Micro-Cap 7.0

Electronic Circuit Analysis Program

Reference Manual

Copyright 1982-2001 by Spectrum Software
1021 South Wolfe Road
Sunnyvale, CA 94086
Phone: (408) 738-4387
FAX: (408) 738-4702
Internet:www.spectrum-soft.com
Support:support@spectrum-soft.com
Sales:sales@spectrum-soft.com

Seventh Edition
First Printing
September 2001

Copyright
© Spectrum Software. 1982-2001. This manual and the software described in it are copyrighted, with all rights reserved. No part of this publication, or the software, may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form without the written permission of Spectrum Software.

Software license notice
Your license agreement with Spectrum Software, which is included with the product, describes the uses of the software which are permitted and the uses which are prohibited. Any unauthorized duplication of Micro-Cap 7, in whole or in part, in print, or in any other retrieval or storage system is expressly forbidden.

Table of Contents

Chapter 13 - Scope

Chapter 14 - Probe

Chapter 15 - Stepping

Chapter 16 - Monte Carlo Analysis

Chapter 17 - The MODEL Program

Chapter 18 - Convergence

Chapter 19 - Expressions

Chapter 20 - The Command Statements

Chapter 22 - Analog Devices

Chapter 23 - Digital Devices

Chapter 28 - Optimizer

Appendices

Typographic conventions

Certain typographic conventions are employed to simplify reading and using the manuals. Here are the guidelines:

1. Named keys are denoted by the key name alone. For example:

   Press HOME, then press ENTER.

2. Text that is to be typed by the user is denoted by the text enclosed in double quotes. For example:

   Type in the name "TTLINV".

3. Combinations of two keys are shown with the key symbols separated by a plus sign. For example:

   ALT + R

4. Option selection is shown hierarchically. For example this phrase:

   Options / Preferences / Common Options / General / Sound

   means the Sound item from General section of the Common group on the Preferences dialog box, from the Options menu.

5. Square brackets are used to designate optional entries. For example:

   [Low]

6. The characters "<" and ">" bracket required entries. For example:

   <*emitter_lead*>

7. User entries are shown in italics. For example:

   emitter_lead

8. The OR symbol (|) designates mutually exclusive alternatives. For example, PUL | EXP | SIN means PUL or EXP or SIN.

Chapter 1 Windows Basics

What's in this chapter

This chapter introduces and reviews the basics of working with Windows. It describes the common Windows structures including menus, dialog boxes, text boxes, list boxes, drop-down list boxes, option buttons, and check boxes. It also covers basic mouse and keyboard handling techniques and describes the difference between selecting and choosing.

Introduction

MC7 is a Windows program. To use the program, it is necessary to understand how Windows itself operates. Even though it is assumed that you are familiar with the Windows system, this chapter provides a brief introduction and review of the basic Windows features. If you feel a need for more information after reading this chapter, review the first chapter in the Windows User Guide that came with your Windows operating system. It has an excellent introduction to Windows. Much of what we present here is adapted from the Guide.

Parts of a window

Most MC7 operations are performed from within overlapping rectangular regions called windows. In this section we describe the parts common to most of these windows, and defer for later the more specialized MC7 functions.

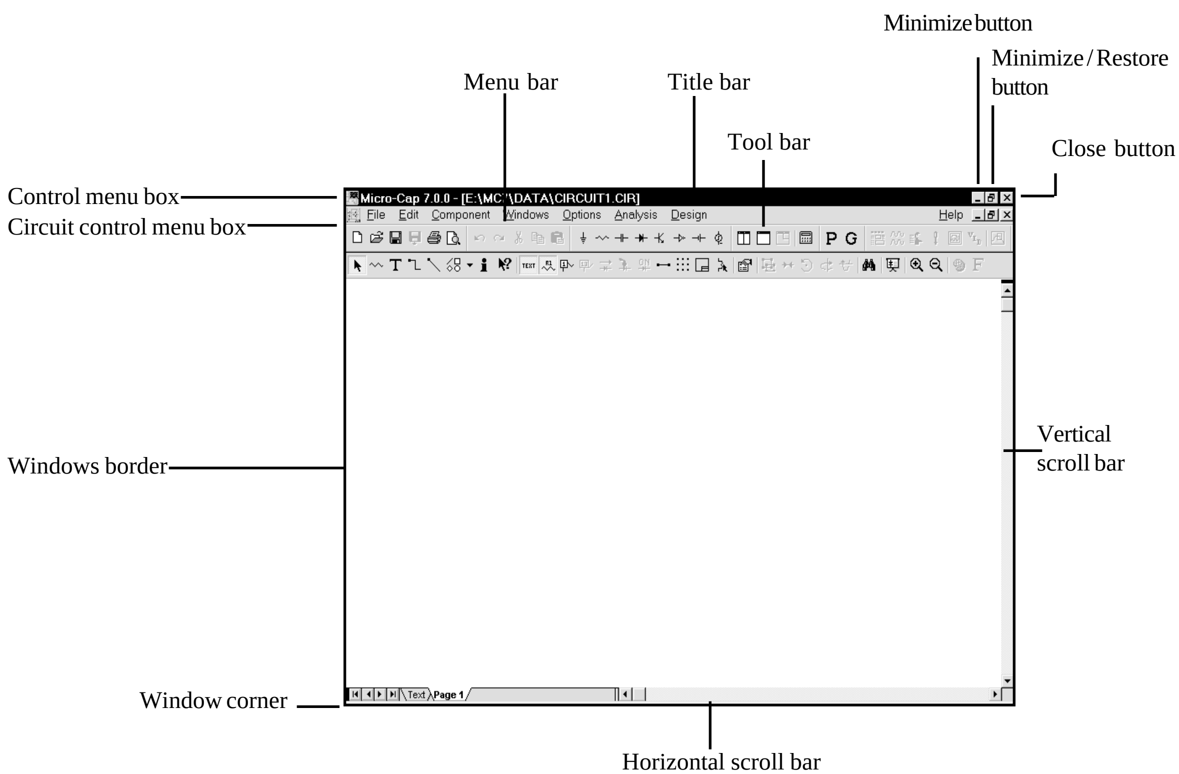


Figure 1-1 The parts of a window

The various parts of a window and their purposes are as follows:

Control menu box
The Control menu box is located in the upper-left corner of the window. This box is a standard feature of all Windows applications and is mainly used to control the MC7 window size and location on the desktop. It lets you re-size, move, maximize, minimize, and close the MC7 window. The mouse can also be used to perform these tasks by dragging.

Circuit control menu box
The Circuit control menu box is similar to the standard control-menu box except that it controls a circuit window only. There can be many circuit windows open simultaneously. This standard Windows structure is provided for management of circuit windows, but there are easier ways to manipulate circuit windows. These will be described in later chapters.

Menu bar
This bar shows the available menus. A menu is a list of commands or options that lets you access various program features. Some items on a menu have an immediate effect. For example, when you choose **Save** from the **File** menu, the current circuit is saved to disk immediately. Other menu items control deferred actions or behavior. For example, when you choose **Wire** from the **Mode** item of the **Options** menu, nothing happens immediately. When you later drag the mouse in the circuit window, it draws a wire, instead of drawing a component or text.

Title bar
The title bar shows the name of the window. If the window is a circuit window, the title shows the circuit name and directory path. If the window is a dialog or text box the title shows the name of the dialog or text box. If the window is an analysis output window, such as numeric output, the title shows the file name and path of the file where the numeric output has been saved to disk.

Tool bar
The tool bar shows the tool buttons. These are graphical equivalents of the menu items. Clicking on a tool bar button is the same as clicking on its equivalent menu item. Tool bar buttons provide convenient, quick access to frequently used menu items. Immediate action buttons temporarily depress when clicked, then spring back. Modal buttons stay depressed until another mode is chosen. A depressed modal button means that the mode is enabled. This is the same as its corresponding menu item having a check mark.

Close button
The Close button closes the window.

Maximize / Restore button
The Restore button restores the windows former size.

Minimize button
The Minimize button reduces the window to an icon.

Scroll bars
The scroll bars are used to pan the window document. If the window is a circuit, the scroll bars are one of several ways to view different parts of the circuit. If the scroll bars are part of a list box, they let you browse the list.

Window border
The window border is a control object on the outside edge of the window. When the mouse passes over the edge, the cursor changes to a vertical or horizontal double arrow. A mouse drag then moves the underlying border, changing the window size and proportions.

Window corner
The window corner is a control object on the corner of the window. When the mouse passes over the corner, the cursor changes to a diagonal double arrow. A drag here simultaneously moves the two corner sides, changing the window size.

Basic mouse and keyboard techniques

This section explains the basic terms and techniques used to select and choose items from windows, menus, list boxes, and dialog boxes. The definitions of the more common terms are as follows:

This Term	**Means**
Click	To quickly press and release a mouse button.
Double-click	To click a mouse button twice in rapid succession.
Drag	To hold the mouse button while you move the mouse.
Point	To move the mouse until the mouse pointer is at the desired location on the screen.

Mouse button means the *left* mouse button, unless otherwise specified. The right mouse button is reserved for specialized functions.

The terms choose and select have two distinct meanings. To select something means to mark it, usually as a prelude to choosing it. To choose means to use the mouse or keyboard to pick or activate the selected item, option, or action.

Selecting is done by clicking the item with the cursor or dragging the cursor over a region containing the object. Selected text in a text field or in the text area appears in reversed video. Selected schematic objects, including text objects, are shown in the user-specified select color. Selected items from a window or dialog box may be shown as a highlight, or a dotted rectangle.

To choose a selected item in a dialog box, you click it with the mouse or press the SPACEBAR key. Highlighted buttons are chosen by pressing ENTER.

Accelerator keys are provided for some of the more frequently used menu items. Like tool bar buttons, they provide a quick and ready way to choose or activate common menu items. The keys are shown on the menus, adjacent to the items they activate.

Accelerator keys are well worth learning. For frequently used features, they can save a lot of time. Appendix D has a list of the accelerator keys.

Menus

Menus provide the basic commands and options that let you use MC7 features.

To select a menu:

Mouse
Using the mouse pointer, point to the name of the menu on the menu bar, and click the left mouse button. This opens the menu and displays its contents. You can also drag the mouse from the menu name directly to the item name and release the button to choose it.

Keyboard
If the menu name contains an underlined letter, press ALT + underlined letter. Alternatively, use this method:

- Press ALT to select the menu bar.
- Press LEFT ARROW or RIGHT ARROW to select the desired menu.
- Press ENTER to open the selected menu.

To close a menu:
Click anywhere outside the menu or press the ESC key.

To choose an item from a menu:
Click the mouse on the item name or use UP ARROW or DOWN ARROW keys to select the item, and then press ENTER to choose the item.

Menu conventions:

Convention	**Meaning**
Dimmed or missing item	The item is not available or simply inappropriate. For example, the **Copy** command is dimmed when nothing is selected.
Ellipsis (...)	A dialog box appears revealing more choices to be made before the command can be completed.
Check mark	A check mark means the option is in effect.
A key combination	The key combination is a shortcut for this item.
A triangle	This indicates that a cascading menu appears listing additional choices.

Dialog boxes

Dialog boxes are used to request information about a task you are performing or to provide information you might need. An ellipsis after a menu item means a dialog box will appear when you choose that item. For example, choosing the **Open** item from the **File** menu invokes the File Open dialog box. It looks like this:

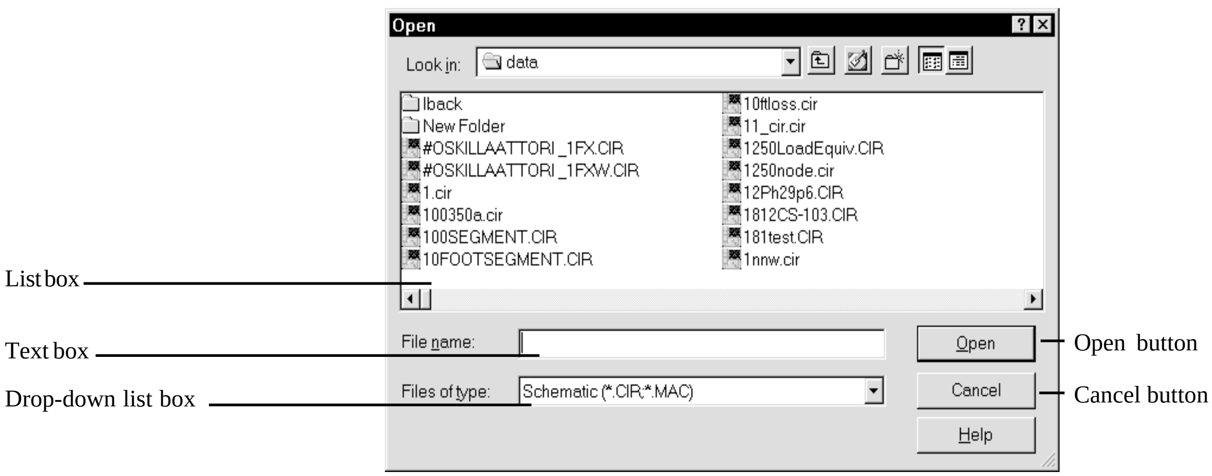


Figure 1-2 The File Open dialog box

This particular dialog box lets you choose a file to open. It also lets you choose the drive, directory, and selectively display available file names.

Dialog boxes let you choose different program features and control the program's behavior. Choices are made using several types of controls, including command buttons, text boxes, list boxes, drop-down list boxes, option buttons, and check boxes.

Dialog box option types:

Command buttons
You choose a command button to initiate immediate action, such as executing or canceling a command. The OK and Cancel buttons shown in Figure 1-2 are typical command buttons.

Text boxes
A text box is used to enter text information such as the name of a file, analysis run time, or an algebraic formula. For example, here is the text box used by the Find command.

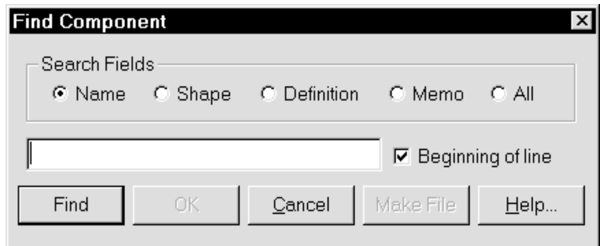


Figure 1-3 The Find text box

When the text box appears, old text is selected and any typing replaces the old text. You can also delete selected text by pressing the Delete key. If no old text exists the text cursor is placed at the left of the text field.

To select text in a text box:

Mouse
Drag the text cursor across the text to be selected. To select a single word, double-click on it.

Keyboard
1. Use the cursor arrow keys to move to the left of the first character you want to select.

2. To extend the selection, hold the SHIFT key down while pressing either the left or right arrow key. Press SHIFT + END to extend the selection to the last character in the field.

List boxes
A list box displays a list of choices. If there are more choices than can fit in the box, scroll bars appear that let you browse the remainder of the list.

Normally, you select only one item from a list box, but some list boxes let you select more.

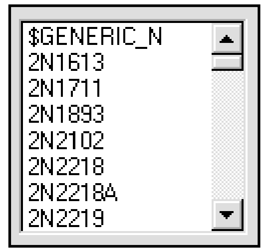


Figure 1-4 A typical list box

To select a single item from a list box:

Mouse
Click on the scroll arrows until the item you want is displayed. Click on the item you want to select. Double-click on the item to both select and choose it in a single step.

Keyboard
Use the arrow keys to scroll to the desired item or type the first letter of the item. Press ENTER to choose the selected item.

To select multiple contiguous items from a list box:

Mouse
Click on the first item. Then press the SHIFT key down and click on the last item. This selects all of the items between the first and the last, inclusive. To cancel the selection, click on any item in the list with the SHIFT key up.

Keyboard
Use the arrow keys to scroll to the desired item or type the first letter of the item. Press and hold the SHIFT key. Press the UP or DOWN ARROW keys until the items you want are selected. To cancel the selections, release the SHIFT key, and press either the UP or DOWN ARROW key.

To select multiple noncontiguous items from a list box:

Mouse
This procedure can only be done with the mouse. Press and hold CTRL and click on each item you want to select. To cancel a selection, press and hold CTRL and click on the selection you want to cancel.

Drop-down list boxes
Drop-down list boxes appear initially as a rectangular box with the current selection marked. When you select the arrow in the square box at the right, a list of alternative choices drops down. If the list is too big for the box, scroll bars appear that let you browse the list.

Drop-down list box

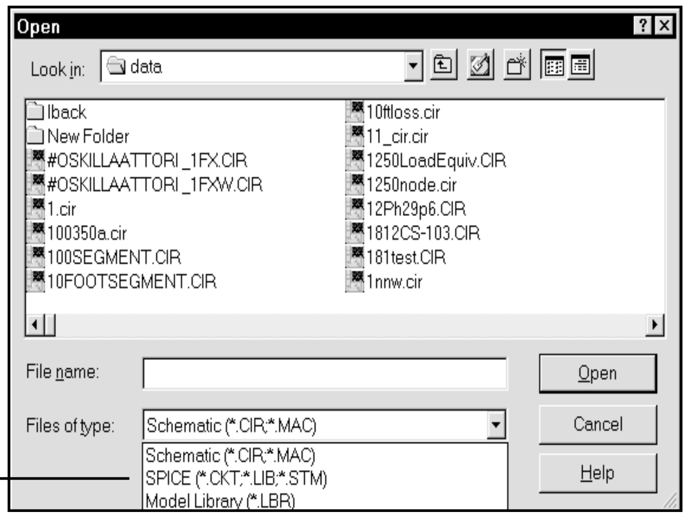


Figure 1-5 A drop-down list box

To open a drop-down list box and select an item:

Mouse
1. Click the arrow in the square box at the right to open the box.
2. Click the up or down scroll arrow, or drag the scroll box until the item you want to select is visible.
3. Click the item.

Keyboard
1. Press ALT + DOWN ARROW to open the box.
2. Press the UP or DOWN ARROW key to select the desired item.
3. Press ALT + UP ARROW or ALT + DOWN ARROW to choose it.

Option buttons

Option buttons are used when you must choose between mutually exclusive options. The selected item button contains a black dot. Items which are not available are dimmed. In the dialog box below, the Method option is chosen from the mutually exclusive group containing Linear, Log, and List.

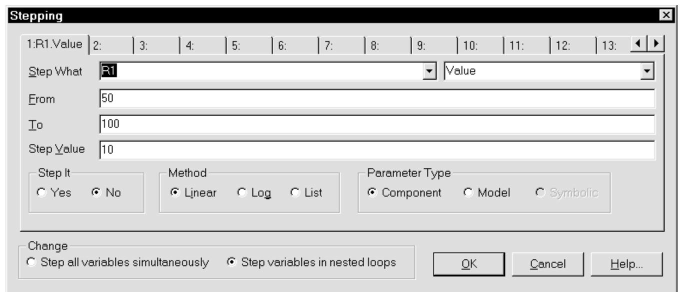


Figure 1-6 Option buttons

To select an option button:

Mouse

Click the option button.

Keyboard

1. Press TAB to move to the desired option group.
2. Use the UP or DOWN ARROW keys to select the desired button.

If the option has an underlined letter, you can select the option button by typing ALT + the underlined letter. This selects the option. Actually choosing the selected option occurs when you exit the dialog box by choosing the OK button.

Check boxes

A check box adjacent to an option means you can select or clear the option. If the box contains an X the option is selected, or in effect. If the box is empty the option is clear, or not in effect. When you select the option you toggle its state between selected and clear. The names of unavailable options are dimmed.

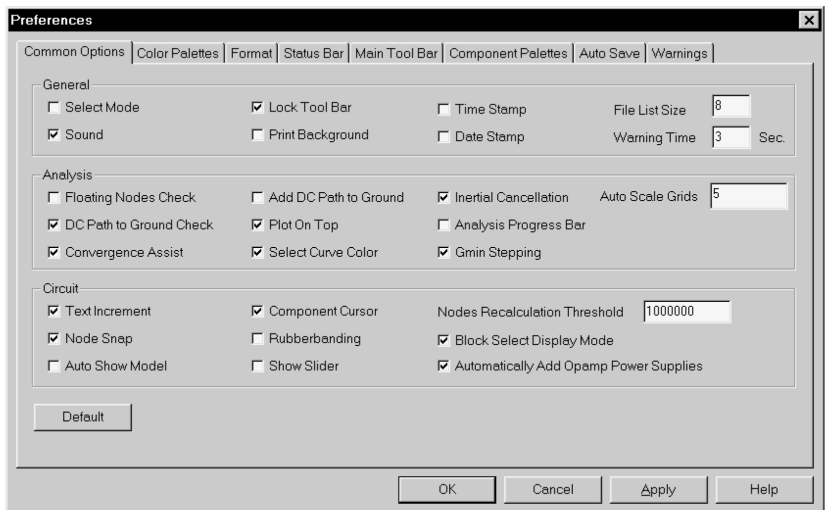


Figure 1-7 Check boxes

To select or deselect a check box:

Mouse

Click each blank check box that you want to select. Click each selected box that you want to clear.

Keyboard

1. Press TAB to move to the desired check box group.
2. Press the SPACEBAR to select the box.
3. Press the SPACEBAR again to clear the selection.

If the option has an underlined letter, you can select or clear the option by typing ALT + the underlined letter.

Chapter 2 The Circuit Editor

What's in this chapter

This chapter describes the Circuit editor, the foundation of the Micro-Cap 7 user interface. The Circuit editor provides access to all of the basic program functions, including circuit construction, analyses, and the various editors.

