIB interface documentation for more information.

Assigning Unit Names

When instruments are added to the active instrument list, the corresponding units are added to the Unit Table. The Unit Table contains an entry for each active unit.



Contains entries for each active unit

Configuration of HP4141:1

HP-IB Interface

◇ hpib

Instrument

Address ◁ 17 ▷

Unit Table

SMU1	SMU1
SMU2	SMU2
SMU3	SMU3
SMU4	SMU4
VS1	SMU6
VS2	SMU7
VM1	SMU8
VM2	VM2

The information listed for each unit consists of a unit's physical name matched to the unit's user-defined name. By default the user-defined name is the same as the unit's physical name.

The unit names assigned by IC-CAP to the physical units are listed in the description of individual supported instruments (see Chapter 1, “Supported Instruments,” in the *Reference* manual). For example, the physical name of the first SMU of an HP 4145 appears in the Unit Table as:

SMU1

The user-defined name defaults to the IC-CAP defined unit name:

SMU1

This value appears under the Unit Name column. All Unit Name fields can be edited. You may set this unit name to any name, but keep in mind that the user-defined name must be used when specifying the units in the Inputs and Outputs of a setup.

When a duplicate unit name is specified in the Unit Name field, a warning that a duplicate name exists appears on the window running the IC-CAP process. For example, if you have two unit names called SMU1, the following warning is issued:

WARNING: Unit name <SMU1> used 2 times.

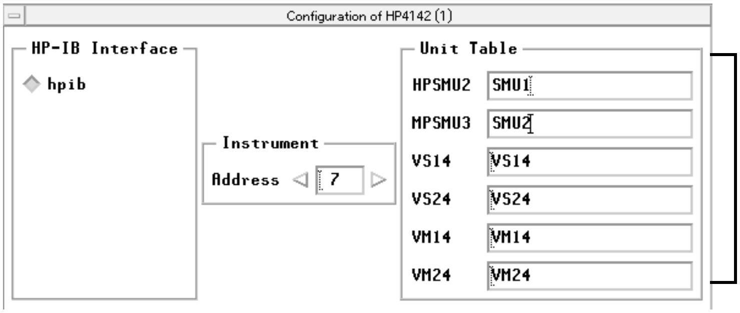
To assign unit names:

1 In the IC-CAP/Main window, click **Hardware Setup** on the toolbar. HP-IB.

2 Select the instrument to configure in the Instrument List.

3 Click **Configure**. Configure...

4 Enter the new name(s) in the Unit Table and click **OK**.



Configuration of HP4142 (1)

HP-IB Interface

◇ hpib

Instrument

Address ◁ 7 ▷

Unit Table

HPSMU2	SMU1
MPSMU3	SMU2
VS14	VS14
VS24	VS24
VM14	VM14
VM24	VM24

New names assigned

Using Unit Names in a Setup

Each IC-CAP model includes setup specifications. The unit names assigned by IC-CAP to the physical units are listed in the Unit field of the Input and Output tables displayed in the Measure/Simulate folder.

IC-CAP must be able to recognize the instruments and their corresponding units, so the unit names in the hardware configuration must match the unit names assigned by the program.

For example, to take a CV measurement using the Capacitance Meter and an HP 4141 DC Analyzer for an external DC bias, you specify the unit names *CM* and *SMU1* in the Unit fields of the Setup. Since the CM unit is from the HP 4271 and the SMU1 unit is from the HP 4141, both the HP 4271 and HP 4141 Instrument Options tables are available for this setup.

The options listed in the Instrument Options table vary for each instrument. Refer to the *Reference* manual for a list of all available instrument options, along with their descriptions, for each instrument supported in IC-CAP.

To specify unit names in a setup:

1 In the Model window, select **DUTs-Setups**.

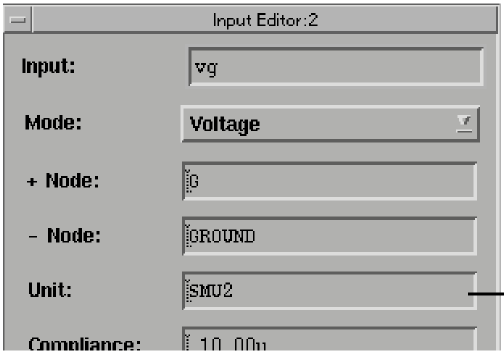
2 Select the setup.

3 Select **Measure / Simulate**.

4 Select the Input or Output table.

5 Click **Edit**.

6 In the dialog box, edit the Unit Table as necessary to match the unit names specified in the hardware configuration.



Input Editor:2

Input: vg

Mode: Voltage

+ Node: G

– Node: GROUND

Unit: SMU2

Compliance: 10.00u

Change Unit name

NOTE

Alternatively, you can edit the Unit field directly in the Input or Output table.

Adding a Ground Unit

A ground unit, which does not appear in the Unit Table of the hardware configuration, can be added to a setup. The ground unit is non-programmable and is available so that ground connections can be specified in a setup to reflect actual physical connections to ground.

The ground unit name is case insensitive and can be entered as `GND`, `GNDU`, or `GROUND` in the Unit field of the Input and Output tables. Do not assign these reserved names to any of the units in the Unit Table. For example, entering the name *GROUND* into a Unit Name field in the Unit Table causes a warning message to appear:

```
WARNING: ‘GROUND’ is a reserved Unit Name
```

Using Multiple Instruments

Some measurements may require more than one instrument. For example, a capacitance measurement using a CV meter and an external bias source requires a CV meter and a DC analyzer. In this case, both the CV meter and the DC analyzer must be connected and recognized by IC-CAP.

IC-CAP also allows measurements using multiple instruments of the same type. For example, you may perform a DC measurement using SMUs from two different HP/Agilent 4142 instruments. To do this, the two 4142 instruments must be set to different GPIB addresses, connected to the system, and recognized by IC-CAP. Assign unique unit names to the SMUs and VS/VMs for each instrument since this unit name, which is entered in the Setup specification, determines the instrument and unit to be used to bias an input or monitor an output.

NOTE

Because an GPIB interface is locked by IC-CAP while making measurement and calibration, it is possible to share a single GPIB interface with multiple users on an HP workstation. However, the GPIB interface on a Sun workstation is not sharable since this interface does not offer a device locking mechanism. Simultaneous access of the GP-IB interface on a Sun workstation is not supported by IC-CAP.

Specifying Instrument Options

After the unit names are specified in the Input and Output tables of the setup, you can edit options for each instrument.

Measurement instruments use both internal (system) sweep and user sweep modes. For a description of each mode and the instruments that support sweep modes, see “Sweep Modes and Input/Output Types” on page 203.

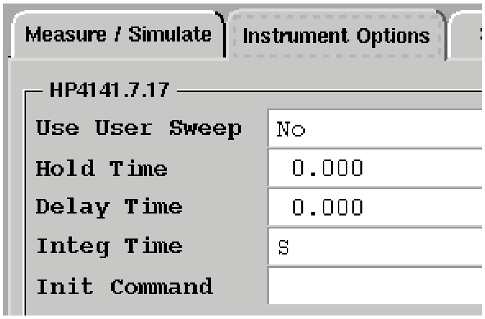
To view or edit instrument options:

1 In the Model window, select the **DUTs-Setups** folder.

2 Select the setup.

3 Select **Instrument Options**.

4 Edit the option fields directly in the table by selecting the field and typing the new option.



HP4141.7.17

Use User Sweep	No
Hold Time	0.000
Delay Time	0.000
Integ Time	S
Init Command	

Saving Instrument Options

You can save the instrument options in a file. The Save As command saves any active options tables. If no active instrument of the same type is available, IC-CAP keeps the options table information in memory. All inactive options tables are cleared when an active instrument is added to the Instrument Setup or when a measurement or calibration is made. The instrument options file is assigned the default suffix, *.iot* (Instrument Option Tables).

```
data

{

TABLE "HP4141.7.17"

{

element 0 "Use User Sweep" "No"

element 0 "Hold Time" " 0.000 "

element 0 "Delay Time" " 0.000 "

element 0 "Integ Time" "S"

element 0 "Init Command" ""

}

}
```

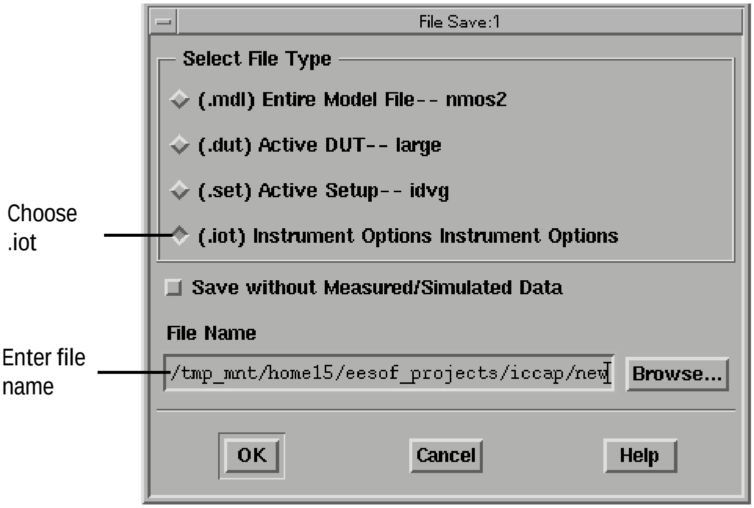
To save the instrument options to a file:

1 In the Model window, select the **DUTs-Setups** folder.

2 Select the setup and **Instrument Options**.

3 Select **File > Save As** and choose **(.iot) Instrument Options**.

4 Type a file name in the File Name field and choose **OK**.



File Save:1

Select File Type

◇ (.mdl) Entire Model File-- nmos2

◇ (.dut) Active DUT-- large

◇ (.set) Active Setup-- idvg

◆ (.iot) Instrument Options Instrument Options

□ Save without Measured/Simulated Data

File Name: /tmp_mnt/home15/eesof_projects/iccap/new

Performing a Calibration

Most instruments have internal or hardware calibration capability. For example, CV meters and network analyzers have an internal calibration menu to perform appropriate calibration for each test condition. Refer to Chapter 1, “Supported Instruments,” in the *Reference* manual for specific information regarding instrument calibration.

NOTE

Calibration data for CV meters remains in memory until IC-CAP terminates. At start up, you must calibrate a CV meter for the first measurement.

An HP 54120 series digital oscilloscope requires manual calibration on its front panel. Manual calibration means that an operator must be present to measure calibration standards. Refer to the description of this instrument family in this chapter for more information on calibration.

Software calibration is also supported by IC-CAP for several instruments. For example, a simple offset error reduction is possible with the HP 4271. More elaborate 12-term calibration is provided for all network analyzers.

One-port calibration is not supported for network analyzers because the 2-port conversion function used in extraction functions needs all four S-parameters. You can work around this by measuring uncalibrated data and then performing 1-port calibration in PEL.

To perform a calibration of the applicable instruments being used in the Setup:

1 In the Model window, select the **DUTs-Setups** folder.

2 Select the setup and **Measure / Simulate**.

3 Click **Calibrate**.

Performing a Measurement

After you have entered the instrument configuration and have done the necessary calibrations, you are ready to perform measurements.

To perform measurements for the active setup only:

1. In the Model window, select **DUTs-Setups**.
2. Select the DUT and the setup.
3. Select **Measure / Simulate**.
4. Click **Measure Setup**.

   The system status line in the IC-CAP/Status window displays:

   Measure in progress...

   When the measurement is done, the status line displays:

   IC-CAP Ready

   The IC-CAP measurement is complete.

To perform measurements for all setups in the active DUT:

1. In the Model window, select **DUTs-Setups**
2. Select the DUT.
3. Choose **Measure DUT**.

   The system status line in the IC-CAP/Status window displays:

   Measure in progress...

   When the measurement is done, the status line displays:

   IC-CAP Ready

   The IC-CAP measurement is complete.

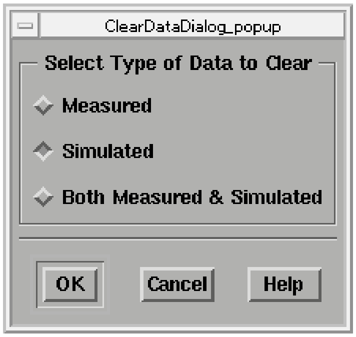
To clear measured data for a selected setup:

1. In the Model window, select **DUTs-Setups**.
2. Select the DUT and the setup.

3 Select **Measure / Simulate**.

4 Click **Clear** and choose the type of data to clear.

Choose type of data to clear



Viewing Results

You can view results of both the measured and simulated data in a graphic display. Measured data is represented by solid lines and simulated data is represented by dashed lines. For details on viewing results, see Chapter 10, “Printing and Plotting.”

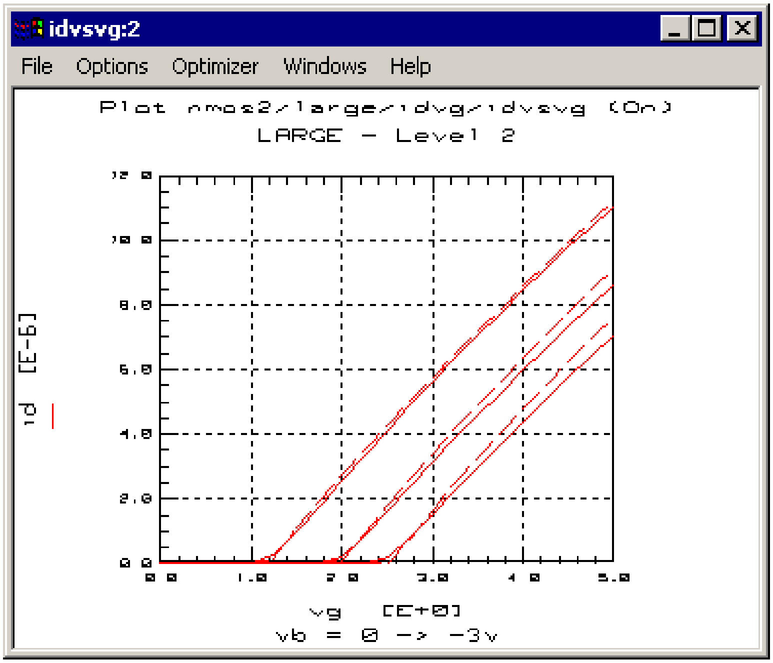
To view the results of the measurement:

1 In the Model window, select **DUTs-Setups**.

2 Select the DUT and the setup.

3 Select **Plots**.

4 Click **Display Plot** or **Display All**.



A Measurement Example

This example provides a general overview for performing a measurement. Using the supplied model *bjt_npn.mdl*, the example measures the *forward early voltage* of an npn bipolar device using the SMUs of the HP 4141 DC Analyzer.

NOTE

Before starting the measurement example, follow the procedures in "Opening a Model File" on page 76 to open the model *bjt_npn.mdl*.

Hardware Setup The HP 4141 contains four SMUs (Source/Measurement Units), each capable of sourcing and measuring voltage and current. For each SMU, a single triaxial cable carries the input signal to the device and the output signal from the device.

To connect the hardware:

1 Connect a cable from each of the four connectors (marked SMU1 - SMU4) on the back of the HP 4141 to each of the four connectors (marked SMU1 - SMU4) on the back of the HP 16058 Test Fixture.

2 Insert the bipolar npn test device into the HP 16058 Test Fixture and make the appropriate SMU to device lead connections.

3 Connect an GPIB cable from the GPIB connector on the back of the HP 4141 to the GPIB bus connector on the computer.

4 Make sure the GPIB address is set to a value that does not conflict with other GPIB addresses on the bus.

5 Turn on the HP 4141.

To setup the hardware:

1 In the IC-CAP/Main window, click **Hardware Setup**.

2 Select **Add Interface**.

3 A dialog box opens. In the Name field, type the name of the interface, `hpib`.

4 Choose **OK**.

IC-CAP finds the device and adds HP 4141 to the Active Instruments list.

When the HP 4141 is added to the active instrument list, the corresponding units are added to the Unit Table. The Unit Table contains an entry for each active unit.

To view unit names:

1 Select **Configure**. A dialog box opens, displaying the Unit Table and Instrument Address.

For this example, the units of the HP 4141 are: HP4141.7.23.SMU1, HP4141.7.23.SMU2, HP4141.7.23.SMU3, HP4141.7.23.SMU4, HP4141.7.23.VS1, HP4141.7.23.VS2, HP4141.7.23.VM1, and HP4141.7.23.VM2.

2 No changes are made. Choose **Cancel**.

Assigning Units to a Setup The next step explains how to use these unit names in a setup to specify a particular measurement. The bjt_npn model already includes setup specifications for the inputs *vb*, *vc*, *ve*, and *vs*, and the output *ic*. The assigned unit names are entered in the Unit field of the Input and Output tables. By specifying the unit names, you assign the HP 4141 to the setup *fearly*.

To specify unit names in a setup:

1 In the Model window, select **DUTs-Setups**.

2 Select the DUT dc and the setup **fearly**.

3 In the Measure/Simulate folder, select the Input table **vc**.

4 Click **Edit**.

5 In the dialog box, edit the Unit field by entering **SMU1**.

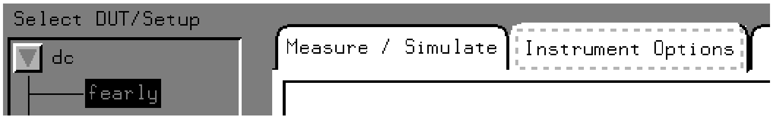
6 Choose **OK**.

7 Repeat steps 3 through 6, assigning SMU2 to the Unit field of the Input *vb,* SMU3 to the Unit field of the Input *ve,* and SMU4 to the Unit field of the Input *vs*. To monitor the output, assign SMU1 to the Unit field of Output *ic*.

Specifying Instrument Options The HP 4141 has four instrument options that may be set before taking a measurement.

To view instrument options:

1 Select **Instrument Options**.



2 For this example, the default values of these options are used.

- Since the option *Use User Sweep* is set to *No*, the main sweep Input *vc* (Sweep Order = 1) is swept from 0.000 to 5.000 volts in 21 steps using an internal instrument sweep.

NOTE

When *Use User Sweep* is set to Yes, the measurement is taken from 0.000 to 5.000 volts in 21 steps, with each voltage being set separately or point-by-point. A measurement taken with a user sweep is slower than the same measurement taken with an internal instrument sweep. However, the advantage for using a user sweep is increased flexibility in the types of measurements that can be taken.

- The *Hold Time* option is set to 0.000. This means that the instrument waits 0.000 seconds before starting the main sweep.
- The *Delay Time* option is set to 0.000. This means that the instrument waits 0.000 seconds before the measurement is taken at each step in the sweep.
- The *Integration Time* is set to S. This means that the integration time of the HP 4141 is short.

3 Since no changes are made, choose **Cancel**.

To take a measurement:

1 Select **Measure / Simulate**

2 Click **Measure**.

The system status line in the IC-CAP/Status window displays:

Measure in progress...

When the measurement is done, the status line displays:

IC-CAP Ready

The IC-CAP measurement is complete.

To view results:

1 Select **Plots**
2 Click **Display Plot**. A plot of the measured data displays in a separate window.

Sweep Modes and Input / Output Types

Measurement instruments use both internal (system) sweep and user sweep modes. This section describes each of these modes and the instruments that support sweep modes. For additional information on defining setups, refer to “Simulation Types” on page 235.

Sweep Types

The sweeping of a source for an instrument is controlled by the instrument or by IC-CAP. This applies only for the instrument to which the primary unit belongs. The primary unit is a unit with a sweep order *1*. The instrument with the primary unit is called the primary instrument.

Internal (System) Sweep A primary instrument can perform its internal sweep when the *Use User Sweep* option of that instrument is set to *No.* Some instruments, such as the HP 4271, cannot perform a sweep measurement and do not have this option. A spot measurement with the internal sweep option enabled is converted to a single point measurement with the user sweep. It is impossible to perform a single spot internal sweep. Internal sweep is much faster than the user sweep (described in the next section), but not all sweep types are supported by the internal sweep of a particular instrument.

User Sweep When the *Use User Sweep* is *Yes* for a primary instrument, IC-CAP performs a set of spot measurements to make up a single sweep measurement. Even though all supported instruments except time-domain instruments perform spot measurement, instruments like Network Analyzers need to use the internal sweep for calibrated data. Most sweep types are possible with user sweep because IC-CAP controls each point directly. However, a user sweep is much slower than an internal sweep.

Multiple Instruments

When multiple instruments are involved in a measurement setup, non-primary sweep instruments use the user (spot-mode) sweep regardless of how the *Use User Sweep* option is set. The sweep capabilities of the primary sweep instrument and the nature of the measurement determine whether internal or user sweep is appropriate.

When the primary instrument has internal sweep capability and other instruments are only used for non-primary sweeps or constants, the internal sweep for the primary instrument is possible. This includes the case where a network analyzer sweeps its frequency as a primary sweep while a DC bias is given as a secondary sweep from some DC instruments.

When multiple instruments have to synchronize at each measurement point, the user sweep must be used because these instruments don't know about each other. Only in this fashion can IC-CAP control them properly. An example is to measure both S parameters and DC currents at each frequency point.

Supported Internal Sweeps

The following table and Table 9 list the inputs, outputs, and internal sweeps that are possible with each instrument, with the following exceptions:

- Several time-domain pulse parameters can be extracted with Output T, like RISETIME.
- The 8510A supports only LIN sweep.
- 54120 Series includes HP 54121/122/123/124.
- The 54122 does not support V-TDR Input, because the necessary pulse generator is not available in this instrument.
- Two-port data is taken as S parameters, then converted by IC-CAP to other parameters.
- The pulse generators have no measurement capability, thus no Output modes.

For more information, refer to the individual instrument descriptions in the *Reference* manual.

Table 7 Internal Sweeps for DC and CV Instruments

Instrument Type			**DC Analyzer**			**CV Meter**					
Model Number			**4140**	**4141 / 42 / 45**	**4155 / 56**	**4194**	**4271**	**4275**	**4280**	**4284 / 85**	**E4980A**
Input	**Mode**	V	X	X	X	X		X	X	X	
		I		X	X					X	
		F									
		T									
	Type	LIN	X	X	X	X			X		
		LOG		X							
		SYNC		X	X						
		LIST									
		CON	X	X	X	X	X	X	X	X	
		TDR									
Output	**Mode**	V		X	X						
		I	X	X	X						
		C†				X	X	X	X	X	X
		G†				X	X	X	X	X	X
		R†								X	X
		X†								X	X
		SHYZKA									
		T									

†

If Output Mode is...	then Measurement is...
C	Cp-Gp (only Cp read)
G	Cp-Gp (only Gp read)
C, G	Cp-Gp (both Cp and Gp read)
R	Cs-Rs (only R read)
C, R	Cs-Rs (both Cs and Rs read)
X (type Y)	G-B (complex data of form G +j*B read)
X (type Z)	R-X (complex data of form R + j*X read)

Table 8 Internal Sweeps for Noise Instruments

Instrument Type			**Dynamic Signal Analyzer**	
Model Number			**HP / Agilent 35670A**	
			Source	**Channel**
Input	Mode	V	X	
		I		
		F		X
		T		
	Type	LIN	X	X
		LOG	X	
		SYNC		
		LIST	X	
		CON		
		TDR		
Output	Mode	V		
		I		
		C		
		G		
		SHYZKA		
		T		
		N		X

Table 9 Internal Sweeps for Network Analyzers and Time-Domain Instruments

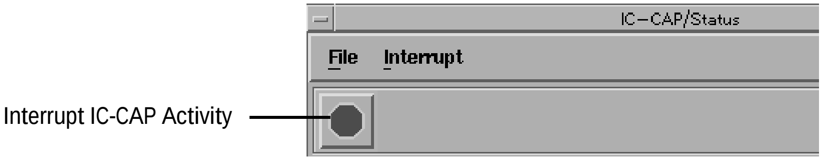
Instrument Type			**NWA**		**Oscilloscope**			**Pulse Gen**
Model Number			**3577**	**8510, 8702, 8719, 8720, 8722, 8753**	**54120 Series**		**54510**	**8130, 8131**
Input	Mode	V			X			
		I						
		F	X	X				
		T				X	X	
	Type	LIN	X	X		X	X	
		LOG		X				
		SYNC						
		LIST		X				
		CON	X	X				
		PULSE			X			X
		TDR			X			X
Output	Mode	V			X	X	X	
		I						
		SHYZKA	X	X				
		T			X	X	X	

Aborting a Measurement

You can interrupt a measurement from the Status window. If you abort a measurement while an internal system sweep is in progress, the measurement in IC-CAP is aborted, but the instrument continues to step through its sweep values until the sweep is completed. If another IC-CAP measurement using this instrument is attempted before the sweep is completed, IC-CAP waits until the sweep is done before performing the measurement.

To abort a measurement:

In the IC-CAP/Status window, click Interrupt IC-CAP Activity.



You can use the Tools menu in the Hardware Setup window to control some measurement activities. For example, you can stop an internal sweep by sending a command byte to instruments on the bus.

To stop an internal sweep:

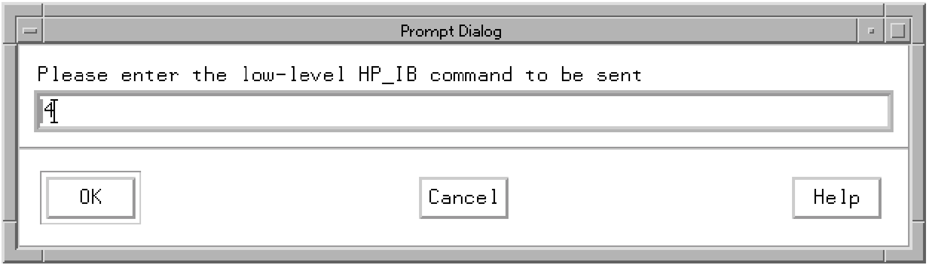
1 Open the Hardware Setup window.

2 Select **Tools > Send / Receive**.

3 Select **Send Byte**.

4 In the dialog, enter the low-level GPIB command 4 to clear a single device or a `20` to clear all instruments.

5 Click **OK**.


Prompt Dialog

Please enter the low-level HP_IB command to be sent

4

NOTE

Do not send *SIGKILL* to the IC-CAP process unless that is the only known way to abort a measurement.

Speeding Up Repetitive Measurements

To ensure the safest possible instrument operation, IC-CAP performs checking and instrument initialization at the beginning of each measurement. When the same measurement is performed repeatedly, this checking and initialization is unnecessary. Repeated measurements, such as might be programmed within an IC-CAP Macro, can be accelerated if only the first such measurement is subject to this checking and initialization.

This section explains an IC-CAP feature called Fast Measurement that improves the speed of repetitive measurements. These Fast Measurement techniques minimize the use of I/O and instrument operations that are unnecessary when a measurement is repeated.

Using Fast Measurement

You can eliminate a sizable amount of overhead associated with instrument setup and initialization by using a Fast Measurement feature. The feature can only be used when certain criteria (listed below) are met. Although this set of conditions may seem somewhat restrictive, they are necessary to provide reasonable reliability when benefiting from the flexibility of certain IC-CAP features, such as expression evaluation and the availability of different Instrument Option values in different Setups.

Fast Measurement Criteria

In order to enable Fast Measurement and ensure reliable operation, the following criteria must be met before measuring:

- Create a variable named *MEASURE_FAST* (a reserved variable name) and set its value to Yes. Another variable, *NO_ZEROING*, is associated with a higher level of optimization and can be used for second level speedup.
- The preceding measurement must have succeeded so that complete instrument initialization has taken place.
- The setup being measured must be the same as in the preceding measurement.
- The Instrument Options values must generally evaluate to the same values as they did during the preceding measurement.
- The functions offered by the Hardware Manager must not have been used since the preceding measurement. For example, deleting an instrument from the Active List disables Fast Measurement for the first measurement that follows the deletion.

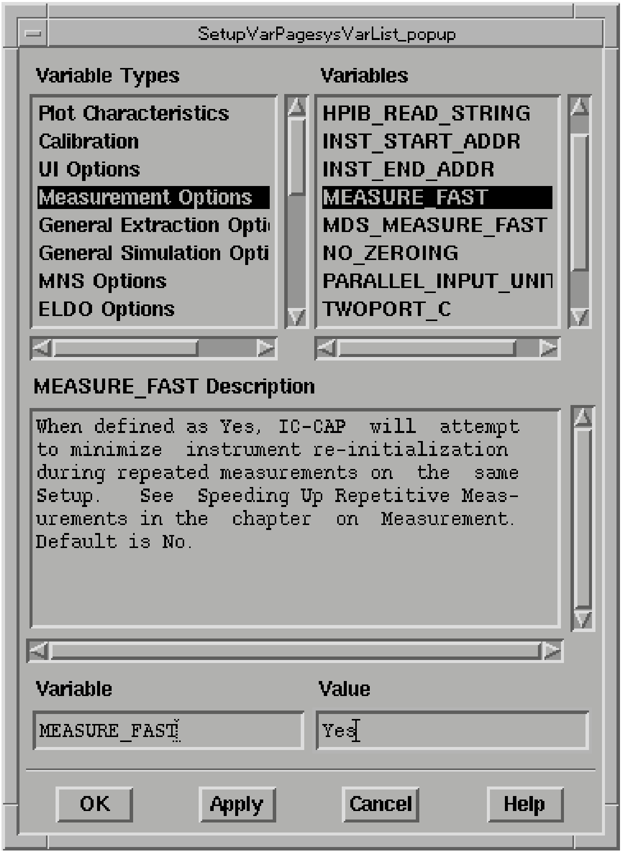
When these criteria are not met, IC-CAP reverts to its normal manner of complete instrument initialization prior to each measurement. Note that Fast Measurement is not disabled by a change in Input specification.

Enabling Fast Measurement

This section explains the steps necessary for enabling Fast Measurement and explains how to temporarily disable Fast Measurement and force IC-CAP to fully initialize instruments during the next measurement. Two types of Fast Measurement are available: First Level and Second Level.

To enable first level speedup:

1. Create an IC-CAP variable named `MEASURE_FAST`.
   - To perform Fast Measurement for a particular setup, create the variable at the setup level.
   - To perform Fast Measurement globally for all setups you repeatedly measure, create the variable at the system level.
2. Set the value of MEASURE_FAST to `YES`.



MEASURE_FAST Description

When defined as Yes, IC-CAP will attempt to minimize instrument re-initialization during repeated measurements on the same Setup. See Speeding Up Repetitive Measurements in the chapter on Measurement. Default is No.

Variable: MEASURE_FAST

Value: Yes

The functions offered by the Hardware Manager must not have been used since the preceding measurement. For example, deleting an instrument from the Active List disables Fast Measurement for the first measurement that follows this operation. The section “Requesting Complete Initialization on the Next Measurement” on page 215 explains a simple, recommended way to use the Hardware Manager for ensuring the next measurement undertakes complete instrument initialization.

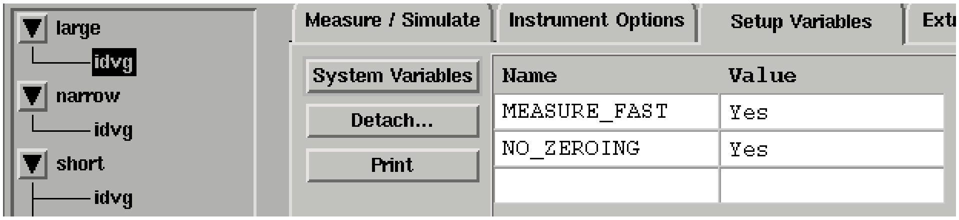
NOTE

MEASURE_FAST skips instrument initialization. This could be useful if an instrument is controlled additionally with a macro program. For simple GPIB operations with library functions, refer to Chapter 9, “Using Transforms and Functions.” The Init Command instrument option is another method to send an arbitrary control command per setup to an instrument.

Second level speedup applies only when the conditions necessary for first level are also met. Second level speedup does not yield speed improvements as substantial as those from first level speedup.

To enable second level speedup:

1 Create an IC-CAP variable named `MEASURE_FAST`.
   - To perform Fast Measurement for a particular setup, create the variable at the setup level;
   - To perform Fast Measurement globally for all setups you repeatedly measure, create the variable at the top level.
2 Set the value of MEASURE_FAST to `YES`.
3 Create an additional variable named `NO_ZEROING`.
4 Set the value of NO_ZEROING to `YES`.



Name	Value
MEASURE_FAST	Yes
NO_ZEROING	Yes

NOTE

Setting the value of NO_ZEROING to YES prevents the program from disabling or zeroing instruments prior to each measurement. As a safety measure, IC-CAP still ensures that each instrument involved in the measurement ceases sourcing bias or other types of signals after the measurement concludes.

Forced Instrument Initialization

In some cases it might be inappropriate for IC-CAP to provide Fast Measurement, but it does so nonetheless. This section lists the cases where this can happen and explains how to temporarily override Fast Measurement and ensure complete instrument initialization.

IC-CAP may not detect certain instances of calibration that have been invalidated by changed Input specification. Input specifications can be changed explicitly (with the mouse and keyboard), or implicitly by a macro (such as, a macro that alters the values of IC-CAP variables used within expressions in the Input editors). In such cases, when Fast Measurement has been requested, IC-CAP may proceed with the measurement without warning about the changes in the Input specification.

IC-CAP always detects and downloads changes in Instrument Options settings. Changes here result in warnings about invalid calibration when appropriate. When an Input or Output is added to, or removed from a Setup, it might be appropriate for IC-CAP to fully re-initialize the instruments used by that Setup, even if Fast Measurement is requested. However, IC-CAP will not do so, unless you use a method such as the one described in the next section.

Requesting Complete Initialization on the Next Measurement

Here is an easy way to ensure that IC-CAP undertakes complete instrument initialization at the beginning of the next measurement. From the Hardware Manager menu, execute the function *Disable Supplies*. This function ensures that all instruments listed in the active list will cease sourcing bias and other signals to the DUT.

You must request complete re-initialization with this method any time a setup has been modified in any way except when changing sweep end points. You should also use this method when you have altered an instrument’s settings though any of the following:

- The instrument’s front panel
- IC-CAP’s GPIB Analyzer
- Arbitrary instrument I/O in Programs or Transforms
- Additional software programs

6
Simulating

Simulation is the process of generating device and circuit output characteristics based on an available model. The accuracy of a model determines how well simulated characteristics agree with the real physical behavior of devices and circuits. The following figure illustrates the IC-CAP simulation process.

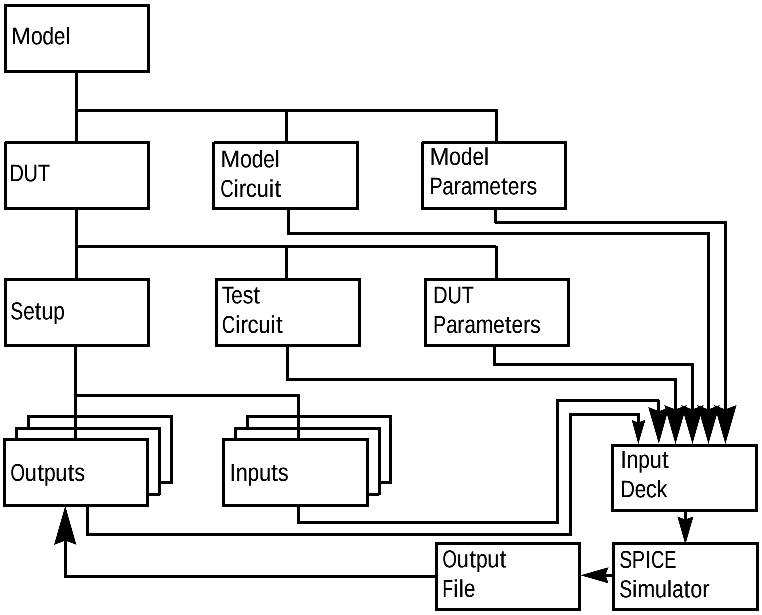


Figure 12 Simulation Flow Diagram

Five types of simulators can be used in IC-CAP:

- SPICE simulators
- HSPICE simulator
- Saber simulator
- Microwave Nonlinear simulator (MNS)
- Advanced Design System simulator (ADS)

NOTE

The PC version of IC-CAP supports ADS version 2002 or newer. Older versions of ADS can not be used with the PC version of IC-CAP.

NOTE

Simulators are provided with IC-CAP as a courtesy to IC-CAP users and are not supported by Agilent Technologies except for the ADS simulator.

Interfaces, known as *templates*, are provided to the simulators (see the following table).

Table 10 IC-CAP Supported Simulators and Corresponding Template Names

SIMULATOR	TEMPLATE NAME
UCB SPICE2G.6	spice2
UCB SPICE3E2	spice3
HPSPICE	hpspice
HSPICE	hspice
ELDO	eldo
Microwave Nonlinear Simulator (MNS)	mns
Saber	saber
SPECTRE	spectre (native circuit syntax) spectre443 (spice circuit syntax)
Advanced Design System (ADS)	hpeesofsim (native circuit syntax)

The IC-CAP simulator interface is an open system, so you can add any simulator similar to one of the templates. For details, refer to "Adding a Simulator" on page 252.

Selecting a Simulator

You can set a simulator in one of three ways:

- By specifying a default startup simulator (setting DEFAULT_SIMU variable)
- By specifying a simulator for a specific model, DUT, or setup (setting SIMULATOR variable)
- By specifying a simulator without a variable (using *Select Simulator* command)

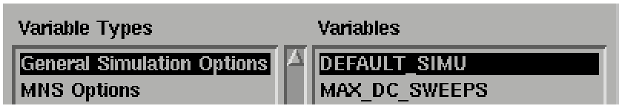
Specifying a Default Startup Simulator

You can specify a simulator as the default on startup by setting the DEFAULT_SIMU variable to one of the simulators. This setting is only effective if set at the global level and is overridden if a different simulator is specified by setting a SIMULATOR variable or applying the Select Simulator command. When you exit the program, the DEFAULT_SIMU setting is saved in the *.icconfig* file. If this variable is not defined, the default on startup is *spice2*.

To specify a default startup simulator:

1. In the IC-CAP/Main window, select **Tools** > **System Variables**.
2. In the System Variables window, click **System Variables**.
3. In the dialog box, select **General Simulation Options** as the Variable Type and select the **DEFAULT_SIMU** variable.

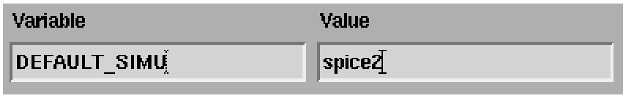
Select Variable Type and Variable



Variable Types	Variables
General Simulation Options	DEFAULT_SIMU
MNS Options	MAX_DC_SWEEPS

4. Enter the simulator name in the Value field and choose **OK**.

Enter simulator type



Variable	Value
DEFAULT_SIMU	spice2

NOTE

You can type the variable name and value in the System Variables window directly without going through the dialog box.

Specifying a Simulator for a Specific Model, DUT, or Setup

Some models require or perform better with a specific simulator. In these model files, you can specify a simulator for a model, DUT, or setup by setting the SIMULATOR variable. This allows you to use different simulators for different models, DUTs or setups, since a SIMULATOR variable can be specified at any level. The model files for which the SIMULATOR variable is defined are shown in the following table.

Table 11 Model Files with Predefined Simulators

Model File Name	SIMULATOR Value
bjt_ft.mdl	hpspice
bjt_ncehf.mdl	hpspice
HPEEbjt2.mdl	mns
hpsimbjt_ncehf.mdl	hpeesofsim
hpsimbjt_nhf.mdl	hpeesofsim
hpsimnpn.mdf	hpeesofsim
hpsimvbic.mdl	hpeesofsim
mnsnpn.mdl	mns
mxt3t_npn.mdl	mns
mxt4t_npn.mdl	mns
mxt504_npn.mdl	hpeesofsim
sabernpn.mdl	saber
spectre_ncehf.mdl	spectre
spectrenpn	spectre
vbic_npn	mns

Table 11 Model Files with Predefined Simulators (continued)

Model File Name	SIMULATOR Value
HPDiode.mdl	mns
juncap.mdl	spice3
pn_diode.mdl	spice2
HPEEhemt1.mdl	mns
CGaas1.mdl	hpspice
CGaas2.mdl	hpspice
CGaashf.mdl	hpspice
CGaashfax.mdl	hpspice
HPEEfet3.mdl	mns
HPRootFet.mdl	mns
hpsimHPEEfet3.mdl	hpeesofsim
UCBGaas.mdl	spice3
UGaashf	spice3
lc.mdl	spice3
sabercirc.mdl	saber
sys110_verify.mdl	hpspice
hnmos6.mdl	hspice
hnmos28.mdl	hspice
hpmos28.mdl	hspice
HPRootMos.mdl	mns
sabernmos.mdl	saber
noise_simu.mdl	spice3
bjt_1f_noise.mdl	hpspice
mos_1f_noise.mdl	spice3
mnsopamp.mdl	mns

Table 11 Model Files with Predefined Simulators (continued)

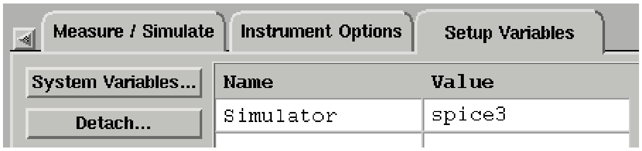
Model File Name	SIMULATOR Value
opamp.mdl	hpspice
bjt_ncehfp.mdl	hpspice
BSIM3_DC_CV_Measure.mdl	spice3
BSIM3_DC_CV_Extract.mdl	spice3
BSIM3_RF_Measure.mdl	spice3
BSIM3_RF_Extract.mdl	spice3
BSIM3_AC_Noise_Tutorial.mdl	spice3
BSIM3_CV_Tutorial.mdl	spice3
BSIM3_DC_Tutorial.mdl	spice3
BSIM3_Temp_Tutorial.mdl	spice3
BSIM3_DC_CV_Finetune.mdl	spice3
BSIM4_DC_CV_Measure.mdl	spice3
BSIM4_DC_CV_Extract.mdl	spice3
BSIM4_RF_Measure.mdl	spice3
BSIM4_RF_Extract.mdl	spice3
BSIM4_DC_CV_Tutorial.mdl	spice3
BSIM4_DC_CV_Finetune.mdl	spice3

When a simulation is performed, IC-CAP looks for the SIMULATOR variable first, and if found, makes that the active simulator. The *Select Simulator* dialog box changes to reflect the name of the active simulator. If the SIMULATOR variable is not defined, IC-CAP uses the simulator displayed in the *Select Simulator* dialog box.

To set a simulator for a specific model, DUT, or setup:

1. Open the appropriate model, DUT, or setup folder and click the **Variables** tab.
2. Type `SIMULATOR` in an empty variable Name field and type the name of the simulator in the corresponding Value field.

Enter the variable and simulator



Name	Value
Simulator	spice3

NOTE

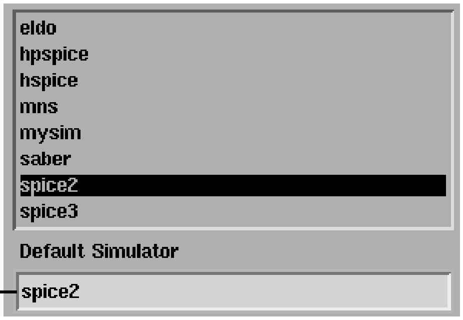
To use a different simulator after one has been specified by the SIMULATOR variable, reset the simulator using the *Select Simulator* command.

Specifying a Simulator without a Variable

The Select Simulator command sets the simulator to be used for all simulations performed in the current session, except when simulating a model, DUT, or setup for which a SIMULATOR variable has been defined.

To set a simulator without using a variable:

1. In the IC-CAP Main window, select **Tools > Select Simulator**. A dialog box listing simulator names appears.



Default Simulator: spice2

The active simulator

2. Select a simulator name or type the name of a simulator you have linked with IC-CAP and choose **OK**.

Specifying Inputs and Outputs

When running a simulation, IC-CAP builds the simulation input deck using the circuit description and the input and output specifications. The circuit description provides all of the model information. The input and output specifications provide the input stimuli and requested output data, as well as the information needed to determine the type of simulation being performed.

Specifying inputs and outputs is independent of the type of simulation being performed. When specifying input for a simulation, you enter the sweep mode in the Mode field of the Input table, and the node connections. The input fields change, depending on the type of mode specified. For details, refer to “Simulation Types” on page 235.

Conventions for Connecting Nodes

Be aware of the convention used for node connections when sourcing voltage and current.

- When you specify an Input Mode of *V*, the +Node and –Node fields are available in the Input table. In this case, the +Node is considered to be the positive side of the voltage source and the –Node is the negative side.
- When you specify an Input Mode of *I*, the To Node and From Node fields are available in the Input table. Current flows from the From Node to the To Node.

When IC-CAP builds the simulation input deck, the program creates the source name by concatenating the mode character, the first three characters of the +Node (or To Node) and the first three characters of the –Node (or From Node). These source names are used in the simulation input deck to specify the sweeps and constants. Specified outputs may also reference these names.

Source names are limited to 8-characters. This limit can cause problems in a simulation if, for example, two inputs are specified as follows:

Mode = V		Mode = V
+Node = BASE	and	+Node = BASE
–Node = EMITTER		–Node = EMITTER2

From this input, IC-CAP creates the same source names: VBASEMI and VBASEMI. You can avoid this potential conflict with source names by choosing node names in circuit descriptions carefully. When choosing node names with more than three characters, make sure that the first three characters are unique with respect to the first three characters of any other node names.

NOTE

When you enter an invalid node name, such as K in any of the input and output node fields and try to simulate, the program sends an error message:

```
ERROR: Invalid Input node name K used
ERROR: Unable to simulate.
Check the Input and Output specifications.
```

Specifying Parameter or Variable Sweeps

In addition to the list of valid sweep modes for each simulation type listed, you can sweep parameter values and variables. For example, you can generate a family of beta versus IC curves by using the BF parameter of the bipolar transistor model as the step input or you can sweep the operating temperature variable TEMP to analyze temperature effects.

NOTE

You can set the value for a constant or values for sweeps of the simulation temperature by adding the TEMP variable to the variable table and creating an input (Mode = Parameter and Name = TEMP) in the setup.

To sweep parameters or variables:

1 In the Model window, select the DUT and setup.

2 Select **Measure / Simulate**.

3 Click **New Input**.

4 In the Mode field, toggle to **Parameter**.

5 Specify a parameter or variable by entering its name in the Param Name field.

6 Enter all other necessary information and choose **OK**.

A parameter sweep is a valid input mode for all simulation types. Specifying parameter sweeps may differ for devices and circuits depending on the type of simulator being used. For examples of simulator-specific parameter sweeps, refer to the simulator chapters in the *Reference* manual:

- “SPICE Parameter Sweeps"
- “Saber Parameter Sweeps"
- “MNS Parameter Sweeps"
- “ADS Parameter Sweeps"

Hierarchical Parameter Sweeps

You can perform parameter sweeps when using hierarchical models. However, when you sweep a parameter from a model lower than the level from which the simulation is being performed, you must specify the complete path name of the parameter in the Name field of the input.

Sychronized List Sweeps (LSYNC)

To synchronize a parameter sweep with other parameter sweeps, set the sweep type to LSYNC (*Synchronized List* when using the *Edit* feature on an input). Just like a normal SYNC sweep, you must specify the name of the master sweep by entering it in the *Master Sweep* field.

Unlike a normal SYNC sweep, the entries are not limited to an offset and a multiplier. The LSYNC sweep enables you to specify an arbitrary list of points. IC-CAP automatically provides the required number of points after the master sweep is set. If the number of points in the master sweep changes, simply click on the LSYNC *Master Sweep* field to update the number of points in the input.

One application of LSYNC sweeps is to simulate an arbitrary collection of device Lengths and Widths.

If a master sweep and multiple LSYNC sweeps are saved to an MDM file, they can only be imported into a setup where the same sweeps are either all CON sweeps or the sweeps are synchronized using LSYNC. One sweep cannot be a LIN sweep and another one be a CON sweep, even if that combination exist in the MDM file. To use a LIN sweep with a CON sweep, use LSYNC to synchronize the CON sweep to the LIN sweep and enter the same value for all list points.

NOTE

The LSYNC sweep type is only available with Parameter sweeps.

The LSYNC sweep is not supported with the Saber simulator.

Simulating Open Circuits

IC-CAP uses the OPEN_RES variable to handle any floating nodes. This variable allows an open circuit to be simulated as a large resistance. The value of the resistance is equal to the value of the OPEN_RES variable. A resistor of this magnitude is automatically connected to all external circuit nodes not connected to a specified source. When the OPEN_RES variable is not specified, a current source set to zero (0) amps is used instead. However, using the current source may cause simulation convergence problems.

Performing a Simulation

You can perform a simulation on the active setup or on all setups in the active DUT.

- To perform a simulation on the active setup, select the setup and click **Simulate Setup**.
- To perform a simulation on all setups in the active DUT, select the DUT and click **Simulate DUT**.

Using the Simulation Debugger

When a simulation fails, the program sends an error message:

```
Simulation Failed: Data Unchanged Use Simulation Debugger in
Utilities Menu for more information
```

The Simulation Debugger is a useful tool for determining why a simulation failed.

The SPICE-type simulators accept an input deck that contains both the circuit description and analysis commands.

The Saber simulator requires two separate decks. The Saber input deck, displayed in the Input editor, contains the circuit description, written in the MAST modeling language. The Saber command deck, displayed in the Command editor, contains the analysis commands to be performed by the simulator. The Command editor is only used with the Saber simulator.

The input editor displays the input information used in the simulation. For Saber simulations, the command editor information is used also. You can quickly see how changes would affect your results by changing the input (and for Saber, command) files, performing a manual simulation, and observing the results in the output editor. For more information, refer to "Using the Manual Simulate Function" on page 233.

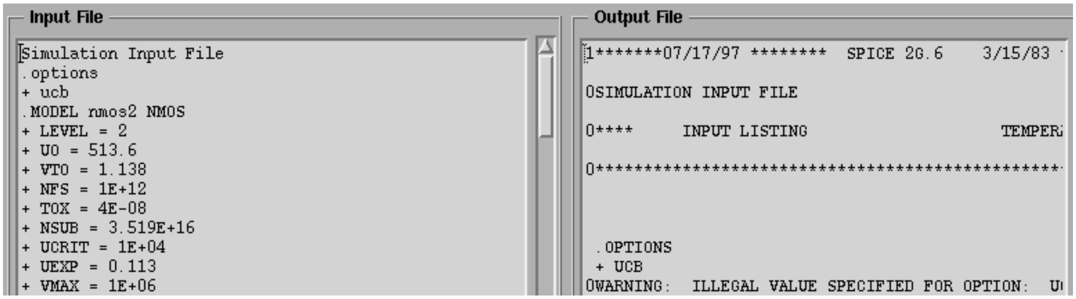
In many cases, the output text file includes the error messages displayed when the simulation fails. You cannot edit the output text file.

To use the Simulation Debugger:

1 In the IC-CAP/Main window, click **Simulation Debugger**.

  When the window opens, the Input, Command, and Output editors are blank.

2 Initiate the simulation in the Model window. The program sends the input deck and output text files to the input and output editors of the Simulation Debugger.



```
Simulation Input File
.options
+ ucb
.MODEL nmos2 NMOS
+ LEVEL = 2
+ UO = 513.6
+ VTO = 1.138
+ NFS = 1E+12
+ TOX = 4E-08
+ NSUB = 3.519E+16
+ UCRIT = 1E+04
+ UEXP = 0.113
+ VMAX = 1E+06
```

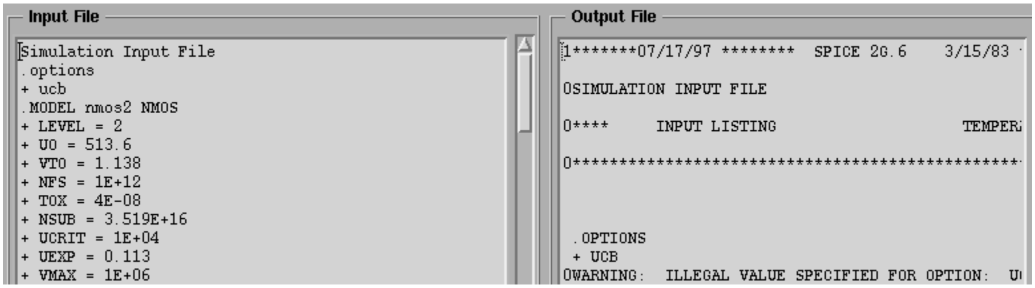
```
1*******07/17/97 *********  SPICE 2G.6    3/15/83 ·

0SIMULATION INPUT FILE

0****     INPUT LISTING                          TEMPER

0************************************************·



 .OPTIONS
 + UCB
0WARNING:  ILLEGAL VALUE SPECIFIED FOR OPTION:  U
```

Using the Manual Simulate Function

The *Manual Simulate* function simulates the input deck displayed in the Input editor. For example, you can perform a manual simulation after changing some parameter values or sweep values directly in the input file deck, without having to change these values in the IC-CAP Circuit definition, parameter tables, or input and output specifications.

To execute a manual simulation:

1. In the Simulation Debugger window, edit the input and command decks displayed in the Input and Command editors.
2. Select **File > Manual Simulate**.

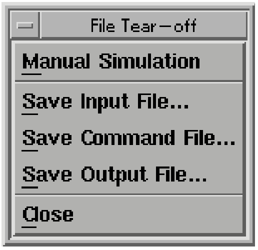


```
Simulation Input File
.options
+ ucb
.MODEL nmos2 NMOS
+ LEVEL = 2
+ UO = 513.6
+ VTO = 1.138
+ NFS = 1E+12
+ TOX = 4E-08
+ NSUB = 3.519E+16
+ UCRIT = 1E+04
+ UEXP = 0.113
+ VMAX = 1E+06
```

```
1*******07/17/97 ********   SPICE 2G.6    3/15/83

0SIMULATION INPUT FILE

0****     INPUT LISTING                        TEMPER

0************************************************



 .OPTIONS
 + UCB
0WARNING:  ILLEGAL VALUE SPECIFIED FOR OPTION:  U
```

Edit values directly in tables

Then, execute manual simulation



Saving the Simulation Debugger Files

You can save the input, command, and output files displayed in the individual editors of the Simulation Debugger.

To save Simulation Debugger files:

1 Select **File > Save** and choose the appropriate command for the type of file you want to save.

- Input File
- Command File (for Saber simulation only)
- Output File

2 In the dialog box, enter a filename.

- If you enter a filename only, the file is saved to the current working directory.
- If you want to save the file to another directory, enter the full path and filename.

Simulation Types

IC-CAP recognizes eight basic simulation types. This section describes the input and output specifications required for a valid setup for each simulation type. Each of the eight types can be categorized as either a *standard simulation* or a *special simulation*.

NOTE

Special simulation types are not directly available in the SPICE simulators. IC-CAP builds the additional circuitry required in the simulator input files to perform the simulation.

Standard simulation types are available in the SPICE simulators. The standard simulation types are:

- DC
- AC
- Transient
- Noise

The special simulation types are:

- Capacitance Voltage (CV)
- 2-Port (S,H,Y,Z,K,A parameter)
- Time-Domain Reflectometry (TDR)
- Harmonic Balance

If you attempt a simulation with input and output specifications that do not match any of the eight simulation types, the simulation is not attempted and the following error message appears:

```
ERROR: Unable to simulate.
Check the Input and Output specifications.
```

The simulators interfaced with IC-CAP may only support a subset of these simulation types and IC-CAP may not support all of the analysis types available in a particular simulator. For example, non-electrical analyses for Saber is not supported, and Harmonic Balance is supported only on the ADS simulators.

For information on the types of simulation that each simulator supports, refer to these chapters in the *Reference* manual:

- Chapter 3, “SPICE Simulators”
- Chapter 4, “SPECTRE Simulator”
- Chapter 5, “Saber Simulator”
- Chapter 6, “MNS Simulator”
- Chapter 7, “ADS Simulator”

Simulation Input and Output Requirements

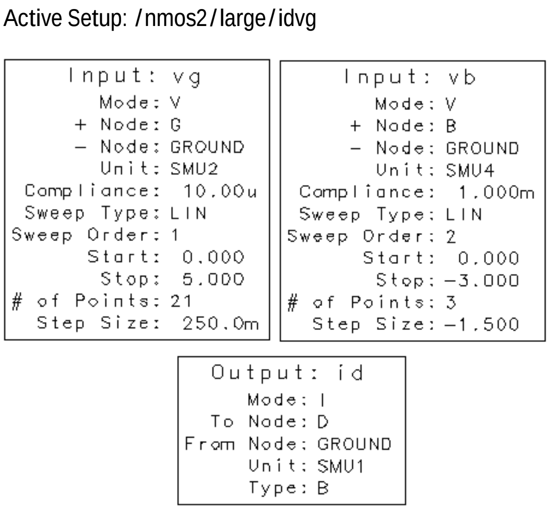
The following input and output requirements for each simulation type are also required for the corresponding measurement types.

DC Simulation

The following table describes the input and output specifications required for a valid setup for DC simulation and the following figure shows an example of input and output specifications for a MOSFET *id versus vg* setup.

Table 12 Input and Output Requirements for a DC Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS
V	LIN, LOG, LIST, SYNC, CON	V, I
I	LIN, LOG, LIST, SYNC, CON	V, I



Active Setup: /nmos2/large/idvg

```
         Input: vg
          Mode: V
        + Node: G
        - Node: GROUND
          Unit: SMU2
    Compliance:  10.00u
    Sweep Type: LIN
   Sweep Order: 1
         Start:  0.000
          Stop:  5.000
   # of Points: 21
     Step Size:  250.0m
```

```
         Input: vb
          Mode: V
        + Node: B
        - Node: GROUND
          Unit: SMU4
    Compliance:  1.000m
    Sweep Type: LIN
   Sweep Order: 2
         Start:  0.000
          Stop: -3.000
   # of Points: 3
     Step Size: -1.500
```

```
     Output: id
          Mode: I
       To Node: D
     From Node: GROUND
          Unit: SMU1
          Type: B
```

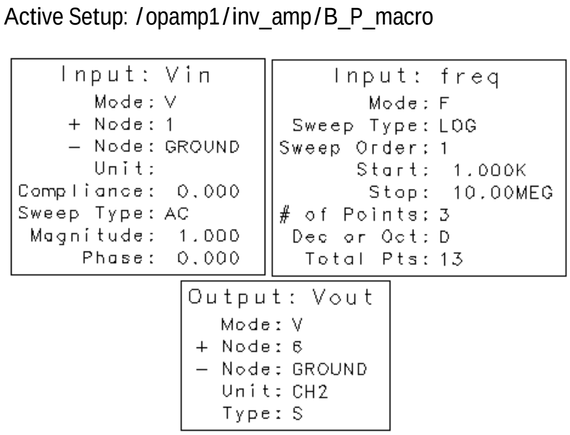
Figure 13 Example Input and Output, DC Simulation

AC Simulation

The following table describes the input and output specifications required for a valid setup for AC simulation. The following figure shows an example of input and output specifications for simulating the output voltage versus frequency of an inverting operational amplifier.

Table 13 Input and Output Requirements for an AC Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
V	LIN, LOG, LIST, SYNC, CON, AC	V, I	1. Exactly one frequency sweep required.
I	LIN, LOG, LIST, SYNC, CON, AC	V, I	2. At least one AC source required.
F	LIN, LOG, LIST, CON	V, I	3. SYNC is not a valid sweep type when using ADS simulators.



```
Active Setup: /opamp1/inv_amp/B_P_macro

        Input: Vin                    Input: freq
           Mode: V                      Mode: F
         + Node: 1               Sweep Type: LOG
         - Node: GROUND         Sweep Order: 1
           Unit:                      Start:  1.000K
  Compliance:  0.000                   Stop:  10.00MEG
  Sweep Type: AC                # of Points: 3
   Magnitude:  1.000             Dec or Oct: D
       Phase:  0.000              Total Pts: 13

                  Output: Vout
                     Mode: V
                   + Node: 6
                   - Node: GROUND
                     Unit: CH2
                     Type: S
```

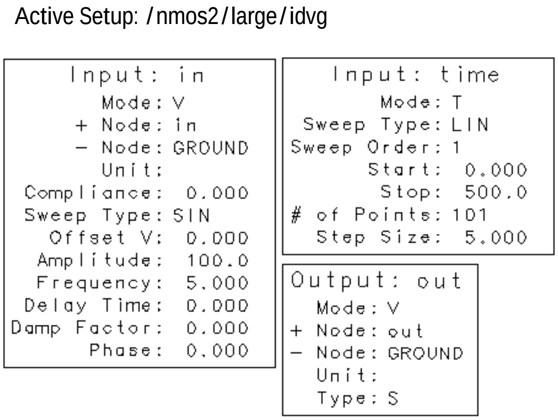
Figure 14 Example Input and Output, AC Simulation

Transient Simulation

The following table describes the input and output specifications required for a valid setup for Transient simulation and the following figure shows an example of input and output specifications for voltage versus time characteristics of a differential pair.

Table 14 Input and Output Requirements for a Transient Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
T	LIN, LIST, CON	V, I	1. Exactly one time sweep required
V	LIN, LOG, LIST, SYNC, CON, EXP, PULSE, PWL, SFFM, SIN	V, I	2. LIST sweep not supported with Saber
I	LIN, LOG, LIST, SYNC, CON, EXP, PULSE, PWL, SFFM, SIN	V, I	



Active Setup: /nmos2/large/idvg

```
      Input: in
         Mode: V
       + Node: in
       - Node: GROUND
         Unit:
  Compliance:  0.000
  Sweep Type: SIN
    Offset V:  0.000
   Amplitude:  100.0
   Frequency:  5.000
  Delay Time:  0.000
 Damp Factor:  0.000
       Phase:  0.000
```

```
      Input: time
         Mode: T
  Sweep Type: LIN
 Sweep Order: 1
       Start:  0.000
        Stop:  500.0
 # of Points: 101
   Step Size:  5.000
```

```
 Output: out
   Mode: V
 + Node: out
 - Node: GROUND
   Unit:
   Type: S
```

Figure 15 Example Input and Output, Transient Simulation

Noise Simulation

The following tables describe the input and output specifications required for a valid setup for Noise simulation depending on the simulator being used.

Table 15 Input and Output Requirements for a Noise Simulation using SPICE Simulators

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
V	LIN, LOG, LIST, SYNC, CON, AC	N	1. Exactly one noise output required.
I	LIN, LOG, LIST, SYNC, CON, AC	N	2. Exactly one frequency sweep required.
F	LIN, LOG, CON	N	3. At least one AC source required.

Table 16 Input and Output Requirements for a Noise Simulation using ADS Simulators

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
V	LIN, LOG, LIST, SYNC, CON, AC	N, V, I	1. There can be multiple outputs.
I	LIN, LOG, LIST, SYNC, CON, AC	N, V, I	2. Exactly one frequency sweep required.
F	LIN, LOG, SEG, CON	N, V, I	3. Both V and I Outputs are DC outputs.

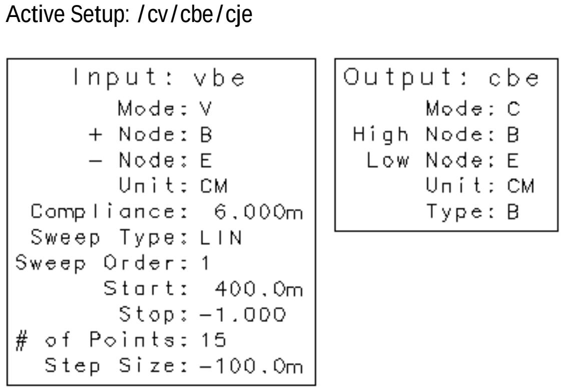
CV Simulation

The following table describes the input and output specifications required for a valid setup for CV simulation and the following figure shows an example of input and output specifications for a BJT base-emitter pn-junction capacitance *versus* voltage setup.

NOTE The frequency at which the CV simulation is performed can be specified using the System Variable CV_FREQ in Hz. If this variable is not specified, the simulation will be performed at 1-MG (Hz).

Table 17 Input and Output Requirements for a CV Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
V	LIN, LOG, LIST, SYNC, CON	a single C, a single G, a single R, a single X, C + G, C + R	1. Any single output type of C, G, R, or X; or exactly one C and one G; or exactly one C and one R.
I	LIN, LOG, LIST, SYNC, CON	a single C, a single G, a single R, a single X, C + G, C + R	2. A single C or a C + G is the capacitance using Cp-Gp mode, while a C + R will simulate Cs-Rs data



Active Setup: /cv/cbe/cje

```
        Input: vbe
             Mode: V
           + Node: B
           - Node: E
             Unit: CM
       Compliance:  6.000m
       Sweep Type: LIN
     Sweep Order: 1
            Start:  400.0m
             Stop: -1.000
     # of Points: 15
        Step Size: -100.0m
```

```
Output: cbe
          Mode: C
 High Node: B
   Low Node: E
          Unit: CM
          Type: B
```

Figure 16 Example Input and Output, CV Simulation

2-Port Simulation

The following table describes the input and output specifications required for a valid setup for 2-port simulation and the following figure shows an example of input and output specifications for an H21-parameter *versus Vbe* setup.

Table 18 Input and Output Requirements for a 2-Port Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
F	LIN, LOG, LIST, CON	S, H, Y, Z, K, A, F	1. Exactly one frequency sweep required.
V	LIN, LOG, LIST, SYNC, CON	S, H, Y, Z, K, A, F	2. Exactly one 2-port output (S,H,Y,Z,K or A) required.
I	LIN, LOG, LIST, SYNC, CON	S, H, Y, Z, K, A, F	3. Only ADS supports F output.*
			4. Exactly one 2-port output required for F output.
			5. F output can be multiple outputs.

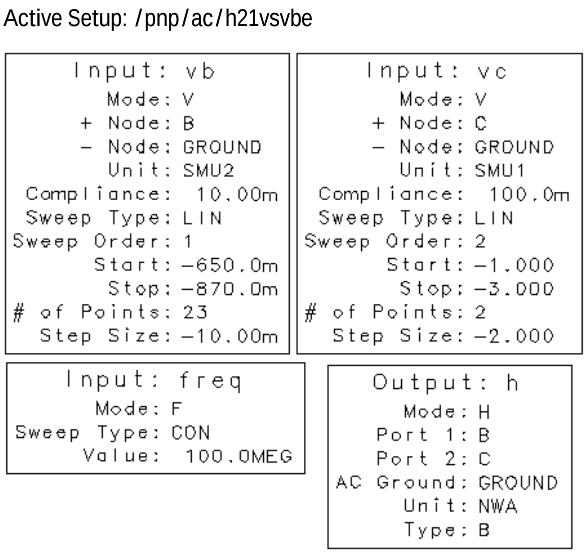
* 2-Port noise simulation supports ADS only. If the SIMULATOR is not equal to hpeesofsim, the following error message appears.
Error: in "Output xxxx"
High frequency noise output is not supported with current simulator.

Table 19 High Frequency Noise Output and Its Data Type Description

F Output	Name (Output Editor)	Symbol shown on Setup Page	Description	Port Input Requirement
Mode	High Frequency Noise	F	High frequency noise mode type	
Data Type	Noise Figure	NF	Noise figure data	Yes*, †
	Gamma Opt	GAMMAOPT	Optimum source reflection coefficients	No
	Equivalent R Noise	RN	Equivalent noise resistance data	No
	Min Noise Figure	NFMIN	Minimum noise figure data	No
	Equivalent Noise Temperature	TE	Equivalent Noise Temperature data	Yes *, †

* The port field of the NF/TE noise parameter can not be blank.
If the port field *set NF/TE type* is blank, the following error message appears.
Error: in "Output xxxx"
Blank output node name for NF specified.

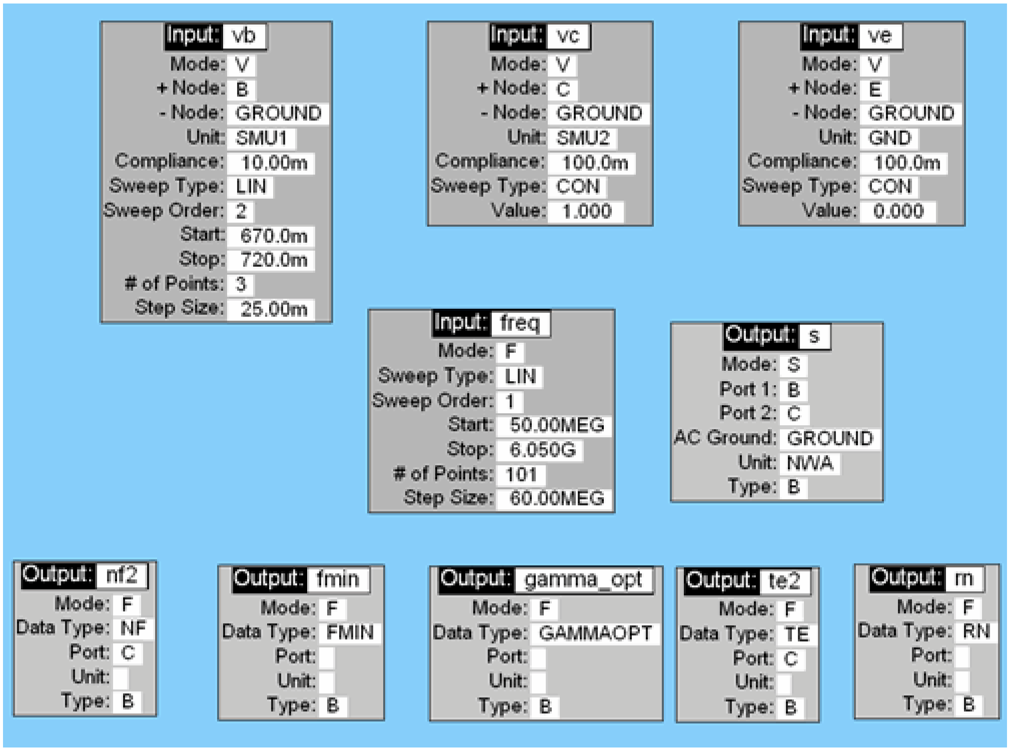
† The port name of the NF/TE parameter must be consistent with the port name of the 2-port output.
The port name of the NF/TE output noise must be equal to one of the port names of the 2-port output; otherwise, the following error message appears.
Error: in "Output xxxx"
Port "xx" is not consistent with the 2-port specification node: "xx" or "xx"



```
Active Setup: /pnp/ac/h21vsvbe

Input: vb
         Mode: V
       + Node: B
       - Node: GROUND
         Unit: SMU2
   Compliance:  10.00m
   Sweep Type: LIN
  Sweep Order: 1
        Start: -650.0m
         Stop: -870.0m
  # of Points: 23
    Step Size: -10.00m

Input: vc
         Mode: V
       + Node: C
       - Node: GROUND
         Unit: SMU1
   Compliance:  100.0m
   Sweep Type: LIN
  Sweep Order: 2
        Start: -1.000
         Stop: -3.000
  # of Points: 2
    Step Size: -2.000

Input: freq
         Mode: F
   Sweep Type: CON
        Value:  100.0MEG

Output: h
         Mode: H
       Port 1: B
       Port 2: C
    AC Ground: GROUND
         Unit: NWA
         Type: B
```

Figure 17 Example Input and Output, 2-Port Simulation

Active Setup: /bjt_ncehf/sp/sparm_noise_bias



Input: vb
Mode: V
+ Node: B
- Node: GROUND
Unit: SMU1
Compliance: 10.00m
Sweep Type: LIN
Sweep Order: 2
Start: 670.0m
Stop: 720.0m
# of Points: 3
Step Size: 25.00m

Input: vc
Mode: V
+ Node: C
- Node: GROUND
Unit: SMU2
Compliance: 100.0m
Sweep Type: CON
Value: 1.000

Input: ve
Mode: V
+ Node: E
- Node: GROUND
Unit: GND
Compliance: 100.0m
Sweep Type: CON
Value: 0.000

Input: freq
Mode: F
Sweep Type: LIN
Sweep Order: 1
Start: 50.00MEG
Stop: 6.050G
# of Points: 101
Step Size: 60.00MEG

Output: s
Mode: S
Port 1: B
Port 2: C
AC Ground: GROUND
Unit: NWA
Type: B

Output: nf2
Mode: F
Data Type: NF
Port: C
Unit:
Type: B

Output: fmin
Mode: F
Data Type: FMIN
Port:
Unit:
Type: B

Output: gamma_opt
Mode: F
Data Type: GAMMAOPT
Port:
Unit:
Type: B

Output: te2
Mode: F
Data Type: TE
Port: C
Unit:
Type: B

Output: rn
Mode: F
Data Type: RN
Port:
Unit:
Type: B

Figure 18 Example Input and Output, 2-Port Noise Simulation

2-Port Circuits

An L-network of LCR is added to port 1 and port 2 to uncouple an AC signal from a DC bias to simulate a DUT using AC analysis. To see an actual input circuit deck, use the *Simulator Debugger.*

When the port 1 has an AC source, its signal goes through an R whose value is defined by TWOPORT_Z0 and its default is 50 [ohm]. Then this signal is given to a port 1 through a C whose value is defined by TWOPORT_C, and its default is 100 [F]. The port 1 is also connected to a DC bias source through an L whose value is defined by TWOPORT_L and its default is 100 [H]. The port 2 has a similar L-network whose AC source is replaced by a short to ground.

NOTE

Because the default C and L values are so large compared to actual DUT values, sometimes it is necessary to specify smaller values to reduce numeric errors in simulation. For example, 1mF for C and 1mH for L are more realistic values.

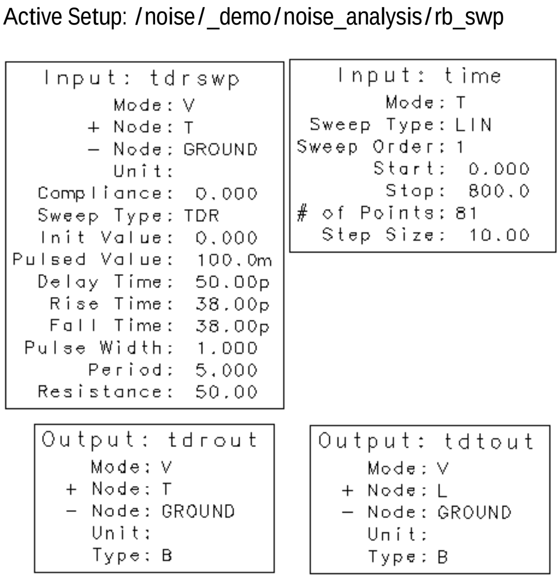
There are two circuits in a single deck to represent two cases where the port 1 has a source and the port 2 has a source. These circuits are generated and added to a DUT for simulators.

TDR Simulation

The following table describes the input and output specifications required for a valid setup for TDR (Time Domain Reflect) simulation. The following figure shows an example of input and output specifications for simulating the reflected and transmitted signal of a simple TDR circuit.

Table 20 Input and Output Requirements for a TDR Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
T	LIN, LIST, CON	V	1. Exactly one time sweep required.
V	LIN, LOG,L IST, SYNC, CON, TDR	V	2. Only voltage outputs allowed.
I	LIN, LOG, LIST, SYNC, CON	V	3. Exactly one voltage sweep of type TDR required.



Active Setup: /noise/_demo/noise_analysis/rb_swp

```
      Input: tdrswp                Input: time
           Mode: V                     Mode: T
         + Node: T                Sweep Type: LIN
         - Node: GROUND          Sweep Order: 1
           Unit:                       Start:  0.000
     Compliance:  0.000                 Stop:  800.0
     Sweep Type: TDR             # of Points: 81
     Init Value:  0.000            Step Size:  10.00
   Pulsed Value:  100.0m
     Delay Time:  50.00p
      Rise Time:  38.00p
      Fall Time:  38.00p
    Pulse Width:  1.000
         Period:  5.000
     Resistance:  50.00

   Output: tdrout             Output: tdtout
        Mode: V                    Mode: V
      + Node: T                  + Node: L
      - Node: GROUND             - Node: GROUND
        Unit:                      Unit:
        Type: B                    Type: B
```

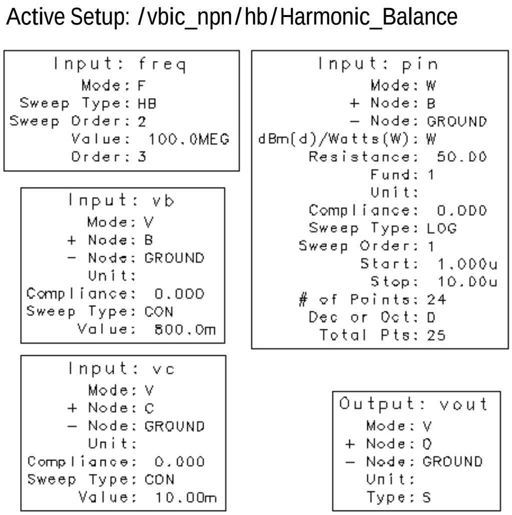
Figure 19 Example of Input and Output Specifications for a TDR Simulation

Harmonic Balance Simulation

The following table describes the input and output specifications required for a valid setup for Harmonic Balance simulation and the following figure shows an example of input and output specifications for a BJT power-in versus power-out setup.

Table 21 Input and Output Requirements for a Harmonic Balance Simulation

INPUT MODE	VALID SWEEPS	VALID OUTPUTS	COMMENTS
V	LIN, LOG, LIST, SYNC, CON, AC	V, I	1. Exactly one frequency sweep with Sweep Type = HB required.
I	LIN, LOG, LIST, SYNC, CON, AC	V, I	2. At least one AC source required: V or I (AC), or W.
F	HB	V, I	3. The Test Circuit includes elements such as DCFEED or DCBLOCK.
W (Power)	LIN, LOG, LIST, CON	V, I	4, Units for the Power (W) source can be set to dBm(d) or Watts (W).



Active Setup: /vbic_npn/hb/Harmonic_Balance

```
      Input: freq
         Mode: F
  Sweep Type: HB
 Sweep Order: 2
        Value:  100.0MEG
        Order: 3
```

```
      Input: vb
         Mode: V
     + Node: B
     - Node: GROUND
         Unit:
 Compliance:  0.000
 Sweep Type: CON
       Value:  800.0m
```

```
      Input: vc
         Mode: V
     + Node: C
     - Node: GROUND
         Unit:
 Compliance:  0.000
 Sweep Type: CON
       Value:  10.00m
```

```
           Input: pin
                 Mode: W
               + Node: B
               - Node: GROUND
dBm(d)/Watts(W): W
         Resistance:  50.00
                 Fund: 1
                 Unit:
         Compliance:  0.000
         Sweep Type: LOG
        Sweep Order: 1
               Start:  1.000u
                Stop:  10.00u
        # of Points: 24
         Dec or Oct: D
          Total Pts: 25
```

```
Output: vout
   Mode: V
 + Node: O
 - Node: GROUND
   Unit:
   Type: S
```

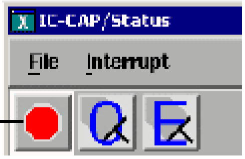
Figure 20 Example of Input and Output Specifications for a Harmonic Balance Simulation

Aborting a Simulation

You can abort a simulation at any time from the IC-CAP Status window.

To abort a simulation:

In the IC-CAP/Status window, click Interrupt IC-CAP Activity.



The simulation stops and the following message appears in the Status window:

```
HALTED: Simulation interrupted by user
```

Linking a Simulator to IC-CAP

The interface for linking a simulator to IC-CAP depends on the type of simulator being used.

A non-piped simulation receives the input deck information from a file, performs the simulation, and sends the binary output data and resulting text output to other files. The simulator process is restarted for every simulation. The non-piped simulations are identical, regardless of simulator type.

The definition of a piped simulation differs for SPICE simulators, Saber simulators, the MNS simulator, and the ADS simulator. For descriptions of these differences, refer to one of these simulator-specific sections in the *Reference* manual:

- “Piped and Non-Piped SPICE Simulations"
- “Piped and Non-Piped SPECTRE Simulations"
- “Piped and Non-Piped Saber Simulations"
- “Piped MNS Simulations" and “Non-Piped MNS Simulations"
- “Piped ADS Simulations" and “Non-Piped ADS Simulations"

The following simulator links have been tested to work for IC-CAP 2002 PC:

- Remote to spectre on UNIX
- Local to HSPICE on PC
- Local to hpeesofsim on PC
- Local to SPICE2 on PC
- Local to SPICE3 on PC
- Local to HPSPICE on PC

The following simulator links may work for IC-CAP 2002 PC, but were not thoroughly tested:

- Remote to MNS on UNIX
- Remote to HSPICE on UNIX
- Remote to HSPICE on another PC
- Remote to hpeesofsim in CANNOT_PIPE mode on another PC
- Remote to hpeesofsim in CANNOT_PIPE mode on UNIX

Adding a Simulator

The IC-CAP Open Simulator Interface allows the addition of any simulator to IC-CAP. The following figure shows a data flow diagram of this interface.

In this design, the IC-CAP system has no knowledge of your simulator. The circuit description is entered in the format corresponding to the template name in the usersimulators field (2nd argument).

Template Name	Syntax
spice2 spice3 hpspice hspice eldo spectre443	spice
mns (obsolete)	mns
spectre spmodeads	spectre
saber	saber
hpeesofsim	hpeesofsim (ADS)

These simulators are referred to as the *template* simulators when writing interface code. For example, when you specify a circuit description with syntax matching the syntax of *spice2,* then *spice2* is referred to as the *template* simulator.

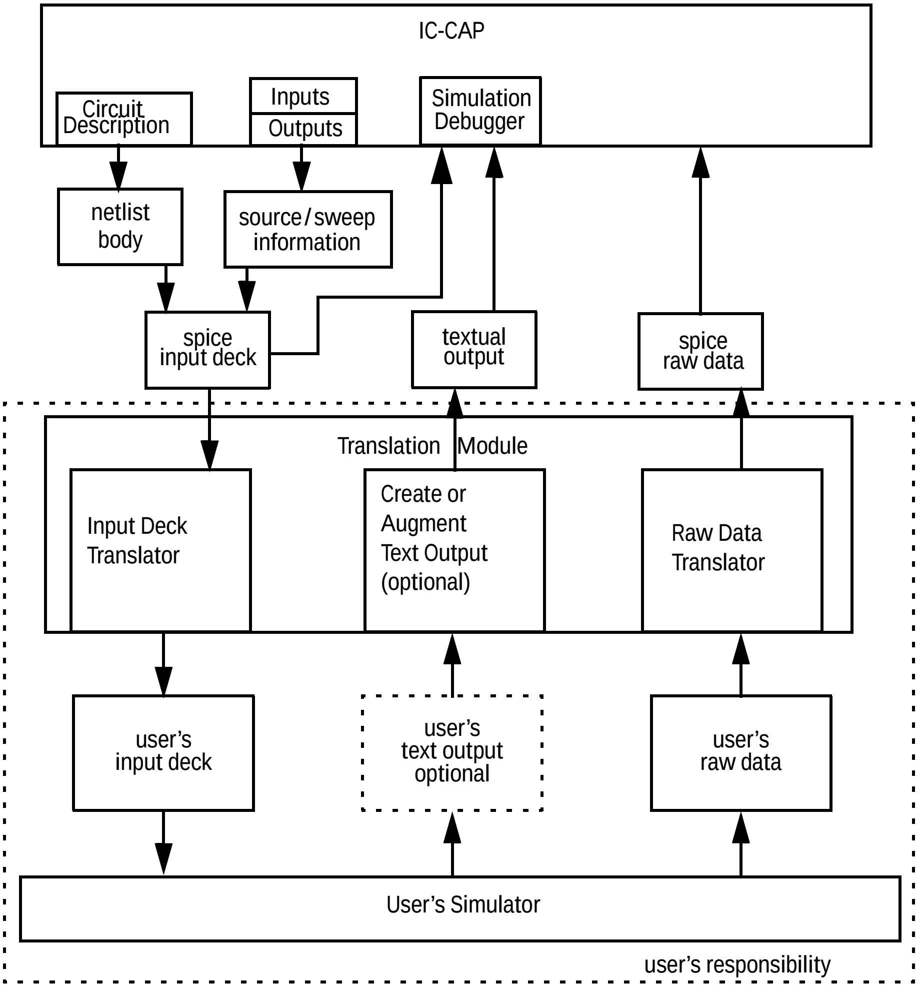


Figure 21 IC-CAP Open Simulator Interface Data Flow Diagram

IC-CAP generates the input deck as if the selected simulator were the template simulator. When you enter your simulator in the *Select Simulator* dialog box or the SIMULATOR variable, the input deck is sent through a module of code that you have written. This module of code is referred to as the *Translation Module* in the figure. The Translation Module consists of two translators. The first translates the IC-CAP generated input deck to your simulator's input deck. The deck is then sent to your simulator for analysis. When the simulation is completed, the second translation accepts your simulator's raw data and converts this to the raw data format of the template simulator. This data is returned to IC-CAP for evaluation.

Starting with IC-CAP 2002 PC and IC-CAP 2004 UNIX, you can specify whether the simulator returns the raw data in big-endian or little-endian format. If you do not specify a format, IC-CAP assumes that the data is returned in the order native to the platform, which is big endian for UNIX and little endian for the PC. To specify big endian, append .be or the alias .hpux or .sparc to the template name. To specify little endian, append .le or the alias .pc to the template name. For additional information, see the *README.byteorder* file in the source directory *$ICCAP_ROOT/src* directory.

It is possible to have a circuit description in the native syntax of the simulator that you will use. This is done by using the Open Circuit Parsing Interface that is available in IC-CAP. Though the netlist body is provided in your selected simulator's native syntax, the source and sweep information that IC-CAP adds to the netlist body is still in the template syntax. To use this interface, you must specify the name of the executable responsible for generating the netlist body.

When the Simulation Debugger is running, the file displayed as the input file is the template simulator's input deck generated by IC-CAP. The Simulation Debugger's output file can be any text file generated in the Translation Module section of code. This allows many options with respect to what information can be included in this text file that may be helpful for debugging purposes.

The Translation Module section of code consists of the two translators, but may also include any other desired functionality. For example, you may read the text file back from your simulator and send this file back to IC-CAP to be displayed in the Simulation Debugger's output file. The Translation Module can also be written to generate debug statements in a text file to show the progress of the input and output translations. This text file can then be sent to IC-CAP and displayed in the Simulation Debugger's output file.

As part of the interface code, you are supplied with a file called *usersimulators* in the directory *$ICCAP_ROOT\iccap\lib*. This file must contain the user-specified information for each simulator added to the IC-CAP system. Five fields of information must be specified when adding a user simulator to IC-CAP. The fields of information must be separated by a space. The fields may or may not be surrounded by quotation marks. A blank, such as *host_name*, is indicated by a pair of quotation marks (""). An optional sixth field is available to use the Open Circuit Parsing Interface.

The general format is as follows:

```
simulator_name template_name path_name host_name pipe_capability [parser_path][special_path]
```

where:

simulator_name is the name of the user simulator being added to the list. This is the name you will specify when selecting the simulator in the *Select Simulator* dialog box or the SIMULATOR variable. You may assign any name to this field EXCEPT for any of the reserved IC-CAP template names. The reserved template names are shown in Table 10.

template_name is the name of the *template* simulator. The user-written translation modules map the input file format of the user's simulator to the input file format of the *template* simulator. Likewise, the output file format of the *template* simulator is translated into the output file format of the user's simulator. To specify that the simulator returns the raw data in big endian format, append *.be* or the alias *.hpux* or *.sparc*. To specify that the simulator returns the raw data in little endian format, append *.le* or the alias *.pc*.

path_name is the complete path name of the user's simulator executable file or translation module. Use back slashes when naming the path to a simulator on a PC and forward slashes when naming the path to a simulator on UNIX.

host_name is the host machine name on which the simulator can be used. The purpose of this information is for remote simulations where only a particular computer is able to access a simulator. If this field is blank, indicated by a pair of quotation marks (""), the simulation is executed on the machine currently running IC-CAP. The format of *host_name* is <*host*> [<*tmp_dir*>], where <*host*> is any host name permitted by *rsh* and *remsh*. Examples include remotebox, remotebox.my.com, 192.168.4.4, and icuser@remotebox. The last form enables users with sufficient permission to simulate to the machine *remotebox* as if the user *icuser* was performing the simulation. This is useful when simulating to a UNIX machine from a PC when the login names for the PC don't match the login names for the UNIX machine. <*temp_dir*> is optional and it enables you to specify a location for IC-CAP's temporary files. The default location is */var/tmp* on the remote machine. For example, if a PC is running services and it meets the requirements in "Network Security" on page 264, */var/tmp* (UNIX notation) may not work for the PC. You can override this by specifying something like *c:\temp\*. For more information, see "Remote Simulation" on page 263.

pipe_capability is either CAN_PIPE or CANNOT_PIPE. It specifies whether or not the simulator has the ability to perform piped simulations. When CANNOT_PIPE is specified in this field, a non-piped simulation is done even when the IC-CAP simulation debugger is off.

parser_path is an optional entry that specifies the name of the executable responsible for generating the netlist body and providing IC-CAP with the necessary parameter/node information.

special_path is a simulator-specific field and may have different meanings for each simulator. Currently it is only required by the saber interface and will be ignored for any other simulator template. The field can be completely omitted from all templates but saber. For saber, it should provide the path to the *aimsh* executable in your saber installation. Note, to specify this field without declaring a parser_path, you must specify two quotations "" for the *parser_path* field.

Using the Open Circuit Parsing Interface

To use the interface and generate a circuit description in the native syntax of the simulator, you must specify the executable in the *usersimulators* file. This optional field in *usersimulators* is the path to the circuit parser. This enables the simulator link to use your simulator's native syntax in IC-CAP's circuit description shown in the Circuit folder. Without this interface, your circuit must be represented as *spice*, *hpeesofsim*, or *sabre*. If you specify an executable, it is responsible for the following actions:

- Creates a *parsed_file* that IC-CAP will use to merge parameters at each simulation.
- Identifies all pertinent parameters for in the DUT Parameters and Model Parameters folders.
- Identifies the number and names of all nodes that will be used in IC-CAP.

Your parser will be invoked with two arguments, the source file name and the output file name. The parser is responsible for generating the output file which is the same as the input file with substitutions for node names, parameters, and model names. In addition, the circuit type must be declared.

Determining the circuit type differs depending on the template being used. For any of the *spice* templates, the netlist should consist of one instantiation of one model or a subcircuit.

```
<instance line>
<model card>
```

or

```
.subckt
.
.
.ends
```

The first should be declared a circuittype of the first character of the instance line. The second should be declared a circuit of type *X*. The instance line should be omitted from the output file as IC-CAP will generate this line with the proper node numbers for the type of simulation being performed.

For *hpeesofsim* simulations, the circuit is similar, either a subcircuit, or a model and an instance. circuittype for a subcircuit is still *X*, but for the instance netlist, the type is always *D*.

For *saber* simulations, circuittype is again a *D* for device netlists and an X for subcircuit netlists, but for this template, you must set device type as well which is the actual name of the device type.

The output file should place the token $ where the name of the model should appear in the netlist. It should place the token <name>$ where the value for parameter named <name> should appear.

```
Example Device circuit:
D1  1 = A  2 = C  DIODE
.MODEL DIODE D
+ IS = 1E-14
+ N = 1.0
+ BV = 1000
+ IBV = 1m
+ RS = 0
+ CJO = 0
+ VJ = 1.0
+ M = 0.5
+ FC = 0.5
+ TT = 0
+ EG = 1.110
+ XTI = 3.0

Output File:
.MODEL $modname$ D
+ IS = $pvalIS$
+ N = $pvalN$
+ BV = $pvalBV$
+ IBV = $pvalIBV$
+ RS = $pvalRS$
+ CJO = $pvalCJO$
+ VJ = $pvalVJ$
+ M = $pvalM$
+ FC = $pvalFC$
+ TT = $pvalTT$
+ EG = $pvalEG$
+ XTI = $pvalXTI$

Example subcircuit circuit
.OPTION gmin=1e-30
.SUBCKT LED 1=A 2=C
RS  1 11 1m
DLO 11 2 DLO
DHI 11 2 DHI
.MODEL DLO D
+ IS  = 1E-29
+ N   = 1
.MODEL DHI D
+ IS  = 1E-34
+ N   = 1
+ CJO = 100p
+ M   = .4
+ VJ  = 2
+ FC  = .5
.ENDS

Output File:
.SUBCKT $modname$ 1 2
RS 1 11 $pvalRS$
DLO 11 2
+ DLO
DHI 11 2
+ DHI
.MODEL DLO D
+ IS = $pvalDLO.IS$
+ N = $pvalDLO.N$
.MODEL DHI D
+ IS = $pvalDLO.IS$
+ N = $pvalDLO.N$
+ CJO = $pvalDLO.CJO$
+ M = $pvalDLO.M$
+ VJ = $pvalDLO.VJ$
+ FC = $pvalDLO.FC$
.ENDS
```

The parser must print the commands to standard output that tell IC-CAP about the circuit it has parsed. Each line must meet one of the following formats:

PARAM <name> <value>

MODELPARAM <name> <value>

DEVPARAM <name> <value>

DEVMODELPARAM <name> <value>

> DEVPARAMs and DEVMODELPARAMs are parameters that are to appear at the DUT level. The difference between DEVPARAMs and DEVMODELPARAMs are that DEVMODELPARAMs appear in model cards.
>
> MODELPARAMs are PARAMs that appear in model cards. PARAMs and MODELPARAMs appear in the model parameters page.
>
> <name> is the name of the model. <value> is its default value.
>
> For certain saber parameters that can be altered, you may prepend SPECIAL to any of the PARAM keywords.

NODE <nodename>

Each NODE line declares a node to be recognized in IC-CAP setups. The order of the NODE lines must match the order the nodes are to appear when IC-CAP instantiates the instance card.

CIRCUITTYPE <x>

Here <x> is a single character. See above discussion of circuit types for proper values.

DEVICETYPE <x>

Here <x> is the name of the device for a device type circuit. See discussion about circuit types. This line is only required for saber.

UNRESOLVED <x>

Here <x> is the name of a model which was referenced in the netlist, but had no associated model card. In this case IC-CAP will try to find a model in its loaded list of models to insert.

ERROR: <x>

Here <x> is any arbitrary error message. The space after the colon is required. The entire line, including ERROR: will be reported in an error dialog.

DECKCOMPLETE

This should be the last line issued indicating that the parse was successful and that the output file has been generated.

Translation Module Example

An example translation module, *$ICCAP_ROOT/src/mysim.c,* is provided with IC-CAP. The executable version of this program is *$ICCAP_ROOT/bin/mysim.* The following line is an example for adding a simulator called *mysim* to the IC-CAP simulator list:

```
mysim spice2 $ICCAP_ROOT/bin/mysim "" CAN_PIPE
```

where:

The simulator *mysim* uses *spice2* as the template simulator.

mysim is a user-written module that does the following:

- Translates a *spice2* input format deck to a *mysim* input format deck.
- Makes the call to the user's simulator. In this example, the executable simulator is *spice2*.
- Translates the user's binary output format to *spice2* binary output format.
- Optionally sends information to the output text file.

mysim is located in the *$ICCAP_ROOT/bin* directory.

The current host computer can perform a *mysim* simulation. The quotation marks ("") mean that no remote host is specified and therefore the simulation can be done on the current host machine.

The simulator *mysim* is capable of piped simulations.

After creating a translation module, you must compile it, using the system command:

```
cc -o mysim mysim.o -lm
```

NOTE

Whenever *$ICCAP_ROOT/iccap/lib/usersimulators* is modified, always restart IC-CAP to read the new simulator configuration. This file may be a symbolic link on SunOS so that each host served by a single file server can have a different simulator configuration.

Reserved Simulator Names

The following simulator names are reserved by IC-CAP and you cannot assign the same name to a different simulator:

spice2, spice3, hpspice

The following simulator names are defined in the *usersimulators* file but you can change their name and assign the same name to a different simulator.

hspice, saber, mns, eldo, precise, spectre, spectre_spi, pspice, hpeesofsim

Simulator Argument Syntax

The command syntax for each simulator differs depending on whether a piped or non-piped simulation is being invoked. For details, refer to the *Reference* manual.

Remote Simulation

You can perform a simulation on a computer other than the computer on which you are working by using the remote simulation feature. You might do this for one of the following reasons:

- Running the simulation on a faster machine
- Running the simulation on a computer authorized to run a particular simulator

To execute a remote simulation, the remote machine must meet the following requirements:

- It runs Linux, SunOS, or a similar operating system that supports execution of Berkeley's remote shell *(remsh* or *rsh)* and remote file copy *(rcp)* commands.
- Both local and remote machines *know* each other. This means both machines are connected by a network and the IP address database is updated to talk to the other machine. This database is usually found in */etc/hosts*.
- It allows remote shell and copy program execution from your local host without entering a password (relaxed network security).
- It allows for removal of files using */bin/rm*.
- The remote machine is expected to have a directory named */var/tmp* for writing temporary files unless an alternate directory is specified in the *usersimulators* file for that simulator's *host_name* field. See *host_name* (page 255) for details.

The procedure for setting up the appropriate network security for your simulator depends on your remote host's operating system.

Remote Simulation Algorithm

The name of the remote host is specified in the *usersimulators* file found in *$ICCAP_ROOT/iccap/lib*. Remote simulation is supported in both CAN_PIPE and CANNOT_PIPE mode for *most* simulators. However, some simulators may only work in CANNOT_PIPE mode. See "Linking a Simulator to IC-CAP" on page 250.

The machine name for a simulator in this file determines where each simulator runs.

- When a remote machine is not specified, the simulation takes place locally on your host computer.
- If the remote machine is specified, check to see if this name is the same as the current host name. When the remote machine is identical to the current host, the simulation is executed on the current host directly.
- If a remote machine is specified and this name is not the same as the current host name, a remote simulation is performed by a remote shell command which is */usr/ucb/rsh* on SunOS and Linux, and *cygwin rsh* on the PC.
- In non-piped simulation, necessary files are copied to the remote machine using a remote file copy command which is */usr/ucb/rcp* on SunOS and Linux, and *cygwin rcp* on the PC.

Network Security

When the remote commands listed above are executed, the current user ID is used to establish access to the remote machine. Therefore, it is necessary to have the same user ID on both local and remote machines. Also, the following files may be modified to allow remote program execution from a particular host.

- */usr/adm/inetd.sec*
- */etc/hosts.equiv*
- *.rhosts*

When the security is set up, make sure the following command returns the current date without any errors (substitute your remote machine name where <*remote_machine*> appears in the example).

On SunOS or Linux, type:

```
rsh  <remote_machine>  date
```

For the PC, execute the following line of PEL in an IC-CAP macro. The results are displayed in the status window.

```
print system$("rsh <remote machine> date")
```

If your remote simulator requires licensing variables, you should write a small shell-script on the remote host machine that sets all required environment variables before invoking the simulator. This shell script is then specified in your *usersimulators* file instead of the actual simulator executable.

Example:

```
#!/bin/sh
LM_LICENSE_FILE=my_license_file.lic
export LM_LICENSE_FILE
PATH=/path/to/my/simulator/bin:$PATH
export PATH
/path/to/my/simulator/bin/xxxxx $*
```

The $* on the last line is required to pass along the IC-CAP command line parameters.

Executing a Remote Simulation

After the *usersimulators* file is set up correctly and the network security is adjusted, the steps for performing a remote simulation are identical to those required to perform a non-remote simulation. For more information, refer to "Performing a Simulation" on page 231.

NOTE

Make sure that there are no commands in the *.cshrc* file on the remote host that may generate output. Also, do not perform terminal related operations in your *.cshrc* file such as termset or stty. Since there is no physical terminal with remote shell commands, commands expecting one in your *.cshrc* file lead to errors.

Remote Simulation Examples

The following example specifications for running the template simulators remotely can be added to the *usersimulators* file in *$ICCAP_ROOT/iccap/lib*.

NOTE

If you prefer to preserve the defaults as shipped, you can add the specifications to your local, or Home, directory by copying and editing the usersimulators file. If you set the specifications in your home directory, you must change the pointers in your configuration file. Copy the file *$ICCAP_ROOT/config/iccap.cfg* to *$HOME/hpeesof/config/iccap.cfg*. See the *Installation and Customization Guide* for additional information.

The examples contain sample user-assigned simulator names, remote host machine names, and directory path (on the remote machine) information that should be replaced by the actual names in your system. The purpose of these examples is to show the names of the simulator executable files.

You must specify a full path name for each simulator because the *PATH* variable on the remote machine may not have the necessary search path to find your simulator.

NOTE

User-assigned simulator names can be whatever you choose except for reserved names. See "Reserved Simulator Names" on page 261. To use a user-assigned simulator, make sure the simulator name is listed in the first column of the *usersimulators* file, then set your simulator in IC-CAP to the same name.

To run spectre on the remote Solaris machine called *cadencebox*, enter the line:

```
remspectre_SS spectre443 /cadence/5.0.0/tools/bin/spectre "cadencebox" CANNOT_PIPE
```

where:

spectre443 is the template for spectre version 4.4.3 and greater, which causes IC-CAP to parse its circuit page expecting spice syntax.

From a PC, enter the line:

```
remspectre_SS spectre443.be /cadence/5.0.0/tools/bin/spectre "cadencebox" CANNOT_PIPE
```

Since the Solaris machine's byte order is big endian, the *.be* extension must be appended to the template name.

To run native spectre on the remote Solaris machine called *cadencebox*, enter the line:

```
remspectre spectre /cadence/5.0.0/tools/bin/spectre "cadencebox" CANNOT_PIPE
```

where:

spectre is the template for native spectre, which causes IC-CAP to parse its circuit page expecting native spectre syntax. See SPECTRE Interfaces in the *Reference* manual, chapter 4, "SPECTRE Simulator."

From a PC, enter the line:

```
remspectre spectre.be /cadence/5.0.0/tools/bin/spectre "cadencebox" CANNOT_PIPE
```

Since the Solaris machine's byte order is big endian, the *.be* extension must be appended to the template name.

To run UCB SPICE 2G.6 on the remote machine called *spice2mach*, enter the line:

```
spice2rem spice2 /usr/iccap/bin/ucbspice2g6 "spice2mach" CAN_PIPE
```

where:

spice2rem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls.

To run UCB SPICE 3E2 on the remote machine called *spice3mach,* enter the line:

```
spice3rem spice3 /usr/iccap/bin/spice3e2 "spice3mach" CAN_PIPE
```

where:

spice3rem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls

To run HPSPICE on the remote machine called *hpspicemach,* enter the line:

```
hpspicerem hpspice /usr/iccap/bin/shpspice "hpspicemach" CAN_PIPE
```

where:

hpspicerem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls

When performing remote simulations using the HPSPICE simulator, *both* of the executable files called *shpspice* and *spice2.4n1* must be present on the remote machine.

To run HSPICE on the remote machine called *hspicemach,* enter the line:

```
hspicerem hspice /usr/bin/hspice "hspicemach" CANNOT_PIPE
```

where:

hpspicerem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls

Depending on the version of HSPICE you have installed, the execution script called *hspice* may exist in a different directory path from */usr/bin/hspice.* In this case, create a symbolic link from */usr/bin/hspice* to the actual *hspice* script that will be called. For example, if your hspice script exists under */usr/meta/h9007/bin/hspice,* then execute the following command to create the required symbolic link:

```
ln -s /usr/meta/h9007/bin/hspice /usr/bin/hspice
```

Note that you must be in the *root* directory when executing the above command.

To run ELDO on the remote machine called *eldomach,* enter the line:

```
eldorem eldo<anacad_root>/eldo/<version>/com/eldo"eldomach" CANNOT_PIPE
```

where:

<*anacad_root*> and <*version*> are replaced with the home directory of the ANACAD software and the current version number of ELDO, respectively; *eldorem* represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls.

To run Saber on the remote machine called *sabermach* enter the line:

```
saberrem saber /usr/saber/bin/saber "sabermach" CAN_PIPE
```

where:

saberrem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls

To run MNS on the remote machine called *mnsmach* enter the line:

```
mnsrem mns /usr/hp85150/lib/mns "mnsmach" CAN_PIPE
```

where:

mnsrem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls.

mns is the simulator name.

The third field (an arbitrary example in this statement) is the path to the simulator’s location installed on the remote machine.

To run HPEESOFSIM on the remote machine called *hpsimmach* enter the line:

```
hpsimrem hpeesofsim <simulator path> “hpsimmach” CAN_PIPE
```

where:

hpsimrem represents the name of the simulator and *pipe* creates an interprocess channel that responds to read/write calls.

hpeesofsim is the simulator name.

The third field is the path to the simulator’s location installed on the remote machine.

To launch an external simulation, you must copy the file *$ICCAP_ROOT/bin/hpeesofsim_start* to some location on the remote machine. You must then modify the file (as explained within the file) to set HPEESOF_DIR and HPEESOFSIMFRONT_DIR for the remote machine. Then you must insure *hpeesofsim_front* is on the remote machine. If the remote machine is the same OS as the local machine, then you can copy *$ICCAP_ROOT/bin/hpeesofsim_front* from the local machine. If the remote machine is a different architecture, *hpeesofsim_front* for all architectures can be found on your distribution CD under the subdirectory *simlinks*. Finally, modify the file *usersimulators* to refer to the remote host and the path to *hpeesofsim_start* on the remote host.

7
Optimizing

This chapter provides descriptions and use models for IC-CAP optimizers.

Optimization in IC-CAP

Optimization is the process of creating an optimum set of model parameter values. This optimum parameter set is created by adjusting the initial model parameter values in an iterative process. The process continues until simulated output data matches the actual measured output data within specified tolerances.

NOTE

Past versions of IC-CAP contained four optimizers: Levenberg-Marquardt, Random, Hybrid, and Sensitivity Analysis. IC-CAP 2004 has added nine new optimizers. For more information, see Table 22 on page 274.

Given a set of measured data, the optimizer iteratively solves for a set of model parameters which produce simulated data that optimally approximates the measured data.

The algorithm works as follows:

1 Using the Circuit description and model parameter values extracted from measured data, the optimizer invokes the currently selected simulator to obtain a set of simulated data corresponding to the measured data used in the extraction process. This step is called *function evaluation*.

2 The optimizer compares the simulated and measured data and calculates the RMS error between them.

3 Based on the results, the optimizer calculates a new set of model parameter values and again compares the simulated and measured data.

4 This process continues with another function evaluation until the RMS error between the simulated and measured data either falls within a specified range, or no further improvement is possible.

Figure 22 illustrates the optimization process.

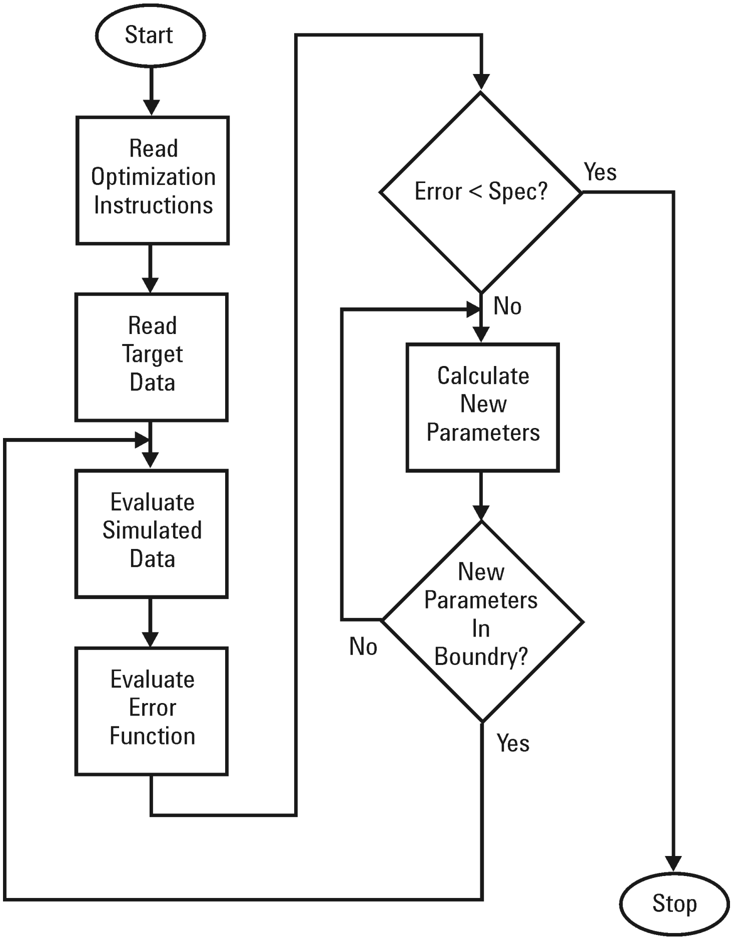


Figure 22 Optimization Flow Diagram

Optimization Algorithms

The choice of optimization algorithms depends upon:

- the goal of the optimization, and
- the nature of the model equations involved.

The optimization algorithm is selected with the Algorithm drop-down list on the Extract/Optimize table, described in "Selecting the Optimizer Algorithm" on page 300.

The following table includes a short description of each optimization algorithm.

Table 22 IC-CAP Optimization Algorithms

Algorithm	Description
Levenberg-Marquardt	Non-linear search method with least-squares error function.
Random	Random search method with stochastic gradient error function.
Hybrid (Random/LM)	Combination of Random and Levenberg-Marquardt algorithms and error functions.
Sensitivity Analysis	Single-point or infinitesimal sensitivity analysis of a design variable. Prints partial derivatives with respect to each parameter.
Random (Gucker)[1]	Random search method with least-squares error function.
Gradient[1]	Gradient search method with least-squares error function.
Random Minimax[1]	Random search method with minimax error function.
Gradient Minimax[1]	Gradient search method with minimax error function.
Quasi-Newton[1]	Quasi-Newton search method with least-squares error function.

Table 22 IC-CAP Optimization Algorithms

Algorithm	Description
Least P^{th} [1]	Quasi-Newton search method with least P^{th} error function.
Minimax[1]	Two-stage, Gauss-Newton / Quasi-Newton method with minimax error function.
Hybrid (Random / Quasi-Newton)[1]	Combines the Random and Quasi-Newton search methods.
Genetic[1]	Direct search method using evolving parameter sets.

[1] Uses full working precision of 15 digits during simulation and error calculation while optimizing. The IC-CAP Status window displays results based on the WORKING_PRECISION variable, which is 6 by default. At the end of the optimization, RMS and MAX error are calculated at the default precision. Therefore, results displayed at the end of the optimization may differ from those obtained during optimization steps.

Search Methods

Nine search methods are available within IC-CAP:

- "Levenberg-Marquardt Search" on page 276
- "Random Search" on page 279
- "Hybrid (Random/LM) Search" on page 280
- "Hybrid (Random/Quasi-Newton) Search" on page 280
- "Sensitivity Analysis" on page 281
- "Gradient Search" on page 281
- "Quasi-Newton Search" on page 282
- "Minimax (Gauss-Newton/Quasi-Newton) Search" on page 282
- "Genetic Search" on page 283

NOTE For initial optimization, use Random search. As the optimization proceeds, move to a Gradient search to increase process speed.

Levenberg-Marquardt Search

The Levenberg-Marquardt method uses a nonlinear, least-squares-fit (Levenberg-Marquardt) algorithm. This algorithm combines the steepest descent and Gauss-Newton methods.

The Levenberg-Marquardt method calculates the specified model parameters until the RMS error between measured and simulated data is minimized.

NOTE If the model is well-behaved and local minima are not a problem, use Levenberg-Marquardt Optimization.

For example, assume the model parameters to be optimized are represented by the vector X:

$X = (x_1, x_2, \ldots x_n)$

The cost function to be minimized, F(X), is the sum of the squares of the difference between measured and simulated values of the specified output variable (such as, *Id* in the case of MOSFET).

$$F(X) = \sum \left[I_{simu} - I_{meas} \right]^2$$

The optimizer finds the vector X of the model parameters that minimizes F(X).

The Levenberg-Marquardt method is a minimization algorithm, requiring the first derivative of this function. This method combines the Steepest Descent and Gauss-Newton methods.

Steepest Descent Method

The Steepest Descent method is an iterative algorithm for finding $X_k + 1 = X_k - a \cdot G_k$ where G is the gradient of F_k and the scalar $a \geq 0$ minimizes $F (X_k - a \cdot G_k)$. The algorithm searches along the negative gradient $-G_k$ from the point X_k to a minimum point $(X_k + 1)$ on the line.

Gauss-Newton Method

Newton's method is based on the assumption that near the minimum point X_{min}, the function F(x) can be approximated by a quadratic function. The truncated Taylor's vector series uses the following formula:

$$F(x + D_x) = F(x) + D_x \cdot G + \frac{D_x \cdot H \cdot D_x}{2}$$

where H is the Hessian or second gradient of F. The above expression is minimized at $D_x = \frac{-G}{K}$ therefore $X_k + 1 = X_k - \frac{G_k}{H_k}$.

The Gauss-Newton method is a modified version of Newton which approximates the Hessian by $2 \cdot J_t \cdot J$ where J is the Jacobian or the first derivative of F and J_t is its transposed matrix.

Levenberg-Marquardt Method

This method generates a sequence of approximation to the minimum point by the formula:

$X_k + 1 = X_k + P_k$

where

$P_k = -(J_t \cdot J + \text{Lambda} \cdot I)-^1 \cdot J_t \cdot F(x)$

I is the identity matrix.

Lambda is the Marquardt parameter, which is the sequence of non-negative real constants.

As Lambda becomes very large relative to the norm $J_t \cdot J$, P_k tends toward the direction of steepest descent. When Lambda is very small, the Gauss-Newton solution is obtained.

For a bad initial estimate, Lambda is chosen large. The algorithm then behaves like the steepest descent, giving large improvements in the minimization of the objective function F(x) to guarantee convergence.

Within the iteration, Lambda is increased until a reduction in F(x) is achieved. Between iterations, Lambda is reduced successively so that as the minimum is reached, P_k tends to move closer to the Gauss-Newton direction. This ensures that the overall algorithm has second order convergence near the minimum.

Jacobian Calculation

The Jacobian is numerically calculated using the forward difference approximation:

$J = (F(X + dX) - F(X))/dX$

This requires n+1 function evaluations for each iteration point X_k. To reduce the number of function evaluations, Broyden's rank one correction method approximates the Jacobian:

$$J_{k+1} = J_k + \frac{(Y_{k+1} - Y_k - J_k \cdot P_k) \cdot P_k^t}{||P_k||^2}$$

IC-CAP calculates the actual Jacobian after every n iterations (n = the number of parameters) or when the Lambda is increased.

Random Search

The Random, Random (Gucker), and Random Minimax optimizers arrive at new parameter values by using a random-number generator to choose random values within a range until the specified RMS error value is obtained.

NOTE

This random process can be slower compared to the optimizers using complex gradient search methods.

Random optimization is useful for avoiding local minima, before or after a Levenberg-Marquardt optimization. It can be used before Levenberg-Marquardt optimization to reduce the error to an acceptable starting point for Levenberg-Marquardt, or after to make sure that the global minimum has been reached.

Optimization with random search method is a trial and error process. Starting from an initial set of parameter values for which the error function is known, a new set of values is obtained by perturbing each of the initial values, and the error function is re-evaluated.

For optimization with random search method, a *trial* consists of two error function evaluations. A trial is completed by reversing the algebraic sign of each parameter value perturbation and re-evaluating the error function. These two values, corresponding to positive and negative perturbations, are compared to the value at the initial point.

If either value is less than the initial value, then the set of parameter values for which the error function has its least value becomes the initial point for the next trial. If neither value is less than the initial value, then the initial point remains the same for the next trial.

Hybrid (Random / LM) Search

The Hybrid (Random/LM) search is a combination of the Random and Levenberg-Marquardt methods. At each random jump, the Levenberg-Marquardt optimizer is called and the RMS error is evaluated.

Hybrid (Random/LM) optimization takes advantage of the speed of Levenberg-Marquardt while using random jumps to avoid local minima.

NOTE

If local minima present a problem, consider using Hybrid (Random / LM) optimization.

Hybrid (Random / Quasi-Newton) Search

The Hybrid (Random/Quasi-Newton) optimizer combines the Random and quasi-Newton method. It offers a compromise between the ability to find a minimum quickly, using the fewest possible parameter analyses (the strength of quasi-Newton optimization) and the possibility to find the global cost minimum in the presence of many local minima (the strength of Random optimization).

When Hybrid (Random/Quasi-Newton) optimization is selected, the process begins with a quasi-Newton search, quickly finding the nearest local minimum. When the gradient approaches zero, it is near a minimum and can do little to decrease the cost function. The system then uses a random search technique to generate a new initial guess. Another quasi-Newton search is then performed, starting at this initial guess.

This process continues until the optimizer can no longer improve the performance of the model, or it reaches the specified maximum number of iterations.

NOTE

Hybrid (Random / Quasi-Newton) optimization is an excellent choice if time is available for relatively long analyses.

In summary, the Hybrid (Random/Quasi-Newton) optimizer:

- uses gradient information during the optimization, and
- is useful for finding a local, as well as the global, minimum.

Sensitivity Analysis

In sensitivity analysis mode, the partial derivatives of the data with respect to each parameter are printed. In addition, the partial derivatives at each data point are stored as transform data and can be viewed graphically. To gain insight into the sensitivities involved in a particular optimization, sensitivity analysis should be selected.

Gradient Search

The optimizers using gradient search method (Gradient and Gradient Minimax optimizers in Table 22 on page 274) find the gradient of the network's error function. These optimizers usually progress more quickly to a point where the error function is minimized, though it is possible for them to terminate in a local minimum.

The optimizers find the gradient of the error function (i.e., the direction to move a set of parameter values in order to reduce the error function). Once the direction is determined, the set of parameter values is moved in that direction until the error function is minimized and the gradient is re-evaluated. This cycle equals one gradient optimizer iteration.

NOTE

A single iteration usually includes many function evaluations. Therefore, a gradient search method iteration takes much longer than a random search method trial.

Quasi-Newton Search

The optimizers using Quasi-Newton search method (Quasi-Newton and Least Pth optimizers in Table 22 on page 274) use second-order derivatives of the error function and the gradient to find a descending direction.

The optimization routine using Quasi-Newton search method estimates the second-order derivatives using the Davidson-Fletcher-Powell (DFP) formula or its complement. Appropriately combined with the gradient, this information is used to find a direction and an inexact line-search is conducted.

The optimization terminates when the gradient vanishes, the change ratio in the variables is small (less than 1.0e-5), or the number of specified iterations is reached.

NOTE

An iteration in the optimizers using Quasi-Newton search methods consists of many function evaluations, and requires more time than a random search method trial.

Minimax (Gauss-Newton / Quasi-Newton) Search

The Minimax optimizer consists of two stages. In the first stage of the algorithm, the optimizer solves a minimax problem using a linear programming technique. In doing so, the status and potential of each individual error function component are analyzed. Its contribution to the minimax problem is mathematically assessed and taken into account during optimization.

In the second stage, the optimizer works with a Quasi-Newton method using approximate second-order derivatives. Such extra effort becomes necessary for an accurate and efficient solution to certain ill-conditioned problems (i.e., singular problems).

NOTE

The bounds imposed on the variables are formulated and treated directly as linear constraints without having to resort to variable transformation. Therefore, a source of nonlinearity is eliminated.

The Minimax optimizer terminates when responses become optimally equal-ripple, the relative change in the variables is less than 0.05 percent, or when the user-defined termination conditions are met (number of iterations, number of evaluations).

Genetic Search

Genetic algorithms (GA's) provide another direct search optimization method. The basis of the procedure is a set of trial parameter sets, sometimes called chromosomes, which are allowed to evolve towards a set that gives progressively better performance.

The key to the genetic optimization is the strategy of change, sometimes likened to survival of the fittest. Based on the idea that with each change in the parameter population, i.e., each generation of parameters, the performance given by the parameter population improves.

This performance evolution is achieved using a five-step process including (1) representation, (2) evaluation, (3) reproduction, (4) breeding and crossover, and (5) mutation.

1 *Representation:* Genetic algorithms require the input parameter set to be represented as a string of digits. It is straightforward to map each parameter onto the interval 0 to 1, for instance, and then have each of the *n* parameters occupy a position in the string of *n* bounded numbers. The algorithm then manipulates and optimizes this string of numbers as a whole. An individual string of parameters is called an *element* within the population of parameter strings.

2 *Evaluation:* Each generation of parameters begins with a performance evaluation of each string in the population. Usually this involves determining the performance G(P) for each representation of P in the population. Each element is

then graded as to how well it performed, often using an error function, known as the *fitness* function.

3 *Reproduction:* Some of the generation members are reproduced and added to the next generation population. The number of copies depends on the performance evaluation. The elements that perform well are copied several times, and those with poor performance are not copied at all. The copies, or *offspring*, comprise the next generation. Elements that are not copied are not represented in the next generation.

   The number of elements in each generation is constant.

   There are several suggested methods of evaluation and reproduction, including *ratioing*, where the number of copies is directly related to the element's performance, and *ranking* where the performances are ranked, with the top performers being copied more times than the lower ranked performers.

4 *Breeding and crossover:* The previous step, reproduction, produced a population of strings where each evaluated well. Breeding then combines parts of two strings to form two different and new strings. In this way good representations are mixed with poorer representations, with the result eventually being evaluated in the next generation of the algorithm.

   There are many methods for breeding; the most common is *crossover*. Crossover typically takes two elements, splits them at a random location in the string, and swaps the two parts to create two new strings (see the following figure). This provides a controlled statistical exploration of the performance space.

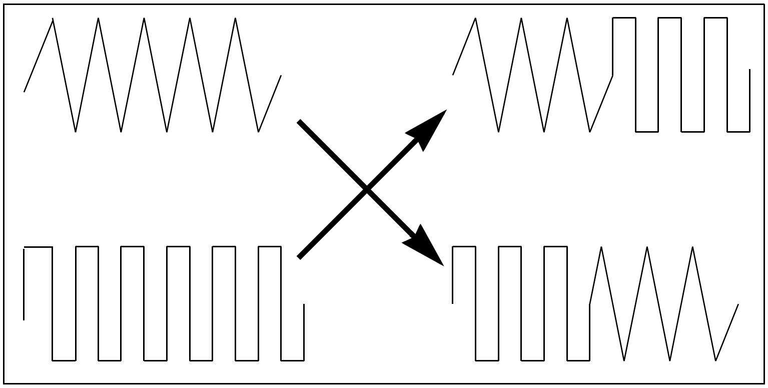


Figure 23 Breeding and Crossover in the Simple Genetic Algorithm

5 *Mutation:* The last step in creating a new generation of elements is the random changing of parameters in some of the surviving strings. This comprises a completely random search of the performance space, and can be viewed as the injection of information into the surviving population.

The following figure presents a completed flow diagram of the genetic algorithm. The application of these techniques requires many tuning parameters that are not available with yield optimization.

NOTE

Genetic optimization will prove useful for many complex optimization problems, including discrete value and tolerance optimization.

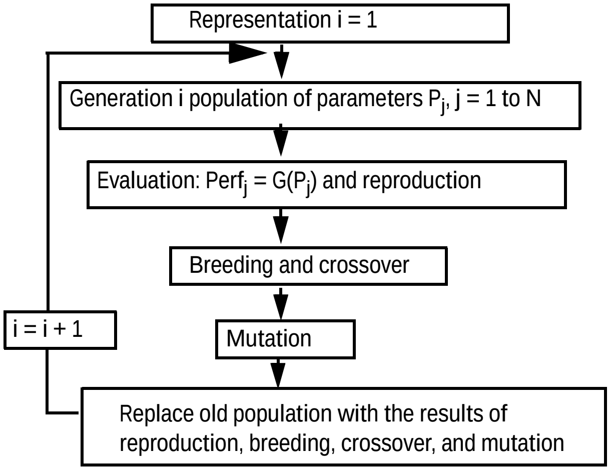

Representation i = 1
Generation i population of parameters P_j, j = 1 to N
Evaluation: $Perf_j = G(P_j)$ and reproduction
Breeding and crossover
i = i + 1
Mutation
Replace old population with the results of reproduction, breeding, crossover, and mutation

Figure 24 Random-Search Optimization Using a Genetic Algorithm

Relative / Absolute Error Formulation

With the exception of the Minimax, Gradient Minimax, and Random Minimax optimizers, all other algorithms use two error formulations: relative error and absolute error. (Minimax, Gradient Minimax, and Random Minimax optimizers use only absolute error.)

These error formulations can be expressed using the following equations:

Relative error:

$$e(i) = \left| \frac{I_{simu}(i) - I_{meas}(i)}{I_{meas}(i)} \right|$$

Absolute Error:

$$e(i) = \frac{|I_{simu}(i) - I_{meas}(i)|}{\sqrt{\frac{\sum_{n} (I_{meas}(i))^2}{n}}}$$

where

I_{simu} and I_{meas} represent the simulated and target data sets specified in the Inputs page of the optimizer, and

n the number of data points included in each data set.

Special handling is done for the Relative Error case to avoid situations where $I_{meas}(i)$ is zero or much smaller than the numerator. For the Levenberg-Marquardt optimizer, the formulation is

$$e(i) = \left| \frac{I_{simu}(i) - I_{meas}(i)}{max(|I_{meas}(i)|, |I_{simu}(i)|)} \right|$$

For the other optimizers, $I_{meas}(i)$ is used for the denominator for all points except when $I_{meas}(i)==0$ and only then is $I_{simu}(i)$ used in the denominator.

To select the desired error formulation, choose **Absolute** or **Relative** from the Error drop-down box on the Extract/Optimize page.

The absolute error is normalized using the RMS of the measured or target data set, which remains a constant quantity during optimization. This normalization assists the optimizer when different inputs are defined, otherwise inputs with a higher order of magnitude would prevail.

Error Function Formulation

The optimizers use different methods for error function formulation. The error function formulations are shown in the following table.

Table 23 Error Function Formulation

Error Function Formulation	Optimizers
Least-squares	Levenberg-Marquardt, Gradient, Quasi-Newton, Genetic, Hybrid (Random/LM), Hybrid (Random/Quasi-Newton)
Minimax	Random Minimax, Gradient Minimax
Least P^{th}	Least P^{th}

Least-Squares Error Function (L2)

The least-squares error function (also called mean square, MS) is calculated by evaluating the error for each specified goal at each data set point individually, then squaring the magnitudes of those errors. The squared magnitudes are then averaged over the number of points.

To help you understand the error function calculation in more generality for a measurement as a function of frequency, consider the following variable definitions.

p Total number of input columns (pair of Target and Simulated data sets) in the optimizer Inputs page. IC-CAP uses the index (j= 0, ..., p-1) to iterate between inputs.

n_j number of data set points of the j-th input. (j = 0, ..., p-1). IC-CAP uses the index i to identify a point within the target or simulated data set.

T_j the j-th Target data set defined in the optimizer Inputs page.

S_j the j-th Simulated data set defined in the optimizer Inputs page.

W_j the j-th input weighting factor as defined in the optimizer Inputs page for the j-th input.

$e_j(i)$ the absolute or relative error (see “Relative/Absolute Error Formulation” on page 287) calculated using the i-th point of the j-th Target and Simulated data sets in the optimizer Inputs page.

The total mean square error, MS is defines as:

$$E_{MS} = \sum_{j=0}^{p-1} W_j \cdot \frac{\left(\sum_{i=0}^{n_j-1} (e_j(i))^2\right)}{n_j}$$

The square root of the MS error is the well-known root mean square which is one of the optimizer termination conditions:

$$E_{RMS} = \sqrt{E_{MS}}$$

Minimax Error Function

The Minimax optimizer calculates the difference between the desired response and the actual response over the entire measurement parameter range of optimization. The optimizer then tries to minimize the point that constitutes the greatest difference between actual response and desired response.

Minimax means *minimizing the maximum (of a set of functions generally denoted as errors)*. The error function is defined as the maximum among all error contributions, expressed mathematically:

$$E_{MM} = max\{W_j \cdot e_j(i)\} \text{ among all of the i and j}$$

where

error term $e_j(i)$ is defined as in the previous section.

Note that the error is always positive (see “Relative/Absolute Error Formulation” on page 287.)

The minimax objective function always represents the worst case, where the specifications are most severely violated. The minimax optimizer will spend all its effort trying to minimize these. The goal of a minimax optimization is to meet specifications in an optimal, typically equal-ripple manner.

Least Pth Error Function

The Least Pth optimizer uses an error function formulation similar in makeup to the least squares method found in the Random, Gradient, and the Quasi-Newton optimizers. But, instead of squaring the magnitudes of the individual errors at each data set point, it raises them to the Pth power, where $p = 2$, 4, 8, or 16. The optimizer automatically increases p in the 2, 4, 8, or 16 sequence. This emphasizes the errors that have high values much more strongly than those that have small values. As p increases, the Least Pth error function approaches the minimax error function.

The Least Pth optimization routine is the exponential sum of the error function, where the exponent p is not necessarily equal to 2. It can be a positive number, usually an integer.

First of all, the maximum error is found as:

$$E_{MM} = max\{W_j \cdot e_j(i)\} \text{ among all of the i and j}$$

Since the error terms are always positive and the mimimax error $E_{MM} > 0$, we can define the Least Pth error function, E_{pth}, as follows:

$$E_{pth} = \left(\sum_j (e_j(i))^P\right)^{1/P}$$

The Least Pth formulation is used as an indirect method to achieve a minimax design.

Minimax error function can contain *edges* or discontinuities in their derivatives. These occur at points where the error contributions resulting from different goals intersect in the parameter space. The Least P^{th} error functions avoid this problem.

For a large value of *p*, the errors having the maximum value ($e_j(i) = E_{MM}$) are more strongly emphasized over the other errors, i.e., they are given higher priority in optimization. As *p* increases to infinity, the Least P^{th} formulation leads to a minimax error function. The problem is solved though a sequence of Least P^{th} optimizations with *p* being gradually increased. The sequential Least P^{th} optimization used in the program uses *p* = 2, 4, 8, 16. This strategy often provides a smooth path towards a minimax solution.

For more information on the least-squares error function, refer back to “Least-Squares Error Function (L2)” on page 289.

Termination Conditions

Termination conditions are the prerequisite factors for ending the function of optimization algorithms. When a termination condition value is met, the optimization algorithm ceases functioning.

Levenberg-Marquardt, Gradient, Quasi-Newton, Minimax, Gradient Minimax, Least Pth Algorithms

These algorithms cease functioning when one of the following termination conditions are met:

RMS error When the RMS deviation between all of the simulated and measured data points is less than the value specified in the optimizer Options table.

Maximum error When the absolute maximum error between the simulated and measured data is less than the value specified in the Options table.

Parameter tolerance When the percent change in each parameter value from one iteration to the next is less than the value specified in the Options table.

Function tolerance When the percent change in the RMS error from one iteration to the next is less than the value specified in the Options table.

Maximum function evaluations When the number of function evaluations exceeds the value specified in the Options table. This number specifies the number of times the setup is simulated. This, in turn, specifies the number of optimizer iterations.

Maximum iterations When the number of iterations meets the value specified in the Options table.

Random, Hybrid (Random / LM), Hybrid (Random / Quasi-Newton), and Genetic Algorithms

The random, hybrids, and genetic algorithms cease functioning when one of the following termination conditions are met:

RMS error When the RMS deviation between all of the simulated and measured data points is less than the value specified in the optimizer Options table.

Random iterations When the number of random optimizer iterations meets the number specified on the Options table.

Maximum function evaluations When the number of function evaluations exceeds the value specified in the Options table. This number specifies the number of times the setup is simulated. This, in turn, specifies the number of optimizer iterations.

RMS Error Definition

The error $e_j(i)$ can be absolute or relative (see "Relative/Absolute Error Formulation" on page 287). In both cases, the error is either zero or a positive value.

When multiple inputs are defined in the input table, the Levenberg-Marquardt algorithm uses one single array of points. When choosing absolute error formulation, the error terms e(i) are normalized using the RMS of all the target points:

$$RMS_{target} = \sqrt{\frac{\sum_{i=1}^{N} (I_{meas}(i))^2}{N}}$$

where

N is the total number of points from all the inputs.

The absolute error term used by the Levenberg -Marquardt optimizer becomes:

$$e(i) = \frac{|I_{simu} - I_{meas}|}{RMS_{target}}$$

and the RMS error becomes:

$$RMS = \sqrt{\frac{\sum_{i=1}^{n} (e(i))^2}{N}}$$

In optimizers other that the Levenberg-Marquardt, when selecting absolute error, the error term uses different normalizing factors, called RMStarget (j), for each input column on the Inputs page.

These normalizing factors are calculated using the target data sets related to each input column on the Inputs page. The RMS error is calculated as shown:

$$RMS_{\text{target}}(j) = \sqrt{\frac{\sum_{i=0}^{n_j-1}(I_{meas}(i))^2}{n_j}}$$

where

j is the target index (or column on the Inputs page)

n_j is the number of points of the j-th input.

For a definition of the *RMSerror* function, see Chapter 8, “IC-CAP Functions” in the *Reference* manual.

Transform Data

Different information is saved as transform data, depending on the optimization mode that is selected. This data is accessible to other IC-CAP objects, such as plots or PEL programs. The behaviors of the different modes are listed in the following table.

The RMS and maximum errors are normalized to 1 (1% = 0.01). If an invalid optimization setup is defined, no transform data is generated. If an error occurs during the optimization, the error data is set to 1.0.

Table 24 Optimizer Transform Data

Algorithm	Array Length	Contents
Levenberg-Marquardt	2	RMS error (element 0) and the maximum error (element 1)
Random	1	RMS error
Hybrid (Random/LM)	1	RMS error
Sensitivity Analysis*		The partial derivatives at each bias point are stored.
Random (Gucker)	2	RMS error (element 0) and the maximum error (element 1)
Gradient	2	RMS error (element 0) and the maximum error (element 1)
Random Minimax	2	RMS error (element 0) and the maximum error (element 1)
Gradient Minimax	2	RMS error (element 0) and the maximum error (element 1)
Quasi-Newton	2	RMS error (element 0) and the maximum error (element 1)
Least Pth	2	RMS error (element 0) and the maximum error (element 1)

Table 24 Optimizer Transform Data

Algorithm	Array Length	Contents
Minimax	2	RMS error (element 0) and the maximum error (element 1)
Hybrid (Random/Quasi-Newton)	2	RMS error (element 0) and the maximum error (element 1)
Genetic	2	RMS error (element 0) and the maximum error (element 1)

* This feature is described in detail in “Sensitivity Analysis Data” on page 332.

Performing an Optimization

The basic steps required to perform an optimization are:

1 Selecting an optimization transform.

2 Selecting an optimization algorithm.

3 Specifying the error as absolute or relative.

4 Specifying the inputs to the optimizer.

5 Specifying the parameters to be optimized.

6 Specifying the optimizer options.

7 Executing the optimization.

The following sections describe each of these steps, as well as undoing an executed optimization.

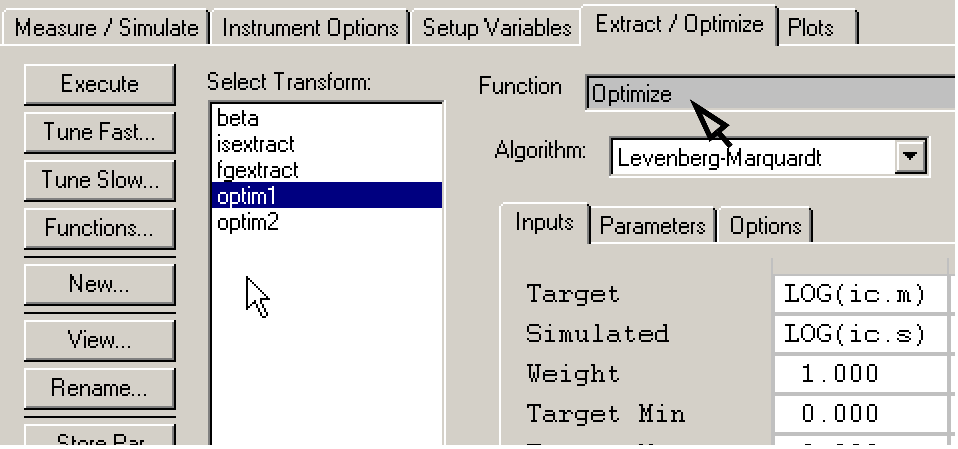
Select the Optimizer Transform

The optimizer in IC-CAP is provided in the form of a function, the *Optimize* transform.

To select the *Optimize* transform:

1 Select the setup.

2 In the Extract/Optimize folder Function data entry field, enter `Optimize`.



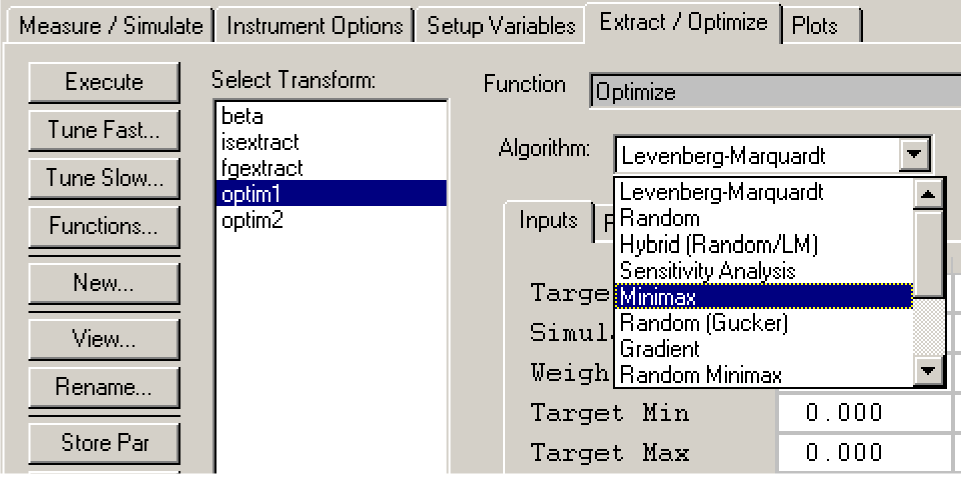
Function: Optimize

Algorithm: Levenberg-Marquardt

Target	LOG(ic.m)
Simulated	LOG(ic.s)
Weight	1.000
Target Min	0.000

Selecting the Optimizer Algorithm

After selecting the transform, choose an algorithm from the drop-down list.



Function: Optimize

Algorithm: Levenberg-Marquardt

Targe

Simul

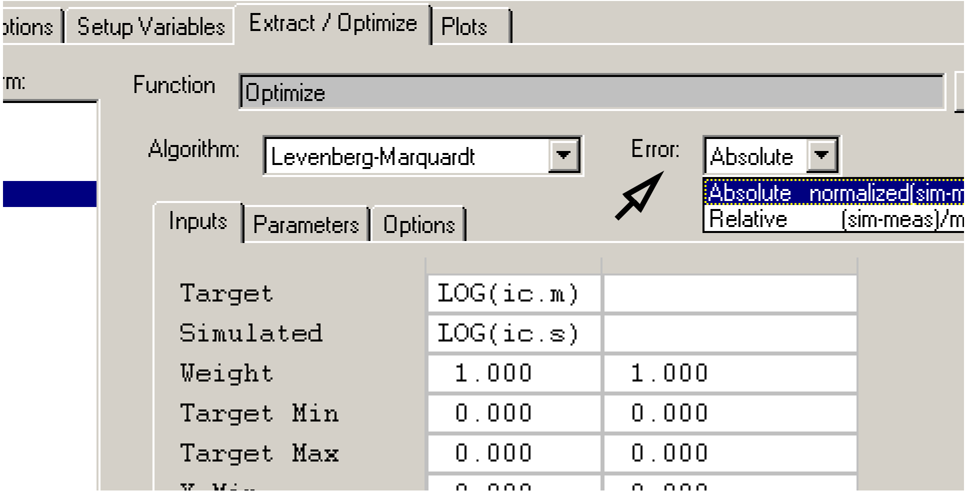
Weigh

Target Min 0.000

Target Max 0.000

Select the Error as Absolute or Relative

Choose an error type from the Error drop-down list.



Function: Optimize

Algorithm: Levenberg-Marquardt

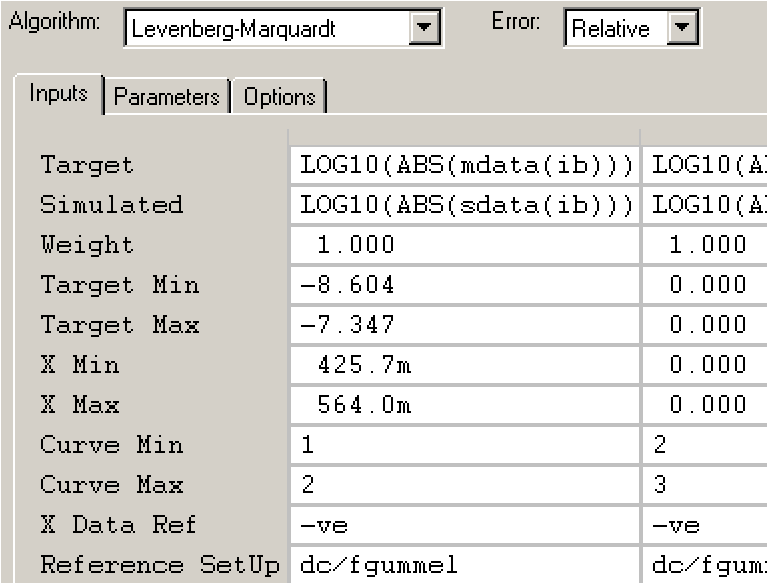
Error: Absolute

Target	LOG(ic.m)	
Simulated	LOG(ic.s)	
Weight	1.000	1.000
Target Min	0.000	0.000
Target Max	0.000	0.000

You must specify the inputs, parameters, and options for the selected transform. The following sections describe these tables. For further details, refer to Chapter 9, “Using Transforms and Functions.”

Defining Optimizer Inputs

You can define the optimizer inputs by completing the Inputs table.



Algorithm: Levenberg-Marquardt

Error: Relative

Target	LOG10(ABS(mdata(ib)))	LOG10(A
Simulated	LOG10(ABS(sdata(ib)))	LOG10(A
Weight	1.000	1.000
Target Min	-8.604	0.000
Target Max	-7.347	0.000
X Min	425.7m	0.000
X Max	564.0m	0.000
Curve Min	1	2
Curve Max	2	3
X Data Ref	-ve	-ve
Reference SetUp	dc/fgummel	dc/fgum

Figure 25 Optimizer Inputs

The fields available in the Inputs table (figure above) are:

Target Data set containing the measured data against which simulated data is matched.

Simulated Data set containing simulated data.

Neither target nor simulated data files must come from the same setup. The Inputs table may be used to specify either target or simulated data files from any location. The only requirement is that both data sets must have the same number of data points.

There is no limit on the number of data points.

You can also run multiple setup optimizations by entering more than one target simulated pair in the Inputs table. Enter as many target pairs as desired.

Weight Relative weighting factor to apply to each pair of Targeted and Simulated data. For details, see "Weighted Optimization" on page 326.

Target Min Lower limit for the target data values. Data values lower than this limit are ignored during optimization.

Target Max Upper limit for the target data values. Data values higher than this limit are ignored during optimization.

X Min Lower limit for the X-axis data values. Data with X-axis values lower than this limit are ignored during optimization.

X Max Upper limit for the X-axis data values. Data with X-axis values higher than this limit are ignored during optimization.

Curve Min Lower limit for the curve. For a family of curves, curves numbered lower than this limit are ignored during optimization. A single curve can be selected by entering the same curve number in both the Curve Min and Curve Max fields.

Curve Max Upper limit for the curve. For a family of curves, curves numbered higher than this limit are ignored during optimization.

X Data Ref Reference data set used for X boundary conditions X Min and X Max or when the variables X_LOW and X_HIGH are defined. When left blank (default), the reference data set is the first order sweep defined in the setup.

When no target or X-axis bounds are set, the bounds default to zero (0) and no bounding is performed. When minimum and maximum bound values are equal, no bounding is performed. These per-target bounds take precedence over the bounds specified in the optimizer Options table. That is, if bounds are entered in one or more Min or Max fields in an Inputs table, the bounds from the Options table are not applied to that row.

Reference SetUp The location of the setup that all inputs will reference. When this field is left blank (default), the inputs reference the current setup.

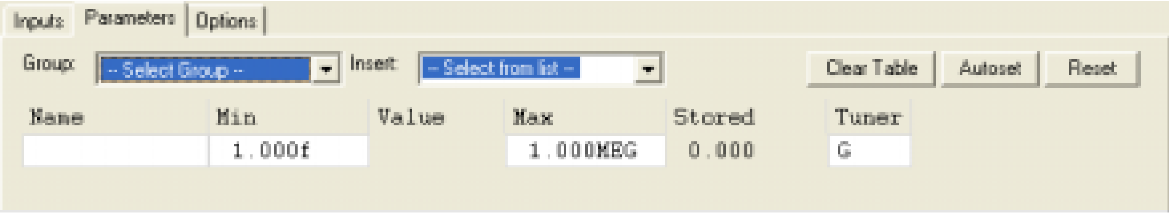
For example, if *ib.m* is set as the target, *ib.m* must be an output or a transform in the current setup to be resolved correctly. When it points to another setup, then inputs are referenced to that setup.

You can also point to a setup in another file (for example, *npn_01/dc/fgummel*).

When reading, the syntax *dc/fgummel* is changed to *../../dc/fgummel* to make it consistent with the relative path.

Defining Optimizer Parameters

You can define the parameters or variables to be optimized by completing the Parameters table. There is no limit on the number of parameters.



Name	Min	Value	Max	Stored	Tuner
	1.000f		1.000MEG	0.000	G

Figure 26 Optimizer Parameters

The settings available in the Parameters table (figure above) are:

Insert a Group Select a group in the first Drop-down, and then all the parameters or variables defined in the group can be selected in the second Drop-down.

Insert Param or Var No group selected in the first Drop-down, the second Drop-down list of all parameters defined in the Model Parameters table. Select a parameter to load from the Model Parameters table into the Optimizer Parameters table.

Clear Table Select to clear values from the optimizer Parameters table.

Auto Set Sets minimum and maximum values based on the value of the coefficient defined with the AUTOSET_COEFF variable on the System Variables page. The default coefficient value is 5.

if the Value > 0, then

Min = Value ÷ Coefficient

Max = Value × Coefficient

if the Value < 0, then

Min = Value × Coefficient

Max= Value ÷ Coefficient

if the Value =0

Min = –Coefficient

Max = +Coefficient

Using an AUTOSET_COEFF value of 5, the minimum would be set to 1/5 of the Value and the maximum would be set to 5 times the Value setting. An AUTOSET_COEFF value of 8 would result in the minimum set to 1/8 of the Value and the maximum would be set to 8 times the Value setting.

Name Name of a model parameter, DUT parameter or variable that is to be optimized. A parameter can be removed temporarily from an optimization by placing an exclamation point (!) before its name.

Min Minimum acceptable value of parameter.

Max Maximum acceptable value of parameter.

Stored The last stored value for the parameter.

The **Store Par** button stores the current parameter Value setting in the optimizer Parameters table to the Model Parameters table.

The **Recall Par** button recalls the last stored parameter Value setting in the optimizer Parameters table from the Model Parameters table.

Tuner Sets the Tuner to linear (L) or logarithmic (G) scale for the parameter. Default is G logarithmic.

The **Tune Fast** and **Tune Slow** buttons open a tuner window. The tuner enables you to adjust the parameter values. You can see the effects in a plot as you vary parameter values using the tuner.

Tune Fast recalculates constantly as a slider is moved. Tune Slow recalculates only when a slider is released.

While the tuner window is open, you can change the *Min* and *Max* values. It's best to use Tune Slow when the time per calculation is slow (about one second or more).

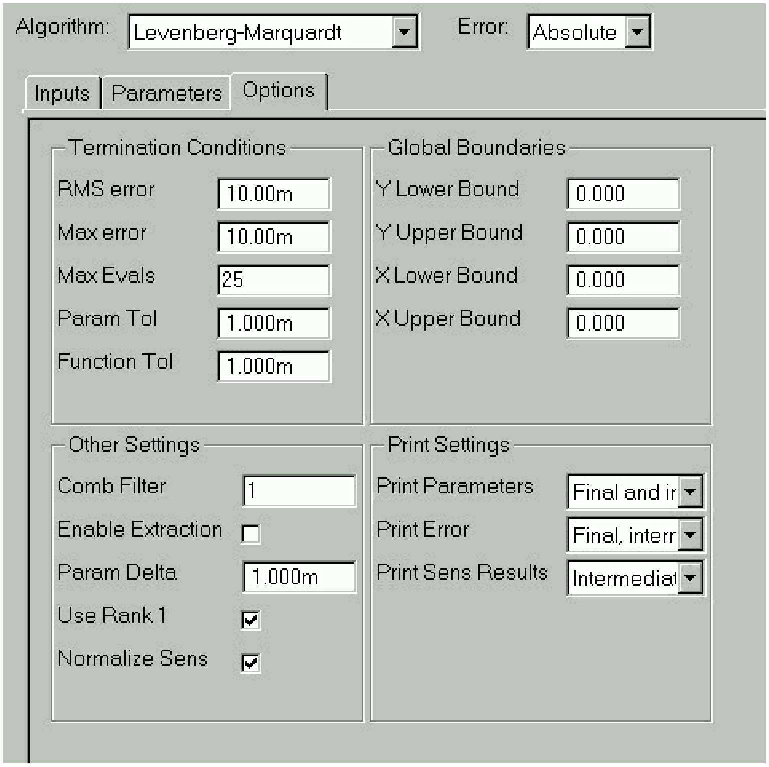
Store Param Locally stores the current parameter values to the Stored column on the Parameters table. When the file is saved, these values are stored as part of the optimizer settings.

Recall Param Resets the Parameters table values to the locally stored values.

Defining Optimizer Options

The Options table is divided into the following groups: Termination Conditions, Global Boundaries, Other Settings, and Print Settings. The specific variables contained within these four groups will change depending on the currently selected algorithm.

As an example, the following figure shows the optimizer Options table for the Levenberg-Marquardt algorithm.



Algorithm: Levenberg-Marquardt

Error: Absolute

Termination Conditions

RMS error 10.00m

Max error 10.00m

Max Evals 25

Param Tol 1.000m

Function Tol 1.000m

Global Boundaries

Y Lower Bound 0.000

Y Upper Bound 0.000

X Lower Bound 0.000

X Upper Bound 0.000

Other Settings

Comb Filter 1

Enable Extraction □

Param Delta 1.000m

Use Rank 1 ☑

Normalize Sens ☑

Print Settings

Print Parameters Final and ir

Print Error Final, interr

Print Sens Results Intermediat

Figure 27 Optimizer Options, Levenberg-Marquardt Algorithm

Termination Conditions

This group of settings determine the values at which the optimization process terminates. The following data fields may be active, depending on which algorithm is selected.

Function Tol Optimization stops when the percentage change in RMS error value, from one optimization to the next, is less than the specified value.

Max error Optimization stops when the maximum error between measured and simulated data is less than the specified value.

Max Evals Optimization stops when the number of function evaluations (which normally are simulations) exceeds the specified value.

Max Iters Optimization stops when the number of iterations reaches the specified value.

Parameter Tol Optimization stops when the percentage change in each parameter value, from one optimization to the next, is less than the specified value.

Rand Iters Optimization stops when the number of iterations reaches the specified value.

RMS error Optimization stops when the RMS deviation between all measured and simulated data points is less than the specified value. Percentages are normalized to 1. For example, 1% is specified as 0.01.

Global Conditions

This group of settings determine the rectangular limitations of the optimized data set.

Y Lower Bound Sets a lower limit for Y-axis data values. Data values lower than this limit are ignored during optimization. Bounding is performed according to the target data values. If a target data point falls within the specified range, the simulated data point is also used in the optimization, regardless of its value.

Y Upper Bound Sets an upper limit for Y-axis data values. Data values higher than this limit are ignored during optimization.

X Lower Bound Sets a lower limit for X-axis data values. Data values lower than this limit are ignored during optimization. X-axis bounding only applies to the main sweep of the Setup that contains the target data set. This cannot be changed by simply specifying a different X-axis on a plot.

X Upper Bound Sets an upper limit for X-axis data values. Data values higher than this limit are ignored during optimization.

You can set these bounds by using the Y_LOW, Y_HIGH, X_LOW, and X_HIGH variables. However, any functions, such as *log,* that are applied to the data sets in the Inputs table must be applied to the bounds also.

When no bounds are set, the option defaults to zero (0) and no bounding is performed. Also, when upper and lower bound values are equal, no bounding is performed. These bounds are only applied to targets that do not have any per-target bounds specified. That is, if bounds are entered in one or more of the target Min/Max, X Min/Max, or curve Min/Max fields, the bounds from the Options table are not applied to that Inputs table row.

Other Settings

This group of settings determine values specific to the chosen algorithm. The following data fields may be active, depending on which algorithm is selected.

Comb Filter Filters the target data to be used in the optimization by using every n^{th} point, where n is the value set in this field. For example, set the value to 2 and the optimizer uses every other data point. Set the value to 3 and the optimizer uses every third data point, and so on. The comb filter uses at least one data point from each data set.

Enable Extraction Select checkbox when optimization is to run when *Extract* is selected from the DUT or Setup Menu. The optimization is performed after any extractions that are defined in the Setup have been run.

Normalize Sens Controls the sensitivity analysis. Select check box for normalized sensitivities. Deselect checkbox for raw partial derivatives.

Param Delta Controls the parameter delta used in the optimizer when calculating the numerical derivatives. Typically this option is not altered, but altering it may be useful for certain time domain optimizations. The default is *0.001*.

Rand Std Dev Controls the standard deviation of the Gaussian distribution for Random optimization. The default is *0.3*.

Rand Reward Specifies the reward coefficient for Random optimization. The default = *0.5*.

Rand Penalty Specifies the penalty coefficient for Random optimization. The default = *0.5*.

Rand Seed Specifies the seed that controls the initial random jump. The default = *0*.

Use Rank 1 Select checkbox to allow the Broyden rank one Jacobian update.

Print Settings

This group of settings determine print values specific to the chosen algorithm. The following data fields may be active, depending on which algorithm is selected.

Print Parameters Sets option for printing parameter data.

Print Error Sets option for printing error data.

Print Sens Results Sets option for printing sensitivity information.

Executing an Optimization

To perform an optimization on the active setup,

1 Select the setup.

2 Click Optimize Setup.

To perform an optimization on all setups in the active DUT,

1 Select the DUT.

2 Click Optimize DUT.

Undoing an Optimization

To undo an optimization, click the **Undo Optim** button on the Extract/Optimize page.

Pressing this button resets the optimized parameter values to values prior to the optimization and runs a simulation.

An Example Optimization

For an example of the optimization process, you can optimize the UCB MOS Level 2 saturation region parameters VMAX and NEFF, using *id versus vd* measured data.

1. Load the model file **mosfet / nmos2**.
2. Select the DUT/Setup **large / idvg**.
3. In the Extract/Optimize folder, select the **optimize** transform.
4. In the optimizer Parameters table, click **Clear Table**.
5. From the Insert Pram or Var: drop-down box, select **VMAX**.
6. To specify the minimum and maximum acceptable values that the VMAX parameter can be set to, enter a minimum acceptable value of `100.0` in the Min field and a maximum acceptable value of `1.000MEG` in the Max field.
7. Similarly, select the parameter name `NEFF`, and adjust the with a minimum acceptable value to `0.000` and a maximum acceptable value to `10.00`.
8. In the Options tab Termination Conditions group box, enter `35` in the Max Evals data entry field.
9. To perform the optimization, click Execute on the Extract/Optimize page or ⇪ from the menu bar.

Optimization Output

Sample optimization output from the different types of optimization modes are shown in the following 4 tables.