Features new in Micro-Cap 7

- Portable schematic files save shape, component, and model data.
- User-specified paths (folders) for circuit and model library files.
- Edit and plot within the Attribute dialog box.
- Multistage Undo and Redo commands.
- Improved Component Find command.
- Status bar shows digital expressions, current, and power at mouse location.
- Formula text.
- Smooth one-pixel schematic panning.
- Bill of materials.
- PADS PCB output.
- Model parameter range checker.
- Multiple file opener opens many files at once.
- CTRL+delete cuts wires at box boundary.
- Automatic file saver.
- Expandable component palettes.
- JPG, BMP, and GIF picture files.

Circuit structure

MC7 analyzes circuits. A circuit is an interconnection of electrical components. There are two basic types of circuit description that MC7 uses:

- Schematic, a picture of the circuit topology.
- SPICE text file, a textual description of the circuit topology.

Schematics
Schematics are comprised of drawings and text. Drawings provide the circuit description and text provides the modeling and analysis information required for an AC, DC, or transient analysis. MC7 runs analyses from data contained in the schematic. It is not necessary to convert it to a SPICE netlist first, although MC7 can also analyze SPICE netlists. Sometimes MC7 also uses modeling information from libraries referenced in the schematic.

Each schematic contains a drawing area and a text area.

Drawing area
The drawing area contains one or more pages of analog and/or digital components interconnected by wires. It may also contain graphical objects (which contain no electrical information) and grid text (which may contain electrical information). Grid text can be moved from the drawing area to the text area and vice-versa by selecting it and pressing CTRL + B. This is called the *shuttle* command. Pages can be added or deleted from the Edit menu or with a right click on any page tab in the page selector at the bottom of the schematic window. Page display is controlled by the page scroll bar located at the lower left part of the circuit window.

Text area
The text area contains only text. It can hold *local* command statements, subcircuit descriptions, model statements, and digital stimulus statements that may be too bulky for convenient display in the drawing area. There is only one page, but it can be as long as you like. The editor is capable of handling multi-megabyte text files.

The display can be toggled between the text area and the current schematic page by pressing CTRL + G. The text area can be selected by clicking on the text area tab in the page scroll bar area. To return to the schematic, click on one of the desired page tabs or press CTRL + G.

You can split the display to simultaneously show different parts of the drawing and/or text areas by dragging the vertical or horizontal splitter bar towards the middle of the window.

SPICE text files

SPICE text files are standard SPICE text descriptions of a circuit, a subcircuit, or a model statement. MC7 generally follows the original Berkeley SPICE2G format, with many PSpice™, SPICE3, and even a few HSPICE extensions. Text files are entered and edited in a text document similar to what a text editor or a word processor creates. When you create a new SPICE text file, MC7 opens a text document and lets you edit it as desired. Only when you start an analysis does MC7 error check it.

Note that MC7 can also translate schematics into SPICE text files of various flavors, in case you want to check the accuracy of an external SPICE simulator against MC7. MC7 will, of course, accept such files for analysis.

Schematic objects

Object, in the context used here, is a general term for anything that can be placed in a schematic. There are several types of objects:

• **Components:** This includes all analog and digital sources, active and passive parts, and connectors. In short, it includes any object chosen from the Component menu.

• **Wires:** Wires are used to interconnect components. There are two types of wires: orthogonal and diagonal. Orthogonal wires are restricted to the vertical or horizontal direction. They may contain both a vertical and a horizontal segment. Diagonal wires are drawn at any angle and only contain one segment.

• **Text:** Text, sometimes referred to as grid text because its origin is restricted to an imaginary grid, is used for naming nodes, creating command statements, and general documentation.

• **Graphics:** Graphical objects include lines, rectangles, ellipses, diamonds, arcs, pies, and pictures. They are used for aesthetic or documentation purposes and have no effect on the electrical behavior of the circuit.

Schematics may contain picture files in any of the following formats: WMF, EMF, JPG, BMP, and GIF. The most common use of this feature is to place a picture of an analysis plot in a schematic. Picture files of any MC7 window with graphics may be captured by the **Copy The Entire Window to a Picture File** command on the **Edit** menu. Once the picture file has been created, it can be included in a schematic as a graphic object.

• **Flags:** Flags mark locations in a schematic that you are likely to want to view many times. They facilitate rapid switching of the display between multiple circuit locations. They are most useful in very large circuits, where the display redraw time may be too long for convenient use of the scroll bars.

These are the only objects that can be placed in a schematic.

Note that grid text is the only object that can be shuttled between the drawing and text areas.

Schematic modes

The Schematic editor is a modal editor. That means the behavior of the mouse when it is clicked or dragged in the schematic depends upon what mode it is in. Schematic modes affect either the mouse behavior or the schematic display.

- Behavioral modes
  - Circuit altering modes
    - Select mode
    - Component mode
    - Text mode
    - Wire mode
    - Diagonal wire mode
    - Graphics / Picture File mode
    - Flag mode

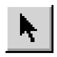
  - Circuit querying modes
    - Info mode
    - Help mode
    - Point to end paths mode
    - Point to point paths mode

- View modes
  - Attribute Text
  - Grid Text
  - Node Numbers
  - Node Voltages / States
  - Current
  - Power
  - Condition
  - Pin Connections
  - Grid
  - Cross-hair Cursor
  - Border
  - Title

The behavioral modes are mutually exclusive. At any given moment, only one mode is active. You can tell which mode is active by looking at the mode buttons. An active mode button is pushed in. An inactive mode button is pushed out.

To select a mode click its button. This deselects the prior mode button.

Behavioral modes:



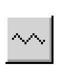
Select mode: This selects an object, region, or location for one of several actions:

- Editing the selected object (Requires a double click)
- Clearing the selected region (Deleting without copying to the clipboard)
- Cutting the selected region (Deleting with copying to the clipboard)
- Moving the selected region
- Rotating the selected region
- Stepping the selected region
- Mirroring the selected region
- Making a macro of the selected region
- Copying the selected region to the clipboard
- Making a BMP file of the selected region
- Drag copying the selected region
- Moving the selected object to front or back
- Shuttling the selected text object between the text and drawing areas
- Changing the selected object's color or font
- Defining the upper-left corner position of a subsequent paste operation

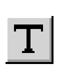
To select an object you must first be in Select mode. To enter Select mode:

- Click on the Select mode button  or press CTRL + E.
- Press the SPACEBAR key. Press it again to restore the original mode.
- Press and hold the SHIFT or CTRL key down. In this mode you can select an object or a region, but you can't drag the mouse. When the key is released, the original mode is restored.

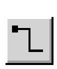
To select a contiguous group of objects, drag the mouse over the region containing the objects. To select a noncontiguous group of objects, hold down the SHIFT button and click the mouse on each object.



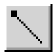
Component mode: This mode lets you add components to the schematic. The component that gets added is the last one chosen.



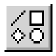
Text mode: This mode lets you add grid text to the schematic. Grid text is used for node naming, defining electrical characteristics, and documentation.



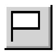
Wire mode: This mode lets you add orthogonal wires to the schematic. Orthogonal wires can have one or two segments and be vertical or horizontal or both.



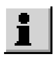
Diagonal wire mode: This mode lets you add diagonal wires to the schematic.



Graphics/Picture File mode: This mode lets you add a graphical object (line, rectangle, ellipse, pie, arc, diamond, or picture file) to a circuit. Picture files are typically created with the **Edit / Copy The Entire Window to a Picture File** command.

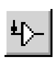


Flag mode: This mode lets you add flag markers for rapid navigation of large schematics with many pages.

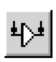


Info mode: This mode provides model information about a part when you click on it. The information is usually a model, subckt, or define command statement for the part. If the part is a macro, the macro circuit which it represents is displayed. If it is a subckt, the subcircuit description is displayed. If it is a device that uses a model statement, the model statement is displayed. Double-clicking on the part also displays model information in the Attribute dialog box.

Help mode: In this mode, you click on a component to display its parameter or attribute syntax. ALT + F1 gives syntax help on selected SPICE part names.



Point to End Paths: In this mode, you click on a digital component and MC7 traces all of the digital paths that originate at the component and end when they drive no other combinatorial component.

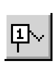


Point to Point Paths: In this mode, you click on two successive digital components and have MC7 trace all of the digital paths between the two components.

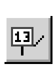
View modes:

Attribute text mode: This mode controls the display of all attribute text. Attribute text will be displayed only if the individual attribute display check boxes are enabled and the Attribute text mode is enabled.

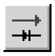
Grid text mode: This mode controls the display of all grid text. If disabled, all grid text is invisible. If enabled, all grid text is visible except command text, which is visible only if this button and the command text button are both enabled.



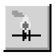
Node numbers: This mode displays the numbers assigned by MC7 and used to identify circuit nodes.



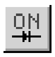
Node Voltages / States: This mode displays time-domain values of node voltages for analog nodes and node states for digital nodes.



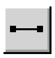
Current: This mode shows the last time-domain currents for each component. Currents are shown with an arrow indicating the direction of positive current.



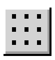
Power: This mode displays the last time-domain values of stored, generated, and dissipated power for each component.



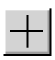
Condition: This mode displays the last time-domain condition for each component which has conditions. Typical conditions for a BJT are LIN (linear), SAT (saturated), OFF (both junctions off), and HOT (excessive power dissipation).



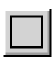
Pin connections: This mode marks pin locations with a dot. These are the places where connections are made to wires or other component pins.



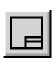
Grid: This mode displays the schematic grid to which objects are anchored. All wires, components, and other objects originate on a grid.



Cross-hair cursor: This mode adds a full-screen schematic cross-hair cursor. Its principle use is in aligning components.



Border: This mode adds a border around each printed sheet of the schematic.



Title: This mode adds a title block to the corner of each schematic sheet. There is an option to have only one title block per page. This option is accessible from the Properties dialog box, accessed with F10.

Circuit editor

The Circuit editor creates and edit circuits. When the circuit is a schematic, it becomes the Schematic editor and its display looks like the figure below:

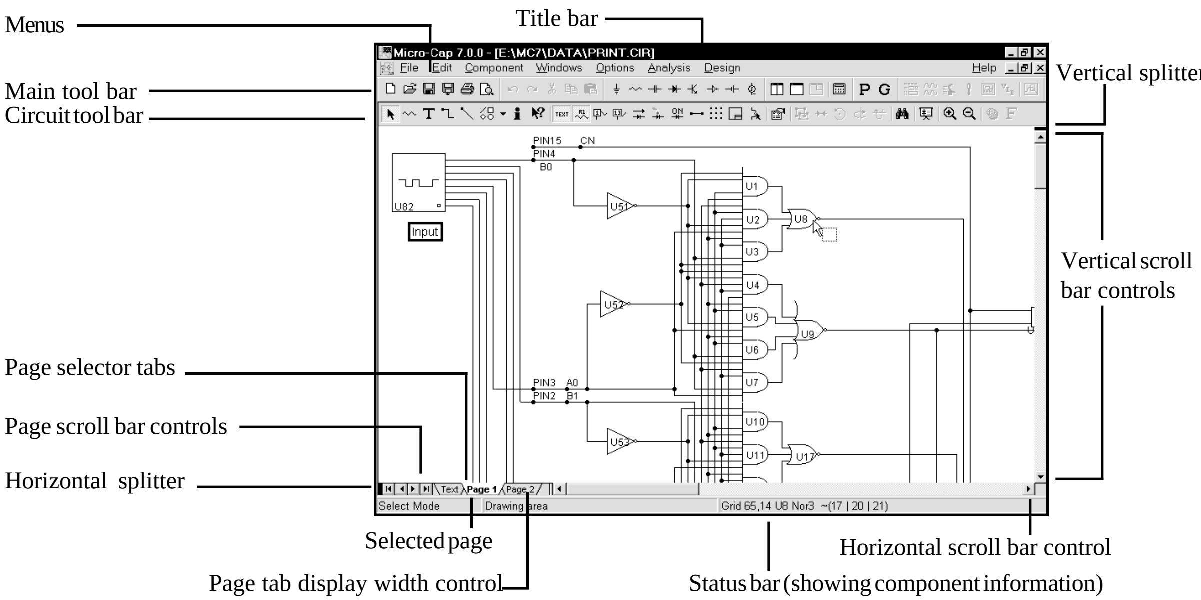


Figure 2-1 The Schematic editor

The principal components of the editor are as follows:

Menus: Located beneath the title bar, menus select major functional groups.

Title bar: Located in the top center of the window, the title bar displays the name of the window. The mouse can be used to move the window by dragging the title bar.

Main tool bar: Shown below the title bar, the main tool bar contains buttons that represent the most frequently used options, such as file operations and access to the various editors. The tool bar provides a quick way to access those options. Clicking on a button will activate the desired option.

Circuit tool bar: Shown below the main tool bar, the circuit tool bar contains buttons that represent the most frequently used menu options that apply specifically to the circuit. The tool bar provides a quick way to access those options. Clicking on a button will activate the desired option.

Schematic scroll bars: These scroll bars, located to the right and the bottom of the window, let you pan the schematic vertically or horizontally. Clicking on the scroll arrows pans the display roughly 10% of the window width per mouse click. Moving the scroll box, either by dragging it or by clicking in the scroll bar area, moves the view to a location proportionate to the new scroll box position relative to the scroll bar end points.

Page selector tabs: Located in the lower left of the window, the page selector tabs let you select a particular page for viewing.

Selected page: The color of the selected schematic page tab is the window background color.

Text area tab: The text area is selected by clicking on the Text tab. CTRL + G may also be used.

Splitter bars: Dragging these bars across the window splits it into two parts, with a vertical or horizontal bar between. Each part can be independently scrolled and toggled between text and drawing areas. Only one split section at a time can display the text area. Each split section can be scrolled or panned to view different parts of the schematic.

Status bar: This bar describes the items the mouse pointer is currently passing over. It is also used to print informative messages.

When the mouse is over a part, the center pane will show its Component library memo field if it has any descriptive text.

The pane at the right shows the part type and information about its last current and power values.

If the part is a digital primitive gate, it will print the logic expression.

Selecting and deselecting objects

To manipulate an object after it has been placed in the schematic, it must first be selected. Selection or de-selection requires the use of the mouse and follows the general Windows rules.

First step: If not already in Select mode, click on the Select mode button  in the Tool bar or press CTRL + E. You can also toggle between Select mode and the current mode using the SPACEBAR key.

Second step:

To select a single object:
Click on the object. The object is selected.

To select or deselect a single object:
Press SHIFT and click on the object. The object's selection state is toggled. If the object was formerly selected, it becomes deselected. If it was formerly deselected, it becomes selected.

To select multiple contiguous objects:
Drag the mouse creating a region box. This selects all objects that originate within the box.

To select or deselect multiple contiguous objects:
Press SHIFT and drag the mouse creating a region box. This toggles the selection state of all objects originating within the box.

To select or deselect multiple noncontiguous objects:
Press and hold SHIFT while clicking on each individual object or dragging on each group whose selection status you want to change. This toggles the selection state of the object or group.

When would you want to deselect specific objects? Sometimes it is necessary to select most but not all of a large group of contiguous objects. It is easier and faster to drag select the entire group and then deselect one or two of them than to individually select each of the desired objects.

If you click on a component's attribute text, the text will be selected, not the component. Be careful to click within the component where there is no attribute text. Single-clicking on attribute text selects it and lets you move it. Double-clicking on attribute text opens a local edit box at the text location and lets you edit it there without using the Attribute dialog box.

To select all objects in the window, press CTRL + A. To deselect all objects, click on any empty space in the window.

Rapid switching to/from Select mode:
In the course of building or editing a schematic, it often happens that you need to rapidly switch between modes, especially between Select mode and Component or Wire mode. To make this process easier several short term mode switchers are available:

- To temporarily suspend the current mode and enable Select mode, press and hold the SHIFT key down. You can use this pseudo Select mode for object selection only. You can't drag objects. When the SHIFT key is released the former mode will be restored.

- To temporarily suspend the current mode and enable Select mode, press and hold the CTRL key down. The mode will change to Select. You can use the Select mode as needed. When the CTRL key is released the former mode will be restored. Note that under this mode you can't move objects with the mouse. CTRL + mouse drag is used to drag copy an object.

- To suspend the current mode and switch to Select mode, press the SPACEBAR key. It is not necessary to hold the key down. The mode will change to Select. You can use the Select mode as needed, then press the Spacebar when you're finished and the former mode will be restored.

SPACEBAR is by far the easiest method of alternating between Select and other modes.

Creating a new circuit

To create a new circuit, select the **New** item from the **File** menu. This presents the New dialog box which looks like this:

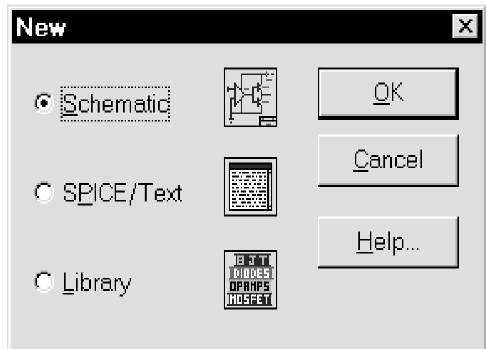


Figure 2-2 The New dialog box

This dialog box lets you create one of three types of files:

- Schematics: These are combined drawing and text documents.
- SPICE/Text files: These are text documents. They include SPICE circuit descriptions, and subcircuit and model statements for the Digital Library, Analog Vendor Library, and Spectrum supplied model libraries.
- Library: These are binary files that store the model parameters for some of the Analog Library.

To create a particular type of file, click on the item and then click on the OK button. This opens an empty window of the appropriate type and gives it a generic file name.

Adding and editing components

To add a component to a schematic:

First step: Select a component from the Component menu, from one of the component buttons in the Tool bar, from one of the Component palettes, or from the Find Component dialog box (SHIFT + CTRL + F). You can skip this step if the component you want has already been selected by a prior step. The main part of the Component menu is divided into six broad categories:

1. Analog Primitives
2. Analog Library
3. Digital Primitives
4. Digital Library
5. Animation
6. Filters (Visible after creating a filter macro.)
7. Macros (Visible after creating a macro with the Make Macro command.)

The *Analog Primitives* section contains each of the basic analog components like resistor, capacitor, or NPN transistor. These parts all require the user to provide the defining electrical attributes such as resistance, capacitance, or a suitable model statement. Model parameters can be entered directly from the Attribute dialog box, or you can select a pre-modeled part from the Analog Library by picking its name from the Model list box. Editing a model statement localizes the information by saving a copy of the edited model statement in the circuit text area. The same applies to subcircuits. Editing a subckt from the Attribute dialog box causes a copy to be placed in the text area of the circuit.

The *Analog Library* section contains pre-modeled analog components referenced by part names similar to their commercial names. In this section, you'll find parts like IRF510, 2N2222A, and 1N914. These parts have electrical attributes already defined in subckts, macros, or model statements somewhere in one of the model library files referenced by the master index file, ".LIB NOM.LIB", which MC7 automatically includes in every circuit file.

The *Digital Primitives* section contains one each of the basic digital components like AND gates, OR gates, flip-flops, PLAs, and stimulus sources. These parts also require the user to provide a suitable model statement, or to rely on .LIB statements to access modeling information.

The *Digital Library* section contains commercial digital components with part names similar to their corresponding commercial part names. In this section, you'll find parts like 7400, 7475, and 74LS381. These parts also have electrical attributes defined in the model library files referenced by the file, ".LIB NOM.LIB".

The *Animation* section contains components that are used to provide behavioral animation. These include LEDs, seven segment displays, and digital switches. The *Filters* section contains filter macros created by the filter design function, accessed from the Design menu. The *Macros* section holds macros created with the Make Macros command.

Second step: Make sure MC7 is in the Component mode. Selecting a part from the Component menu will automatically switch to Component mode. If MC7 is in the Component mode, its Tool bar button  will be depressed. If you are not sure, click on the button in the Tool bar, or press CTRL + D.

Third step: Click the mouse in the schematic and drag the component to the desired location. Click the right mouse button until the part is oriented the way you want. Release the button when the part is where you want it.

Fourth step: Enter the appropriate attributes into the Attribute dialog box. If the part is from the Animation, Filters, Analog Library, or Digital Library sections, this step is unnecessary and the Attribute dialog box will not even appear. If the part is from the Analog Primitives or the Digital Primitives sections, this step lets you enter important information like the part name and value or model name.

To edit a component's attributes:

Click on the Select button  in the Tool bar. Double-click on the component shape in the schematic. This invokes the Attribute dialog box. Click on the name of the attribute that you want to edit, then click in the Value field and type in any changes. You can also change the model name by choosing a new model name from the Model list box.

Subcircuits and model parameters can be edited from within the dialog box. Any edits localize the information by placing a copy of the model statement or subckt within the text area of the circuit.

You can change an attribute's font by first selecting the attribute and then clicking on the Font button and choosing a new font, style, size, or effect. You can also change the display flag for attribute text and pin names. Choose OK to put your changes into effect.

The Attribute dialog box

When you add a component that has not been modeled, the Attribute dialog box is presented to let you enter certain component information. The exact information required varies with the component but generally a part name and a value are the minimum required for the simpler parts like resistors, capacitors, and inductors. More complex parts usually require a part name and a model statement. Digital parts sometimes require logic expressions, block names, or stimulus table names. The number of attributes is determined by the type of component. Figure 2-3 shows the dialog box as it appears for a typical NJFET primitive.