Table 25 Levenberg-Marquardt Optimization

```
======= Levenberg-Marquardt Optimization Selected ========
Number of data points used: 40 of 40

========== Initial Values ==========

Parameter      Value

---------      -----

TRAN.BR        60.00

TRAN.IKR       10.00m

TRAN.ISC       2.000f

TRAN.NC        1.400

PAR.IS         20.00a

MAX error = 282.1 %     RMS error = 212.5 %

========== Jacobian Calculation ==========

Parameter      Value          Sensitivity

---------      -----          -----------

TRAN.BR        60.00          129.4m

TRAN.IKR       10.00m         58.54m

TRAN.ISC       2.000f         59.54m

TRAN.NC        1.400          558.4m

PAR.IS         20.00a         194.1m

========== Iteration 1 Results ==========

Parameter      Value          Initial

---------      -----          -------

TRAN.BR        6.158          60.00

TRAN.IKR       11.41m         10.00m
```

Table 25 Levenberg-Marquardt Optimization (continued)

```
TRAN.ISC       1.833f         2.000f
TRAN.NC        1.343          1.400
PAR.IS         23.00a         20.00a
MAX error = 25.28 %     RMS error = 10.40 %
========== Rank One Update ==========
Parameter      Value          Sensitivity
---------      -----          -----------
TRAN.BR        6.158          32.94m
TRAN.IKR       11.41m         71.99m
TRAN.ISC       1.833f         59.35m
TRAN.NC        1.343          589.6m
PAR.IS         23.00a         246.1m
Number of function evaluations :  11
Number of jacobian evaluations :   1
The Levenberg-Marquardt parameter is      :  113.4m
Exit Status: Parameter improvement less than specified
tolerance
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
========== Final Values ==========
Parameter      Value          Initial        Sensitivity
---------      -----          -------        -----------
TRAN.BR        6.158          60.00          279.9m
TRAN.IKR       11.41m         10.00m         49.47m
TRAN.ISC       1.833f         2.000f         64.72m
TRAN.NC        1.343          1.400          537.0m
PAR.IS 23.00a 20.00a 68.93m
Initial least square norm of the residuals:  13.44
```

Table 25 Levenberg-Marquardt Optimization (continued)

```
Final   least square norm of the residuals:  657.9m
Final MAX error = 25.28 %     Final RMS error = 10.40 %
```

Table 26 Random Optimization

```
========== Random Optimization Selected ==========
Number of data points used: 40 of 40
========== Initial Values ==========
Parameter      Value
---------      -----
TRAN.BR        60.00
TRAN.IKR       10.00m
TRAN.ISC       2.000f
TRAN.NC        1.400
PAR.IS         20.00a
Initial: RMS Error = 212.5 %
========== Random Iteration 1 ==========
========== Random Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        95.37          60.00          60.00
TRAN.IKR       43.61m         10.00m         10.00m
TRAN.ISC       3.581p         2.000f         2.000f
TRAN.NC        1.697          1.400          1.400
PAR.IS         14.26a         20.00a         20.00a
Best:  RMS Error = 212.5 %
Guess: RMS Error = 117.8 %
Guess is Accepted
```

Table 26 Random Optimization (continued)

```
========== Random Iteration 2 ==========
========== Random Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        69.30          95.37          60.00
TRAN.IKR       34.88m         43.61m         10.00m
TRAN.ISC       6.828p         3.581p         2.000f
TRAN.NC        1.823          1.697          1.400
PAR.IS         3.897a         14.26a         20.00a
Best:  RMS Error = 117.8 %
Guess: RMS Error = 175.3 %
Guess is Rejected
========== Opposite Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        60.05          95.37          60.00
TRAN.IKR       52.35m         43.61m         10.00m
TRAN.ISC       333.4f         3.581p         2.000f
TRAN.NC        1.572          1.697          1.400
PAR.IS         30.42a         14.26a         20.00a
Best:  RMS Error = 117.8 %
Guess: RMS Error = 140.1 %
Guess is Rejected
========== Random Iteration 3 ==========
========== Random Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        77.96          95.37          60.00
```

Table 26 Random Optimization (continued)

TRAN.IKR	29.36m	43.61m	10.00m
TRAN.ISC	4.618p	3.581p	2.000f
TRAN.NC	1.637	1.697	1.400
PAR.IS	40.01a	14.26a	20.00a
Best: RMS Error = 117.8 %			
Guess: RMS Error = 72.33 %			
Guess is Accepted			
========== Random Iteration 4 ==========			
========== Random Guess ==========			
Parameter	Guess	Best	Initial
---------	-----	----	-------
TRAN.BR	94.53	77.96	60.00
TRAN.IKR	34.09m	29.36m	10.00m
TRAN.ISC	4.766p	4.618p	2.000f
TRAN.NC	1.437	1.637	1.400
PAR.IS	25.31a	40.01a	20.00a
Best: RMS Error = 72.33 %			
Guess: RMS Error = 94.69 %			
Guess is Rejected			
========== Opposite Guess ==========			
Parameter	Guess	Best	Initial
---------	-----	----	-------
TRAN.BR	61.39	77.96	60.00
TRAN.IKR	24.64m	29.36m	10.00m
TRAN.ISC	4.469p	4.618p	2.000f
TRAN.NC	1.949	1.637	1.400
PAR.IS	54.71a	40.01a	20.00a
Best: RMS Error = 72.33 %			

Table 26 Random Optimization (continued)

```
Guess: RMS Error = 145.9 %
Guess is Rejected
========== Random Iteration 5 ==========
========== Random Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        81.24          77.96          60.00
TRAN.IKR       57.93m         29.36m         10.00m
TRAN.ISC       4.534p         4.618p         2.000f
TRAN.NC        1.398          1.637          1.400
PAR.IS         18.04a         40.01a         20.00a
Best:  RMS Error = 72.33 %
Guess: RMS Error = 96.93 %
Guess is Rejected
========== Opposite Guess ==========
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        37.16          77.96          60.00
TRAN.IKR       789.0u         29.36m         10.00m
TRAN.ISC       4.702p         4.618p         2.000f
TRAN.NC        1.876          1.637          1.400
PAR.IS         61.98a         40.01a         20.00a
Best:  RMS Error = 72.33 %
Guess: RMS Error = 47.70 %
Guess is Accepted
Random Opt Exit Status: RMS error is less than specified value
========== Final Values ==========
Parameter      Value          Initial
```

Table 26 Random Optimization (continued)

---------	-----	-------
TRAN.BR	37.16	60.00
TRAN.IKR	789.0u	10.00m
TRAN.ISC	4.702p	2.000f
TRAN.NC	1.876	1.400
PAR.IS	61.98a	20.00a
Final: RMS Error = 47.70 %		

Table 27 Hybrid (Random/LM) Optimization

======= Hybrid (Random/LM) Optimization Selected ========		
Number of data points used: 40 of 40		
========== Levenberg-Marquardt Optimization 1 ==========		
========== Initial Values ==========		
Parameter	Value	
---------	-----	
TRAN.BR	4.103	
TRAN.IKR	147.3u	
TRAN.ISC	15.69f	
TRAN.NC	1.857	
PAR.IS	30.00a	
MAX error = 112.1 % RMS error = 64.02 %		
========== Jacobian Calculation ==========		
Parameter	Value	Sensitivity
---------	-----	-----------
TRAN.BR	4.103	344.8m
TRAN.IKR	147.3u	42.41m
TRAN.ISC	15.69f	68.02m

Table 27 Hybrid (Random/LM) Optimization (continued)

```
TRAN.NC        1.857          413.5m
PAR.IS         30.00a         131.2m
========== Iteration 1 Results ==========
Parameter      Value          Initial
---------      -----          -------
TRAN.BR        6.585          4.103
TRAN.IKR       1.899m         147.3u
TRAN.ISC       23.57f         15.69f
TRAN.NC        2.000          1.857
PAR.IS         69.83a         30.00a
MAX error = 62.76 %      RMS error = 38.96 %
========== Jacobian Calculation ==========
Parameter      Value          Sensitivity
---------      -----          -----------
TRAN.BR        6.585          320.6m
TRAN.IKR       1.899m         122.6m
TRAN.ISC       23.57f         34.06m
TRAN.NC        2.000          261.9m
PAR.IS         69.83a         260.8m
========== Iteration 2 Results ==========
Parameter      Value          Initial
---------      -----          -------
TRAN.BR        8.074          4.103
TRAN.IKR       2.251m         147.3u
TRAN.ISC       100.0a         15.69f
TRAN.NC        1.593          1.857
PAR.IS         68.88a         30.00a
MAX error = 48.74 %      RMS error = 30.00 %
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
Number of function evaluations :  16
Number of jacobian evaluations :   2
The Levenberg-Marquardt parameter is      :  0.000
Exit Status: No minimum was reached after specified
function evaluations
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
========== Final Values ==========
Parameter      Value          Initial        Sensitivity
---------      -----          -------        -----------
TRAN.BR        8.074          4.103          409.0m
TRAN.IKR       2.251m         147.3u         150.5m
TRAN.ISC       100.0a         15.69f         2.273m
TRAN.NC        1.593          1.857          23.20m
PAR.IS 68.88a 30.00a 415.1m
Initial least square norm of the residuals:  4.049
Final   least square norm of the residuals:  1.897
Final MAX error = 48.74 %     Final RMS error = 30.00 %
========== Random Iteration 1 ==========
========== Random Guess ==========
========== Levenberg-Marquardt Optimization 2 ==========
.
.
.
Number of function evaluations :  10
Number of jacobian evaluations :   1
The Levenberg-Marquardt parameter is      :  30.63m
Exit Status: No minimum was reached after specified
function evaluations
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
========== Final Values ==========
Parameter      Value          Initial        Sensitivity
---------      -----          -------        -----------
TRAN.BR        43.45          43.45          2.783m
TRAN.IKR       35.86m         35.86m         7.565m
TRAN.ISC       3.579p         3.579p         75.31m
TRAN.NC        1.504          1.504          912.1m
PAR.IS 36.62a 36.62a 2.212m
Initial least square norm of the residuals:  5.369
Final   least square norm of the residuals:  5.369
Final MAX error = 115.4 %     Final RMS error = 84.89 %
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        43.45          8.074          4.103
TRAN.IKR       35.86m         2.251m         147.3u
TRAN.ISC       3.579p         100.0a         15.69f
TRAN.NC        1.504          1.593          1.857
PAR.IS         36.62a         68.88a         30.00a
Best:  RMS Error = 30.00 %
Guess: RMS Error = 84.89 %
Guess is Rejected
========== Opposite Guess ==========
========== Levenberg-Marquardt Optimization 3 ==========
.
.
.
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
Number of function evaluations :  16

Number of jacobian evaluations :   2

The Levenberg-Marquardt parameter is      :  7.043u

Exit Status: No minimum was reached after specified
function evaluations

========== Calculating Sensitivities ==========

========== Jacobian Calculation ==========

========== Final Values ==========

Parameter      Value          Initial        Sensitivity

---------      -----          -------        -----------

TRAN.BR        10.74          29.30          232.6m

TRAN.IKR       2.566m         31.56m         133.7m

TRAN.ISC       685.6a         3.579p         41.35m

TRAN.NC        1.330          1.969          352.0m

PAR.IS 60.53a 98.86a 240.4m

Initial least square norm of the residuals:  6.191

Final   least square norm of the residuals:  1.016

Final MAX error = 28.41 %      Final RMS error = 16.06 %

Parameter      Guess          Best           Initial

---------      -----          ----           -------

TRAN.BR        10.74          8.074          4.103

TRAN.IKR       2.566m         2.251m         147.3u

TRAN.ISC       685.6a         100.0a         15.69f

TRAN.NC        1.330          1.593          1.857

PAR.IS         60.53a         68.88a         30.00a

Best:  RMS Error = 30.00 %

Guess: RMS Error = 16.06 %

Guess is Accepted
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
========== Random Iteration 2 ==========
========== Random Guess ==========
========== Levenberg-Marquardt Optimization 4 ==========
.
.
.
Number of function evaluations :  10
Number of jacobian evaluations :   1
The Levenberg-Marquardt parameter is      :  109.8m
Exit Status: No minimum was reached after specified
function evaluations
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
========== Final Values ==========
Parameter      Value          Initial        Sensitivity
---------      -----          -------        -----------
TRAN.BR        10.61          46.07          11.34m
TRAN.IKR       5.738m         6.370m         13.45m
TRAN.ISC       3.104p         3.248p         62.64m
TRAN.NC        1.458          1.456          910.5m
PAR.IS            13.78a 44.37a 2.034m
Initial least square norm of the residuals:  5.826
Final   least square norm of the residuals:  5.810
Final MAX error = 122.7 %     Final RMS error = 91.87 %
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        10.61          10.74          4.103
TRAN.IKR       5.738m         2.566m         147.3u
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
TRAN.ISC      3.104p        685.6a        15.69f
TRAN.NC       1.458         1.330         1.857
PAR.IS        13.78a        60.53a        30.00a
Best:  RMS Error = 16.06 %
Guess: RMS Error = 91.87 %
Guess is Rejected
========== Opposite Guess ==========
========== Levenberg-Marquardt Optimization 5 ==========
.
.
.
Number of function evaluations :  10
Number of jacobian evaluations :   1
The Levenberg-Marquardt parameter is      :  63.11m
Exit Status: No minimum was reached after specified
function evaluations
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
========== Final Values ==========
Parameter     Value         Initial       Sensitivity
---------     -----         -------       -----------
TRAN.BR       10.57         26.59         2.973m
TRAN.IKR      10.32m        11.30m        5.854m
TRAN.ISC      3.110p        3.247p        32.53m
TRAN.NC       1.457         1.456         958.2m
PAR.IS        1.000a        76.69a        430.0u
Initial least square norm of the residuals:  5.805
Final   least square norm of the residuals:  5.762
```

Table 27 Hybrid (Random/LM) Optimization (continued)

```
Final MAX error = 122.1 %     Final RMS error = 91.10 %
Parameter      Guess          Best           Initial
---------      -----          ----           -------
TRAN.BR        10.57          10.74          4.103
TRAN.IKR       10.32m         2.566m         147.3u
TRAN.ISC       3.110p         685.6a         15.69f
TRAN.NC        1.457          1.330          1.857
PAR.IS         1.000a         60.53a         30.00a
Best:  RMS Error = 16.06 %
Guess: RMS Error = 91.10 %
Guess is Rejected
Random Opt Exit Status: Iteration limit reached
========== Final Values ==========
Parameter      Value          Initial
---------      -----          -------
TRAN.BR        10.74          4.103
TRAN.IKR       2.566m         147.3u
TRAN.ISC       685.6a         15.69f
TRAN.NC        1.330          1.857
PAR.IS         60.53a         30.00a
Final: RMS Error = 16.06 %
```

Table 28 Sensitivity Analysis

```
========== Sensitivity Analysis Selected ==========
Number of data points used: 40 of 40
========== Calculating Sensitivities ==========
========== Jacobian Calculation ==========
```

Table 28 Sensitivity Analysis

Parameter	Value	Sensitivity
---------	-----	-----------
TRAN.BR	13.59	146.0m
TRAN.IKR	2.538m	78.65m
TRAN.ISC	7.568f	40.64m
TRAN.NC	1.575	528.2m
PAR.IS	59.03a	206.5m

Weighted Optimization

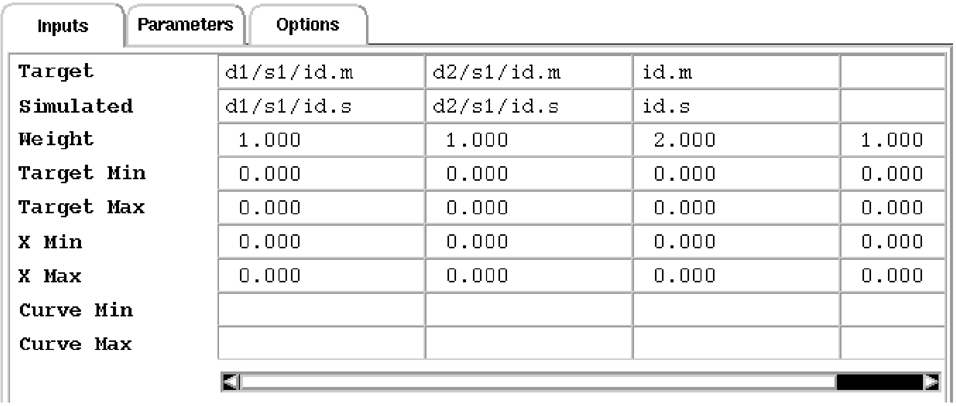
Each data set being optimized can be assigned a different weight. For example, suppose data sets A and B are being optimized simultaneously and A has a weight of 1 while B has a weight of 2. Data set B is assumed to be twice as important as A, and the optimizer tries twice as hard to fit B.

In weighted optimization the error vector in the optimizer is directly weighted, so weighting can be used in absolute error or percent error optimization. Since the error vector is used to calculate the Jacobian (sensitivities) and terminating conditions, these are affected also.

All of the weights are normalized to 1.00 during the optimization, so weights of 1.00 and 2.00 have the same effect as 0.50 and 1.00. This means that data with a normalized weight of less than 1.00 has a lower sensitivity to the parameters being optimized and looser terminating conditions. If the specified terminating condition is a one percent error, data with a normalized weight of 0.50 only needs to be below two percent error to cause the optimization to terminate.

Multiple Setup Optimization

Optimizations can be performed using the measured or target data from more than one setup by entering more than one target and simulated input pair in the Inputs table. By using this feature, you can use different type measurements from different sized devices in the optimization process. The following figure shows an example of a multiple setup optimization.



Target	d1/s1/id.m	d2/s1/id.m	id.m	
Simulated	d1/s1/id.s	d2/s1/id.s	id.s	
Weight	1.000	1.000	2.000	1.000
Target Min	0.000	0.000	0.000	0.000
Target Max	0.000	0.000	0.000	0.000
X Min	0.000	0.000	0.000	0.000
X Max	0.000	0.000	0.000	0.000
Curve Min				
Curve Max				

Figure 28 Multiple Setup Optimization Transform Specification

Optimization Time

This section explains optimization time for Levenberg-Marquardt, Random, and Hybrid (Random/LM) optimization.

Levenberg-Marquardt Optimization

The largest amount of time spent by the optimizer is in performing a function evaluation (specified as *Function_eval*). *Function_eval* calls the model a number of times in order to generate the data points defined by the setup.

The following equations illustrate the relationship between the number of calls to the model (specified as *Model_eval*) and the other factors involved in the optimization. *Model_eval* is an integer that is proportional to the time it takes to perform an optimization.

I = number of iterations

A = number of parameters optimized when calculating Jacobian

A = 0 when updating Jacobian by rank 1

Function_eval = I + 1 + (I • A)

Model_eval = total number of data points per Setup • Function_eval

To help reduce optimization time, use good initial values for parameters, use the least practical number of data points, and avoid optimizing large numbers of parameters at the same time.

For relatively simple functions that can be represented by a few equations, a transform can be written to evaluate the function by simply solving the equations using specified parameter values and bias values. By entering the name of this function transform into the simulated field of the optimizer Inputs table, and the name of the measured data set into the Measured field, the optimization is performed using the transform function's equation playback instead of a simulation for each iteration.

The advantage is that a function executes in much less time than a simulation. The *cj* setup in the *bjt_npn* model provides an example of this.

Random Optimization

There is no Jacobian calculation in a Random optimization, so the optimization time is not affected by the number of parameters optimized. A random guess and function evaluation occurs at the start of each iteration. If the guess reduces the error, the next iteration begins. If the error is not reduced, the parameter values are moved in the opposite direction of the guess and the function is reevaluated. As a result, each iteration will contain either one or two function evaluations, regardless of the number of parameters that are optimized.

However, the additional degrees of freedom that result from additional parameters generally result in a larger number of iterations to achieve the specified exit condition. This effect can be controlled by bounding the parameters as tightly as possible.

Hybrid (Random / LM) Optimization

In Hybrid (Random/LM) optimization, each function evaluation of Random optimization is replaced by a full Levenberg-Marquardt optimization. As a result, Hybrid (Random/LM) optimization begins with a Levenberg-Marquardt optimization. If the initial optimization is successful, no Random optimization is performed and the optimization time is identical to that of performing a standard Levenberg-Marquardt optimization. If the initial optimization is not successful, the optimization time depends on how many iterations are required.

One possible mechanism for speeding up Hybrid (Random/LM) optimization is by reducing the maximum number of function evaluations allowed for Levenberg-Marquardt optimization (Max Evals). This mechanism causes the optimizer to take another random guess if it is not able to reduce the error quickly. This same effect can also be achieved by increasing the Function Tol parameter.

The Optimizer Algorithm

This section explains the optimization algorithms included in IC-CAP.

Random Optimization Algorithm

The Random optimizer uses a simplified stochastic gradient algorithm. Initially an update vector is chosen using a spherical Gaussian distribution (that is, one that favors no particular direction in the optimization variable space).

A trial point is calculated by adding the update vector to the current point and then the cost is calculated at the trial point. If the cost is less than the initial cost, the trial point is accepted and the Gaussian distribution is modified to an ellipsoid that favors the update direction. If the cost is not decreased, a new trial point is calculated by subtracting the update direction from the initial point and the cost is recalculated. If the cost decreases, then the trial point is accepted and the Gaussian distribution is modified to favor the update direction. If the cost does not decrease, the trial point is rejected and the Gaussian distribution is modified so that the unsuccessful update direction is less favored. The whole process is then repeated until either the RMS is reduced to the specified value or the number of iterations is specified by the *Rand Iter* parameter is reached.

If there are several successive successful steps, then the Gaussian distribution becomes elongated in the direction of the jumps. Several successive failures tend to shrink the Gaussian distribution in the direction of the failed jumps. Because of this, with success, the Random optimizer acts more aggressively in its steps and, with failure, it acts more conservatively.

The *Rand Reward* parameter controls how much the Gaussian distribution is expanded upon a successful step. A large value increases the expansion. A value of zero eliminates any change in the Gaussian distribution. Likewise, the *Rand Penalty* parameter controls the compression effects of an unsuccessful step.

Hybrid (Random / LM) Optimization Algorithm

The Hybrid (Random/LM) optimizer is a combination of the Random and Levenberg-Marquardt optimizers. It is the Random optimizer modified so that, at each of its error evaluations, it calls the Levenberg-Marquardt optimizer to evaluate the RMS error.   The parameters specified in the Options Tables that effect the Levenberg-Marquardt and Random optimizers will have the same effects during Hybrid (Random/LM) optimization.

Sensitivity Analysis Method

The Levenberg-Marquardt optimizer normally prints the sensitivity analysis information every time the Jacobian is calculated. The optimizer options choices *Normalize Sens* and *Print Sens Results* control the form and presentation of this information. For more information, see “Defining Optimizer Options” on page 305.

The displayed sensitivities are the normalized RMS partial derivatives of the error vector (difference between measured and simulated data) with respect to each of the parameters being optimized. The RMS partial derivatives are normalized to each other, then all are normalized to 1. For example, suppose two parameters are being optimized, and one has a sensitivity of 0.67 while the other has a sensitivity of 0.33. This means that a change in the first parameter will have approximately twice as much effect on the data as an equal percentage change in the second parameter.

One possible use of sensitivities is in defining an optimization strategy. Successful optimizations require a reasonable balance of sensitivities between the parameters. For example, if four parameters are being optimized and one of the parameters has sensitivity of 0.99, it will be very difficult to determine useful values for the other three parameters. This situation suggests that the dominant parameter must be accurately extracted before optimizing the values of the other three parameters.

Sensitivity Analysis Data

While in sensitivity analysis mode, the partial derivatives at each bias point are stored as transform data, provided that the following conditions are met:

- There is only one set of target and simulated data sets.
- There is no bounding of the data in the X or Y dimensions.

At each bias point an N by 1 matrix, where N is equal to the number of parameters specified in the Param Table, is used to store the data. For example, for a sensitivity analysis transform named *sens* where there are three parameters, ISE, BF, and IKF, the data is stored as follows:

- ISE data is stored in sens.11
- BF data is stored in sens.21
- IKF data is stored in sens.31

This data can be plotted along with the target and simulated data, as shown in the following figure. This feature is particularly useful for illustrating the interdependencies between parameters and determining the region where each parameter has its largest effect on the simulated data.

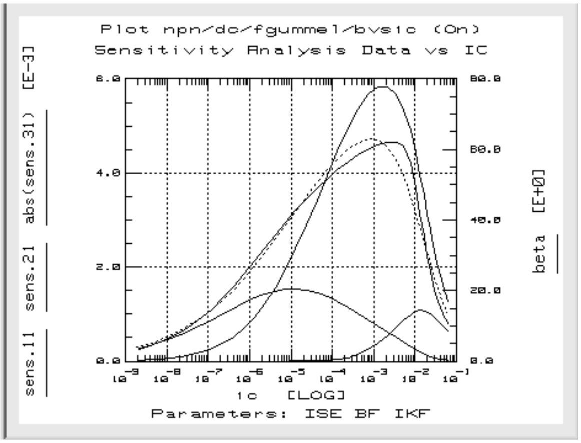


Figure 29 Plot of Sensitivity Analysis Data *Versus* IC

Extraction Versus Optimization

IC-CAP makes a distinction between parameter extraction and parameter optimization. Parameter extraction (often used to refer generically to the process of obtaining model parameters) is defined more narrowly in the discussion of the IC-CAP system. Here, parameter extraction refers to the process of using device or circuit model specific algorithms to obtain model parameters directly from measured data. In contrast, parameter optimization is a general purpose, model independent process for numerically adjusting model parameter values. Parameter optimization keeps to a minimum some cost function (usually the difference between the model's simulated characteristics and the corresponding circuit measured characteristics).

The distinction between parameter extraction and optimization is an important concept to understand in discussing the IC-CAP methodology for extraction.

Global Optimization

A drawback to blind global optimization is that since the optimization routines know nothing about device physics or about the meaning of the model parameters, optimizing all of the model parameters to all of the measured data at once without any starting value determination often produces meaningless results. Also, global optimization consumes a great deal of computing power to analyze the myriad of possible parameter combinations without any model specific guidance. Many times, final parameter values are caught in local minima resulting in inaccurate values. Much time can be wasted attempting to optimize parameters to data on which they have no physical effect. Furthermore, parameters that have similar effects on the device characteristics (such as, RE and RB for the Bipolar model) are usually determined incorrectly. For these reasons, global optimization should not be performed in IC-CAP, although it can be done.

If global optimization is performed, Random or Hybrid (Random/LM) optimization should be used to avoid local minima. Random optimization can be used prior to Levenberg-Marquardt optimization to reduce the error to a reasonable starting value that is considered to be below that of any local minima. In Hybrid (Random/LM) optimization, random parameter jumps take place if the Levenberg-Marquardt optimizations are unsuccessful. When a local minimum is found, a random jump attempts to bypass it.

IC-CAP Extraction Methodology

The approach used in IC-CAP resolves the deficiencies of global optimization by using a parameter determination method. Specifically, any good parameter determination method must provide three very important things:

- Computational efficiency to minimize extraction run times
- Device and circuit model specific knowledge inserted into the process for obtaining parameter values to ensure that meaningful parameter values result
- Accuracy of the resulting parameters to ensure accurate circuit simulation even for analog designs

Device or circuit operation is divided into several different regions. These regions are selected so that subsets of the model parameters have a clear effect on the model characteristics. This mapping of model parameter subsets onto regions of device operation where they have a dominant effect is referred to as an extraction strategy. Sensitivity analysis can be used as an aid in determining the proper regions. These regions of operation are predefined for the IC-CAP extraction Transforms. However, you may select these regions of operation for new or modified extractions. After the regions have been defined and the parameters with the greatest influence in those regions are selected, the actual parameter calculation process can begin.

Model Specific Extraction Routines

IC-CAP uses a 2-step model dependent parameter extraction process shown in the following figure.

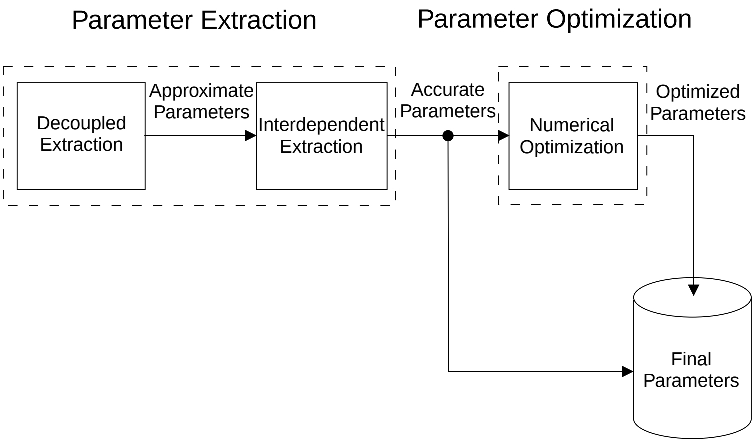


Figure 30 IC-CAP Parameter Extraction Process

The purpose of parameter extraction is to obtain the most accurate model parameters possible directly from the measured data. The model-specific extraction algorithms solve the model equations in reverse. That is, given the measured electrical characteristics, the values of the model parameters that reproduce these characteristics must be determined. To accomplish this, a set of decoupled algorithms determine the initial values of the subset of the model parameters relevant in a given region. This decoupled extraction assumes no interaction between the model parameters and employs the use of curve fitting techniques.

To resolve the parameter interdependencies, an interdependent parameter extraction algorithm must be used. Iteration using model-specific equations is performed to balance the effects between interacting parameters.

The result of this 2-step process is a set of accurate parameters which can often be used without further modification in some applications.

Partial Optimization

The IC-CAP optimizer can fine tune extracted parameters. The optimization, known as partial optimization, is applied only to the parameters with dominant effects in a given region of operation. Since the initial values for the optimization now contain model specific information provided by the parameter extraction transforms, results from the optimization process are more accurate. Also, run time is minimized since the initial parameters are very close to the absolute minimum of the cost function represented by the difference between the measured and simulated data.

The optimizer is most effective when used in conjunction with model specific extraction transforms. However, several special features of the optimizer allow control of the solution in a manner that makes optimization starting from a less accurate initial value estimate feasible. The IC-CAP optimizer transform provides parameter constraints, data selection controls and convergence controls to tailor the optimizer's performance to the specific application.   You can set these features in the optimizer transform editor.

References

Jorge More, *The Levenberg-Marquardt Algorithm: Implementation and Theory,* Numerical Analysis: Seventh Biannual Conference, University of Dundee, Scotland, Springer Verlag, N.Y., 1977.

8
Using the Plot Optimizer

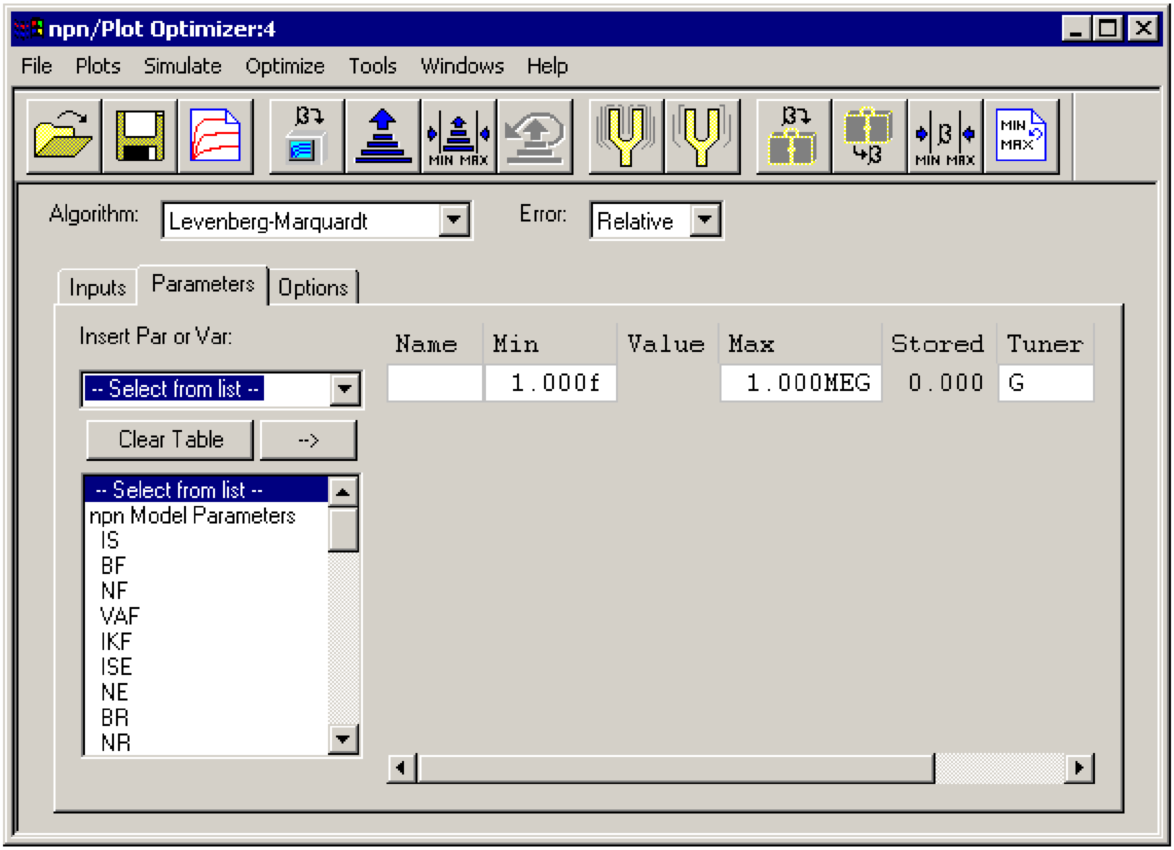
This chapter describes how to use the Plot Optimizer. For descriptions of optimization algorithms, search methods, and error functions, see Chapter 7, "Optimizing."

The Plot Optimizer enables you to dynamically create optimizations simply by selecting the data you wish to optimize from an XY Data plot, Real/Imaginary plot, Smith plot, or Polar Chart plot.

You can open the Plot Optimizer window from a Model window or from a Plot window.

- To open a Plot Optimizer window from a Model window, choose **Tools > Plot Optimizer**
- To open a Plot Optimizer window from a Plot window, choose **Optimizer > Open Optimizer**

Each Plot Optimizer window is associated with one Model window. For each Model window, you can open one Plot Optimizer window.



Algorithm: Levenberg-Marquardt

Error: Relative

Insert Par or Var: -- Select from list --

Name	Min	Value	Max	Stored	Tuner
	1.000f		1.000MEG	0.000	G

-- Select from list --
npn Model Parameters
IS
BF
NF
VAF
IKF
ISE
NE
BR
NR

The basic steps required to perform an optimization/tuning using the Plot Optimizer are:

- Open and enable plots
- Specify the optimizer inputs
- Specify the parameters to be optimized
- Specify the optimizer options
- Run the optimization

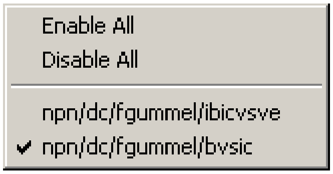
The following sections describe each of these steps.

Enabling Plot Windows

All plot optimizer inputs, except *Curve Min, Curve Max,* and *Weight,* are defined using Plot windows. Before you can use a Plot window to define plot optimizer inputs, you must enable the Plot window. Enabling the Plot window synchronizes it with the Plot Optimizer window. You can synchronize one or more Plot windows in a model file with the Plot Optimizer window. You can then use the Plot windows to configure optimizer inputs and to set limits for target and X-axis values.

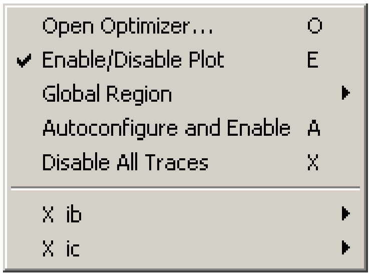
Plots in a model file can be enabled from the Plot Optimizer window or from the Plot window.

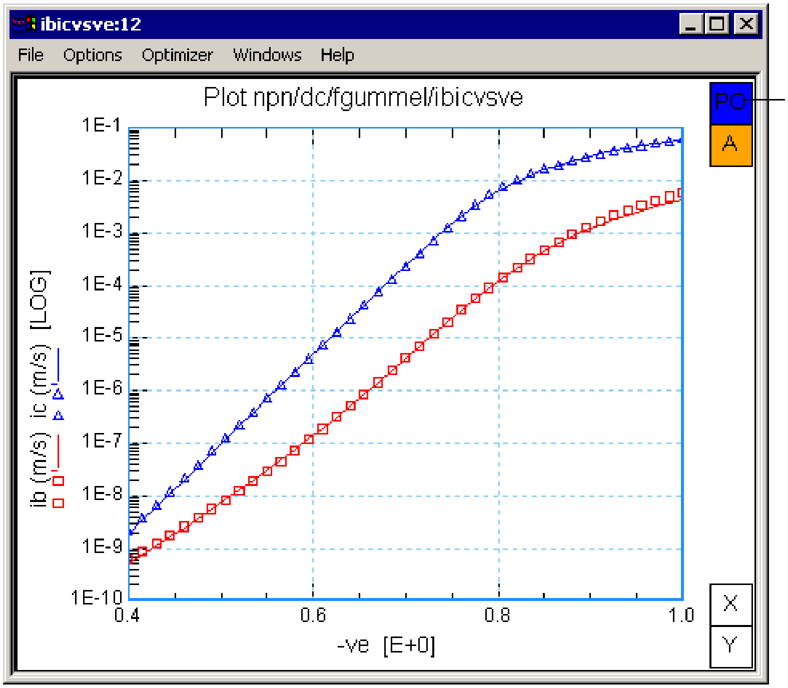
- To enable all open Plot windows in a model file from the Plot Optimizer window, choose **Plots > Enable All**.
- To enable a single Plot window from the Plot Optimizer window, choose **Plots** then select the plot name from the menu. A check mark next to the menu item indicates that the plot is enabled.



- To enable the plot from a Plot window either:
  - Choose **Optimizer > Enable / Disable Plot**
  - Choose the **PO** area tool if Area Tools are enabled. To enable Area Tools, choose *Options > Area Tools.*

  A check mark before the *Enable/Disable Plot* menu item and a blue border around the plot indicates that the plot is enabled.

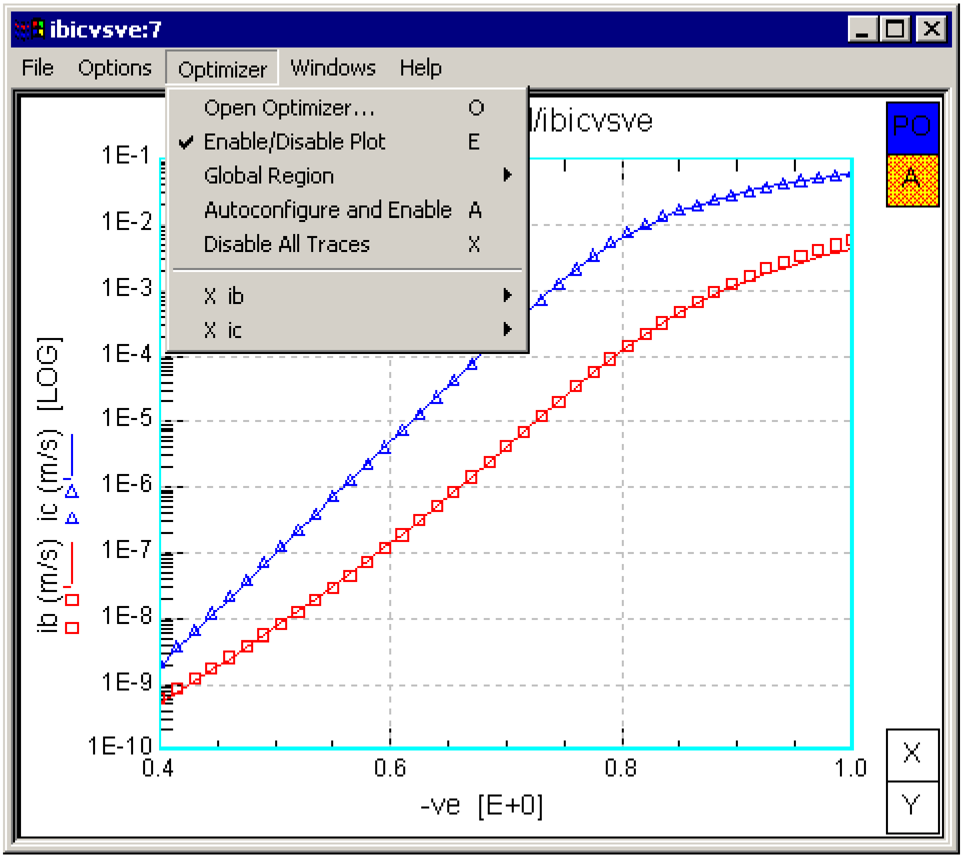




Plot Optimizer Enable/Disable Plot box

Configuring Plot Optimizer Inputs

Traces are normally disabled when the plot is first defined. An **X** before the trace name in a Plot window's Optimizer menu indicates that the trace is disabled. Disabled traces are not used or synchronized with the Plot Optimizer.



Before a trace can be used in the Plot Optimizer, it must be configured as either Target or Simulated based on its definition.

Traces are defined by typing an equation into the Y or X Data fields in the Plot Editor. An equation uses datasets (e.g., inputs, outputs, or transform results), constants, and variables. Similar to a transform, evaluating an equation results in a dataset. Depending on the datasets used to define the equation, the resulting dataset is one of the following four dataset types:

- MEAS—includes only measured data
- SIMU—includes only simulated data
- BOTH—includes both measured and simulated data
- COMMON—includes common data

Usually MEAS, SIMU, and BOTH dataset types are the result of evaluating equations that use outputs. COMMON dataset types can be the result of equations that use input sweeps or a program.

The equation's Y or X Axis Type determines which functions are used to insert the trace equation into the Plot Optimizer.

- If a trace equation is defined as *Y Data 0* to *Y Data 7* and the *Y Axis Type* is LINEAR, then the equation is inserted without using a function except for the *mdata()* and *sdata()* functions which may be used to separate measured and calculated data.
- If a trace equation is defined as *Y Data 0* to *Y Data 7* and the *Y Axis Type* is LOG10, then the equation is inserted using the *LOG10()* function.

  For example: equation *ib* of dataset type BOTH in a LOG10 plot is inserted as:

  ```
  Target: LOG10(ABS(mdata(ib)))
  Simulated: LOG10(ABS(sdata(ib)))
  ```

NOTE

The *ABS()* function is used to avoid *LOG10()* or *DB()* of negative numbers.

- If a trace equation is defined as *Y Data 0* to *Y Data 7* and the *Y Axis Type* is DB, then the equation is inserted using the *DB()* function.

  For example: equation *mag(s.21)* of dataset type BOTH in a DB plot is inserted as:

  ```
  Target: DB(ABS(mdata(mag(s.21))))
  Simulated: DB(ABS(sdata(mag(s.21))))
  ```

Trace equations defined as *Y2 Data* follow the same rules depending on how the *Y2 Axis Type* is defined (LOG10, LINEAR or DB).

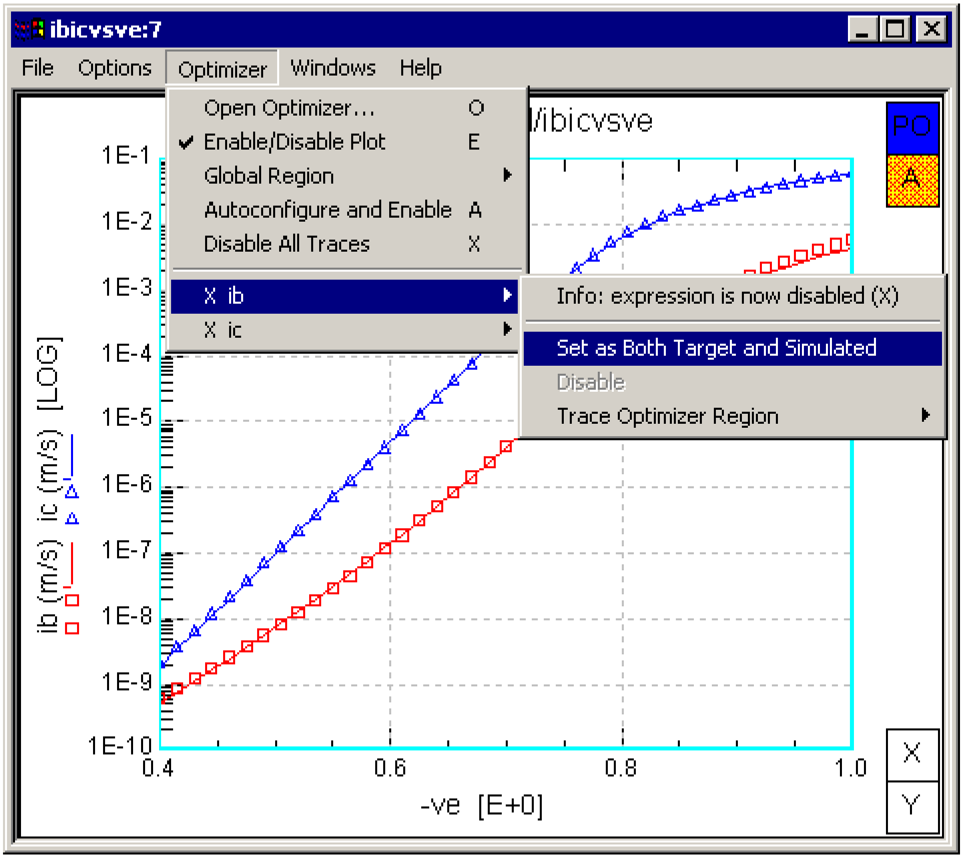
Setting the system variable PLOTOPT_USE_YAXES_TYPE to No (the default is Yes) prevents the Plot Optimizer algorithm from using the LOG10() and DB() functions regardless of the Axis Type.

Manually Configuring Optimizer Inputs

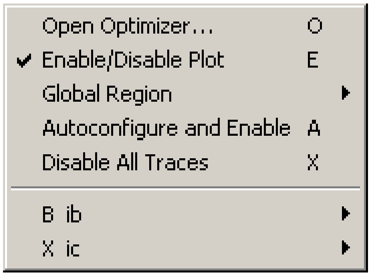
To manually configure the inputs, choose **Optimizer** then select a trace from the menu. Depending on the trace you select, you will have one or more of the following choices:

- Set as Both Target and Simulated
- Set as Target vs.
- Set as Simulated vs.

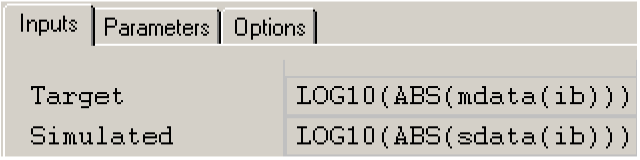
If you selected a BOTH dataset type trace, your choice is *Set as Both Target and Simulated*.



After selecting *Set as Both Target and Simulated*, a **B** is displayed before the trace name in the Optimizer menu.



In the Plot Optimizer Inputs table, the trace is configured with its measured data set as Target and its simulated data set as Simulated. The *mdata()* and *sdata()* functions fill the Inputs table with the correct data. For example, if the trace *ib* is set to *Both Target and Simulated*, the Inputs table is configured as shown in the following illustration:

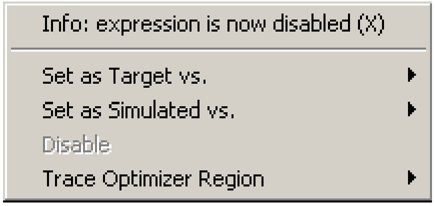


Target	LOG10(ABS(mdata(ib)))
Simulated	LOG10(ABS(sdata(ib)))

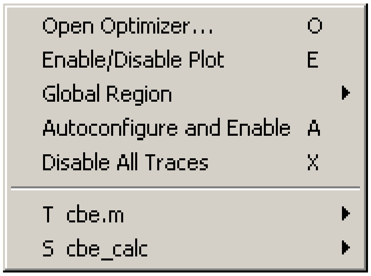
If you selected a MEAS dataset type trace, your choice is *Set as Target vs.* After selecting *Set as Target vs.*, select a trace from the resulting drop-down menu. Your choices will be SIMU or COMMON dataset type traces.

If you selected a SIMU dataset type trace, your choice is *Set as Simulated vs.* After selecting *Set as Simulated vs.*, select a trace from the resulting drop-down menu. Your choices will be MEAS or COMMON dataset type traces.

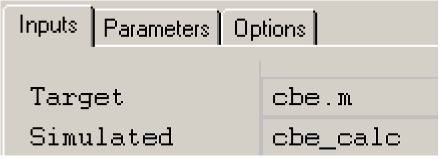
If you selected a COMMON dataset type trace, your choices are *Set as Target vs.* or *Set as Simulated vs.* After selecting *Set as Target vs.* or *Set as Simulated vs.*, select a trace from the resulting drop-down menu. Your choices will be MEAS or COMMON dataset type traces.



After configuring a trace as *Target*, a **T** is displayed before the trace name in the Optimizer menu. After configuring a trace as *Simulated*, an **S** is displayed before the trace name in the Optimizer menu.



The Target trace and its associated Simulated trace are inserted together in the Plot Optimizer Inputs table.



Target	cbe.m
Simulated	cbe_calc

NOTE

If you disable a Target or Simulated trace, its associated trace is also disabled and both entries are deleted in the Plot Optimizer Inputs table. To disable a trace, choose **Optimizer** select a trace, then choose **Disable.**

Automatically Configuring Optimizer Inputs

To automatically configure the traces in a Plot window, choose **Optimizer > Autoconfigure and Enable.**

Traces that meet the following criteria are automatically configured:

- If a plot has only two traces and one trace's dataset type is MEAS and the other trace's dataset type is SIMU or COMMON, then the MEAS dataset type trace is set as *Target* versus the other trace regardless whether it was a SIMU or COMMON dataset type.

  For example:

  ```
  Y Data 0:  id.m
  Y Data 1:  id_calc
  ```

  results in:

  ```
  T id.m
  S id_calc
  ```

- If a trace's dataset type is BOTH, then that trace is configured as *Both Target and Simulated.*
- If a plot has multiple traces and one trace's dataset type is MEAS, then an algorithm searches for an SIMU dataset with the same name. If found, the two traces are paired with the MEAS dataset type trace set as *Target* versus the SIMU dataset type trace.

To disable all traces in a Plot window, choose **Optimizer > Disable All Traces**.

Defining Limits for Target and X-axis Values

You can set limits for all target and X-axis values in a plot by defining a global trace region. You can also set limits for specific target and X-axis values in a plot by defining a trace optimizer region.

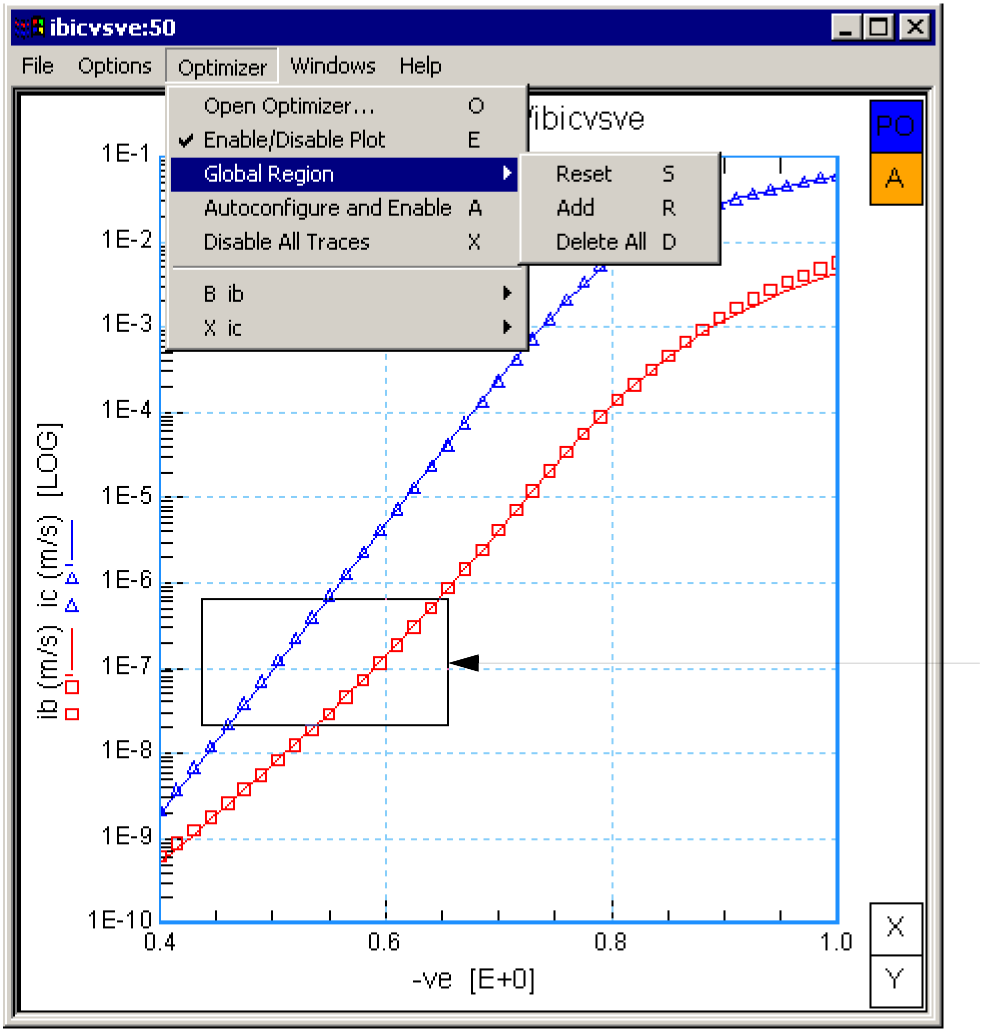
Global Region

To define a trace region for all traces in the plot, form a rectangle of the desired trace region then:

- Choose **Optimizer > Global Region > Reset** to delete all existing global trace regions on the plot and define a new global trace region.

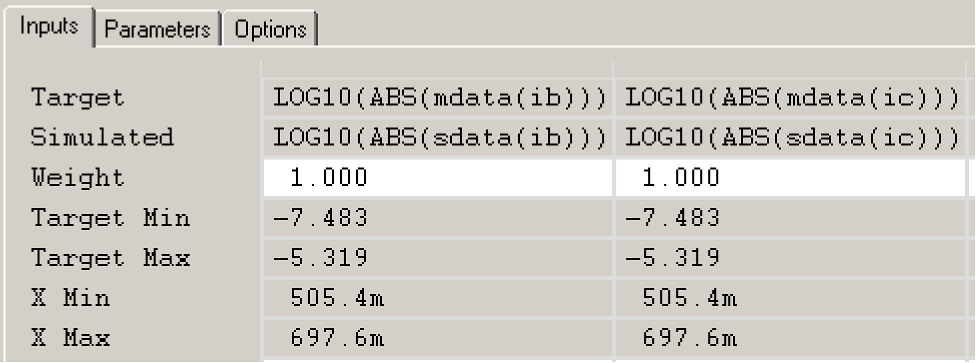
  or

- Choose **Optimizer > Global Region > Add** to add a global trace region without deleting existing global trace regions.



Form a rectangle of the trace region

The target and X-axis values for all traces are automatically set to the trace region defined by the rectangle.



Target	LOG10(ABS(mdata(ib)))	LOG10(ABS(mdata(ic)))
Simulated	LOG10(ABS(sdata(ib)))	LOG10(ABS(sdata(ic)))
Weight	1.000	1.000
Target Min	-7.483	-7.483
Target Max	-5.319	-5.319
X Min	505.4m	505.4m
X Max	697.6m	697.6m

To remove all global trace regions, choose **Optimizer > Global Region > Delete All**.

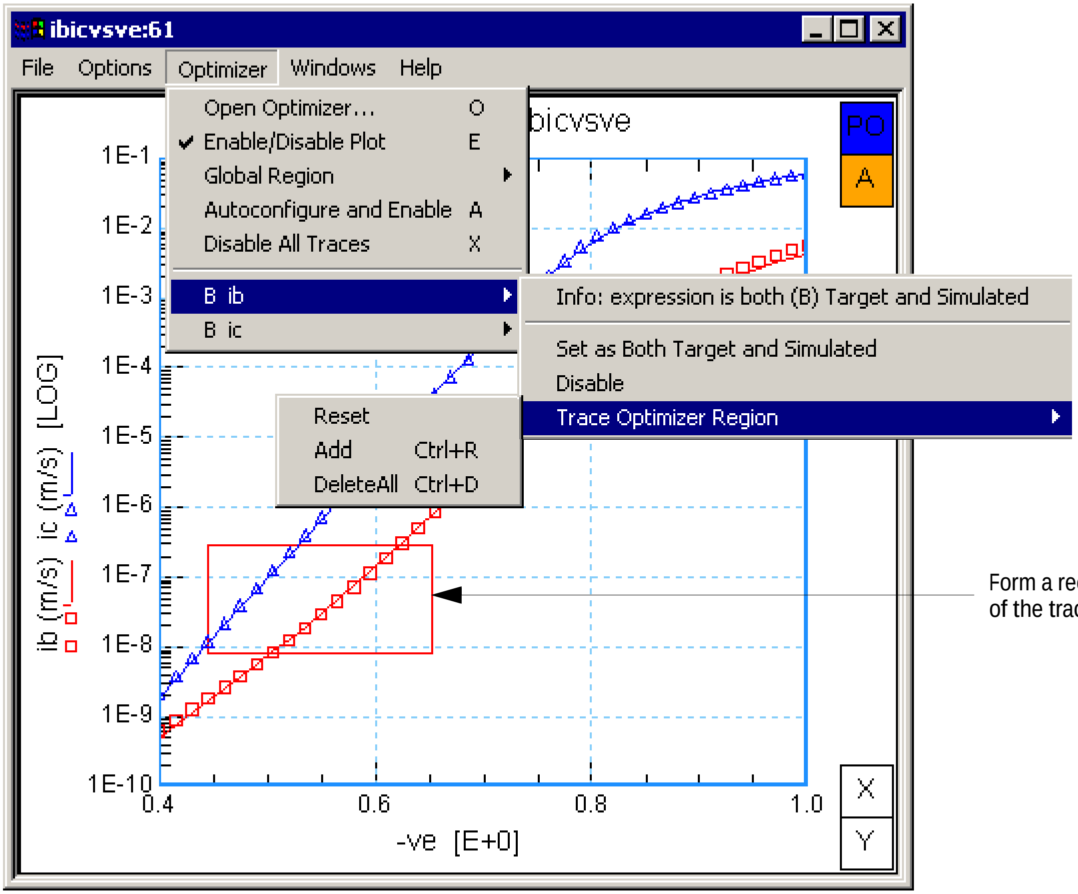
Trace Optimizer Region

To define a trace region for a specific trace in a plot, form a rectangle of the desired trace region then:

- Choose **Optimizer**, select a trace, then choose **Trace Optimizer Region > Reset** to delete all existing trace optimizer regions for the selected trace and define a new trace optimizer region.

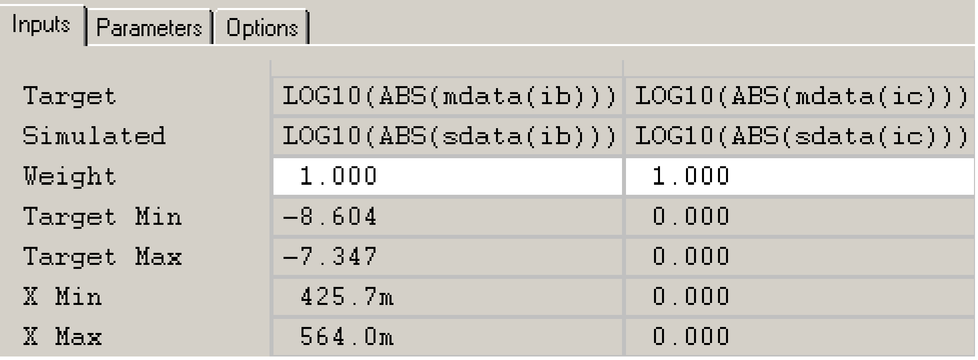
  or

- Choose **Optimizer**, select a trace, then choose **Trace Optimizer Region > Add** to add a trace optimizer region.



Form a rectangle of the trace region

The target and X-axis values for the selected trace are automatically set to the trace optimizer region defined by the rectangle.



Target	LOG10(ABS(mdata(ib)))	LOG10(ABS(mdata(ic)))
Simulated	LOG10(ABS(sdata(ib)))	LOG10(ABS(sdata(ic)))
Weight	1.000	1.000
Target Min	-8.604	0.000
Target Max	-7.347	0.000
X Min	425.7m	0.000
X Max	564.0m	0.000

To remove all trace regions for a trace, choose **Optimizer**, select a trace, then choose **Trace Optimizer Region > Delete All**.

Defining Curve Min, Curve Max, and Weight

You can define Curve Min, Curve Max, and Weight optimizer inputs by completing the Inputs table.

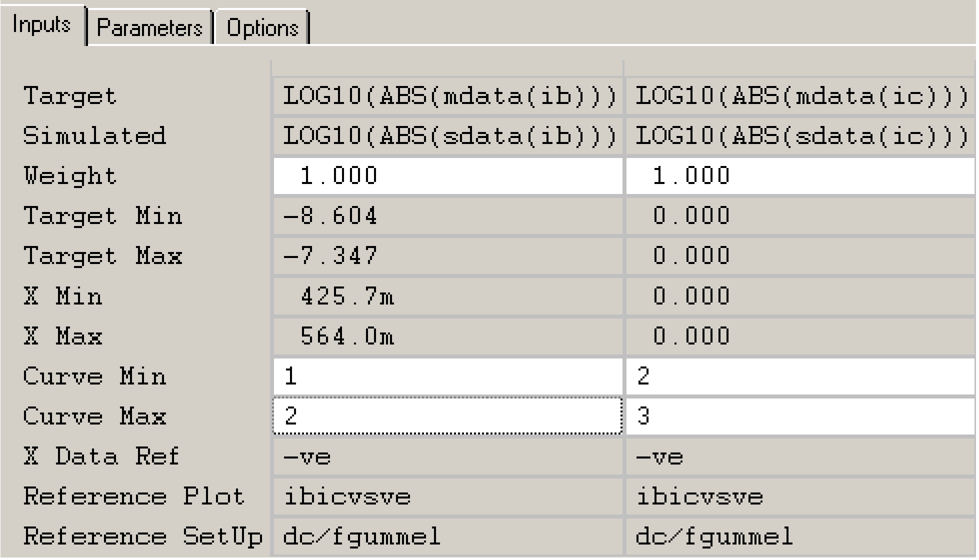
NOTE If you disable then re-enable a plot or trace, the values for that plot or trace are reset to their default values.

The following describes the fields that can be edited in the Inputs table:

Weight Relative weighting factor to apply to this data.

Curve Min Lower limit for the curve. For a family of curves, curves numbered lower than this limit are ignored during optimization. A single curve can be selected by entering the same curve number in both the Curve Min and Curve Max fields.

Curve Max Upper limit for the curve. For a family of curves, curves numbered higher than this limit are ignored during optimization.



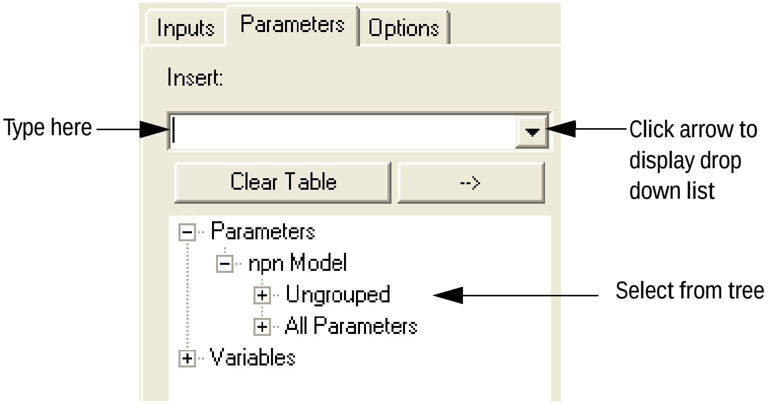
Target	LOG10(ABS(mdata(ib)))	LOG10(ABS(mdata(ic)))
Simulated	LOG10(ABS(sdata(ib)))	LOG10(ABS(sdata(ic)))
Weight	1.000	1.000
Target Min	-8.604	0.000
Target Max	-7.347	0.000
X Min	425.7m	0.000
X Max	564.0m	0.000
Curve Min	1	2
Curve Max	2	3
X Data Ref	-ve	-ve
Reference Plot	ibicvsve	ibicvsve
Reference SetUp	dc/fgummel	dc/fgummel

Defining Plot Optimizer Parameters

Inserting a Parameter or Variable

You can insert parameters and variables in the Parameters table in several ways:

- Type in the name of a parameter or variable in the *Insert Param or Var* text field, then press Enter.
- Type in the first part of the name using the desired capitalization in the *Insert Param or Var* field. Capitalization is important since the sort function is case sensitive. Click the down arrow to display a list of variables and parameters that begin with the letters and cases you typed. Select the variable or parameter from the drop-down list.
- From the tree, double-click on a parameter or variable, or select a parameter or variable then choose the --> button.
- From the tree, select a group to select all parameters or variables in the group, then choose the --> button.
- From the tree, press the Ctrl key while selecting multiple groups, parameters or variables then choose the --> button.

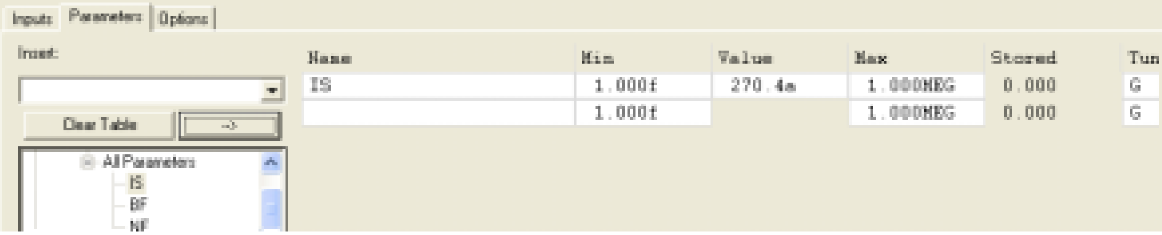


When a parameter is inserted, its Min and Max values are automatically set to:

- The Min and Max values defined in the Model window’s Parameters table.

  or

- If Min and Max are blank in the Model window’s Parameters table, Min is set to 1.0 e-15 and Max is set to 1.0 MEG.



Name	Min	Value	Max	Stored	Tun
IS	1.000f	270.4m	1.000MEG	0.000	G
	1.000f		1.000MEG	0.000	G

Setting Parameter Min, Max, and Value

To automatically set Min and Max, choose **Tools > AutoSet Min and Max** or choose the AutoSet Min and Max icon.

The Min and Max parameters are calculated as follows:

```
if (value > 0)
    Min = value/coeff
    Max = value · coeff
if (value < 0)
    Min = value · coeff
    Max = value/coeff
if (value ==0)
    Min = -coeff
    Max = +coeff
```

where *coeff* is 5 as default. To change the value of *coeff*, use the `AUTOSET_COEFF` system variable.

To manually set Min, Max, or Value, highlight the desired cell in the table then type in a new number. Remember that Min must be less than or equal to Value and Value must be less than or equal to Max at all times.

- If you attempt to set Min to greater than Max or Max to less than Min, an error message is displayed and no change occurs.
- If you attempt to set Min to greater than Value, a dialog box appears asking if you want to set Value to Min. If you select Yes, Min and Value are both changed to the new Min. If you select No, neither Min nor Value is changed.
- If you attempt to set Max to less than Value, a dialog box appears asking if you want to set Value to Max. If you select Yes, Max and Value are both changed to the new Max. If you select No, neither Max nor Value is changed.
- If you attempt to set Value outside the range of Min and Max, Value is set to Min or Max and a warning is displayed in the Status window.

Restoring Parameter Min, Max, and Value

To restore Min and Max to their default values, choose **Tools > Reset Min and Max** or choose the Reset Min and Max MIN MAX icon.

This restores Min and Max to the Min and Max values defined in the Model window’s Parameters table or Min to 1.0 e-15 and Max to 1.0 MEG if Min and Max are blank in the Model window’s Parameters table.

You can also store Values in the Parameters table for later recall. This allows you to perform multiple optimizations or tunings, then return to the stored values.

- To store parameter values, choose **Tools > Store Parameters** or choose the Store Parameters icon.
- To recall the stored parameters, choose **Tools > Recall Parameters** or choose the Recall Parameters icon.

You can also return parameters to the state they were in just before you performed an optimization or tuning.

- To restore parameters to their previous state, choose **Tools > Undo Optim** or choose the Undo Optim icon. This also preforms a simulation and updates the plots.

  This feature is inactive until you perform an optimization or tuning.

Deleting Parameters and Variables

You can delete parameters and variables from the Parameters table one at a time or all together:

- To delete a single parameter or variable, highlight to select then press the Delete key.
- To delete all parameters and variables, choose the **Clear Table** button.
- To delete all parameters and variables, disable all open plots, and disable all traces and regions in the plots, choose **File > Clear Plot Optimizer** or choose the Clear Plot Optimizer icon.

Defining Plot Optimizer Options

The Plot Optimizer Options are the same as the Optimizer Options. See “Defining Optimizer Options” on page 305 for descriptions of the available options.

To reset all values in the Options table to their default values, choose **File > Reset Option Table**.

Running an Optimization

After setting up the Inputs, Parameters, and Options folders, do the following to run an optimization.

1 Choose an Algorithm from the drop-down list.

   For a description of the available algorithms, see “Optimization Algorithms” on page 274.

2 choose **Optimize > Run Optimization** or choose the Run Optimization icon.

   Alternatively, to automatically set the minimum and maximum parameter values then run an optimization, choose **Optimize > Autoset Min / Max and Optimize** or choose the Autoset Min/Max and Optimize MIN MAX icon.

Saving and Opening a Plot Optimizer Setup

Plot Optimizer setup files can be saved then opened in the Plot Optimizer or as an optimize transform in a model file.

To save a Plot Optimizer setup:

1 Choose **File > Save As...**.

2 In the File name field, enter a unique filename and choose **Save**. A *.pop* file extension is appended automatically.

The Plot Optimizer settings along with the configuration information for each enabled plot is saved in the *.pop* file.

The saved Plot Optimizer settings include:

- Options
- Inputs table
- Parameters table, except the Values field

  The Values field is not saved since parameters and their values are defined in the Model window's DUT or Model Parameters table.

The saved configuration information for each enabled plot includes:

- Plot name
- Trace settings (B, T, S)
- Global Region definitions
- Trace Optimizer Region definitions

To open a previously saved Plot Optimizer file (*.pop*) in a Plot Optimizer window:

1 Choose **File > Open...**.

2 Change directories as needed, select the desired file, and choose **OK**.

When a *.pop* file is opened, the saved optimizer information is loaded into the local Plot Optimizer and the plots are enabled and configured according to the saved settings overriding existing trace settings or regions. If a plot can not be found, a warning is sent to the Status window and the settings for the missing plot are deleted from the Plot Optimizer Inputs table. This occurs because the Inputs table is always synchronized with the enabled plots.

To open a previously saved Plot Optimizer file (*.pop*) as an optimize transform in a Model window:

1 From a Model window, select a setup then choose the **Extract / Optimize** tab.

2 Choose **File > Open...**.

3 Select the file type **(.pop) Transform**.

4 Browse to the location of the *.pop* file, select the desired file, and choose **OK**.

When reading a Plot Optimizer file (.pop) into a normal transform, the target, simulated, and region information in the Inputs table is read into the optimizer transform's Inputs table. Reference Setup provides the context (or relative path) for the Target, Simulated, and X Data Ref datasets. This ensures that if the current setup belongs to the Model file associated with the Plot Optimizer window that saved the .pop file, the optimizer transform can be run without modifying the Inputs or Parameters tables.

9
Using Transforms and Functions

An IC-CAP transform is a framework for mathematical or logical functions that operate on data. A transform can operate on any combination of data sets, parameters, and variables. These inputs are used to calculate either a new data set or new values for parameters and variables.

IC-CAP provides a comprehensive list of functions that can be used in a transform. These functions range from simple functions, such as addition and subtraction, to advanced functions that perform parameter extractions and optimizations. Extraction functions are supplied for all standard models contained in the system.

By defining IC-CAP transforms, you can analyze data in forms other than those provided by direct measurement or simulation and transform the data into new domains. A new transform can use an existing function or a new expression or series of expressions. Many functions such as *log* (natural logarithmic function), and *derivative* (for calculating first or second derivative), are available. More complicated expressions such as $IS \cdot (exp(vd / 0.026) - 1)$ can be defined using the *equation* or *program* functions.

Some of the ways you can use IC-CAP transforms are:

- Divide two data sets to create a third data set. Then view the new data set on a plot.
- Calculate the values of several model parameters based on the inputs and outputs contained in a setup.
- Calculate linear regression information for a particular output.
- Run a fast simulation by programming model equations directly into a transform.

Since transforms can operate on all data sets, the output of one transform can be used as input to another transform. For example, you can use this feature to:

- Perform extractions or optimizations on the calculated gain of a device.
- Convert 2-port data from the network analyzer to a different domain, correct the data, then convert the corrected data back to the original domain.

While certain transforms, such as those that perform extractions, are executed with menu selections, many transforms are executed automatically when the input data sets change. For example, a transform that calculates the gain of a MOSFET is executed automatically when the measured or simulated outputs of the device change. For complete information on this topic, refer to “Performing Transforms” on page 370 and “Automatic Transform Execution” on page 370.

You can create new functions by writing programs using IC-CAP’s Parameter Extraction Language (PEL) or by using C language. Functions written in C language provide total flexibility for manipulating the data. Use this feature for many applications, such as implementing a new algorithm for extracting model parameters or reading data from another application into the IC-CAP system.

This chapter contains procedures for creating, editing, and using IC-CAP transforms and functions and describes how to augment IC-CAP's standard function library by writing additional functions in the C language. For details on writing programs using IC-CAP's Parameter Extraction Language (PEL), refer to Chapter 9, "Parameter Extraction Language," in the *Reference* manual.

Implementing New Extractions

You can implement new extraction algorithms by using the *Program2* or *Program* Transform. This Transform provides a powerful method for accessing measured data and calculating model parameters from this data. IC-CAP also includes a number of functions that greatly simplify this operation. For example, a number of rather tedious operations are replaced by functions that calculate derivatives, or least-squares fit to the data. You can create new extractions simply by assembling a series of existing IC-CAP functions.

C functions can be used to increase both the performance and functionality of the above operations. Most of IC-CAP's existing extraction functions are implemented in the C language.

The optimizer can also be used for extracting model parameters. To do this, define Optimize Transforms for the Setups and specify in them the model parameters to be optimized. For further information, refer to Chapter 7, "Optimizing."

Customizing IC-CAP

Implementing New Model Equations

Test new models in IC-CAP by defining them in the form of a series of equations in a *Program* Transform or a User C function. Implement a complete model or subset of a model in this way. To define a model, calculate the node current or currents, using the node voltages and model parameters. An example of this is the *BJT_dc_model* function in the *userc.c* file.

To simulate a model defined in this manner, execute the Transform. After the model is fully tested, implement it in the circuit simulator.

Implementing New Extractions

You can implement new extraction algorithms by using the *Program2* or *Program* Transform. This Transform provides a powerful method for accessing measured data and calculating model parameters from this data. IC-CAP also includes a number of functions that greatly simplify this operation. For example, a number of rather tedious operations are replaced by functions that calculate derivatives, or least-squares fit to the data. You can create new extractions simply by assembling a series of existing IC-CAP functions.

C functions can be used to increase both the performance and functionality of the above operations. Most of IC-CAP's existing extraction functions are implemented in the C language.

The optimizer can also be used for extracting model parameters. To do this, define Optimize Transforms for the Setups and specify in them the model parameters to be optimized. For further information, refer to Chapter 7, “Optimizing.”

Defining Transforms

Transforms are created as members of setups, since they typically operate on data from a particular setup.

To define a transform:

1 In the Model window, select the setup.

2 Select **Extract / Optimize**.

3 Click **New**. New...

4 In the prompt, type a name for the new transform. This name is used to refer to the data set created by this transform.

5 Click **OK**. The new name appears in the transform list and is selected.

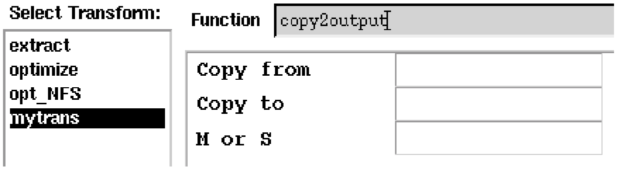
6 In the Function field, enter the name of the function to be used for the new transform by doing one of the following:

- Click **Browse** to select a predefined function from the list.
- Type `Program2` or `Program` and press Enter to create your own function.

NOTE

For details on making user-defined functions appear in the Function Browser, refer to "Adding Functions to the Function Browser" on page 382.

7 If inputs are required for the function, specify the inputs in the editor fields.



Select Transform:

extract
optimize
opt_NFS
mytrans

Function copy2output

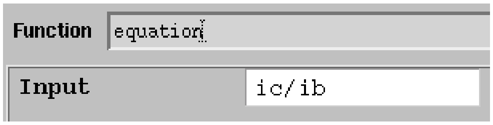
Copy from
Copy to
M or S

Assign a name

Enter inputs

Example Transforms

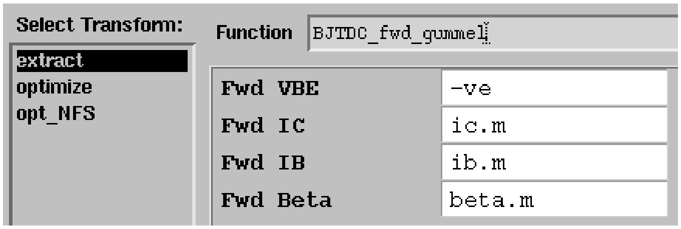
Transform Using the Equation Function The following figure shows an example of a simple transform that divides two data sets. For this transform, the *equation* function is used. With this function, a single line equation can be defined. The input parameters must be IC-CAP recognized data names and not strings in quote marks. In this particular case, the equation is used to calculate *beta* for a bipolar transistor *is: ic / ib*.



Function	equation
Input	ic/ib

Figure 31 Transform Using the Equation Function

Transform Using an Extraction Function A more complicated example is shown in the following figure. This transform calculates the parameters that control the forward mode behavior of the bipolar transistor. The function used is *BJTDC_fwd_gummel* which requires four input data sets. One of these is an IC-CAP input (Fwd VBE), two are outputs (Fwd IC and Fwd IB), and one is another transform (Fwd Beta). In the Fwd VBE field, the expression *-ve* negates the emitter voltage.



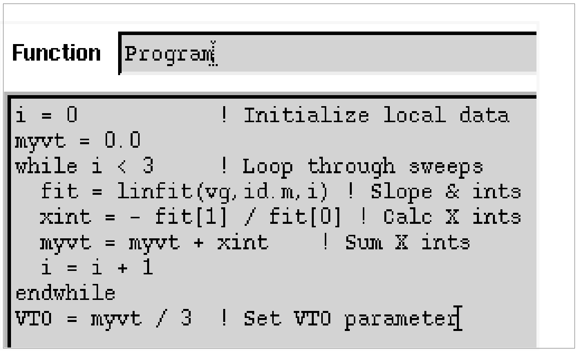
Select Transform:

- extract
- optimize
- opt_NFS

Function	BJTDC_fwd_gummel
Fwd VBE	-ve
Fwd IC	ic.m
Fwd IB	ib.m
Fwd Beta	beta.m

Figure 32 Transform Using an Extraction Function

Transform Using the Program2 or Program Function The Program2 function and Program function (figure below) both provide a BASIC-like programming language. Using this language, you can read from and write to data sets, parameters, and variables, as well as perform many advanced constructs, such as conditional branching and looping. Also, you can execute IC-CAP menu functions and operating system functions within a program. You write the program in a text editor that appears when Program2 or Program is specified in the Function field. The program in the example performs a simple calculation of the VTO parameter for a MOSFET. For detailed information on IC-CAP's programming language, see Chapter 9, "Parameter Extraction Language," in the *Reference* manual.



Function Program

```
i = 0              ! Initialize local data
myvt = 0.0
while i < 3        ! Loop through sweeps
  fit = linfit(vg,id.m,i) ! Slope & ints
  xint = - fit[1] / fit[0] ! Calc X ints
  myvt = myvt + xint     ! Sum X ints
  i = i + 1
endwhile
VTO = myvt / 3   ! Set VTO parameter
```

Figure 33 Transform Using the Program Function

IC-CAP uses a single interpreter to implement the program function, the Macro facility (refer to Chapter 11, "Creating and Running Macros"), and the expression-evaluation capability used in most editor tables. This guarantees a consistent syntax for expressions and program statements throughout the system.

Another way to create new functions is to write them in the C language and add them to IC-CAP. Since doing this involves relinking and restarting IC-CAP, it is slower than using the program functions. However, the advantage is that then the full capabilities of the C language are available. For detailed information on this topic, refer to "Creating C Language Functions in IC-CAP" on page 376.

Performing Transforms

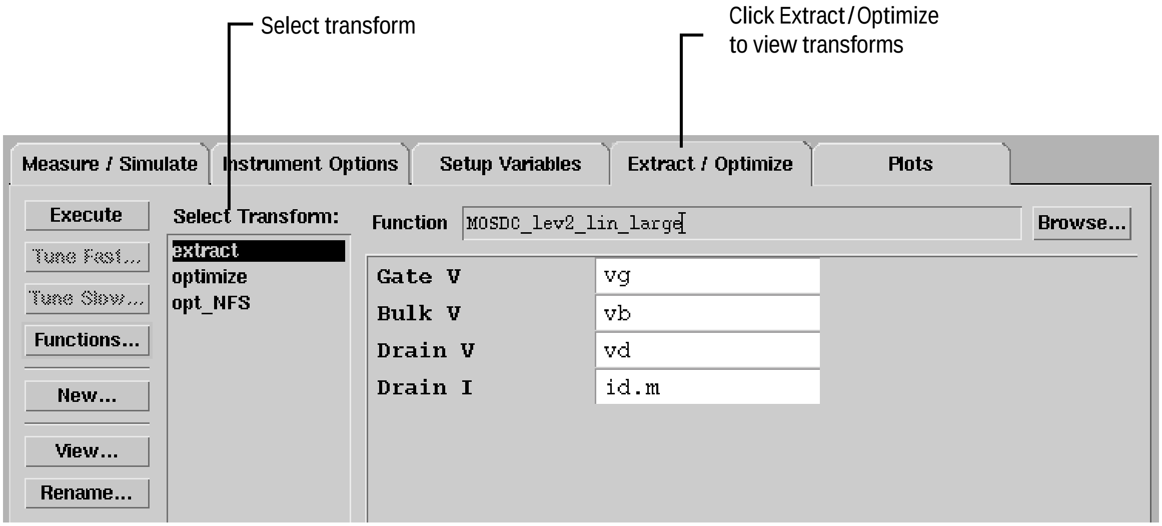
You can perform transformations manually or automatically.

To perform transforms manually:

1 In the Model window, select the setup.

2 Select **Extract / Optimize**.

3 Select the transform you want to perform.



Select transform

Click Extract / Optimize to view transforms

Select Transform:

extract
optimize
opt_NFS

Function MOSDC_lev2_lin_large

Gate V	vg
Bulk V	vb
Drain V	vd
Drain I	id.m

4 Click **Execute**. Execute

Automatic Transform Execution

Transforms are performed automatically when one of the following occurs:

- The transform uses a math function or the program function to calculate a new data set and one of the input data sets changes.
- The transform uses an extraction function or the program function to calculate new parameter values and the *Extract* command is selected for the setup.

- The transform uses the *Optimize* function and the *Optimize* command is selected for the setup.

When using the program function, several statements are also available to directly control these policies.

When Data Set Inputs Change In the *npn/dc/fgummel* setup that contains the *beta* transform, the data sets *ic* and *ib* are updated by the measurement when a measurement is executed. The data sets are modified by a simulation also. When the data sets are updated, the result (ic/ib) stored in beta would be obsolete if beta were not recomputed. IC-CAP recognizes the dependence of beta on these measured data sets and automatically performs the beta transform. Similarly, any plot that uses *ic, ib,* or *beta* is updated. This rule for automatic transform execution applies to all functions designated as *Math Functions*. In addition, a transform using the program function executes automatically when the program uses the *RETURN* statement to generate a data set and the function accesses data sets that change.

When Extraction is Executed on a Setup Issuing the *Extract* command for a setup causes transforms that have been designated as extraction transforms to execute automatically. In addition, a transform using the program function executes automatically if the function assigns new values to model or DUT parameters. In the *npn* model, an example of assigning to a parameter is:

```
BF = max(ic) / max(ib)
```

Since the program assigns new values to parameters, it is considered an *extraction* function. This rule takes precedence over the above rule in which the *RETURN* statement causes automatic execution. A program that assigns new values to parameters and returns a data set does not run automatically when its data set inputs change.

A transform using the *Optimize* function is executed automatically when the Extract command is selected if the *Extract Flag* for that particular *Optimize* function is set to *Yes*. This transform executes after any extraction transforms that are defined for the setup.

When Optimization is Executed on a Setup Issuing the Optimize command for a setup causes transforms that use the *Optimize* function to execute automatically. During an optimization, the Simulated data used in the optimizer is repeatedly updated. When transforms are used to calculate this data, they are automatically performed for each optimizer iteration. For complete information about the operation of the *Optimize* function, refer to Chapter 7, “Optimizing.”

Statements that Force Automatic Execution

Five statements are available to program transforms for providing direct control over the automatic execution of any given program transform. In each program transform, you should not use more than one statement.

UPDATE_EXPLICIT	Suppresses all auto-execution
UPDATE_MANUAL	Suppresses all auto-execution except during an optimization
UPDATE_AUTO	Auto-executes on a data set input change
UPDATE_OPTIMIZE	Auto-executes on Optimize menu function
UPDATE_EXTRACT	Auto-executes on Extract menu function

These statements have no effect in a Macro.

Measured, Simulated, and Common Data

Data sets can contain data associated with measurement and simulation. Since inputs can be freely associated with either, input data is *common.* The program always assumes that sweep limits for measurement and simulation in a setup should be identical in order to make meaningful comparisons. Therefore, an input cannot maintain different stimuli for measurement and simulation. On the other hand, outputs can have an associated type. You must specify these types:

M = Measured Data
S = Simulated Data
B = Both Measured and Simulated Data

Since there may be output quantities that are not measurable but can be simulated, outputs permit this selection to be made.

The types of data sets that receive special handling by transforms are:

- When the input data sets contain both measured and simulated data, the data set created by the transform contains both types of data. The computation is performed once for the measured data and once for the simulated data. For example, in the *beta* transform in the npn model, a plot can refer to the simulated data as *beta.s* or refer to both measured and simulated data as *beta*.
- When the input data sets contain only measured data, the data set created by the transform contains only measured data.
- When the input data sets contain only simulated data, the data set created by the transform contains only simulated data.

Less common types of data sets that receive special handling by transforms are:

- When one, but not all of the data sets, has both types of data, then the transform generates both types. For the *beta* transform, if *ib* has only measured data, then *beta* proceeds as if *ib* contains simulated data identical to its measured data.

- When a mixture of purely measured data and purely simulated data is combined in a transform, the result is common data.
- If a data set with common data is combined with any mixture of data sets that are common, measured only, simulated only, then common data results.

When editing transforms, specify measured or simulated data within the inputs to the transform. For detailed information, refer to “Data Types” in the *Reference* manual.

Using Data from Another Setup

Data Set Size

When creating a transform that references an input, output, or transform from a different setup, make sure all involved data sets are the same size. If they are not the same size, the transform may not work. Generally, the size of a data set is equal to the size of the setup the data set is in. The size of the setup is the product of the number of points shown in all of the input tables in the setup. For example, in a setup with a primary sweep of 20 points and a secondary sweep of 3 points, the size is 60. All of the inputs, outputs, and most of the transforms in this setup have 60 points. Usually, transforms that use functions that compute scalar numbers contain fewer points and transforms that use extraction functions contain no points.

Data Set Name

When defining the path to a data set from another setup, you can use *absolute* addressing or a form of *relative* addressing. For example, in a transform in the rgummel setup of the npn model, address the Input *vb* from the fgummel setup as *fgummel/vb* or */npn/dc/fgummel/vb*. In the relative addressing scheme, when the model, DUT, or setup are not specified, the data set names are assumed to be the same as the setup in which the name is entered. When *fgummel/vb* is used, the model defaults to *npn* and the DUT defaults to *dc*.

Creating C Language Functions in IC-CAP

You can create custom mathematical and other functions in the C language to augment the capabilities of the existing executable code. User-written functions appear in IC-CAP's Function List. The files associated with this customization capability reside in the *$ICCAP_ROOT/src* directory.

You can add functions to a source file named *userc.c,* then re-link and re-run IC-CAP. A *Makefile* and other necessary provisions exist in *$ICCAP_ROOT/src.* Functions written in C must be consistent with the examples in *userc.c.* The file *userc.c* in *$ICCAP_ROOT/*src includes several working examples of C functions added to IC-CAP by users. The examples demonstrate how to receive data sets and other arguments and how to return results.

To add a new function to the IC-CAP system:

1. Change to the *$ICCAP_ROOT/src* directory.
2. Using functions declared in *xf_util.c.h,* add the new function to *add_users_c_funcs()* in *userc.c* with an appropriate call. For example:
   - Use *add_double_c_func1.*
   - Use one of the 3 function calls (*add_input_name, add_variable_name,* and *add_parameter_name)* to establish the labels that appear in the editor of any transform that uses the new function. These functions are declared in *xf_util.c.h* and are described in "Declarations and Implementation Examples" on page 379. (Consult declarations in *userc.h,* where various types are declared.)

To compile the file containing the new function:

1. Create a work directory for the source files (for example, *mkdir my_source*, and change it to (*cd my_source*).
2. Copy the set of source files from *$ICCAP_ROOT/src* to the new work directory (*cp $ICCAP_ROOT/src/* . *).
3. Use the *touch* command on the *.o files so that all *.c and *.o files appear to have been created at the same time

(*touch *.o*). (This step is important for the *make* procedure.)

NOTE

If the drive you're copying to is NFS mounted, clock skews can result if the NFS drive's system has a slightly different system time than the local system. If you think this might apply to you, first, execute *touch* * then execute *touch *.o*. The first *touch* synchronizes all files to your local system's time; the following *touch* causes the *make* system to believe that all of the *.o* files were generated later than the source files, so it will not attempt to rebuild any unnecessary files.

4 Copy your source code to the working directory. Modify the function *add_users_c_funcs()* in *userc.c* to add your C functions to IC-CAP's list of functions, and/or modify the function *add_users_drivers()* in *user_meas.cxx* to add your drivers to IC-CAP's library of instrument drivers. Modify the *Makefile* to add your source code modules to the list of objects.

5 Issue the *make* command and debug any compiler errors.

6 Set the environment variable ICCAP_OPEN_DIR to point to the directory containing the *libicuserc.<ext>* or *libicusercxx.<ext>* file where *ext* is a platform-specific file extension (*ext* is *.so* on Solaris).

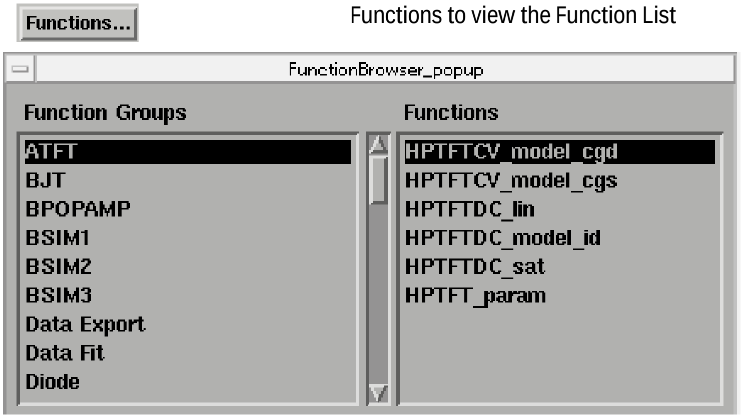
Alternately, if you want to use the new files site wide, you can replace the original files (after copying to another name to preserve them) under $ICCAP_ROOT/lib/<platform>, where <platform> is *sun2x* on SUN Solaris 2.X.

7 Start IC-CAP as usual.

For more information about generating shared libraries for new functions, see "Creating a New Shared Library" in the *Reference*.

To test the function:

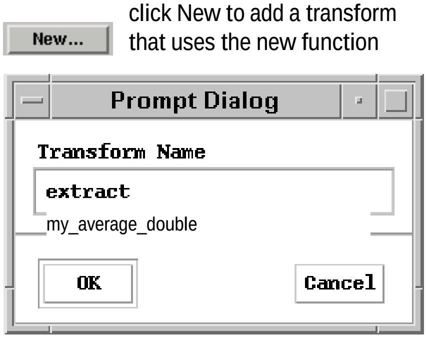
1 Confirm that the new function appears in the Function List.



In the Extract/Optimize window, click Functions to view the Function List

If the new function does not appear in the Function List, check to be sure that an appropriate call was made inside *add_user_c_funcs()*. The system calls *add_user_c_funcs()* once at startup time to add user-written C functions to IC-CAP's Function List.

2 Create a transform that uses the new function.



Transform Name

extract

my_average_double

In the Extract/Optimize window, click New to add a transform that uses the new function

3 If you specified input labels for transforms that use the function, verify that the transform editor associated with the new function displays the expected labels.

Declarations and Implementation Examples

These program files contain a variety of example information:

- *userc.h* provides declarations for important types, such as complex
- *xf_util.c.h* provides declarations for the utility functions that operate on real, complex, and matrix data
- *userc.c* and *userc_io.c* contain examples showing how to write C functions and how to add them to IC-CAP

Function Types

You can add three types of functions to IC-CAP. These functions are designed to operate on Real, Complex, and Matrix data. Although you must choose one of these types, your function does not need to perform mathematical operations. For example, you can implement functions that perform instrument I/O operation.

The function *average_double()* in *userc.c* is an example of a function that operates on real data. This function is defined in Figure 34 where the function arguments are:

inputs A 2-dimensional array of double precision real numbers that contains zero or more input data sets

num_inputs An integer indicating the number of input data sets contained in inputs

points_per_input An array of integers indicating the number of data points in each data set in inputs

output An array of double precision real numbers used to store the results of the function.

num_out_points An integer indicating the number of data points in output

variables An array of double precision real numbers containing zero or more variables passed to the function

num_variables An integer indicating the number of values contained in variables *parameters* An array of strings (char*) containing zero or more parameter names or other strings passed to the function

num_parameters An integer indicating the number of strings contained in parameters

points_per_curve An integer indicating the number of points in the main sweep of the setup which owns the input data sets.

The function *cplx_conj()* in *userc.c* is an example of a function that operates on complex data. Its arguments are identical to *average_double()* except that the *inputs, output,* and *variables* arguments contain complex numbers instead of real numbers.

The function *matrix_transpose()* in *userc.c* is an example of a function that operates on matrix data. Its arguments are identical to *cplx_conj()* except that the *inputs* and *output* arguments contain matrices instead of complex numbers. Refer to the *userc.h* and *userc.c* files for information on manipulating complex numbers and matrices.

```
/*********************************************************
** average_double ()
** return average of data sets of real data
*********************************************************/
 static int average_double (inputs, num_inputs, points_per_input, output,
  num_out_points, variables, num_variables, parameters, num_parameters,
  points_per_curve )

  double_2d_array inputs;
  int num_inputs;
  int_array points_per_input;
  double_array output;
  int num_out_points;
  double_array variables;
  int num_variables;
  string_array parameters;
  int num_parameters;
  int points_per_curve;
{
  int dataset_num ;
  int point_num ;
  double sum ;
  int dataset_size = points_per_input[0] ;
  /* see if at least one input received */
  if ( num_inputs < 1 ) {
  standard_fail_msg ( "no input dataset(s) received" ) ;
  return -1 ;    /* indicates failure to system */
}
/* see if input datasets are all of same size */
for ( dataset_num = 0; dataset_num < num_inputs ; dataset_num++ ){
  if ( points_per_input[dataset_num] != dataset_size ) {
    standard_fail_msg ( "input datasets have different sizes" ) ;
    return -1 ;    /* indicates failure to system */
  }
}
/* see if output dataset has same size as inputs */
if ( num_out_points != dataset_size ) {
  standard_fail_msg ( "output dataset has different size than inputs" ) ;
  return -1 ;    /* indicates failure to system */
}
/* do the averaging */
for ( point_num = 0; point_num < dataset_size ; point_num++ ) {
  sum = 0 ;
  for ( dataset_num = 0; dataset_num < num_inputs ; dataset_num++ ) {
    sum += inputs [ dataset_num ] [ point_num ] ;
  }
  output [ point_num ] = sum / num_inputs ;
}
return 0 ; /* indicates success */
}
```

Figure 34 The *average_double()* Function Operates on Real Data

Adding Functions to the Function Browser

This section describes six functions for adding the new function to the Function Browser and provides declarations for them. These IC-CAP functions (Table 29) make a pointer to the new function, as well as counts indicating how many instances of each possible input type the function expects to receive. For example, an extraction function may need to receive *two* data set inputs, *zero* real numbers, and *four* model parameters. The system stores the address of the function, and uses the counts to construct editors at run-time, when the function is selected for use by a transform. You can elect to receive data sets in the form of real arrays, complex arrays, or matrix arrays. You can also choose to run the function automatically when its inputs change or to run it when the *Extract* command is executed for a setup.

NOTE

For the following functions, you must use the *funcptr* argument as described; otherwise, a core dump or other erratic behavior may occur when IC-CAP calls the new function.

The arguments for these functions are defined as:

name A string (char*) containing the name for the new function as it should appear in the IC-CAP Function Browser

funcptr A function pointer that is the actual name of the new function as defined in *userc.c*

ni An integer indicating the number of input data sets required by the new function

op An integer indicating the number of points in the output data set created by the new function. This is often set to -1, which indicates that the number of points returned by the function will be the same as the number of points in the Outputs within the current setup. A value of -2 indicates that the number of points returned by the function will be the same as the number of points in the largest data set passed to the function.

nvr An integer indicating the number of variables (real/complex number arguments) required by the new function

npr An integer indicating the number of parameter names (or other strings) required by the new function

xfft An enumerated type used to set the type of the new function. A function can be an extraction, a math function, or a special function. An extraction is executed when you issue the Extract command for the Active setup or Active DUT and a math function is automatically executed if any of its inputs change. A special function must always be executed manually. Refer to the description of the xffunc_type *enum* in xf_util.c.h for the available choices. The functions, add_double_c_func1, add_complex_c_func1, or add_matrix_c_func1 do not contain this argument. If they are used to add the new function to the Function Browser, the function is assumed to be a math function.

bdf This argument is reserved for future use. Always set it to 0.

After a function is installed in the Function Browser, a series of calls should be made immediately to provide labels for the data set expressions, numeric expressions, and parameter inputs that appear in the editor of any transform using the function. For information about providing labels, refer to “Labeling the Editor Input Fields” on page 388.

Table 29 Functions for Adding a New Function to the Function Browser

Function	Definition
add_double_c_func1	Installs a user-defined function in the Function Browser. *Funcptr* should be the name of a C function taking an argument list identical to *average_double()*. Refer to *userc.c*, for examples of this function call. Syntax: `void add_double_c_func1 ( /* char* name, FUNCPTR_TYPE funcptr, int ni,int op, int nvr, int npr */ ) ;`
add_double_c_func2	Installs a user-defined function in the Function Browser. *Funcptr* should be as in *add_double_c_func1*. The type of the new function can be set with the *xfft* argument. The *bdf* argument is reserved for future use. Always set it to 0. Syntax: `void add_double_c_func2 ( /* char* name, FUNCPTR_TYPE funcptr, int ni, int op, int nvr, int npr, xffunc_type xfft, int bdf */ ) ;`
add_complex_c_func1	Installs a user-defined function in the Function Browser. *Funcptr* should be the name of a C function taking an argument list identical to *cplx_conj()*. Refer to *userc.c* for examples of this function call. Syntax: `void add_complex_c_func1 ( /* char* name, FUNCPTR_TYPE funcptr, int ni, int op, int nvr, int npr */ ) ;`
add_complex_c_func2	Installs a user-defined function in the Function Browser. *Funcptr* should be as in *add_complex_c_func1*. The type of the new function can be set with the *xfft* argument. The *bdf* argument is reserved for future use. Always set it to 0. Syntax: `void add_complex_c_func2 ( /* char* name, FUNCPTR TYPE funcptr, int ni, int op, int nvr, int npr, xffunc type xfft, int bdf */ ) ;`

Table 29 Functions for Adding a New Function to the Function Browser

Function	Definition
add_matrix_c_func1	Installs a user-defined function in the Function Browser. Data sets consisting of complex matrices (s-parameter data) are supplied when this function is called. Declarations for the matrices can be found in *userc.h*. *Funcptr* should be the name of a C function taking an argument list identical to *matrix_transpose()*. Refer to userc.c for an example of this function call. Syntax: `void add_matrix_c_func1 ( /* char* name, FUNCPTR_TYPE funcptr, int ni, int op, int nvr, int npr */ ) ;`
add_matrix_c_func2	Installs a user-defined function in the Function Browser. *Funcptr* should be as in *add_matrix_c_func1***.** The type of the new function can be set with the *xfft* argument. The *bdf* argument is reserved for future use. Always set it to 0. Syntax: `void add_matrix_c_func2 ( /* char* name, FUNCPTR TYPE funcptr, int ni, int op, int nvr, int npr, xffunc type xfft, int bdf */ ) ;`

Argument List Guidelines

The argument lists for new functions must agree with IC-CAP's requirements. For example, *add_complex_c_func1()* tells the system that a function such as *cplx_conj* expects complex data set inputs, output array, and numeric variables. This is also the case for *cplx_conj()*.

As another example, *add_double_c_func1()* is used to tie in *average_double()*, which expects double data set inputs, output array, and numeric variables. New C functions, like *cplx_conj()* and *average_double()*, must be installed using the appropriate call, so that IC-CAP can invoke the function. Otherwise, core dumps can result.

Adding Function Descriptions to the Function Browser

By default, when you create a new function in IC-CAP, the function appears in the User Defined category in the Function Browser dialog box. You can create dialog box documentation for the function and/or place it in another category. The process consists of two basic steps:

- Create a text file with the documentation
- Declare a variable to identify the location and name of the text file

To document and categorize user-defined functions:

1 Create a text file using any text editor.

2 Create a record for each function using the following form:

<*function_name*>
"<*group1*>"[,"<*group2*>"]...
<*multiline description*>
–
<*multiline list of extracted parameters*>

where

<*function_name*> appears exactly (spelling/case) as you created it in IC-CAP

<“*group1*”> is the label of the category (new or existing) you want the function to be listed under in the Function Browser. Note that additional groups are optional. If you want a function to appear in more than one category, list each category name, separated by commas.

<*multiline description*> is the text you want displayed in the Description field

– indicates the end of text for the Description field and beginning of text for the Extracted Parameters field

<*multiline list of extracted parameters*> is the list of parameters you want displayed in the Extracted Parameters field (where applicable)

When creating each record observe the following guidelines:

- Use an exclamation mark (!) as a comment symbol in column 1
- Use a single asterisk (*) on a line to indicate a record separator (no spaces allowed)
- Use a single hyphen (-) on a line to indicate the separation between text for the Description field and text for the Extracted Parameters field (no spaces allowed)

NOTE

Multiple functions can be declared in a single file or in separate files.

To declare the variable identifying your function documentation:

1 Create the variable *ICCAP_USER_FUNCTION_BROWSE_DATA* in any of the following files:

   - *$ICCAP_ROOT/iccap.cfg* (for site-wide use)
   - *$HOME/.iccap* (for individual use, all sessions)
   - *.iccap* in any startup directory (for individual use, particular session)

   The program searches the startup directory first, then *$HOME*, then the installation directory. For information on changing these files, refer to the *Installation* manual.

2 Set the variable to the path and filename(s) of your function documentation file(s); for example:

```
ICCAP_USER_FUNCTION_BROWSE_DATA = {$HOME}/my_func.txt
```

   When listing multiple files, separate filenames by colons.

3 Re-start IC-CAP. The program reads the listed files, in addition to the supplied function file, and display all functions in the Function Browser.

Labeling the Editor Input Fields

This section describes the functions used for labeling the fields appearing in transform editors. Also, custom functions and transforms may show appropriate labels to remind you of the types of arguments needed at run-time.

The following definitions are used in this section:

- An input, or data set input, denotes a data set or an expression involving datasets.
- A numeric variable is a field in which a user can provide a real number or expression.
- A parameter is a field in which a user can provide the name of a model or DUT parameter, an arbitrary string expression (as is the case with the USERC_system function), or the name of an IC-CAP system variable.

Table 30 Labeling the Fields Functions

Function	Definition
add_input_name	Specifies a label appearing for a data set input field when a user-defined function is used in a transform. Refer to *userc.c* for examples of this call. This should be called once for each data set input required. The number of data set inputs required is specified as *ni* in one of the calls described in "Adding Functions to the Function Browser" on page 382. Limit the length of the *name* argument to 12 characters; otherwise it may be truncated when displayed by IC-CAP. Syntax: `void add_input_name ( /* char* name */ ) ;`
add_variable_name	Specifies a label appearing for a numeric variable field when a user-defined function is used in a transform. Refer to *userc.c* for examples of this call. This should be called once for each numeric variable required. The number of numeric variables required is specified as *nvr* in one of the calls described in "Adding Functions to the Function Browser" on page 382. Limit the length of the *name* argument to 12 characters; otherwise it may be truncated when displayed by IC-CAP. Syntax: `void add_variable_name (/* char* name */ );`

Table 30 Labeling the Fields Functions (continued)

Function	Definition
add_parameter_name	Specifies a label appearing for a parameter field when a user-defined function is used in a transform. Refer to *userc.c* for examples of this call. This should be called once for each parameter required. The number of parameters required is specified as *npr* in one of the calls described in “Adding Functions to the Function Browser” on page 382. Limit the length of the *name* argument to 12 characters; otherwise it may be truncated when displayed by IC-CAP. Syntax: `void add_parameter_name (/* char* name */ );`

Utility Functions

IC-CAP provides a set of utility functions to use when writing transforms in C language. Most of these functions are declared in *xf_util.c.h*. Some functions, for which the source is provided, appear only in *userc.c* or *userc.h*. Each is described in the following tables.

Table 31 Accessing Model and DUT Parameters Functions

Function	Definition
get_par_w_check	Finds a parameter’s value, given its name. The value is passed back via the *double** second argument. Return values are: 0 = parameter found –1 = parameter not found Syntax: `extern int get_par_w_check ( /* char *param_name, double *return_data */ );`
set_par_w_check	Sets a parameter’s value. The value is passed in via the *double** second argument. Return values are: 0 = parameter found –1 = parameter not found Syntax:

Table 31 Accessing Model and DUT Parameters Functions (continued)

Function	Definition
	`extern int set_par_w_check ( /* char *param_name, double val */ );`
set_par_w_check_ptr	Sets a parameter's value. The value is passed in via the *double** second argument. This is more suitable for FORTRAN calling than C, since the second argument is by address. Return values are: 0 = parameter found –1 = parameter not found Syntax: `extern int set_par_w_check_ptr ( /* char *param_name, double *val */ );`

Table 32 Accessing System Variables Functions

Function	Definition
get_var_w_check	Finds a system variable's value, given its name. The value is passed back via the *double** second argument. Return values are: 0 = system variable found –1 = system variable not found Syntax: `extern int get_var_w_check ( /* char *variable_name, char *return_data */ );`
get_var_str_w_check	Finds a system variable's string value, given its name. The value is returned *by reference* in the second argument. A static area is used for the returned string, so copy the returned string if you want to save the value and intend to call this function again. Return values are: 0 = system variable found –1 = system variable not found Syntax: `extern int get_var_str_w_check ( /* char *variable_name, char **return_data */ );`

Table 32 Accessing System Variables Functions (continued)

Function	Definition
set_var_w_check	Sets a system variable’s value. The value is passed in via the *double* second argument. Return values are: 0 = system variable found –1 = system variable not found Syntax: `extern int set_var_w_check ( /* char *variable_name, double val */ );`
set_var_str_w_check	Sets a system variable’s value. The value is passed in via the *char** second argument. This differs from the previous call in that the IC-CAP system variable is assigned a string rather than a numeric value. Return values are: 0 = system variable found –1 = system variable not found Syntax:
	`extern int set_var_str_w_check ( /* char *variable_name, char* val */ );`

The following utilities compute values for parameters or system variables (which the above two sections address separately). These functions also generate output automatically in the same style as the built-in IC-CAP extraction functions. Source for these functions appears in *userc.c*

Table 33 Accessing Parameters or System Variables Function

Function	Definition
get_par_or_var	Finds a parameter’s or a system variable’s value, given its name. In a conflict, preference is given to the parameter over the system variable. The value is passed back via the *double** second argument. Return values are: 0 = parameter or system variable found –1 = parameter or system variable not found Syntax: `extern int get_par_or_var ( /* char *parameter_name, double *return_data */);`

Table 33 Accessing Parameters or System Variables Function

Function	Definition
set_par_or_var	Sets a parameter's or a system variable's value. In a conflict, preference is given to the parameter over the system variable. The value is passed in via the double second argument. Return values are: 0 = parameter or system variable found –1 = parameter or system variable not found Syntax: `extern int set_par_or_var ( /* char *parameter_name, double val */ ) ;`

Routines are provided to read values from IC-CAP data sets and to obtain information about the sweeps present in the setup. You can write new data values onto existing outputs and transforms.

Table 34 Accessing Parameters or System Variables Functions

Function	Definition
get_data_w_check	Gets the value of a single point in an IC-CAP data set (an Input, Output, or transform). A complex number will be fetched from the IC-CAP data set and placed in data_fetched. The data set name is the first argument. The second argument should specify 'M' (measured) data, or 'S' (simulated). The row and col arguments should be 1,1 unless dealing with 2-port data. The index argument can range from 0 to (size of dataset -1). The final argument is the address of a 'complex' struct (see userc.h). An example is provided in *xf_util.c.h*. Return values are: 0 = dataset is found, and msb, row, col, and index are all in range –1 = any error condition Syntax: `extern int get_data_w_check ( /* char* dsname, char msb, int row, int col, int index, complex* data_fetched */ ) ;`

Table 34 Accessing Parameters or System Variables Functions

Function	Definition
set_data_w_check	Sets the value of a single point in an IC-CAP Output or transform. Setting values in an Input is not supported, because IC-CAP computes and overwrites these automatically. The data set name is the first argument. The second argument should specify 'M' (measured) data, or 'S' (simulated). When 'B' (both) is specified, the complex number will be stored in both the Measured and Simulated parts of the data set. The row and col arguments should be 1,1 unless dealing with two-port data. The index argument can range from 0 to (size of dataset -1). The final argument is the address of a 'complex' struct (see userc.h). An example is provided in *xf_util.c.h*. Return values are: 0 = dataset is found, and msb, row, col, and index are all in range –1 = any error condition Syntax: `extern int set_data_w_check ( /* char* dsname, char msb, int row, int col, int index, complex* data_poked */ ) ;`

Advanced functions are available for querying about the inputs used in a setup. Consult */usr/iccap/src/xf_util.c.h* for usage and examples of the following functions.

```
int get_num_of_points ( /* char* setup_path, int swp_order */) ;
double get_sweep_start ( /* char* setup_path, int swp_order */) ;
double get_sweep_stop ( /* char* setup_path, int swp_order */) ;
double get_sweep_stepsize (/* char* setup_path,int swp_order */);
int get_num_of_curves ( /* char* setup_path */) ;
int get_points_per_curve ( /* char* setup_path */) ;
int get_max_sweep_order ( /* char* setup_path */) ;
int get_sweep_type ( /* char* setup_path, int sweep_order */) ;
int get_sweep_mode ( /* char* setup_path, int sweep_order */) ;
int get_log_sweep_dec_or_oct ( /* char* setup_path, int swp_order
*/) ;
```

Table 35 Using a Dialog Box for User Input Functions

Function	Definition
get_number	Uses a dialog box to query the user for a number. It returns -I if the user chooses *CANCEL,* otherwise it returns 0. The desired value is returned *by reference* in *return_data*. `int get_number ( /* char* prompt , double default_value, double* return_data*/ ) ;`
get_string	Uses a dialog box to query the user for a string. It returns -1 if the user chooses *CANCEL*, otherwise it returns 0. The desired value is returned *by reference* in *return_data*. `int get_string ( /* char* prompt, char* default_value, char** return_data */) ;`

NOTE

A static area is used for the returned string. You should copy the returned string, or use it before calling this function again.

Table 36 Posting to the Error Box Functions

Function	Definition
error_message	Sends an error message to the system error box. The message should have "\n" at its end for multiple messages. `void error_message ( /* char* msg */ ) ;`
standard_fail_msg	Sends object name, function name, and *msg* to errbox in a standard format. Refer to *userc.c*, for the source to this function. The message should have "\n" at its end for multiple messages. `static void standard_fail_msg ( /* char* msg */ );`
object_name	Gets the name of the transform or Macro under which this function is running.
	`extern char* object_name ();`
function_name	Gets the name of the present user-defined function, as it appears in the Function Browser.

Table 36 Posting to the Error Box Functions (continued)

Function	Definition
	`extern char* function_name () ;`

Table 37 Calling Menu Functions

menu_func	Permits your C functions to invoke IC-CAP menu selections. It behaves like the *menu_func* statement in PEL. However, this function accepts only 2 arguments, and does not provide the ability for *Anticipating Dialog Boxes* that are provided by the *menu_func* call in PEL**.** It returns -1 if the IC-CAP object does not exist or does not support the selected menu function name. Otherwise it returns 0, including when the menu function is found and executed but produces errors. For additional information on calling menu functions, refer to Chapter 11, “Creating and Running Macros.” `int menu_func ( /* char* obj_name, char* func_name */ ) ;`

Handling Signals and Exceptions

Two important signals generated within IC-CAP under certain conditions are: SIGFPE and SIGINT.

SIGFPE This signal occurs when the code executes an operation like a divide by zero. By default, there is no provision in IC-CAP for trapping this signal. If this signal occurs during the execution of a transform, the function or macro will continue to execute and upon completion, an error message is displayed indicating a floating point error occurred.

SIGINT This signal is generated when you issue the Interrupt command. By default, there is no provision in IC-CAP for trapping this signal. The function or macro is interrupted immediately. Note: For complex operations, it may take several minutes before control is returned.

These signals, and others, are declared in *$ICCAP_ROOT/src/sighandler.hxx*. They are explained in UNIX manual pages, including *signal(2)* and *sigvector(2)*. If your application requires special error recovery for the SIGFPE or SIGINT signals, it is possible to modify the signal handling to trap them.

Modifying Signal Handling Behavior

To modify the handling of either of these signals (or any others), you must comply with certain IC-CAP programming conventions; failure to do so could cause the program to terminate.

If a user C function modifies signal handling during its execution, it must also restore the prior state of signal handling before it returns. The example demonstrates how this might be accomplished.

If you modify the signal handling for SIGFPE, you should also modify it for SIGINT. This is based on the need to restore the default SIGFPE handler before returning from the User C code. If you modify it for SIGFPE and don't modify it for SIGINT, an Interrupt could force a premature return from User C code, resulting in a failure to restore IC-CAP's signal handler for floating point errors. The example in Figure 35 satisfies this need by causing SIGINT to be ignored while the SIGFPE handling is changed and then restored to IC-CAP's default. Do not alter the handling of SIGUSR1 and SIGUSR2; both signals are used internally by IC-CAP for error trap and recovery purposes.

NOTE

IC-CAP employs the *sigvector* signal handling facility (see the UNIX manual page *sigvector(2)*). Any User C code that modifies signal handling should employ the same facility. That is, the system calls described under *signal(2)* and *sigset(2V)* should not be used within IC-CAP.

```
#include <stdio.h>
#include <sighandler.hxx>

struct sigvec newINTvec, oldINTvec;
struct sigvec newFPEvec, oldFPEvec;

/* arrange to ignore SIGINT signal */
newINTvec.sv_mask = 0 ;
newINTvec.sv_flags = 0 ;
newINTvec.sv_handler = SIG_IGN;

/* oldINTvec receives 'backup' of IC-CAP sigvector for
SIGINT */
sigvector(SIGINT, &newINTvec, &oldINTvec);

/* now you can do what you want, without fear of interruption
*/

/* this arranges to ignore SIGFPE signal, e.g. */
newFPEvec.sv_mask = 0 ;
newFPEvec.sv_flags = 0 ;
newFPEvec.sv_handler = SIG_IGN;

/* oldFPEvec receives 'backup' of IC-CAP sigvector for
SIGFPE */
sigvector(SIGFPE, &newFPEvec, &oldFPEvec);

/* do any math, with floating point errors ignored ... */

/* Restore IC-CAP handler for SIGFPE! */
/* Should be done while SIGINT is still being ignored */
sigvector(SIGFPE, &oldFPEvec, (struct sigvec *) NULL);

/* Restore IC-CAP handler for SIGINT! */
sigvector(SIGINT, &oldINTvec, (struct sigvec *) NULL);
```

Figure 35 Signal Handling Example

The following functions are provided for signal handling:

icecap_ignore_sigfpe() Allows execution to continue without an error when a floating point error occurs in User C code. Save the return value for use with the icecap_reenable_sigfpe function to reenable normal signal handling—stop execution and present an error message when a floating point error occurs in User C code.

Example:

```
userc_function(....)
{
   int oldFPEval;
   .
   .
   oldFPEval=icecap_ignore_sigfpe();
   /* Following code will ignore floating point errors */
   x=x/0;
   .
   .
   icecap_reenable_sigfpe(oldFPEval);
   /* following code will again trigger FPE errors */
.
.
}
```

icecap_ignore_sigint() Allows execution to continue without an error message when the stop sign is pressed. Save the return value for use with the icecap_reenable_sigint function to reenable normal signal handling—stop execution and present an error message when the stop sign is pressed. See the example for "icecap_ignore_sigfpe()."

icecap_reenable_sigfpe(oldfpeVal) Reenables signal handling on exit from your User C code for floating point errors. Use the return value from icecap_ignore_sigfpe to reenable the normal signal handling for a floating point error—stop execution and present an error message. See the example for "icecap_ignore_sigfpe()."

icecap_reenable_sigint(oldintVal) Reenables signal handling on exit from your User C code when the stop sign is pressed. Use the return value from icecap_ignore_sigint to reenable the normal signal handling that occurs when the stop sign is pressed—stop execution and present an error message. See the example for "icecap_ignore_sigfpe()."

Agilent 85190A IC-CAP 2008
User's Guide

10
Printing and Plotting

This chapter contains procedures for definition and creation of graphic displays and textual reports and describes printing and plotting from IC-CAP.

You can send output to printers and plotters as well as to a file in a variety of formats. When printing to a file, the format of the file is determined by the current output device and the file is saved in the current project directory.

You can connect any output device that is supported by your operating system. To connect additional printers and plotters, and select a default printer, choose the appropriate method for your platform:

- WindowsXP—Start menu > Settings > Printers
- UNIX—Choose File > Print Setup

The basic Print commands are summarized next:

- Use the Print command to print the contents of the current window
- Use the Print Setup command to establish a default printing configuration, although you can modify it at the time of printing

Creating an IC-CAP Plot

This section contains procedures for definition and creation of graphic displays and textual reports. The MOSFET model supplied with IC-CAP, *nmos2*, contains a variety of plotted data sets and is used as the example in this chapter.

A plot is a graphical representation of data that gives a quick view of measured and simulated data, making it easier to spot any differences between the two. Using IC-CAP's plot analysis features, you can identify a data point or slope of a curve easily.

If you need to get raw data in the form of numbers rather than a graph, you can view such data in a list form that can be sent to a printer.

The general procedure for creating a plot is:

- Open the setup and add a plot.
- Assign a data set to a graph axis.
- Specify a report type and an axis type.
- Put in a header and footer if necessary.
- Open a plot window.
- Rescale the plot if necessary.
- Analyze the plot.
- Print a hardcopy of the plot.

NOTE

The term *data set* means either Input, Output, or Transform, as used in Chapter 11, “Creating and Running Macros.”

Because the print methods vary significantly between UNIX and the PC, this chapter describes printing from these platforms separately. See the appropriate section for your platform:

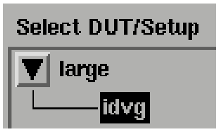
- “Printing from UNIX” on page 443
- “Printing from the PC” on page 457

Adding a New Plot

The first step in creating a plot is to add a plot.

To create a new plot:

1 In the DUTs-Setups folder, select the setup.

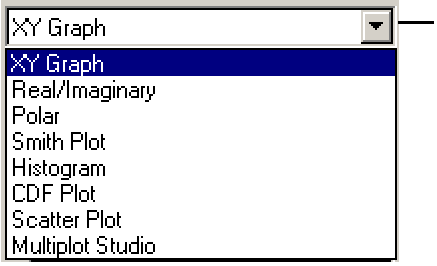


2 Then select **Plots**.

3 Click **New**.

4 In the Plot Editor, enter a name for the new plot.

5 Select one of the Report Types from the drop-down list.



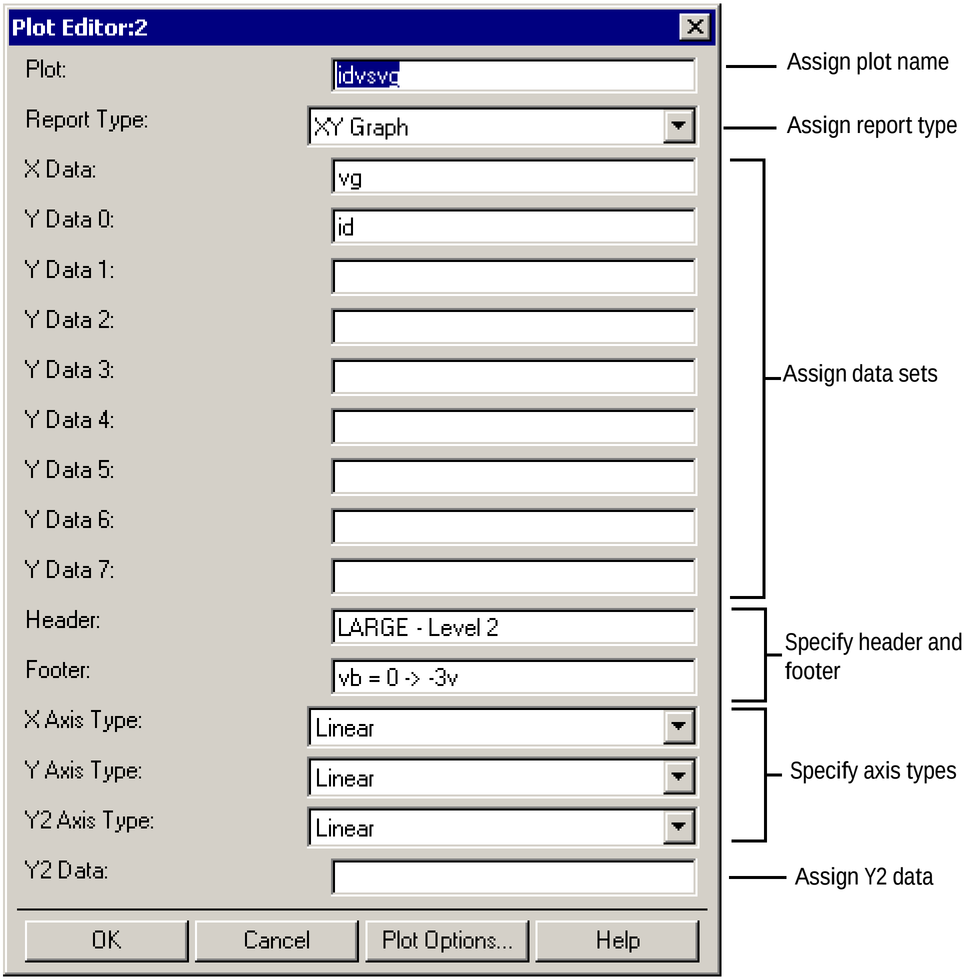
Click arrow for list of types

6 The required fields change to reflect the selected report type. Fill in the necessary fields for the selected report type. For details, see "Defining a Plot for Specific Data" on page 403 and "Defining Axis Types and Data Sets" on page 413.

NOTE

You can specify an alternate name for a plot legend by adding two exclamation marks (*!!*) followed by the alternate name after the real data item. For example, the following statements label the X data as *vin* and the Y data as *calc_a*:

X Data : /a/really/really_long_path/vin !! vin
Y Data 0 : (a + complicated)*expression !! calc_a



Plot Editor:2

Plot: idvsvg

Report Type: XY Graph

X Data: vg

Y Data 0: id

Y Data 1:

Y Data 2:

Y Data 3:

Y Data 4:

Y Data 5:

Y Data 6:

Y Data 7:

Header: LARGE - Level 2

Footer: vb = 0 -> -3v

X Axis Type: Linear

Y Axis Type: Linear

Y2 Axis Type: Linear

Y2 Data:

Assign plot name

Assign report type

Assign data sets

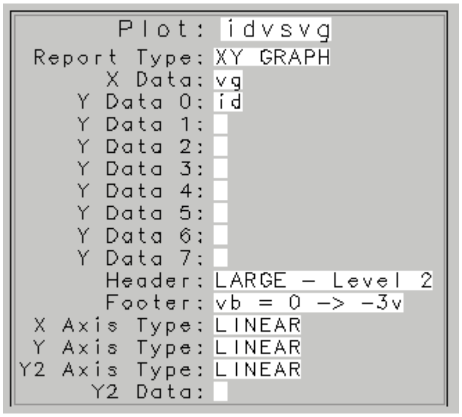
Specify header and footer

Specify axis types

Assign Y2 data

7 Click **Plot Options** to define plot options. See “Setting Plot Options” on page 431.

8 Click **OK**. The new plot is added to the Plots folder.



```
         Plot: idvsvg
  Report Type: XY GRAPH
       X Data: vg
     Y Data 0: id
     Y Data 1:
     Y Data 2:
     Y Data 3:
     Y Data 4:
     Y Data 5:
     Y Data 6:
     Y Data 7:
       Header: LARGE - Level 2
       Footer: vb = 0 -> -3v
  X Axis Type: LINEAR
  Y Axis Type: LINEAR
 Y2 Axis Type: LINEAR
      Y2 Data:
```

Defining a Plot for Specific Data

The following 8 tables list definitions for plot data types. Certain plot characteristics can be set with system variables. When a variable applies to a specific plot type, the variable is listed in the table for that type. Variables that apply to any plot are listed in Table 46.

Table 38 Defining a Plot for XY Data

Data Type	Definition
X Data:	The Input name
Y Data 0:	The Output name (or expression that is a function of the outputs)
Y Data 1-7:	Names of additional outputs sharing the same scale as Y0
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph
X Axis Type:	X Axis Type (Linear, Log 10, and dB) †
Y Axis Type:	Y Axis Type (Linear, Log 10, and dB) †
Y2 Axis Type:	Additional Y axis type used for showing another data set with a different magnitude or domain. For example, use the Y axis for gain and the Y2 axis for phase.

Table 38 Defining a Plot for XY Data (continued)

Data Type	Definition
Y2 Data:	The name of a data set to display with a different scale.

† dB axis type represents 20 × log10(x), not 10 × log10(x).

Table 39 Defining a Plot for Real / Imaginary Data

Data Type	Definition
Sweep Data:	The Input data set name
RI Data 0:	The Output data set name or expression that is a function of the outputs
RI Data 1-7:	Additional data sets or expressions
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph

Table 40 Defining a Polar Chart

Data Type	Definition
Sweep Data:	The Input data set name
Polar Data 0:	The Output data set name or expression that is a function of the outputs
Polar Data 1-7:	Additional data sets or expressions
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph

Table 41 Defining a Smith Plot

Data Type	Definition
Sweep Data:	The Input data set name
Smith Data 0:	The Output data set name or expression that is a function of the outputs

Table 41 Defining a Smith Plot

Data Type	Definition
Smith Data 1-7:	Additional data sets or expressions
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph

A histogram depicts either the frequency or the relative frequency of data distribution. Typically a histogram is used to identify both the ranges around which most of the data converges and the outliers of the data set. In a histogram, the X-axis displays the value ranges while the Y-axis plots the frequency or relative frequency for each range. A histogram is used for the visualization of data for a single variable.

Table 42 Defining a Histogram

Data Type	Definition
Data-set:	The Input data set name
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph
Variables for setting Plot Characteristics	
HISTOGRAM_NUM_BINS	Number of bins in the histogram plot. Default is 10.

Table 42 Defining a Histogram (continued)

HISTOGRAM_NORMALIZATION	Can be set to either TRUE or FALSE. Setting it to TRUE will normalize the histogram. Default is TRUE. NOTE: definition of the histogram normalization: Assume Variable *X* indicated the width of the histogram bin, Variable *n* indicated the number of the histogram bins. Variable Y_i indicated the number of samples for the histogram bin with index of *i* Variable $normalize\_Y_i$ indicated the normalized number of samples for the histogram bin with index of *i* Then we have, $normalize\_Yi = (Yi)/\left(\sum_{i-1}^{n}(X \times Y_i)\right)$
HISTOGRAM_GAUSSIAN_FIT	Can be set to either TRUE or FALSE. Setting it to TRUE will draw the Gaussian curve fitted to the histogram. Default is TRUE.

A cumulative density plot depicts either the cumulative frequency or the cumulative relative frequency of data distribution. Typically a cumulative density plot is used to track variations between contemporaneous observations by highlighting changes in mean levels. In a cumulative density plot the X-axis displays the value ranges while the Y-axis plots the cumulative frequency or cumulative relative frequency for each range.

Table 43 Defining a CDF Plot

Data Type	Definition
Data-set:	The name of an Input data set

Table 43 Defining a CDF Plot

Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph
Variable for setting Plot Characteristics	
CDF_ERROR_FIT	Can be set to TRUE or FALSE. TRUE will draw the Gaussian curve fitted to the Cumulative plot. Default is TRUE.

A scatter plot depicts the distribution of pairs of values on a rectangular, 2-dimensional plane. A scatter plot is the most effective method of analyzing the correlation between two variables. Values for one variable are tracked on the X-axis while values for the other variable are tracked on the Y-axis.

Table 44 Defining a Scatter Plot

Data Type	Definition
X Data:	The X-axis parameter name
Y Data:	The Y-axis parameter name
Header:	Annotation that appears above the graph
Footer:	Annotation that appears below the graph
X Axis Type:	X Axis Type (Linear, Log 10, and DB)
Y Axis Type:	Y Axis Type (Linear, Log 10, and DB)
Variable for setting Plot Characteristics	
SCATTER_CONTOURS	Can be set to either TRUE or FALSE. Setting it to TRUE will draw the contours. Default is TRUE.
SCATTER_NUM_SEGMENTS	Number of segments used in drawing the contours of the plot. Default is 1500. Using a larger number results in an accurate contour, but takes more time.

Table 45 Defining a Multiplot Studio

Variable	Definition
# of Plots	Number of plots displayed in the Multiplot. The number can be from 1 to 100. The default is 4.
Plot 1 - 100:	Name of each plot.
# of GUIs:	Number of GUI regions. The number can be 0 to 8. The default is 0.
GUI 1 - 8:	Name of each GUI region.
GUI 1- 8 Location:	Location on the Multiplot display for each GUI region. The location can be upper left (UL), upper right (UR), upper (U), left (L), right (R), bottom left (BL), bottom right (BR), or bottom (B).
Orientation:	Orientation of the Multiplot. The orientation can be vertical (V) or horizontal (H). The default is horizontal.
Plots Per Row:	Number of plots in each row. If the number is 0, the plots will be arranged so that the number of rows and columns are equal if possible (squared layout). The default is 0.

Table 45 Defining a Multiplot Studio (continued)

Variable	Definition
Variable for setting Plot Characteristics	
DYNAMIC_MULTIPLOT_MODE	Sets the location where the manual scaling setting is saved for a Multiplot. If set to FALSE, the manual scaling setting for each subplot in a Multiplot is saved with the Multiplot, allowing different scaling to be saved for the same plot if opened as a single plot or on one or more Multiplot. If set to TRUE, the manual scaling setting for each subplot in a Mulitplot is saved with the subplot. In this mode, no matter where the plot is opened (as a single plot or on one or more Multiplots), it will use the same scaling information. However, if the same subplot appears on multiple Multiplots, or in multiple positions of the same Multiplot, then the settings for only the last subplot closed is saved. Default is FALSE.

Table 46 Plot Characteristics Variables

Variable	Definition
ANNOTATE_AUTO	Sets a flag to enable or disable automatic annotation update upon data changes. Default is No.
ANNOTATE_CSET	Sets a *Starbase* character set to be used for annotation texts. Default value is determined by */usr/lib/starbase/defaults.*
ANNOTATE_FILE	Sets a file name from which a plot reads in an annotation text. Default is no file to read.
ANNOTATE_MACRO	Sets a macro name that is executed by a Plot for generating an annotation text. Default is no macro to execute.

Table 46 Plot Characteristics Variables (continued)

Variable	Definition
CHECK_PLOT_MATCH	Lets IC-CAP check if a given XY pair belongs to the same Setup. If No, potentially mismatched XY pair can be shown in a tabular format with Display Data. Default is Yes.
DASH_DOT	Sets the number of data points at which a simulated line changes from a dashed to a dotted line. Used in Plot. Default value is 32.
FIX_PLOT_SIZE	If Yes, Plot windows open using the size specified by GWINDX and GWINDY. If No, they open using the last displayed size. Default is No.
GWINDX	Sets the initial Plot window horizontal size in 1/100mm. Used in Plot. Default value is 12500.
GWINDY	Sets the initial Plot window vertical size in 1/100mm. Used in Plot. Default value is 9000.
IGNORE_PLOT_LOC	If Yes, Plot windows open using the X windows system configuration. If No, they open using the last displayed location. Default is No.
MINLOG	Can be set to a real value. Defines the value to be used in a LOG plot, if data point value is zero or negative. Default is 10e-18.
OFFSCREEN_PLOT_LINE_WIDTH	Sets the line thickness used when drawing a plot to an HPGL file or HPGL printer. OFFSCREEN_PLOT_TRACE_LINE_WIDTH will override this value for the traces on a plot. Default is 0.
OFFSCREEN_PLOT_TRACE_LINE_WIDTH	Sets the line thickness used when drawing the traces of a plot to an HPGL file or HPGL printer. Default is 0.
PLOT_LINE_WIDTH	Sets the line thickness used when drawing a plot. PLOT_TRACE_LINE_WIDTH will override this value for the traces on a plot. Default is 1.

Table 46 Plot Characteristics Variables (continued)

Variable	Definition
PLOT_TRACE_LINE	Sets whether trace line is drawn or not. If defined as Yes, the trace lines are drawn. If defined as No, the trace lines will not be drawn, and instead markers will be drawn. Default is Yes.
PLOT_TRACE_LINE_WIDTH	Sets the line thickness used when drawing the traces of a plot. PLOT_TRACE_LINE_WIDTH will override this value for the traces on a plot. Default is 1.
RETAIN_PLOT	When Yes and Auto Scale is off, plot is not erased when updated to allow overlay of curves if the X server has *backing store* capability. Default is No.
RI_GRAPH_SYMMETRY	When defined as Yes, the plot title is displayed. If defined as No, the plot title is not displayed. Default is Yes.
SHOW_GRID	When No, plot eliminates XY grids and leaves tics. Default is Yes.
USE_PLOT_LOOKUP	Lets IC-CAP perform auto-lookup of X data from each Y data. Another way to disable auto-lookup is to use an arbitrary expression for an X data. Default is Yes.

Editing a Plot

To edit an existing plot definition:

1. Open the Plot Editor:

    - From the plot window, click **Options** > **Edit Definition** or right click on the plot then choose **Edit Definition**.
    - From the DUTs-Setups folder, select the setup. Then select **Plots**. Now double-click the plot table you want to edit or select the plot table and click **Edit**.

    Alternatively, you can edit directly in the plot table. See "Table and Text Editors" on page 40.

2. Change the fields as needed and click **OK**. The plot definition is updated.

Notes:

- If the plot window is open when you click OK, the plot window closes then immediately reopens with the new definition. The exception is if you were editing the definition of a sub-plot in a Multiplot window. In that case, the Multiplot window refreshes with the updated definition. If the selected plot was also displayed in a single plot, the single plot closes and does not reopen.
- In a Multiplot window, if you right click on a plot to open the Plot Editor and change the plot's name, when you click OK the plot disappears from the Multiplot window.

Tips:

- Use Display Plot to view the plot window for an individual, selected plot.
- Use Display All to view plot windows for all defined plots.
- Use Close All to close all open plot windows.
- For all plots except Multiplot, use View to see data in a tabular format.

Defining Axis Types and Data Sets

In the example definition, both X axis and Y axis are defined as LINEAR. You can change this type to either LOG10 or DB. Although the example does not need to have a log Y axis, try changing it:

1. Select the example plot, **idvsvg**. Click **Edit**.
2. In the Y Axis Type field, toggle to **LOG10** and click **OK**.
3. Click **Display Plot** to view the graph. The graph *idvsvg* is updated with a logarithmic Y axis.

NOTE The XY GRAPH is not a convenient means for viewing complex numbers. If you want to make a graph with complex numbers, use either the RI GRAPH or the POLAR GRAPH. These graphs take one sweep data set, such as frequency, and generate a Real / Imaginary graph or a Polar graph.

Y2 Data

Another Y axis, called Y2 data, is available for XY GRAPH. The Y2 data axis is useful for showing another data set with a different magnitude or domain. For example, in a gain and phase versus frequency graph you can use the Y axis for gain and the Y2 axis for phase. The Y2 axis always requires one or more Y data sets because a slope of a Diag Line is derived from the left axis values.

The program assumes that Y2 data shares the same setup with the X data, so the program never looks up the Y2 data and always draws against the X data in the plot definition.

Expression

An expression can be entered into the data set fields described above, just like any other fields where an expression is allowed. For example, you can enter a natural log of *id* as *log(id)* in the Y Data 0 field. The calculated data set belongs to the setup where this plot definition exists.

NOTE

A scalar value is expanded to an array of the constant value so that it shows up as a straight line on a graph. For more information on functions, refer to Chapter 9, “Using Transforms and Functions.”

Multiple X Data

IC-CAP allows plots to have multiple XY pairs in a single graph by looking up the same X data name in another setup. This is useful when comparing multiple XY pairs with different X ranges.

Auto Lookup

Each Y data set is examined for its corresponding X data set. This feature, called Auto Lookup, takes the X data name from the plot definition and searches this name in the setup that has this particular Y data in question. If there is no such X data in that setup, then the X data in this plot definition is used.

For example, the next definition draws two different sets of *id vs vg* curves based on large and short setups so that one set represents id vs vg of nmos2/large/idvg setup while the other set shows that of nmos2/short/idvg. The Y data names do not have to be the same. This feature supports all the report types.

Table 47 Id vs Vg #1

Report Type	XY GRAPH
X Data	/nmos2/large/idvg/vg
Y Data 0	/nmos2/large/idvg/id
Y Data 1	/nmos2/short/idvg/id

If this plot exists in the nmos2/large/idvg setup, then the next definition is good enough to show the same curves. In both cases, the number of data points and points per curve may vary among Y data sets because each Y data is shown against its corresponding X data from their home setup.

Table 48 Id vs Vg #2

Report Type	XY GRAPH
X Data	vg
Y Data 0	id
Y Data 1	nmos2/short/idvg/id

The tabular report drops each Y data set that does not share the X data of the plot definition, because representing multiple, possibly independent X and Y data pairs in a simple 2-dimensional table is difficult.

For example, the plot definition in the second table shows the *vg* and *id* of nmos2/large/idvg setup only. When you must show all data sets that do not share the same X data, set the system variable CHECK_PLOT_MATCH to *No*.

Calculated Data

If an expression is entered into a plot definition, then the calculated result belongs to the setup where the plot definition exists. Therefore, when using a transform, you must calculate data within its home setup.

For example, the next definition based on the previous example gives the wrong curves for Y Data 1 because the *id* in the short setup is drawn against *vg* in the large setup. If the data size does not match between the X and Y data sets, the program issues an error message.

Table 49 Calculated Id

Report Type	XY GRAPH
X Data	vg
Y Data 0	id
Y Data 1	nmos2/short/idvg/id*2

Disabling Lookup

You can disable the Auto Lookup so that all the Y data are drawn against the same X data even if these Y data sets do not belong to the same setup with the X data. To turn off the Auto Lookup set the system variable USE_PLOT_LOOKUP to *No*.

Displaying Raw Data

You can view the raw data for an input, output, transform, or plot (except Multiplot) by selecting it and clicking View (in their respective folders). The data format displays the data point number, data index, and the real and imaginary part of the data.

- Point shows the number of sweep data point starting from 0 (zero).
- Index represents a pair of subscripts for a matrix data, such as S-parameters. These are in row, column format.

The actual data points for an input, output, or transform are shown in real and imaginary format represented in the header by *R* and *I*.

The data format for plots follows the report type of the plot itself. When the report type is *XY GRAPH* only the real part of the data is displayed.

- Pnt/cv shows two digits, as discussed in “Marking a Point” on page 426.
- M indicates measured data and S indicates simulated data.
- C denotes common data that is shared between measurements and simulations (for example, an Input).

The tabular data of a plot shows the Y data that belongs to the setup of this plot. Foreign Y data is not displayed because of the different X data. See “Y2 Data” on page 413.

Displaying Plots

You can open a plot window to view your measured and simulated data (see the following figures). The currently-defined graphs are listed in the Plots folder in each setup. You can open one or more plot windows at a time and each display appears in a separate window. For all plots except Multiplot, you can view the same data at the same time on a graph and in a tabular format.

Measured data is displayed as a solid line; simulated data is displayed as small squares by default. Plots are automatically updated each time a measurement or simulation is performed. After an extraction and subsequent simulation, you can view the plots for agreement between measured and simulated data.

- To view the plot window:
    - For an individual, selected plot, click **Display Plot** Display Plot
    - For all plots defined in a setup, click **Display All** Display All
- To see data in a tabular format for all plots except Multiplot, click **View**. View...
- To close all open plot windows, click **Close All**. Close All

NOTE

Alternatively, you can display a plot by selecting a plot definition and clicking the right mouse button. Then choose an option from the menu.

If you want to display or close plots while a different folder is open, use the tool bar buttons at the top of the Model window: Display Plots (Setup), Close Plots (Setup), Display Plots (DUT), Close Plots (DUT).

Area Tools

To display Area Tools, choose *Options > Area Tools* or right click then choose *Graphic > Area Tools*. The following figure shows a Plot window with Area Tools displayed.

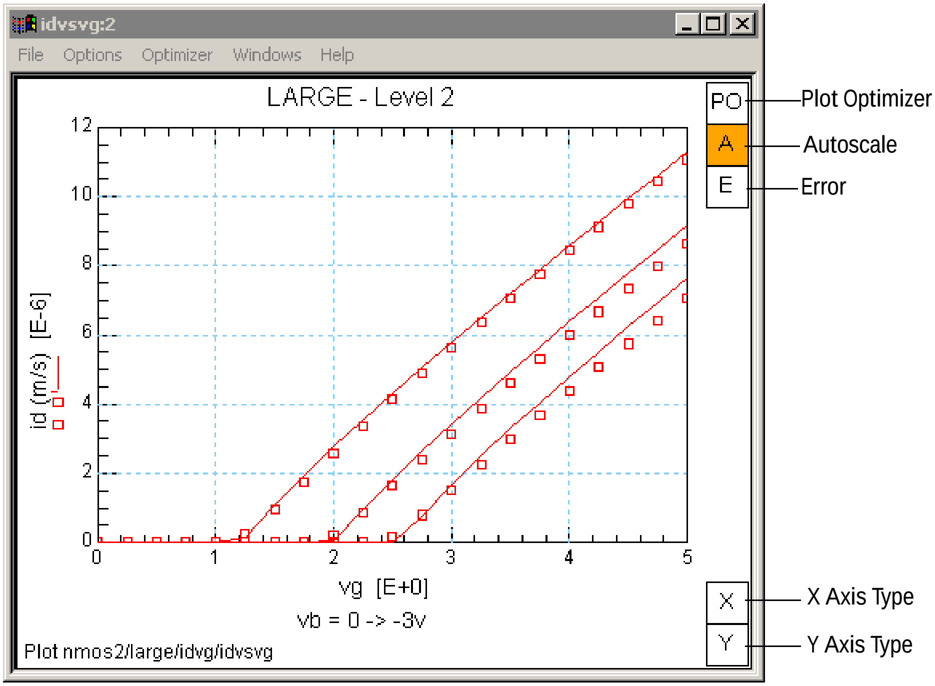


Figure 36 Example Plot Window

The *PO* (Plot Optimizer) area tool performs the same function as the *Optimizer > Enable/Disable Plot* menu pick. When enabled, the PO area tool is blue and the plot has a blue border. For additional information about the Plot Optimizer, see Chapter 8, “Using the Plot Optimizer.”

The *A* (Autoscale) area tool performs the same function as the *Options > Autoscale* menu pick. See “Scaling” on page 421.

The *E* (Error) area tool performs the same function as the *Options > Error* menu pick. See “Defining and Displaying Errors” on page 429.

The *X* (X Axis Type) area tool enables you to toggle through the available X axis types from the plot window. To view the current selection, position and hold the cursor over the area.

The *Y* (Y Axis Type) area tool enables you to toggle through the available Y axis types from the plot window. To view the current selection, position and hold the cursor over the area.

The *Y2* (Y2 Axis Type) area tool enables you to toggle through the available Y axis types from the plot window. This area tool is only shown when a Y2 trace is present.

Multiplot windows also have a Zoom area tool. The Zoom area tool enlarges the selected plot. A red border indicated which plot is selected. When Zoom is enabled, you can change which plot is enlarged simply by clicking on a different plot. In addition, the Plots menu enables you to zoom in or display a full page plot. Figure 38 shows a 2 plot Multiplot window with Zoom enabled.

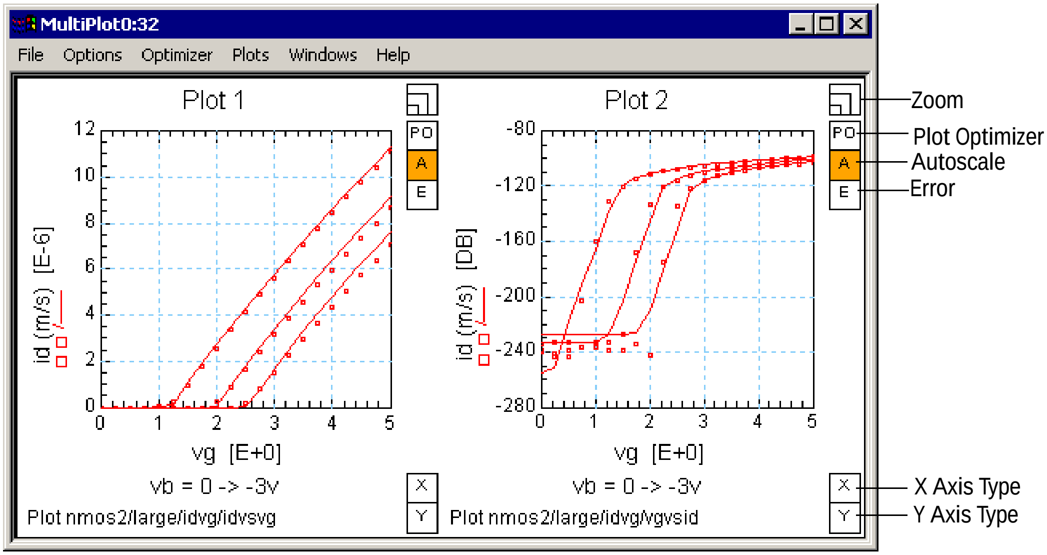


Figure 37 Example Multiplot Window

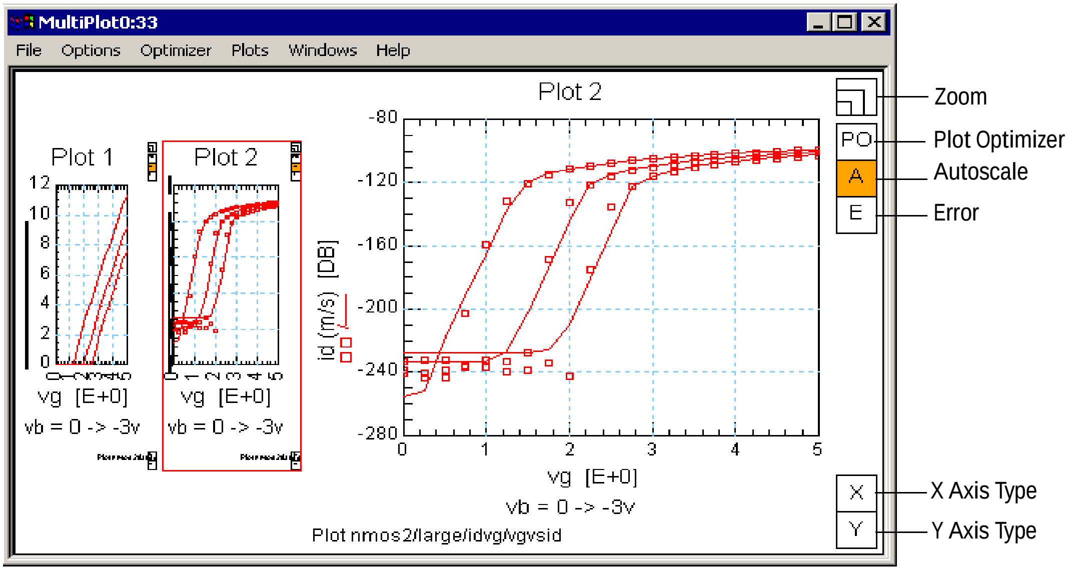


Figure 38 Example Zoomed Multiplot Window

Scaling

Autoscale

Autoscale can be toggled on and off by either:

- Choosing *Options* > *Autoscale*
- Right clicking on the plot then choosing *Scaling* > *Autoscale*
- Clicking the plot's *A* area tool.

If a check mark is before the *Autoscale* menu item or the plot's *A* area tool is orange, Autoscale is on. When Autoscale is on, the axes will autoscale so all data is visible. When Autoscale is off, the state of this menu item is saved to *.mdl* files. For example, if you save a plot (or a model containing the plot) with Autoscale turned off, the plot retains the setting, and opens in the Autoscale OFF state the next time the plot is opened.

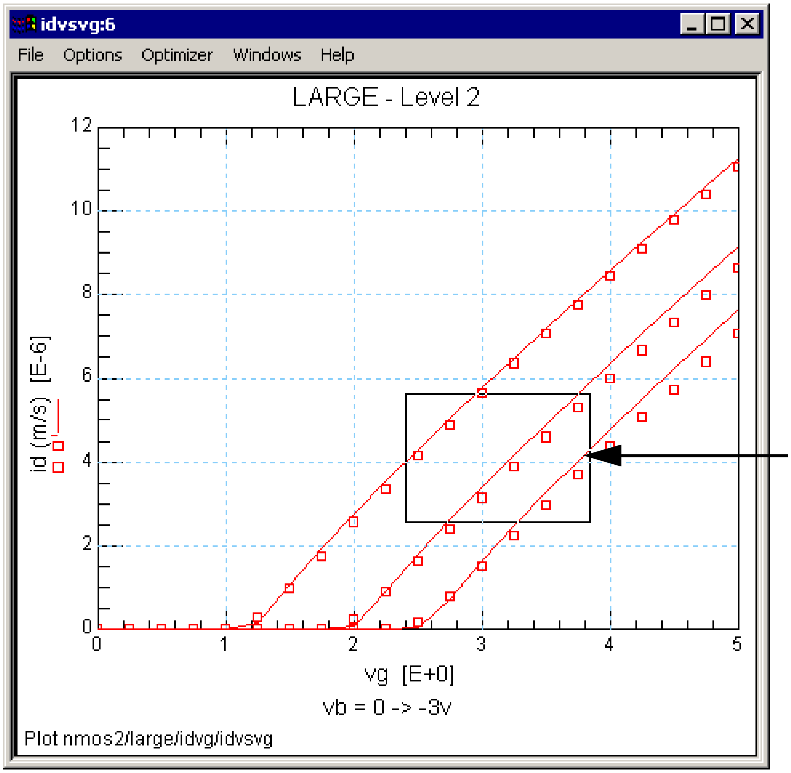
Axes scaling is controlled by settings you choose in the Manual Rescale dialog box (Options > Manual Rescale). If you have a new plot, or a plot that has never had the scaling modified, turning Autoscale off locks the current settings in place. Any future toggling of Autoscale ON/OFF will switch between full scale, and this remembered setting. The remembered setting can be further modified by using the Options > Manual Rescale or Options > Set Scale menu choices.

Rescale

You can zoom into a selected area of a plot by drawing a box around a portion of the plot, then either:

- *Choose Options* > *Rescale*
- Right click on the plot and choose *Scaling* > *Rescale*
- Click the *A* area tool
- In a Multiplot window, select a plot then choose *Plots* > *Selected Plot Menu* > *Scaling* > *Rescale*

Also, you can pan across the plot by selecting a single point on the plot then choose *Rescale*. The plot moves so the selected point is at the display's center. When Autoscale is on, the rescaling is temporary. The next replot restores the plot to full Autoscale.



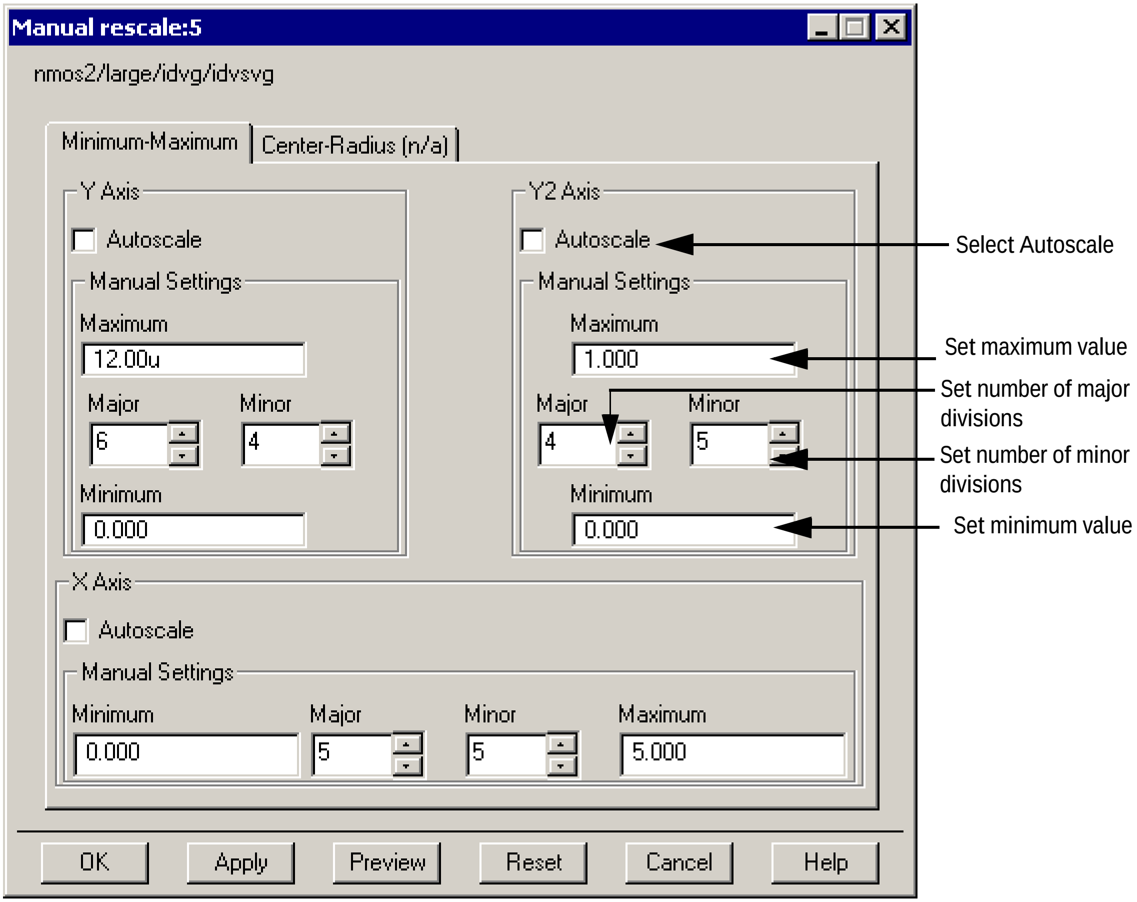
Form a rectangle to define the area to rescale

Set Scale

The *Options* > *Set Scale* menu item is a shortcut to establish Manual Rescale settings. You can also access this menu item by right clicking on the plot then choosing *Scale* > *Set scale* or from a Multiplot window by first selecting a plot then choosing *Plots* > *Selected Plot Menu* > *Scaling* > *Set scale*. Whether you are in autoscale mode or had performed a Rescale to zoom in on a region of the plot, choosing Options > Set Scale sets the Manual Rescale dialog box to the plot’s current scaling values (e.g., X min, X max, Y min, Y max, etc.).

Manual Rescale

The *Manual rescale* dialog box enables you to fully describe all three axes of XY plots in terms of minimum value, maximum value, number of major divisions, and number of minor divisions. You can access this dialog box by choosing *Options* > *Manual rescale* or right clicking on the plot then choosing *Scaling* > *Manual rescale*. In a Multiplot window, you can access this dialog by selecting a plot then choosing *Plots* > *Selected Plot Menu* > *Scaling* > *Rescale*.