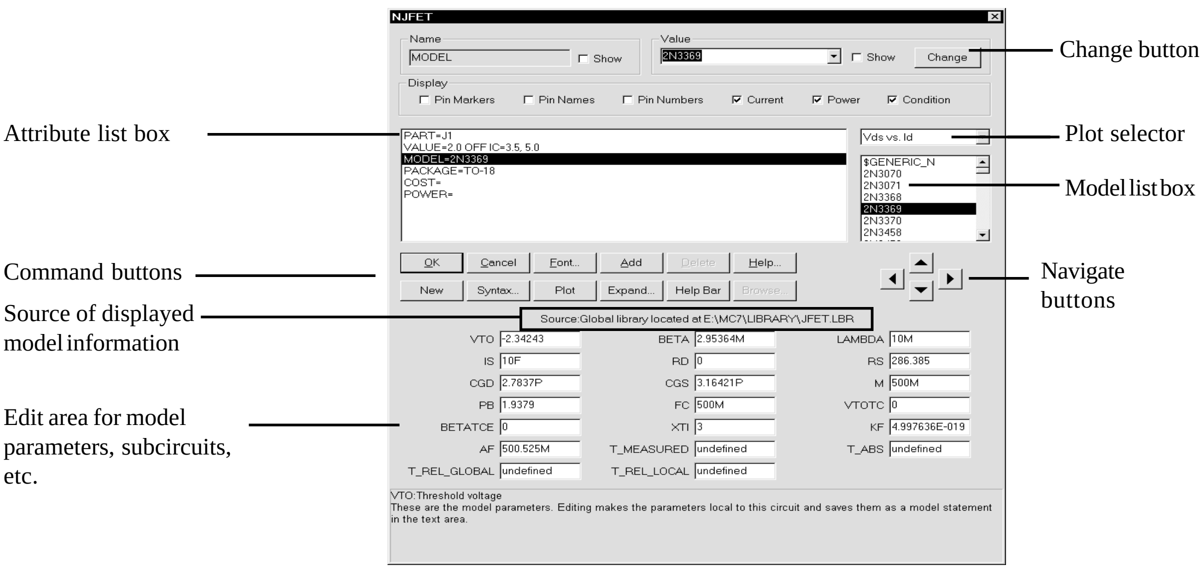


Figure 2-3 The Attribute dialog box

The attributes are shown in the *Attribute list box*. For an NJFET transistor the attributes are: PART, VALUE, MODEL, PACKAGE, COST, and POWER. Each attribute has a name and a value as shown in the table.

Attribute Name	**Attribute Value**
PART	J1
VALUE	2.0 OFF IC=3.5, 5.0
MODEL	2N3369
PACKAGE	TO-18

The first attribute is the part name. The optional second attribute is a text string describing the area, off flag, and initial conditions. The MODEL attribute specifies the model name which in turn specifies a set of model parameters which may be located locally within the circuit in the text or drawing areas of the schematic, or may be located globally in the libraries. The PACKAGE attribute specifies the physical package used by the part. Package information is used to create netlists for external PCB (Printed Circuit Board) programs.

The COST and POWER attributes specify the unit cost and power budget for the Bill of Materials report. This is a type of netlist report that shows the parts count, cost, and power budget of the circuit.

To edit a particular attribute value, select it by clicking on the attribute name in the Attribute list box. Next, click in the Value field and edit it as desired.

You can add other attributes. These have no effect on the electrical behavior of the device. To add an attribute, click on the Add button. This adds a new attribute and places the name 'USER' in the Name field. The Value field is left blank. You can then edit the attribute Name and Value fields.

To delete an attribute, select it by clicking on its name in the Attribute list box. Then click on the Delete button. Only user-added attributes may be deleted.

To change an attribute font, size, style, color, or effects, click on the Font button. This displays the Font dialog box. The Font dialog box lets you edit text features. It is described later in this chapter.

The *Model list box* displays a list of available model names from the libraries. You can scroll through the list and choose a model by clicking on it.

Command buttons provide the following functions:

> **OK:** This accepts any edits that have been made and exits the dialog box.
>
> **Cancel:** This cancels any edits that have been made and exits the dialog box.
>
> **Font:** This lets you alter the font and style of the selected attribute.
>
> **Add:** This adds a new attribute.
>
> **Delete:** This deletes the selected attribute. Only user-added attributes may be deleted.

Help: This accesses the help topics for the Attribute dialog box.

New: If the MODEL attribute is currently selected, this creates a new model name local to the circuit. You can also create a new model simply by typing in a model name not already in the library.

Syntax: This displays the syntax for the primitive from the help file.

Plot: This produces zero to three plots for each basic part type (primitive). The plots are created on the fly from mini-simulations of test circuits designed to produce the plot in question. Some parts, like the Pulse source, have a simple simulation and a single plot (two cycles of its waveform). Others, like the NPN, have several plots that can be chosen from the plot selector. Some parts, like macros and subckts, have no plot at all. Once the plot is displayed, it responds dynamically to parameter edits.

Expand: This button causes the data field where the text cursor is located to expand into a much larger text box. This lets you enter or edit a large piece of text. Normally, this is used only when an attribute value field requires a very large body of text, as for example, a table source with many pairs of data or a digital stimulus with many entries. These situations can also be handled by simply entering an alias in the value field and adding a .define statement in the text area that equates the alias with the larger table.

Help Bar: This toggles the display of the Help bar. Its job is to give a brief explanation of the field, button, or control under the cursor.

Browse: This button lets you browse when working with FILE attributes.

The *Edit area* lets you edit model parameters, subcircuits, digital stimulus statements, and other model information.

The *Show check boxes* adjacent to the Name and Value fields let you control whether the selected attribute's name and value are displayed on the schematic. Normally the attribute name is not displayed and only some of the attribute values are. Typically, only the part attribute value (in this case 'J1') is displayed, although occasionally the model attribute value (in this case '2N3369') is also.

The *Pin Markers* check box controls the display of markers showing the location of the pin connection points where wires would connect the pin to other circuit nodes. This is helpful when the actual pin location is not obvious.

The *Pin Names* check box controls the display of pin names. Sometimes this is helpful for vendor-supplied subcircuit part models where node numbers are used instead of more meaningful names. Normally this option is turned off.

The *Pin Numbers* check box controls the display of package pin numbers. This is primarily used to identify the physical pin numbers for PCB work. Normally this option is turned off.

The *Current* check box controls the display of the last time domain current value for the part, if the Current display button  is enabled.

The *Power* check box controls the display of the last time domain power value for the part, if the Power display button  is enabled.

The *Condition* check box controls the display of the last time domain condition (ON, OFF, etc.) for the part, if the Condition display button 

is enabled.

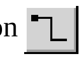
The *Change* button lets you copy the displayed attribute value to the attribute fields of parts selected from a list. For example, you could use this command to change all 2N2222 model attributes to 2N3903.

The *Plot Selector* lets you select from a list of characteristic plots for the device. Some devices have as many as three plots, while some have none.

The *Navigate* buttons scroll through other parts in the circuit. Each click on the left or right navigate button will display another part of the same type. Each click on the up or down navigate button will display a different part type.

Adding wires to a schematic

Wires are added to a schematic by dragging the mouse from one point to another. If the mouse moves horizontally or vertically a straight wire is created. If it moves diagonally a corner is created.

First step: Click on the Wire mode button  in the Tool bar.

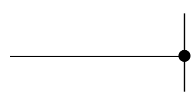
Second step: Point the mouse at the desired starting position and press and hold the left mouse button down.

Third step: Drag the mouse to the desired end point of the wire and release the left mouse button.

Several points are worth keeping in mind:

- The segment orientation of a two-segment wire can be changed by clicking the right mouse button before the wire is completed (before the left mouse button is released).

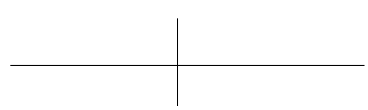
- Connections are indicated by a dot.



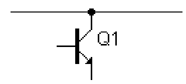
- If two wires are connected at their end points the wires fuse together and become one wire. The connection dot then disappears.

- If either end point of a wire touches another wire, the wires are connected.

- Wires that cross at interior points do not connect. An end point of a wire can connect only to another end point, a component lead, or an interior point of another wire.



• A wire that crosses another wire's end point or a component's lead dot will connect to the wire or component lead.



• If the Node Snap option is enabled, wires will snap to the nearest node within a one grid distance. Although this is usually helpful with analog parts, it is sometimes a problem with dense digital connections on a one grid pitch. You may want to temporarily disable this option if you notice a wire end point jumping to an undesired location. The Node Snap option is enabled or disabled from **Options / Preferences / Common Options / Circuit / Node Snap**.

• Although wires are the principal method of interconnecting nodes, grid text may also be used. *Two nodes with the same piece of grid text placed on them are automatically connected together*. This feature is especially useful in cases where you have many components sharing a common node, such as analog VDD and ground nodes, or digital clock, preset, and clear nodes.

• Holding the Shift key down *after* starting a wire forces the wire to stay horizontal or vertical, depending upon its principal direction prior to pressing the Shift key. It is necessary to press the Shift key *after* starting the wire, since pressing it *before* starting the wire would temporarily exit Wire mode and enter Select mode.

Adding and editing grid text

There are several kinds of text visible in a schematic. First, there is the attribute text of individual components. This type of text is created and edited from within the Attribute dialog box. The second kind of text is called grid text. Although grid text can be used for any purpose, its most important use is for naming nodes.

To add grid text to a schematic:

First step: Click on the Text mode button  in the Tool bar.

Second step: Click the mouse in the schematic where you want to add the text. A Text dialog box will appear. It offers the following options:

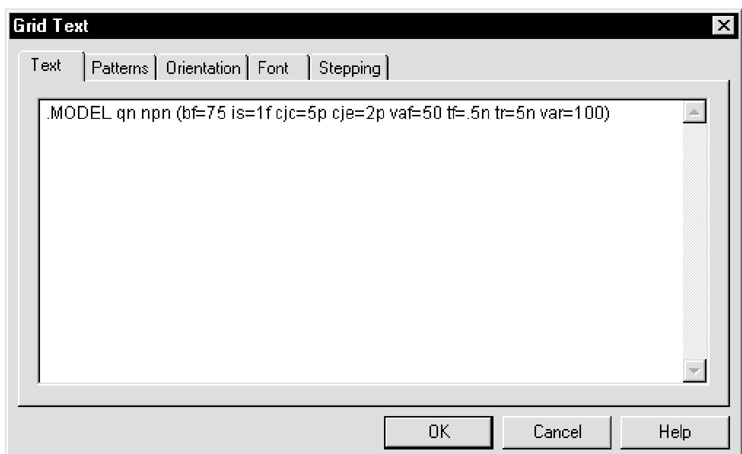


Figure 2-4 The Text dialog box

Command buttons

OK: This accepts user text entry or edits and places the new or edited text at the mouse site.

Cancel: This ignores any user edits and exits the dialog box.

Help: This accesses the Help system.

Options

Text: This lets you edit the text content.

Patterns: This lets you edit the text color, fill (background) color, and the border width and color.

Orientation: This lets you edit the print order / orientation. There are four choices. Normal specifies left to right order. Up specifies bottom to top order. Down specifies top to bottom order. Upside Down specifies right to left order and upside down orientation.

Font: This lets you edit the text font, size, style, and effects.

Stepping: These options are present only when adding text. They let you add multiple rows and/or columns of incremented text.

Instances X: This sets the number of rows of text.

Pitch X: This specifies the spacing in number of grids between rows of text.

Instances Y: This sets the number of columns of text.

Pitch Y: This specifies the spacing in number of grids between columns of text.

Stepping is used to name wires or nodes with grid text. For example, if you wanted sixteen names A0, ... A15 vertically arranged, you enter text of "A0", select Instances X = 1, Instances Y = 16, and Pitch Y = 2 (grids).

To add or edit existing text, select the Text panel of the dialog box. Type in or edit the text and press ENTER. To make a piece of text start on a new line place the insertion point text cursor at the desired position and press ENTER.

Grid text can be moved between text and drawing areas by selecting the text, then pressing CTRL + B. The display of the schematic window can be toggled between the text and drawing areas by typing CTRL + G.

To edit grid text:

First step: Click on the Select mode button  in the Tool bar.

Second step: Double-click the mouse on the text you want to edit. A Text dialog box will appear showing the selected text. Edit the text as desired.

Formula text

MC7 provides a special kind of schematic grid text that can be used to calculate formulas. Its form is as follows:

=formula

Grid text of this form behaves like a small spreadsheet. The presence of the "=" at the first character position tells MC7 that the text is really a formula and that it should calculate the value of the formula every time there is any change to schematic text. For example, the following four pieces of text define three variables, L1, C1, and C2 and a piece of formula text:

```
.DEFINE C1 1N
.DEFINE C2 1N
.DEFINE L1 10U
=2*PI/SQRT(L1*C1*C2/(C1+C2))
```

Note that these are individual pieces of text, not a single block of text. The formula text must be placed in the schematic, not the text area. The .define statements may be placed in either the text area or in the schematic.

The formula is for the resonant frequency of a Colpitts oscillator. Any change to any text updates the value and MC7 prints the following:

```
2*PI/SQRT(L1*C1*C2/(C1+C2))=88.857E6
```

Formula text is designed to provide a convenient way to automatically compute and display new design values from complicated formulas. For example, if you change the value of C1 to 10nF, MC7 prints a new resonant frequency as follows:

```
2*PI/SQRT(L1*C1*C2/(C1+C2))=65.898E6
```

Formula text is entirely visual. The values can not be used in other expressions. To do that the formula must be defined with a symbolic variable name like this:

```
.DEFINE  F0  2*PI/SQRT(L1*C1*C2/(C1+C2))
```

F0 could now be used in an expression or in, for example, a capacitor's VALUE attribute.

Adding and editing graphic objects

First step: Click on the Graphics mode button in the Tool bar. A menu appears, listing the available graphic objects. Choose one of the objects by clicking on it. Drag the mouse in the schematic to create the object.

Graphic objects include the line, ellipse, rectangle, diamond, arc, pie, and picture. All have handles which can be dragged to alter the size and shape of the object. Drag on a corner handle to change the size of the object. Drag on a non-corner handle to change the width or length of the object's bounding box. This lets you change the aspect ratio or relative shape of the object.

Double-clicking on a graphics object invokes a dialog box for editing the object.

If the object is a picture, a file dialog box will be presented. Select the name of the picture file, border and fill options, and then click on the OK button.

Here is an example of a circuit file with a JPG picture of Mona Lisa.

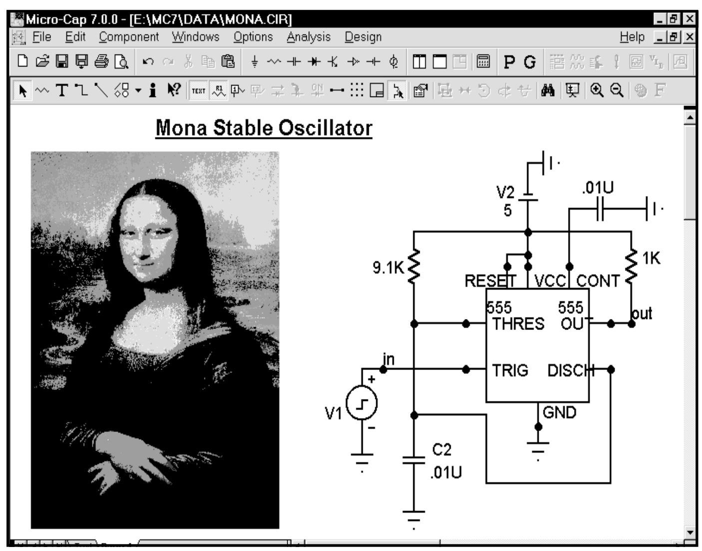


Figure 2-5 A circuit with a JPG file

MC7 can import pictures in JPG, BMP, GIF, WMF, and EMF formats and can export (create) pictures in WMF, EMF, and BMP formats.

Adding and editing flags

First step: Click on the Flag mode button in the Tool bar.

Second step: Click in the schematic at the position where you want to place the flag. A text box is presented. Type in the name of the flag and click OK.

To edit a flag, double-click on it and edit the name in the text box as desired.

Moving schematic objects

First step: Click on the Select mode button in the Tool bar.

Second step: Select the object or group of objects.

Third step: Drag the selected object or group. Slide the mouse across the drawing to the desired location and release the mouse button. Be sure to drag within the body of the object, or within the selected region. If you don't, you will deselect whatever had been selected. If this happens go to the second step and try again.

Rotating schematic objects

To rotate a single component, click on it and hold the left mouse button down, then click on the right mouse button. Each click rotates the part 90 degrees.

To rotate a group of objects follow the following procedure:

First step: Click on the Select mode button in the Tool bar.

Second step: Select the object or group of objects.

Third step: Click on one of the three rotate buttons. The Flip X axis button rotates the region about the horizontal axis in 180 degree increments, flipping it about the horizontal axis. The Flip Y axis button rotates the region about the vertical axis in 180 degree increments, flipping it about the vertical axis. The Z axis button rotates the region about the Z axis (the Z axis is perpendicular to the schematic plane) in 90 degree increments, producing four distinct orientations.

Stepping schematic objects

Stepping lets you define a rectangular region and then duplicate its contents a specified number of times. It is called stepping after the photolithographic term 'step and repeat' used in the semiconductor industry. This feature is most useful for creating highly repetitive circuit structures such as PLA, RAM, or ROM. The stepping procedure is as follows:

First step: Click on the Select mode button in the Tool bar.

Second step: Drag the mouse to define a region containing the object or group of objects that you want to step. This creates a box encompassing the desired area. Normally, when you first create the box, you draw its boundaries just outside the area of interest, because if you touch one of the objects, you end up dragging it instead of creating the box. If you step with the box this way, you'll end up with unconnected sections due to the space between the objects and the original box boundaries. This is usually not what you want. The way around this is to use the box handles. These handles are on all four corners and each side of the box. After the initial box creation, the handles can be dragged to eliminate the spaces. This lets the stepping process create contiguous, connected sections.

Third step: Click on the Stepping button, to invoke its dialog box.

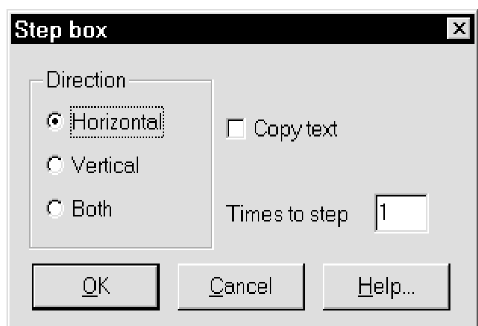


Figure 2-6 The Stepping dialog box

This dialog box lets you make three choices; the direction in which to step the box, the number of times to copy it, and whether or not you want to copy the box's grid text. All attributes of components found in the box are copied, except Part name, which is always incremented. Grid text, if copied, will be incremented if the Text Increment option in the Preferences dialog box is enabled.

Mirroring schematic objects

Mirroring lets you define a rectangular region called a box and then copy a reflected image of the box contents. The mirroring procedure is as follows:

First step: Click on the Select mode button  in the Tool bar.

Second step: Drag the mouse to define a region containing the object or group of objects that you want to mirror. This creates a box encompassing the desired area. Normally, when you first create the box, you draw its boundaries just outside the area of interest, because if you touch one of the objects, you end up dragging it instead of creating the box. If you mirror this way, you'll end up with unconnected sections due to the space between the objects and the original box boundaries. This is usually not what you want. The way around this is to use the box handles. These handles are on all four corners and each side of the box. After the initial box creation, the handles can be dragged so that the box has no spaces in it. This lets the mirroring process create a contiguous, connected copy.

Third step: Click on the Mirror button, . This invokes the Mirror dialog box. It look likes this:

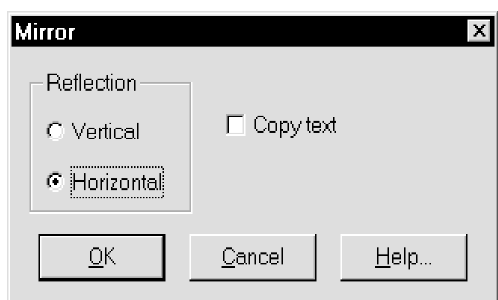


Figure 2-7 The Mirror dialog box

This dialog box lets you choose a vertical or horizontal reflection and whether or not you want to copy the box's grid text. A vertical reflection produces a copy vertically below the original region. A horizontal reflection produces a copy horizontally to the right of the original region. All attributes are copied except Part name, which is incremented. Grid text, if copied, will be incremented if the Text Increment option in the Preferences dialog box is enabled.

Deleting schematic objects

First step: Click on the Select mode button in the Tool bar.

Second step: Select the object (component, text, wire, flag, or graphic object) to be deleted by clicking on it. To select multiple objects for deletion, hold the SHIFT key down while you click on each object. You can also select one or more objects by dragging the mouse to define a rectangular region (box). All objects within the box will be selected when the mouse button is released.

Third step: Press the Delete key to delete the object(s). Deleting in this fashion does not copy the deleted objects to the clipboard, so they will not be available for pasting to a new location. If you want to delete the objects and have them copied to the clipboard for a future paste operation, use the Cut command (CTRL + X).

The Delete key alone removes all selected wire segment entirely from one endpoint to another. CTRL + Delete cuts wire segments that cross the select region box exactly at the box boundaries.

The Undo and Redo commands

The Undo command undoes the effect of an operation. It is activated by pressing CTRL + Z, by clicking on the Undo button , or by selecting the **Undo** item from the **Edit** menu. Undo is available for most edits to text fields, schematics, shapes, and analysis plot objects. Changes available for undoing include editing, deleting, moving, rotating, reflecting, and stepping. Search and replace operations on text documents and the text area of schematics are not reversible. Text edits are not reversible after the text cursor has been moved to another field. If a command is not reversible, you can still reverse its effects by saving the file prior to using the command, and then using the **Revert** option from the **File** menu to load the old version of the file stored on disk.

The Undo command is multistage. It can undo many times and is only limited by available RAM memory.

There is also a multistage Redo command. It is activated by pressing CTRL + Y, by clicking on the Redo button , or by selecting the **Redo** item from the **Edit** menu. It is the complement of Undo. It restores the undone state.

Node number assignment

MC7 automatically assigns numbers to the circuit nodes when an analysis is requested, when the circuit is saved to disk, or when any change is made to the circuit and the **Options / View / Node Numbers**  option is active. Node numbers are displayed on the schematic only when this option is enabled. When you want to plot or print a node's voltage waveform you refer to the waveform as V(node name), where the node name may be either the node number assigned by the program or a text name assigned by you. For MC7 to associate the node names with the node, the lower left corner of the text outline box must be placed directly on the node. Node Snap, if enabled, makes this easier by moving the text to the nearest node within one grid. The Node Snap option is selected from Preferences dialog box (CTRL + SHIFT + P).

The system assigns and displays node numbers according to the following rules:

1. Ground is node number 0, but its node number is never displayed.

2. The other nodes are numbered from 1 to the highest node number.

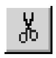
3. When an analog node and a digital node touch, or are connected by a wire, both nodes are assigned a unique number. The program automatically inserts an interface circuit between the two nodes. The interface circuit generates an interface node of the form <num>$ATOD or <num>$DTOA depending upon whether the digital node is an input or an output, respectively. If the interface node is accessible (can be referenced in a plot expression), it will be printed on the schematic. In general, interface nodes between analog parts and digital primitives are accessible unless they occur at a subcircuit interface.

4. Analog node numbers are displayed in a box with rounded corners while digital node numbers are displayed in a box with square corners.

5. Nodes with the same grid text node name are connected together. That is, if two separate nodes each have a piece of identical grid text placed on them, they are considered to be connected and will share the same node name. This provides a convenient way of connecting large numbers of common nodes, without having to use wires to connect them.

The clipboard

The clipboard is a temporary storage area that is used to save schematic or text fragments, as a prelude to pasting them to a new location. It is an invaluable tool for editing schematics or text, and can save considerable time. The clipboard commands are as follows:

	Command	**Shortcut key**	**Effect**
	Clear	DELETE	Deletes the selected object without copying to the clipboard.
	Cut	CTRL + X	Deletes the selected object and copies it to the clipboard.
	Copy	CTRL + C	Copies the selected object to the clipboard.
	Paste	CTRL + V	Pastes the clipboard contents to the last mouse click location.

To copy something to the clipboard, select the object or drag select a region. Press CTRL + C or click on the Copy button. All objects within the rectangular region will be copied to the clipboard.

To paste the clipboard contents, first click at the desired location. Then press CTRL + V or click on the Paste button. The entire contents of the clipboard will be pasted into the schematic or text area at the site of the last mouse click.

Like most other operations, the clipboard actions can be undone by using the Undo command. Simply choose **Undo** from the **Edit** menu, press CTRL + Z, or click on the Undo button to undo the operation.

MC7 maintains two separate clipboards: a text box clipboard and a schematic clipboard. Text copied from a text box will not mix with text copied from the schematic. Text from the text area of a schematic is regarded as schematic material and is stored to or copied from the schematic clipboard. Thus, it is possible to copy text from the text area and paste the same text to the drawing area. Of course, the shuttle command, CTRL + B does this directly. Schematic objects saved to the clipboard are available only for use within MC7 and cannot be pasted to other applications. To export schematic pictures or plots, use the **Copy the Entire Window to a Picture File** command on the **Edit** menu.

Drag copying

Copy and paste works well when you want to duplicate a part of a circuit, but there is another, often easier, option called drag copying. It works as follows:

- Select a circuit object or region to be duplicated.
- Press and hold the CTRL key down.
- Drag on any item in the selected area.

This procedure copies the selected objects and drags the copy along with the mouse, leaving the original objects undisturbed. As with other copying operations like stepping, mirroring, and pasting, part names are always incremented, but grid text is incremented only if the Text Increment option (Preferences) is enabled.

Drag copying can be used on any schematic object, whether it is a component, wire, graphics object, flag, or piece of text.

Drag copying is especially convenient since it replaces the two step process of cut and paste with a simple one step operation with immediately visible results. If you don't like the result, press CTRL + Z to undo it and try again.

After the drag operation has begun, you can release the CTRL key and continue dragging.

Drag copying is often a convenient method for adding new components. If you want to add another 10K resistor, and a 10K already exists in the schematic, drag copying is fast and easy. For example, the standard process consumes four steps:

1. Enter Component mode.
2. Select Resistor.
3. Click in the schematic.
4. Enter the resistor VALUE and (sometimes) MODEL attributes.

The drag method uses one step:

1. CTRL + drag a 10K resistor to the new location.

Drag copying is especially useful when an entire region, like the differential stage of an amplifier, can be copied. Even if some editing of the copied region's objects is required, the entire process is usually simpler, faster, and less error prone.

The Info command

The Info command displays model information about a command statement or a component. It works like this:

First step: Click on the Info mode button in the Tool bar.

Second step: Click on the command or component.

The Info function will provide some information for all components and some command statements. It generally provides information that is not easily visible, such as a subckt listing or a model statement from an external file, a digital source stimulus table, or a Laplace or Function source data table. For simpler components without model statements, the Info function usually invokes the Attribute dialog box.

Text information found in the schematic is highlighted and shown as a part of the schematic. Other information is displayed in the Model editor or the Text editor.

For the .INCLUDE and .LIB command statements, Info displays the contents of the file referenced in the command statements.

If the information is found in the text area, the Info command switches to it and displays the information. To return to the schematic page, press CTRL + G.

The Help command

In addition to the general Help command, there is a component specific Help mode. Here is how it works:

First step: Click on the Help mode button in the Tool bar.

Second step: Click on the component for which you want help.

The Help function provides parameter and syntax information for every component. This is the same information available from the Help system. It is just a little easier to access in the context of working with the Circuit editor.

The digital path commands

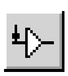
One of the most important properties of digital circuits is path delay. Path delay is how long a signal takes to propagate through each of the many possible paths in a circuit. MC7 provides several related commands for analyzing and displaying these paths. These commands do three things:

- Identify and list the gates involved in the paths.

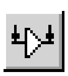
- Show the delay of each path, calculated from the individual delays of the gates in the path list.

- Graphically show selected paths on the schematic by highlighting the gates in the selected path.

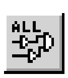
There are three path commands:



1. Point to End Paths command: This shows all paths from the component at the point of a mouse click to the end points. A path ends when a gate does not drive another standard or tri-state gate.



2. Point to Point Paths command: This shows all paths from the point of the first mouse click to the point of the second mouse click.



3. Show All Paths command: This shows all paths in the circuit starting on STIM, FSTIM, or flip-flop outputs and ending when a gate in the path does not drive another standard or tri-state gate.

The procedure to use these commands is as follows:

First step: Select one of the three path commands from the **Options** menu.

Second step: For the Point to End command, click on the body of a gate or stimulus component. For the Point to Point command, click on the starting component and on the ending component.

At this point, MC7 shows a list of paths appropriate to the chosen command in the Path dialog box. It typically looks like this:

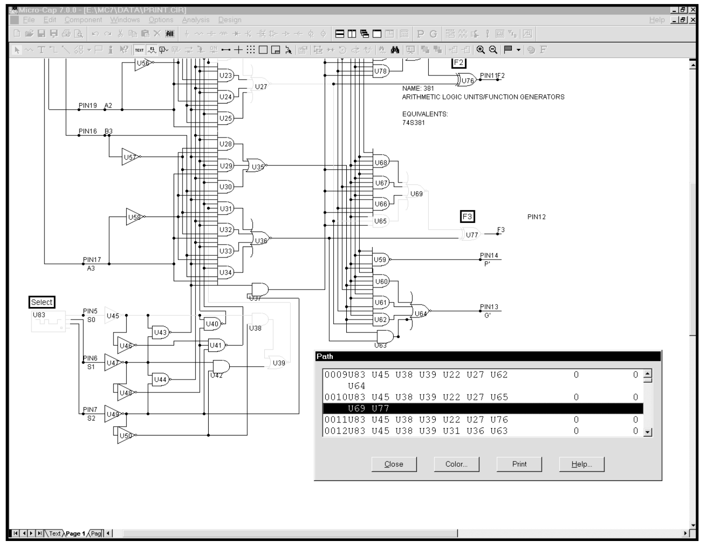


Figure 2-8 The Path dialog box

Scroll through the list and click on a path and MC7 will redraw the schematic and highlight the path. You can use the DOWN ARROW key to trace each path in the list from top to bottom.

For each path, the path list shows the name of each gate in the path, and the total delay through the path for the low to high and high to low signal transitions at the path start. To calculate the LH delay, the program assumes a low to high transition at the path start. It then traverses the path, summing the delays using the actual signal transitions at each gate, the gate's timing model, and the gate's MNTYMXDLY value. To calculate the HL transition, the program assumes a high to low transition at the path start and performs a similar analysis.

A path ends when any gate in the path does not drive a standard gate or a tristate gate. Path commands handle combinatorial gates only. Subcircuits are not expanded.

Navigating schematics

Navigating means being able to quickly display the part of the schematic you are interested in. There are several methods to accomplish this:

Schematic scrolling: Scroll the schematic using the vertical or horizontal scroll bars. This is the conventional method. It is slow but sure.

Scaling: Use the Zoom-Out  or Zoom-In  buttons in the Tool bar to resize the schematic and get your bearing.

Panning: Pan the schematic. Panning means to move the view to a different part of the circuit. In keyboard panning you use CTRL + <CURSOR KEY> to move the view in the direction of the cursor key arrow. In mouse panning you click and hold the right mouse button, while dragging the mouse. The effect is like sliding a piece of paper across a desktop.

Centering: Use the SHIFT + right click method to re-center the view. While holding down the Shift button, click the right mouse button at the point you want centered in the window. Clicking toggles the scale between 1:1 and 1:4 and centers the schematic at the mouse position.

Flagging: Place flags at locations you expect to revisit. Then select a flag from the Select Flag dialog box which is accessed from the Tool bar  or from the **Go To Flag** item on the **Edit** menu.

Page scrolling: Use the Page scroll bar if the desired area is on another page. You can also use CTRL + PGUP and CTRL + PGDN to navigate the pages.

Although users are encouraged to adopt the method that works best for them, each method tends to work best in different circumstances. For very small view changes, use scrolling. For medium changes, use panning. For large changes in medium size schematics, use centering. For very large changes in very large schematics, use flags.

Global Settings

These numeric values and options control device model and circuit analysis options. Some of these definitions can be found in the individual device models:

ABSTOL: Absolute current tolerance. This venerable SPICE parameter specifies the absolute tolerance to be added to the relative current tolerance. Together, their sum must exceed the difference in successive solution values for each current in the circuit to achieve convergence at a particular solution point. Increasing ABSTOL often helps to converge high current devices.

CHGTOL: Absolute charge tolerance. This parameter is like ABSTOL but applies only to charge variables. Adjusting CHGTOL is occasionally required to converge MOSFET devices.

DEFAD: MOSFET default drain area.

DEFAS: MOSFET default source area.

DEFL: MOSFET default channel length.

DEFW: MOSFET default channel width.

DIGDRVF: Minimum drive (forcing) resistance for digital IO models.

DIGDRVZ: Maximum drive (high impedance) resistance for digital models.

DIGERRDEFAULT: Default maximum error message limit for individual digital constraint devices.

DIGERRLIMIT: Default maximum error message limit for all constraint devices during each analysis run.

DIGFREQ: The minimum digital time step is 1 / DIGFREQ.

DIGINITSTATE: Initial state of flip-flops and latches: 0=clear, 1=set, 2=X.

DIGIOLVL: Default digital IO level, 1 to 4. Specifies one of four DtoA and AtoD interface circuits.

DIGMNTYMX: Specifies the default digital delay: 1=Min, 2=Typical, 3=Max, 4=Min/Max.

DIGMNTYSCALE: Specifies scale factor used to calculate unspecified minimum delays from specified typical delays.

DIGOVRDRV: Minimum ratio of drive resistances for one gate to overdrive another driving the same node.

DIGTYMXSCALE: Specifies scale factor used to calculate unspecified maximum delays from specified typical delays.

GMIN: Specifies the minimum branch conductance.

ITL1: Operating point iteration limit before supply relaxation is attempted.

ITL2: DC transfer curve iteration limit for each point of the DC sweep.

ITL4: Transient analysis iteration limit for each time point.

PERFORM_M: This is the number of data points on each side of a data point that must satisfy a performance function search criteria before the data point is accepted. It is used to minimize the effect of noisy data.

PIVREL: Minimum relative value required of a pivot in the matrix solver.

PIVTOL: Minimum absolute value required of a pivot in the matrix solver.

RELTOL: This parameter sets the relative voltage and current tolerance. Successive solution values for each iterated voltage or current variable in the circuit must be less than the sum of the relative and absolute tolerances to achieve convergence at a solution point. Increasing or decreasing RELTOL is occasionally required to converge difficult circuits.

RMIN: This is the minimum absolute value of the resistance of a resistor or an active device lead resistance, (e.g. BJT RB, RE, and RC).

SD: The number of standard deviations in the tolerance band.

TNOM: Default model parameter measurement and analysis temperature.

TRTOL: This is the amount by which the standard LTE formulas are presumed to overestimate the true error.

VNTOL: Absolute voltage tolerance. This parameter specifies the absolute voltage tolerance to be added to the relative voltage tolerance. Together, their sum must exceed the difference in successive solution values for each iterated voltage in the circuit to achieve convergence at a particular solution point. Increasing VNTOL is useful in converging high voltage devices.

WIDTH: Controls input and output column width. Must be either 80 or 132.

NOOUTMSG: Disables output error messages.

PRIVATEANALOG: If enabled, all analog devices have private model libraries. A device can have a private copy of its model library or a public copy. If the copy is private, alterations made to the model values by stepping, optimization, or Monte Carlo features affect only the one device. If the copy is public, it will be shared by all analog parts with the same model name and changes made to the model values by the stepping or Monte Carlo features affect all devices which share the copy. The presence of a DEV tolerance in a model statement forces a private copy, regardless of the state of the PRIVATEANALOG flag. The default value for this option is enabled.

PRIVATEDIGITAL: If enabled, all digital devices have private model libraries. A device can have a private copy of its model library or a public copy. If the copy is private, alterations made to the model values by stepping, optimization, or Monte Carlo features affect only the one device. If the copy is public, it will be shared by all digital parts with the same model name and changes made to the model values by the stepping or Monte Carlo features affect all devices which share the copy. The presence of a DEV tolerance in a model statement forces a private copy, regardless of the state of the PRIVATEDIGITAL flag. The default value for this option is disabled.

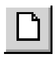
TRYTOCOMPACT: If enabled, this flag causes MC7 to compact the past history of lossy transmission line input voltages and currents.

GEAR: If enabled, selects Gear integration.

TRAPEZOIDAL: If enabled, selects trapezoidal integration.

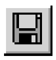
The File menu

The **File** menu provides commands for the management of schematic circuit files, SPICE circuit files, document text circuit files, and model library files.



• **New: (CTRL + N)** This command invokes the New dialog box to create a new file.

• **Open: (CTRL + O)** This command invokes the Open dialog box to load an existing file from the disk.



• **Save: (CTRL + S)** This command saves the active window file to disk using the name and path shown in the title bar.

• **Save As:** This command saves the active window file to disk after prompting the user for a new file name and (optionally) a new path.

• **Paths:** This lets you specify one or more default paths for DATA (circuits), LIBRARY (model libraries), and PICTURE (wmf and bmp picture files). If more than one path is specified, they must be separated by a semicolon (;). MC7 first looks locally in the circuit for model data. It looks in the LIBRARY paths next, searching multiple paths in left to right order.

• **Translate:**

  • **Binary Library to SPICE Text File:** This command translates a binary MC7 library parameter file, FILE.LBR, into a text file, FILE.LIB, containing model statements.

  • **SPICE Text File to Binary Library:** This command translates a SPICE text file, FILE.LIB, containing model statements, into a binary MC7 library parameter file, FILE.LBR.

  • **Schematic to SPICE Text File:** This command creates a SPICE netlist from the active schematic. Any or all analyses may be specified and several types of SPICE formats may be specified.

  • **Schematic to Printed Circuit Board:** These commands create netlist files to be used as input to the Protel, Accel, OrCad, or PADS.

  • **MC7 Schematic to MC5 Version 1 Schematic File:** This command saves the circuit file to disk under the older MC5 Version 1.0 format

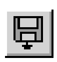
• **MC7 Schematic to MC5 Version 2 Schematic File:** This command saves the circuit file to disk under the older MC5 Version 2.0 format.

• **MC7 Schematic to MC6 Schematic File:** This command saves the circuit file to disk in the MC6 format, creating a file that MC6 can read.

• **Bill of Materials:** This command creates a bill of materials for the schematic listing part name, type, value, quantity, cost, and power budget.

• **Load MC File:** This command loads a circuit numeric output file and scans it for references to Monte Carlo performance function error reports. It then creates the circuits which caused errors during the Monte Carlo run, showing each circuit with parameter values which caused failure. Failure means not meeting a performance function limit (like rise time or delay) specified in the original Monte Carlo run.

• **Revert:** This command restores the file in the active window to the one currently on disk.



• **Close: (CTRL+F4)** This command closes the active file. This means it removes it from the MC7 memory but not from the disk. It will optionally ask if you want to save any changes on disk.

• **Print Preview:** This option previews what the printed schematic, analysis plot, 3D plot, performance plot, or Monte Carlo plot will look like with current print options. It also lets you select, move, and resize the schematic and plot.

• **Print: (CTRL+P)** This command prints a copy of the document shown in Print Preview in accordance with the instructions in Print Setup.

• **Print Setup:** This option changes the printer settings and paper choices. Its format will vary with different printers, but usually it lets you specify a particular printer to use and the paper size, source, and orientation.

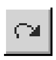
• **Recent Files:** This is a list of the most recently used files. You can reload or display any one of them by clicking on its file name. The number of files displayed is set by the **File List** value in the **Common Options** section of the **Preferences** dialog box.

• **Exit: (ALT+F4)** This exits Micro-Cap 7.

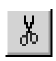
The Edit menu

This menu provides the following options. Equivalent tool bar buttons appear next to the commands below.

• **Undo: (CTRL + Z)** MC7 has a multistage undo. It can undo the last N operations that change a circuit file. N is limited only by RAM memory and is usually greater than 20. Undo can also restore the last state of a text field, if the text cursor is still in the field.



• **Redo: (CTRL + Y)** Redo operates in the opposite direction of Undo. It restores the prior circuit state. Like Undo, it can redo the last N operations that change a circuit file.



• **Cut: (CTRL + X)** This command deletes the selected objects and copies them to the clipboard. Objects include text field text and schematic objects.

• **Copy: (CTRL + C)** This command copies selected objects to the clipboard. Objects include text from text fields and schematic objects.

• **Paste: (CTRL + V)** This command copies the contents of the clipboard starting at the current cursor position. If a group of characters in a text field is currently selected at the time of the Paste operation, they are replaced with the text in the clipboard. If the front window is a schematic, the paste is done from the last point in the schematic where the mouse was clicked.

• **Clear: (DELETE)** This command deletes the selected items without copying them to the clipboard. This command deletes selected wires from their end points.

• **Clear Cut Wire: (CTRL + DELETE)** This command deletes selected wires by cutting them exactly at the sides of the select box.

• **Select All: (CTRL + A)** This command selects all objects in the current window or all text in the current text field or document.

• **Copy to Clipboard:** This command copies all or part of the visible portion of the front window in graphics format to the clipboard. It copies all or a part of the window contents in BMP, WMF, or EMF format to the clipboard. The picture in the clipboard may then be imported via pasting to other programs, such as Word or Excel.

You can't paste a *clipboard picture* to MC7, though you can import a picture *file* to MC7 using the **Picture** item from the **Mode** part of the **Options** menu. After selecting the Picture mode, drag or click in the schematic and MC7 prompts for the picture file name.

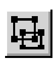
• **Copy the Entire Window to a Picture File:** This command creates a picture file in BMP, WMF, or EMF format from the front window. The file can then be imported into MC7 or other Windows applications.

• **Add Page:** This command adds a new page to the schematic.

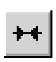
• **Delete Page:** This command deletes one or more schematic pages.

• **Refresh Models:** This command localizes circuit model information by copying subckts and model statements from the libraries to the circuit. It is a handy way to embed model information in the file prior to sending a copy of the circuit to a colleague who may not have your models. It can also serve as a refresh step, restoring model information from the master libraries in case you have edited the models and want to restore them.

• **Box:** These commands affect objects enclosed in the selected region.