Manual rescale:5

nmos2/large/idvg/idvsvg

Minimum-Maximum | Center-Radius (n/a)

Y Axis

□ Autoscale

Manual Settings

Maximum: 12.00u

Major: 6

Minor: 4

Minimum: 0.000

Y2 Axis

□ Autoscale

Manual Settings

Maximum: 1.000

Major: 4

Minor: 5

Minimum: 0.000

X Axis

□ Autoscale

Manual Settings

Minimum: 0.000

Major: 5

Minor: 5

Maximum: 5.000

In addition

- You may leave any of the three axes autoscaled if you like.
- You may scale the real and imaginary axes of RI plots as well. RI, Smith, and Polar plots can be centered around a specified point with a specified radius displayed.
- You may specify the number of major and minor divisions for RI and Polar plots, but not for Smith plots.

Histogram and cumulative density plots permit scaling of the X axis and scatter plots permit scaling of the X and Y axes. Some requests may be denied due to algorithmic constraints, and in this case, the closest match is made updating the dialog with the actual values used. For example, LOG scaled axes are forced to the next decade, and the number of major and minor divisions are ignored. The number of major divisions in polar plots must be even.

Manual Rescale enables you to set the minimum to a specific value, such as .123, without rounding the value up or down. While this gives you much greater control, it can cause some problems with numbers overwriting each other, usually on the X axis. However, LOG scaled axes will still round up or down to the nearest decade.

Any settings established using *Manual Rescale*, the associated *iccap_func* command, or Options > Set Scale are saved with the plot in the .mdl file. To establish settings, choose OK or Apply. (OK also dismisses the dialog box.) To see what a plot looks like before establishing the current settings, choose Preview. To restore settings to the previously established ones, choose Reset or Cancel. To dismiss the dialog box, choose Cancel.

The settings in this dialog box update dynamically whenever a plot's scale changes. Scales change dynamically for various reasons, including when a simulation changes an autoscaled limit, or the Options > Rescale menu item is chosen. Updated settings do not change your established settings until you choose OK, Apply, or Options > Set Scale. Dynamic updating is convenient when you want to establish new settings.

Manual Rescale is fully controllable through PEL. To rescale a plot, use the iccap_func command for Plot:

```
iccap_func("Plot","<keyword>")
```

The allowed keywords include:

- *Scale Plot*
- *Scale RI Plot*
- *Scale Plot Preview*
- *Scale RI Plot Preview*

NOTE

The keywords Scale RI Plot Preview and Scale Plot Preview function in the same way as Scale RI Plot and Scale Plot, but the scaling will be lost on the next Replot command.

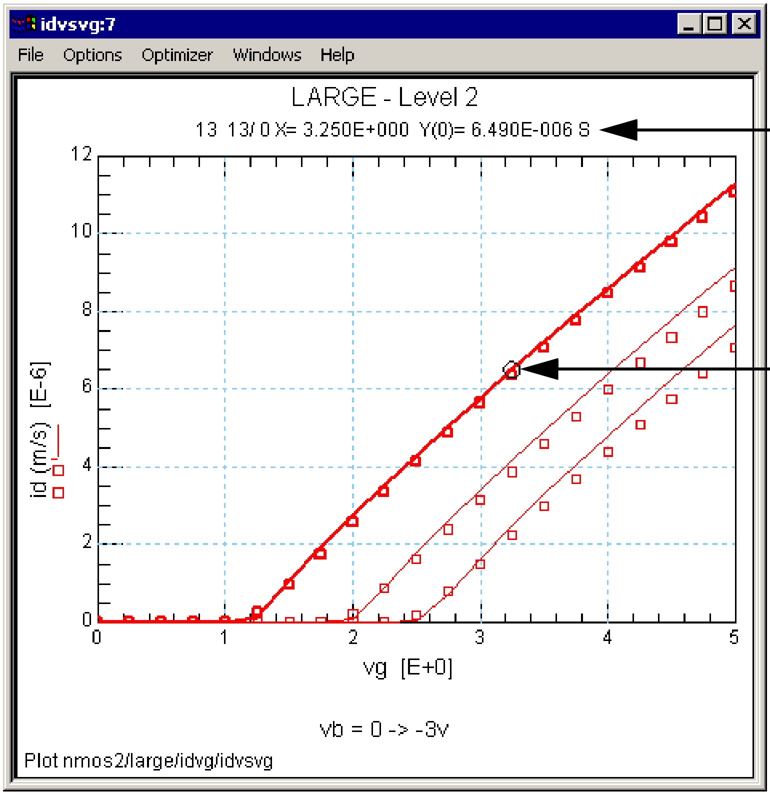
When the *iccap_func* command is used as shown above, prompts appear for the operator to enter required values for the parameters that describe the plot's scaling values. For additional information, see Scale Plot/Scale Plot Preview or Scale RI Plot/Scale RI Plot Preview in the Reference manual.

Marking a Point

You can identify the nearest data point on the graph and show its X and Y values and data point number by marking a point.

To mark a data point:

Click the Left mouse button on the graph.



13 13/ 0 X= 3.250E+000 Y(0)= 6.490E-006 S

Plot nmos2/large/idvg/idvsvg

Shows the values of the point

Marking a Point

Click in this area to erase the point.

To erase the circle that marks the point:

> Click the Left mouse button in the graph window, but on the outside of the graph itself. This erases the circle that marks the point, as well as its data values.

The data point number has two digits separated by a slash. The number on the left of the slash represents the primary sweep point number (sweep order 1), and the number on the right shows the secondary sweep point number (sweep order 2). A data point on Y2 axis can not be identified.

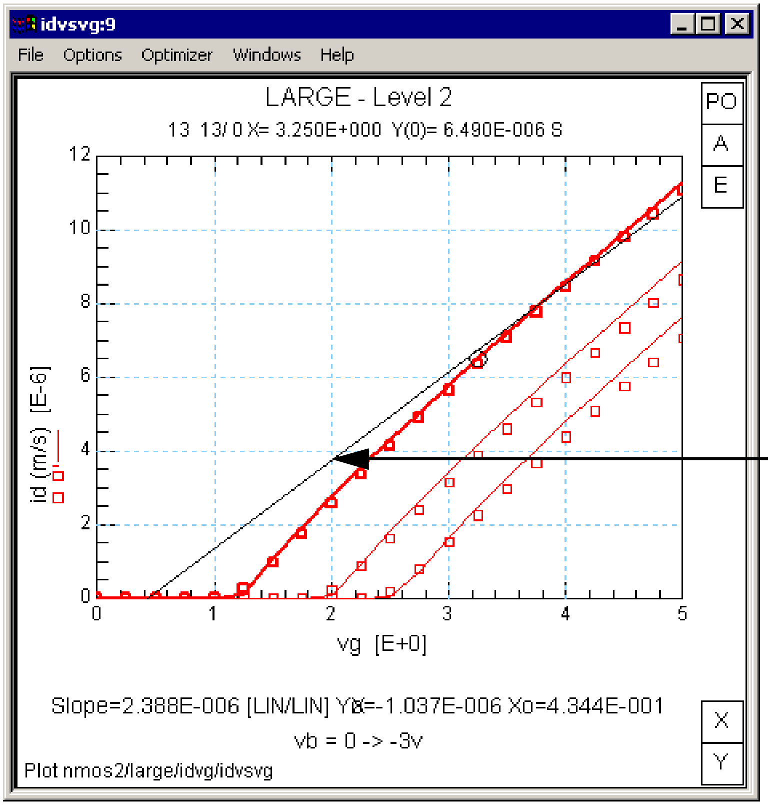
Drawing a Diagonal Line

You can draw a diagonal line connecting two clicked points and its slope, with both X and Y axis intercepts.

To draw a diagonal line:

1 Click and drag the Left mouse button from one point on a curve to a different point forming a rectangle.

2 Choose **Options > Draw Diag Line** (or from a Multiplot window, choose **Plots > Selected Plot Menu > Graphic > Draw Diag Line**).



LARGE - Level 2

13 13/ 0 X= 3.250E+000 Y(0)= 6.490E-006 S

Slope=2.388E-006 [LIN/LIN] Yo=-1.037E-006 Xo=4.344E-001

vb = 0 -> -3v

Plot nmos2/large/idvg/idvsvg

Drawing a diagonal line

NOTE

To erase the diagonal line, select **Options > Draw Diag Line**, without going through the mouse clicks forming the rectangle.

Setting Variables

When a rescale rectangle is shown on a plot, its X and Y values can be copied to system variables. Four variables (X_LOW, X_HIGH, Y_LOW, and Y_HIGH) are reserved for this purpose. You can use this feature when setting X and Y limits for optimization by specifying these variables in the Optimizer Options Table.

1. To perform *Set Variables* define these system variables at any level. Only the variables of interest must be defined.
2. With the left mouse, click and drag on a plot to form a rectangle.
3. Choose **Options > Copy to Variables** (or press C on the keyboard). From a Multiplot window, choose **Plots > Selected Plot Menu > Graphic > Copy to Variable.**

NOTE

When a Multiplot includes plots from different setups, choosing *Copy to Variable* updates the variables in the setup where the single plot was initially defined and does *NOT* update the variables in the setup for the Multiplot.

NOTE

The data values are taken directly from the plot. Therefore, you may need to transform those values to another form when they are used in the Optimizer Options Table. For example, if the optimization target is log(ic) and the plot shows ic versus ve with LOG10 Y axis type, the Y Bound in Optimization should be log(Y_LOW) and log(Y_HIGH).

Defining and Displaying Errors

You can display a plot’s relative or absolute error based on either the entire plot or a selected region. The Error menu is only active when the Y2 trace is not occupied by another trace. In addition, only one trace of type Both can be present and if 2 traces are present, one must be of type Measured and the other of type Simulated or calculated. To view the Error menu, choose *Options* > *Error* or right click on the plot then choose *Error*. The Error menu contains the following choices:

- *Show Relative Error* toggles the MAX and RMS relative errors in the footer area on or off.
- *Show Absolute Error* toggles the MAX and RMS absolute errors in the footer area on or off.
- *Select Whole Plot* uses all points in the measured/simulated datasets to calculate the error.
- *Select Error Region* uses only the points within a defined region to calculate the error. To define a region, click and drag on the plot to form a box. When *Select Error Region* is selected, a green box that delimits the error calculation replaces the white box.

NOTE

A simulated/calculated data trace usually varies during tuning/simulation. Therefore once a region is selected, only a fixed number of measured points inside the green box are used to calculate the error terms.

If *Area Tools* are enabled, you can click on the plot's *E* area tool to toggle between displaying relative error, absolute error, or not displaying any error. The E area tool is green if error calculation is enabled. To enable Area Tools, choose *Options* > *Area Tools*.

Setting Plot Options

Using the Plot Options dialog box, you can define trace options, plot options, text annotation, and for Multiplots, specify a PEL callback to run whenever the selected plot changes.

The Plot Options dialog box can be opened from the:

- Main window, Tools menu
- Model window, Tools menu
- Plot Editor dialog box
- Plot window, Options menu or right click menu (except from Scatter, Histogram, or CDF plots that were opened by the Statistic Package)

If the Plot Options dialog box is opened from the Plot Editor dialog box or from a plot window, a *Preview* button is displayed between the *OK* and *Cancel* buttons. To apply the current settings to the plot without closing the Plot Options dialog box, choose the *Preview* button.

To apply the current settings and close the dialog box, choose the *OK* button.

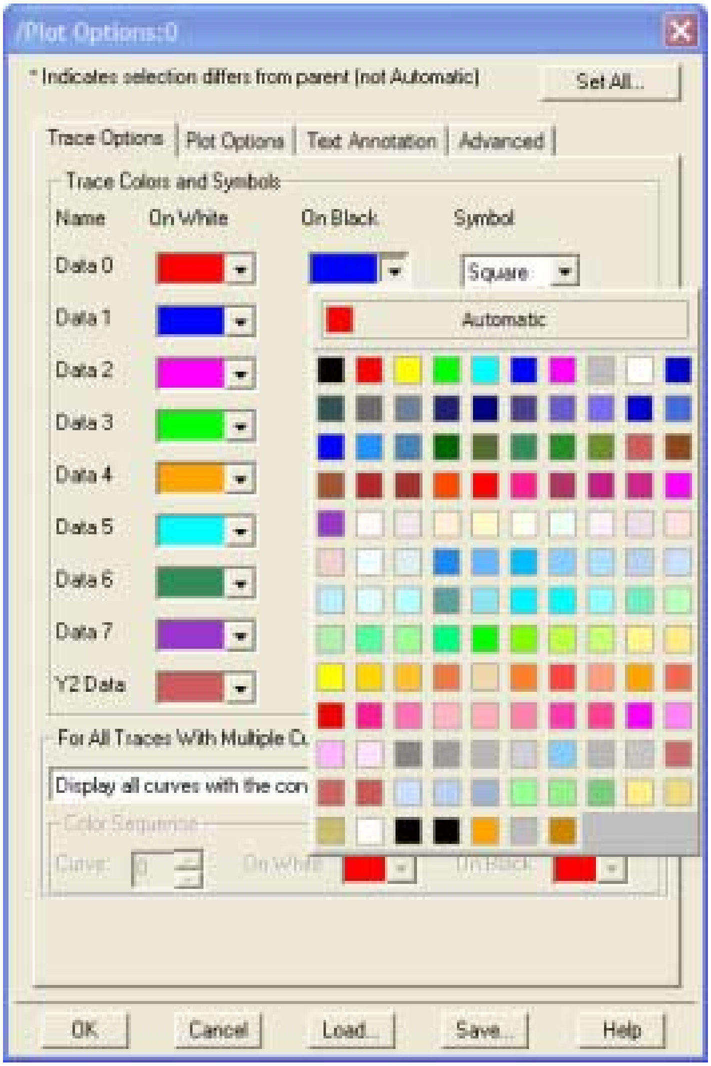
To discard all changes and close the dialog, choose the *Cancel* button.

To save the current settings to a file for later use, choose the *Save* button. The *Save Plot Options* dialog box appears. Enter a file name with a *.pot* extension and choose a file location.

To load previously saved plot option settings, choose the *Load* button. The *Open Plot Options File* dialog box appears. Locate the saved file then choose Open.

Trace Options

The *Trace Options* tab enables you to define trace colors and symbol shapes for all plots except Scatter, Histogram, and CDF plots. If you select Automatic, the default IC-CAP settings is used. If you select a specific color, an asterisk (*) is display next to the color to indicate that it is not the default color.



/Plot Options:0

* Indicates selection differs from parent (not Automatic)

Trace Colors and Symbols

Name	On White	On Black	Symbol
Data 0	🟥	🟦	Square
Data 1	🟦		
Data 2	🟪		
Data 3	🟩		
Data 4	🟧		
Data 5	🟦		
Data 6	🟩		
Data 7	🟪		
Y2 Data	🟥		

For All Traces With Multiple Cu

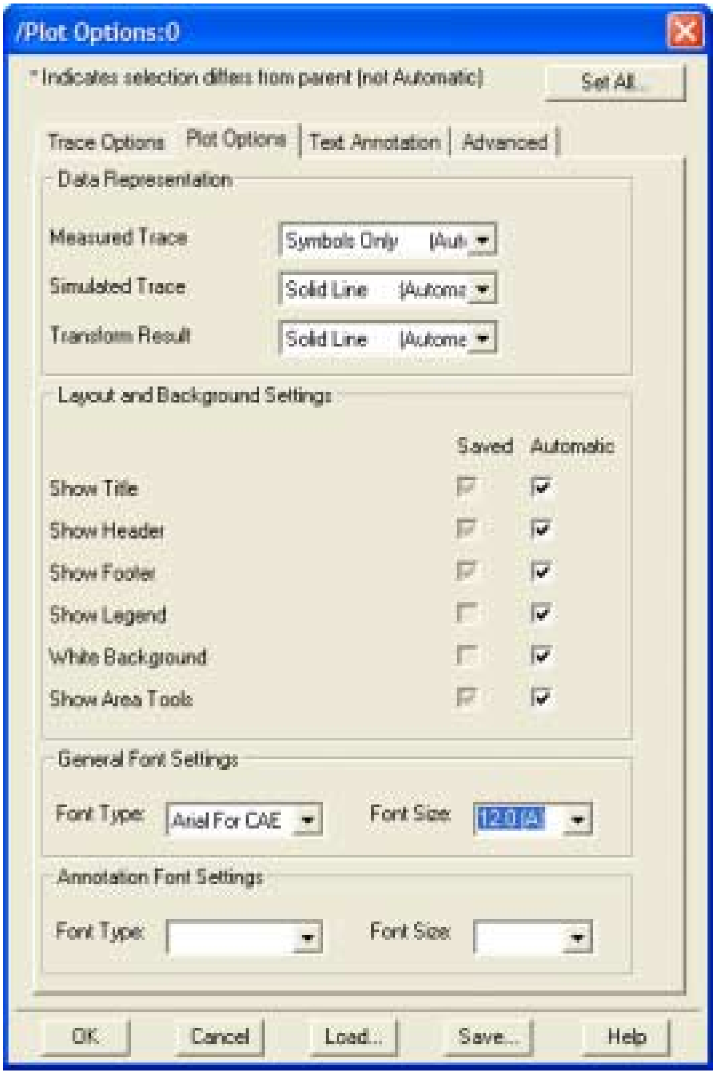
Display all curves with the con

Color Sequence

Curve: 0 On White 🟥 On Black 🟥

Plot Options

The *Plot Options* tab enables you to define Data Representation, Layout and Background, and Text Font. However, for Scatter, Histogram, and CDF plots, the Data Representation section is not available.



/Plot Options:0

* Indicates selection differs from parent (not Automatic)

Data Representation

Measured Trace	Symbols Only (Aut
Simulated Trace	Solid Line (Automa
Transform Result	Solid Line (Automa

Layout and Background Settings

	Saved	Automatic
Show Title	☑	☑
Show Header	☑	☑
Show Footer	☑	☑
Show Legend	☐	☑
White Background	☐	☑
Show Area Tools	☑	☑

General Font Settings

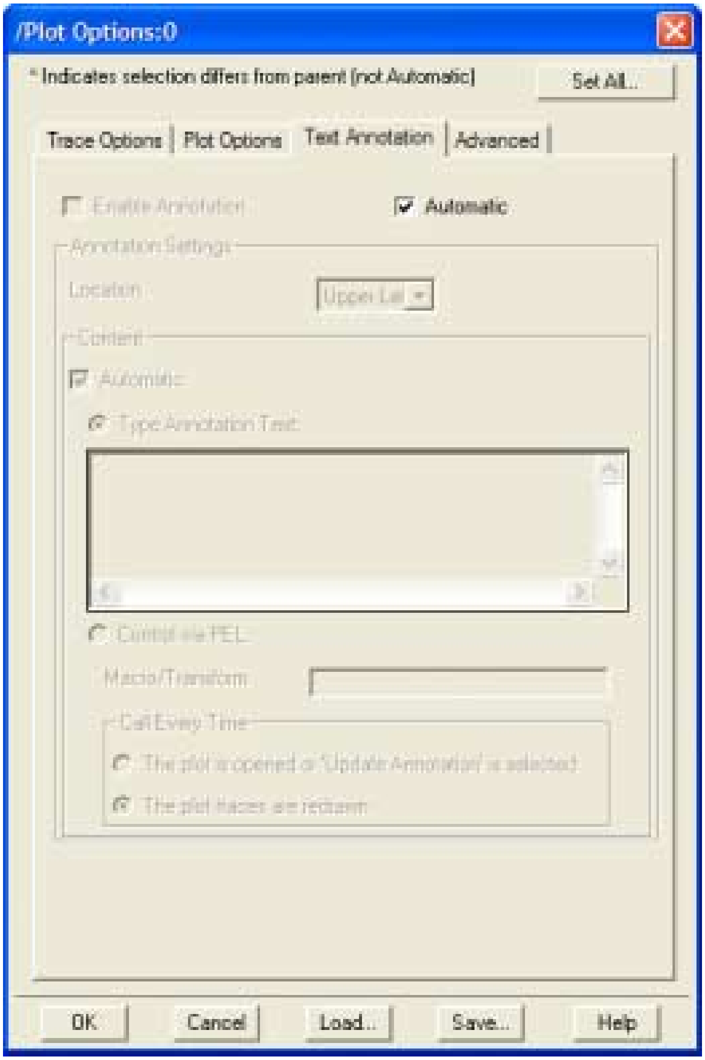
Font Type: Arial For CAE   Font Size: 12.0 (A

Annotation Font Settings

Font Type: Font Size:

Text Annotation

The *Text Annotation* tab enables you to add annotation text to document information such as date, lot number, simulation parameters, and so on. You can either directly enter text annotation or you can specify a PEL macro (see "Annotating a Plot Using a PEL Macro" on page 439).



* Indicates selection differs from parent (not Automatic)

☐ Enable Annotation ☑ Automatic

Annotation Settings

Location: Upper Left

Content

☑ Automatic

● Type Annotation Text

○ Control via PEL

Macro/Transform:

Call Every Time

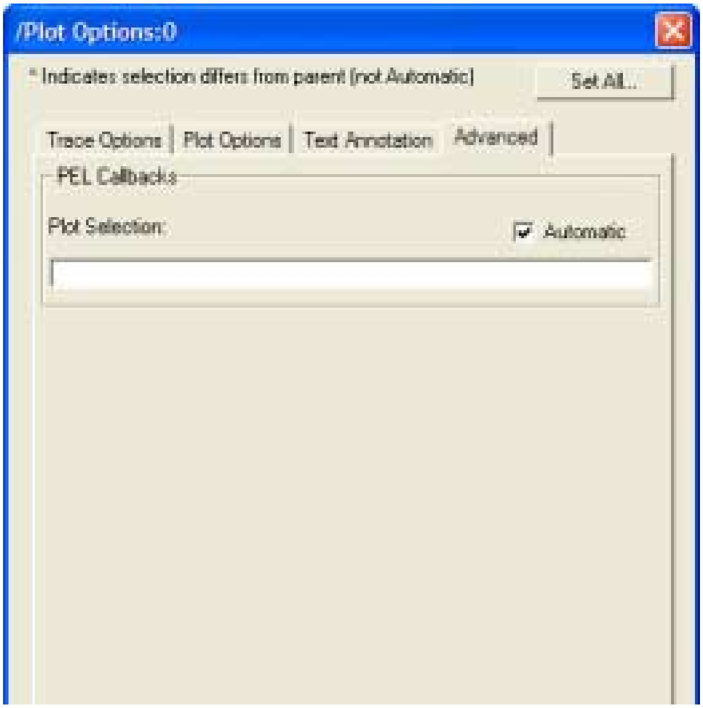
○ The plot is opened or 'Update Annotation' is selected

● The plot traces are redrawn

Advanced

The *Advanced* tab enables you to specify a PEL callback (macro or transform) to run whenever the selected plot in a Multiplot changes. Note that reselecting the same plot will not invoke the callback.

To specify a PEL callback, unselect *Automatic* then type in the callback path and name. Using a relative path, the callback is found relative to the Multiplot. Therefore, the path for macros is ../../MacroName and for transforms is ./TransformName. This is true regardless of where you open Plot Options. For example, if you open Plot Options from the Model window, the relative path for a macro is still ../../MacroName.


/Plot Options:0

* Indicates selection differs from parent (not Automatic)

Plot Selection: ☑ Automatic

The callback passes 3 arguments to the PEL function that are received by calling linput 3 times at the beginning of the callback. The first linput receives the name of the Multiplot where the selection is occurring. The second linput receives the new slot number selected (according to the entries on the plot definition, i.e., Plot 0 will be slot 0, Plot 2 will be slot 2, and so on). The third linput receives the old, no longer selected, plot number. A slot number of -2 indicates that no plots were selected.

Example Plot Selection Callback

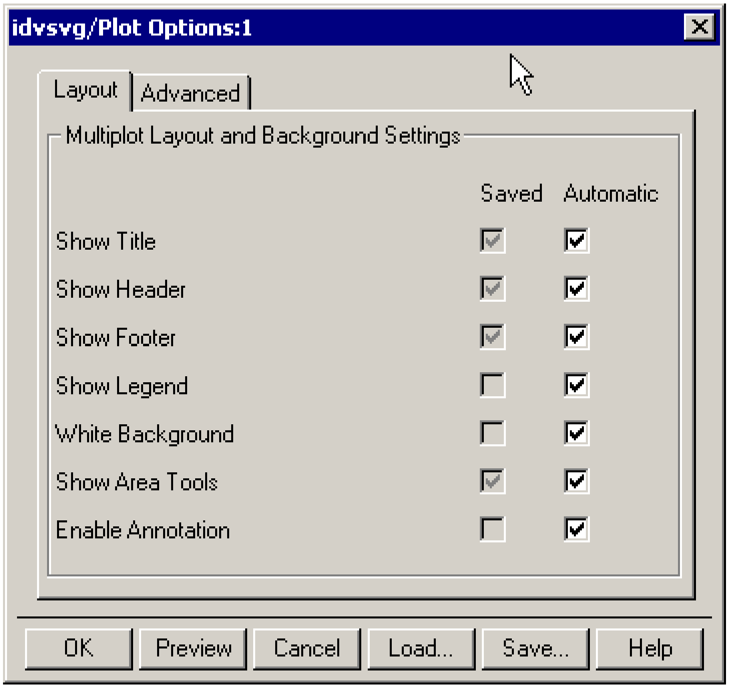
```
linput "plot?",plot
linput "slot?", slot
linput "oldSlot?", oldSlot
print "newSlot=";slot;" oldSlot=";oldSlot;" Plot: ";plot
```

Restrictions

- This callback can only identify when the selection changes. Currently no mechanism exists that can detect when a plot is clicked on *again,* reselecting the same plot.
- Calling *iccap_func()* with “Select Plot” is not allowed during the selection callback and doing so will have no effect. Selection can occur just prior to displaying a right click menu or at the start of a click and drag. Changing the selection at these moments would be problematic, and allowing selection changes for some selections but not others can be equally problematic, so it is not allowed.
- Deleting the invoking Multiplot or the plot that is being selected is not allowed during the selection callback.
- You can modify Multiplot entries during a callback, but any call to *Replot* will do nothing during the callback. This is because changing a graph between selection and display of the right mouse menu can lead to problems.
- In general you should not change the Multiplot, its selection, or any of the child plots during a selection callback. The purpose of the callback is to allow an embedded GUI to change states in reaction to the selected plot.

Multiplot Layout

The *Layout* tab enables you to define the layout and background settings for a Mulitplot.



Multiplot Layout and Background Settings

	Saved	Automatic
Show Title	☑	☑
Show Header	☑	☑
Show Footer	☑	☑
Show Legend	□	☑
White Background	□	☑
Show Area Tools	☑	☑
Enable Annotation	□	☑

Setting Other Plot Characteristics

You can enable or disable several other plot characteristics that are available on the Plot window's *Options* > *Session Settings* menu. Although IC-CAP only retains these settings during the current session, you can save the current setting to Plot Options, or you can reset your settings back to the saved Plot Options.

- *Area Tools* turns the graph's area tools on or off.
- *Legend* turns the graph's legend on or off.
- *Text Annotation* turns the graph's text annotation on or off.
- *Title* turns the graph's title on or off.

- *Header* turns the graph’s header on or off.
- *Footer* turns the graph’s footer on or off.
- *Exchange Black-White* reverses the black and white settings for the graph’s grid, text, and background.
- *Color* turns color on or off for the traces and markers. When *Color* is off, the traces and markers are the same color as the graph’s grid and text.
- *Reset to Saved Options* resets the current session settings back to the saved Plot Options. This menu pick is not available from Scatter, Histogram, or CDF plots that were opened by the Statistic Package.
- *Save Current Settings* saves the current session’s settings as the saved Plot Options. This menu pick is not available from Scatter, Histogram, or CDF plots that were opened by the Statistic Package. However, for these plots you can open the Plot Options dialog box from the IC-CAP/Main window to save the settings.

Saving a Plot Image

You can save an image of a plot with its current characteristics and size.

To save a plot image:

1 Select **File > Save Image**.

NOTE

On Windows, if the plot image is partially hidden by another window when you select Save Image, the saved plot image will be obscured by that window. Therefore, make sure the plot image is not partially hidden by another window before you select Save Image.

2 In the Plot Image File Name dialog box, set the location where you want to save the file.

3 Enter the file name in the form of *filename.ext* where *ext* is gif or jpg.

NOTE

You can save plots in various formats in Plot windows, by using the File > Save Image menu item. This feature uses ImageMagick's "Convert" program. See Help > About IC-CAP for more information. Agilent Technologies officially supports the .GIF and .JPG formats, though .EPS, .PS, .HTML, .TIF, and other formats are available.

4 Select **OK** or **Save**. There may be some delay while IC-CAP processes the image.

NOTE

On Windows, you can copy a plot image to the Windows clipboard, paste it into another applications (such as Microsoft Paint), then save it in the other application.

To copy a plot image to the Windows clipboard, select **Options > Copy to Clipboard** or press Ctrl+C on the keyboard.

Annotating a Plot Using a PEL Macro

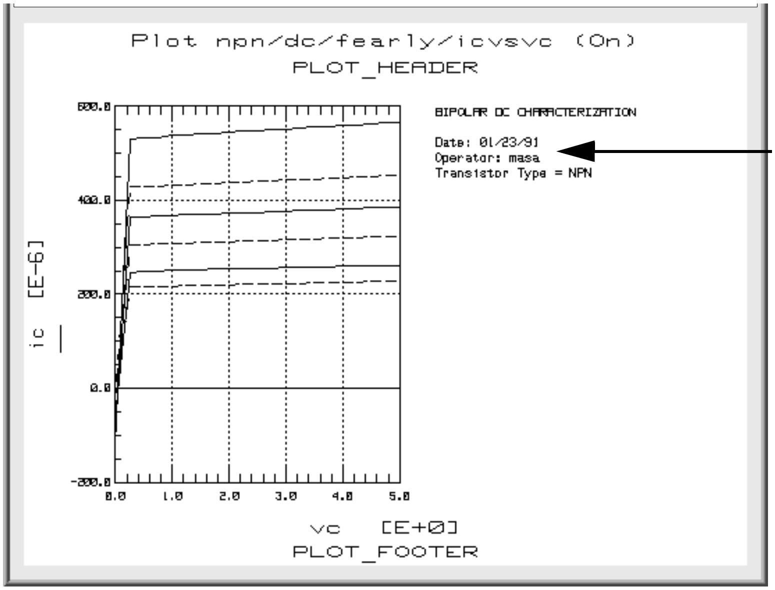
Each plot can read a text file and display this text next to its graph. The text area is 40 characters by 25 lines maximum for the USASCII character set (see *ANNOTATE_CSET* in the section "Annotation Variables").

To add annotation text to a plot:

1 Create a Macro program to generate an annotation text.

2 Assign appropriate values to annotation system variables.

3 Open a plot window.



Plot annotation

Annotation Variables

The variables that control the annotation in a plot can be defined at any level, providing flexibility and programming capability. These variables are:

- ANNOTATE_MACRO. This variables sets a macro name which is executed by a plot for generating annotation text. If this variable is undefined or blank, then no program is executed. A model name may precede a macro name to locate a macro in another model (for example, */npn/legend*).
- ANNOTATE_FILE. This variable sets a file from which a plot reads in annotation text. When this variable is undefined or blank, a plot does not read a file to update its annotation.
- ANNOTATE_AUTO. When this variable is either *Y* or *Yes,* then a plot updates its annotation whenever its datasets are updated, for example, when measured or simulated. Otherwise, the annotation is updated when a plot window is opened or its menu choice *Update Annotation* is selected.

- ANNOTATE_CSET. This is an optional variable to specify a character set for the text. A 16-bit character code is possible by giving a value like *jpn, korean,* or *chinese-t* when *NLIO* (Native Language I/O) is installed. If this variable is undefined or blank, a default character set (*usascii* as set in *$ICCAP_ROOT/config*) is used.

Annotation Example

The annotation example shows how to use the annotation text from the example plot used in this chapter. The following table shows variable values for this example, except ANNOTATE_FILE, which is defined in the *legend* macro program.

Table 50 System Variables for Annotation

Variable	Value
ANNOTATE_MACRO	legend
ANNOTATE_FILE	
ANNOTATE_AUTO	No
ANNOTATE_CSET	

The *legend* macro program is shown in the following figure. This macro sets the ANNOTATE_FILE variable and writes text into this file. Later, a plot reads this file automatically for its annotation.

```
! Plot annotation generation program
!
file_name = system$("echo $HOME/.icplotnotes")
ANNOTATE_FILE = file_name
x = system("rm " & file_name)
!
printer is file_name
!
print "BIPOLAR DC CHARACTERIZATION"
print
print "Date: "; system$("date +%D")
print "Operator: "; system$("logname")
print "Transistor Type = "; POLARITY
!
printer is CRT
```

Figure 39 A Macro Program for Annotation

Advanced Annotation

Annotation for Each Plot Multiple plots in a single setup share the same annotation macro and text file. To have different annotation for each plot in a single setup, create a macro program that generates texts and opens each plot in turn. In this case ANNOTATE_MACRO should be blank.

16-bit Code Annotation If you have installed *NLIO* for 16-bit character code such as Japanese, plots can display these native languages by assigning an appropriate character set to ANNOTATE_CSET. A macro program can include these 16-bit characters in a string and print them out into a file. However, you need to use an external text editor such *vi* in an *hpterm* window to input such characters. To edit a macro program with *vi,* add a new macro program in IC-CAP, write it out to a file, edit this file with *vi,* and then read it back into IC-CAP. IC-CAP can show you these 16-bit characters in a Macro Editor if a 16-bit code font is assigned. For example, for Japanese font, execute the program with the option shown below. Be sure to specify *jpn* for ANNOTATE_CSET in this case.

```
iccap -xrm "iccap*XmText.?*FontList: jpn.8x18"
```

Printing from UNIX

Printing and plotting from IC-CAP on UNIX is accomplished by establishing the desired print setup and then choosing **File > Print**. The Print Setup and related dialog boxes enable you to:

- Select a printer/plotter other than the default
- Install additional printers or plotters
- Set the print resolution (dpi)
- Scale the output

When you select a printer/plotter, you can also change the following default printer/plotter-specific options:

- Page Size
- Source (Paper Tray)
- Duplexing (Double-sided)
- Orientation (Portrait or Landscape)
- Color (Black and White or Color)

When you choose **File > Print**, you can select from the following additional options:

- Choose to send output to a printer/plotter or print to file
- Scale the output to fit to the page
- Select a file format (if printing to file)
- Specify the number of copies

Your print setup is saved in *$HOME/.Xprinterdefaults*. If you do not have a local copy of this file, or the file *.Xpdefaults* (from a previous release), the default file is read from the *$HPEESOF_DIR/xprinter* directory. When you change your print setup, the changes are saved (as new defaults) to *$HOME/.Xprinterdefaults*.

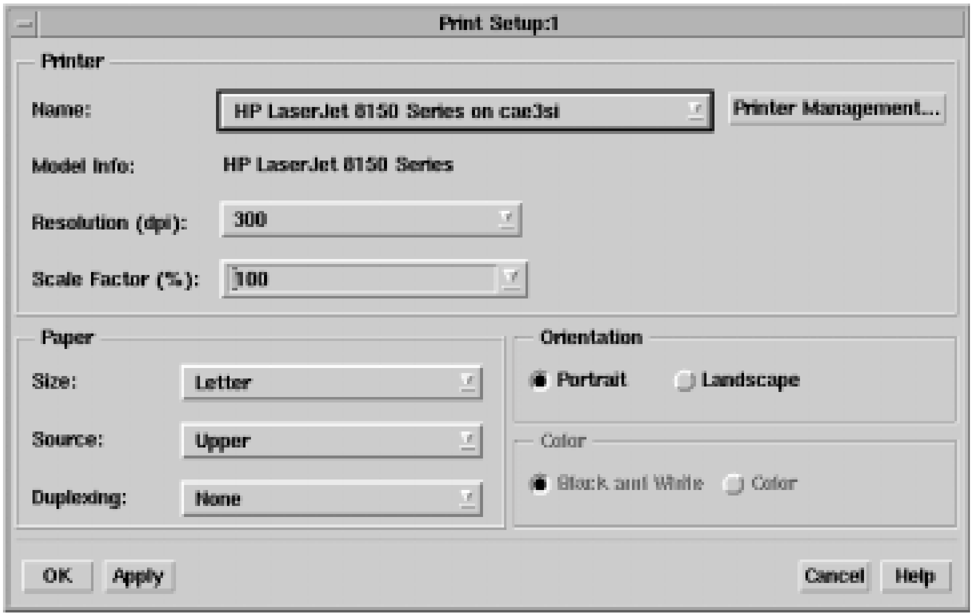
NOTE

If you do have a file *.Xpdefaults* (from a previous release), the settings of this file are copied to the new filename to serve as the starting point for your print setup. Both files are valid, depending on which release of IC-CAP you are using. The old file is maintained for running an earlier version of IC-CAP, but the new file is used when you run IC-CAP 2006 (or later).

Setting Up a Printer

The *Print Setup* dialog box enables you to setup and manage your printer options. To access the Print Setup dialog box,

1 Choose **File > Print Setup**. The *Print Setup* dialog box appears.



Print Setup:1

Printer

Name: HP LaserJet 8150 Series on cae3si

Model Info: HP LaserJet 8150 Series

Resolution (dpi): 300

Scale Factor (%): 100

Paper

Size: Letter

Source: Upper

Duplexing: None

Orientation

● Portrait ○ Landscape

Color

● Black and White ○ Color

For detailed information on using the Print Setup dialog box, refer to the following table.

Table 51 Using the Print Setup Dialog box

Option		Description
Printer		Use this section of the Print Setup dialog box to define and manage your printers.
	Name:	Select a printer from the Name drop-down list. If the printer you want to use is not available in the list, click the Printer Management button. For more information, refer to “Managing Printers” on page 447.
	Printer Management	If you want to add, replace, or remove a printer, click the Printer Management button to access the Printer Management dialog box. For more information, refer to “Managing Printers” on page 447.
	Model Info:	This section displays the selected printer model information.
	Resolution (dpi):	This option enables you to set the print quality (resolution) in dots per inch (dpi).
	Scale Factor (%):	This option enables you to set a scaling factor that defines the percentage of normal size by which to enlarge or reduce the document on the page. Default is 100%.

Table 51 Using the Print Setup Dialog box (continued)

Option		Description
Paper		Use this section of the Print Setup dialog box to define the paper settings.

	Size:	Use the Size drop-down menu to select the paper size. Default is Letter.
		Note that paper sizes can vary depending on the paper manufacturer. If you own a printer that provides a specific paper size that you are not familiar with, consult your printer manual for information on the paper size options available. The more common American paper sizes are listed below in inches (Width vs. Height).
		Letter = 8.50" (W) x 11.00" (H)
		Executive = 7.25" (W) x 10.55" (H)
		Legal = 8.50" (W) x 14.00" (H)
		Tabloid = 17.00" (W) x 11.00" (H)
		Ledger = 11.00" (W) x 17.00" (H)
		Consult the International Organization for Standardization (ISO) for standard A through D-sizes as well as RA and SRA-sizes.

	Source:	Use the Source drop-down menu to select the tray that has the paper you want to use. Default is the Upper tray.
	Duplexing:	The Duplexing drop-down menu enables you to print on both sides of the paper. The options available are: - None - No duplexing, the document will only print on one side of the paper. - Flip on Short Edge - This option prints on both sides of each sheet and flips the page along the short edge of the paper. - Flip on Long Edge - This option prints on both sides of each sheet of paper and flips the page along the long edge of the paper. The default Duplexing option is None.

Table 51 Using the Print Setup Dialog box (continued)

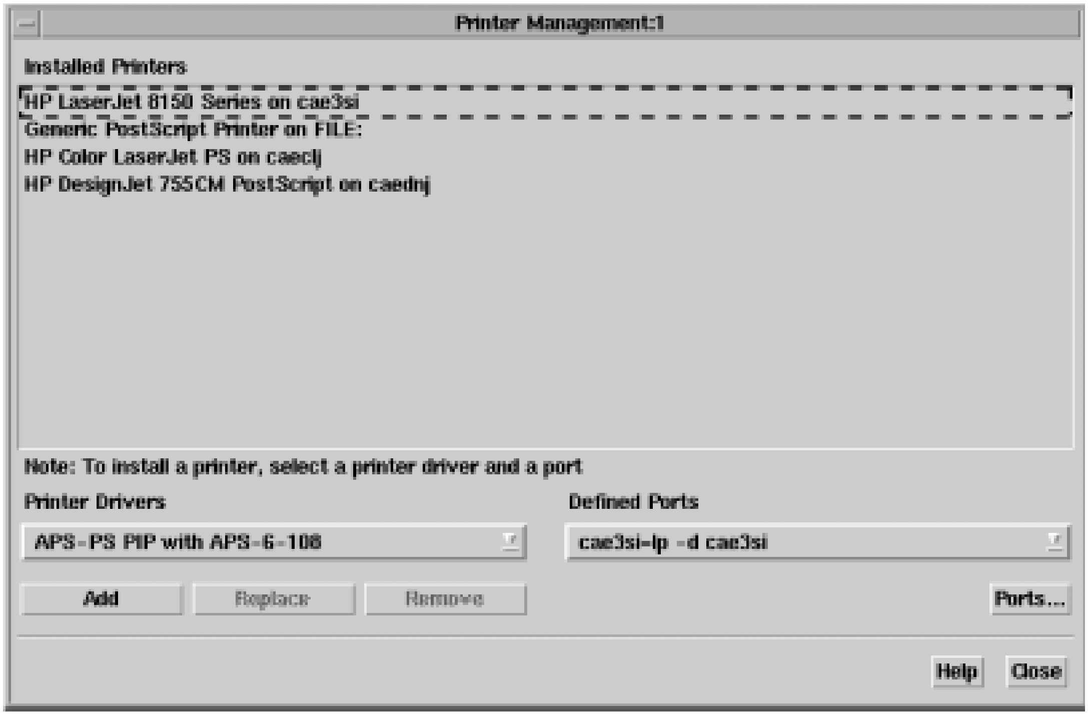
Option		Description
Orientation		Use this section of the Print Setup dialog box to define the orientation of your printed page.
	Portrait	Click this option if you want your output printed in portrait orientation mode. Portrait is taller than it is wide when you view the text right-side up. Default is activated.
	Landscape	Click this option if you want your output printed in landscape orientation mode. Landscape is wider than it is tall when you view the text right-side up. Default is deactivated.
Color		Use this section of the Print Setup dialog box to define your color settings.
	Black and White	Click this option if you want your output in black and white. Default is activated unless the system detects support for color printer. If this is the case, the system will default to the color option.
	Color	Click this option if you want your output in color. Default is deactivated unless the system detects support for color printer. If this is the case, the system will default to the color option.

Managing Printers

The *Printer Management* dialog box enables you to manage an individual printer or group of printers. To access the Printer Management dialog box,

1 Choose **File > Print Setup**. The *Print Setup* dialog box appears.

2 Click the **Printer Management** button in the *Print Setup* dialog box. The *Printer Management* dialog box appears.



Installed Printers

HP LaserJet 8150 Series on cae3si
Generic PostScript Printer on FILE:
HP Color LaserJet PS on caeclj
HP DesignJet 755CM PostScript on caednj

Note: To install a printer, select a printer driver and a port

Printer Drivers: APS-PS PIP with APS-6-108

Defined Ports: cae3si-lp -d cae3si

For detailed information on using the Printer Management dialog box, refer to the following table.

Table 52 Using the Printer Management Dialog Box

Options	Description
Installed Printers	The Installed Printers field displays a list of all currently installed printers.
Printer Drivers	Use the Printer Drivers drop-down list to select a printer driver.
Defined Ports	Use the Defined Ports drop-down menu to select a defined port.
Ports	To define a new printer port and / or replace or remove an existing printer port, click the Ports button to access the Ports dialog box. For more information on the Ports dialog box, refer to "Defining Printer Ports" on page 449.

Table 52 Using the Printer Management Dialog Box (continued)

Options	Description
Add	To install a new printer, select a printer driver and a defined port, then click the Add button. The new printer appears in the Installed Printers list.
Replace	To replace an existing printer, click the printer you want to replace in the Installed Printers list. Then select a new printer driver and a new defined port. Click the Replace button to replace the printer. The new printer appears in place of the old printer in the Installed Printers list.
Remove	To remove an existing printer or group of printers, click the printer(s) you want to remove in the Installed Printers list and then click the remove button. A confirmation dialog box appears asking if you really want to remove the printer. Click OK to confirm or Cancel to abort the removal process. If you click OK, the printer(s) no longer appears in the Installed Printers list.
Help	Click the Help button to access the online context sensitive help.
Close	Click the Close button to dismiss the Printer Management dialog box and accept the changes. Settings will be saved upon exit.

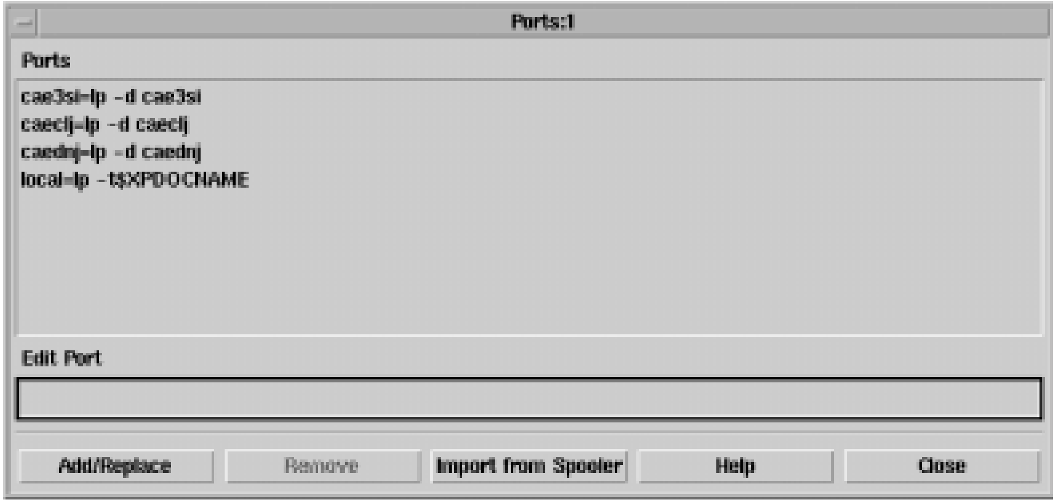
Defining Printer Ports

The *Ports* dialog box enables you to define new printer ports and/or replace or remove existing printer ports. To access the Ports dialog box,

1 Choose **File > Print Setup**. The *Print Setup* dialog box appears.

2 Click the **Printer Management** button. The *Printer Management* dialog box appears.

3 Click the **Ports** button. The *Ports* dialog box appears.



```
cae3si=lp -d cae3si
caeclj=lp -d caeclj
caednj=lp -d caednj
local=lp -t$XPDOCNAME
```

For detailed information on using the Ports dialog box, refer to the following table.

Table 53 Using the Ports Dialog Box

Option	Description
Ports	The Ports field displays a list of ports. Port names can be any names you choose with the exception of *FILE:* which is a reserved port name.
Edit Port	The Edit Port field is used to enter a new port. After you have entered the port definition, click the Add/Replace button.
Add/Replace	Click the Add/Replace button to update the Ports list with contents of Edit Port field.
Remove	If you want to remove a port from the Ports list, click the port name in the Ports list to activate the Remove button. Click the Remove button to remove the selected port.
Import from Spooler	Click the Import from Spooler button to generate a list of ports (based on your *printcap* file). Note that this third party utility is NOT guaranteed to operate on currently supported platforms.
Help	Click the Help button to access the online context sensitive help.

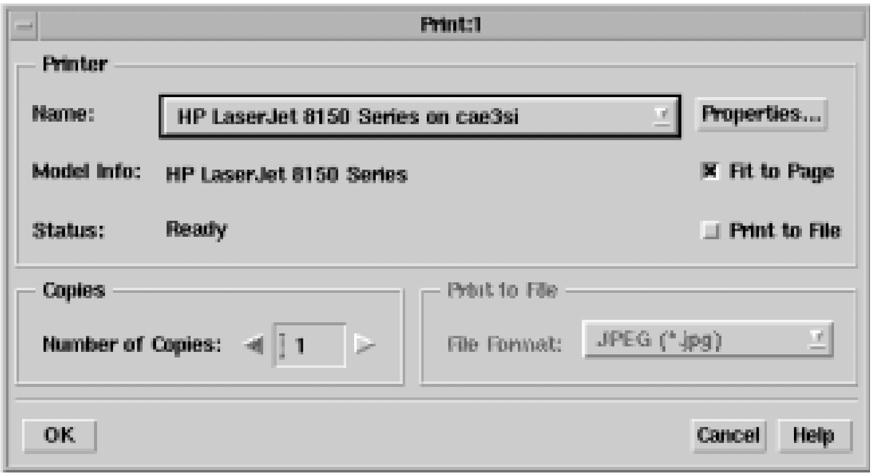
Table 53 Using the Ports Dialog Box (continued)

Option	Description
Close	Click the Close button to dismiss the Ports dialog box and accept the changes. Settings will be saved upon exit.

Printing to a Printer, Plotter, or File

The *Print* dialog box enables you to output your information to a supported printer, plotter, or specified file. To access the Print dialog box,

1 Choose **File > Print**. The *Print* dialog box appears.



Name: HP LaserJet 8150 Series on cae3si

Model Info: HP LaserJet 8150 Series

Status: Ready

☒ Fit to Page

□ Print to File

Number of Copies: 1

File Format: JPEG (*.jpg)

For detailed information on using the Print dialog box, refer to the following table.

Table 54 Using the Print Dialog Box

Option	Description
Printer	The Printer options in the Print dialog box enable you to define a printer and the output characteristics.

Table 54 Using the Print Dialog Box (continued)

Option		Description
	Name:	Select a printer from the Name drop-down list. If the printer you want to use is not available in the list, click the Cancel button and access the Print Setup dialog box.
	Model Info:	This section displays the printer model information.
	Status:	This section displays information on the printer status. If the printer is ready to print, the Status will display Ready.
	Properties	Click the Properties button to view or set additional options used in the Print Setup dialog box. For more information, refer to "Setting Up a Printer" on page 444.
	Fit to Page	Activate the Fit to Page option if you want your output to be automatically scaled to fit on the page.
	Print to File	Activate the Fit to Page option if you want your output to be directed to a file. When this option is selected, the Number of Copies feature is deactivated and the Print to File feature is activated. When this option is deactivated, the Number of Copies feature is activated and the Print to File feature is deactivated.
Copies		This section is used to set the number of copies to print.
	Number of Copies	Use this feature to specify the number of copies to print. If Print to File is selected, this feature is deactivated.
Print to File		This section is used to define the type of file you want to send your output to.

Table 54 Using the Print Dialog Box (continued)

Option		Description
	Name:	Select a printer from the Name drop-down list. If the printer you want to use is not available in the list, click the Cancel button and access the Print Setup dialog box.
	Model Info:	This section displays the printer model information.
	Status:	This section displays information on the printer status. If the printer is ready to print, the Status will display Ready.
	Properties	Click the Properties button to view or set additional options used in the Print Setup dialog box. For more information, refer to “Setting Up a Printer” on page 444.
	Fit to Page	Activate the Fit to Page option if you want your output to be automatically scaled to fit on the page.
	Print to File	Activate the Fit to Page option if you want your output to be directed to a file. When this option is selected, the Number of Copies feature is deactivated and the Print to File feature is activated. When this option is deactivated, the Number of Copies feature is activated and the Print to File feature is deactivated.
Copies		This section is used to set the number of copies to print.
	Number of Copies	Use this feature to specify the number of copies to print. If Print to File is selected, this feature is deactivated.
Print to File		This section is used to define the type of file you want to send your output to.

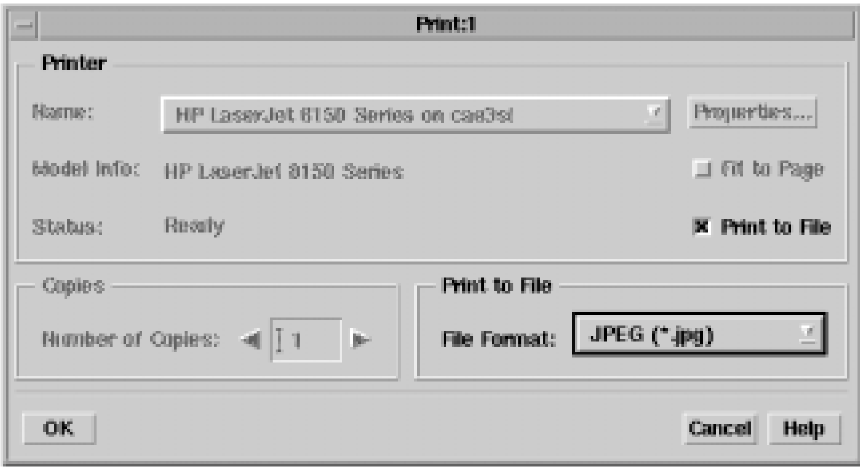
Table 54 Using the Print Dialog Box (continued)

Option	Description
File Format:	Use this feature to specify the file format to save your output to. After you have selected your file format, click OK in the Print dialog box. A Print to File dialog box will appear enabling you to define the path and name of the file. For more information on the Print to File dialog box, refer to “Printing to a File” on page 455. If the Print to File radio button is deselected, this feature is deactivated.
OK	Click the OK button to accept the settings and send your output to the selected printer or file, depending on how you have set your options. Settings will be saved upon exit.
Cancel	Click the Cancel button to dismiss the Print dialog box. Settings will not be saved upon exit.
Help	Click the Help button to access the online context sensitive help.

Printing to a File

The *Print to File* dialog box enables you to define a destination path and file name and then output your information to the specified file. To use the Print to File dialog box,

1 Choose **File > Print**. The *Print* dialog box appears.



Print:1

Printer

Name: HP LaserJet 8150 Series on cae3sl

Model Info: HP LaserJet 8150 Series

Status: Ready

□ Fit to Page

☒ Print to File

Copies

Number of Copies: 1

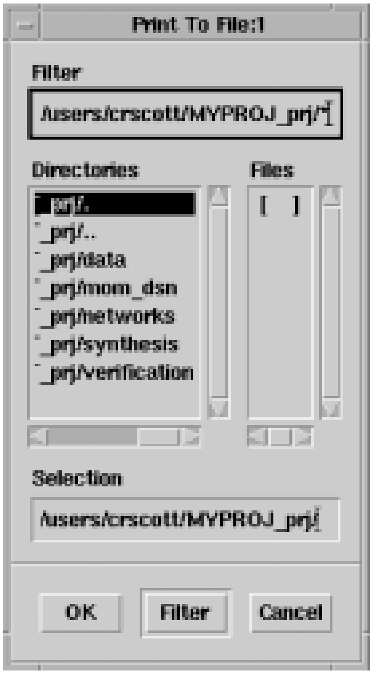
Print to File

File Format: JPEG (*.jpg)

2 Select the *Print to File* option in the *Printer* section of the Print dialog box. Notice that when you select the *Print to File* option, the *Print to File* section is activated enabling you to select a file format.

3 Use the *File Format* drop-down list to select the desired file format. Options include JPEG, GIF, PDF, Bitmap, and HP-GL/2.

4 Click **OK** in the *Print* dialog box. The *Print to File* dialog box appears.



Print To File:1

Filter

/users/crscott/MYPROJ_prj/*

Directories

~_prj/.
~_prj/..
~_prj/data
~_prj/mom_dsn
~_prj/networks
~_prj/synthesis
~_prj/verification

Files

[ ]

Selection

/users/crscott/MYPROJ_prj/

5 Enter the file name for your output in the Selection field.

6 Click the **OK** button to output the file.

Printing from the PC

This section describes some of the actual printing features available on the PC. For information on basics (such as adding a printer), refer to your Windows documentation.

Printing from IC-CAP on the PC is accomplished by establishing the desired print setup and then choosing **File** > **Print**. Listed below are some of the more common options you can set through the Print Setup and related dialog boxes:

NOTE

The options available vary based on the printer/printer driver you select.

- Printer (select any installed printer)
- Paper size and source
- Orientation
- Number of copies
- Single- or two-sided printing
- Scaling

When you choose **File** > **Print**, you can select from the following additional options:

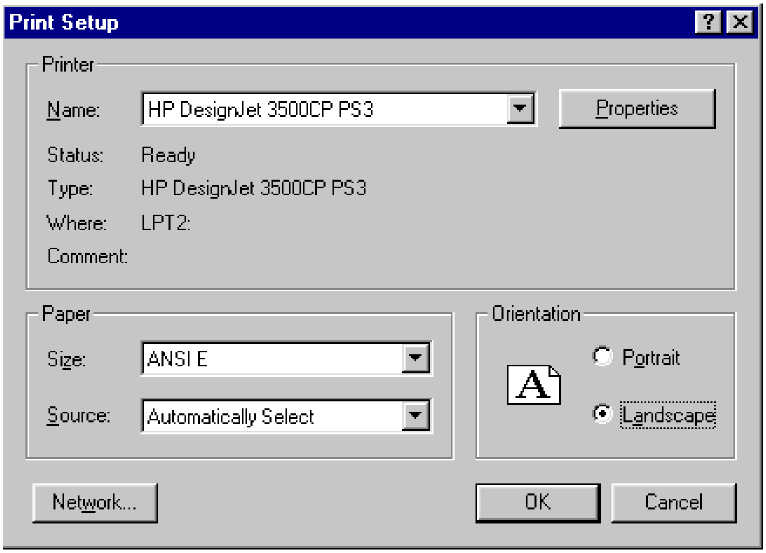
☑ Print to file	Select this option to send output to file for printing at a later time. Select Enhanced Metafile, Windows Metafile, or HP-GL/2 as the file format when the Print to File dialog box appears.
☑ Color output	Select this option to print in color (on a color printer) or in grayscale, rather than black and white (on a monochrome printer).
☑ Copy to clipboard	Select this option to place the image on the Windows clipboard (Bitmap) for pasting in any Windows application.

Establishing a Print Setup

To establish a print setup:

1 Choose **File** > **Print Setup**.

2 Select the desired printer from the drop-down list.



Name: HP DesignJet 3500CP PS3

Status: Ready

Type: HP DesignJet 3500CP PS3

Where: LPT2:

Comment:

Size: ANSI E

Source: Automatically Select

○ Portrait

● Landscape

3 Change any of the options here as desired, or click **Properties** to set additional options, such as *Scaling*. Note that the appearance of the Properties dialog box varies depending on the selected printer.

4 Change any other options as desired and click **OK** to dismiss the Properties dialog box.

5 Click **OK** in the Print Setup dialog box and you are ready to print.

Basic Printing

To send the entire contents of the window to the printer:

1 Choose **File** > **Print** and a dialog box appears.

2 Change any print options as needed, and click **OK**.

Automating Print Functionality

The following print functionality can be automated via PEL:

- Print to the following supported file types:
  - hpgl2
  - black and white postscript
  - emf for the PC only
- Display the File > Print dialog box, which enables the user to select a printer and to request a printout.

Starting with IC-CAP 2004, you can no longer send a print file to the printer from PEL without user intervention. You can not create print files other than the ones listed above. For example, you can not create a color postscript file formatted for a color printer. To generate that type of file, you must use the File > Print dialog boxes.

Also, the semantics of the Print Via Server command have changed slightly with IC-CAP 2004. In previous releases, if a printer was setup as the default, issuing a Print Via Server command would print the file with no user intervention. Now, the Print Via Server command *always* displays the print dialog box requiring user intervention. See the following examples.

```
   ! Displays dialog box as though File > Print menu had been
   ! selected.  You cannot anticipate the dialog with
   ! optional arguments, it will always come up.
iccap_func("myPlot","Print Via Server")

   ! following line creates an HPGL2 file
iccap_func("myPlot","Dump Via Server","Y","hpglfile.hgl",
           "HPGL2")

   ! following line creates a black & white postscript file
iccap_func("myPlot","Dump Via Server","Y","psfile.ps","PS")

   ! following line creates a .emf file on PC only
iccap_func("myPlot","Dump Via Server","Y","emffile.emf",
           "EMF")

   ! Note, the following form of the command worked in
   ! previous releases of IC-CAP but is no longer supported.
iccap_func("myPlot","Dump Via Server","N")

   ! Displays dialog box as though the Print button had been
   ! pressed on a tabular data window
iccap_func("myOutput","Print Via Server")
```

For additional information, see “Dump Via Server” and “Print Via Server” in the *Reference* manual.

11
Creating and Running Macros

This chapter contains procedures for creating, editing, and using macros. Macros can provide a powerful tool for automating routine or repetitive procedures by increasing the efficiency of your operations. Macros, consisting of short programs or scripts that execute IC-CAP commands in a predetermined sequence, automate routine test procedures so that the execution of a single command measures, extracts, simulates, optimizes, plots, and reports on the characteristics of the tested device or circuit.

Macros are simply a way to automate routine operations or to simplify complex operations. For example, you can create a macro to do the following:

- Measure all four setups of the *dc* DUT in the *npn* model.
- Run extractions, simulation and optimization.
- Display all plots in a model.

In addition to these routine tasks, you can perform more sophisticated operations, such as:

- Writing data to an external data base.

- Making choices and branches, based on the value of data sets, parameters or variables.

Pre-defined macros are provided for each model and you can execute macros manually or automatically upon IC-CAP startup. Macros are easy to write—the facilities for creating, editing, and running macros are available in the Macros folder. Each model can have any number of macros, as the example in the next section demonstrates.

Creating and Running a Macro

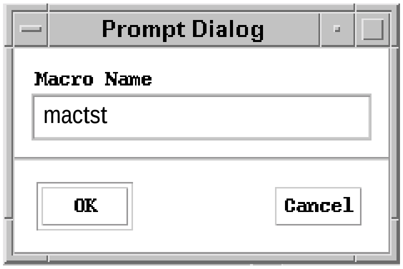
This example uses the Bipolar Transistor model *bjt_npn* and creates a small macro that gives you an immediate visible result.

1 Open the Example model **bjt_npn**.

2 In the Model window click **Macros**.

3 Click **New**.

4 In the prompt, type the macro name mactst.


Prompt Dialog

Macro Name: mactst

5 Click **OK**. The macro name, mactst, appears in the macro list.

6 Select **mactst**. The macro editing text window clears.

7 Click anywhere in the text editing area and type:

```
iccap_func("/npn/dc/fgummel","Display Plots")
```

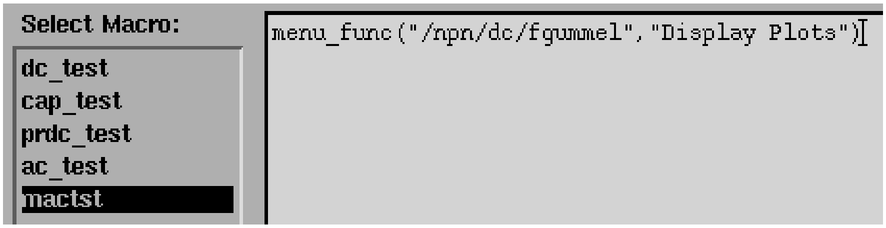
where:

iccap_func specifies the execution of a specific action

/npn/dc/fgummel is the path name of the setup containing the plots

a *comma* (,) separates the path name from the command

Display Plot specifies the action to perform



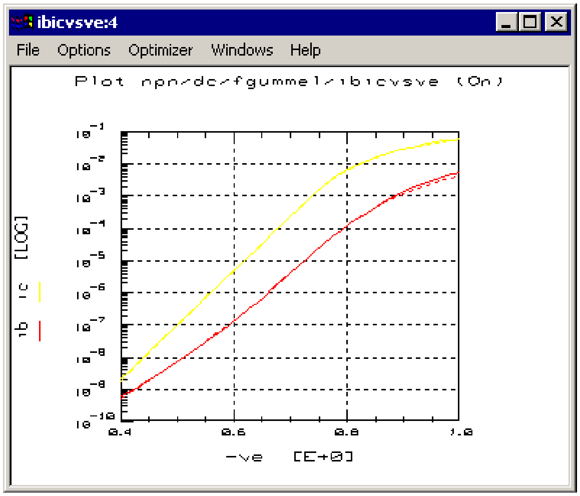
Select Macro:

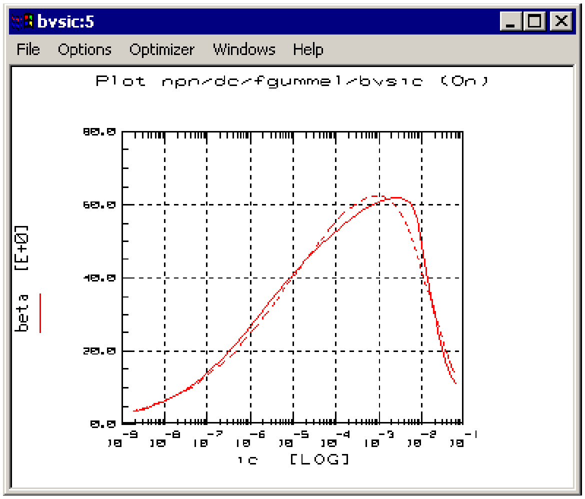
- dc_test
- cap_test
- prdc_test
- ac_test
- mactst

```
menu_func("/npn/dc/fgummel","Display Plots")
```

8 Click **Execute** to run the macro.

9 Plot windows, for the two plots defined, appear in the fgummel setup.





This completes the procedure for creating and testing a simple macro. You can run the macro anytime by selecting it and clicking *Execute*.

NOTE

Alternatively, you can execute macros from the Macros menu in the Model window's main menu bar, **Macros > Execute >** <*macro name*>. This is useful when you want to run a macro and the Macros folder is not open.

You can save the macro by selecting it and selecting **File > Save As**. Select the file type Macro, select the macro from the list, and click **OK**.

Macro Features

As demonstrated in the example, basic macro syntax is quite simple, yet reasonably powerful. This section gives more detail about macro features.

In macros you can write statements that use IC-CAP's Parameter Extraction Language (PEL). The PEL syntax is based on Rocky Mountain Basic or HP BASIC. You can run operating system commands and send data to external files. Macros can store IC-CAP data to data bases.

IC-CAP uses a single interpreter for both the Program function and macros. As a result, these have nearly identical features, except that a macro does not store a data set when it finishes.

NOTE

For details on PEL statements, refer to Chapter 9, "Parameter Extraction Language," in the *Reference* manual.

Program Statements

Several IF statements and a WHILE statement allow looping and conditional execution for a statement or a block of statements.

The LINPUT and PRINT statements permit 2-way communication with an operator and permit sending output to files and instruments.

Built-in Functions

Functions and operators for numerical computation and for handling strings are available. You can compute using single numbers or whole data sets, without additional programming effort. The functions and operators use the same names (such as, LOG SQRT VAL$ and SYSTEM$) as HP BASIC or RMB. The SYSTEM$ function is one of two functions available for invoking HP-UX shell commands. For information refer to "Built-in Functions" in the *Reference* manual.

Calling the Function List Library

Within macros, you can invoke the functions used by transforms without creating transforms. On the other hand, you can also perform an existing transform, if that is more convenient (see the next section, *Calling Menu Functions*). The example macro at the end of this chapter demonstrates a call to the *RMSerror* function.

Calling Menu Functions

A special command, iccap_func, provides a simple but powerful means for controlling the IC-CAP system in an automated fashion (see “Controlling the IC-CAP System” on page 467). The example at the end of this chapter demonstrates the use of iccap_func to perform simulation and extraction.

NOTE

The iccap_func statement replaces the menu_func statement used in earlier releases. Macros using the menu_func statement will still work, but when creating new macros, we recommend using the iccap_func statement. However, note that not all UI functionality is accessible with iccap_func.

Autostart Capabilities

You can request the execution of a particular macro on startup. For details, refer to “Autostart Macros” on page 471.

Controlling the IC-CAP System

This section provides details on the use of the *iccap_func* command. The first example in this chapter illustrated the *iccap_func* command:

```
iccap_func("/npn/dc/fgummel","Display Plots")
```

The *iccap_func* command needs at least two arguments. The first argument is the name of an object on which you want to act and the second argument is the action you want to perform.

In addition, you can provide responses when the macro requires user input (through dialog boxes) during execution. This capability to *answer* provides truly automated operation. If you do not provide *answers* in advance, the dialog boxes appear when the macro executes and wait for input. For further information on this capability, refer to the section *Anticipating Dialog Boxes*.

Specifying an Object

IC-CAP is implemented in an object-oriented fashion. To the user, this means selecting an object in the system, such as a setup, and then selecting an action, such as *Simulate*. The *iccap_func* command reflects this object-oriented style, since the command requires an object name and then an action. The command searches first for the object within the IC-CAP system and then, having found the object, it executes the action to be performed on that object.

When using the iccap_func command, particularly within a macro, you specify objects by using a leading “/” character in their names. (When viewing the setup in the Model window, the leading character is not shown, so it is easy to forget this detail when programming with iccap_func.) For example:

```
iccap_func(“/npn/dc/fgummel”,”simulate”)
```

The hierarchy of objects is illustrated next. An asterisk (*) next to an object (in the illustration) indicates that multiple objects or items with different names can be specified for that object. For example, a model may have many DUTs, but only one Parameter Set.

```
IC-CAP
  |– Variables
  |– GUI Items
    |– GUI Items
  |– Simulation Debugger
  |– Hardware
    |– HPIB Analyzer
  |– PlotOptions
  |– MODEL(*)
    |– Variables
    |– GUI Items
      |– GUI Items
    |– Circuit
    |– PlotOptimizer
    |– PlotOptions
    |– Parameter Set
    |– MACRO(*)
    |– DUT(*)
      |–Variables
      |– GUI Items
        |– GUI Items
      |–Test Circuit
      |–Device Parameter Set
      |–SETUP(*)
        |–Variables
        |– GUI Items
          |– GUI Items
        |–Instrument Options
        |–INPUT(*)
        |–OUTPUT(*)
        |–TRANSFORM(*)
        |–PLOT(*)
          |– PlotOptions
```

NOTE

For exact specification of these objects and descriptions of available arguments, see Appendix F, “ICCAP_FUNC Statement,” in the *Reference*.

Specifying an Action

Having identified an object, such as /npn/dc/fgummel, you then provide the name of an action to perform on it. Many of the available actions in IC-CAP are made up of multiple words, such as *Rebuild Active List* in the Hardware window. When specifying a multi-word action, spaces and capitalization do not matter. However, when specifying the object, spaces (or the lack of) and capitalization are critical. IC-CAP allows you to name one model as *npn* and another as *Npn*, so the spelling and capitalization of an object must be exact.

Ambiguous Object Names

In the IC-CAP system, it is possible for two objects to have the same name. For example, in an npn model, there is nothing to prevent you from naming a DUT as *dc_dut* and naming a macro the same thing. Perhaps this Macro performs all necessary characterization operations for the DUT of the same name.

To eliminate confusion in such cases, include an optional object *type* when specifying the name of the object. For example, suppose you have a DUT and a macro called xyz. Specifying:

```
iccap_func("/npn/xyz","execute")
```

does not work. Even though the DUT does not have an execute action associated with it, IC-CAP does not know how to differentiate the object. Instead, include the object type:

```
iccap_func("macro /npn/xyz","execute")
```

Anticipating Dialog Boxes

It is possible to provide answers, in advance, for dialog boxes that arise during execution of a macro. These are provided as additional arguments to the *iccap_func* command. The following example calls the *Save As* dialog box, which requires a filename for saving the setup:

```
iccap_func("/npn/dc/fgummel","Save As","fgummel.set")
```

In this example, *fgummel.set* is provided as the filename that you would otherwise type in the dialog box displayed by the *Save As* command. You can include any number of string expressions, separated by commas, to anticipate any number of dialog boxes. If you fail to anticipate all of the dialog boxes and do not provide answers for all of them, then the system displays the dialog boxes when the macro runs, and you can provide answers at that time.

Another interactive technique is to anticipate the dialog boxes and use the LINPUT command to prompt the user for the information to be supplied. Without this prompt, you may see a dialog box in which a filename is requested, but not know what the file is to be used for.

```
LINPUT "Enter filename to save the Setup fgummel:",
user_answer iccap_func("/npn/dc/fgummel","Save As",
user_answer)
```

It is possible to stop a macro from running if *iccap_func* generates an error message box. Include the following command to monitor the error log, and read the number of characters in the error box text:

```
x=check_error_log()
```

If the number of characters has increased since the last call to *iccap_func*, the macro can be programmed to stop.

Autostart Macros

You can request the execution of the npn/dc_test macro after the model file is loaded, by starting IC-CAP in this fashion:

```
iccap bjt_npn.mdl -x npn/dc_test
```

You can intersperse any number of model file names and autostart macro execution requests. Just be sure to precede each macro with the -x option, and do not execute a macro before you load the model file containing it. You can load several models before executing any macros, if desired. You can execute 0, 1, or more autostart macros for each model loaded. These options are demonstrated below:

```
iccap bjt_npn.mdl -x npn/dc_test nmos2.mdl
iccap bjt_npn.mdl -x npn/dc_test -x npn/prdc_test nmos2.mdl
iccap bjt_npn.mdl -x npn/dc_test nmos2.mdl -x npn/prdc_test
iccap bjt_npn.mdl nmos2.mdl -x npn/dc_test -x npn/prdc_test
```

Unattended IC-CAP Sessions

It is possible to run an IC-CAP session completely unattended, by executing all needed menu functions within one or more autostart macros. Such a session can be terminated with the following macro statement:

```
iccap_func("IC-CAP","Exit!")
```

In this example, *Exit!* is used to exit without prompting to save changes; to prompt the user to save files before exiting, use *Exit* only.

Disallowed Deletes and Reads

Because the *iccap_func* function can access objects in the system and execute actions on them, some potentially dangerous situations arise. The following is an example of a Macro attempting to *Delete* itself:

```
iccap_func("/npn/mactst","delete")  ! delete myself
```

The system prevents such commands from executing, because reliable operation could not otherwise be guaranteed. Another case which can be less obvious to recognize, but which would be equally dangerous is as follows:

```
iccap_func("/npn","delete") ! delete my parent
```

In this attempt to delete the model that owns the running Macro, the Macro would be destroyed. This is not permitted.

In general, when a Program or Macro is running, attempts to execute the *Delete* or *Open* functions fail when that operation would result in the running Program or Macro being destroyed or overwritten. However, most requests to perform *Delete* or *Open* should succeed. The following example shows a supported case in which a Macro within the *npn* model deletes the *fgummel* Setup:

```
iccap_func("/npn/dc/fgummel","delete")
```

This is not a problem, since the Macro would not be deleted when the *fgummel* Setup is deleted.

Computations with Data Sets

Like Transforms using the Program function (Programs), Macros can apply a variety of operators and functions to data sets. However, it is more difficult for Macros to access the data sets. This is because Macros are further removed from the data sets, which exist inside Setups, alongside Programs. As is discussed in "Using Data from Another Setup" on page 375. Programs can readily access data sets in another Setup by preceding the data set's name with the name of its Setup, as in *rgummel/vb*. (This data set is an Input appearing in the *bjt_npn.mdl* model file.) Simpler still is the case where a Program accesses a data set in the same Setup. In this case it can access an Input named *vb* just by using its name. For example:

```
PRINT max(vb)
```

Since Macros are not inside the Setups, they must fully specify the names of data sets that they use. A leading "/" character must be used. In a Macro, the previous PRINT statement needs to be written as follows:

```
PRINT max(/npn/dc/rgummel/vb)
```

Within a currently executing Macro or Transform, the notation for relative path (../ and ../..) can be used when accessing a data set in another DUT or Setup.

NOTE

Some of the functions in the IC-CAP Function List will not execute within a Macro, although they would succeed in a Program. These functions depend on being able to determine the number of points per curve in a Setup. This is generally because they operate on a single curve in a data set (or on each curve separately). Example: If you use the derivative function in a Macro, the following error box message will appear:

Cannot perform derivative function because points per curve cannot be determined. This function must be done inside a Setup.

The other functions with this limitation are these: *correlation, fit_line, circlefit,* and *linfit*. Starting with IC-CAP 2006B addon3, a new derivative function called *derivative2* is available, which can be called from a macro by passing in the number of points as an additional argument. See *derivative2* for more information. To use the classic *derivative* or the other functions mentioned during the execution of a Macro, these functions might be placed within dedicated Transforms inside Setups, which the Macro can Perform via the *iccap_func* command.

An Example Macro

This section provides an example to improve your understanding of what a Macro can accomplish and how some if its features can be used. This example (code listing below) benchmarks the quality of the DC extraction in the *pn_diode.mdl* file. Load this file to view and run the example. Before running this example, you must define TNOM in your System Variables Table.

After performing *Read Model* you should be able to edit the *diode* model. By selecting *Read Macro* from the menu associated with its list of Macros, you can load *pn_test.mac,* so you don't have to enter the text appearing below. After loading the Macro file, perform the *Edit* menu choice for the newly loaded Macro *test_extract,* Execute the Macro by selecting its name and clicking on *Execute.*

The Macro code is included below. The comments within the Macro should clarify what the Macro is doing.

```
! This macro will perturb model parameters,
! automatically call an extraction,
! and then compare the extracted parameters
! to the original, "perturbed" parameters
! perturb model parameters
IS = IS * 0.5
N   = N * 1.2
RS  = RS * 2.1
! save perturbed model parameter values
is1 = IS
n1  = N
rs1 = RS
! simulate with perturbed parameters and
! then compute and print rms error between
! meas. & sim. current
iccap_func("/diode/dc/idvd", "simulate")
error = RMSerror(/diode/dc/idvd/ia.m,/diode/dc/idvd/ia.s,0)
print "RMS error before extraction: " ; error
! call extraction to clean up parameters
iccap_func("/diode/dc", "extract")
! display plot if user desires
```

Figure 40 Example Macro

12
Creating Graphic User Interfaces

The **Model GUI Items** tab accesses a page for adding, customizing, testing and diagnosing the widgets in a GUI.

The widgets include various types of selection boxes, with single or multiple selection, read only and editable text displays, spin boxes, sliders, and standard and radio type push buttons.

The IC-CAP Studio provides an exchange of data between PEL Macros/Transforms, Variable Table entries and the GUI. GUI items can perform IC-CAP automation including execution of PEL Macros/Transforms, which can access the state of the GUI Items stored in variables in the variable table. In turn these macros/transforms can launch other GUI dialogs to create the use model you wish.

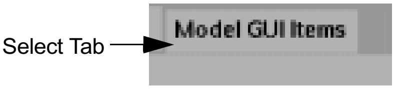
The placement of widgets is typically organized in row type place holders (known as Horizontal Tables) nested within column type place holders (known as Vertical Tables). When a caption is added to a table, a pleasing bounding box is drawn as a border around the items in it, and the caption is inset within the border.

Basic Commands - The Model GUI Items Menu

First the Model GUI Items page of IC-CAP must be displayed.