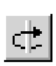
• **Step Box:** This command steps all objects within the selected region vertically or horizontally (or both) a specified number of times.



• **Mirror Box:** This command creates a horizontal or vertical mirror image of the objects enclosed in a selected region.

• **Rotate: (CTRL + R)** This command performs a counterclockwise rotation of the objects in a selected region.

• **Flip X:** This command flips the objects in a selected region about the X-axis. The X-axis is defined as the horizontal line that bisects the selected region.



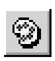
• **Flip Y:** This command flips the objects in a selected region about a Y-axis. The Y-axis is the vertical line that bisects the selected region.

• **Make Macro: (CTRL + M)** This command makes a macro circuit from the circuitry inside of the current box region. It gathers the circuitry into a new circuit, labels the pin names, saves the circuit to disk under a name you choose, records an entry in the macro components library file, MACRO.CMP, and replaces the circuitry within the box region with an adjustable macro symbol representing the circuitry. After the macro command, the circuit will simulate just as before, but the schematic will have less clutter.

• **Change:** This option lets you change several attribute features:

• **Properties:** (F10) This accesses the Properties dialog box for the circuit window. From here you can change the color of circuit features such as wire or component color. Changes affect all objects rather than just selected objects.

Attributes: This lets you change the display status of the five main attributes of *all* components in the circuit, *en masse*.



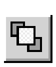
Color: This lets you change the attribute color of *selected* components.

Font: This lets you change the attribute font of *selected* components.

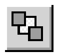
Rename Components: This command renames all components using standard naming conventions. It also reorders the components to make the node numbers flow from left to right and top to bottom. It updates the component names in analysis plot expressions, so that R(RRX) changes to R(R5) if the part name is changed from RRX to R5. It does not update node numbers in analysis plot expressions, however, so expressions like V(10) will not be changed to V(2) if node 10 gets changed to node 2.

Rename Defines: This command renames all .defined symbolic names where conflicts exist between the symbol name and predefined names.

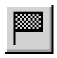
Reset Node Positions: The relative printing position of the node number, node voltage, current, power, and condition can be changed by dragging its text. This command provides a way to restore their original default positions.



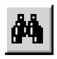
• **Bring to Front:** A mouse click on a stack of overlapping objects selects the front object in the stack. If the front object isn't the one you want, a way is needed to select other objects from the stack. This command makes the selected object the front object.



• **Send to Back:** This command makes the selected object the back object.



• **Go To Flag:** This command invokes the Go To Flag dialog box. It lets you reposition the display on a flag. Select a flag and click on the OK button and the program redraws the schematic, with the selected flag at the center of the display. This item is enabled only when there are flags in the schematic.



• **Find: (CTRL + F)** This command invokes the Find dialog box, which is used to search the circuit window for a variety of objects, including part names, node names, attributes text, component type, and grid text.

It looks like this:

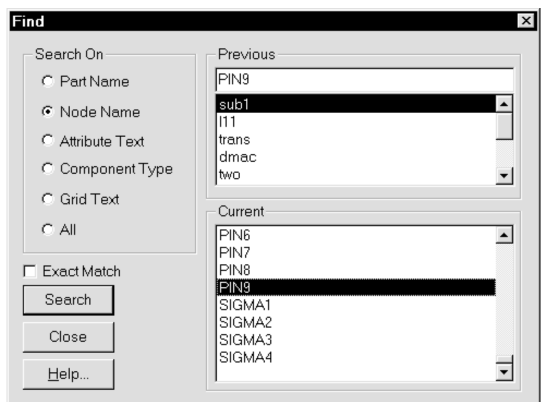


Figure 2-9 The Find dialog box

• **Repeat Last Find: (F3)** This command repeats the search and finds the next object in the circuit that matches the search criteria.

• **Replace:** This option replaces text in a text document or in the text area of a schematic. The replace function for schematic attributes is done from the Attribute dialog box using the Change button.

The Component menu

This hierarchical menu shows the contents of the Component library. The library was created and can be edited by the Component editor, although most users will not need to do much editing. One part of the menu looks like this:

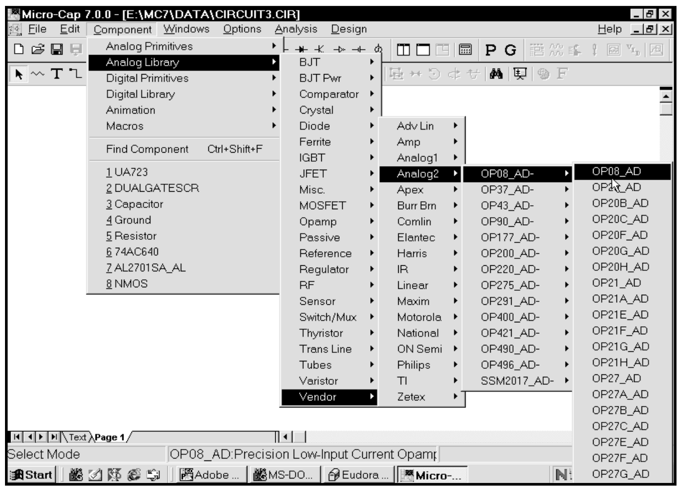


Figure 2-10 The Component menu

The Component menu lets you select a component for placing in the schematic. It is designed to provide easy access to any of the more than 15,000 parts in the library. Access to more common generic parts is best done from a Component palette or a Component button. The menu contains these sections:

Analog Primitives: This section has generic analog components for which the user supplies the values or model statements that define their electrical behavior.

Analog Library: This section has models for commercial analog components. The values or model statements that define their electrical behavior are provided in the Model library.

Digital Primitives: This section has generic digital components for which the user supplies the values or model statements that define their electrical behavior.

Digital Library: This section has models for commercial digital components. The values or model statements that define their electrical behavior are provided in the Model library.

Animation: This section contains objects which change their display or respond to user clicks during a simulation. The objects include a seven-segment display, an LED, and a switch which toggles its state between 1 and 0 when you click on it. These objects are designed to operate in transient analysis in conjunction with the Animate mode. In this mode, a single analysis time step is taken and the states and animate devices updated on the schematic. MC7 then waits for either a key press or a specified time delay before taking the next time step. The purpose is to slow the simulation down to show individual node voltage or state changes.

Filters: This section is present if filter macros have been created by the filter design feature, accessible from the **Design** menu.

Macros: This section is present if macros have been created by the Make Macro command, accessible from **Edit / Box / Make Macros**.

Find Component: (SHIFT + CTRL + F) The Find Component command lets you select a part by its name, shape, definition, or memo field. It look like this: To select a component for placement, use the cursor keys or click the mouse on

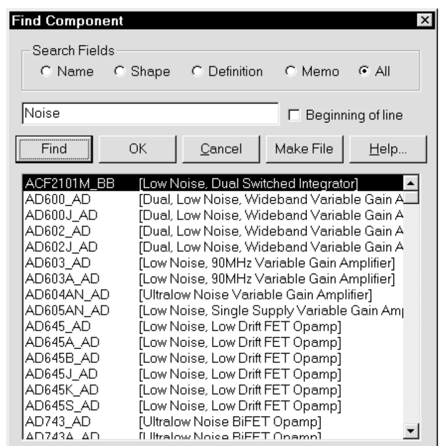


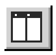
Figure 2-11 The Find Component dialog box

one of the parts on the Component menu items. When you select a component, MC7 automatically changes the tool mode to Component. To actually place the component, click the left mouse button in the schematic, or click and drag the component to where you want it to be placed. To rotate the component, click the right button before the left button is released.

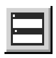
Last Used: This part of the menu shows recently selected parts.

The Windows menu

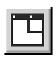
• **Cascade: (SHIFT + F5)** This command cascades the open circuit windows in an overlapping manner.



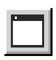
• **Tile Vertical: (SHIFT + F4)** This command vertically tiles the open circuit windows in a non-overlapping manner.



• **Tile Horizontal:** This command horizontally tiles the open circuit windows in a non-overlapping manner.



• **Overlap:** This command overlaps the circuit window and the analysis plot window. If the Plot on Top option is enabled, the circuit is maximized and a 1/4 size analysis plot is placed on top, otherwise the analysis plot is maximized and a 1/4 size circuit is placed on top. This option is available only when a normal or Probe-mode transient, AC, or DC analysis is active.



• **Maximize:** This command maximizes a selected circuit window or circuit window icon.

• **Arrange Icons:** This command neatly arranges any minimized circuit window icons.

• **Zoom-In: (CTRL + numeric keypad +)** If the active circuit window is a schematic, this command magnifies the schematic. This command affects only the display, not the printed size. If the active circuit window is a text window, this command draws the window using the next largest font size.

• **Zoom-Out: (CTRL + numeric keypad -)** If the active circuit window is a schematic, this command shrinks it. The command affects only the display, not the printed size. If the active circuit window is a text window, this command redraws the window using the next smallest font size.

• **Toggle Drawing/Text: (CTRL + G)** Every schematic has a drawing area and a text area. The drawing area contains the schematic. The text area is where local model, subckt, stimulus, and source table statements are stored. This command toggles the window display between the drawing and text areas.

• **Split Horizontal:** This horizontally splits the front schematic window into its drawing and text areas for simultaneous viewing.

• **Split Vertical:** This command vertically splits the front schematic window into its drawing and text areas for simultaneous viewing.

• **Remove Splits:** This command removes any split panes in the window.

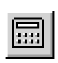
• **Component Editor:** This command accesses the Component editor, which is used to define part names and their electrical definition.

• **Shape Editor:** This command accesses the Shape editor, which is used to create and maintain graphical shapes for the components.

• **Package Editor: This** accesses the Package editor, which manages the information needed to generate netlists to popular PCB programs.

• **Model Program:** This command accesses the Model program, which is used to extract model parameters from data sheet information

• **Check Model Library Parameters:** This checks the entire model library against the upper and lower limits specified at **Options / Model Parameter Limts Editor**.



• **Calculator:** This invokes the Calculator window. It lets you type in expressions and it calculates the result. You can use circuit variables if you are in an analysis mode. It can even do symbolic derivatives. Here are some sample expressions:

(1+2*j)*(1+2*j)....complex product returns -3 + 4*j

VBE(Q1)*IB(Q1) + VCE(Q1)*IC(Q1)....power in a BJT

SERIES(N,0,25,1/FACT(N))...Evaluates the first 26 terms of 1/FACT(N).

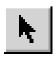
The derivative of I(L1)*V(L1) W.R.T. I(L1) returns V(L1). This example works only in analysis mode only since I(L1) and V(L1) are defined only in analysis mode. Expressions using generic symbols work in schematic mode. For example, the derivative of X^X returns X*X^(X-1)+X^X*LN(X).

• **Check Model Library Parameters:** This checks Model library parameters against the limits set in the Model Parameter Limits Editor.

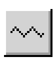
• **Files in Memory:** This group lists the open files in memory. If multiple circuits have been loaded, you can select one for display from this list.

The Options menu

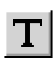
- **Main Tool Bar: (CTRL + 0)** This toggles the main tool bar on and off.
- **Default Main Tool Bar:** This option restores the default main tool bar.
- **Status Bar:** This option toggles the Status bar on and off.
- **Mode:** This option accesses the Mode submenu. It contains these items:



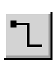
  - **Select: (CTRL + E)** This mode selects objects for editing. To change attribute values like part name or model name, to edit a fragment of grid text, or to select a schematic region for moving or deleting you must be in the Select mode. Normally, you invoke this mode by clicking on the Select button, pressing the SPACEBAR, or by typing CTRL + E, but you may also invoke it by selecting this menu item.



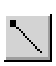
  - **Component: (CTRL + D)** This mode lets you add a component to the schematic. Invoke this mode by clicking on the Component button, typing CTRL + D, or selecting this menu item.



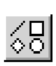
  - **Text: (CTRL + T)** This mode lets you add grid text to the schematic. Grid text can be used for node names and model statements (which can also be placed in the text area). Normally, you invoke this mode by clicking on the Text button, or by typing CTRL + T, but you may also invoke it by selecting this menu item.



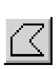
  - **Wire: (CTRL + W)** This mode adds orthogonal wires to the schematic. Wires are used to connect components together. Normally, you invoke this mode by clicking on the Wire button, or by typing CTRL + W, but you may also invoke it by selecting this menu item.



  - **WireD:** This mode is used to draw diagonal wires in the schematic.

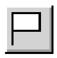


  - **Line, Rectangle, Diamond, Ellipse, Arc, Pie:** These modes let you draw graphic objects on the schematic or plot. You can select a mode from here, or click on the Graphics button in the Tool bar, then select the desired object from the menu that pops up.

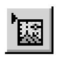


  - **Polygon:** This command lets you place a polygon in an analysis plot. The polygon is intended for use in defining valid specification ranges.

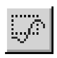
After rough drawing a polygon object, double-clicking on it lets you type in more exact values. The key words MIN and MAX place a polygon vertex coordinate flush with the plot sides.



• **Flag:** This mode is used to place flags on the schematic. Flags are used to mark locations for quick navigation. Normally, you invoke this mode by clicking on the Flag button, but you may also use this menu item.

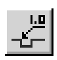


• **Picture:** This mode lets you place picture files in the schematic.



• **Scale: (F7)** This command puts the analysis plot in Scale mode.

• **Cursor: (F8)** This command puts the analysis plot in Cursor mode.



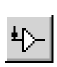
• **Point Tag:** This mode lets you place a point tag on a plot. A point tag labels the X and Y values of a single data point on a waveform.

• **Horizontal Tag:** This mode lets you place a horizontal tag between two data points. This tag measures and labels the horizontal difference between two data points on one or two waveforms, yielding pulse width or time delay if the X expression is Time.

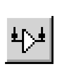
• **Vertical Tag:** This mode lets you place a vertical tag between two data points. This tag measures and labels the vertical difference between two data points on one or two waveforms.

• **Help: (CTRL + H)** This command invokes the component help mode. This mode lets you click the mouse on a schematic component to see its parameter and model syntax.

• **Info: (CTRL + I)** This invokes the Info mode. In this mode, clicking on a component displays its model parameters, model statement, subcircuit description, or command statement, depending upon the component.



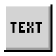
• **Point to End Paths:** This command invokes the Point to End Paths mode. In this mode, clicking on a digital component displays all paths from the component clicked to all possible end points. End points include flip-flops and gates which drive no other gates.



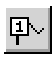
• **Point to Point Paths:** This command invokes the Point to Point Paths mode. In this mode, clicking on a digital component displays all paths from the first component clicked to the second component clicked.

- **View:** These options determine what will be drawn on the schematic.

  - **Attribute Text:** If checked, this shows component attribute text.

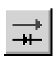


  - **Grid Text:** This option shows grid text. Grid text is any text created with the Text tool.

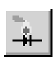


  - **Node Numbers:** This option shows the node numbers assigned by the program to each node. Analog nodes have rounded rectangles and digital nodes have normal angular rectangles. Grid text placed on a node may serve as an alias for the node number.

  - **Node Voltages / States:** This option displays the last AC, DC, or transient analysis time-domain node voltages and digital states. If these are the result of an operating point only run, the display shows the DC operating point values.

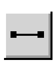


  - **Current:** This option displays the last AC, DC, or transient analysis time-domain currents.

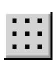


  - **Power:** This option displays the last AC, DC, or transient analysis time-domain power.

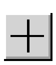
  - **Condition:** This option displays the last AC, DC, or transient analysis time-domain conditions. Conditions define the operating state for devices such as ON, OFF, LIN, SAT, and HOT for BJTs.



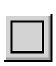
  - **Pin Connections:** This option displays a dot at the location of each pin. This helps you see and check the connection points between parts.



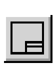
  - **Grid:** This option displays the schematic grid.



  - **Cross-hair Cursor:** This option adds a cross-hair cursor.



  - **Border:** This option adds a border to the schematic.



  - **Title:** This option adds a title block to the schematic.

- **Show All Paths:** This command shows all digital paths and their delays. Selected paths are highlighted on the schematic. It is an immediate command as opposed to the Point to End Paths and Point to Point Paths modes, which require mouse clicks to specify path endpoints before a path list appears.

• **Preferences: (CTRL + SHIFT +P)** This accesses the Preferences dialog box, where many user preferences are selected. These include:

- **Common Options:** This accesses a list of operational choices:

  - **Select Mode:** This option causes the schematic mode to revert to Select mode after any other mode is completed. For example, to place a component you must be in the Component mode. After placing the component, the mode normally stays in Component mode. If this option is selected, the schematic mode reverts to Select immediately after a component is placed. The same thing would happen when drawing wires, placing text, querying a component, or any other mode-based action.

  - **Sound:** This controls the use of sound for warnings and to indicate the end of a simulation run.

  - **Lock Tool Bar:** This option disables movement and locks the tool bar into its current position.

  - **Print Background:** If enabled, this option adds the user-selected background color to the printouts. It is usually disabled, since most background colors do not print well.

  - **Time Stamp:** This option adds a time stamp to all numeric output files and plots.

  - **Date Stamp:** This option adds a date stamp to all numeric output files and plots.

  - **File List Size:** This value sets the number of file names to include in the Recent Files section of the File menu.

  - **Warning Time:** This sets the duration of warning messages.

- **Analysis Options:** This accesses a list of options related to analysis:

  - **Floating Nodes Check:** This provides a warning for floating nodes (those for which only one component pin is attached).

  - **DC Path to Ground Check:** This checks for the presence of a DC path to ground. It should normally be on.

- **Convergence Assist:** This option enables the Convergence Assist routine which attempts to optimize selected Global Settings parameters to help a circuit converge. It may modify RELTOL, ABSTOL, VNTOL, ITL4, ITL2, METHOD, GMIN, and others to achieve convergence. If it succeeds, it adds a .OPTIONS statement to the circuit with the modified parameters, so that subsequent runs converge more readily.

- **Add DC Path to Ground:** This option automatically adds a 1/GMIN resistor to any path where there is no path to ground.

- **Plot on Top:** If enabled, this option places the analysis plot on top of the schematic if they are overlaid. If unchecked, the schematic floats above the plot.

- **Select Curve Color:** If enabled, this option colors the selected curve branch in the Select Color Primary. It is nearly always on, except when many hundreds of branches would slow down the redraw required by the select color.

- **Inertial Cancellation:** If enabled, this option causes the logic simulator to employ inertial cancellation, which cancels logic pulses whose durations are shorter than the device delay. If this option is disabled, the simulator does not cancel short pulses.

- **Analysis Progress Bar:** If enabled, this option displays a progress bar during a simulation run in the Status Bar area, if the Status Bar feature is enabled on the Options menu.

- **Gmin Stepping:** If enabled, this option tries Gmin stepping after both normal methods and source stepping have failed to achieve DC convergence.

- **Auto Scale Grids:** This sets the number of grids to use when auto scaling a plot.

- **Circuit Options:** This accesses a list of options related to circuits:

- **Text Increment:** This controls whether non-command grid text is incremented during a paste, step, drag copy, or mirror operation. Incrementing means that the ending numerals in the text are increased by one. If no numerals are present, a '1' is added to the text.

• **Node Snap:** This flag compels components, wires, and text to originate on a node if a node's pin connection dot is within one grid of the object being placed or moved.

• **Auto Show Model:** This mode causes model statements added to the schematic text area to automatically split the circuit window and show the newly added model statement.

• **Component Cursor:** If enabled, this option replaces the mouse arrow with the shape of the currently selected component whenever Component mode is active. This has the advantage of showing the currently selected component and its size.

• **Rubberbanding:** If enabled, this option causes drag operations to extend the circuit wires to maintain node connections. When disabled, drag operations sever selected wires at their endpoints and drag them without changing their shape or length.
SHIFT + CTRL + R toggles the feature on and off to allow the rapid mode switches, necessary when dressing up a schematic.

• **Show Slider:** If enabled, this option places a slider control on the schematic adjacent to batteries and resistors during Dynamic DC Analysis. This lets the user change the battery or resistor values by dragging the slider with the mouse. With or without the slider, the values can be changed by regular amounts with each press of an UP ARROW or DOWN ARROW cursor key.

• **Nodes Recalculation Threshold:** If the Show Node Numbers option is enabled, nodes are recalculated and displayed whenever any schematic editing is done. For large schematics this can be time consuming. This value sets an upper limit for node count beyond which automatic node recalculation will be ignored, even if the Show Node Numbers option is enabled.

• **Block Select Display Mode:** This option enables the block select mode, which shows the background of selected objects in the Block Select color. This makes the selected objects easier to spot, especially when there is only one object. If disabled, the selected object is drawn in the standard foreground Select color.

- **Automatically Add Opamp Power Supplies:** This option automatically adds and connects the VCC and VEE power supplies for level 3 opamp primitives. It places the batteries on a schematic page called Power Supplies. Note that it will not work for vendor-generated opamp subckt models.

• **Color Palettes:** This option lets you define your own palettes. Click on any of the color squares to invoke a color editor that lets you customize the hue, saturation, and luminance of the selected palette color.

• **Format:** This option controls the numeric format to be used when displaying analysis plot tag values, DC operating point values in the numeric output text page, schematic node voltages, currents, and powers, schematic path delays, and formula text.

• **Status Bar:** This lets you change the Status bar text attributes.

• **Main Tool Bar:** This panel lets you show or hide buttons and tool bars that normally reside in the main tool bar area.