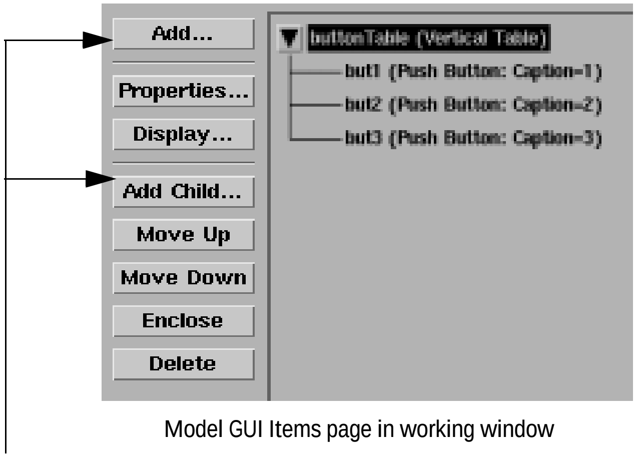
This done by adding this line to the iccap.cfg file:
`ICCAP_VIEW_GUI_PAGES=1`

This will cause the Model GUI Items tab to appear in the working window. Select this tab.



The Model GUI Items page is used for building and testing GUI's. Once built GUI's could conceivably be used from this page. However it is not intend to be used that way. It is more convenient to have macros launch the appropriate GUI's as an operation proceeds.

The Model GUI Items page has a menu at the left as shown below.


Add...
Properties...
Display...
Add Child...
Move Up
Move Down
Enclose
Delete
buttonTable (Vertical Table)
but1 (Push Button: Caption=1)
but2 (Push Button: Caption=2)
but3 (Push Button: Caption=3)

Model GUI Items page in working window

These buttons
create widgets

The commands available in the left window operate on whatever object is selected in the main window. The only exception is the Add... command which adds the parent widget to a new GUI. Any object may be selected by clicking on it, and only one object remains selected (single item selection).

Add...

The Add... button is used to add the first item of a GUI, usually a Table which acts as an organizational tool to layout the children GUI Items to be added later. See “More on Tables (Examples)” on page 536.

Add Child...

The Add Child... button, through the Widget Properties dialog box, adds children to whatever is selected in the main window. In the example above the Vertical Table is selected and three buttons are added to it as children. When children are added to a widget a black triangle (the node symbol) appears to the left of it. Clicking on the node symbol causes it to point downwards to a list of the children.

Properties

The Properties button launches the Widget Properties dialog box for displaying and customizing widget. This box is described in detail in the Widget Properties Dialog Box Section. The dialog box will also be displayed if you double-click on an item in the window.

Display

The Display button will create and display a working version of whatever widget is selected in the window.

Move Up / Move Down

The Move Up and Move Down buttons move whatever is selected. For example if the third button but3 is selected it can be moved up and down the list of buttons. The position in the layout will change accordingly. Thus the children can be rearranged in any order.

Delete

The Delete button deletes whatever is selected in the window. For example we could select and delete but3 if we wanted only two buttons. If we select and delete the Vertical Table we will also delete all of the children. Caution: There is no Undo.

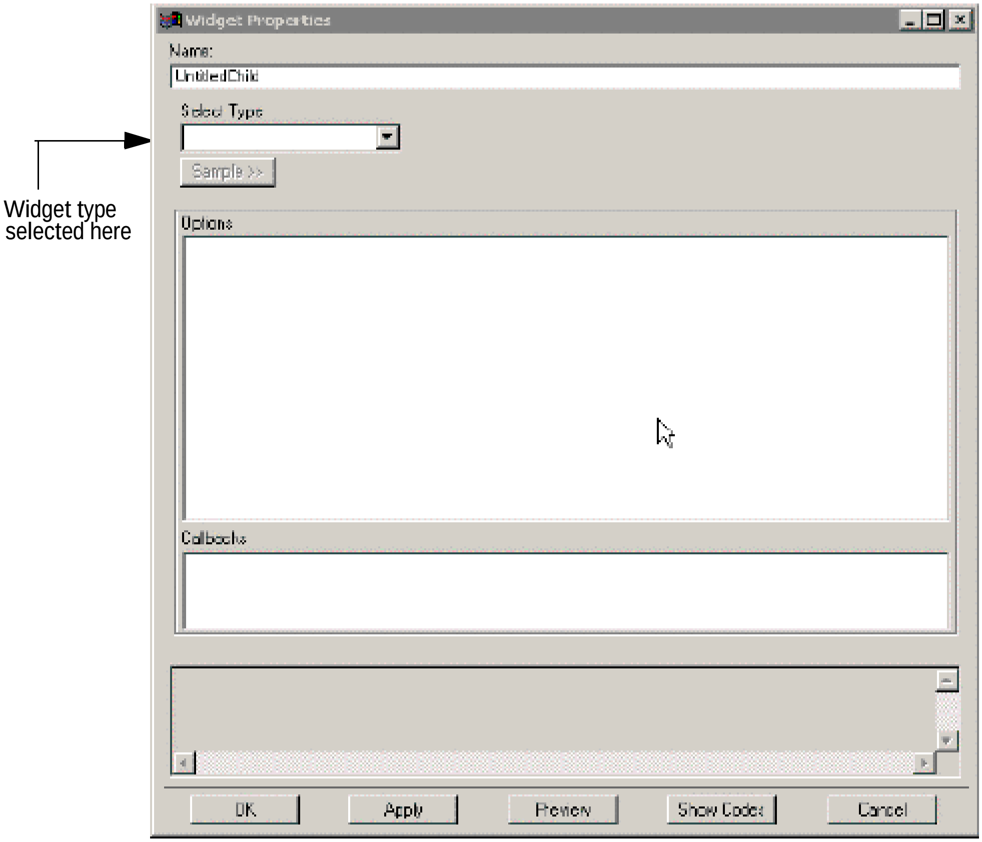
Enclose

The Enclose command allows you add an additional level of hierarchy into the GUI structure. For example a Vertical Table contains three push buttons. We want to add three more as a second stack of three to the right of the first three. We select the Table and click Enclose. The Table now has a new parent with the default name NewTable. We select NewTable and change its Orientation to Horizontal. Then we add to it as a child a further Vertical Table which will contain the second three Push Buttons.

See the tutorial “Enclose” on page 543 for more details.

The Widget Properties Dialog Box

The dialog box is shown below



Name:
UntitledChild

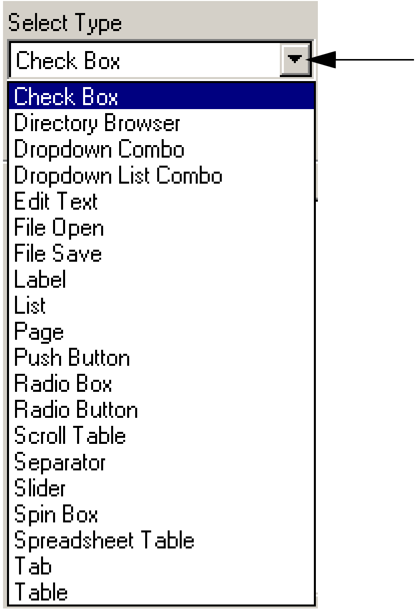
Widget type selected here

The Widget Properties Dialog Box

The Widget Properties Dialog Box is launched

- when the Add... or Add Child... buttons are selected.
- when the Properties button is selected.
- when a widget in the right window is double-clicked.

The top level item in this box is the Select Type box. Press the activate button on the right side of it. A dropdown menu appears offering sixteen choices of widgets.



Click here
to show list
of GUI items

Selecting the Widget Type

Widget Classes

The available widgets perform four general functions: organizing placement, handling single variables, and handling lists.

Placement Widgets

The Table (of anything) and the Tabbed Folder (of pages) organize layout of widgets within a GUI.

Single Variable Widgets

The Spin Box and the Slider increment and decrement variables declared with the Current Value and Real Value options respectively.

The Edit Box can be used for both input and output, while the Label is like a read only Edit Box. Labels can be large blocks of text including many carriage returns. They are used to hold the text on Pages.

Multiple Variable (List) Widgets

Notice that the Dropdown Combo, dropdown List Combo, List, Radio Box, and Spreadsheet are all different styles of list from which the user may select one or more values. The Option Items allows you to declare an ICCAP_ARRAY that stores the values that are displayed. The option Selected Index allows us to declare an integer which is an index into the selected member of the array. This can then be read by macros and programs. The is unique in that it uses a two-dimensional array as its primary list.

Widget Naming

Once the widget type has been chosen it can be named in the top box of the Widget Properties dialog box. This is not essential but it is very desirable since it makes the widget map (as distinct from the GUI itself) much more intelligible.

NOTE

Duplicate naming of siblings is prevented.

The top level name of in a GUI becomes the name of the GUI which a macro uses to execute it. Lower level GUI’s are called with a hierarchical a.b.c convention.

Using the Properties Dialog Box

Once the widget type has been selected and named it is mandatory that it be assigned a variable to modify and/or a function to call if the widget is to handle input or output.

Once a widget type has been selected, the left side of the properties box shows a list of the Options and a list of Callbacks available for this widget.

Select, for example, the option Caption.

- The information box at the bottom will show relevant information.
- The title of the right side editing area changes to Editing: Caption.

For a fixed caption:

1 Click on the box titled **Please Enter Text Value** and enter the text for the caption, e.g “Start”. Do not use the enter key unless you are entering multi-line captions.

2 Click on the **OK** or **Apply** button immediately below. In the Options list Caption will be shown as Caption (Start).

For a variable Caption:

1 In the Editing: Caption box set Track Variable to on (so that an x shows in the box).

2 Click on the box titled **Variable** and enter the variable name, e.g. buttonName.

3 Then click on the **OK** or **Apply** button immediately below. In the Options list Caption will be shown as Caption (Variable buttonName)**.**

4 The variable buttonName must be declared in a variable table, e.g. the Model Variable table accessed by clicking on the **Model Variables** tab.

For the GUI to accept the information from the Widget Properties dialog box, click on either the **OK** or the **Apply** buttons at the bottom left corner. OK dismisses the dialog box, while Apply leaves it in place.

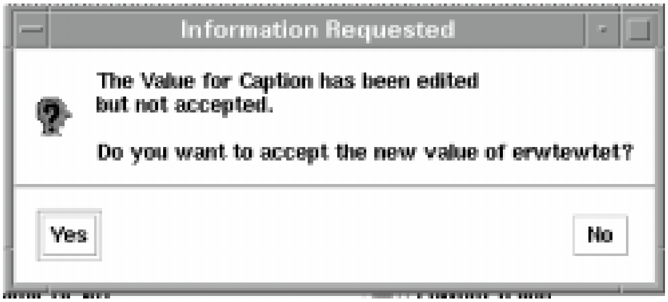
Troubleshooting

Problem: Widget Properties dialog the Editing box will not accept text, even though an insertion point marker is visible.

Solution: There is usually more than one box so you must click on one to select it for text entry.

Problem: Option not found in the Options list. For most GUI Items there are not enough options to fill the option box. When there are enough options to overflow the box a scroll bar appears at the right hand side. Use this to access the options at the end of the list.

Problem: The following Dialog Box appears:



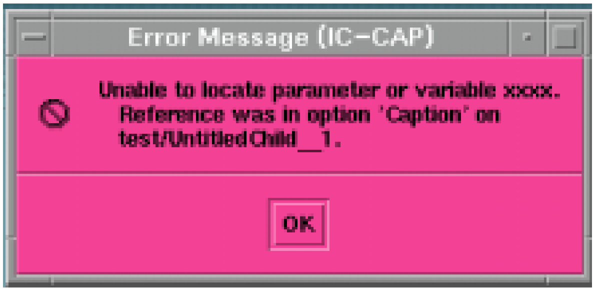
Information Requested

The Value for Caption has been edited but not accepted.

Do you want to accept the new value of erwtewtet?

Go back to the Properties dialog and click on **OK** or **Apply** under the Editing area.

Problem: The following Dialog Box appears:



Error Message (IC-CAP)

Unable to locate parameter or variable xxxx. Reference was in option 'Caption' on test/UntitledChild__1.

This means that a variable xxxx has been set to be tracked but has not been decalred in a table.

Solution: Add xxxx to, for example, the Model Variables table. Another possibility is that the variable name was misspelled.

Problem: After executing a macro the ‘watch’ symbol remains in the Model window, preventing any further commands from being accepted. This is probably because the macro launched a GUI (maybe hidden under another window) that is waiting for input or waiting to be closed.

Solution:

- Enter some input or close the GUI.
- Another possibility is that a macro is running in an infinite loop. To kill it, click on the “Stop” sign in the status window.

GUI Item Options Controlling Variables

Once the widget type has been chosen the left side of the dialog box lists the Options available for that widget. The Caption option should usually be set because it appears on the widget and adds to the intelligibility of the GUI.

There are only a few Options vital to the functionality. These options control data, whereas the rest affect only the appearance and cosmetics of the GUI.

The Options which allow a variable to track and store data are shown in the following table which lists the name of the option and the type of variable it controls.

Modifying Variables

- All variables and arrays must be declared and defined, typically in the Model Variables Table, and declared in the Widget Properties box.
- Some options can only track an iccap_variable in the form of an ICCAP_ARRAY. In this case a single blank will be presented with the instruction to **Please enter the name of an ICCAP_ARRAY**. Other options may be ‘hardcoded’ with a literal, or can be tied to a variable. For the later case, please check the box labeled Track Variable and fill in the Variable Name blank.

Options that can be tied to Variables:

Widget	Option Name	Value Type
Check Box	Toggle State	Integer: 1 or 0
Dropdown Combo	Entered Text	String
	Selected Index	Integer
	Typed Text	String
	Visible Item Count	Integer
Dropdown List Combo	Selected Index	Integer
	Visible Item Count	Integer
Edit Text	Editable	Integer: 1 or 0
	Field Value	String (editable)
Label	<none>	
List	Selected Index	Integer
	Visible Item Count	Integer
Page	< none >	
Push Button	< none >	
Radio Box	Selected Index	Integer
Radio Button	Toggle State	Integer: 1 or 0
Scroll Table	Margin Height	Integer
	Margin Width	Integer
Separator	<none>	
Slider	Current Position	Integer
Spin Box	Real Value	Real
Spreadsheet Table	<none>	
Tab	Current Page	Integer
	Margin Height	Integer

Widget	Option Name	Value Type
	Margin Width	Integer
	Page Change OK	Integer: 1 or 0
Table	<none>	

The Callback section of the Widget Properties dialog box list the conditions under which a Macro or Transform will be executed. These possible conditions, depending on the GUI item, are:

Callback	Triggered by:
< Enter > Pressed	The Enter key was pressed.
Button Pressed	Button down detected.
Combo Pulled Down	A menu is visible.
Exit Function	Top level widget was closed.
Focus Entered	Cursor is in the GUI item area.
Focus Lost	Cursor has left the GUI area.
Item Double Clicked	An item in a list was double clicked.
Item Selected	An item selected from a menu or list.
Page Changed	New Page number.
Page Validation	Attempt to change a page.
Radio Changed	A Radio Button was pressed.
Slider Moved	Slider stopped moving.
Slider Moving	Slider still moving.
Spin Box Decreased	Value decreasing.
Spin Box Increased	Value increasing.
Spin Field Modified	Value changed.
Toggle State Changed	A Check Box state changed.

Callback	Triggered by:
Variable Changed	A tracked variable changed.

By selecting a callback condition, e.g., for the Push Button widget we select the ButtonPressed condition in the Callbacks box. A sub dialog appears on the right prompting for two arguments to iccap_func. The arguments are the *object* to operate on and the *action* to take.

These could be a macro name (e.g., countPresses for the item to act on, and the second argument could be Execute for the action to take.

This would cause the Button Pressed condition in the Callbacks box to be shown as:

```
Button Pressed (iccap_func(“countPresses”,”Execute”))
```

Common Options

These options are common to most widgets, so they are described in this section. Widget specific options are in individual tables in each GUI item description.

Option	Description
Background Color	Specifies the Background Color for the GUI Item.
Caption	Optional on all widgets, but doesn't make sense to leave blank for Buttons, Labels or Pages. For all other widgets if will appear above them and left justified with them. For Buttons, the caption text appears directly on the button, thus identifying it and distinguishing it from other buttons. For Pages, it is the text on the tab. For Tables, it is the header text (optional). For Edit fields it is the text describing the field (not the text in the field). For Labels, it is the text itself as it appears on a page or table. Use the Sample button on the Properties dialog to see how text appears on an item.
EditableText	The Option Editable may be set to TRUE or FALSE. When TRUE it allows the text insertion point to be placed between text characters so that they may be edited.
Foreground Color	Specifies the Foreground Color for the GUI Item.
Font	Indicates the type of font to use.
Height and Width	These are the widget dimensions in pixels. Note that these options are only used if applied to a widget which is actually displayed. If the options are set on children of the widget being displayed, they will be ignored.
Items	Indicates the names of the items (text strings) contained in the widget such as a Dropdown List, List or Spin Box. Specify the name of a variable in a variable table. It is expected that this variable will have a value of the format: ICCAP_ARRAY[x] where x is a number specifying the length of the array.

Option	Description
Managed	This option indicates whether the item is managed (shown) or unmanaged (not shown). This allows a widget to be removed from the GUI at times when it is not needed, and rapidly restored when needed without the rebuilding time. In the sequential display of pages it is used to make only the active page visible (see tutorial on sequential display). Most useful when Track Variable is selected. Default is 1.
Orientation	Can be Vertical or Horizontal. Default depends on widget type. See tutorial "Captions, Frames and Sizing" on page 536.
Row and Column Span	Indicates how many rows or columns this table should span in a table. Note, a dummy table must be placed at the cell(s) of the table that will be overlapped.
Sensitivity	When sensitivity is set to 0 the widget is non-responsive and appears greyed out to indicate that it is not active. When sensitivity is 1 the widget has normal appearance and response. This option is useful if Track Variable is selected. PEL code can then set the tracked variable to 0 or 1 for a dynamic look to your GUI.
Sizing Options	These indicate how the widget should behave in a Table when you expand the GUI window. See tutorial "Captions, Frames and Sizing" on page 536.

Unmanage a Dialog

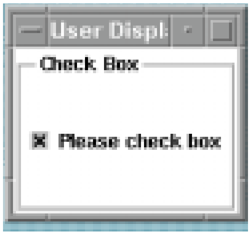
Dialogs can be removed from the screen without being really destroyed. Some dialogs are somewhat time consuming to build. If they are unmanaged instead of destroyed, the next time they are displayed, they will not need to be rebuilt and will appear quickly.

```
iccap_func(mydialog, Close GUI, DialogName)
iccap_func(mydialog,Close Single GUI)
```

Widget Descriptions

This section describes the widgets that are listed in the Select Type dropdown box in the Widget Properties dialog box.

Check Box



The Check Box widget is actually a toggle box. It has a Toggle State of 1 or 0.

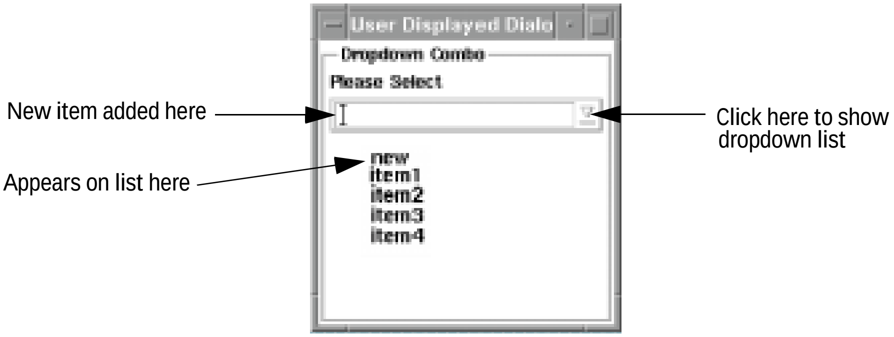
To assign a variable, such as checkVal to a variable it must be declared in a variables table. For example, the Model Variables Table is accessed through the Model Variables tab. Each time you click in the box, checkVal will toggle between the binary values 1 and 0.

After selecting **Check Box** in the Widget Properties dialog box, select **Toggle State** from the Options list. This can be set to Toggle On or Toggle Off. Then select **Track Variable** and type `checkVal` into the Variable box.

Special Options	Description
Toggle State	Toggles a value between 1 and 0 each time the box is checked.

See "Using the Properties Dialog Box" on page 484.

Dropdown Combo



The Dropdown Combo widget enables you to choose from a large dropdown list, which disappears after use to save space.

The Dropdown Combo is editable in that the user can add items by typing them in the top editable box as well as choose them from a list. When Enter is pressed on the keyboard the new item is added to the displayed list. The widget can also be configured to automatically add new entries to the list.

The items displayed in the dropdown list are declared, for example, in an array in the Model Variables table as shown below. listElems is declared as an ICCAP_ARRAY of size 4, and the four values are defined individually for each element of the array.

After selecting **Dropdown Combo** in the Widget Properties dialog box, select the option **Items**. Then click the box captioned **Please enter the name of an ICCAP_ARRAY** and type `listElems`. Click both **OK** buttons.

To return a value from the dropdown list, select the option **Selected Index**. Click on **Track Variable** and then in the Variable box enter, for example, `index`. The item selected will be listElems[index]. The variable name index, of course, must be declared, for example in the Model Variables table.

To set the Dropdown Combo to be user extendable, select the option **Add Entry on <Enter>**

The dialog box will show **<none specified>**

Click on the activate button at the right of the New Value box and select **TRUE.**

Click on **OK** in the box below it, and then **OK** at the bottom left.

The dropdown Combo will now have a cursor indicating that it is editable.

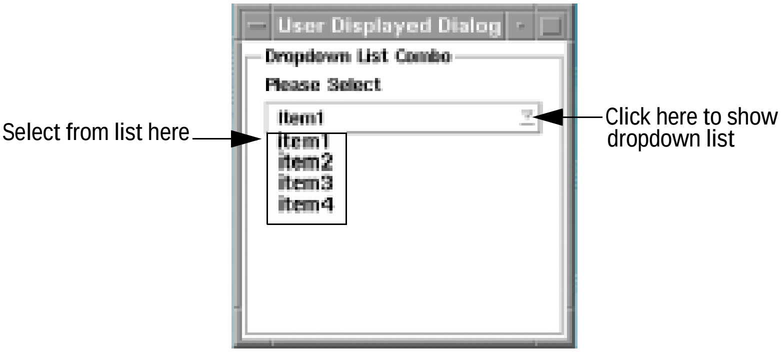
Display the Dropdown Combo and edit the contents of the edit field, then press Enter. Pressing Enter causes the contents of the edit field window to be added to the list of items.

Next time you press on the activate button, the last new item added will be shown at the top of the list.

Special Options	Description
Add Entry on <Enter>	Allows you to add items to the list.
Entered Text	This is the text in the edit field. If tied to a variable, the variable will only update when Enter is pressed.
Scroll Bar Configurations	Indicates which scrollbars to use: Vertical, Horizontal, Vertical & Horizontal, or no scroll bars.
Selected Index	Index into the selected member of the ICCAP_ARAY.
Typed Text	This is the text in the edit field. If tied to a variable, it will always reflect the current value.
Visible Columns	The integer value for the number of columns that are visible at once.
Visible Item Count	The integer value for the number of items that are visible at once.

See “Using the Properties Dialog Box” on page 484.

Dropdown List Combo



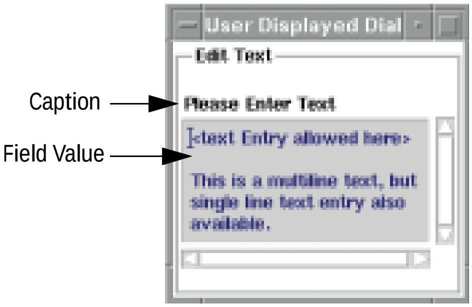
The Dropdown List Combo is the same as the Dropdown Combo except that the users must select entries from the list, and cannot type their entries by hand.

See previous item: "Dropdown Combo" on page 495.

Special Options	Description
Scroll Bar Configurations	Indicates which scrollbars to use: Vertical, Horizontal, Vertical & Horizontal, or no scroll bars.
Selected Index	Index into the selected member of the ICCAP_ARAY.
Visible Columns	The integer value for the number of columns that are visible at once.
Visible Item Count	The integer value for the number of items that are visible at once.

See "Using the Properties Dialog Box" on page 484.

Edit Text



<text Entry allowed here>

This is a multiline text, but single line text entry also available.

The Edit Text box serves primarily as an input where the user can enter and change the value of numbers or strings of text. However it also operates as a monitor in that will show the new values of a variable or text string any time that variable changes.

Fixed Label

To show a fixed (hard coded) title, in the Widget Properties dialog box, select **Caption** from the Options list. Then select the **Track Variable** box to set it to off (no x shown in it). Then in the box titled Please Enter Text Value type, for example, `Please Enter Text` and click **OK**. The caption in the options list will now be displayed as

Caption (Please Enter Text)

Click **OK** and display the edit box and its title.

In the Widget Properties dialog box, select **Field Value** from the Options list. Then select **Track Variable** setting it to on (so that an x is shown in it) and then type, for example, `newText` into the Variable box. Click the **OK** button directly underneath it.

In the Options list the Caption will now be shown as:

Field Value (newText)

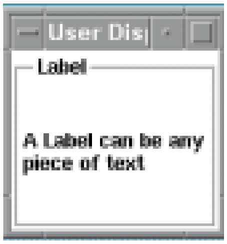
Then click the bottom left **OK** button.

To display the value of the variable newText it must be declared otherwise there will be an error message when the GUI is run saying **Unable to locate parameter or variable newText...** For demonstration purposes click the **Model Variables** tab and scroll down to the empty cells at the bottom of the table and add `newText` in the Name column.

Special Options	Description
Columns	The integer value for the number of columns shown.
Edit Mode	Controls if the field can have multiple lines or just a single line.
Editable	Controls whether the edit field can be modified by the user.
Field Value	This is the text in the edit field
Rows	If Edit Mode is *Multiple Lines*, controls the number of lines displayed.

See “Using the Properties Dialog Box” on page 484.

Label



A Label can be any piece of text

Fixed Label

The text is normally a fixed string of characters and is contained in the Caption.

It can be single or multi-line.

To show a fixed (hard coded) label, in the Widget Properties dialog box, after selecting **Label**, select **Caption** from the Options list. Then in the box titled Please Enter Text Value type, for example, `A Label can be any piece of text` and click **OK**. The caption option in the option in the list will now be displayed as

Caption (A Label can be any piece of text)

The displayed label will appear as shown above.

Variable Label

A variable containing the text can be generated by a macro which generates the text message or selects it from a list.

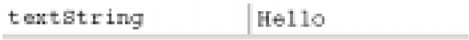
Select **Track Variable** and then type, for example, `textString` into the Variable box. Click the **OK** button directly underneath it.

The caption option in the option in the list will now be displayed as

Caption (Variable textString)

Then click the bottom left **OK** button.

To display the message in textString it must be given a value. For demonstration purposes click the **Model Variables** tab and scroll down to the empty cells at the bottom of the table and add `textString` in the Name column, and `"Hello"` in the Value column.



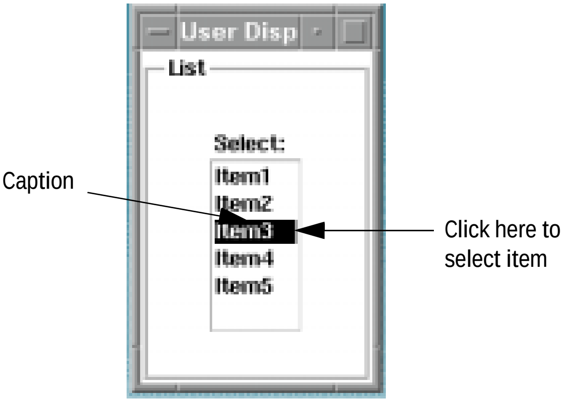
textString	Hello

Display the Label with its text “Hello” then select the **Model Variables** tab and edit the string “Hello” to `"Goodbye"`. When you press Enter you will see the text in the widget change.

Special Options	Description
None	

See “Using the Properties Dialog Box” on page 484.

List



Caption

Click here to select item

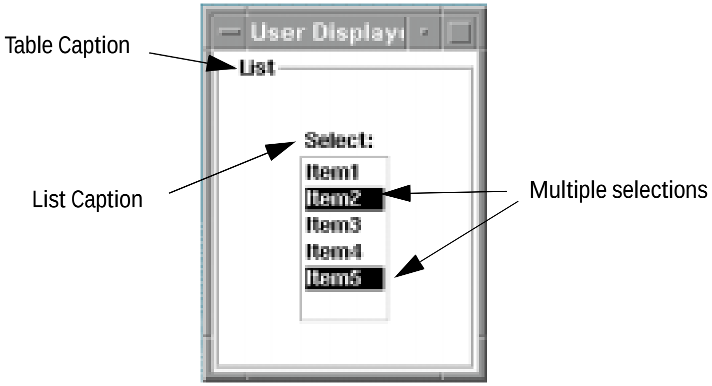
A List widget displays a permanent list of items which may be selected with the mouse. When an item is selected it remains selected after the mouse button is released.

The List may set up for single selection (like the Radio Buttons) or multiple selection where any number of items may be selected. In the Select Multiple Items mode clicking on selected items in the list will unselect them.

The List remains on display to show the selections.

To return a selection from a list select the option **Selected Index**. Click on **Track Variable** and then in the Variable box enter, for example, `index`.

The item selected will be listElems[index]. The variable name index, of course, must be declared, for example in the Model Variables table.



The Selection Policy has four options:

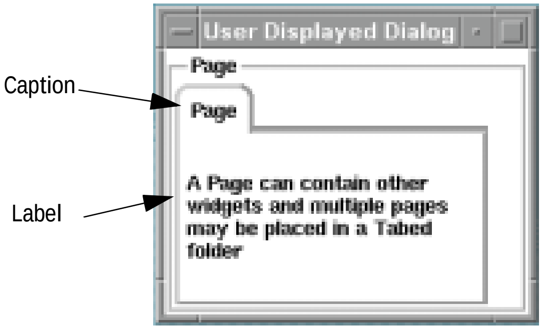
Selection Mode	Description
Select Multiple Items	Requires the option Selected Indices (plural) to declare the name of an ICCAP_ARRAY which stores the list of selected indices. Clicking on a selected item unselects it and has no effect on other selected items.
Single Item Select	Uses the Option Selected Index (singular) to the store the selection. Clicking on a selected item unselects it.
Browse Style Selection	This is the same as the Select Single Item mode except that clicking on a selected item has no effect. After the first selection one item always remains selected.
Extended Style Selection	This is the same as the Browse Style Selection mode except that you may select multiple items by using the <Shift> and <Ctrl> keys while clicking.

The Option specific to the List GUI item are:

Special Options	Description
Scroll Bar Configuration	Indicates which scrollbars to use: Vertical, Horizontal, Vertical & Horizontal, or no scroll bars.
Selected Index	Index into the selected member of the ICCAP_ARAY.
Selected Indices	Indices into the selected members of the ICCAP_ARAY.
Selection Policy	The type of selection policy to be used: Select Single Item, Select Multiple Items, Browse Style Selection, Extended Style Selection.
Visible Columns	The integer value for the number of columns that are visible at once.
Visible Item Count	The integer value for the number of items visible at once.

See “Using the Properties Dialog Box” on page 484.

Page



A Page can contain other widgets and multiple pages may be placed in a Tabed folder

Pages are used in a Tabbed Folder. They are like tables whose caption shows up on a tab.

A Tabbed Folder is a stack of overlapping pages with non-overlapping tabs on each page. When a tab is selected its page comes to the top of the stack. See the Tab description.

A pages are added to a Tabbed Folder as a children by selecting the Tab item and then **Add Child...**

From the Widget Properties dialog select:

Page > Option > Caption

is used to set the caption on the tab of each page.

On each page you can add text (as a child) as a Label or as an Edit Text Box

On each page you can also add any combination of widgets.

NOTE

On UNIX platforms, some widgets on Tabbed-folder pages may not appear properly. To avoid this potential problem, place your content within a single Table widget (it can have 0 margin width and 0 margin height) inside the page. For example, instead of placing 4 widgets on a page, place a single table on a page, then place the 4 widgets inside the table.

On Window platforms, you should also place your content within a single Table widget inside the page just in case the *mdl* is ever used on a UNIX platform.

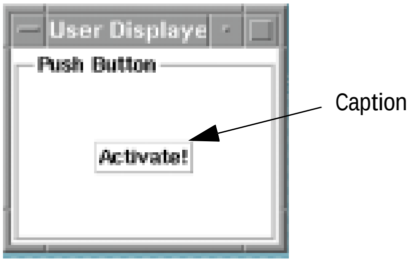
The Option Allow Resize allows a page to grow dynamically with the text strings in the Labels in it. This allows arbitrary text strings generated or selected by a macro to be displayed.

See "Using the Properties Dialog Box" on page 484.

Special Options	Description
Allow Resize	Permits the page to dynamically resize to fit the string(s) of text.
Column Spacing	The integer value for the number of pixels between columns in a table.
Default Sizing Options for children	The sizing options for all members of the table if they don't declare their own sizing options.
Equalize All	When TRUE all widgets in a table are normalized to the same size. The default is FALSE.
Frame Visible	Indicates if a frame is to be visible or not. The default is FALSE.

Special Options	Description
Margin Height	The integer value for the number of pixels of margin at the top of the page.
Margin Width	The integer value for the number of pixels of margin at the sides of the page.
Number Rows/Columns	The integer value for the number of rows displayed for a horizontal orientation or the number of columns displayed for a vertical orientation. The children will be divided up evenly between the rows/columns.
Row Spacing	The integer value for the number of pixels between rows of a table.
Use Radio Behavior	When this is TRUE only one radio button in the table may be selected at a time.

Push Button



A Push Button is only active when while the cursor is over it and while the mouse button is depressed. Unlike the Check Box widget, the Push Button widget has no memory of being selected, and does not control any variable.

It operates by performing a callback, such as Button Pressed, whenever the button is pressed.

Typically, you will want to set Fix Width and Fix Height under Sizing options for buttons.

Button Caption

To show a fixed (hard coded) name for the button, in the Widget Properties dialog box, select Caption from the Options list. Then in the box titled Please Enter Text Value type, for example, Activate! and click **OK**. The caption in the options list will now be displayed as:

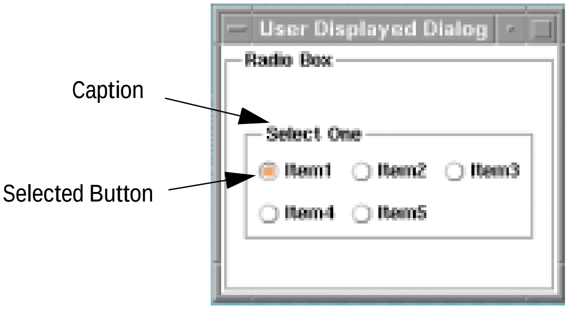
Caption (Activate!)

and the button will appear as shown above.

Special Options	Description
None	

See “Using the Properties Dialog Box” on page 484.

Radio Box



The Radio Box *automatically creates* and labels Radio Buttons according to the text strings stored in the array identified by the Entries Option.

Radio Buttons act as a group with all other radio buttons in a given Table. A group of radio buttons in a table is very similar to the Radio Box, but placing radio buttons in a table gives you more flexibility in the button placement. Radio buttons in a table can also provide a ‘multiple selection’ radio box. The Option Radio Box Behavior on the parent table sets this mode. If Radio Button Behavior is true, all radio button's siblings in a table will act just like radio box (single selection). If Radio Button Behavior is FALSE the buttons behave like Check Boxes and allow multiple selections as show above right.

The items selected by a group of radio buttons are stored in an ICCAP_ARRAY.

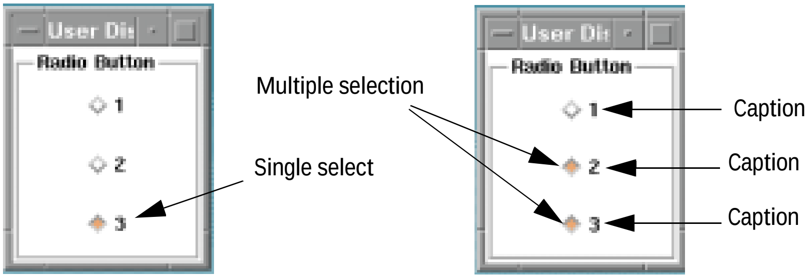
The Option Selected Index allows you to assign a variable which takes the index value of the selected item n the ICCAP_ARRAY.

The array and index variables must be declared and defined in a variables table, for example in the Model Variables Table.

See "Using the Properties Dialog Box" on page 484.

Special Options	Description
Desensitized Indices	This is a list of 1's or 0's in an ICCAP_ARRAY to indicate which buttons in a Radio Box are insensitive.
Number Rows/Columns	The integer value for the number of rows displayed for a horizontal orientation or the number of columns displayed for a vertical orientation. The children will be divided up evenly between the rows/columns.
Orientation	Indicates the orientation of the items: Horizontal or Vertical.
Selected Index	The index number of the selected item.

Radio Button



Radio buttons are always used in groups.

The default behavior when placed in a Table is Radio Button Behavior in which only one button can be selected at a time. When an unselected button is selected the previously selected button automatically becomes unselected. The selected one remains visually different to show the last selection, as shown above left.

When the Table Option Radio Button Behavior is set to FALSE the buttons behave like Check boxes and allow multiple sections as shown above right.

Special Options	Description
Toggle State	Toggles a value between 1 and 0 each time the box is checked.

NOTE

The Radio Box widget automatically creates its own buttons which are ROUND. The Radio Button widget is a DIAMOND shaped button.

See “Using the Properties Dialog Box” on page 484.

Scroll Table

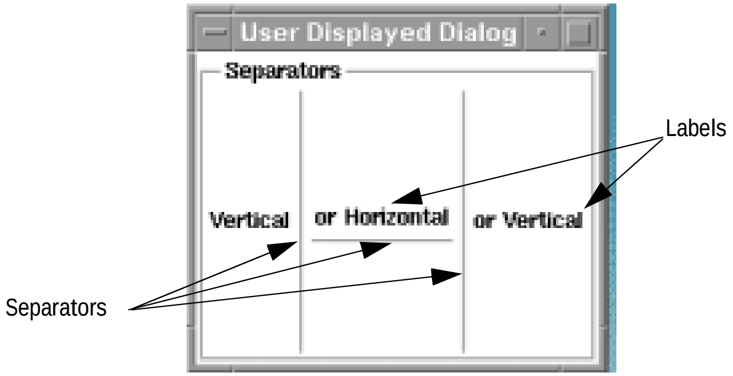
Tables are used to organize the placement of widgets into a hierarchy of rows and columns, and are usually the first item to be defined in a GUI. The default orientation is vertical.

When widgets are added as children to a vertical table they will be placed one above the other in the same order as listed in the hierarchy in the Model GUI Items page. Horizontal tables may also be added as children to vertical tables.

Scroll tables are like normal tables except the table requires scrollbars to access table entries beyond the values set by Columns Visible and Visible Rows unless the rest of the dialog can accommodate more space. Setting *Columns Visible* and *Visible Rows* to 0 allows the table to shrink smaller than the first entry in the table and scrollbars will be shown to see the entire first entry.

Special Options	Description
Allow Resize	Permits items in the table to dynamically resize themselves and force the entire dialog to resize to accommodate the change.
Column Spacing	The integer value for the number of pixels between columns in a table.
Columns Visible	The integer value for the number of columns of widgets to show before scrollbars appear.
Default Sizing Options for children	The sizing options for all members of the table if they don't declare their own sizing options.
Equalize All	When TRUE all widgets in a table are normalized to the same size. The default is FALSE.
Frame Visible	Indicates if a frame is to be visible or not. The default is FALSE.
Margin Height	The integer value for the number of pixels of margin at the top of the table.
Margin Width	The integer value for the number of pixels of margin at the sides of the table.
Number Rows/Columns	The integer value for the number of rows displayed for a horizontal orientation or the number of columns displayed for a vertical orientation. The children will be divided up evenly between the rows/columns.
Row Spacing	The integer value for the number of pixels between rows of a table.
Use Radio Behavior	When this is TRUE only one radio button in the table may be selected at a time.
Visible Rows	The integer value for the number of rows of widgets to show before scrollbars appear.

Separator



Separators are purely for cosmetic use and do not handle data.

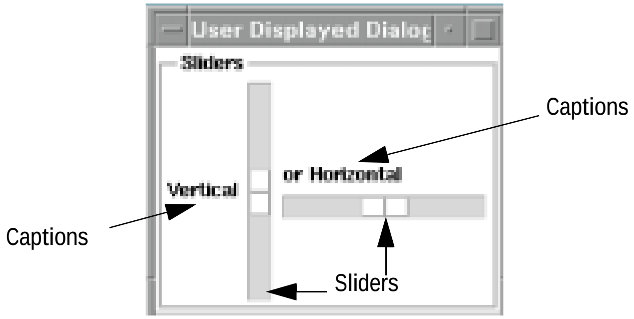
As a GUI develops and contains more and more widgets the use of horizontal and/or vertical separators helps to keep the widgets and associated text visually organized.

Their positions are determined by the position in the list of siblings.

Special Options	Description
Length	The integer value for the length (in pixels) of the separator.

See "Using the Properties Dialog Box" on page 484.

Slider



The Slider is an input in which the slider button is selected and dragged by holding down the left mouse button while it is moved. It has the advantage avoiding typing, and it is faster than the spin box.

The Option Current Value specifies the slider position as a integer, and a range may be established with the Minimum Value and Maximum Value options. The default range is 0 to 100.

There are three callbacks.

Slider Moved—Name the Macro/Transform to be executed when the button is released after a slider has been moved.

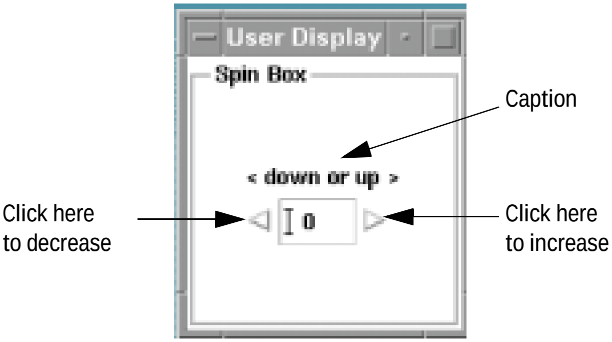
Slider Moving—Name the Macro/Transform to be executed while the slider is being moved via the UI.

Variable Changed—Name the Macro/Transform to be executed whenever the specified variable/parameter changes.

See “Using the Properties Dialog Box” on page 484.

Special Options	Description
Current Position	The current position of the slider
Maximum Value	The maximum allowed value of the position
Minimum Value	The minimum allowed value of the position

Spin Box



See “Using the Properties Dialog Box” on page 484.

The Spin Box allows you to change a value by clicking on the increment or decrement buttons. The advantage is that no typing is required. A minimum and maximum value can be set so as to keep the number in a useful (predetermined) range.

Scroll down the Options list and select **Spin Type**.

The choices are Spin Integer (default), Spin Real, Spin Text.

Integer Values

The default Spin Box controls an integer in increments of 1, with minimum and maximum values of 0 and 100.

The Options Maximum Value, and Minimum Value are only used for Spin Integer.

For the Option Spin Position set Track Variable on and assign a variable to indicate the position.

Real Values

Set the Spin Type to **Spin Real**.

For the Option Spin Real set Track Variable on and assign a variable to indicate the position. The Option Delta Between Real Steps must be set to a finite value. It is used only for real values.

Text Values

Set the Spin Type to **Spin Text**

Select the **Option Items** and enter the name of an ICCAP_ARRAY to hold the text strings to be displayed. Items is used only for text strings.

Spin Box Wraps

When set to TRUE (the default) any attempt to increase the value past the maximum will cause it to go to the minimum value, and vice versa. When the limit is reached the appropriate arrow changes from white to grey.

When set to FALSE the value increases to the specified maximum and stays there. The arrow changes from white to gray (insensitive) to indicate the value has hit the limit.

Split Arrows

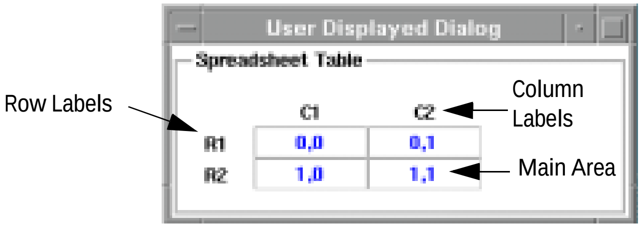
This is a styling feature. When set to TRUE the up and down arrows appear on opposite sides of the value display box. On FALSE the will both be to the right.

See “Using the Properties Dialog Box” on page 484.

Special Options	Description
Columns	The integer value for the number of columns shown.
Delta Between Real Steps	Indicates the delta between steps when the spin box is in real mode.
Editable	Controls whether the edit field can be modified by the user.
Maximum Value	Indicates the maximum value.
Minimum Value	Indicates the minimum value.
Real Value	This is the real value of a spin box when the mode is set to real numbers.

Special Options	Description
Spin Box Wraps	Indicates whether a spin box should wrap around when it hits its minimum/maximum value.
Spin Position	Indicates where the Spinbox is.
Spin Type	Items is only used for Spin Text. Maximum Value, Minimum Value are only used for Spin Integer. Real Value, Delta Between Real Steps are only used for Spin Real.
Split Arrows	Indicates whether the arrows should be split.

Spreadsheet Table



	C1	C2
R1	0,0	0,1
R2	1,0	1,1

The Spreadsheet Table is an array of rows and columns of edit fields. It is much faster and more manageable than a table with the same number of Edit Text items in it, but is more restrictive in that it is always regular in shape.

It is usually needed only for very large data in very regular structures. The children will normally be Edit Text boxes.

The Spreadsheet Table, with its 46 Options, is the most complex of the widgets. It has three main regions (see Figure above):

- Main area which is an array of cells in rows and columns.
- Space for the column labels.
- Space for the row labels.

The Spreadsheet Table displays a two-dimensional array of data from an ICCAP_ARRAY. The spreadsheet will be automatically sized. The primary index is the number of columns and the secondary index is the number of rows.

The name of the ICCAP_ARRAY is specified with **Widget Properties > Options > Entries**.

The iccap_array is actually a linear array of linear arrays.

In the Editing: Entries dialog box select the editing box and enter the name of the array of data for the cells, and click on the **OK** button directly underneath it.

The Option Label Type is typically set to Row and Column Labels. The labels for the rows and columns are set in ICCAP_ARRAY(s) which are named using the **Widget Properties > Options**. Select: **Row Label Entries** and **Column Label Entries** respectively and specify the names of ICCAP_ARRRAYS holding the row and column labels.

The Spreadsheet Table may also be used as an input device by setting the option Main Area Cells Editable to TRUE.

See “Using the Properties Dialog Box” on page 484.

Special Options	Description
Main Area Options	The 2-Dimensional Array Area.
Entries	Specifies an ICCAP_ARRAY two dimensional array of entries to show up in the spreadsheet.
Main Area Background Color	Indicates the background color that should be used in the main area of the spreadsheet.
Main Area Cell Alignment	This resource indicates how the values should be aligned within the cells of the main area of the spreadsheet.
Main Area Cells Editable	This resource indicates If the cells should be editable in the main area of the spreadsheet.
Main Area Cells Traversable	This resource indicates if focus should move with the keyboard in the main area of the spreadsheet.

Special Options	Description
Main Area Foreground Color	Indicates the foreground color that should be used in the main area of the spreadsheet.
Main Area Grid Type	Indicates if gridlines should be used in the main area of the spreadsheet.
Repaint	Specify a Variable Table Variable which when set to 0 allows you to make extensive modifications to the entries in the table without the visual changes. Resetting the variable to 1 will repaint the whole graph with the new values in place.
Resize Rule	Specifies the resize rule: No Resize, Horizontal Resize, Vertical Resize, or Horizontal and Vertical Resize. Affects every cell in the table.
Label Type	Specifies the label type. The default is No Labels. If defining Column Label Entries, set Label Type to Column Labels or Row And Column Labels. If defining Row Label Entries, set Label Type to Row Labels or Row And Column Labels.
Column Area Options	The Column Label Area.
Column Label Entries	Specifies an ICCAP_ARRAY one dimensional array of labels to show up in the column headers. If you specify Column Labels, you must also set Label Type to either Column Labels or Row And Column Labels in order for the spreadsheet to display properly.
Column Label Cell Alignment	This resource indicates how the values should be aligned within the cells of the main area of the spreadsheet.
Column Label Area Cells Traversable	This resource indicates if focus should move with the keyboard in the Column Label area of the spreadsheet
Column Label Area Row Height	Indicates the height of the spreadsheet column headers.
Column Label Background Color	Indicates the background color that should be used in the column label area of the spreadsheet.

Special Options	Description
Column Label Foreground Color	indicates the foreground color that should be used in the column label area of the spreadsheet.
Column Label Grid Type	Indicates if gridlines should be used in the column label area of the spreadsheet.
Row Area Options	The Row Label Area.
Row Label Entries	Specifies an ICCAP_ARRAY one dimensional array of labels to show up in the row headers. If you specify Row Labels, you must also set Label Type to either Row Labels or Row And Column Labels in order for the spreadsheet to display properly.
Row Label Area Cells Traversable	This resource indicates if focus should move with the keyboard in the Row Label area of the spreadsheet.
Row Label Area Column Width	Indicates the width of the spreadsheet row headers.
Row Label Background Color	Indicates the background color that should be used in the row label area of the spreadsheet.
Row Label Foreground Color	Indicates the foreground color that should be used in the row label area of the spreadsheet.
Row Label Grid Type	Indicates if gridlines should be used in the row label area of the spreadsheet.
Column by Column Options	Set each column individually.
Column Background Colors	Set to 0 for <None Specified>. Set to 1 for Default Color. Set to 2 for Black. Set to 3 for Red ... See “Color Table” on page 520.
Column Cell Alignments	This is an ICCAP_ARRAY of attribute codes, one for each column. Press 'Show Codes' and see 'Main Area Cell Alignment' for valid entries. It is used to set the alignment type attribute individually for each column.

Special Options	Description
Column Grid Types	Set to 1 for No Grid. Set to 2 for Horizontal Grid. Set to 3 for Vertical Grid. Set to 4 for Horizontal And Vertical Grid.
Columns Editable	This has values set to -1, 0, or 1 which indicate whether the corresponding column of the spreadsheet should be editable. -1 indicates that the cell should be whatever is set by Main Area Cells Editable, 0 means not editable, and 1 means editable.
Column Foreground Colors	This is an ICCAP_ARRAY of attributes, one for each column. It is used to set the foreground color individually for each column. Note, indicating <none specified> means use what is in 'Main Area Foreground Color' if specified. Press 'Show Codes' and see 'Main Area Foreground Color' for valid values for the array entries.
Columns Traversable	This is an IC-CAP Array that of values set to 0 or 1 which indicates whether the corresponding column of the spreadsheet is to be traversable or not.
Column Width	This is an ICCAP_ARRAY-- one element for each column. Each entry determines the width of that column of the cell. Each entry should be an integer between 1 and 200. These values will not be changed if the user resizes a column.
Row by Row Options	Set each row individually.
Row Area Cell Alignment	This resource indicates how the values should be aligned within the cells of the main area of the spreadsheet.
Row Background Colors	Set to 0 for <None Specified>. Set to 1 for Default Color. Set to 2 for Black. Set to 3 for Red ... See “Color Table” on page 520.

Special Options	Description
Row Cell Alignments	Set to 0 for <None Specified>. Set to 1 for Left Alignment. Set to 2 for Right Alignment. Set to 3 for Center Alignment.
Row Foreground Colors	Set to 0 for <None Specified>. Set to 1 for Default Color. Set to 2 for Black. Set to 3 for Red ... See “Color Table” on page 520.
Row Grid Types	Set to 1 for No Grid. Set to 2 for Horizontal Grid. Set to 3 for Vertical Grid. Set to 4 for Horizontal And Vertical Grid.
Row Height	This is an ICCAP_ARRAY-- one element for each row. Each entry determines the height of that row of the cell. Each entry should be an integer between 1 and 20. These values will not change if the user resizes a row.
Row Span	Indicates how many rows this widget should span in a table. Note, a dummy table must be placed at the cell(s) of the table that will be overlapped.
Rows Editable	This has values set to -1, 0, or 1 which indicate whether the corresponding row of the spreadsheet should be editable. -1 indicates that the cell should be whatever is set by Main Area Cells Editable, 0 means not editable, and 1 means editable.
Rows Traversable	This is an IC-CAP Array that of values set to 0 or 1 which indicates whether the corresponding row of the spreadsheet is to be traversable or not.

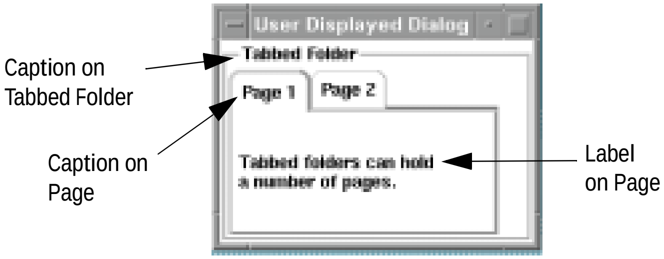
See “Using the Properties Dialog Box” on page 484.

Color Table

Set to D for <None Specified>
Set to C0 for Default Color
Set to C1 for Black
Set to C2 for Red
Set to C3 for Yellow
Set to C4 for GREEN
Set to C5 for CYAN
Set to C6 for BLUE
Set to C7 for MAGENTA
Set to C8 for GRAY
Set to C9 for WHITE
Set to C10 for MEDIUM BLUE
Set to C11 for DARK_SLATE_GRAY
Set to C12 for DIM_GREY
Set to C13 for SLATE_GREY
Set to C14 for MIDNIGHT_BLUE
Set to C15 for NAVY_BLUE
Set to C16 for DARK_SLATE_BLUE
Set to C17 for SLATE_BLUE
Set to C18 for MEDIUM_SLATE_BLUE
Set to C19 for ROYAL_BLUE
Set to C20 for DODGER_BLUE
Set to C21 for STEEL_BLUE
Set to C22 for DARK_GREEN
Set to C23 for DARK_OLIVE_GREEN
Set to C24 for SEA_GREEN
Set to C25 for FOREST_GREEN
Set to C26 for OLIVE_DRAB
Set to C27 for INDIAN_RED
Set to C28 for SADDLE_BROWN
Set to C29 for SIENNA
Set to C30 for FIRE_BRICK
Set to C31 for BROWN
Set to C32 for ORANGE_RED
Set to C33 for DEEP_PINK
Set to C34 for MAROON
Set to C35 for MEDIUM_VIOLET_RED
Set to C36 for VIOLET_RED
Set to C37 for DARK_ORCHID
Set to C38 for SNOW1
Set to C39 for SNOW2
Set to C40 for ANTIQUE_WHITE1
Set to C41 for LEMON_CHIFFON1
Set to C42 for IVORY1
Set to C43 for HONEYDEW1
Set to C44 for LAVENDER_BLUSH1
Set to C45 for LAVENDER_BLUSH2
Set to C46 for MISTY_ROSE1
Set to C47 for MISTY_ROSE2
Set to C48 for AZURE1
Set to C50 for AZURE2
Set to C51 for DODGER_BLUE2
Set to C52 for STEEL_BLUE1
Set to C53 for DEEP_SKY_BLUE1
Set to C54 for SKY_BLUE_1
Set to C55 for LIGHT_SKY_BLUE1
Set to C56 for SLATE_GRAY2
Set to C57 for LIGHT_STEEL_BLUE1
Set to C58 for LIGHT_BLUE1
Set to C60 for LIGHT_CYAN1
Set to C61 for PALE_TURQUOISE1
Set to C62 for CADET_BLUE
Set to C63 for CADET_BLUE2
Set to C64 for TURQUOISE1
Set to C65 for CYAN1
Set to C66 for DARK_SLATE_GRAY1
Set to C67 for AQUAMARINE2
Set to C68 for DARK_SEA_GREEN1
Set to C69 for DARK_SEA_GREEN2
Set to C70 for SEA_GREEN1
Set to C71 for PALE_GREEN1
Set to C72 for SPRING_GREEN1
Set to C73 for GREEN1
Set to C74 for CHARTREUSE1

Set to C75 for OLIVE_DRAB1
Set to C76 for DARK_OLIVE_GREEN1
Set to C77 for KHAKI1
Set to C78 for LIGHT_GOLDENROD1
Set to C79 for YELLOW1
Set to C80for GOLD1
Set to C81 for GOLDENROD1
Set to C82 for SIENNA2
Set to C83 for WHEAT2
Set to C84 for CHOCOLATE1
Set to C85 for BROWN1
Set to C86 for LIGHT_SALMON1
Set to C87 for ORANGE1
Set to C88 for CORAL2
Set to C89 for RED2
Set to C90 for DEEP_PINK1
Set to C91 for HOT_PINK1
Set to C92 for PINK1
Set to C93 for LIGHT_PINK1
Set to C94 for PALE_VIOLET_RED1
Set to C95 for MAROON1
Set to C96 for VIOLET_RED1
Set to C97 for MAGENTA1
Set to C98 for ORCHID1
Set to C99 for PLUM1
Set to C100 for THISTLE1
Set to C101 for GRAY52
Set to C102 for GRAY62
Set to C103 for GRAY72
Set to C104 for GRAY82

(Tab) Tabbed Folder



Tabbed folders can hold a number of pages.

The GUI Item Tab creates a folder of overlapping pages with non-overlapping tabs on each page. When a tab is selected, its page comes to the top of the stack.

The folder is named with

Properties > Tab > Options > Caption

Pages are added as children to the Tabbed folder using

Add Child... > Widget Properties > Select Type > Page

The caption on the tab of each page is set with **Tab > Caption**.

On each page you can add (as a child) unlimited text (including carriage returns) as

- A Label which will produce read-only (uneditable) text
- An Edit Text Box which produces editable text in a scrolling Edit Text box.

NOTE

On UNIX platforms, some widgets on Tabbed-folder pages may not appear properly. To avoid this potential problem, place your content within a single Table widget (it can have 0 margin width and 0 margin height) inside the page. For example, instead of placing 4 widgets on a page, place a single table on a page, then place the 4 widgets inside the table.

On Window platforms, you should also place your content within a single Table widget inside the page just in case the *mdl* is ever used on a UNIX platform.

NOTE

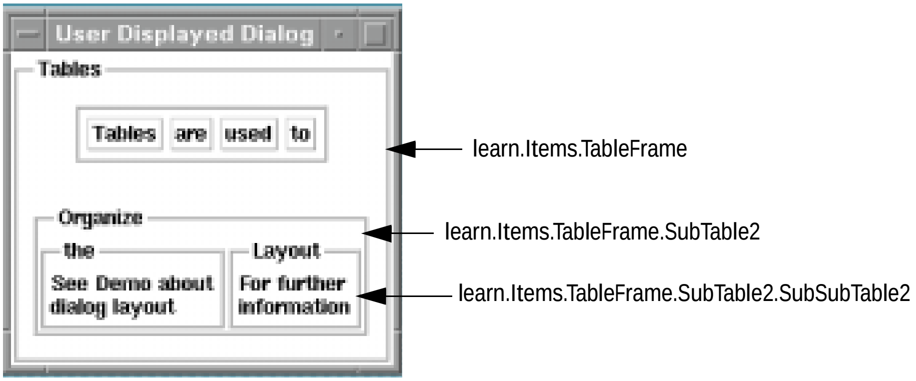
Any combination of widgets can also be entered on a page so as to produce a folder of GUIs.

Special Options	Description
Current Page	Indicates the initial page to display when an integer is set. If tracking a variable, the variable is updated with each page change to reflect the currently selected page.
Frame Visible	Indicates if a frame is to be visible or not. The default is FALSE.
Margin Height	The integer value for the number of pixels of margin at the top of the table.
Margin Width	The integer value for the number of pixels of margin at the sides of the table.
Page Change OK	Set to 1 just before a page change. The Page Validation callback is then invoked. If you set this option to track a variable, you may set it to 0 during the Page Validation callback to prevent a page change.

Callbacks	Description
Page Changed	Invoked whenever the current page is changed. If you track the Current Page with a variable, the variable will have the new page in the callback.
Page Validation	Invoked *before* a page is changed. Prior to invocation, Page Change OK is set to 1 if a variable is being tracked. Set this variable to 0 during the callback if the page change should not occur.

See “Using the Properties Dialog Box” on page 484.

Table


User Displayed Dialog
Tables
Tables
are
used
to
Organize
the
See Demo about dialog layout
Layout
For further information
learn.Items.TableFrame
learn.Items.TableFrame.SubTable2
learn.Items.TableFrame.SubTable2.SubSubTable2

Tables are used to organize the placement of widgets into a hierarchy of rows and columns, and are usually the first item to be defined in a GUI. The default orientation is vertical.

When widgets are added as children to a vertical table they will be placed one above the other in the same order as listed in the hierarchy in the Model GUI Items page. Horizontal tables may also be added as children to vertical tables.

Widgets are then added as children in the horizontal tables. Tables may be nested within other tables without limit to organize a complex hierarch of input and output devices.

See tutorial “More on Tables (Examples)” on page 536.

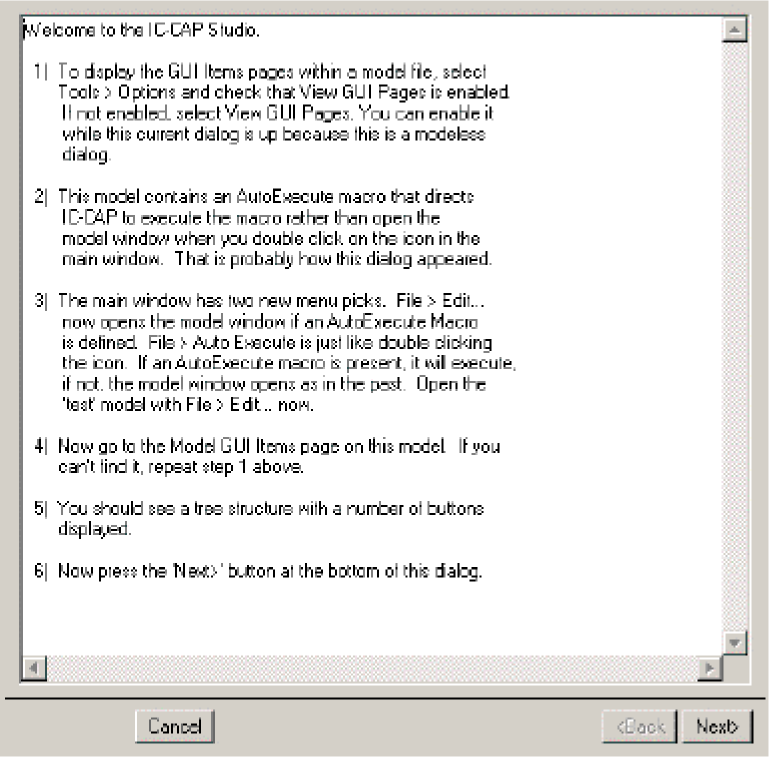
The Diode Extraction Demonstration is a good example.

Special Options	Description
Allow Resize	Permits items in a Table to dynamically resize themselves and force the entire dialog to resize to accommodate the change. Note that all tables in the hierarchy must be set to Allow Resize for the resizing to propagate all the way to the main dialog.
Default Sizing Options for children	The sizing options for all members of the table if they don't declare their own sizing options.
Equalize All	When TRUE all widgets in a table are normalized to the same size. The default is FALSE.
Frame Visible	Indicates if a frame is to be visible of not. The default is FALSE unless there is a Caption.
Margin Height and Margin Width	The integer value for the number of pixels of margin at the sides and top of the table.
Number Rows/Columns	The integer value for the number of rows displayed for a horizontal orientation or the number of columns displayed for a vertical orientation. The children will be divided up evenly between the rows/columns.
Row and Column Spacing	Indicates how many pixels of spacing between rows or columns.
Use Radio Behavior	When this is TRUE only one radio button in the table may be selected at a time.

See Chapter 12, “Using the Properties Dialog Box.”

OnLine Briefing

The following pages may also be viewed on line. We suggest you add ICCAP_VIEW_GUI_PAGES=1 to the configuration file *iccap.cfg*. This ensures that the GUI Model Pages tab is always present. Open the model file *gui_tutorial.mdl* located in the *examples/iccap_studio* directory.



Welcome to the IC-CAP Studio.

1| To display the GUI Items pages within a model file, select Tools > Options and check that View GUI Pages is enabled. If not enabled, select View GUI Pages. You can enable it while this current dialog is up because this is a modeless dialog.

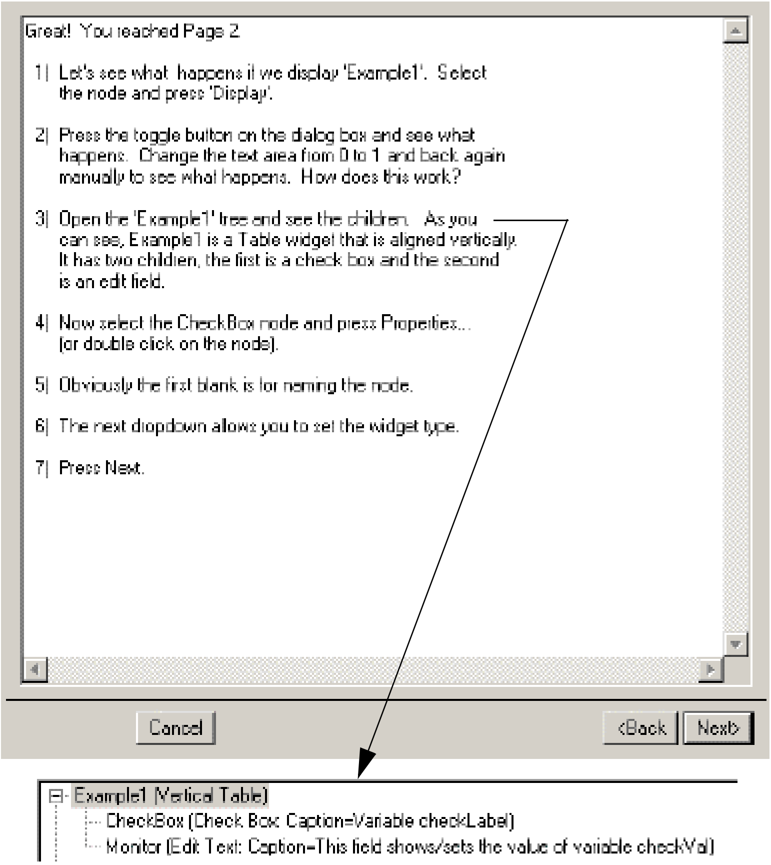
2| This model contains an AutoExecute macro that directs IC-CAP to execute the macro rather than open the model window when you double click on the icon in the main window. That is probably how this dialog appeared.

3| The main window has two new menu picks. File > Edit... now opens the model window if an AutoExecute Macro is defined. File > Auto Execute is just like double clicking the icon. If an AutoExecute macro is present, it will execute, if not, the model window opens as in the past. Open the 'test' model with File > Edit... now.

4| Now go to the Model GUI Items page on this model. If you can't find it, repeat step 1 above.

5| You should see a tree structure with a number of buttons displayed.

6| Now press the 'Next>' button at the bottom of this dialog.



Great! You reached Page 2

1) Let's see what happens if we display 'Example1'. Select the node and press 'Display'.

2) Press the toggle button on the dialog box and see what happens. Change the text area from 0 to 1 and back again manually to see what happens. How does this work?

3) Open the 'Example1' tree and see the children. As you can see, Example1 is a Table widget that is aligned vertically. It has two children, the first is a check box and the second is an edit field.

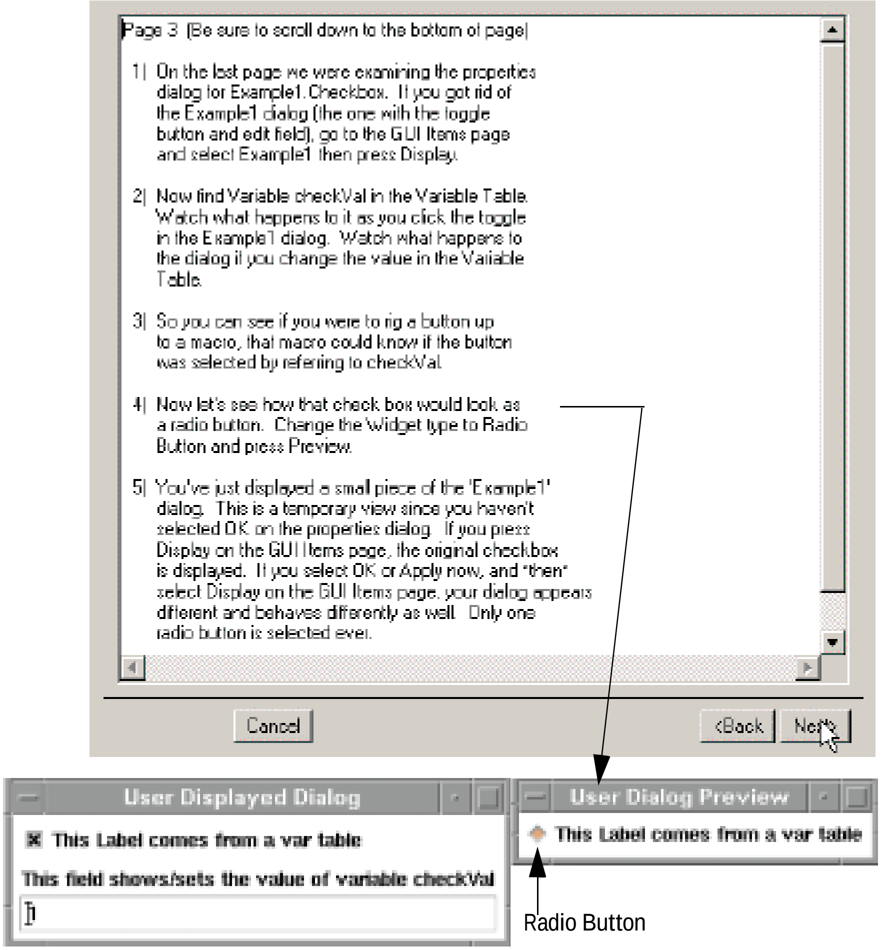
4) Now select the CheckBox node and press Properties... (or double click on the node).

5) Obviously the first blank is for naming the node.

6) The next dropdown allows you to set the widget type.

7) Press Next.

Example1 (Vertical Table)
CheckBox (Check Box: Caption=Variable checkLabel)
Monitor (Edit Text: Caption=This field shows/sets the value of variable checkVal)



Page 3 (Be sure to scroll down to the bottom of page)

1) On the last page we were examining the properties dialog for Example1.Checkbox. If you got rid of the Example1 dialog (the one with the toggle button and edit field), go to the GUI Items page and select Example1 then press Display.

2) Now find Variable checkVal in the Variable Table. Watch what happens to it as you click the toggle in the Example1 dialog. Watch what happens to the dialog if you change the value in the Variable Table.

3) So you can see if you were to rig a button up to a macro, that macro could know if the button was selected by refering to checkVal.

4) Now let's see how that check box would look as a radio button. Change the Widget type to Radio Button and press Preview.

5) You've just displayed a small piece of the 'Example1' dialog. This is a temporary view since you haven't selected OK on the properties dialog. If you press Display on the GUI Items page, the original checkbox is displayed. If you select OK or Apply now, and *then* select Display on the GUI Items page, your dialog appears different and behaves differently as well. Only one radio button is selected ever.

☒ This Label comes from a var table

This field shows/sets the value of variable checkVal

1

● This Label comes from a var table

Controlling Dialogs

Launching Dialogs

You can launch dialogs defined in the GUI page via four iccap_func commands:

```
iccap_func("/mdl/dut/orSet/GuiName","Display Modal GUI",
          "InstanceName")
iccap_func("/mdl/dut/orSet/GuiName","Display Modeless
          GUI","InstanceName")
```

The above forms require that you provide a third argument, which is a unique name for that instance of the dialog. In this way you can raise three identical dialogs if it makes sense to do so.

```
iccap_func("/mdl/dut/orSet/GuiName","Display Single Modal
           GUI")
iccap_func("/mdl/dut/orSet/GuiName","Display Single
           Modeless GUI")
```

These forms don’t require a name, and thus you can only show the dialog once using this command. It’s useful for callbacks—see the properties for the Wizard GUI Item's Cancel button in the *gui_tutorial.mdl*.

Auto Execute Feature

If you name a macro AutoExecute, it will run when the icon is double clicked on the main window. The normal Model window will not open. To open the normal Model window, select File > Edit... or right click on the icon and choose Edit... If an AutoExecute macro is not declared, double clicking the icon opens the Model window as usual. You can create an AutoExecute Dialog that creates a particular GUI. This alternate GUI could allow an extraction to be preformed without opening the model file. The following example shows a simple 2 line AutoExecute macro that first calls a transform in a particular setup, which resets the controlling flags of the GUI to a known state, then it displays a modeless dialog.

```
iccap_func("/mdl/dut/setup/resetGUI","Execute")
           ! restore known state
iccap_func("/mdl/dut/setup/GuiName","Display Modeless
           GUI","your GUI name here")
```

For additional examples, see gui_tutorial.mdl or gui_demo.mdl.

Removing Dialogs

You can remove dialogs in two ways—Close and Destroy:

```
iccap_func("/mdl/dut/orSet/GUIName","Close GUI",
           "InstanceName")
iccap_func("/mdl/dut/orSet/GUIName","Destroy GUI",
           "InstanceName")
iccap_func("/mdl/dut/orSet/GUIName","Close Single GUI")
iccap_func("/mdl/dut/orSet/GUIName","Destroy Single GUI")
```

Close GUI merely takes the dialog off the screen. The dialog is still built and is reacting to modifications to the variables page, etc. Recalling Display after you've closed simply redisplays the dialog and is very fast.

Destroy GUI actually frees the memory associated with the dialog and the next Display needs to build the entire dialog again.

Automating GUI Items

Adding Items

You may add items to a GUI page via iccap_func on either the GUI Items page, or another GUI item. Two additional parameters are required to name the item and to declare the item's type. The type specification must be a code that uniquely declares the Item type. Select the **Show Codes** button on the Properties dialog of a GUI Item to see the codes for all Item types.

```
iccap_func("./GUIItems","Add GUI","ANewTable","TL")
iccap_func("./ANewTable","Add GUI","Button","PB")
```

Setting Options

You may change options on any Item on a GUI Page from PEL by using the Set GUI Options action on a GUI Item. You may enter as many additional arguments as desired. The last argument must be "" (a null string) to signify completion. Each argument is of the form <code>=<val>. Code is the associated code for the option as found by pressing Show Codes on the properties dialog. The <val> can be one of four forms.

- A literal

  <val> may be a string, or a number

- A Variable Changed

  If the first character after the equal sign is a %, the value is taken to track a variable by the given name.

- An Enumerated type code

  Many options have enumerated types such as list selection mode. Each of these possible values has an associated code which can be viewed on the properties dialog.

- Multiple Enumerated types

  Some options (sizing options, for example) allow multiple enumerated types to be set (fixed width and fixed height). These are indicated by specifying <code>|<code>.

Examples:

```
iccap_func("./someItem","Set GUI Options","OR=HO",
            "CA="title","")
iccap_func("./some.other.item","Set GUI Options",
            "TO=FW|FH","")
```

Setting Callbacks

You may set a callback on a GUI item by using the action Set GUI Callback. This action requires three other arguments. The first is the code for the callback. The second and third extra argument are the two parameters to the callbacks associated iccap_func().

Example:

```
iccap_func("./someItem","Set GUI Callback","AC","aMacro",
           "Execute)
```

Data Input Slider (Example)

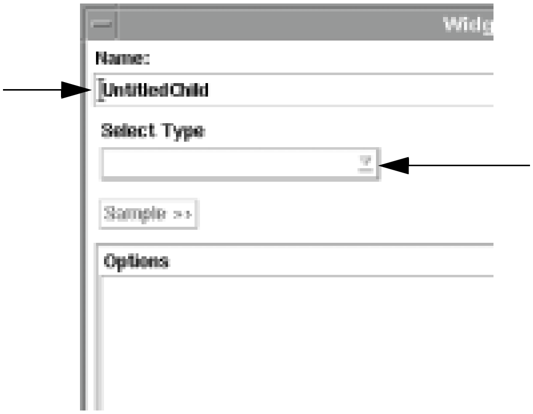
This example shows the use of the Widget Properties dialog box to create a small GUI with two linked input/output items in it.

In the main window click Tab Model Gui Items**.** A menu of GUI manipulation commands appears at the left.

To start a new GUI from the left menu Select Add...

Start a new GUI

A large dialog box is displayed titled Widget Properties.



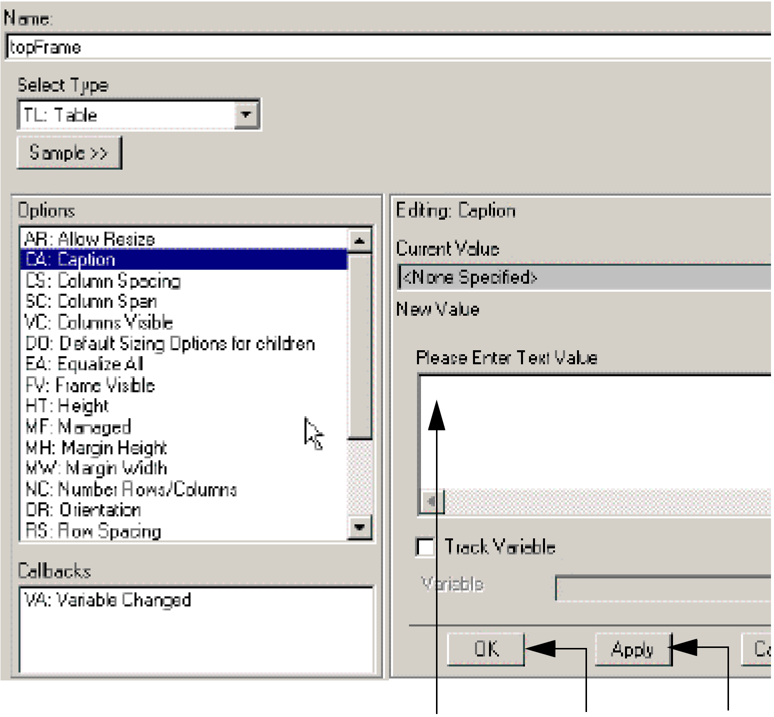
Edit name here

Click here to show dropdown list of GUI Items

In the Name: box edit the default name UntitledChild, changing it to `topFrame`**.**

From the Select Type box pick **Table** from the dropdown list.

Once a widget type has been chosen an Options list appears.



Name:
topFrame

Select Type
TL: Table

Options
- AR: Allow Resize
- CA: Caption
- CS: Column Spacing
- SC: Column Span
- VC: Columns Visible
- DO: Default Sizing Options for children
- EA: Equalize All
- FV: Frame Visible
- HT: Height
- MF: Managed
- MH: Margin Height
- MW: Margin Width
- NC: Number Rows/Columns
- OR: Orientation
- RS: Row Spacing

Callbacks
- VA: Variable Changed

Editing: Caption

Current Value
<None Specified>

New Value

Please Enter Text Value

□ Track Variable

Variable

Enter Caption here

Accept Caption here or here

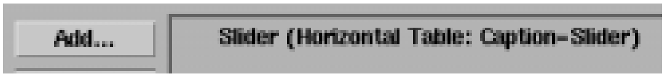
Select **Caption** from the Options list and a new dialog box appears on the right.

Add the caption name `Slider` in the Enter Text Value box and click **OK** in the box.

Select **Orientation** and pick **Horizontal** from the dropdown list, then **OK**.

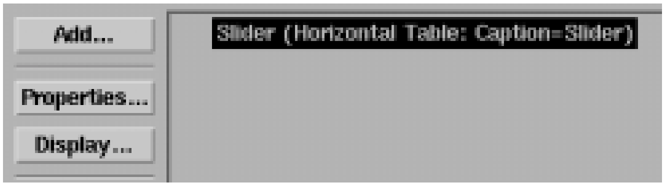
Click **OK** at the bottom of the Widget Properties box.

The text in the main window will now look like this:



Slider (Horizontal Table: Caption=Slider)

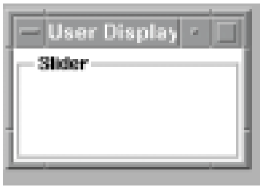
Click on the text to select it.



Slider (Horizontal Table: Caption=Slider)

Select **Display...**

An empty table captioned Slider appears.



We will now add a slider and a value display to the GUI.

With the text in the main window still selected, click **Add Child...**

The Widget Properties box appears.

Edit the Name to `Slider`.

Click the button in the **Select Type** box and select **Slider**.

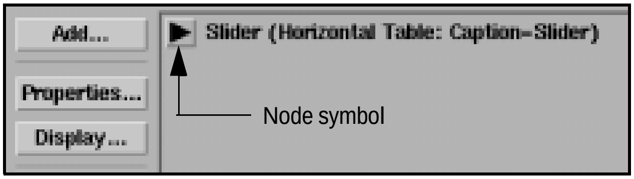
From the Options list select **Current Position**.

In the Editing: Current Position panel click the **Track Variable** box.

Click on the **Variable** box and enter `sliderVal` and **OK** in the box under it.

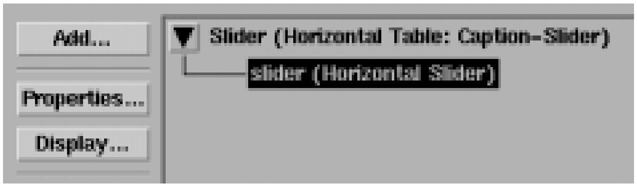
Click **OK** in the bottom left box.

The text in the main window will now look like this:



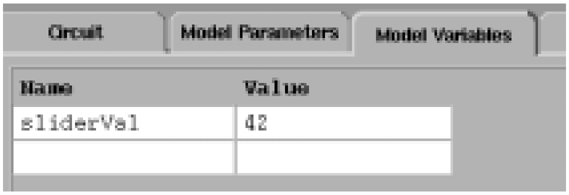
Note the node symbol which indicates that there is now hierarchy in the GUI.

Click on the node. It will rotate, and the contents will be shown.


Add...
Properties...
Display...
Slider (Horizontal Table: Caption-Slider)
slider (Horizontal Slider)

Before the slider can communicate its Current Position in the variable we named *sliderVal*, this variable has to be declared to the system.

In this example we will do it by clicking on the **Model Variables** tab. A table, probably empty will appear. On the name side type `sliderVal`.



Name	Value
sliderVal	42

Note that a value for *sliderVal* appears automatically.

Now click the **Model GUI Items** tab.

Select the top level item **Slider (Horizontal Table...**

Click **Add Child...**

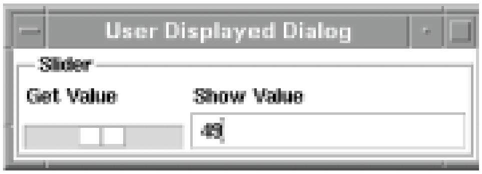
The Widget Properties dialog box appears again.

In the Select Type box click **Edit Text**.

Edit Name to `Show Value`.

Select **Field Value** in the Options list, set **Track Variable**, and set the variable name to `sliderVal`, and click **OK** directly underneath it.

The slider GUI now looks as shown below.



NOTE

When the slider is moved, the numerical value displayed changes. Also, when the number is edited, the slider position moves in response, acting as an output device.

More on Tables (Examples)

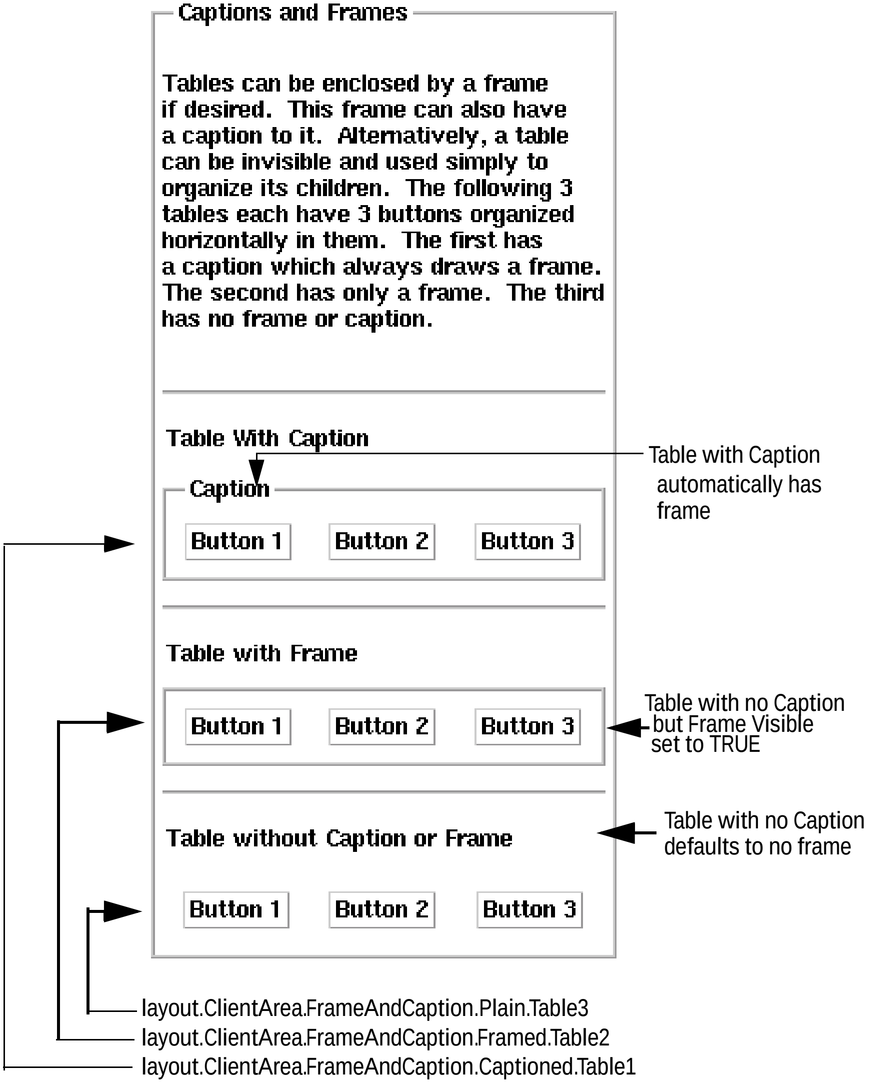
To keep the tabs less cluttered IC-CAP, by default, does not display the GUI Items tab.

To make the tab visible, from the ICCAP/Main Window, choose the menu pick **Tools > Options > View GUI Pages**. Then when you open a new model (or close and reopen the Model window of a previously loaded model) you will see a GUI Items page at the Model level, DUT level, and Setup level.

Captions, Frames and Sizing

The following examples are taken from the Wizard. You can follow them both with this manual or by showing them through IC-CAP.

In the following figure, the path to each of the three tables that control the style are shown at the bottom.



Captions and Frames

Tables can be enclosed by a frame if desired. This frame can also have a caption to it. Alternatively, a table can be invisible and used simply to organize its children. The following 3 tables each have 3 buttons organized horizontally in them. The first has a caption which always draws a frame. The second has only a frame. The third has no frame or caption.

Table With Caption

Table with Frame

Table without Caption or Frame

Table with Caption automatically has frame

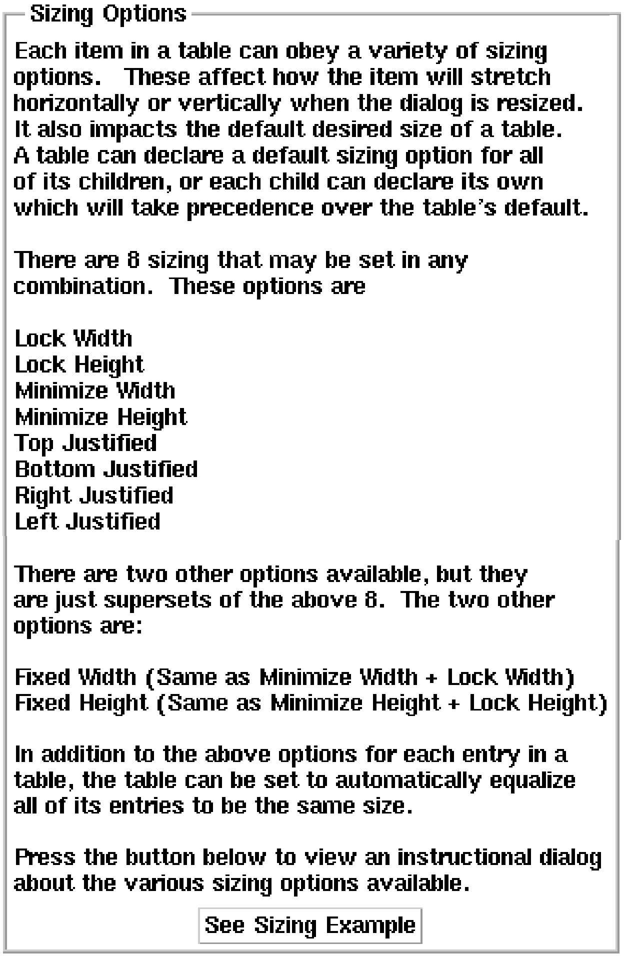
Table with no Caption but Frame Visible set to TRUE

Table with no Caption defaults to no frame

layout.ClientArea.FrameAndCaption.Plain.Table3

layout.ClientArea.FrameAndCaption.Framed.Table2

layout.ClientArea.FrameAndCaption.Captioned.Table1



Sizing Options

Each item in a table can obey a variety of sizing options. These affect how the item will stretch horizontally or vertically when the dialog is resized. It also impacts the default desired size of a table. A table can declare a default sizing option for all of its children, or each child can declare its own which will take precedence over the table's default.

There are 8 sizing that may be set in any combination. These options are

Lock Width
Lock Height
Minimize Width
Minimize Height
Top Justified
Bottom Justified
Right Justified
Left Justified

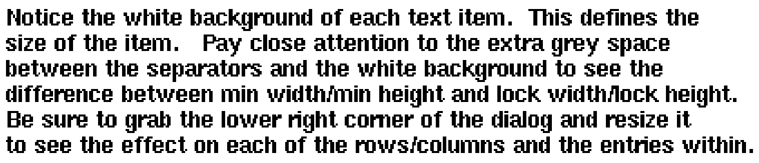
There are two other options available, but they are just supersets of the above 8. The two other options are:

Fixed Width (Same as Minimize Width + Lock Width)
Fixed Height (Same as Minimize Height + Lock Height)

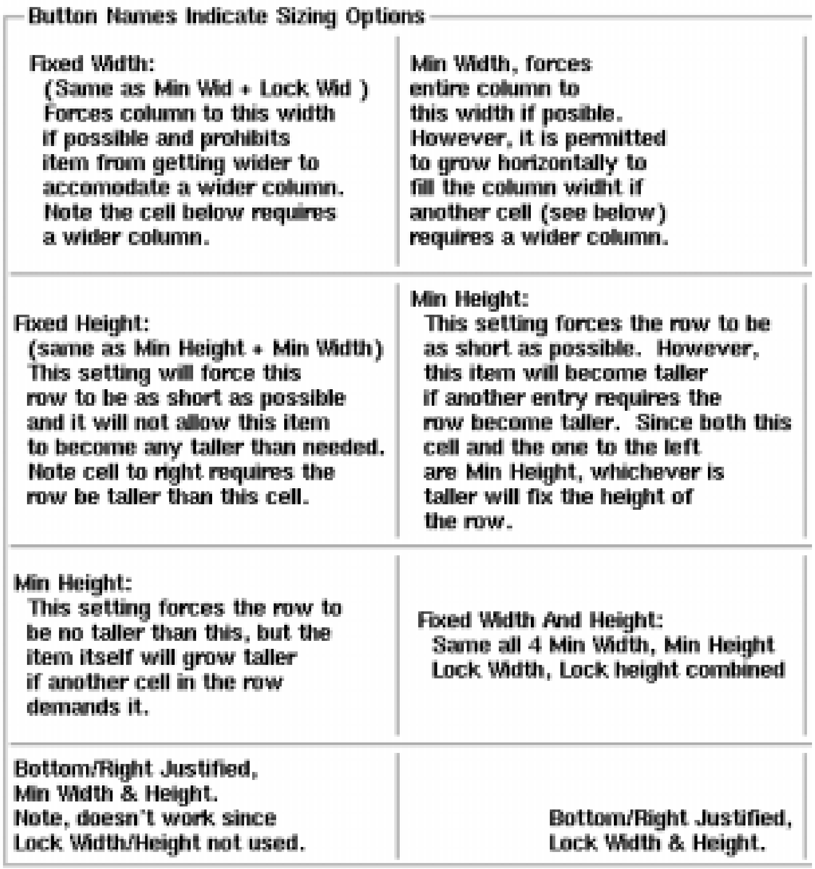
In addition to the above options for each entry in a table, the table can be set to automatically equalize all of its entries to be the same size.

Press the button below to view an instructional dialog about the various sizing options available.

In the examples below display the following window, and then stretch it vertically and horizontally to really see the affect of the options.



Notice the white background of each text item. This defines the size of the item. Pay close attention to the extra grey space between the separators and the white background to see the difference between min width/min height and lock width/lock height. Be sure to grab the lower right corner of the dialog and resize it to see the effect on each of the rows/columns and the entries within.



Button Names Indicate Sizing Options

Fixed Width:
(Same as Min Wid + Lock Wid)
Forces column to this width
if possible and prohibits
item from getting wider to
accomodate a wider column.
Note the cell below requires
a wider column.

Min Width, forces
entire column to
this width if posible.
However, it is permitted
to grow horizontally to
fill the column widht if
another cell (see below)
requires a wider column.

Fixed Height:
(same as Min Height + Min Width)
This setting will force this
row to be as short as possible
and it will not allow this item
to become any taller than needed.
Note cell to right requires the
row be taller than this cell.

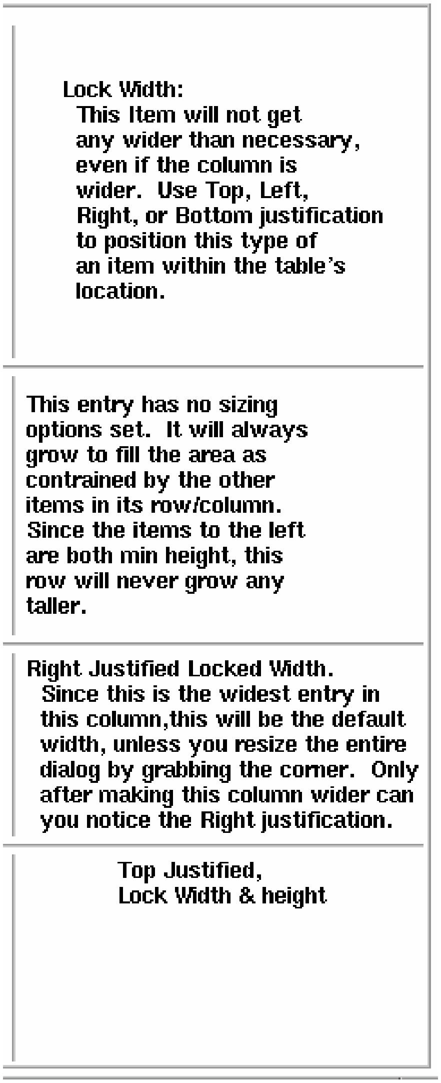
Min Height:
This setting forces the row to be
as short as possible. However,
this item will become taller
if another entry requires the
row become taller. Since both this
cell and the one to the left
are Min Height, whichever is
taller will fix the height of
the row.

Min Height:
This setting forces the row to
be no taller than this, but the
item itself will grow taller
if another cell in the row
demands it.

Fixed Width And Height:
Same all 4 Min Width, Min Height
Lock Width, Lock height combined

Bottom/Right Justified,
Min Width & Height.
Note, doesn't work since
Lock Width/Height not used.

Bottom/Right Justified,
Lock Width & Height.

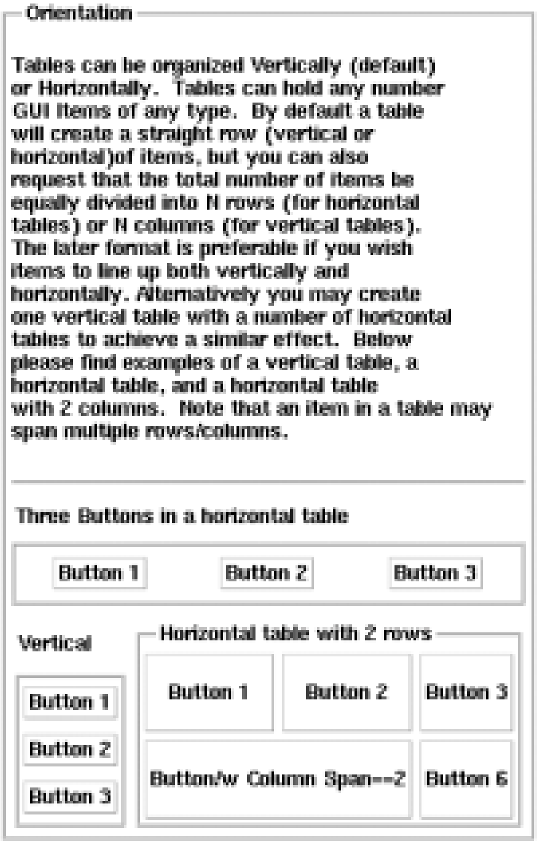


Lock Width:
This Item will not get
any wider than necessary,
even if the column is
wider. Use Top, Left,
Right, or Bottom justification
to position this type of
an item within the table's
location.

This entry has no sizing
options set. It will always
grow to fill the area as
contrained by the other
items in its row/column.
Since the items to the left
are both min height, this
row will never grow any
taller.

Right Justified Locked Width.
Since this is the widest entry in
this column,this will be the default
width, unless you resize the entire
dialog by grabbing the corner. Only
after making this column wider can
you notice the Right justification.

Top Justified,
Lock Width & height



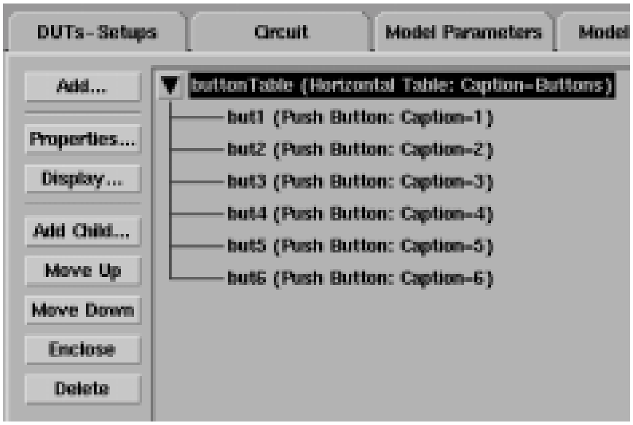
Orientation

Tables can be organized Vertically (default) or Horizontally. Tables can hold any number GUI items of any type. By default a table will create a straight row (vertical or horizontal)of items, but you can also request that the total number of items be equally divided into N rows (for horizontal tables) or N columns (for vertical tables). The later format is preferable if you wish items to line up both vertically and horizontally. Alternatively you may create one vertical table with a number of horizontal tables to achieve a similar effect. Below please find examples of a vertical table, a horizontal table, and a horizontal table with 2 columns. Note that an item in a table may span multiple rows/columns.

Tables within Tables

Rows and Columns within Tables

The figure below shows the tree for six Push Buttons added to a Table.



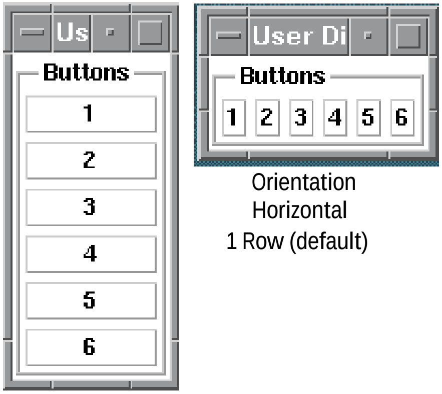
```
▼ buttonTable (Horizontal Table: Caption=Buttons)
    but1 (Push Button: Caption=1)
    but2 (Push Button: Caption=2)
    but3 (Push Button: Caption=3)
    but4 (Push Button: Caption=4)
    but5 (Push Button: Caption=5)
    but6 (Push Button: Caption=6)
```

6-Button Table

The resultant set of buttons can take many forms. In the figure below the Option Number Rows/Columns is at the default value of 1. The left set of buttons has the default Orientation of Vertical, the right is Horizontal.

Orientation Vertical

1 Column (default)

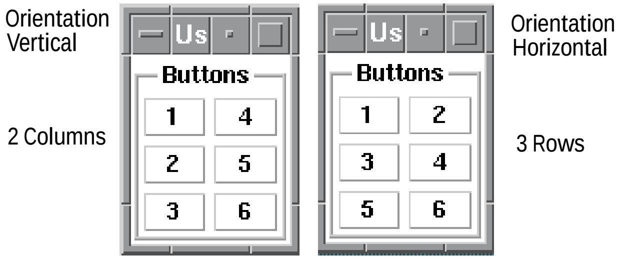


Orientation Horizontal

1 Row (default)

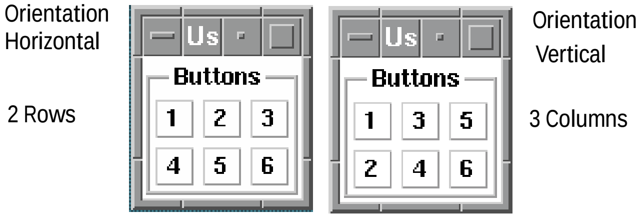
Orientation Vertical and Horizontal

There is no limit to the number of items in a row or column unless it is set with the Number Rows/Columns option set to a value greater than one as shown below. The six buttons are divided between two columns (left) and three rows (right).



Placement using Rows or Columns

Yet another ordering is shown below.



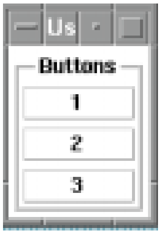
Placement using Rows or Columns

The use of multiple rows or columns is simple and flexible. It works best with items which are all the same. The problems are when items are added or removed as the GUI is developed causing the number of items to be no longer exactly divisible by the number of rows or columns.

Enclose

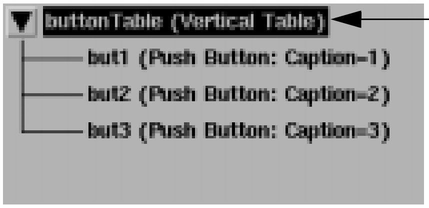
The hierarchy of tables nested within tables controls the relative placements of all the widgets in a GUI. If we need to add more widgets in new places not permitted by the current hierarchy of place holders then we need to insert a new level into the overall hierarchy. This is what the Enclose command does.

For example we have a vertical table called buttonTable which contains three Push Buttons in a vertical column.



We wish to add three sliders but a six-high column of items would be too tall to fit within the GUI layout. What we want is to place the three new items in a vertical column to the right of the existing column, but there is no place-holder for them.

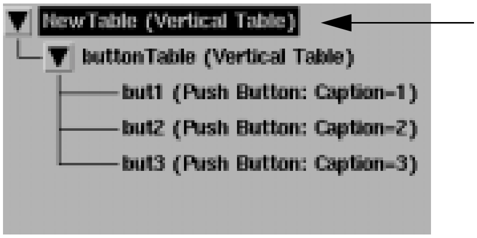
On the Model GUI Page tab select **buttonTable** and click on **Enclose**.



```
▼ buttonTable (Vertical Table)
  ├── but1 (Push Button: Caption=1)
  ├── but2 (Push Button: Caption=2)
  └── but3 (Push Button: Caption=3)
```

Select this and click Enclose

You will see that a new table with the default name Newtable has been inserted into the hierarchy as a new level, and that buttonTable is nested in it. (At this point the displays of Newtable and buttonTable are identical.)



```
▼ NewTable (Vertical Table)
  └ ▼ buttonTable (Vertical Table)
      ├── but1 (Push Button: Caption=1)
      ├── but2 (Push Button: Caption=2)
      └── but3 (Push Button: Caption=3)
```

New level created

Newtable has the default orientation of Vertical so we select it and click **Properties** so we can change the orientation to Horizontal.

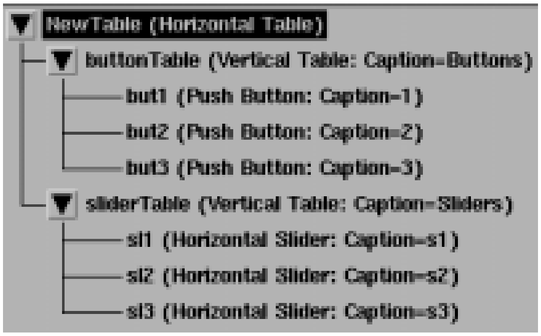
Back in the Model GUI Items we now select **Newtable** and click on **Add Child...**.

In the Widget Properties dialog select **Table** and name it, for example `sliderTable`.

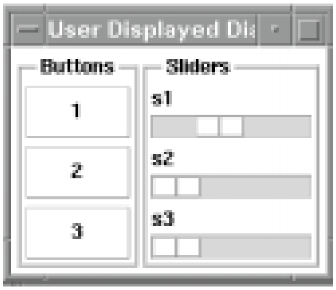
Back in the Model GUI Items select **sliderTable** and click on **Add Child...**

In the Widget Properties dialog select **Slider**.

Repeat this step two more times and we now have two 3-high columns, one of Push Buttons and the other of Sliders, as shown below.



```
NewTable (Horizontal Table)
  buttonTable (Vertical Table: Caption=Buttons)
    but1 (Push Button: Caption=1)
    but2 (Push Button: Caption=2)
    but3 (Push Button: Caption=3)
  sliderTable (Vertical Table: Caption=Sliders)
    sl1 (Horizontal Slider: Caption=s1)
    sl2 (Horizontal Slider: Caption=s2)
    sl3 (Horizontal Slider: Caption=s3)
```



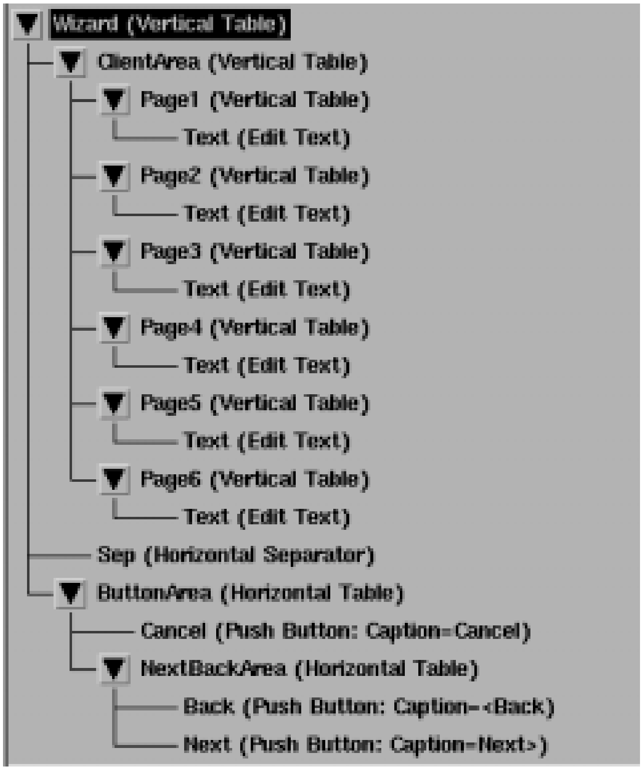
The Enclose command gave you two advantages.

- You can change your mind during the development of the GUI and add a new level of hierarchy.

- The buttons and sliders are now in separate Tables. If you change the number in either table there will be no overflow from one to the other as would happen with a single table split into rows and columns.

Viewing Pages Sequentially (Example)

Explore the Wizard tree by setting all the node symbols downward so as to show the entire hierarchy. It should look like this:



```
▼ Wizard (Vertical Table)
├─▼ ClientArea (Vertical Table)
│  ├─▼ Page1 (Vertical Table)
│  │  └── Text (Edit Text)
│  ├─▼ Page2 (Vertical Table)
│  │  └── Text (Edit Text)
│  ├─▼ Page3 (Vertical Table)
│  │  └── Text (Edit Text)
│  ├─▼ Page4 (Vertical Table)
│  │  └── Text (Edit Text)
│  ├─▼ Page5 (Vertical Table)
│  │  └── Text (Edit Text)
│  └─▼ Page6 (Vertical Table)
│     └── Text (Edit Text)
├── Sep (Horizontal Separator)
└─▼ ButtonArea (Horizontal Table)
   ├── Cancel (Push Button: Caption=Cancel)
   └─▼ NextBackArea (Horizontal Table)
      ├── Back (Push Button: Caption=<Back)
      └── Next (Push Button: Caption=Next>)
```

Page structure of the Wizard GUI

The Wizard (Vertical Table) contains ClientArea (Vertical Table) and ButtonArea (Vertical Table). Both of these can be selected by clicking on them and then clicking on **Display...**

In ClientArea there are six tables each containing an Edit Text item. If you double-click on any Text item you will see in the Properties dialog box that the Field Variable is set to a specific member of the ICCAP_ARRAY wizText which stores the contents of each page. The index into wizText is one less than the page number.

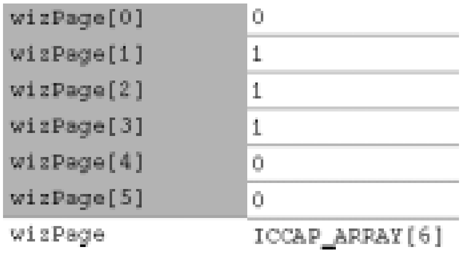
So why are not all six pages displayed simultaneously?

You can see the answer by double-clicking on the parent Table of each Edit Text item. You will see in the Properties dialog box that Managed is controlled by the array wizPage.

- wizPage is an array of flags.
- Only one of the flags is TRUE.
- Only the page with the TRUE flag is managed (i.e. visible).
- The rest are not visible and have zero size.
- The macro showPage manages the wizPage flag array

NOTE

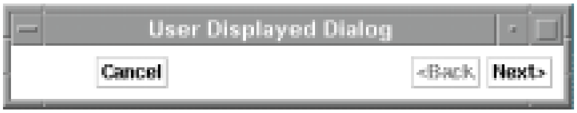
Click on the **Model Variables** tab to see the contents of the wizPage array.



wizPage[0]	0
wizPage[1]	1
wizPage[2]	1
wizPage[3]	1
wizPage[4]	0
wizPage[5]	0
wizPage	ICCAP_ARRAY[6]

It will normally have only one member set to 1 (TRUE). If you edit it as shown here with three members set to 1, and then select and display the wizard it will show three pages instead of one!

In the ButtonArea are the three buttons that appear at the bottom of the Wizard. The nested table scheme is simply to keep the Next and Back buttons next to each other at the right, and the Cancel button separate at the left.



The ButtonArea of the Wizard

Macros started by Push Buttons

Click on the **Next (Push Button)** at the bottom of the tree. In the Properties dialog box under Callbacks you will see **Button Pressed (iccap_funct("nextWizard","Execute"))**. This simply means that the Next button calls the macro nextWizard. Similarly the Back button calls the macro backupWizard.

You can see the text of these macros by clicking on the **Macros** tab and selecting **nextWizard** and **backupWiizard** under the Select Macro: **header**.

As you can see these macros increase and decrease respectively the current page number, then they call the macro showPage.

The macro showPage

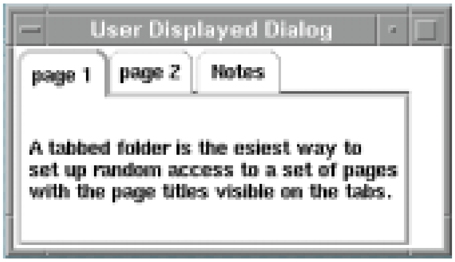
- Keeps the page number within the number of pages.
- Sets the flags backSense and nextSense when the ends of the range are reached.

The flags backSense and nextSense control the greying out of the paging buttons when the limit has been reached. See the greyed and 'insensitive' Back button in the Figure above.

For the Next and Back buttons, look on the Properties dialog for Sensitivity which controls whether an item is disabled (insensitive), or if it is responding normally. They appear as Sensitivity (Variable nextSense) and Sensitivity (Variable backSense).

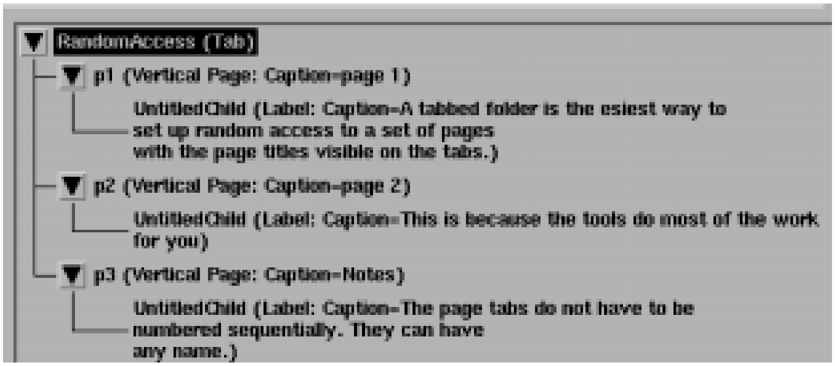
Viewing Pages at Random (Example)

The easiest way is to add Pages to a Tab. The text on each page is added as a Label. A GUI example is shown below:



A tabbed folder is the esiest way to
set up random access to a set of pages
with the page titles visible on the tabs.

The structure of the 3-page Tabbed Folder is shown below:



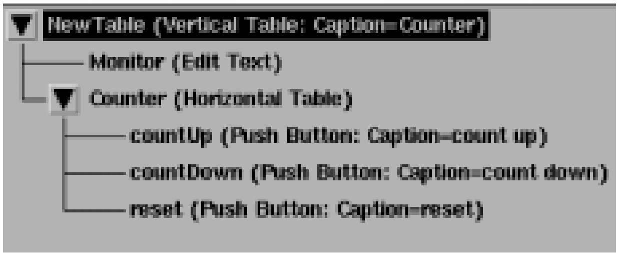
```
▼ RandomAccess (Tab)
  ▼ p1 (Vertical Page: Caption=page 1)
       UntitledChild (Label: Caption=A tabbed folder is the esiest way to
       set up random access to a set of pages
       with the page titles visible on the tabs.)
  ▼ p2 (Vertical Page: Caption=page 2)
       UntitledChild (Label: Caption=This is because the tools do most of the work
       for you)
  ▼ p3 (Vertical Page: Caption=Notes)
       UntitledChild (Label: Caption=The page tabs do not have to be
       numbered sequentially. They can have
       any name.)
```

The tab captions are the Captions of the Page, and the text on each page is a Label. Random access comes automatically with the Tabbed Folder.

Counter (Example)

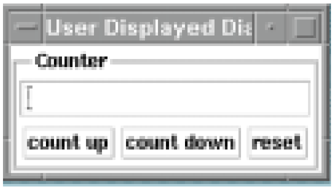
Load the model counterParts.mdl from the Model window.

This is how the tree looks:



```
▼ NewTable (Vertical Table: Caption=Counter)
  ├── Monitor (Edit Text)
  └── ▼ Counter (Horizontal Table)
        ├── countUp (Push Button: Caption=count up)
        ├── countDown (Push Button: Caption=count down)
        └── reset (Push Button: Caption=reset)
```

Display the top level. This is how the GUI looks:

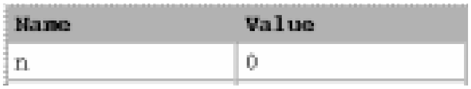


Try to make it work and you get error messages because the buttons are not linked to anything. It won't even display a value. We will now fix that.

We need to create a variable:

1 Click on the **Model Variables** tab. The page will show an empty table.

2 In the Name field type n and in the Value field type 0, the initial value.



Name	Value
n	0

Now we link the variable n *to the* Monitor *(Edit Text) box:*

1 Click on the **Model GUI Items** tab and select the item **Monitor (Edit Text)**.

2 Double-click it to bring up the Properties dialog.

3 In the Options list click on **Field Value**.

4 In the Edit dialog (at the right) click on **Track Variable**.

5 Click on the **Variable** box, and type `n`, the click on the **Apply** box underneath.

6 The Field Value will now be displayed as Field Value (Variable n).

7 The Edit Text box is now linked to the variable n.

8 Click on the bottom left **OK** button.

We now create macros to execute when the buttons are pressed:

1 Click on the **Macros** tab.

2 Click on **New** from the left menu.

3 In the dialog box type the macro name `reset`, and click **OK**.

4 Under the Select Macro: heading click on **reset**.

5 In the blank page on the right type `n = 0`.

This is the contents of the macro named reset.

6 Now create two more macros:

countUp contains the text: n = n + 1

countDown contains the text: n = n - 1

We now link the buttons to the macros:

1 Click on the **Model GUI Items** tab.

2 Double-click the **countUp** push button in the tree.

3 In the Properties dialog, under Callbacks, click on **Button Pressed**.

4 In the Edit dialog enter

- The macro name countUp as the first argument.

- The action Execute as the second argument.

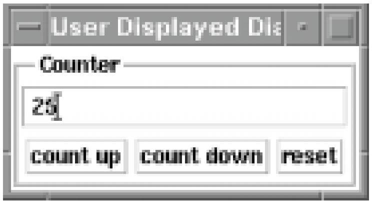
5 Click on **Apply** underneath.

The callback will now show

Button Pressed (iccap funct("countUp","Execute"))

6 Repeat this for the other two buttons, linking them to the macros with the same names.

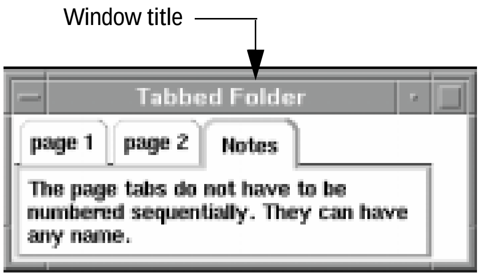
The counter will now count up, count down and reset:



25

Adding a Caption to the Unix Window (Example)

To add a caption (i.e. an instance name) to the Unix window, such as “Tabbed Folder” in the example below you have to launch the GUI through a macro with iccap_func(). This will replace the default caption “User Displayed Dialog”.



The page tabs do not have to be numbered sequentially. They can have any name.

The macro can have any name of course and contains an iccap_func() call such as

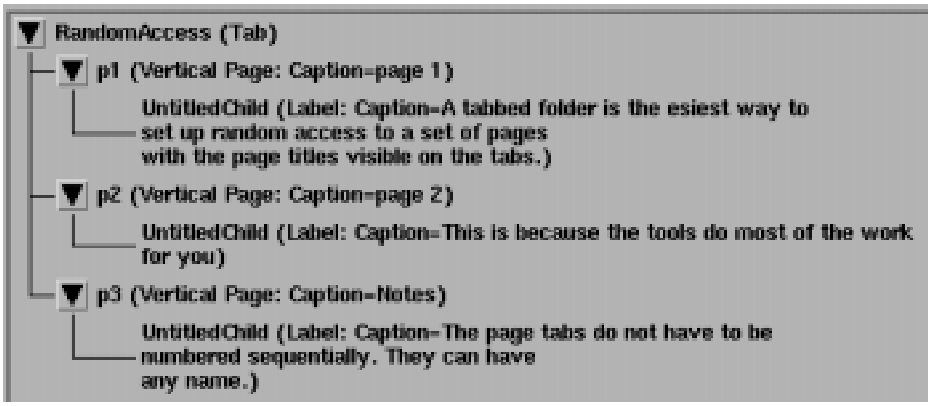
```
iccap_func("./RandomAccess","Display Modal GUI","Tabbed
           Folder")
```

where:

./RandomAccess refers to the top level name of a GUI item (see next figure).

Display Modal GUI is the action.

Tabbed Folder is the caption (title) to appear on the Unix window.



RandomAccess (Tab)

p1 (Vertical Page: Caption=page 1)

UntitledChild (Label: Caption=A tabbed folder is the esiest way to set up random access to a set of pages with the page titles visible on the tabs.)

p2 (Vertical Page: Caption=page 2)

UntitledChild (Label: Caption=This is because the tools do most of the work for you)

p3 (Vertical Page: Caption=Notes)

UntitledChild (Label: Caption=The page tabs do not have to be numbered sequentially. They can have any name.)

More on Sliders (Examples)

Review of Key Features of the Slider

- The Slider operates both as an input and an output device. When the value representing the slider position is changed, the slider position moves accordingly.
- The Slider has no output device of its own for numbers. You would typically use an Edit Text box (for editable numbers) or a Label for read-only numbers.
- The output is an integer.
- The default min/max range is 0 to 100. The minimum value is 0 or greater, and the maximum value is unlimited.

Six Versions of a Slider GUI (slider.mdl)

SliderVersion1 uses the default settings. The range is 0 to 100 in steps of 1.

The slider option Current Position shares the variable slidePosn with the Edit Text Box option Field Value.

To operate SliderVersion1:

1. Select **SliderVersion1** and click on **Display**.
2. Drag the slider button back and forth between its limits. Notice how smoothly it moves. Actually, it moves 2 inches in 100 steps of 0.02 inches.
3. Edit the number in the text box. When Enter is pressed note how the slider position responds to the edited value. Click on the **Model Variables** tab. You will see the value of slidePosn responding to our GUI.
4. When you edit the value of slidePosn in the Model Variables table and then press Enter, you will see both the slider position and the number in the edit box change in response.

SliderVersion2 has the slider Minimum and Maximum values set to 50 and 60 respectively.

To operate SliderVersion2:

1 Select **SliderVersion2** and click on **Display**.

2 Drag the slider button back and forth between its limits.

3 Notice how jerkily it moves. Actually, it moves 2 inches in 10 steps of 0.2 inches.

   The values of slidePosn can only be changed in increments of 1 (a coarse resolution of 10%).

SliderVersion3 has the slider Minimum and Maximum values set to 5 and 6 respectively.

To operate SliderVersion3:

1 Select **SliderVersion3** and click on **Display**.

2 Drag the slider button back and forth between its limits. Notice how it jumps back and forth between 5 and 6. Actually, it moves 2 inches in a single step without being able to stop in between.

   The values of slidePosn can only be changed by an increment of 1 (a coarse resolution of 100%).

SliderVersion4 has the slider Minimum and Maximum reset to the default values of 0 and 100.

To operate SliderVersion4:

1 Select **SliderVersion4** and click on **Display**.

2 Drag the slider button back and forth between its limits. Notice how smoothly it moves again (back to the previous 2 inches in 100 steps of 0.02 inches).

3 The value of the new variable displayVal displayed in the Edit box is now calculated from the value of slidePosn by PEL code in a macro called sliderMoved. Click on the **Macros** tab to see the calculation:

   ```
   displayVal=5.0+slidePosn/100
   ```

Look at the Properties dialog for the slider Callbacks. We are using the Slider Moving callback to call the Macro sliderMoved. Click on the **Slider Moving** callback. In the right hand callback editing dialog sliderMoved is shown as the “Item to act on”, and Execute as the “action” to take.

As long as the slider is moving, the new value of slidePosn keeps changing in increments of 1 and the callback Slider Moving executes the sliderMoved macro which updates displayVal, which in turn automatically updates the number displayed in the Edit box.

But notice that editing the value of slidePosn in the Variables table has no effect on the value displayed in the Edit box. Similarly, editing the value of displayVal in the Variables table has no effect on the slider's position. This situation is corrected in SliderVersion5.

Look at the Properties dialog for the Edit Text box. The callback Variable Changed is used to call the macro valChanged. Click on the **Macros** tab to see the calculation:

```
slidePosn=(displayVal-5.0) * 100
```

This macro recalculates the value of slidePosn any time displayVal changes. Now if a process changes the value of displayVal, the position of the slider will be automatically updated.

The calling of the valChanged macro is not associated with the Edit Text box but with the value of displayVal changing—for any reason.

To view a demonstration of this, see sliderVersion6.

The Variable Changed callback is now in the Slider callbacks, instead of the Edit Text callbacks where we first put it. You could also put the Variable changed callback in the enclosing Table. But wherever you put it, you only need it in one place.

NOTE

The callback Variable Changed is associated with the variable it tracks, not the widget where it is specified.

The Variable Changed callback is so useful for making smooth working GUIs that we made it available, for your convenience, on every single widget.

Random Numbers (Example)

The random numbers example model is an example of a setup with a custom GUI that demonstrates how to use IC-CAP PEL functions to generate random numbers.

The random numbers example model file is located in the product directory:

```
${ICCAP_ROOT}/model_files/misc/random_example.mdl
```

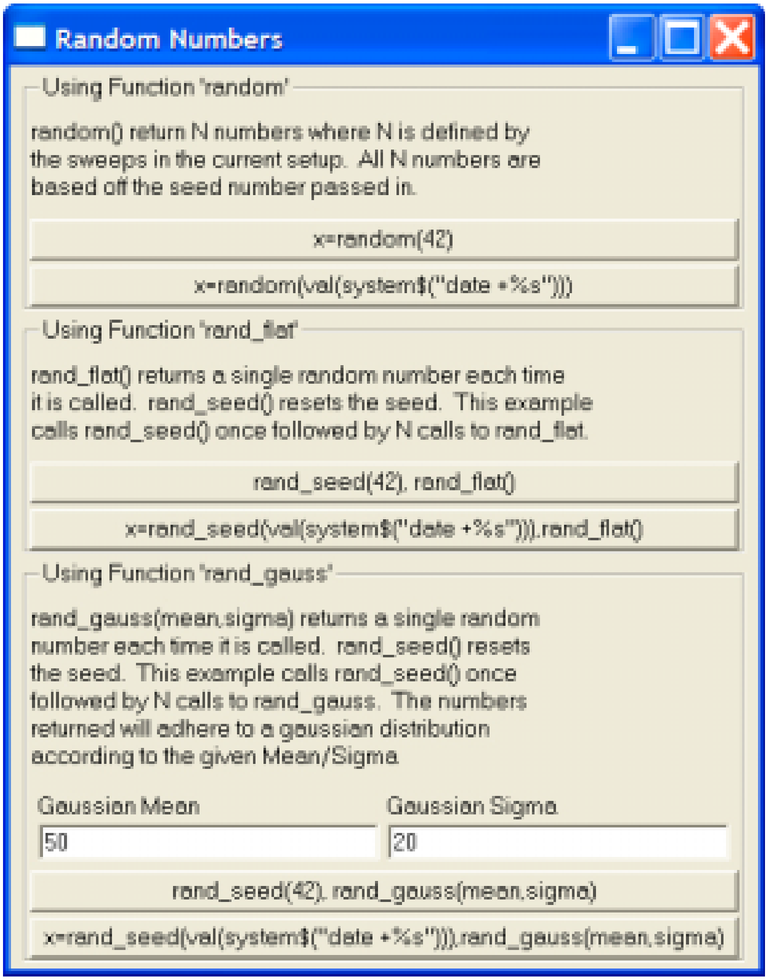
After you load the example model file, double-click on the *Random* model icon (by default the model is in AutoExecute mode) to display the following windows:

- Random Numbers Window
- Random Histogram Plot

Random Numbers Window

The *Random Numbers* window includes the following three sections that you can interactively modify by selecting either a fixed seed or varying seed value (seed = val(system$("date +%s"))) push buttons...):

- *Using Function 'random'*
- *Using Function 'rand_flat'* (uses function *rand_seed()* initially)
- *Using Function 'rand_gauss'* (uses function *rand_seed()* initially)



Random Numbers

Using Function 'random'

random() return N numbers where N is defined by the sweeps in the current setup. All N numbers are based off the seed number passed in.

x=random(42)

x=random(val(system$("date +%s")))

Using Function 'rand_flat'

rand_flat() returns a single random number each time it is called. rand_seed() resets the seed. This example calls rand_seed() once followed by N calls to rand_flat.

rand_seed(42), rand_flat()

x=rand_seed(val(system$("date +%s"))).rand_flat()

Using Function 'rand_gauss'

rand_gauss(mean,sigma) returns a single random number each time it is called. rand_seed() resets the seed. This example calls rand_seed() once followed by N calls to rand_gauss. The numbers returned will adhere to a gaussian distribution according to the given Mean/Sigma.

Gaussian Mean: 50

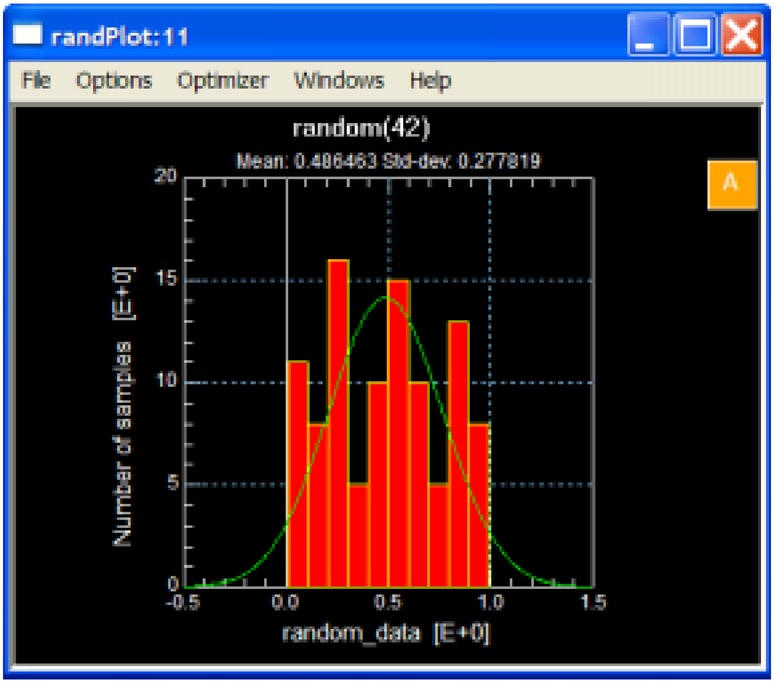
Gaussian Sigma: 20

rand_seed(42), rand_gauss(mean,sigma)

x=rand_seed(val(system$("date +%s"))).rand_gauss(mean,sigma)

Random Histogram Plot

Depending upon which push buttons you select in the Random Numbers window, you can interactively modify the random histogram plot (€*randPlot)*.



Details about how this example works:

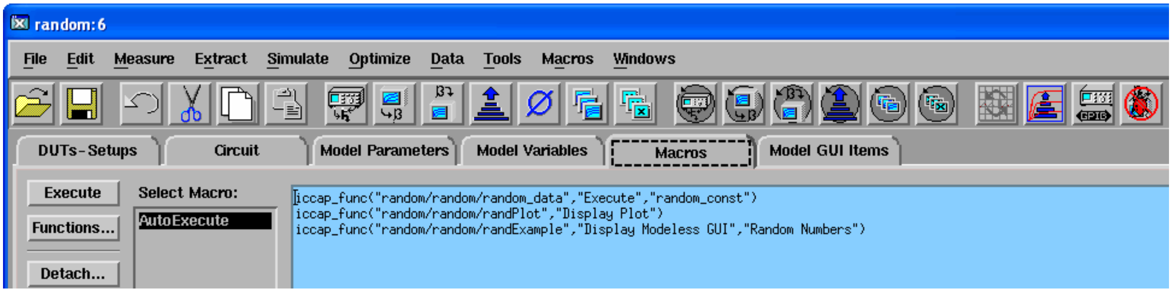
1 To create your own IC-CAP graphical user interface (GUI) items or see how the existing model's GUI items were created, you need to do one of the following before loading your IC-CAP *.mdl:

   - From the IC-CAP/Main window, select **Tools** > **Options** > **View GUI Pages**.
   - Or if you plan to do a lot of IC-CAP GUI programming you can add the *ICCAP_VIEW_GUI_PAGES=1* variable to your *iccap.cfg* file.

2 Load the *random_example.mdl* example model file into IC-CAP 2006B (or greater) Main window, then right-click and select **Edit.**

3 If you go to the example model file's *Macros* tab, you will see that the following random/random DUT Setup's transform, plot, and custom GUI get invoked when you first AutoExecute the example random model.

The following shows the example's *Macros* tab defining the *AutoExecute* macro:



```
iccap_func("random/random/random_data","Execute","random_const")
iccap_func("random/random/randPlot","Display Plot")
iccap_func("random/random/randExample","Display Modeless GUI","Random Numbers")
```

4 The heart of this example is the random/random DUTs-Setups’ PEL transform called *random_data*. Depending upon which buttons you select in the custom GUI window, the following key PEL functions can be executed:

- Using function *random*:

  ```
  return random(42)
  return random(val(system$("date +%s")))
  ```

- Using function *rand_flat* or *rand_gauss*:

  ```
  x=rand_seed(42)
  or
  x=rand_seed(val(system$("date +%s")))
  i=0
  while i < size(X)
     if randType=="rand_flat" then
        nums[i]=rand_flat()
     endif
     if randType=="rand_gauss" then
        nums[i]=rand_gauss(gaussMean, gaussSigma)
     endif
     i=i+1
  endwhile
  return nums
  ```

5 The random/random DUTs-Setups’ *Plots* tab defines the histogram plot that will be used with the custom GUI (Note: *Data-set* is set to *random_data* transform results; *Header* is set on the fly based upon which button you select).

6 Finally, the random/random DUTs-Setups’ *Setup GUI Items* tab shows how the custom GUI window was constructed. Notice that all the *Push Button* GUI items are defined to have a *Button Pressed* Callback action to execute the *random_data* transform via an *iccap_func()* statement.

13
Managing Data

This chapter describes the Data Manager (MDM) and its system variables and file structure. It also provides procedures for:

- Importing and Exporting data to an MDM file
- Exporting data to an MDIF (Model Data Interchange Format) file
- Exporting data to a dataset file (.ds)

Data Manager

The Data Manager (MDM) de-couples the measured or simulated data from the Model file. With this capability, you can save all the data for a collection of device geometries, temperatures, etc. into a single data management file (one file per setup). Subsets of this file (as required by specific extractions) can be imported into a given setup. With the Data Manager, you can import data measured from a non-IC-CAP environment into IC-CAP. You can use this data in a database (commercially available or internal), where you can use an external database program to manage huge amounts of data and then translate the data in the database into the MDM format for subsequent input into IC-CAP. Similarly, you can use the external database to control the exported MDM file.

Using the Data Manager, you can:

- Export measured or simulated data along with non-simulatable attributes (such as, length, width, temperature, etc.)
- Export transforms
- Import data from the MDM file as a whole
- Import data from parts of the MDM file
- Import data with sweep-orders that are different from what was measured, exported or stored in the data file

Managing your measured data consists of two major steps:

- Exporting the measured or simulated data in a setup to an MDM file
- Importing the data back into a pre-defined setup

Setting Data Management System Variables

You can set system variables to control data management. The system variables for data management are:

MDM_FILE_PATH Specifies the Data Manager file name path.

MDM_FILE_NAME Specifies the Data Manager file name.

MDM_EXPORT_COMMENT Specifies the text to be added as a comment at the top of an MDM file when exporting data. The text can include embedded IC-CAP variables such as val$(var-name), and predefined program variables DATE, NEWLINE, TAB, MODEL, DUT, SETUP. Some examples are:

MDM_STD_COM - Today's date : $DATE
MDM_EXPORT_COMMENT - val$(MDM_STD_COM)

MDM_EXPORT_COMMENT - MDM file exported on $DATE

MDM_EXPORT_COMMENT - Date: $DATE $SETUP L=val$(L) W=val$(W) where W and L are Model, DUT or Setup parameters/variables.

MDM_EXPORT_COMMENT_FILE Specifies the Data Manager comment file to be used when exporting data to an MDM file. The contents of the specified file are prepended to the MDM file.

MDM_EXPORT_XFORM_DATA When set to TRUE, exports all transform(s) data in a setup. Default is FALSE.

NOTE

MDM_ERROR is replaced by MDM_REL_ERROR.

MDM_REL_ERROR When IC-CAP reads values from an MDM file, it tries to match requested input values with data in the MDM. If the data does not exist in the MDM file, importing the data is not advised. However, due to roundoff errors, a tolerance must be assumed. IC-CAP uses the formula

MDM_REL_ERROR > (req - mdm)/req

as its acceptance test.

Default value is 1E-10

This value should rarely need to be adjusted, and can be adjusted for an individual Input. By assigning a value to MDM_REL_ERROR_<*name*> (where <*name*> is the name of the Input to which the tolerance is to be applied), specific control is possible. (See MDM_ZERO_TOL below.)

MDM_VALUES_LIST Specifies a space or comma separated list of Parameter or Variable names. The values of these Parameters/Variables are written to the MDM file. When an MDM file with values stored in it is imported or used to auto-create a setup, these Parameters/Variables are automatically reset to the values stored in the MDM. However, if a Parameter or Variable no longer exists in the scope of the setup being imported to, a variable will be created for that value in the Setup Variables Table.

MDM_XFORM_LIST Specifies a comma separated list of transform names that will be exported when this setup is exported. By using this variable, you may specify a subset of all transforms for export. In addition you may specify the mode, nodes, and other data. Each transform entry in the comma separated list will appear in the MDM file as an output. The actual values of each output type are shown below:

```
Mode   Values
----   ------------------------------------------------
V,N,U  <Name> <Mode> <+ Node> <- Node> <Unit> <Type>
I      <Name> <Mode> <To Node> <From Node> <Unit> <Type>
C,G    <Name> <Mode> <High Node> <Low Node> <Unit> <Type>
T      <Name> <Mode> <Node> <Pulse Param> <Unit> <Type>
S,H,Z,K,A,Y   <Name> <Mode> <Port 1> <Port 2> <AC Ground>
<Unit> <Type>
```

Examples:

```
   MDM_XFORM_LIST = calc_ic I C E, calc_ib I B E SMU2
   MDM_XFORM_LIST = Hcalc H, beta
```

Note, only the transform name is required. You may include as many extra entries per transform as desired.

This variable is only referenced if MDM_EXPORT_XFORM_DATA is true.

MDM_ZERO_TOL When IC-CAP reads values from an MDM file, it tries to match requested Input values with data in the MDM.

If the data does not exist in the MDM file, importing the data is not advised. However, due to roundoff errors, a tolerance must be assumed.

IC-CAP uses the formula

MDM_ZERO_TOL > (req - mdm) | req == 0 or mdm == 0

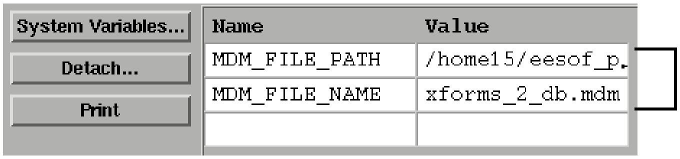
as its acceptance test.

Default value is 1E-30.

This value should rarely need to be adjusted, and can be adjusted for an individual Inputs. By assigning a value to MDM_ZERO_TOL_<*name*> (where <*name*> is the name of the Input to which the tolerance is to be applied), specific control is possible.

See MDM_REL_ERROR

MDM_HEADER_VERBOSE Specifies if the header of the MDM file includes comments describing each field. Default is FALSE.



Name	Value
MDM_FILE_PATH	/home15/eesof_p.
MDM_FILE_NAME	xforms_2_db.mdm

Set default variables for data management

Exporting Data

You can export various types of IC-CAP data for various uses.

- You can export your current model parameters to an MDIF (Model Data Interchange Format) file for use in the Advanced Design System (ADS) schematics.
- You can merge your current model parameters with your circuit definition and export to a file for use in an external simulator.
- You can export all the numeric data in your model file to a dataset file for use in ADS.
- You can export your measurement, simulation, and transformation data to an IC-CAP MDM file. This data (or subsets of this data) can then be read back into IC-CAP.

Exporting Data to MDIF Files

To export your current model parameters to an MDIF (Model Data Interchange Format) file for use in the ADS schematics:

1 Select **File > Export Data > Write Model MDIF**.

The File Save As dialog box appears.

2 In the File name field, type in a unique filename or select an existing filename.

3 Click **OK** to export the data to an MDIF file.

Exporting Data to Extracted Deck Files

To merge your current model parameters with your circuit definition and export to a file for use in an external simulator:

1 Select the DUT and setup.

2 Select **File > Export Data > Extracted Deck**.

The File Save As dialog box appears.

3 In the File name field, type in a unique filename or select an existing filename.

4 Click **OK** to export the data to an extracted deck file.

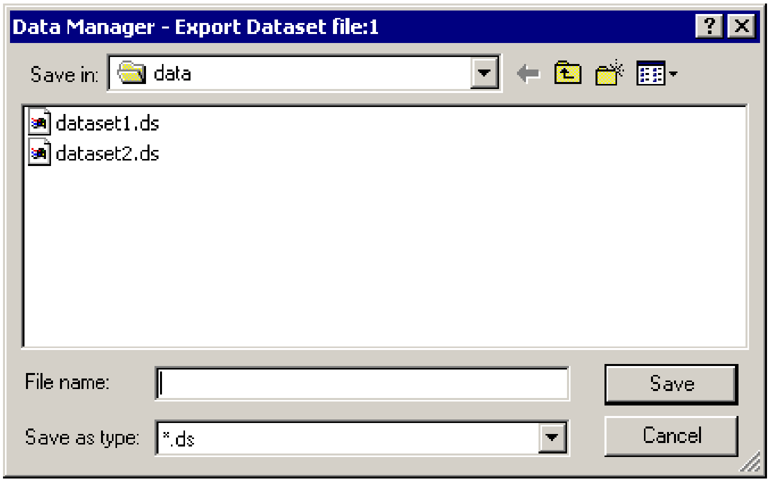
Exporting Data to Dataset (.ds) Files

Dataset files can be read by various tools in ADS. IC-CAP exports all data in the Parameter tables, Variable tables, Inputs, Outputs, and Transforms. You can export this data for an entire model, a single DUT, or a single Setup.

To export an entire model to a dataset file:

1 Choose **File > Export Data > All DUTs in Model**.

The *Data Manager - Export Dataset file* dialog box appears.



2 In the File name field, type in a unique filename or select an existing filename.

The *.ds* file extension is automatically appended to the filename.

3 Click **OK** to export the data to a dataset file.

If you selected an existing filename, it is quietly overwritten.

To export a single DUT to a dataset file:

1 Select a DUT from the DUTs-Setups folder.

2 Choose **File > Export Data > All Setups in Active DUT.**

The *Data Manager - Export Dataset file* dialog box appears.

3 In the File name field, type in a unique filename or select an existing filename.

The *.ds* file extension is automatically appended to the filename.

4 Click **OK** to export the data to a dataset file.

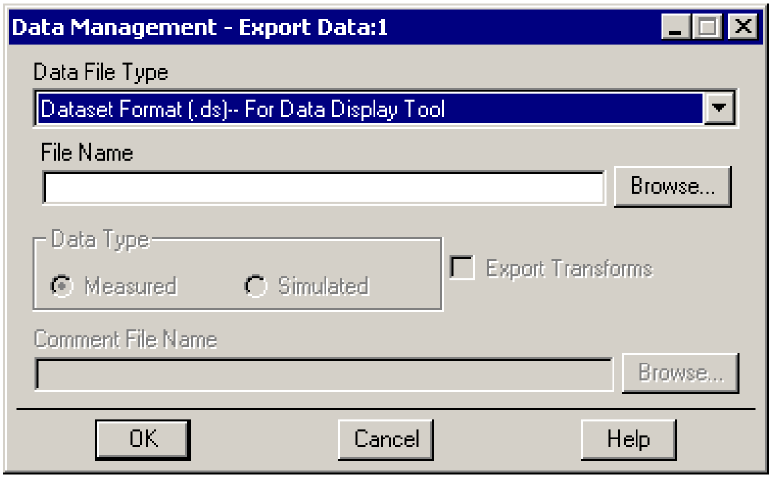
If you selected an existing filename, it is quietly overwritten.

To export a single setup to a dataset file:

1 Select a setup from the DUTs-Setups folder.

2 Choose **File > Export Data > Active Setup** or click **Export Data** on the Measure/Simulate folder.

The *Data Management - Export Data* dialog box appears.



Choose data file type

Enter data filename

Data Management - Export Data:1

Data File Type: Dataset Format (.ds)-- For Data Display Tool

3 Select Dataset Format (*.ds*) from the Data File Type drop-down list.

4 In the File Name field, type in a unique filename or click the Browse button to select an existing filename.

The *.ds* file extension is automatically appended to the filename.

5 Click **OK** to export the data to a dataset file.

If you selected an existing filename, it is quietly overwritten.

Exporting Data to MDM Files

The method you use to export data depends on the type of data you want to export. The type of data you can export can be attribute-independent measurements (such as the measured or simulated data in a setup) or geometry, temperature or other attribute-dependent data.

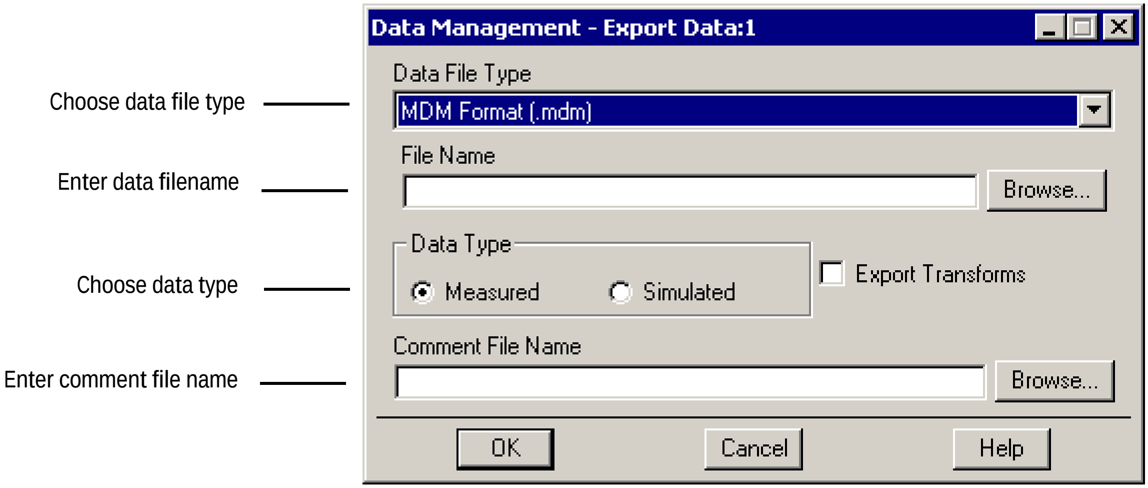
To export geometry, temperature or other attribute-dependent data, you must use a PEL macro. Some of the functions used for this purpose are discussed in “PEL Functions” on page 574. An example for exporting data in this manner is provided in “Exporting the Data Using a PEL” on page 581.

Attribute-Independent Measurements

To export attribute-independent measurements to an MDM file:

1. Select the DUT and setup.
2. Choose **File > Export Data > Active Setup** or click **Export Data** on the Measure/Simulate folder.

   The *Data Management - Export Data* dialog box appears.



Data Management - Export Data:1

Data File Type: MDM Format (.mdm)

File Name:

Data Type: ● Measured ○ Simulated

□ Export Transforms

Comment File Name:

3 Select MDM Format (.mdm) from the Data File Type drop-down list.

4 In the File Name field, type in a unique filename or click the Browse button to select an existing filename.

The .mdm file extension is automatically appended to the filename.

NOTE

The file path and file name are specified by the IC-CAP system variables MDM_FILE_PATH and MDM_FILE_NAME. When exporting, the program automatically sets these variables in the Setup Variables Table. Similarly, exporting transforms is controlled by MDM_EXPORT_XFORM_DATA.

5 Select the Data Type (Measured or Simulated). If appropriate, select **Export Transform**. Enter a comment file name if needed.

NOTE

Use separate files when exporting Measured or Simulated data to an MDM file, e.g., idvg_data_meas.mdm and idvg_data_sim.mdm.

You can specify the name of a comment file by setting the system variable MDM_EXPORT_COMMENT_FILE. When set, the program automatically prepends the comment file to the exported data file.

6 Click **OK** to export the data to an MDM file.

Attribute-Dependent Measurements

You can only export non-measurable/simulatable parameters (such as, length, width, or temperature) by using a PEL macro. Before exporting any data, you must register these parameters to the MDM file using one of the three PEL functions:

- *icdb_register_con_sweep*
- *icdb_register_lin_sweep*
- *icdb_register_list_sweep*
- *icbd_register_lsync_sweep*

Some important points to be noted while exporting or importing data:

- The setup used for importing must have all the input sweeps defined in the MDM file.
- You can import one or more output data.
- You can import all the data in an MDM file, or part of the data, by defining the input sweeps in the setup accordingly.

- You can import the data in any order, independent of the order in which the data is in the MDM file. However, if LSYNC data is in the MDM file, you must either access all sweeps with the same LSYNC master sweep or all with a CON sweep. You cannot request a LIN sweep (of more than 1 point) for the master and then CON sweeps for others. You can, however, access this data by specifing a LIN sweep for the master and LSYNC sweeps for the others. You must enter the same constant value for each entry in the LSYNC when making this request.
- You can export the data in all the transforms in the setup, by setting the IC-CAP variable *MDM_EXPORT_XFORM_DATA* to TRUE.
- Exported transforms are treated as outputs and can only be imported into an output.
- Both measured and simulated data cannot be exported into the same MDM file.
- Data can be exported into an MDM file only from one setup. That is, an MDM file cannot have data exported from multiple setups unless all setups have the identical sweep structure and PEL is used to build the MDM file by varying external parameter sweeps.
- Similarly, data from multiple MDM files cannot be imported into the same setup.
- Attribute-dependent measurements or parameters like geometry, temperature can be imported using a parameter mode type *P*.

PEL Functions

The PEL functions used for exporting the data are:

icdb_open / icdbf_open Opens a file for exporting measured data in IC-CAP's data management file format (.mdm). For details, see “icdb_open" and “icdbf_open" in the *Reference* manual.

icdb_close / icdbf_close Closes a file that has been opened with icdb_open/icdbf_open. For details, see “icdb_close" and “icdbf_close" in the *Reference* manual.

icdb_add_comment / icdbf_add_comment Writes an arbitrary comment string to the opened file. For details, see “icdb_add_comment" and “icdbf_add_comment" in the *Reference* manual.

icdb_export_data / icdbf_export_data Exports the measured or simulated data from the specified setup to the opened file. Header information containing current information about the values of the registered sweep parameters is automatically appended to the file. For details, see “icdb_export_data" and “icdbf_export_data" in the *Reference* manual.

icdb_get_sweep_value / icdbf_get_sweep_value Returns the current value of the specified user sweep at any point in the export loop. For details, see “icdb_get_sweep_value" and “icdbf_get_sweep_value" in the *Reference* manual.

icdb_register_con_sweep / icdbf_register_con_sweep Creates a CON type sweep of an arbitrary parameter in the exported file. Intended primarily to create sweeps of parameters that cannot be swept during a measurement. Returns the total number of points in all the registered sweeps. For details, see “icdb_register_con_sweep" and “icdbf_register_con_sweep" in the *Reference* manual.

icdb_register_lin_sweep / icdbf_register_lin_sweep Creates a LIN type sweep of an arbitrary parameter in the exported file. Intended primarily to create sweeps of parameters that cannot be swept during a measurement. Returns the total number of points in all the registered sweeps. For details, see “icdb_register_lin_sweep" and “icdbf_register_lin_sweep" in the *Reference* manual.

icdb_register_list_sweep / icdbf_register_list_sweep Creates a LIST type sweep of an arbitrary parameter in the exported file. Intended primarily to create sweeps of parameters that cannot be swept during a measurement. Returns the total number of points in all the registered sweeps. For details, see “icdb_register_list_sweep" and “icdbf_register_list_sweep" in the *Reference* manual.

icdb_register_lsync_sweep / icdbf_register_lsync_sweep Creates a LSYNC type sweep of an arbitrary parameter in the exported file. Intended primarily to create sweeps of parameters that cannot be swept during a measurement. Returns the total number of points in all the registered sweeps. For details, see icdb_register_lsync_sweep and icdbf_register_lsync_sweep in the *Reference* manual.

Importing Data

You can import data (.mdm files) to the active setup, to all setups in an active DUT, or to all DUTs in a model.

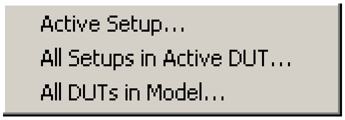
NOTE

Dataset files (.ds) can *NOT* be imported into a model file. Dataset files can be exported from a Model window and displayed in the ADS Data Display window.

To import data into a single setup:

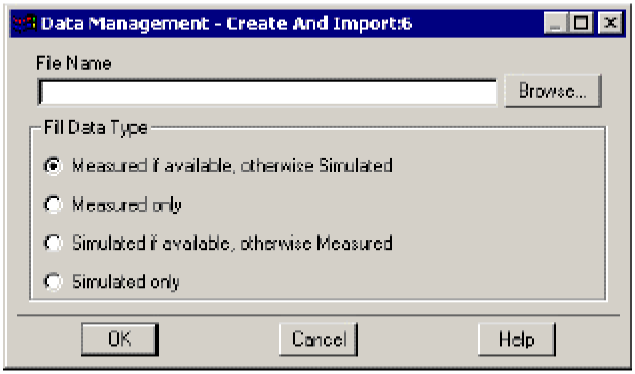
1. Select the DUT and setup.
2. Select **File > Import Data** and choose **Active Setup**.

Choose setup type



Alternatively, you can select **Measure / Simulate** and click **Import Data**. Import Data...

3. Enter the name of the file directly into the field corresponding to the setup, by typing in or by using the File Browser.


Data Management - Create And Import:6

File Name

Browse...

Fill Data Type

● Measured if available, otherwise Simulated

○ Measured only

○ Simulated if available, otherwise Measured

○ Simulated only

OK Cancel Help

4. Select a fill data type.

- **Measured if available, otherwise Simulated:** Outputs of type M or B receive mdm data in their measured array. Outputs of type S receive mdm data in their simulated array.
- **Measured only:** Outputs of type M or B receive data in their measured array. Outputs of type S do not receive any data.
- **Simulated if available, otherwise Measured:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M receive mdm data in their measured array.
- **Simulated only:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M do not receive any data.

5 Click **OK**. The data is now imported into the outputs of the setup.

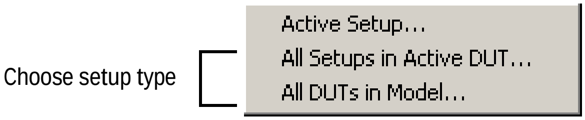
To import data into all setups in an active DUT or all DUTs in the model:

1 Select the DUT.

2 Select **File > Import Data** and choose one of the following:

All Setups in Active DUT

All DUTs in Model



NOTE

All Setups in Active DUT imports an ASCII-based MDM data file of a pre-determined format, to the selected DUT; All DUTs in Model imports an ASCII-based MDM data file of a predetermined format, to all setups in all DUTs of the current model.

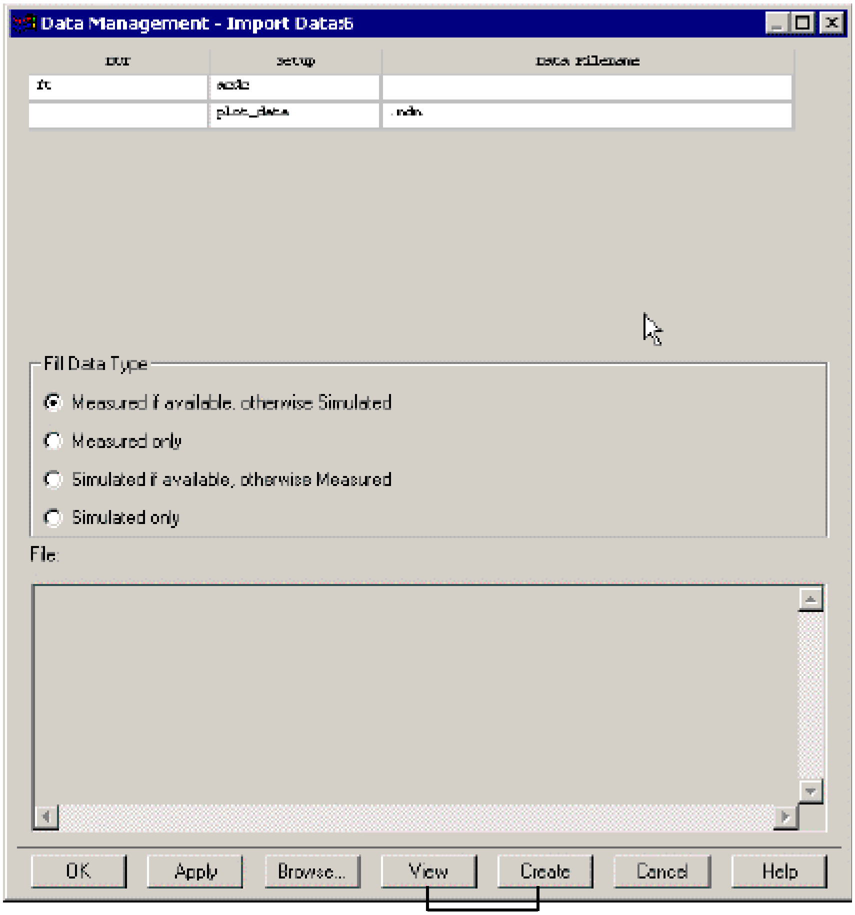
NOTE

ICCAP_FUNC can be used to import data from MDM files into all setups in a DUT, and into all DUTs in a model as shown in these examples:

```
iccap_func(<model/DUT>,“Import Data”) All Setups in Active
          DUT
iccap_func(<model>,“Import Data”) All DUTs in Model
```

The dialog box that appears has three columns and a Fill Data Type selection area. The first column shows the DUT name, the 2nd column the setup names and the 3rd column the MDM file to be imported. By default, the file represented by the variables MDM_FILE_PATH and MDM_FILE_NAME are shown in the 3rd column.