• **Component Palettes**: This lets you name the nine component palettes and control their display in the Main tool bar. You can also toggle their display on and off during schematic use with CTRL + palette number.

• **Auto Save:** This accesses the Auto Save dialog box from which you can enable automatic saving of circuit files to disk every time you run an analysis or on a specific time schedule.

• **Warnings:** This accesses the Warnings dialog box from which you can enable specific warning messages including:

- **File Warning:** This provides a warning when closing an edited file whose changes have not yet been saved.

- **Quit Warning:** This asks if you really want to quit.

- **Opamp Power Supplies:** This alerts the user when MC7 adds VCC and VEE power supplies.

- **Add DC path to ground:** This warns when adding resistors to avoid a DC path to ground.

- **Default Properties For New Circuits:** This dialog box controls the properties of new circuits. It includes:

    - **Schematics:** This controls three types of features for schematics:

        - **Color/Font:** This provides control of text font and color for various schematic features such as component color, attribute color and font, and background color.

        - **Title Block:** This panel lets you specify the existence and content of the title block.

        - **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.

    - **SPICE Files:** This provides two types of features for SPICE text files.

        - **Color/Font:** This provides control of text font and color for SPICE text files.

        - **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.

    - **Analysis Plots:** This provides control for several analysis plot features:

        - **Scales and Formats:** This panel lets you specify the numeric format for the Numeric Output, plot scales, Cursor mode tables, Optimizer values, and Watch values. It allows selection of Same Y Scales for separate plot groups and lets you select the method for measuring slope: normal, dB/octave, and dB/decade. The latter two are more appropriate to certain AC measurements.

        - **Colors, Fonts, and Lines:** This provides control of text font and color for various plot features such as general scale and title text, window and graph background colors, and individual curve color, thickness, and pattern.

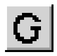
        - **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.

- **3D Plots:** This provides control for several 3D plot features:
    - **Color:** This provides color control of 3D plot features such as general scale and title text, window and graph background colors, axis colors, patch color, and surface line color.
    - **Font:** This provides font control for all text in the 3D plot.
    - **Scales and Formats:** This panel lets you select the numeric format for the X, Y, and Z axis scales and the numeric values in the cursor tables. It also controls the slope calculation method.
    - **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.

- **Monte Carlo Histograms:** This provides control for Monte Carlo histogram plot features:
    - **Color:** This provides color control of histogram features such as general scale and title text, window and graph background colors, and bar colors.
    - **Font:** This provides font control for all text in the histogram.
    - **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.

- **Performance Plots:** This panel controls performance plot features:
    - **Scales and Formats:** This panel lets you select the numeric format for plot scales and values shown in the Cursor tables. It also allows selection of same Y scales for separate plot groups.
    - **Colors, Fonts, and Lines:** This provides control of text font and color for various plot features such as general scale and title text, window and graph background colors, and individual plot color, thickness, and pattern.

• **Tool Bar:** This panel lets you select the tool bars and buttons that will appear in the local tool bar area located just below the main tool bar.



• **Global Settings: (CTRL + SHIFT + G)** This accesses the Global Settings dialog box where many simulation control choices are made. These settings are discussed in greater detail in the Global Settings section of the Circuit editor chapter of the Reference Manual.

• **User Definitions:** This option displays the file MCAP.INC. This file, located on the directory with MC7.EXE, stores global definitions for use in all circuits. The contents of the file are automatically included in all circuit files.

• **Model Parameter Limits Editor:** This editor lets you set the minimum and maximum values that the model parameters can have. It also shows the default values, but does not allow you to edit them. A warning message is generated and placed in the numeric output file for parameters that do not fall within the limit values. You can also run a comprehensive check of all models from **Windows / Check Model Library Parameters**.

• **Component Palettes:** This option accesses the Component Palettes. These provide a quick and convenient alternative to the Component menu when choosing components for placement in the schematic. Membership in a palette is specified from the Component editor. The original disks provide several sample palettes containing common analog and digital components. The palettes can be tailored to suit personal preferences. Palette display can be toggled on and off by pressing CTRL + palette number. For example, CTRL + 1 toggles the display of palette 1.

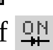
The Analysis menu

The Analysis menu is used to select the type of analysis to run on the circuit in the active window. It provides six basic options:

- **Transient: (ALT + 1)** This option selects transient analysis. This lets you plot time-domain waveforms similar to what you'd see on an oscilloscope.

- **AC: (ALT + 2)** This option selects AC analysis. This lets you plot frequency-domain curves similar to what you'd see on a spectrum analyzer.

- **DC: (ALT + 3)** This option selects DC sweep analysis. This lets you plot DC transfer curves similar to what you'd see on a curve tracer.

- **Dynamic DC: (ALT + 4)** This option selects an analysis mode in which MC7 automatically responds to user edits by finding and displaying the DC solution to the current schematic. You can change battery voltages and resistor values with slider controls or with the cursor keys. You can also make any edit such as adding or removing components, editing parameter values, etc. MC7 responds by calculating the DC solution and will display voltages and states if  is enabled, currents if  is enabled, power dissipation if  is enabled, and device conditions if  is enabled.

- **Transfer Function: (ALT + 5)** This option selects an analysis mode in which the program calculates the small signal DC transfer function. This is a measure of the incremental change in a user-specified output expression, divided by a very small change in a user-specified input source value. The program also calculates the input and output DC resistances.

- **Sensitivity: (ALT + 6)** This option selects an analysis mode in which the program calculates the DC sensitivity of one or more output expressions to one or more input parameter values. The input parameters available for measurement include all those available for component stepping, which is practically all model parameters, all value parameters, and all symbolic parameters. Thus you can create a lot of data with this analysis. A suitably smaller set of parameters is specified in the default set, or you can select only one parameter at a time.

- **Probe Transient: (CTRL + ALT + 1)** This selects transient analysis Probe mode. In Probe mode, the transient analysis is run and the results are stored on disk. When you probe or click part of the schematic, the waveform for the node you clicked on comes up. You can plot all of the variable types, from analog node voltages to digital states. You can even plot expressions involving circuit variables.

- **Probe AC: (CTRL + ALT + 2)** This option selects AC Probe mode.

- **Probe DC: (CTRL + ALT + 3)** This option selects DC Probe mode.

The Design menu

The Design menu accesses the Filter Design functions.

• **Active Filters:** This part of the program designs an active filter from your filter specifications. The filter type can be low pass, high pass, band pass, notch, or delay. The filter response can be Butterworth, Chebyshev, Bessel, elliptic, or inverse-Chebyshev. Polynomials are calculated from the filter specifications, then mapped into one of several different circuit styles, ranging from classical Sallen-Key to Tow-Thomas. Optional plots of the idealized transfer function are available. Filters can be created as circuits or as macro components.

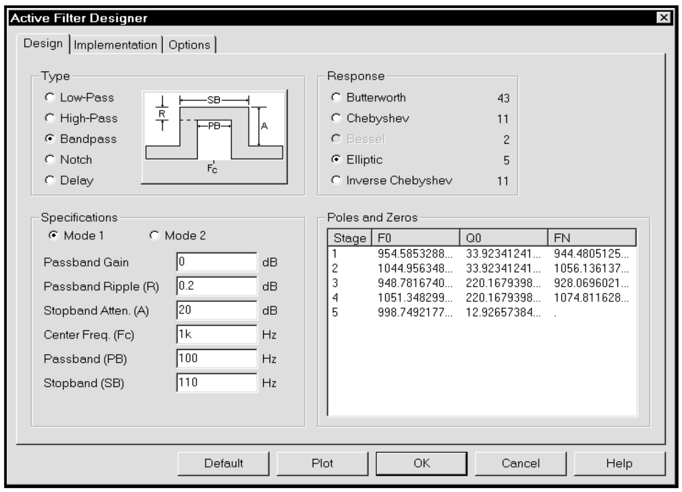


Figure 2-12 The Active Filter Designer

• **Passive Filters:** This part of the program designs a passive filter from your specifications using capacitors and inductors. The filter type can be low pass, high pass, band pass, or notch. The filter response can be Butterworth or Chebyshev. The filter polynomials are calculated and implemented in either standard or dual circuit configuration. Optional plots of the idealized transfer function are available. Filters can be created as circuits or as macro components.

The filter design functions from the Design menu is covered in more detail in Chapter 12.

The Model editor

Model libraries provided with MC7 come in two forms, text and binary.

In *text* form, they are contained in files with the extension LIB, and encode device models as .MODEL, .MACRO, and .SUBCKT statements. Text files can be viewed and edited with any text editor, including the MC7 text editor.

In *binary* form, model libraries are contained in files with the extension LBR, and are implemented as a list of model parameters for the parts. These binary files can be viewed and edited only with the Model editor. The Model editor is invoked when the **File** menu is used to open or create a binary library file.

The Model editor should not be confused with the MODEL program, which is accessed from the Windows menu. MODEL is a separate Windows program which produces optimized analog model parameters from commercial data sheets. MODEL can create model libraries in either the text or binary form.

Once the binary libraries are created, the Model editor can be used to view and edit them. The Model editor display looks like this:

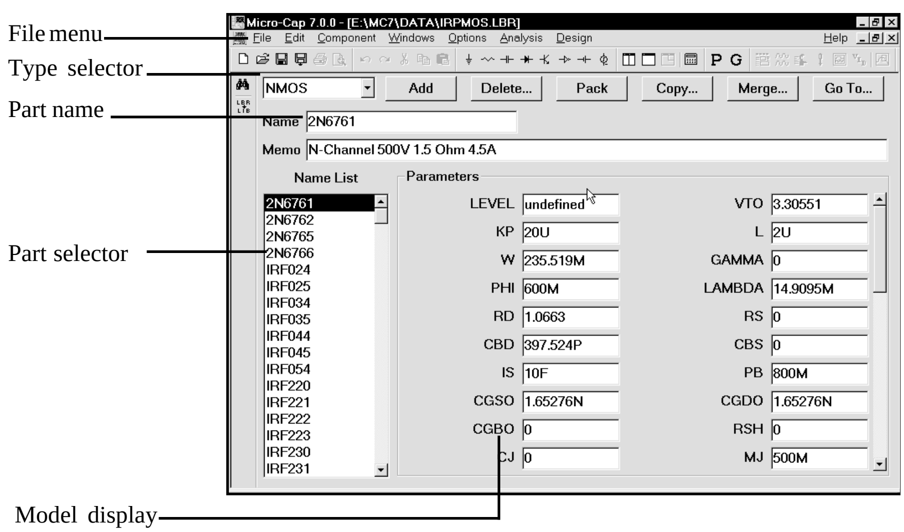


Figure 2-13 The Model editor

The various parts of the editor function as follows:

- **Name:** This field is where the part name is entered. If the part has been imported from the MODEL program, this field is a copy of the T1 text field.

- **Memo:** This is a simple text field that may be used for any descriptive purpose. If the part has been imported from the MODEL program, this field is a copy of the T3 text field.

- **Type selector:** This is used to select which device type to display. Each library may contain a mixture of different device types. Selecting NPN for example, displays all of the NPN bipolar transistors in the file.

- **Part selector:** This selects a part by name for display and possible editing. It provides a window to review the specific model values for the displayed part. As in other windows, the Maximize button may be used to enlarge the window to see more of the model values.

- **Add:** This adds a new part of the current device type to the current library.

- **Delete:** This deletes the displayed part.

- **Pack:** This removes all duplicated and untitled parts and reorders the parts alphanumerically.

- **Copy:** This command copies a part from the displayed library to a target library. If the target library already has a part with the same name, the name of the newly created copy is set to "oldname_copy". The target library may be the current library.

- **Merge:** This command merges a library from disk with the library currently in memory. The merged library is displayed but is not automatically saved to disk.

- **Go To:** This command lets you specify a parameter name, then scrolls the parameter list of the currently displayed part to show the parameter value.

Note that the Find function is available to locate parts by name in the current library file or in one of the library files on disk. Simply click on the Find button  in the Tool bar.

The Help system

The Help system provides information on the product in several ways:

- **Contents (F1):** This section provides help organized by topics.

- **Search for Help On...:** The search feature lets you access information by selecting a topic from an alphabetized list.

- **Product Support:** This option provides technical support contact numbers.

- **Tip of the Day...:** This option provides short tips on product features that are displayed each time MC7 is started.

- **About Micro-Cap:** This option shows the software version number.

- **Statistics:** This displays a list of statistics showing the key ID, MC7 version number and executable date. In an analysis it also shows the setup and run time, the number of analog and digital nodes, the number of data points calculated during the run, a list of the actual parts in the circuit after macro and subcircuit expansion, and a list of the number of iterations and solutions for the run.

- **Key ID:** This displays the security key ID which you sometimes need when upgrading the program or accessing technical support.

- **Demos:** These live demos show the sequence of actions involved in common functions. They include:

    - **General Demo:** This provides a general overview of MC7. It includes each of the demos below and runs continuously until ESC is pressed.

    - **3D Plots Demo:** This illustrates the use of 3D plots, which let you show how circuit waveforms or performance functions vary with two different parameters.

    - **Analog and Digital Demo:** This shows how to mix analog and digital parts in a schematic and their analysis plots.

    - **Analysis Demo:** This shows how to start and control an analysis.

- **Animation Demo:** This illustrates the use of the animation feature, in which animated circuit components like LEDs and seven-segment displays respond to circuit states by turning on and off.

- **Dynamic DC Demo:** This demonstrates the Dynamic DC feature, which automatically calculates the DC operating point of the circuit as it is changed by the user.

- **Filter Demo:** This illustrates the use of the built-in active and passive filter design functions.

- **Fourier Demo:** This illustrates the use of the DSP functions to do Fourier analysis on circuit waveforms.

- **Monte Carlo Analysis Demo:** This illustrates the use of Monte Carlo analysis, which analyzes circuit behavior statistically as circuit parameters are varied around specified distributions.

- **Optimizer Demo:** This illustrates the use of the optimizer function to optimize and fine-tune circuits.

- **Probe Demo:** This illustrates the use of the Probe feature which lets you probe (point and click) the schematic for curves and waveforms.

- **Performance Plots Demo:** This illustrates the use of performance plots, which let you plot important circuit characteristics, like bandwidth or rise time versus model and other parameters.

- **Schematic Demo:** This shows how to create a schematic.

- **Sensitivity Demo:** This illustrates the Sensitivity analysis mode, which calculates the DC sensitivity of the circuit to one or more parameters.

- **Stepping Demo:** This illustrates the use of component parameter stepping, which lets you systematically step or change model and other parameters to see their effect on circuit behavior.

- **Subcircuits Demo:** This demo shows how to create and use new subcircuits, including how to add them to the Component library.

- **Transfer Function Demo:** This illustrates the Transfer Function analysis mode, which calculates the DC transfer function.

Chapter 3 The Shape Editor

What's in this chapter

This chapter describes the Shape editor. This editor is used to build new shapes or change existing shapes. Shapes are used to represent components in a schematic. Each shape is composed of objects such as lines, circles, and rectangles.

Upon exiting and saving changes, the shapes are available for use by the Component editor. That editor maintains the Component library, from which components are selected for use in schematics.

A large library of shapes is supplied with MC7, so most users will not need to use the Shape editor. For those who want to customize their shapes or add new ones, this chapter is for you.

Features new in Micro-Cap 7

- Drag copy command.
- Shape origin icon.
- Font size memory retained.

The Shape editor layout

The Shape editor is selected from the **Windows** menu. Its display looks like this. There are five major regions:

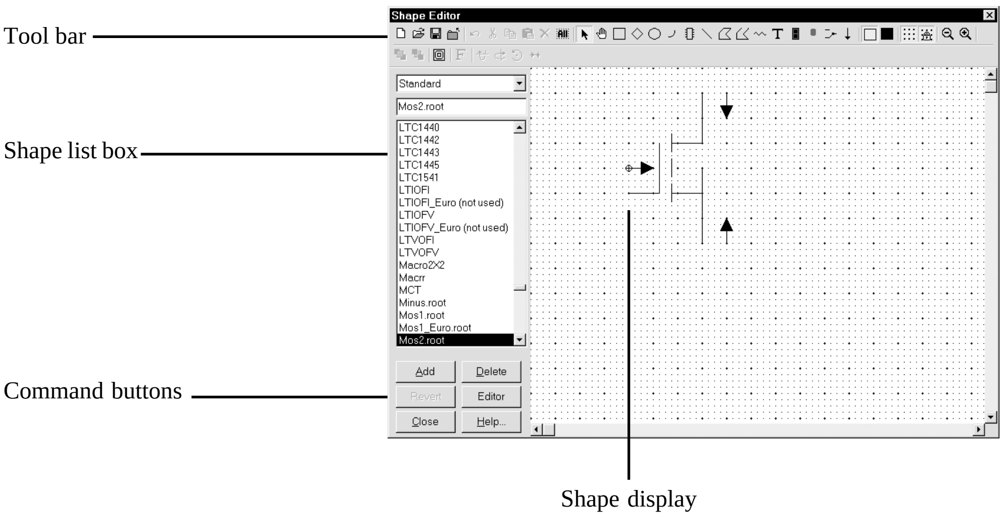


Figure 3-1 The Shape editor

Shape list box:

This list box displays all of the shapes currently in the library.

Command buttons:

Add: This command adds a new shape to the library.

Delete: This command deletes the currently selected shape.

Revert: This command restores the selected shape to its original version when the shape was first displayed. This only affects the displayed shape. Edits on other shapes during the current invocation of the Shape editor remain in effect. Selecting a new shape disables the Revert feature for the old shape. All edits on all shapes are temporary, however, and may be discarded when you exit the editor or close the file. Only when you exit or close the file are the changes saved to disk.

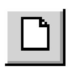
Editor: This invokes the Object editor, which lets you edit the numeric parameters of the fundamental objects comprising the selected shape. This provides finer control over object size and shape than can be obtained with the normal mouse editing. Some features, such as the digital block pin symbols, are only changeable from the Object editor.

Close: This closes the Shape editor and, if any changes have been made, asks if you want to retain them.

Help: This accesses the Help system.

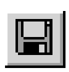
Tool bar:

The Tool bar provides the buttons to select tools for creating, editing, and viewing the selected shape. The tools and their properties are as follows:

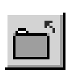


New: (CTRL + N) This command creates a new shape library file. Any shapes added to it are available for use in the Component library.

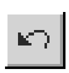
Open: (CTRL + O) This command loads an existing shape library file. Its shapes are then available for use in the Component library.



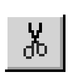
Save As: This command saves the current shape library file under a new name specified by the user.



Remove: This command removes the currently loaded shape library file. Its shapes are no longer available for use in the Component library.

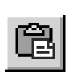


Undo: (CTRL + Z) Most operations that change a shape can be reversed with the Undo command. Undo will only reverse the last change.

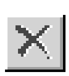


Cut: (CTRL + X) This command deletes the selected objects and copies them to the clipboard.

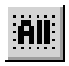
Copy: (CTRL + C) This command copies selected objects to the clipboard from where they may be pasted to the Shape display.

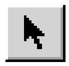


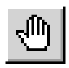
Paste: (CTRL + V) This command copies the contents of the clipboard starting at the last mouse position.

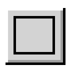


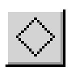
Clear: (DELETE) This command deletes the selected items without copying them to the clipboard.

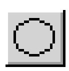

All: (CTRL + A) This command selects all items in the shape.

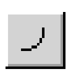

Select: Click this button to activate the Select mode. You must be in Select mode to edit or select a portion of shape for editing.

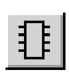

Pan: Click this button to activate the Pan mode. Panning is used to move the display view to see different parts of a large shape. As usual, you can also use the keyboard or the mouse. Mouse panning involves dragging the right mouse button and can be done in any mode. Alternatively, you can select this mode and drag with the left mouse button.

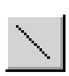

Rectangle: Click this button to create a rectangle in the shape by dragging the mouse in the Shape display. To change the shape of the rectangle, drag one of the eight handles (small black rectangles).

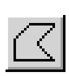

Diamond: Click this button to add a diamond in the shape by dragging the mouse. To change the shape of the diamond, drag one of the eight handles.

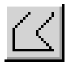

Ellipse: Click this button to add an ellipse to the shape by dragging the mouse in the Shape display. To change the shape of the ellipse, drag one of the eight handles. To create a circle, press the SHIFT key before dragging the mouse.


Arc: Click this button to add an arc to the shape by dragging the mouse in the Shape display. To change the shape of the arc, drag one of the eight handles.

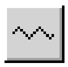

Block: Click this button to create a digital block in the shape by dragging the mouse in the Shape display. To change the shape and number of leads, drag one of the eight handles. To edit the leads to reflect their function, select the block and then invoke the Object editor by clicking on its command button.


Line: Click this button to create a line in the shape by dragging the mouse in the Shape display. To change the direction or length of the line, drag on one of the two handles.

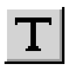

Closed Polygon: Click this button to add a closed polygon. Click the mouse in the Shape display, once for each vertex. There is no limit on the number of vertices. Double-click on the last vertex or click the right mouse button. This finishes the polygon by adding a final edge between the first and last vertices. To change the overall dimensions of the polygon, drag one of the handles. Use the Object editor to edit the individual vertex coordinates.



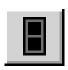
Open Polygon: Click this button to create an open polygon. Click the mouse in the Shape display, once for each vertex. There is no limit on the number of vertices. Double-click on the last vertex or click the right mouse button to end the polygon. To change the overall dimensions of the polygon, drag one of the eight handles. Use the Object editor to edit the individual vertex coordinates.



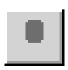
Included Shape: Click this button to include an existing shape in the current shape by clicking the mouse in the Shape display where you want the shape placed. This invokes a list of existing shapes. Choose the shape you want to include from this list. Once the shape is included, it can be dragged about.



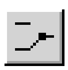
Text: Click this button to add text to the shape by clicking the mouse in the Shape display where you want the text placed. This invokes a text dialog box. Type the text and press Enter. To edit text, double-click on it in Select mode.



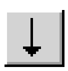
Seven Segment LED: Click this button to create seven segment LED shapes for use in animated components.

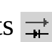


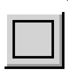
LED diode: Click this button to create an LED diode shape for use in animated components.



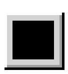
Switch: Click this button to create a switch shape for use in animated components.



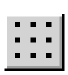
Current: Click this button to add a direction indicator for positive current. It is used only when the schematic Currents  mode is enabled.



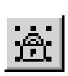
Outline: Click this button to create outlined shapes, rather than filled shapes, when dragging a rectangle, diamond, ellipse, or closed polygon.



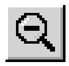
Fill: Click this button to create filled shapes, rather than outlined shapes, when dragging a rectangle, diamond, ellipse, or closed polygon.



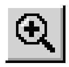
Grid: Click this button to show the grid. The grid is a two-dimensional array of locations where shape objects must originate and terminate if their nodes are accessible in a schematic.



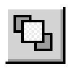
Grid Snap: Click this button to force the coordinates of all shape objects to occur at a grid point. This ensures that the coordinate points will be usable as connecting points (pins) in a schematic.



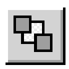
Zoom-Out: Click this button to decrease the displayed image size. This does not affect the size of the image in the schematic.



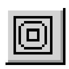
Zoom-In: Click this button to increase the displayed image size. This does not affect the size of the image in the schematic.



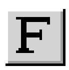
To Front: Click this button to send the selected object in an overlapping stack of objects to the front.



To Back: Click this button to send the selected object in an overlapping stack of objects to the back.

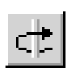


Next Object: Click this button to select a different object in a stack of overlapping objects. Click the button until the one you want is selected.

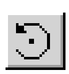


Font: Click this button to change the text attributes of any selected text. This also changes the default text attributes that will be used the next time text is added to the shape.

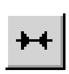
Flip X: This command rotates the selected region about the X axis in 180 degree increments, essentially flipping the object or group about the X axis.



Flip Y: This command rotates the selected region about the Y axis in 180 degree increments, essentially flipping the object or group about the Y axis.



Rotate: This command rotates the selected region about the Z axis (the Z axis is perpendicular to the schematic plane) in 90 degree increments, producing four distinct orientations.



Mirror: This command creates a mirror image copy of the selected region. It invokes the Mirror dialog box which lets you choose a vertical or horizontal reflection and whether you want to copy the text in the region. A vertical reflection produces a copy vertically below the original region. A horizontal reflection produces a copy horizontally to the right of the original region.

Shape drawing window:

This window shows the selected shape. You can change the image size, and pan or scroll the window to see larger shapes.

The Object editor

Most objects can be created and edited by simple mouse operations. For the closed and open polygons and the block, it is necessary to use the Object editor to edit the object's characteristics. For other objects, it is sometimes convenient to use the editor to tweak the numeric values that control the size and shape of the object, after the object has been roughly drawn with the mouse.

The Object editor is invoked by clicking the Editor button or by double-clicking on the object that you want to edit. The editor looks like this:

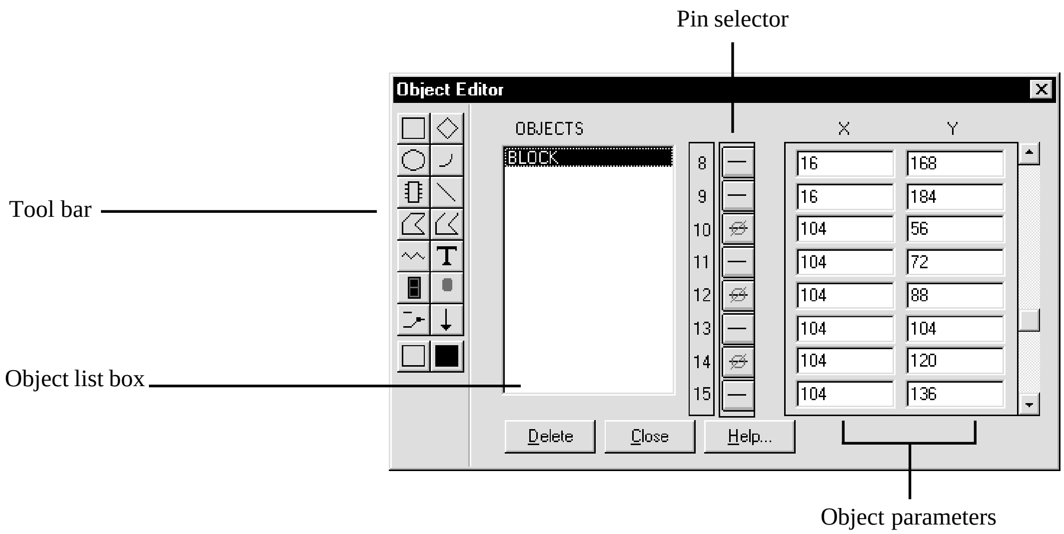


Figure 3-2 The Object editor

There are five major parts to the Object editor:

Tool bar:

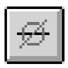
This lets you select new objects to add to the existing list. Click on an object to add it to the object list.

Object list box:

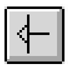
This shows the objects that the shape is composed of.

Pin selector:

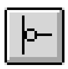
When the selected object is a block, this shows the pin symbols. These symbols are intended to iconically convey the existence and function of the pin. They do not, of course, affect the actual behavior of the pins that they represent. There are five basic pin symbols:



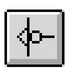
Open: No pin at this location.



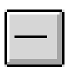
Clock: A clock pin.



Inversion: A logical inversion.



Inverted Clock: An inverted clock pin.



Normal: No indication of the pin function.

Object parameters:

These fields hold the numeric values that characterize the object. Rectangles, diamonds, ellipses, arcs, lines, and text use two sets of coordinates to define the coordinates of two opposite corners of the objects's bounding box. The bounding box is the rectangle that contains the object. Polygon parameters are a set of N coordinates of the form X,Y. The Included Shape parameters are the coordinates of the position where the included shape is placed. Coordinates are numerically equal to the number of grids from the upper left origin at the highest scale.

The Shape library

The Shape library contains the graphical shapes used in schematics to represent components. Each shape is comprised of various graphical primitives. Shapes are created, edited, and maintained by the Shape editor. The name of the file that holds the standard Shape library is called STANDARD.SHP. This file is found on the same directory as the MC7.EXE program.

It is possible to have more than one shape file in the shape library. Multiple files can be maintained, and viewed with the Shape editor, but it is recommended that the original shape library, STANDARD.SHP, be left unmodified in its original state.

Chapter 4 The Component Editor

What's in this chapter

The Component editor manages the Component library. This library provides the circuit components used in MC7 circuits. It stores the name of each component, the shape it uses, the electrical definition, component text placement, and pin information. All components, from resistors to macros and SPICE subcircuits are linked to MC7 using the Component editor.

This chapter is organized as follows:

- The Component editor layout
- Adding components to the library
- Adding subcircuits to the library
- Using the Add Part wizard
- Using the Import wizard
- Making circuit files portable

Features new in Micro-Cap 7

- Add Part wizard
- Automatic import from a circuit file
- Find command
- Import wizard
- Info command
- Move command
- Parts list
- Redo command
- Replace command
- Sort command

The Component editor layout

The Component editor is accessed from the Windows menu. It looks like this:

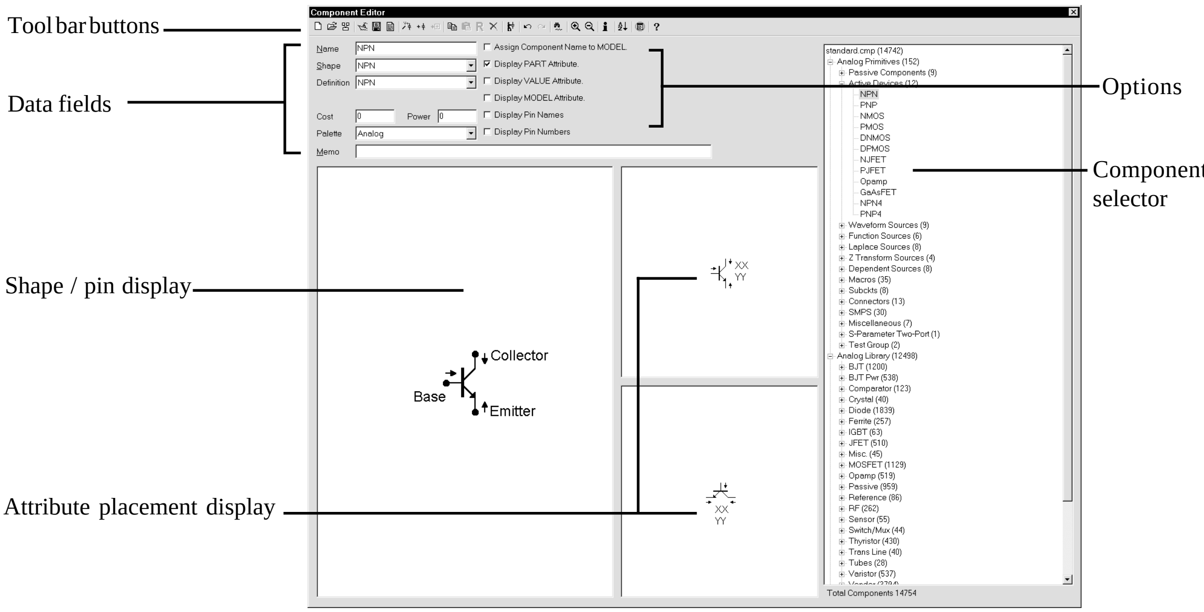
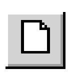
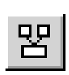


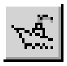
Figure 4-1 The Component editor

Tool bar buttons:

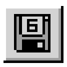

New: (CTRL + N) This command creates a new Component library file with a single group. Its components are then available for use in schematics and shown in the Component menu.


Open: (CTRL + O) This command loads an existing MC5, MC6, or MC7 library file. Its components are then available for use in schematics and are shown in the Component menu.


Merge: This command lets you merge an MC4, MC5, MC6, or MC7 Component library file with the current MC7 Component library. It provides a dialog box to let you locate the external library file that you want to merge into the current library. Only unique components from the external library file are included. Components with duplicate names are not merged. If an incoming part uses a shape whose name isn't in the current MC7 Shape library, the shape is copied from the external shape library to the current MC7 Shape library. A '$' is added to the shape name.



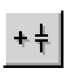
The *Import Wizard* button imports subckt-based parts from a text file *en masse*. It copies the model file to the library folder, adds the file name to the master index file, NOM.LIB, and makes the required entries in the Component library. It finds all of the subcircuits in the file and enters each as a new part using the subckt name. Part names already in the library are ignored. New parts are added using as a guide a part selected by the user from a list of parts whose pin names match the new part. New parts whose pins do not match existing parts are added with generic shapes and their memo fields are annotated to indicate that additional work is needed to complete them. Usually this involves picking a suitable shape and placing the appropriate pins on the shape. The Import wizard is optimized for automating the mass import of vendor-modeled parts.



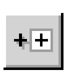
The *Export to MC6* button saves the Component library file in a format that MC6 can read.

The *Parts List* button creates a text file containing the part names from the currently selected group or from the entire library.

The *Add Part Wizard* button integrates all of the actions necessary to add a single part to the MC7 libraries. Like the Import wizard, it copies the model file to the library folder, adds the file name to the master index file, nom.lib, and makes the required entries in the Component library. It differs from the Import wizard in that it handles a single part of any type, not just subcircuits.



Add Component: This adds a new component if the name selected in the Component selector window is any group or component name except the highest level, the library file name.



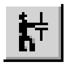
Add Group: This adds a new group name if the item selected in the selector window is any group name. If the current selection is a component name, this option is disabled, since you can't add a group to a component.

Copy: This command copies the current component to the clipboard, awaiting a paste operation.

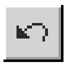
Paste: This command pastes the clipboard component with a generic name to the slot before the currently selected component in the Component Selector.

Replace: The command replaces the current component description with the one in the clipboard except for the part name which remains the same.

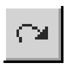
Delete: This deletes the selected file, group, or component. Only empty groups may be deleted. If the group contains another group or a component, an error message will result. Files are deleted only from the library, not from disk.



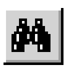
The *Move* button accesses a dialog box which facilitates moving multiple parts from one part of the library to another.



The *Undo* button undoes prior edits. It is a multistage undo for edits to the data fields. It does not undo part additions/deletions.



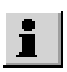
The *Redo* button restores prior edits. It is a multistage redo for edits to the data fields. It does not redo part additions/deletions.



The *Find* button finds parts whose name, shape, definition, or memo fields match a specified text string.

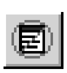
Zoom-In: This command increases the displayed shape size. It doesn't affect the shape size as seen in the schematic.

Zoom-Out: This command decreases the displayed shape size. It doesn't affect the shape size as seen in the schematic.



The *Info* command loads and displays relevant model information for the part, usually a subckt listing or a model statement.

The *Sort* command lets you sort the parts and/or groups alphabetically.



Clear Palettes: This command clears the user-defined content of the component palettes.

Help: This command accesses the help files for the Component editor.

Data fields:

Name: This is the component name as it appears in the Component menu.

Shape: This is the name of the shape that is drawn when the component is added to a circuit.

Definition: This is the electrical definition of the component. It implicitly defines the mathematical model to be used. It does not specify the numeric parameters. These come from a model library or a model statement name in the Attribute dialog box when the component is added to a circuit.

Cost: This optional field is used in the Bill of Materials report where it is listed individually and included in the circuit totals.

Power: This optional field is used in the Bill of Materials report where it is listed individually and included in the circuit totals.

Memo: This field may be used for any desired documentation purpose. It is mainly used to describe the function of the component.

Palette: This list box lets you assign the currently displayed part to one of the component palettes.

Options:

Assign Component Name to MODEL (or NAME , or File, or others): This automatically assigns the component name to the MODEL or NAME attribute when the component is added to a schematic. This bypasses the Attribute dialog box, simplifying the process of adding components. This option is enabled on all Analog Library and Digital Library parts and is disabled on all Analog Primitive and Digital Primitive parts.

Display PART Attribute, Display VALUE Attribute, and Display MODEL Attribute: These options set the display flag for the attributes. They control the initial display only, when the part is first placed. Afterwards, display is controlled by toggling the appropriate display flag in the Attribute dialog box.

Display Pin Names and *Display Pin Numbers:* These options set the initial display flag for showing pin names and package pin numbers. These flags can be individually changed on the part when it is placed in a schematic. Pin names are normally shown for complex LSI parts, but not shown for simpler components, such as diodes and transistors, since their shapes adequately identify the pin names. It is easy to identify the base lead of an NPN, but not so easy to identify pin names on a complex linear part. Pin number display is mostly used to facilitate PCB work.

Shape / pin display:

This display shows the shape and its pins. Placing named pins on the shape associates specific shape locations with electrically, subcircuit, or macro defined pin names. When you add a basic component, like an NPN transistor, pin names automatically appear and need only be dragged to suitable locations on the shape. Pins and their names may be independently located on the shape by dragging the pin dot or the pin name.

Since macros and subcircuits have no intrinsic pin names, the user must specify the pin names and their position on the shape. This is done by clicking in the Shape / pin display, typing the pin name into the Pin Name dialog box, specifying whether it is an analog or a digital pin, and then dragging the pin to the proper position on the shape.

Editing an existing pin name involves double-clicking on the pin name or pin dot to invoke the Pin Name dialog box.

Hidden pins are assigned to a fixed named node in the schematic and are never displayed. This type of pin is occasionally used to simplify the process of wiring up power pins.

Additional pin fields:

Some digital parts have a variable number of pins and thus require an additional Pin field to specify the number of input, output, or enable pins. For example, a nor gate may have an unlimited number of inputs. An Input field then specifies the number of inputs.

Attribute placement display:

This display shows the shape and the location where the attribute text will initially be placed. The initial attribute text location is specified by dragging the generic text block to the desired location relative to the shape in the two basic orientations. This only affects the initial location when the part is first placed in a schematic. Thereafter, attribute text can be dragged to a new location.

Component Selector:

The selector is a hierarchical list box that lets you select a component for viewing or editing. To open or close a group, double-click on it, or click once on its + or - box. To select a component, click on it.

The display order used in the Component editor is also used in the Component menu to select parts for placement in the schematic.

Total Components:

This shows the number of components in all files currently open in the Component library.

Adding components to the library

As originally supplied, the Component library contains all of the usual components that you ordinarily need. The main use of the Component editor is to define new macro and subckt components, although you might find it desirable to add your own common components. You may, for instance, wish to create several types of grounds, or several types of resistors to reflect different physical construction, or to explicitly denote the plus and minus leads. Users may also need to add a part whose model has been supplied by a vendor in the subckt format.

To add a component to the library you first choose its group by clicking on the group name in the component selector. Next, you click on the Add Component button and specify the following:

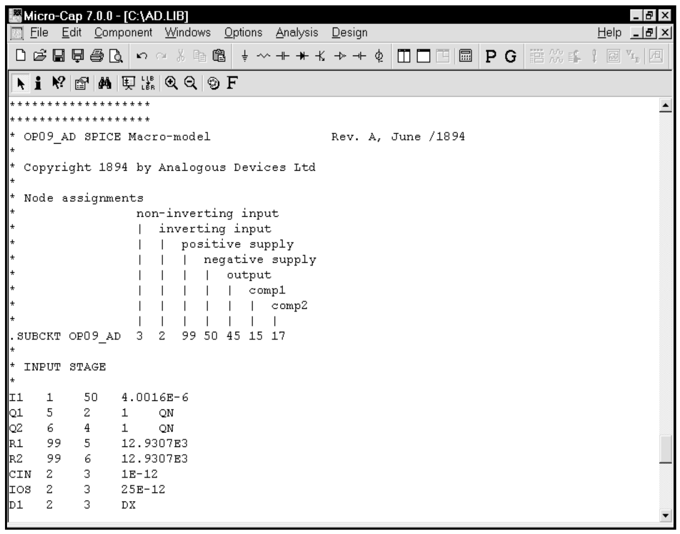
- The name of the component.
- The name of the shape it uses.
- The electrical definition.
- An optional memo description.
- The initial text coordinates of the attributes (by dragging text into place).
- The pin name locations (by dragging pin name origin markers into place).
- The display options.

If the component is a macro, subcircuit, logic expression, pindly, or constraint, then named pins must be explicitly added.

Normally, logic expressions, pindlys, and constraints are used only in subcircuits as building blocks for the Digital library, so the Component library has only token components of this type. It is not expected, for example, that a user will want to add a logic expression to a schematic, but it is possible.

Adding subcircuits to the library

Many vendor-supplied models are already included in the library. This section shows you how to add new ones. We'll add a hypothetical OPAMP called the OP09_AD. This component is similar to the OP08 component supplied by Analog Devices. The subcircuit, as typically supplied by a vendor, is shown below.




```
*******************
*******************
* OP09_AD SPICE Macro-model                Rev. A, June /1894
*
* Copyright 1894 by Analogous Devices Ltd
*
* Node assignments
*                non-inverting input
*                |  inverting input
*                |  |  positive supply
*                |  |  |  negative supply
*                |  |  |  |  output
*                |  |  |  |  |  comp1
*                |  |  |  |  |  |  comp2
*                |  |  |  |  |  |  |
.SUBCKT OP09_AD  3  2  99 50 45 15 17
*
* INPUT STAGE
*
I1   1    50   4.0016E-6
Q1   5    2    1     QN
Q2   6    4    1     QN
R1   99   5    12.9307E3
R2   99   6    12.9307E3
CIN  2    3    1E-12
IOS  2    3    25E-12
D1   2    3    DX
```

Figure 4-2 The OP09_AD subcircuit model

Note that the subcircuit employs seven pins, arranged as follows:

Pin name	Function
3	Non-inverting input
2	Inverting input
99	Positive supply
50	Negative supply
45	Output
15	Comp1
17	Comp2

The first decision is where to put the new component. Since this is a vendor supplied model, put it in the Vendor section. Click on the Analog Library + symbol in the Component selector. Click Vendor, then Analog2, then OP08_AD-. Finally, click on the Add Part button. Type "OP09_AD" in the Name field. Press the Tab key to move to the Shape field. Press O until OPAMP7 is shown. Press the Tab key to move to the Definition field. Press S until Subckt appears. Click on the Assign Component Name to NAME check box. Turn the Display PART Attribute and Display NAME Attribute check boxes on.

The next step is to add the pins. Click in the Shape / pin display. Type "3" into the Pin Name dialog box that appears. Note that the default pin type is analog. Click on the OK button. This places a pin named 3 in the display. Drag the pin dot to the non-inverting input and release it there. Drag the pin text to just above the pin dot. Now add the remaining pins in the same manner.