3 Enter the name of the file directly into the field corresponding to the setup, by typing in or by using the File Browser.



Data Management - Import Data:6

DUT	Setup	Data Filename
ft	acdc	
	plot_data	.mdm

Fill Data Type

- ● Measured if available, otherwise Simulated
- ○ Measured only
- ○ Simulated if available, otherwise Measured
- ○ Simulated only

File:

Click buttons to view data in file or create input sweeps and outputs automatically

4 Select a fill data type.

- **Measured if available, otherwise Simulated:** Outputs of type M or B receive mdm data in their measured array. Outputs of type S receive mdm data in their simulated array.
- **Measured only:** Outputs of type M or B receive data in their measured array. Outputs of type S do not receive any data.
- **Simulated if available, otherwise Measured:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M receive mdm data in their measured array.
- **Simulated only:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M do not receive any data.

Viewing an MDM File Header: To view the header of an MDM data file, select the Setup cell in the dialog box, enter the corresponding MDM filename, then click View.

Automatically creating the inputs- and outputs-based on an MDM file: You can create the input sweeps and outputs in a newly-created setup from the header of an MDM file. To create the inputs and outputs, first create a new setup, select the setup name in the dialog box, enter the corresponding MDM filename, and then click **Create**. Clicking Create does not import data—it simply creates the inputs/outputs based on the MDM file. To create the inputs/outputs and import data in one step, select **Measure / Simulate** and click **Import Create**.

NOTE

You can delete the inputs and outputs from a setup by using ICCAP_FUNC with Import Delete as the second argument, as in this example

```
ICCAP_FUNC(<setup>, “Import Delete”)
```

5 Click **Apply** or **OK**.

Clicking Apply imports data into all setups with filenames. Clicking OK does the same but also closes the dialog box. Clicking Cancel closes the dialog box without importing any data, but does not cancel any setups created using the Create button.

Examples

The data management example is located in the product directory:

```
${ICCAP_ROOT}/examples/demo_features/3_MEAS_ORGANIZE_n_
VERIFY_DATA/2_mdm_basics/2_xforms_2_db.mdl
```

The example model has two setups in the DUT dc_at_temps: fgummel_exp and fgummel_imp_all.

Exporting the Data Using a PEL

In the example, a PEL transform is used to export non-measurable/simulatable attribute-dependent data. The exporting data example uses the setup fgummel_exp. The PEL transform, *mdm_export*, is available in the setup and in this example, exports temperature.

In the macro (code listing below) line 32 registers a non-measurable/simulatable attribute called *Temp*. The sweep type is *LIN* with a start, stop and total number of points of TSTART = –50, TSTOP = 50 and TNUM = 11. The sweep order is *1*. This function also returns the number of points. In lines 50 through 69, the setup (defined by the variable *path*) is simulated and exported (line 67). The MDM file exported in the example has 11 data-blocks corresponding to 11 different simulations/measurements that were made at the different temperatures.

```
! Here is a simple data export example which illustrates the use of the icdb_xxx
! commands.  Although simulated data is exported in this example, it is easy
! to see how a simple replacement of the "Simulate" command inside the main
! export loop with a "Measure" would yield a data file with temperature dep-
! endent Gummel-Poon measurement data.

MDM_EXPORT_XFORM_DATA=1    ! flag to also export transform data

path = "/xforms_2_db/dc_at_temps/fgummel_exp"            !path to the export setup

LINPUT "enter filename for database file",VAL$(filename),filename

file_exists=SYSTEM("ls "&VAL$(filename))
IF file_exists==0 THEN LINPUT "--WARNING--
      overwriting existing file '"&VAL$(filename)&"'","ok",dummy


! This command opens the file that the data will be stored in.
! The function icdb_open returns nothing.

dummy=icdb_open(VAL$(filename))

! This command adds a comment line to the data file

dummy = icdb_add_comment(" Data management file for forward gummel data")
dummy = icdb_add_comment("")

! The next data base function tells IC-CAP to define a linear sweep over
! temperature.  The function arguments are: sweep order, sweep start,
! sweep stop, number of points and sweep name.  The function returns the
! cumulative number of points in all the registered user sweeps.  The value
! "num_pts" is useful for setting up the export loop.

!Note:  In this example the icdb_register_lin_sweep and icdb_register_list_sweep
!       commands will generate the same file.  One of the command must remain
!       commented out.

num_pts=icdb_register_lin_sweep(1, TSTART, TSTOP, TNUM, "Temp")

! The next data base function tells IC-CAP to define a list sweep over
! temperature.  The function arguments are: sweep order, sweep name,
! and IC-CAP variable array name.  The function returns the cumulative
! number of points in all the registered user sweeps.  The value "num_pts"
! is useful for setting up the export loop.

!num_pts=icdb_register_list_sweep(1, "Temp", "TEMP_LIST")

ICCAP_FUNC(path&"/ibicvsvb", "Display Plot")
ICCAP_FUNC(path&"/ibicvsvb", "Autoscale OnOff")
```

Figure 41 Example: Exporting Data Using a PEL

```
! Here is the "export" loop.  The number of loop iterations should be consistent
! with the dimension of the user sweep space.  For example, if a LIST sweep with
! 10 points is defined and a LIN sweep with 5 points is defined, then 5*10 = 50
! iterations of the export loop must be completed.  The export functions will
! error out if too many iterations are attempted.  However, if too few iter-
! ations are performed, the problem will go undetected until an import is tried.
! Use the return value of the icdb_register_xxx_sweep to set the maximum number
! of iterations.  This should minimize the possibility of any errors.

count=0
while count < num_pts

  ! Once a sweep has been registered, IC-CAP automatically internally updates
  ! the values of all the user sweep values after each call to icdb_export_data.
  ! The function icdb_get_sweep_value provides useful means of getting the
  ! current value of any sweep variable.  The function arguments are the index
  ! of the desired sweep value and the sweep name.

  TEMP = icdb_get_sweep_value(count, "Temp")

  ! Now that we've updated the temperature for the  simulator, we can
  ! re-simulate to compute a new set of data.

  ICCAP_FUNC(path, "Simulate")

  ! This call exports the simulated data ("S") in the current setup out
  ! to the file.  Measured data can be exported by specifying "M" for the
  ! second argument.  The first call to icdb_export_data writes the header
  ! information as well as the setup data.  The function icdb_export_data
  ! returns nothing.  To reference a setup other than the one under which
  ! a transform is executing, the full path should be passed in as the
  ! first arguement of icdb_export_data.

  dummy = icdb_export_data(".", "S")

  count=count+1
end while

! This function closes the file opened with icdb_open.  Note, that only one
! file can be opened at a time.  The first opened file must be closed before
! a second one is opened.  The function icdb_close returns nothing.

dummy=icdb_close()

ICCAP_FUNC(path&"/ibicvsvb", "Autoscale OnOff")


LINPUT "Run Import Data in Setup fgummel_imp_all
        and load the beta curves from the data base file too",dummy

! The following code guarantees that the output of this transform will not be
! exported to the mdm file if MDM_EXPORT_XFORM_DATA is set to TRUE.

COMPLEX tmp[1]
RETURN tmp
```

Figure 41 Example: Exporting Data Using a PEL (continued)

Importing the Data

The importing data example uses the setup *fgummel_imp_all*. The setup has the non-simulatable attribute *Temp* defined as a Parameter mode. You can import the data using the *Import Data* button in the setup. If you wish, you can import a slice of the data. For example you can import data at temperatures 50, 100, 150 for *vb* between 300 mV and 700 mV.

MDM File Structure

The MDM file format provides the following advantages:

- ASCII based
- Table-based, row-column format with column header lines that make reading easy—includes a list of the inner-most independent variables.
- All data tables have identical shape. A header at the top of the file provides an outline of all the data in the file. After the header has been parsed, the location of any data group can be computed quickly, permitting rapid location of arbitrary data groups scattered throughout the file. Comment lines are denoted by the exclamation character (!). The file extension for the data files is *.mdm* (measured data management).

File Header Format

The file header contains all the relevant information about the inputs sweeps as well as a listing of all the outputs. The header information begins with the BEGIN_HEADER keyword and ends with the END_HEADER keyword. The header information is contained in one of four separate sections: ICCAP_INPUTS, ICCAP_OUTPUTS, USER_INPUTS, and ICCAP_VALUES. The ICCAP_INPUTS and ICCAP_OUTPUTS sections contain information that would be contained in an IC-CAP setup. These portions of the header are mandatory. The optional USER_INPUTS section contains sweep information on variables that can't be swept in a traditional IC-CAP setup (such as, Length, Width and Temperature). The optional ICCAP_VALUES section contains parameter and variable data. When IC-CAP reads the MDM file, the parameter or variable values are reset to the value in the file. If the parameter or variable does not exist, IC-CAP creates a variable by that name in the setup where the file is being read. The header structure is as follows:

```
BEGIN_HEADER
  USER_INPUTS
     <user_input_name_1> <sweep_type> [<sweep_type_options_list>]
     .
     .
     <user_input_name_n> <sweep_type> [<sweep_type_options_list>]
  ICCAP_INPUTS
     <input_name_1> <mode> [<mode_options_list>] <sweep_type> [<sweep_type_options_list>]
     .
     .
     <input_name_m> <mode> [<mode_options_list>] <sweep_type> [<sweep_type_options_list>]
  ICCAP_OUTPUTS
     <output_name_1> <mode> [<mode_options_list>] <unit> <compliance> <type>
     .
     .
     <output_name_p> <mode> [<mode_options_list>] <unit> <compliance> <type>
  ICCAP_VALUES
    <value_name_1> <value_1>
    .
    .
    <value_name_q> <value_q>
END_HEADER
```

where,

<*input_name*> is a unique but arbitrary name for a user sweep or an IC-CAP sweep

<*mode*> is set to one of the following values: V, I, F, T, P, U, W (inputs) or V, I, C, G, T, S, H, Z, Y, K, A (outputs)

<*mode_options_list*> is a list of fields that depend on the <*mode*>. The following shows the fields for each <*mode*>:

Table 55 *<mode_options_list>*

<mode>	**Fields**
Inputs	
V, U	<+ Node><- Node> <Unit> <Compliance>
I	<To Node><From Node> <Unit> <Compliance>
P	<Param Name><Unit>
W	<+ Node><- Node> <dBm(d) / Watts(W)> <Resistance> <Fund> <Unit> <Compliance>
F, T	(no options)

Table 55 *<mode_options_list>* (continued)

<mode>	Fields
Outputs	
V, N, U	<+ Node><- Node>
I	<To Node><From Node>
C, G	<High Node><Low Node>
T	<Node><Pulse Param>
S, H, Z, K, A, Y	<Port 1><Port 2><AC Ground>

<*sweep_type*> is set to one of the following IC-CAP sweeps: LIN, LOG, SYNC, LIST, CON, AC, HB, EXP, PULSE, PWL, SFFM, SIN, TDR, SEG

<*sweep_options_list*> is a list of fields that depend on the <*sweep_type*>. The following shows the fields for each <*sweep_type*>:

Table 56 <*sweep_options_list*>

<*sweep_type*>	Fields
LIN	<sweep order> <start> <stop> <number of points> <step size>
LOG	<sweep order> <start> <stop> <number of points> <sweep scale (D or O)> <total number of points>
SYNC	<ratio> <offset> <master sweep>
LIST	<sweep order> <n = number of values> <value1> <value2> ... <valueN>
CON	<value>
AC	<magnitude> <phase>
HB	<sweep order> <value> <order>
EXP	<initial value> <pulsed value> <rise delay> <rise const> <fall delay> <fall const>

Table 56 *<sweep_options_list>* (continued)

<sweep_type>	Fields
PULSE	<initial value> <pulsed value> <delay time> <rise time> <fall time> <pulse width> <period>
PWL	<number of pairs> <time 1> <value 1> ... <time 7> <value 7> <start time> <repeat times>
SFFM	<offset> <amplitude> <carrier frequency> <modul index> <signal frequency>
SIN	<offset v> <amplitude> <frequency> <delay time> <damp factor> <phase>
TDR	<initial value> <pulsed value> <delay time> <rise time> <fall time> <pulse width> <period> <resistance>
SEG	<sweep order> <number of segments> <number of points> <value 1> ... <value 10>

<unit> is set to the IC-CAP instrument unit associated with the output (use DEFAULT for unspecified)

<compliance> is the compliance limit for the output (use DEFAULT for unspecified)

<type> is a single character—M, S, or B. M indicates that the output, if created from an MDM file, is for measured data only. S indicates simulated data only and B indicates either.

The data in the ICCAP_OUTPUTS block is restricted to data that can exist in a single IC-CAP setup. For example, since forward and reverse gummel DC measurements on a bipolar device are measured with different sweeps, the file headers for each of these would be incompatible. In order to make all the data within a file importable into a standard IC-CAP setup, two separate files (one for each setup) would be required to store this data.

File Data Format

The data within a file is organized into multiple groups of tabular data. Each group of tabular data is arranged in columns representing the innermost sweep data and its associated dependent data. The innermost sweep is always the first column. Any inputs with SYNC sweep type and with the innermost sweep as its master are listed next. Dependent data may be either real or complex, depending on the following rules:

- C, G, and T output modes are always real—therefore require only 1 column.
- TwoPort (S, H, Z, Y, K, or A) output modes are always 2-port complex—therefore require 8 columns.
- V and I output modes depend on the sweep types of the inputs specified in the setup. If *any* of the inputs in a setup have sweep type AC or HB, V and I output modes require 2 columns—one for real and one for imaginary data. For all other setups, V and I output modes require a single real column.
- Any output mode not covered in the preceding rules have 2 columns for complex data.

For real data, the tabular data is structured as follows:

```
<input_name> <output_name 1> <output_name 2> …
<output_name n> input1 (output 1) 1 (output 2) 1 …
(output n) 1
.
.
.
inputm (output 1) m (output 2) m … (output n) m
```

For matrix data (such as, multiport data), the data format for a single output is structured as follows:

```
<in_name> R:<out_name>(1,1) I:<out_name>(1,1) …
   R:<out_name>(n,m) I:<out_name>(n,m)
input1 real(output 1,1)1 imag(output 1,1)
   1 … real(output n,m) 1 imag(output n,m) 1
.
.
.
inputk real(output 1,1) k imag(output 1,1)
   k … real(output n,m) k imag(output n,m) k
```

Each group of tabular data is preceded by a list of each remaining input name (i.e., all inputs except the innermost sweep) and its current value for the data group in question.

Each group begins with the keyword BEGIN_DB and ends with the END_DB keyword. Note that the IC-CAP inputs always vary faster than the user inputs. The file structure of each group is as follows:

```
BEGIN_DB
<user_input_1> <user_value_1> … <user_input_n>
<user_value_n> <iccap_input_2> <iccap_value_2> …
<iccap_input_m> <iccap_value_m>
[tabular data goes here]
END_DB
```

Since the import function in IC-CAP determines the size of the dataset from the header information alone, the data following the header must be consistent with the description in the header. After parsing the header, IC-CAP knows exactly how many data blocks should be contained in the file, as well as the number of lines occupied by a single data block. If any data blocks are missing, or if the number of lines within a data block are inconsistent with the header, the data import fails.

File Examples

This section contains three example *.mdm* files that were generated with IC-CAP's export functions.

Example 1: Forward Gummel Data

The Forward Gummel Data example (code listing below) shows how some forward Gummel data can be stored in an .mdm file. In this example, the variables vb and Temp are being swept. The variable vb is the inner most (fastest varying) sweep. Since *vc* is synchronized to *vb*, the values of *vc* must appear in each data block. The actual ordering of the sweeps within the USER_INPUTS, ICCAP_INPUTS or ICCAP_OUTPUTS block, as well as the ordering of the variables in the secondary header (USER_VAR, ICCAP_VAR), is immaterial.

```
! VERSION = 6.00
BEGIN_HEADER
 ICCAP_INPUTS
  vb    V  B GROUND SMU1 0.01 LIN  1 0.33 0.83 51 0.01
  ve    V  E GROUND GND  0.1  CON  0
  vc    V  C GROUND SMU2 0.2  SYNC 1 0 vb
 ICCAP_OUTPUTS
  ib    I  B GROUND SMU1 B
  ic    I  C GROUND SMU2 B
END_HEADER

BEGIN_DB
 ICCAP_VAR ve  0

 #vb      vc      ib              ic
  0.33    0.33    4.87574e-011    4.67239e-010
  0.34    0.34    5.77546e-011    6.85381e-010
  0.35    0.35    6.86361e-011    1.00538e-009
  0.36    0.36    8.18976e-011    1.47476e-009
  0.37    0.37    9.82047e-011    2.16325e-009
  0.38    0.38    1.18461e-010    3.17307e-009
  0.39    0.39    1.4391e-010     4.65412e-009
  0.4     0.4     1.76281e-010    6.82619e-009
  0.41    0.41    2.18003e-010    1.00115e-008
  0.42    0.42    2.72518e-010    1.46826e-008
  0.43    0.43    3.44737e-010    2.15321e-008
  0.44    0.44    4.41716e-010    3.15754e-008
  0.45    0.45    5.73646e-010    4.63009e-008
  0.46    0.46    7.55304e-010    6.78903e-008
  0.47    0.47    1.0082e-009     9.95413e-008
  0.48    0.48    1.36373e-009    1.45941e-007
  0.49    0.49    1.86784e-009    2.13956e-007
  0.5     0.5     2.58788e-009    3.13654e-007
  0.51    0.51    3.62273e-009    4.59782e-007
  0.52    0.52    5.11772e-009    6.7395e-007
  0.53    0.53    7.28674e-009    9.87821e-007
  0.54    0.54    1.04447e-008    1.44778e-006
  0.55    0.55    1.50555e-008    2.12177e-006
  0.56    0.56    2.18032e-008    3.10934e-006
  0.57    0.57    3.16961e-008    4.55624e-006
  0.58    0.58    4.62218e-008    6.67595e-006
  0.59    0.59    6.75746e-008    9.78105e-006
  0.6     0.6     9.89915e-008    1.43291e-005
  0.61    0.61    1.45248e-007    2.09898e-005
  0.62    0.62    2.1339e-007     3.07431e-005
  0.63    0.63    3.13803e-007    4.50218e-005
  0.64    0.64    4.61803e-007    6.59204e-005
  0.65    0.65    6.79941e-007    9.64972e-005
  0.66    0.66    1.0014e-006     0.000141213
  0.67    0.67    1.47494e-006    0.000206561
  0.68    0.68    2.17197e-006    0.000301974
  0.69    0.69    3.19683e-006    0.000441095
  0.7     0.7     4.70103e-006    0.000643565
  0.71    0.71    6.9031e-006     0.00093743
  0.72    0.72    1.01148e-005    0.00136231
  0.73    0.73    1.47739e-005    0.00197331
  0.74    0.74    2.14824e-005    0.00284535
```

Figure 42 Example: Forward Gummel Data

```
  0.75    0.75    3.10425e-005    0.00407717
  0.76    0.76    4.44792e-005    0.00579344
  0.77    0.77    6.30296e-005    0.0081425
  0.78    0.78    8.80775e-005    0.0112877
  0.79    0.79    0.000121025     0.015391
  0.8     0.8     0.000163119     0.0205921
  0.81    0.81    0.000215278     0.0269887
  0.82    0.82    0.000277987     0.0346245
  0.83    0.83    0.000351277     0.0434891
END_DB
```

Figure 42 Example: Forward Gummel Data (continued)

Example 2: Saving Transformed Data

The Saving Transformed Data example (code listing below) shows how some forward Gummel data can be stored along with additional data *beta*, which has been computed from the measured data. Since the beta data has been computed rather than measured, information such as node connections is not relevant. The IC-CAP import functions accept outputs in the ICCAP_OUTPUTS block in the abbreviated form shown in the example for beta, as well as in the standard form for measured quantities, such as *ib* and *ic*. In order to differentiate between real data (such as, current, voltage, and capacitance) and 2-port data (such as, s-parameters) non-measured quantities must have one of two mode types listed after the variable name:

- Real quantities should use V as the generic real data mode type.
- Two-port quantities should use S as the generic 2-port mode type.

```
! VERSION = 6.00
BEGIN_HEADER
 ICCAP_INPUTS
  vb      V  B GROUND SMU1 0.01 LIN    1    0.3    0.8    51    0.01
  ve      V  E GROUND GND 0.1 CON      0
  vc      V  C GROUND SMU2 0.2 SYNC    1 0 vb
 ICCAP_OUTPUTS
  ib      I  B GROUND SMU1 B
  ic      I  C GROUND SMU2 B
  beta    U
END_HEADER

BEGIN_DB
 ICCAP_VAR ve    0

  vb      vc      ib              ic              R:beta(1,1)    I:beta(1,1)
  0.3     0.3     2.97359e-011    1.48082e-010    4.9799         0
  0.31    0.31    3.50033e-011    2.17173e-010    6.20438        0
  0.32    0.32    4.12693e-011    3.18536e-010    7.71848        0
  0.33    0.33    4.87574e-011    4.67239e-010    9.58294        0
  0.34    0.34    5.77546e-011    6.85381e-010    11.8671        0
  0.35    0.35    6.8636e-011     1.00538e-009    14.648         0
.
.
.
  0.73    0.73    1.47739e-005    0.00197331      133.567        0
  0.74    0.74    2.14824e-005    0.00284535      132.45         0
  0.75    0.75    3.10425e-005    0.00407717      131.342        0
  0.76    0.76    4.44792e-005    0.00579344      130.25         0
  0.77    0.77    6.30296e-005    0.0081425       129.185        0
  0.78    0.78    8.80775e-005    0.0112877       128.156        0
  0.79    0.79    0.000121025     0.015391        127.172        0
  0.8     0.8     0.000163119     0.0205921       126.24         0
END_DB
```

Figure 43 Example: Saving Transformed Data

Example 3: 2-Port Data

The 2-Port Data (code listing below) example shows how some 2-port data is stored in an *.mdm* file. This a simple example with only one data block. Multiple blocks of 2-port data are easily handled using the same approach as illustrated in the Forward Gummel data example.

```
BEGIN_HEADER
  ICCAP_INPUTS
    freq F LIN 1 1e+09 2e+10 20
    vd V D 0 CON 2
    vg V G 0 CON 0
    vs V S 0 CON 0
  ICCAP_OUTPUTS
    s S G D 0
END_HEADER

BEGIN_DB
  ICCAP_VAR vd 2
  ICCAP_VAR vg 0
  ICCAP_VAR vs 0

freq     R:s(1,1)   I:s(1,1) R:s(1,2)   I:s(1,2)   R:s(2,1)  I:s(2,1) R:s(2,2)   I:s(2,2)
1e+09    0.952765  -0.224466 0.00250463 0.0181728 -9.12695   4.09933  0.826825  -0.133475
2e+09    0.823887  -0.4131   0.00927818 0.0338053 -8.015     7.68155  0.721175  -0.242516
3e+09    0.643912  -0.545612 0.0185764  0.0453976 -6.51225  10.4343   0.572444  -0.313754
4e+09    0.445355  -0.620653 0.0285869  0.0527403 -4.93111  12.3018   0.406526  -0.347128
5e+09    0.251996  -0.648786 0.038045   0.0564729 -3.48076  13.4015   0.242814  -0.35067
6e+09    0.0767755 -0.643938 0.0463266  0.0575208 -2.25457  13.9149   0.0923492 -0.334344
7e+09   -0.0755289 -0.618357 0.0532653  0.0567453 -1.26671  14.0161  -0.0403101 -0.306557
8e+09   -0.204864  -0.58103  0.0589388  0.0548157 -0.492037 13.8452  -0.154513  -0.273196
9e+09   -0.313372  -0.537902 0.0635211  0.0522049  0.10741  13.5041  -0.251554  -0.237929
1e+10   -0.403927  -0.492642 0.0672037  0.0492289  0.569153 13.0635  -0.333484  -0.202836
1.1e+10 -0.479404  -0.447392 0.0701604  0.046091   0.925146 12.5706  -0.402492  -0.16899
1.2e+10 -0.542383  -0.403328 0.0725371  0.0429182  1.20076  12.0562  -0.460617  -0.136866
1.3e+10 -0.595055  -0.361041 0.0744508  0.0397875  1.41544  11.54    -0.509644  -0.106604
1.4e+10 -0.639233  -0.320776 0.0759938  0.0367433  1.5838   11.034   -0.551083  -0.0781668
1.5e+10 -0.676397  -0.282583 0.0772385  0.0338096  1.71675  10.5454  -0.586187  -0.0514345
1.6e+10 -0.707743  -0.246403 0.0782409  0.0309975  1.82244  10.0781  -0.615993  -0.0262538
1.7e+10 -0.734241  -0.212124 0.0790451  0.0283101  1.90696   9.63396 -0.641346  -0.00246532
1.8e+10 -0.756675  -0.179609 0.0796856  0.0257459  1.97487   9.21352 -0.662944   0.0200831
1.9e+10 -0.775683  -0.148716 0.0801897  0.0233003  2.02964   8.81648 -0.681355   0.0415301
2e+10   -0.791784  -0.119307 0.0805791  0.0209675  2.07389   8.44203 -0.69705    0.0619992
END_DB
```

Figure 44 Example: 2-Port Data

A
Menu Descriptions

Main Window

File Menu

New

Creates a blank, untitled model file and displays a symbol for it in the work area of the Main window. Highlight the text below the symbol and type the desired name for the new model. Double-click to open the Model window.

Open

Enables you to open an existing model file.

Examples

Enables you to quickly open a model file in the examples directory

Edit

Opens the selected Model window (select the model symbol, then the command).

Auto Execute

Runs the AutoExecute macro in the selected model if it is declared or opens the selected Model window if AutoExecute in not declared.

Save As

Enables you to save one or more of the models currently in memory.

Change Directory

Enables you to change the directory to which data will be saved during the current session.

<File Names>

Contains list of most recently loaded files.

Exit

Exits IC-CAP. The Save As dialog box appears enabling you to save any models currently in memory before exiting.

Edit Menu

Undo

Undoes the most recent Cut, Copy, or Paste command.

Cut

Cuts the selected model to a buffer from which it can be pasted. This enables you to load and view another model by the same name—without overwriting the first one—and choose between them, pasting the one you cut if desired.

Copy

Copies the selected model to a buffer from which it can be pasted. By copying and pasting, you can have one or more copies of a given model open at the same time.

Paste

Pastes the contents of the buffer.
When pasting a model you cut, the same name is retained.
When pasting a model you copied, an underscore (_) character is appended to the name, as well as a numerical indicator, which is automatically incremented.

Delete (No Undo)

Deletes the selected model from the work area.

Tools Menu

System Variables

Displays the System Variables window for setting values for global (system) variables. You can create user-defined variables by typing names and values here. You can also type names and values of supplied variables here, or click the **System Variables** button and use the dialog box to select variables, view descriptions, and set values. Note that the names of supplied variables are reserved and cannot be used for variables you create.

The Print button enables you to save the currently displayed variables table to file. You are prompted for a filename. The file is saved to the current work directory with a default .asc extension.

File

- Open—Enables you to open a previously saved Variables Table file (.vat). Change directories as needed, select the desired file, and click **OK**.
- Save As—Enables you to save a variables table to file. Change directories as needed, type the desired name in the Selection field, and click **OK**.
- Close—Closes the variables window.

Edit

- Undo—Undoes the previous copy or paste
- Copy—Copies the currently defined global variables enabling you to paste them in the Model window at the Model, DUT, or Setup level.
- Paste—Enables you to paste variables copied at the Model, DUT, or Setup level to the global level.

System GUI Items

Displays the System GUI Items window. For details, see Chapter 12, “Creating Graphic User Interfaces.”

Simulation Debugger

Displays the Simulation Debugger Window.

Stop Simulator

Stops any currently running piped simulator. The command is most useful when using the ADS simulator with the Root model to ensure correct simulations can be made after changing directories to test other models, and to allow model regeneration.

License Status

Displays a window with dynamically updated license information:

- The codewords currently in use
- The codewords currently available

To release a specific license, select it and click **Release**.

Hardware Setup

Displays the Hardware Setup Window.

Statistics

Launches the Statistics program, if licensed.

Select Simulator

Enables you to change the default simulator after startup. Note: this selection is overriden by a SIMULATOR variable. For detailis, refer to Selecting a Simulator.

Functions

Displays the Function Browser dialog box for reviewing available functions.

Plot Options

Displays the Plot Options dialog box, which enables you to define trace options, plot options, and text annotation.

Options

File Debug

Toggles the debugging facility on and off. When on, messages are recorded in the file .icdebug.

Screen Debug

Toggles the debugging facility on and off. When on, messages are displayed in the IC-CAP Status window.

View GUI Pages

If toggled on, the GUI Items page will be shown in Model windows when the window is first opened.

Status Window to Top

If toggled on, the Status window pops to the front of the screen anytime new messages are displayed in it.

Diagnostics

Executes internal diagnostics.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Hardware Setup Window

The Instrument Setup window enables you to define interface filenames, active instruments, instrument addresses, and unit names to be used in Setups.This window can also be used to control a particular instrument manually.

File Menu

The File menu allows you to perform basic file management commands as well as exit the Hardware Setup window.

Open	Enables you to load an instrument configuration from a previously saved file.
Save	Enables you to save the current instrument configuration to a file of your choosing (the default filename is .icconfig).
Close Window	Closes the Hardware Setup window.

Tools Menu

This set of commands provides basic GPIB capabilities to communicate with the instruments on the bus. These may be useful for debugging an instrument driver or manually setting an instrument to a certain state that is not supported by IC-CAP. The Status panel displays continuously updated information about GPIB activity.

The Tools menu offers the following choices:

- Interface
- Address
- Send/Receive
- Settings
- Macros
- Serial Poll

Interface

Status

Provides the current GPIB bus status.

Change

Closes the current interface file and opens a new one.

Lock

Provides exclusive access to the bus.

Unlock

Releases the I-O Lock on the bus.

Reset

Resets all the instruments on the bus using the Interface Clear command.

Address

Set

Sets the address to which certain commands apply. Some of these commands are:

- Send String
- Receive String
- Serial Poll
- Address > Listen
- Address > Talk
- Address > Check

NOTE: Use the command Address > Set first to set a target GPIB address on which to perform various communications such as "Send String" or "Receive String."

Who Are You?

Polls all addresses other than those at which the CPU is configured. Provides a list of all instruments attached to the bus and powered on, along with their addresses.

Check

Polls active address for any response.

Listen

Sets the instrument at the active address to the Listen state.

Talk

Sets the instrument at the active address to the Talker state.

Send / Receive

Send String

Prompts for a string that will be sent to the instrument at the active address. Carriage return and line feed characters can be included using "\r" and "\n", respectively.

Receive String

Sets CPU to Listen status and the active address to talk status, and collects data from the active address until an EOI is received. Usually used with Send String. The result is also placed in the HPIB_READ_STRING system variable if you have defined it.

Receive PEL String

Same as Receive String, but skips the Talk/Listen setup which should be done using the Send Byte command. The result is also placed in the HPIB_READ_STRING system variable if you have defined it.

Display String

Displays data most recently read by the CPU via a Read String operation.

Send/Read/Display

Lets you send a command to the instrument at the active address and receive a response from the instrument. The response is displayed in the Status panel. The result is also placed in the HPIB_READ_STRING system variable if you have defined it.

Send Byte

Sends one command byte specified as a decimal integer. For example, use 63 to send UNLISTEN.

Settings

Timeout

Specifies the timeout value in seconds. Specifying 0 (zero) disables the use of timeout, which is not recommended.

Macros

Specify

Prompts you for the name of an GPIB Analyzer macro file.

Execute

Executes commands contained in the GPIB Analyzer macro file. For the syntax of commands, refer to Chapter 12, “GPIB Analyzer,” in the *Reference* manual.

Serial Poll

Polls the active address and displays the status byte in decimal form.

Instrument Menu

The Instruments menu offers a variety of instrument operations to assist you in making measurements.

Find	Queries the GPIB bus for instruments.
Display	Displays a list of found instruments on the GPIB bus.
Usage	Displays a list of currently active instruments.
Zero Sources	Forces zero to all the found instruments.
Self Test	Causes the execution of self tests on instruments currently connected to the system, powered up, and listed in the Instrument List.

View Menu

The View menu allows you to toggle on and off the screen debugger and the toolbar.

Screen Debug	Enables you to toggle on and off the low-level debugging facility that produces detailed debug messages on each GPIB transaction.
Toolbar	Enables you to toggle the toolbar on and off.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Model Window

File Menu

Open

Enables you to open parts of a model file previously saved with the Save As command in the Model window (see Opening Parts of a Model File).

Save As

Enables you to save parts of a model file (see Opening Parts of a Model File).

Printer Setup

Opens the Print Setup dialog box (see Chapter 10, “Printing and Plotting”).

Import Data

- Active Setup

  Enables you to import an ASCII-based MDM data file, of a pre-determined format, to the selected setup.

- All Setups in Active DUT

  Enables you to import an ASCII-based MDM data file, of a pre-determined format, to all setups in the selected DUT.

- All DUTs in Model

  Enables you to import an ASCII-based MDM data file, of a pre-determined format, to all setups in all DUTs of the current model.

Export Data

- Active Setup

  Enables you to export an ASCII-based MDM (.mdm) or Dataset (.ds) data file, of a pre-determined format representing the selected setup.

- All Setups in Active DUT

  Enables you to export an ASCII-based MDM (.mdm) or Dataset (.ds) data file, of a pre-determined format representing all setups in the active DUT.

- All DUTs in Model

  Enables you to export an ASCII-based MDM (.mdm) or Dataset (.ds) data file, of a pre-determined format representing all DUTs in the model.

- Extracted Deck

  Enables you to export the circuit block in a SPICE deck format.

- Write Model MDIF

  Writes the Parameters table to a file in MDIF format, which can be read by ADS or Series IV.

Close

Closes the Model window, but the model file remains in memory. To re-open a model, double-click the icon in the Main window.

Related Topics:

Overview

Edit Menu

Undo	Undoes the last Cut, Copy, or Paste action.
Text	Applies to text in the model parts DUTs, Circuit, and Macros.
Cut	Cuts selected text.
Copy	Copies selected text.
Paste	Pastes copied or cut text to other textual model parts.
Find	Enables you to find text.
Replace	Enables you to find and replace text.
Goto Line	Enables you to highlight selected line number.
Cut Setup	Enables you to cut a variety of model parts for pasting within other parts of that same model or to another model.
Copy Setup	Enables you to copy a variety of model parts for pasting within other parts of that same model or to another model.
Paste	Enables you to paste a variety of model parts you cut or copied. You can paste within the model that you cut or copied from or to another model.
Delete Setup (No Undo)	Enables you to delete a variety of model parts.

Measure Menu

Active Setup	Performs a measurement for the active setup.
Active DUT	Performs a measurement for all setups in the active DUT

Extract Menu

Active Setup	Performs all extraction transforms for the active setup.
Active DUT	Performs all extraction transforms for all setups in the active DUT.

Simulate Menu

Active Setup Simulates the active setup.

Active DUT Simulates all setups in the active DUT.

Optimize Menu

Active Setup Performs all optimization transforms for the active setup

Active DUT Performs all optimization transforms for all setups in the active DUT

Data Menu

Plots

Display All

In Active Setup Displays all currently defined plots for the active setup.

In Active DUT Displays all currently defined plots for the active DUT.

in Model Displays all currently defined plots for the current model.

Close All

In Active Setup Closes all displayed plots for the active setup.

In Active DUT Closes all displayed plots for the active DUT.

in Model Closes all displayed plots for the current model.

Clear Data

Measured Clears all measured data *In Active Setup*, *In Active DUT*, or *In Model*.

Simulated Clears all simulated data *In Active Setup*, *In Active DUT*, or *In Model*.

Both Clears both measured and simulated data *In Active Setup*, *In Active DUT*, or *In Model*.

Tools Menu

Simulation Debugger	Displays the Simulation Debugger Window.
Stop Simulator	Stops any currently running piped simulator. The command is most useful when using the ADS simulator with the Root model to ensure correct simulations can be made after changing directories to test other models, and to allow model regeneration.
Organize Model	Displays a window that enables you to add, delete, reorder, and rename DUTs, Macros, and Variables in the model. Selecting the Organize DUT button on the window displays a window that enables you to add, delete, reorder, and rename Setups and Variables. Selecting the Organize Setup button on that window displays a window that enables you to add, delete, reorder, and rename Inputs, Outputs, Transforms, Plots, and Variables.
Refresh Last Dataset...	Re-exports the current data to the last exported dataset (.ds).
Plot Optimizer	Displays the Plot Optimizer window.
Plot Options	Displays the Plot Options dialog box, which enables you to define trace options, plot options, and text annotation.
Hardware Setup	Displays the Hardware Setup Window.

Macros Menu

Execute	Enables you to execute any macro available in the model regardless of the visible folder.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Plot Window

File Menu

Print	Opens the Print dialog box which enables you to print to the specified printer, a file, or copy to the Windows clipboard.
Save Image...	Enables you to save the current plot configuration to a file of your choosing.
Printer Setup...	Opens the Print Setup dialog box (see Chapter 10, “Printing and Plotting”).
Close	Closes the Plot window.

Options Menu

Replot	Refreshes the plot.
Update Annotation	Updates the plots annotation.
Edit Definition...	Displays the Plot Editor.
View Data...	Displays the plot’s data file.
Copy to Clipboard	Copies a plot image to the Windows clipboard.
Rescale	Zooms in to the selected area of a plot.
Set scale	Sets the Manual rescale dialog box to the plot’s current scaling values.
Autoscale	Turns the autoscale mode on or off. When turned on, the graph automatically rescales as data changes. The status appears in the graph’s title.
Manual rescale	Displays the Manual rescale dialog box which enables you to fully describe all three axes of XY plots in terms of minimum value, maximum value, number of major divisions, and number of minor divisions.
Draw Diag Line	Draws a diagonal line connecting two clicked points and its slope, with both X and Y axis intercepts. If you do not click two points first, erases the diagonal line.

Copy to Variables
Copies the X and Y values of a rescale rectangle to system variables. Four variables (X_LOW, X_HIGH, Y_LOW, and Y_HIGH) are reserved for this purpose.

Error

Show Relative Error
Toggles the MAX and RMS relative errors in the footer area on or off.

Show Absolute Error
Toggles the MAX and RMS absolute errors in the footer area on or off.

Select Whole Plot
All the points in the measured/simulated datasets will be used to calculate the error.

Select Error Region
Identifies the selected region as the error region. A green box that delimits the error calculation replaces the white box.

Plot Options
Displays the Plot Options dialog box, which enables you to define trace options, plot options, and text annotation.

Session Settings

Area Tools
Toggles the graph’s area tools on or off.

Legend
Toggles the graph’s legend on or off.

Text Annotation
Toggles the graph’s text annotation on or off.

Title
Toggles the graph’s title on or off.

Header
Toggles the header on or off.

Footer
Toggles the footer on or off.

Exchange Black-White
Reverses the black and white settings for the graph’s grid, text, and background.

Color
Toggles color on or off for the traces and markers. When *Color* is off, the traces and markers are the same color as the graph’s grid and text.

Reset to Saved Options
Resets the current session settings back to the saved Plot Options. This menu pick is not available from Scatter, Histogram, or CDF plots that were opened by the Statistic Package.

Save Current Settings
Saves the current session’s settings as the saved Plot Options. This menu pick is not available from Scatter, Histogram, or CDF plots that were opened by the Statistic Package.

Optimizer Menu

Open Optimizer...	Opens the Plot Optimizer window.
Enable / Disable Plot	Enables or disables the Plot window. Enabling the plot window synchronizes it with the Plot Optimizer window.
Global Region	
Reset	Deletes all existing global trace regions on the plot and defines a new global trace region.
Add	Adds a global trace region without deleting existing global trace regions.
Delete All	Deletes all global trace regions.
Autoconfigure and Enable	Automatically enables and configures the inputs in a Plot window.
Disable All Traces	Disables all inputs in a Plot window.
trace	
Set as Both Target and Simulated	Configures the selected trace with its measured data set as Target and its simulated data set as Simulated.
Set as Target vs.	Configures the selected trace as Target.
Set as Simulated vs.	Configures the selected trace as Simulated.
Disable	Disables the selected trace.
Trace Optimizer Region	
Reset	Deletes all existing trace optimizer regions for the selected trace and defines a new trace optimizer region.
Add	Adds a trace optimizer region for the selected trace without deleting existing trace optimizer regions.
Delete All	Deletes all trace optimizer regions for the selected trace.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Multiplot Window

File Menu

Print
Opens the Print dialog box which enables you to print to the specified printer, a file, or copy to the Windows clipboard.

Save Image...
Enables you to save the current plot configuration to a file of your choosing.

Printer Setup...
Opens the Print Setup dialog box (see Chapter 10, “Printing and Plotting”).

Close
Closes the Plot window.

Options Menu

Replot
Refreshes the plot.

Update Annotation
Updates the plots annotation.

Copy to Clipboard
Copies a plot image to the Windows clipboard.

Autoscale
Turns the autoscale mode on or off. When turned on, the graph automatically rescales as data changes. The status appears in the graph’s title.

Error

Show Relative Error
Toggles the MAX and RMS relative errors in the footer area on or off.

Show Absolute Error
Toggles the MAX and RMS absolute errors in the footer area on or off.

Select Whole Plot
All the points in the measured/simulated datasets will be used to calculate the error.

Select Error Region
Identifies the selected region as the error region. A green box that delimits the error calculation replaces the white box.

Plot Options
Displays the Plot Options dialog box, which enables you to define trace options, plot options, and text annotation.

Session Settings

Area Tools	Toggles the graph’s area tools on or off.
Legend	Toggles the graph’s legend on or off.
Text Annotation	Toggles the graph’s text annotation on or off.
Title	Toggles the graph’s title on or off.
Header	Toggles the header on or off.
Footer	Toggles the footer on or off.
Exchange Black-White	Reverses the black and white settings for the graph’s grid, text, and background.
Color	Toggles color on or off for the traces and markers. When *Color* is off, the traces and markers are the same color as the graph’s grid and text.
Reset to Saved Options	Resets the current session settings back to the saved Plot Options.
Save Current Settings	Saves the current session’s settings as the saved Plot Options.

Optimizer Menu

Open Optimizer...	Opens the Plot Optimizer window.
Enable / Disable Plot	Enables or disables the Plot window. Enabling the plot window synchronizes it with the Plot Optimizer window.
Global Region	
Reset	Deletes all existing global trace regions on the plot and defines a new global trace region.
Add	Adds a global trace region without deleting existing global trace regions.
Delete All	Deletes all global trace regions.
Autoconfigure and Enable	Automatically enables and configures the inputs in a Plot window.
Disable All Traces	Disables all inputs in a Plot window.
trace	
Set as Both Target and Simulated	Configures the selected trace with its measured data set as Target and its simulated data set as Simulated.
Set as Target vs.	Configures the selected trace as Target.

Set as Simulated vs. Configures the selected trace as Simulated.

Disable Disables the selected trace.

Trace Optimizer Region

Reset Deletes all existing trace optimizer regions for the selected trace and defines a new trace optimizer region.

Add Adds a trace optimizer region for the selected trace without deleting existing trace optimizer regions.

Delete All Deletes all trace optimizer regions for the selected trace.

Plots Menu

Select Plot Displays a list of available plots. The plot you choose becomes the selected plot.

Unselect All Unselects the currently selected plot.

Zoom Plot Displays a list of available plots. The plot you choose is displayed in zoom format.

Full Page Plot Displays a list of available plots. The plot you choose is displayed in full page format.

Undo Zoom Resets the Multiplot window to the default format.

Selected Plot Menu

Reset to Saved Options Resets the Plot Options to the last saved options.

Plot Options Displays the Plot Options dialog box, which enables you to define trace options, plot options, and text annotation.

Optimizer

Open Optimizer Opens the Plot Optimizer window.

Enable / Disable Plot Enables or disables the Plot window. Enabling the plot window synchronizes it with the Plot Optimizer window.

Global Region

Reset Deletes all existing global trace regions on the plot and defines a new global trace region.

Add	Adds a global trace region without deleting existing global trace regions.
Delete All	Deletes all global trace regions.
Autoconfigure and Enable	Automatically enables and configures the inputs in a Plot window.
Disable All Traces	Disables all inputs in a Plot window.
trace	
Set as Both Target and Simulated	Configures the selected trace with its measured data set as Target and its simulated data set as Simulated.
Set as Target vs.	Configures the selected trace as Target.
Set as Simulated vs.	Configures the selected trace as Simulated.
Disable	Disables the selected trace.
Trace Optimizer Region	
Reset	Deletes all existing trace optimizer regions for the selected trace and defines a new trace optimizer region.
Add	Adds a trace optimizer region for the selected trace without deleting existing trace optimizer regions.
Delete All	Deletes all trace optimizer regions for the selected trace.
Scaling	
Replot	Refreshes the plot.
Rescale	Zooms in to the selected area of a plot.
Set scale	Sets the Manual rescale dialog box to the plot's current scaling values.
Autoscale	Turns the autoscale mode on or off. When turned on, the graph automatically rescales as data changes. The status appears in the graph's title.
Manual rescale	Displays the Manual rescale dialog box which enables you to fully describe all three axes of XY plots in terms of minimum value, maximum value, number of major divisions, and number of minor divisions.
Graphic	Updates the plots annotation.

Draw Diag Line	Draws a diagonal line connecting two clicked points and its slope, with both X and Y axis intercepts. If you do not click two points first, erases the diagonal line.
Copy to Variables	Copies the X and Y values of a rescale rectangle to system variables. Four variables (X_LOW, X_HIGH, Y_LOW, and Y_HIGH) are reserved for this purpose.
Area Tools	Toggles the graph's area tools on or off.
Legend	Toggles the graph's legend on or off.
Text Annotation	Toggles the graph's text annotation on or off.
Title	Toggles the graph's title on or off.
Header	Toggles the header on or off.
Footer	Toggles the footer on or off.
Error	
Show Relative Error	Toggles the MAX and RMS relative errors in the footer area on or off.
Show Absolute Error	Toggles the MAX and RMS absolute errors in the footer area on or off.
Select Whole Plot	All the points in the measured/simulated datasets will be used to calculate the error.
Select Error Region	Identifies the selected region as the error region. A green box that delimits the error calculation replaces the white box.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Plot Optimizer Window

File Menu

Open...	Enables you to open an existing plot optimizer file.
Save As...	Enables you to save the current plot optimizer configuration to a file of your choosing.
Clear Plot Optimizer	Disables all traces and regions in the plots, disables all open plots, and clears the Parameters table.
Reset Option Table	Resets all options in the Options table to the default values.
Close	Closes the Plot Optimizer window.

Plots Menu

Enable All	Enables all open Plot windows in a model file.
Disable All	Disables all open Plot windows in a model file.

Simulate

Simulate All	Simulates all enabled traces.

Optimize

Run Optimization	Performs an optimization of the enabled traces.
Tune Fast...	Opens a tuner window which enables you to adjust the parameter values. You can see the effects in a plot as you vary parameter values using the tuner. Recalculates constantly as a slider is moved. While the tuner window is open, you can change the *Min* and *Max* values.

Tune Slow...	Opens a tuner window which enables you to adjust the parameter values. Recalculates only when a slider is released. You can see the effects in a plot as you vary parameter values using the tuner. While the tuner window is open, you can change the *Min* and *Max* values. It's best to use Tune Slow when the time per calculation is slow (about one second or more).

Tools

Store Parameters	Stores values in the Parameters table for later recall.
Recall Parameters	Recalls the values stored in the Parameters table.
Undo Optim	After running an optimization, restores the parameters to their previous state.
AutoSet Min and Max	Automatically sets the parameter values.
Reset Min and Max	Restores the parameter values to their default values.

Windows

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

Simulation Debugger Window

The Simulation Debugger window displays the circuit input deck and the results of the simulation in text form. To display data, open the Simulation Debugger window and then run a simulation (either from the DUT or Setup level). The data is displayed in the Input, Output, and Command File regions, however, the Command File region is used only with the Saber simulator and displays the Saber commands generated from the IC-CAP Setup.

For debugging, edit the data in the Input editor, then run a simulation from the debugger to test the changes. To run a simulation from the debugger, use the Manual Simulation command on the File Menu. If the changes are successful, change the circuit definition to reflect your changes.

Note: After running a simulation using the SPICE3 simulator, the Output region of the debugger displays the message ".print card ignored since rawfile was produced." To display the output of the simulation in text form, issue the Manual Simulation command.

File Menu

Manual Simulation Executes a simulation using the circuit deck in the Input field of the debugger. The Debugger Input field is not linked to the Model Circuit Definition editor or Input and Output setups, so edits made in the field have no effect on IC-CAP and are irrelevant to a simulation run from a DUT or Setup. To make changes permanent, make them in the Model Circuit Definition editor.

Save Input File Provides a dialog box for specifying a filename for saving the contents of the Input editor to a file.

Save Command File Provides a dialog box for specifying a filename for saving the contents of the Command editor to a file.

Save Output File	Provides a dialog box for specifying a filename for saving the contents of the Output editor to a file.
Close	Closes the Simulation Debugger window.

Windows Menu

The Windows menu provides a quick method of bringing a different window to the foreground. All currently open windows, including those that are minimized, are listed on the menu. Individual models are listed on the Model window submenu. Choose the desired window and it is displayed in front of all other IC-CAP windows.

B
File and Data Management

The directory containing the example model files is protected to preserve an original copy of each model file. The typical method of modifying a model file is to use the Save As command and save a copy of it to another directory. You will probably want to create one or more directories, in any terminal window, for storing your model files and data.

Quick access to the example model files is provided with the File > Examples command in the Main window. When you want to open a model file other than one from the examples directory, by default, the path in the dialog box is set to the directory from which you started the program. Double-click to get to the desired directory.

NOTE

A symbol displayed in the Main window represents a model file currently in memory. If you close a Model window using File > Close, the file remains in memory and can be reopened by double-clicking the symbol.

Opening Files

To open a file:

1 Choose **File > Open** and a dialog box appears. By default, the path in the Filter field is set to the directory from which you started the program and the filter is set to *.ext where ext is an extension specific to the task you are performing. (For example, *.mdl for model files, *.hdw for hardware configuration files, etc.). All files in the current directory with that extension are displayed.

2 Adjust the path as needed. You can type in the Filter field and then click the **Filter** button or use the mouse to navigate the list of Directories.

NOTE

The path for IC-CAP model files cannot contain any folder names that use a space. For example, *C:\Model Files\IC-CAP 2004*. If a model file is saved in a folder name with spaces, you will not be able to open the model file. You will have to move the model file to a folder name that does not use a space.

3 To choose a file from the Files list box, double-click it or click once and choose **OK**. (Tip: If you click once, you will see the selected filename reflected in the Selection field.)

NOTE

If loading a model file, once it is loaded, a symbol with the model name is displayed in the work area. Double-click the symbol to open the Model window.

Opening Parts of a Model File

The File > Open command in the Model window enables you to open a file (previously saved using File > Save As in the Model window) that contains some portion of a model.

To open a file containing a portion of a model:

1. If you want to replace some portion of the current model file with a portion of another file, select the item you want to replace.
2. Choose **File > Open** in the Model window.
3. Select the appropriate file type and click **Browse** to view the files of that type in the directory you specify.

NOTE

The path for IC-CAP model files cannot contain any folder names that use a space. For example, *C:\Model Files\IC-CAP 2004*. If a model file is saved in a folder name with spaces, you will not be able to load the model file. You will have to move the model file to a folder name that does not use a space.

4. Select a file and click **OK**.
5. For Model Parameters and DUT Parameters, select **Replace Parameter Set** or **Read Values Only** and click **OK**.
   - Replace Parameter Set replaces all Value, Min, and Max values with the ones in the selected file. If the selected file does not include Min and Max ranges (e.g., when reading pre-IC-CAP 2004 parameter set files), the existing Min and Max ranges are deleted.
   - Read Values Only replaces parameter Values while maintaining existing Min and Max ranges if possible. If the new Value is outside the existing range, Min or Max is extended to include the new Value and a warning is displayed in the Status window.
6. For all other Model folders, select the **Replace** option to use the specified file as a replacement and click **OK**. To add the specified file, make sure the option is turned off and click **OK**.

Saving Files

To specify a directory and a filename:

1 Choose **File > Save As** and a dialog box appears.

2 Double-click in the Directories list box to locate the desired directory.

3 To save to an existing file, select that filename from the Files list box and click **OK**. To save to another name, type that name in the Selection field and click **OK**.

4 Where applicable, click **OK** to dismiss the previous dialog box.

Saving a Model File

From the Save As dialog box in the Main window:

1. Select all models you want to save to one model file.
2. Accept the default status of saving the file with the measured and simulated data, or enable the option to save without it.
3. Adjust the path and filename as needed, and click **Apply**.

   Hint: To avoid typing a lengthy path change, use the Browser.

NOTE

The path for IC-CAP model files cannot contain any folder names that use a space. For example, do not save to *C:\Model Files\IC-CAP 2004*. If you include a space in a folder name, you will not be able to load the model file. You will have to move the model file to a folder name that does not use a space.

4. Continue saving as needed, clicking **Apply** to effect each change.
5. Click **OK** to dismiss the dialog box.

NOTE

You can also save a model file from the Model Window if you want to save all components to one file.

Saving Parts of a Model File

When you choose File > Save As in a Model window, you can choose which part of the model to save to file: DUT, setup, input, output, transform, plot, macro, variable table, parameter set, circuit description, or instrument options.

1. Select the option representing the portion of the file you want to save. (Where applicable, select the option from the drop-down list.)
2. Accept the default status of saving the file with the measured and simulated data, or enable the option to save without it.

3 Adjust the path and filename as needed.

Hint: To avoid typing a lengthy path change, use the Browser.

NOTE

The path for IC-CAP model files cannot contain any folder names that use a space. For example, do not save to *C:\Model Files\IC-CAP 2004*. If you include a space in a folder name, you will not be able to load the model file. You will have to move the model file to a folder name that does not use a space.

4 Click **OK** to effect the save and dismiss the dialog box.

Changing Default Directory

By default, model files and data will be saved to the directory from which you started the program.

To change the directory:

1 Choose **File > Change Directory**. By default, the path in the Filter field is set to the directory from which you started the program.

2 Adjust the path as needed. You can type in the Filter field and then click the **Filter** button or use the mouse to navigate the list of Directories.

3 When the Selection field reflects the desired directory, click **OK**.

Importing Data into a Single Setup

1 Select the DUT and setup.

2 Select **File > Import Data** and choose **Active Setup**.

Alternatively, you can select **Measure / Simulate** and click **Import Data**.

3 Enter the name of the file directly into the field corresponding to the setup, by typing in or by using the File Browser.

4 Select a fill data type.

- **Measured if available, otherwise Simulated:** Outputs of type M or B receive mdm data in their measured array. Outputs of type S receive mdm data in their simulated array.
- **Measured only:** Outputs of type M or B receive data in their measured array. Outputs of type S do not receive any data.
- **Simulated if available, otherwise Measured:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M receive mdm data in their measured array.
- **Simulated only:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M do not receive any data.

5 Click **OK**. The data is now imported into the outputs of the setup.

See Also

- Importing Data into All Setups in an Active DUT or All DUTs in the Model
- Chapter 13, “Managing Data”

Importing Data into All Setups in an Active DUT or All DUTs in the Model

1 Select the DUT.

2 Select **File > Import Data** and choose one of the following:

All Setups in Active DUT

All DUTs in Model

The dialog box that appears has three columns (DUT, Setup, Data Filename) and a Fill Data Type selection area.

3 Specify a filename for each setup. You can type the path and filename directly in the Filename field, or use the Browser to select it.

4 Click **Apply** each time you specify a filename for a setup.

5 Select a fill data type.

- **Measured if available, otherwise Simulated:** Outputs of type M or B receive mdm data in their measured array. Outputs of type S receive mdm data in their simulated array.
- **Measured only:** Outputs of type M or B receive data in their measured array. Outputs of type S do not receive any data.
- **Simulated if available, otherwise Measured:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M receive mdm data in their measured array.
- **Simulated only:** Outputs of type S or B receive mdm data in their simulated array. Outputs of type M do not receive any data.

6 Click **OK**.

Tips

- To view the header of an MDM data file, select the Setup cell in the dialog box, enter the corresponding MDM filename, then click **View**.
- To create the inputs and outputs for a new setup from the header of an MDM file, first create a new setup, select the setup name in the dialog box, enter the corresponding MDM filename, and click **Create**.

See Also

- Importing Data into a Single Setup
- Chapter 13, “Managing Data”

C
DUT / Setup Editing

Models usually contain several DUTs. DUTs contain groups of Setups that have a similar physical connection to the device. If two Setups require different parameters or different test circuits, they must belong to different DUTs.

See also: "Defining DUT Variables" on page 142.

Adding a DUT or a Setup

To add a DUT to a model:

1 With the model file open, click the **Add** button at the bottom of the DUT/Setup panel.

2 In the dialog box that appears, select **Add New DUT**.

3 Enter a name for the new DUT and click **OK**. The new DUT appears at the bottom of the DUT/Setup panel.

To add a Setup to a DUT:

1 With the model file open, and the appropriate DUT selected, click the **Add** button at the bottom of the DUT/Setup panel.

2 In the dialog box that appears, select **Add New Setup**.

3 Enter a name for the new Setup and click **OK**. The new Setup appears for that DUT. The new Setup appears below any existing setup(s).

See "Organizing DUTs and Setups" on page 636 for an alternate way to add DUTs and Setups to a model.

Deleting a DUT or a Setup

To delete a DUT:

1 Select the DUT you want to delete.

2 Choose **Edit > Cut** and a dialog box appears.

3 Select the DUT option and click **OK**.

To delete a Setup:

1 Select the setup you want to delete.

2 Choose **Edit > Cut** and a dialog box appears.

3 Select the Setup option and click **OK**.

See “Organizing DUTs and Setups” on page 636 for an alternate way to delete DUTs or Setups from a model.

Copying / Pasting a DUT or Setup

To copy and paste a DUT:

1 Select the DUT you want to copy.

2 Choose **Edit > Copy** and a dialog box appears.

3 Select the DUT option and click **OK**.

4 Choose **Edit > Paste** and the copied DUT is added after any existing DUTs.

5 Rename the DUT/Setups as needed (see Renaming a DUT or Setup).

To copy and paste a setup:

1 Select the setup you want to copy.

2 Choose **Edit > Copy** and a dialog box appears.

3 Select the Setup option and click **OK**.

4 Select the DUT you want to copy the setup to and choose **Edit > Paste** Setup. The copied setup is added below any existing setups of that DUT.

5 Rename the setup as needed (see Renaming a DUT or Setup).

Renaming a DUT or Setup

When you paste a DUT or Setup, the copy retains the name of the original, plus an underscore and a number (automatically incremented).

To rename a DUT or Setup:

1 Select the DUT or setup you want to rename.

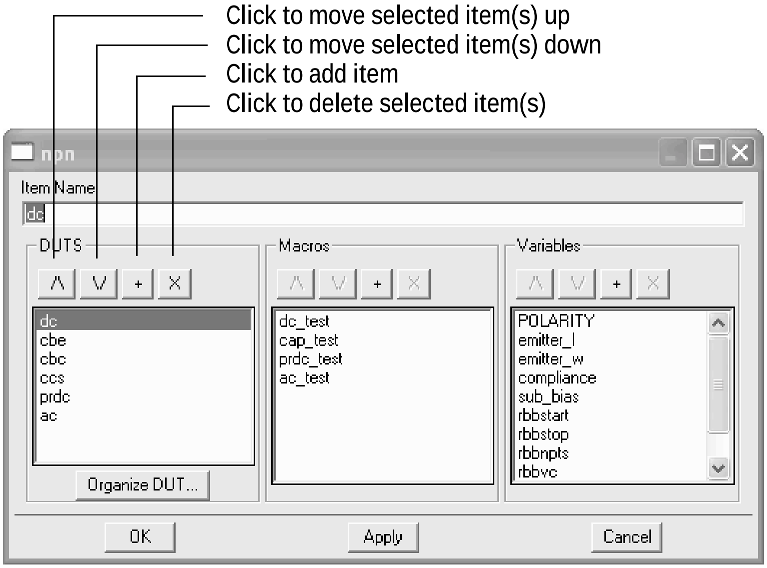
2 Click the **Rename** button and a dialog box appears.

3 Type a new name for the DUT or setup and click **OK**. The name is updated immediately.

See also: Organizing DUTs and Setups

Organizing DUTs and Setups

Selecting *Tools* > *Organize Model* from a Model window displays the following dialog box:



Click to move selected item(s) up
Click to move selected item(s) down
Click to add item
Click to delete selected item(s)

Item Name: dc

This dialog box enables you to move, add, delete, or rename DUTs, macros, and variables.

To move an item:

1 Select one or more items:

2 Above the selected items, click the move up ∧ icon to move the items up in the list, or click the move down ∨ icon to move the items down in the list. Each click moves the selected items one position.

To add an item:

1 Choose where you want to add the item in the list:

- Select an item if you want to add an item after that position.
- Do not select an item if you want to add an item to the first position.

2 Click the plus + icon.

3 A new item named Untitled is added. Type the desired name in the Item Name field then select the Enter key.

To delete an item:

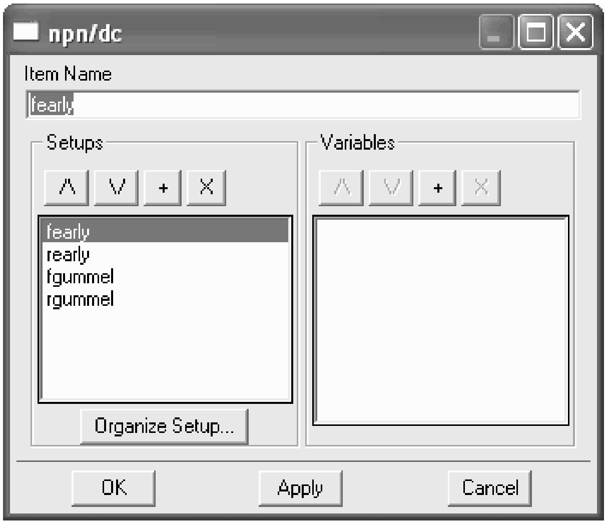
1 Select one or more items:

2 Click the delete ✕ icon.

To display a dialog box that enables you to move, add, delete, or rename a DUT's setups and variables:

1 Select a DUT.

2 Click the **Organize DUT** button. This displays the following dialog box, which functions in the same manner as the previous dialog box.



npn/dc

Item Name: fearly

Setups

fearly
rearly
fgummel
rgummel

Variables

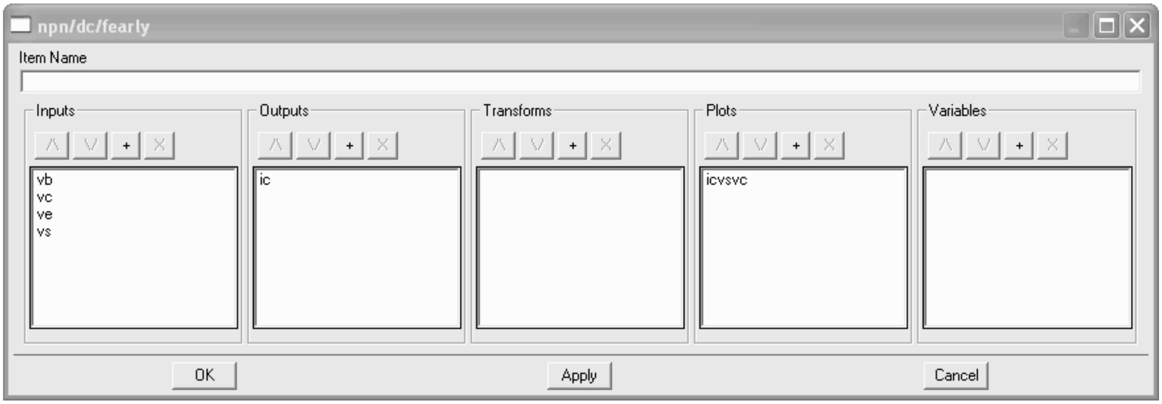
NOTE

You can also display this dialog box by selecting a DUT in the model file, then clicking the **Organize** button at the bottom of the DUT / Setup panel.

To display a dialog box that enables you to move, add, delete, or rename a Setup's inputs, outputs, transforms, plots, and variables:

1 Select a Setup.

2 Click the **Organize Setup** button. This displays the following dialog box, which functions the same as the previous dialog box.



NOTE

You can also display this dialog box by selecting a Setup in the model file, then clicking the **Organize** button at the bottom of the DUT / Setup panel.

To apply you changes and leave the dialog box open, select the **Apply** button.

To apply your changes and close the dialog box, select the **OK** button.

To close the dialog box without making changes, click the **Cancel** button.

Defining a Test Circuit for a DUT

To define a test circuit:

1. Select the DUT for which you want to define a test circuit.
2. Create the definition by typing in the text area or by importing an existing text file.
3. When the file is complete, choose **Parse**.

NOTE

When you import text, the file is automatically parsed.

Defining DUT Parameters

To define parameters for a specific DUT:

1. Select the DUT and click to open the DUT Parameters folder.
2. Click once to activate the text field of a parameter you want to change.
3. Position the cursor as necessary, click again and use the backspace key to erase existing values and retype, or double click to highlight and retype the entire entry.

Tips:

- Use Detach to keep the parameters visible and editable, while viewing other parts of the model. When you are done, close the detached window and any changes made there are reflected in the Model window.
- Use Memory Store All to temporarily store the parameter set prior to extraction; use Memory Recall All to retrieve them if the extracted values are unacceptable.
- Use Reset All to overwrite DUT parameter values with circuit parameter values.
- Use Update Circuit to overwrite circuit parameter values with DUT values.

Setup Editing

Setups contain information for performing specific measurements and simulations on a DUT. The individual components of a Setup are:

- Inputs
- Outputs
- Extraction/Optimization Specifications (Transforms)
- Plots
- Variables
- Instrument Options

See also: Assigning Values to Setup Variables and Organizing DUTs and Setups

Input / Output Editing

The inputs and outputs of a Setup define the various currents, voltages, etc. you want to monitor with respect to the associated DUT.

To add a new input or output:

1. Click **New Input** or **New Output**.
2. Fill in all fields as necessary and click **OK**.

To delete an input or output:

1. Select the input or output.
2. Choose **Edit > Cut**, select the input/output option in the dialog box, and click **OK**.

To edit an input or output:

1. Double-click the input or output or select it and click the **Edit** button in the Measure/Simulate folder or use the on-screen editor.
2. Make all the necessary changes in the dialog box and click **OK**.

   Alternatively, you can edit an input or output directly using the on-screen editor.

See also: Organizing DUTs and Setups

Variables

Several variable names are reserved by IC-CAP and cannot be assigned as user-defined variables. You can assign values to reserved variables and define your own variables at several different levels (variables at lower levels inherit their values from variables above them):

- Global * DUT level
- Model level * Setup level

You define global variables through the IC-CAP Main window. These variables apply to all Models, DUTs, and Setups unless you explicitly set the variables differently at the Model, DUT, or Setup level.

You define Model, DUT, and Setup variables through the Model window:

- Model variables apply to all DUTs and Setups of that Model unless you explicitly set the variables differently for individual DUTs and/or Setups.
- DUT variables apply to all Setups of that DUT unless you explicitly set the variables differently for individual Setups.
- Setup variables apply only to that Setup.

Assigning Values

Assigning Values to Global Variables

To assign values to global variables:

In the Main window, choose **Tools > System Variables** and the Variables window appears.

- To create user-defined variables:

Type the names and values in the fields provided. Additional fields appear as you specify values.

- To assign values to supplied variables:

1 Type their names and values in the System Variables window or click **System Variables** and a dialog box appears.

2 Select the appropriate category and the variables in that category are displayed.

3 Click to select a variable and a description of that variable is displayed.

4 Type the appropriate value in the Value field and click **Apply**. The variable name and its value appear in the Variables window.

5 Continue assigning variable values as needed. When you are finished, click **OK**.

Assigning Values to Model Variables

- To define variables at the model level:

  Click to open the Model Variables folder.

- To create user-defined variables:

  Type the names and values in the fields provided. Additional fields appear as you specify values.

- To assign values to supplied variables:

1 Type their names and values in the System Variables window or click **System Variables** and a dialog box appears.

2 Select the desired Variable Type and a list of variables of that type appears.

3 Select the variable. A description is displayed.

4 Enter the desired value for that variable and click **Apply**.

5 Continue defining variables in this manner, clicking **Apply** to effect each change. When you are through defining variables, click **OK**.

Assigning Values to DUT Variables

To define variables for a DUT:

  Click to open the DUT Variables folder.

- To create user-defined variables:

  Type the names and values in the fields provided. Additional fields appear as you specify values.

- To assign values to supplied variables:

1. Type their names and values in the System Variables window or click **System Variables** and a dialog box appears.
2. Select the desired Variable Type and a list of variables of that type appears.
3. Select the variable. A description is displayed.
4. Enter the desired value for that variable and click **Apply**.
5. Continue defining variables in this manner, clicking **Apply** to effect each change. When you are through defining variables, click **OK**.

Assigning Values to Setup Variables

To define Setup variables

Click to open the Setup Variables folder.

- To create user-defined variables:

  Type the names and values in the fields provided. Additional fields appear as you specify values.

- To assign values to supplied variables:

1. Type their names and values in the System Variables window or click **System Variables** and a dialog box appears.
2. Select the desired Variable Type and a list of variables of that type appears.
3. Select the variable. A description is displayed.
4. Enter the desired value for that variable and click **Apply**.
5. Continue defining variables in this manner, clicking **Apply** to effect each change. When you are through defining variables, click **OK**.

See also: Organizing DUTs and Setups

Creating User-defined Variables

To create user-defined variables:

1. Select **Tools > System Variables** in the Main window if you are creating global variables, otherwise click to open the Model, DUT, or Setup Variables folder, as desired.
2. In the System Variables window that appears, type the names and values in the fields provided. Additional fields appear as you specify values.

Defining Instrument Options

The Instrument Options table is available once you specify the unit names (from the Hardware Setup configuration dialog) in the Inputs and Outputs, as needed. Once you define your instrument options, you can save them to file and read in this file anytime as needed.

To save instrument options:

1. With the Instrument Options folder active, choose **File > Save As**.
2. In the dialog box that appears, select **Instrument Options** (.iot).
3. If desired, choose another directory by typing or using the Browser.
4. Provide a filename (the extension is appended automatically) and click **OK**.

To read a file containing previously saved instrument options:

1. With the Instrument Options folder active, choose **File > Open**.
2. In the dialog box that appears, select **Instrument Options (.iot)**.
3. If desired, choose another directory by typing or using the Browser.
4. Provide a filename (the extension is appended automatically) and click **OK**.

D
Macros

Creating and Running Macros

1 With any of the macros in the Macros folder selected, choose **File > Save As**.

2 Select the **Macro** option.

3 Select the macro name from the drop-down list.

4 If desired, change the path using the Browser.

5 If desired, supply a different filename in the File Name field.

6 Click **OK**.

Saving Macros

Macros are saved when you save the model file. They are saved to the current work directory. You can explicitly save a macro at any time and specify another directory.

To save macros:

1. With any of the macros in the Macros folder selected, choose **File > Save As**.
2. Select the **Macro** option.
3. Select the macro name from the drop-down list.
4. If desired, change the path using the Browser.
5. If desired, supply a different filename in the File Name field.
6. Click **OK**.

E
Transforms

Creating and Running a Transform

A transform is a framework for mathematical or logical functions that operate on data. Extraction and optimization are accomplished through the use of transforms. A transform can operate on any combination of data sets, parameters, and variables. These inputs are used to calculate either a new data set or new values for parameters and variables. Transforms are created for Setups, since they typically operate on data from a particular Setup.

To create and run a transform:

1 Open the Model window and select the desired Setup.

2 Click the **Extract/Optimize** tab and note the list of existing transforms.

3 Click **New** and provide a name for the new transform in the dialog box that appears.

4 Click **OK** and the new transform name is added to the list.

5 Select an existing function from the Browser, or type `Program2` or `Program` in the Function field, and press Enter to create your own.

6 Fill in the argument fields displayed for a selected function, or type the desired text if creating your own.

7 To run the transform, click **Execute**.

Related Topics:

Chapter 9, “Using Transforms and Functions”

Chapter 11, “Creating and Running Macros”

Chapter 9, “Parameter Extraction Language,” in the *Reference* manual

Creating a Function

To create a new function:

1. Click the **Extract / Optimize** tab.
2. Select the transform for which you want to create a function.
3. Type `Program2` or `Program` in the Function field and press Enter. The window changes to display a scrollable text field.
4. Type the desired text.

NOTE

When you create a function within the user interface, you use a BASIC-like language referred to as Parameter Extraction Language (PEL). You can also create functions outside IC-CAP using C language.

Selecting an Existing Function

To choose a function from the Browser:

1. Select the transform for which you want to assign a function and click the **Browse** button.
2. In the dialog box that appears, select the appropriate Function Group to display a list of available functions in that group.
3. Select a Function and a brief description is displayed.
4. Click **Select** and the dialog box disappears and the function name appears in the Function field.
5. Where applicable, assign appropriate values or edit as needed.

Related Topics:

Chapter 9, “Using Transforms and Functions”

Chapter 11, “Creating and Running Macros”

Chapter 9, “Parameter Extraction Language,” in the *Reference* manual

Saving Transforms

By default, transforms are saved when you save the model file. But you can explicitly save a transform at any time and specify another directory, if desired.

To save transforms:

1. With any of the transforms in the Extract/Optimize folder selected, choose **File > Save As**.
2. Select the **Transform** option.
3. Select the transform name from the drop-down list.
4. If desired, change the path using the Browser.
5. If desired, supply a different filename in the File Name field.
6. Click **OK**.

Related Topics:

Chapter 9, “Using Transforms and Functions”

Chapter 11, “Creating and Running Macros”

Chapter 9, “Parameter Extraction Language,” in the *Reference* manual

F
Miscellaneous

Clearing Data from Memory

The Clear command allows you to clear from memory the data for the current setup. You can clear measured data, simulated data, or both.

This command is useful when you have already measured all setups of a DUT, or all DUTs that make up the model, and need to make a change to one setup. You can clear the measured data for that setup and re-measure that setup.

Tuning Parameters

By including the TUNER statement in a PEL macro, you can tune parameters during macro execution. To tune parameters once the dialog box appears:

a Position the pointer over the slider associated with the parameter you want to tune.

b Press the left mouse button and drag the slider in the desired direction. Notice the plot is dynamically updated as you change values.

c To close the dialog box and return control to the macro, click **OK**; to abort the macro, click **Cancel**.

Saving to File

Several parts of the program enable you to save a variety of things to file. For example, you can click the **Print** button in the Model Variables folder, enter a filename and click **OK**, and an ASCII file (.asc) containing parameter names and values is written to your startup directory.

Another example is plotting to file. In this case, the contents of the file vary depending on the source, which may be graphics or text.

Related Topic:

- Chapter 10, “Printing and Plotting”

Cut and Copy Selections

When you cut and copy model parts at the DUT and Setup levels, you are prompted to choose between the DUT or Setup (depending on which is active) and a model part available in the active folder. For example, in the Measure/Simulate folder, you can choose to cut or copy the Setup itself or selected Inputs and/or Outputs. Likewise, in the Test Circuit folder, you can choose to cut or copy the DUT itself or the Test Circuit.

- Select the appropriate option and click **OK**.

Adding an Interface Name

To add an interface name to the list, click **Add Interface**. In the dialog box that appears, supply the interface filename. An interface card by that name should have been previously installed. When you choose **OK**, the existence of a card by that name is verified.

For information about installing and configuring an interface card, see “Check the Supported Instrument Interfaces” in chapter 1, “Installing IC-CAP on PC Systems” or chapter 4, “Installing IC-CAP on UNIX Systems” of the *Installation and Customization Guide*.

Deleting an Interface Name

To delete an interface filename from the list, select the interface name and click **Delete Interface**.

Creating the List of Active Instruments

The Instrument List potentially displays all instruments attached to the GPIB bus. Once on the list, you can configure each instrument to set the address and assign unit names. You can create this list automatically or manually.

- To create the Instrument List automatically:

  Click **Rebuild**. This clears the current Instrument List, polls all available GPIB addresses. and creates a complete list of instruments connected to the GPIB bus, providing they are powered up.

- To create the Instrument List manually

  Select the instrument name and click **Add to List**. This enables you to create a list of instruments that are not necessarily currently powered up or connected to the bus.

Deleting an Instrument from the Instrument List

To delete an individual instrument from the list, select it and click **Delete**. You can delete an instrument any time, regardless of its power status.

NOTE

To save instrument options before deleting instruments from the list, with the Instrument Options folder active, choose **File > Save As**. Select **Instrument Options (.iot)**, set the path and filename as desired, and click **OK**.

Deleting All Instruments from the List

To delete all instruments from the Instrument List, click **Delete All**. A confirmation dialog box appears to confirm this operation because all instrument options will also be removed.

NOTE

To save instrument options before deleting instruments from the list, with the Instrument Options folder active, choose **File > Save As**. Select **Instrument Options (.iot)**, set the path and filename as desired, and click **OK**.

Configuring an Instrument

To define the GPIB bus address of an instrument in the Instrument List, as well as assign unit names, select that instrument and click **Configure**.

Interface

Reflects the names of any interface files added in the Hardware Setup window. Select one to change its address and assign units.

Instrument Address

Enables you to set the address of an instrument on the GPIB.

Unit Table

Contains default unit names for an instrument. Edit these names as needed.

NOTE

After assigning unit names here, specify those same names for the inputs and outputs (in the Measure/Simulate folder) as needed.

Index