Drag the initial attribute text XX YY in the two display areas to a suitable position out of the likely wire connection paths. The final display should look like this:

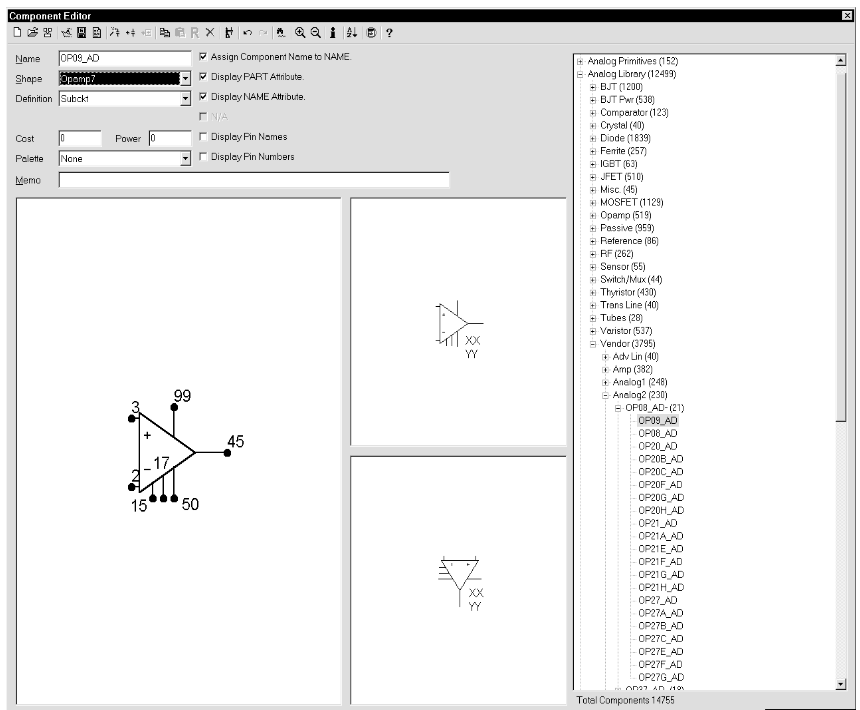


Figure 4-3 Entering the OP09_AD subckt in the Component library

This completes the Component library entry. Exit the Component editor. Click OK if you want to save the part edits. If you do click OK, the edits are saved in the Component library, assuring that it is on the Component menu and available for placement in a schematic. It also means that MC7 knows that it is a subcircuit and will be expecting to find the subcircuit model in one of several places:

1. In the text area.
2. In the optional file listed in the FILE attribute.
3. In a file referenced in a user .LIB statement.
4. In a file referenced in the default .LIB NOM.LIB statement.

.LIB statements are the preferred way of handling vendor-supplied subcircuit models. In this case you would presumably have received a file from the vendor. Lets call it AD.LIB. The file would contain the subcircuit model for the new OP09_AD device and possibly many more besides. The name of the new file must be added to the .LIB statements you plan to use in the schematic. If you do not add .LIB statements to your schematics but rely on the default NOM.LIB statement, then the new file name must be added to the NOM.LIB file so that MC7 can access it. The NOM.LIB file is a standard text file, so you can add the new entry using any text editor, including the MC7 text editor. Load NOM.LIB from the *library* directory. Add the following text to the NOM.LIB file.

.LIB "AD.LIB"
(if AD.LIB is in the current library directory)

.LIB "C:\MYPATH\AD.LIB"
(if AD.LIB is at C:\MYPATH)

On the next access of the NOM.LIB file, MC7 will discover that it has been modified and will regenerate a new index to all of the subckt, model, and macro statements in the file. This can take several seconds. After the index is generated, finding a particular model, subckt, or macro statement is quite fast. Indices are regenerated only when the NOM.LIB file or one of the files referenced in the NOM.LIB file changes.

Note that the component name as entered in the Component library must match the subckt name exactly.

No extra or missing characters are allowed. Many vendors supply standard components using the same name, so MC7 libraries must distinguish them. That is why a two letter name like '_AD' is appended to the name as supplied by the vendor. For example, the LM118 supplied by Linear Technology is LM118_LT. The National Semiconductor version is LM118_NS. If you add duplicate models, you should rename them also. Don't forget to change the name used in the .SUBCKT statement in the vendor supplied file to match the new name.

Using the Add Part wizard

The lengthy procedure just described shows how to add a subckt part from scratch. The Add Part wizard can often simplify this task. It combines all of the steps required to enter a new part and is particularly nice for subckt parts where it can often guess the correct pin placement and shape based upon similar parts already in the library.

We'll illustrate the procedure with the sample file, AD.LIB supplied with MC7, and located in the library folder. Pretend that you have just downloaded it from a vendor. In the file is a subckt description for the OP09_AD opamp. Here is how you add it to the library using the Add Part wizard.

1) Select **Component Editor** from the **Windows** menu.

2) Select the group where you want the part name to appear in the Component menu. Select Analog Library / Vendor / Analog2 / OP08_AD.

3) Click on the Add Part wizard  button.

4) The first prompt is for the part name. Type in "OP09_AD". Click Next.

5) The next prompt is for the electrical definition. Select Subckt. Click Next.

6) The next prompt asks for the name of the file containing the subckt. Type AD.LIB (the file is already in the library folder so no path is needed). Click Next.

7) The next prompt is the tricky one. The wizard scans the Component library and compiles a list of all subcircuit parts that have pin names matching the OP09_AD. Actually it presents only a representative list, as there may be many matching parts. Accept the recommended choice since this is a typical seven-pin opamp model. Click Next.

8) The next prompt asks for an optional memo field. Click Next.

9) The next prompt asks for an optional palette assignment. Click Next.

10) The next panel lets you set the initial display of the various part attributes. Click Next.

11) The next panel lets you make the assignment of the component name to the

NAME attribute. This assignment avoids the need to invoke the Attribute dialog box when one of these parts is added to a schematic. Click to enable the option. Click Next.

12) The final panel advises you to examine the entry to be sure all elements have been selected properly, particularly the choice of shape and where the pin names have been placed on the shape.

For our example, the new part should look like this:

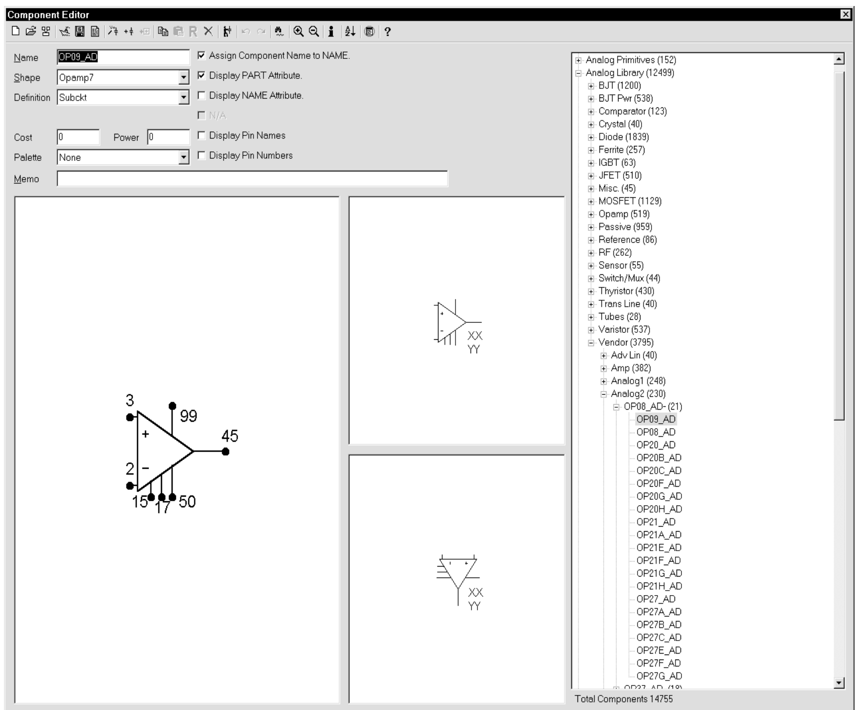


Figure 4-4 Using the Add Part wizard

Using the Import wizard

MC7 also includes an Import wizard for importing large numbers of similar subcircuit-modeled parts.

We'll illustrate the procedure with the sample file, AD.LIB supplied with MC7, and located in the library folder. We'll pretend that you have just downloaded it from a vendor. In the file are multiple subcircuits. Here is how you add them to the library using the Import wizard.

1) Select **Component Editor** from the **Windows** menu.

2) Select the group where you want the part names to appear in the Component menu. For this illustration simply select Analog Library.

3) Click on the Import wizard  button.

4) The first prompt is for the file name. Type in "ad.lib". In this example, you don't need to specify a path since the file is already in the MC7 library folder. If it were not, you would specify or browse to the location of the file. Click Next.

5) The next prompt is for an optional suffix. Click Next.

6) The program scans the ad.lib file looking for subcircuits. When it finds one, it scans the library for parts with the same pin names. There are several possible outcomes from this search:

Exact Match: If the search finds one or more exact pin matches and all matching parts use the same shape it simply enters the part using the shape and pin locations from the matching parts.

Near Miss: If the search finds one or more exact pin matches and the matching parts use different shapes, it presents a list and asks you to select a part. If you're unsure about which part to use, scan the subcircuit listing in the Info dialog box, especially the comments near the .SUBCKT line. These often reveal enough about the part to pick a suitable shape. You may want to create a special shape for the part.

No Match: If the search finds no matches, it enters the part using a generic shape and pin placements and annotates the memo field to indicate additional work is needed.

In this example, the search turned up one near miss part, the IRF5101A, an N-channel MOSFET. Select the MTD10N10EL_ON part, which uses the DNMOS shape. Click OK.

The search results are shown in the dialog box. Several parts with exact matches were found and these parts were entered without assistance. No matches were found for the ODDBALL part so it was entered generically. The IRF5101A was entered with the MTD10N10EL_ON part as a template.

At this point you can click the Finish button or the Cancel button. Pressing the Cancel button ignores the imported parts. Pressing the Finish button adds them, but does not yet write them to disk. When you exit the Component editor, you can elect to save the changes by writing them to disk.

For our example, the display should look like this:

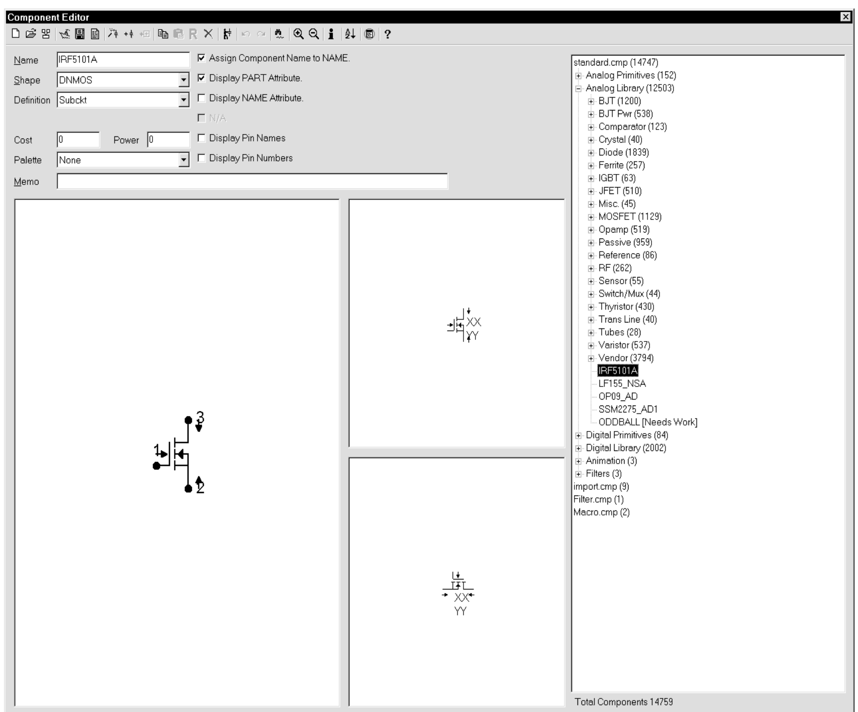


Figure 4-5 Using the Import wizard

Using Copy, Paste, and Replace

The clipboard can also be used for rapid entry of similar parts. To illustrate, assume we want to enter a number of 5 pin opamp parts from Analog Devices that use the same pins as the OP08_AD part listed previously in this chapter.

1) Select **Component Editor** from the **Windows** menu.

2) Select the part that is to serve as a template. For this illustration simply select Analog Library / Vendor / Analog2 / OP08_AD / OP08_AD.

3) Click on the Copy  button. The program copies the OP08_AD to the clipboard.

4) Click on the Paste  button. MC7 pastes the clipboard part (the OP08_AD) to a new position, and names it new_1. Change its name to OP09_AD and you've just entered the OP09_AD part. Each new part added takes one click and a few keystrokes for the name.

The Paste operation adds a new part using all of the properties of the clipboard part except the name which is generic and must be changed by the user.

The Replace operation is similar, but it does not add a part. It merely replaces the properties of the selected part with those from the clipboard part, except for the name, which remains the same.

The purpose of the Replace command is to replace many nearly identical parts with a new template where the pin placements or attribute text placement is to be made uniform.

Making circuit files portable

Older versions of Micro-Cap could not load a circuit file if it contained a part that was not already in the component library. If you wanted to share a circuit with a friend, you had to send your component and sometimes your shape library along with the circuit. That is no longer necessary with MC7.

To improve file portability, circuit files saved in MC7 format contain copies of their shape and component library data. A friend's MC7 can read your circuit file without needing your shape and component libraries because the circuit file itself has that information.

When MC7 loads a circuit file that has a part that is not in the Component library (i.e. not listed in any of the currently open component library files), it reads the shape and component data from the circuit file, automatically creates an entry for the part, and places it in the auxiliary component library file, IMPORT.CMP. This makes the part instantly usable. Note that the circuit file must have been saved by MC7 for this feature to be available.

You can drag or move the part into any group of any component library file. Until deleted, the part is available for use in any circuit.

If your friend also wants to run a simulation, they will need model information as well. The Refresh  button adds or updates model statements and subckts and copies them into the circuit file so that the circuit file alone is all that is needed to run an analysis.

Macro circuits (.MAC) used in a circuit are separate files and are not added or updated by the Refresh command. It will still be necessary to send them along with the circuit file if you want total file portability.*

Chapter 5 The Package Editor

What's in this chapter

The Package editor manages the Package library. This library provides the package information for the components in the MC7 Component library and enables the program to create netlist files for use by external PCB programs. It stores the name of each package and the pin information for each package. All package information is linked to the components using the Package editor.

This chapter is organized as follows:

- The Package editor layout
- Adding basic packages to the library
- Adding complex packages to the library

Features new in Micro-Cap 7

- PADS PCB field added.
- Package library is automatically loaded.
- Package editor shows the last referenced part when invoked.
- Info feature

The Package editor layout

The Package editor is accessed from the Windows menu. It looks like this:

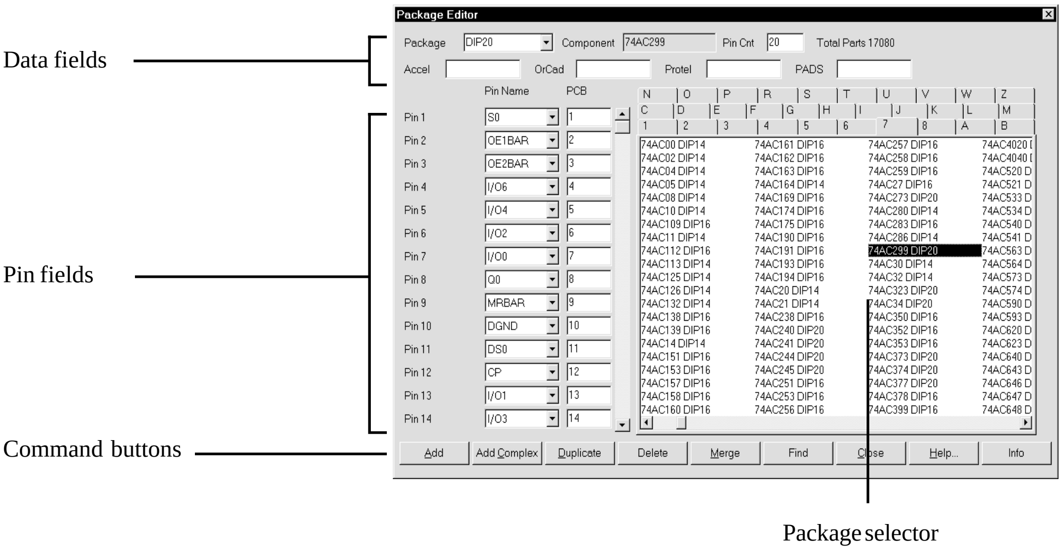


Figure 5-1 The Package editor

Command buttons:

Add: This command adds an entry to the Package library. A Find dialog box appears requesting the name of the component whose package and pin information is to be defined. This command should be chosen when the package contains only one instance of the specified component.

Add Complex: This command adds an entry to the Package library. A Find dialog box appears requesting the name of the component whose package and pin information is to be defined. This command should be chosen when the package contains multiple instances of the specified component.

Duplicate: This command duplicates the currently selected entry, except for the component name, which is chosen from a Find dialog box. The purpose of the command is to speed data entry when the new part is very similar to an existing part, except for the component name.

Delete: This command deletes the highlighted package.

Merge: This command merges a Package library file (*.PKG) with the current Package library file. It provides a dialog box to let you locate the external library file that you want to merge with the current library. Only unique packages from the external library file are included. Packages with duplicate names are not merged.

Find: This command finds a specified library entry. A Find dialog box appears requesting the text of the entry. The text would typically be the part name followed by the package type, although the search routine works with fragments by returning multiple items. For example, performing a find on "740" will return all entries that start with 740.

Close: This command closes the Package editor and optionally saves any changes to the STANDARD.PKG file.

Help: This command accesses the Package editor Help system.

Info*:* This command displays model information for the part.

Data fields:

Package: This is the package as it will appear in the Attribute dialog box of the specified component for the PACKAGE attribute.

Component: This is the component that the package is being defined for. This field is fixed except when the package is first entered from an Add, Add Complex, or Duplicate command.

Pin Cnt: This controls the pin count for the entry.

Accel: This field overwrites the Package field when the schematic is translated to an Accel netlist.

OrCad: This field overwrites the Package field when the schematic is translated to an OrCad netlist.

Protel: This field overwrites the Package field when the schematic is translated to a Protel netlist.

PADS: This field overwrites the Package field when the schematic is translated to a PADS netlist.

Pin fields:

These fields define the configuration of the pins in the PCB netlist. For a basic package, there are two fields: Pin Name and PCB. The Pin Name fields contain the names of the pins as they appear in Micro-Cap for the specified component. These are the pin names as they appear in the Component editor. If the Pin Name is set to NC#, for no connection, the corresponding PCB field is ignored. The PCB fields contain the names of the pins that will be used in the output PCB netlist. Normally, these are the pin numbers from the component's data sheet that correspond to the pins specified in the Pin Name fields. For a complex package, a third field will be present called Gate. Since a complex package has multiple instances of the component in it, the Gate fields specify which instance the pin is being defined for. A '*' in the Gate field indicates that all instances of the component in the package share this pin.

Package selector:

The selector is a list box that lets you select a package for viewing or editing. The packages are sorted in groups according to their first character. The tabs at the top of the selector control the group that is shown in the list box.

Adding basic packages to the library

The Package library contains package information for most of the components that come with Micro-Cap. The main use of the Package editor is to define new packages. Figure 5-2 displays the package information for the 74164 component which uses a basic package. The data sheet for this component can be found in TI's TTL Logic data book.

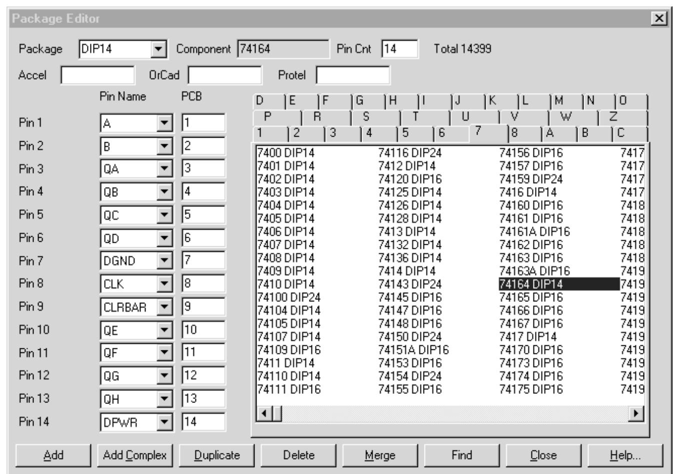


Figure 5-2 The Package editor settings for the 74164

Adding a basic package uses the following procedure:

- Click on the Add command button.

- In the Find Component dialog box, specify the component name such as 74164.

- Define the package type in the Package field such as DIP14.

- Define the number of pins in the Pin Cnt field such as 14.

- Define the Pin Name fields with their corresponding PCB fields. Click on the drop-down list of a Pin Name field to view the available pin names. For example, in TI's data sheet for the 74164, pin 1 on the package is specified as the A pin. In the Package editor, the Pin Name field should be set to A, and its corresponding PCB field should be set to 1.

Adding complex packages to the library

Figure 5-3 displays the package information for the 7400 component which uses a complex package. The data sheet for this component can be found in TI's TTL Logic data book.

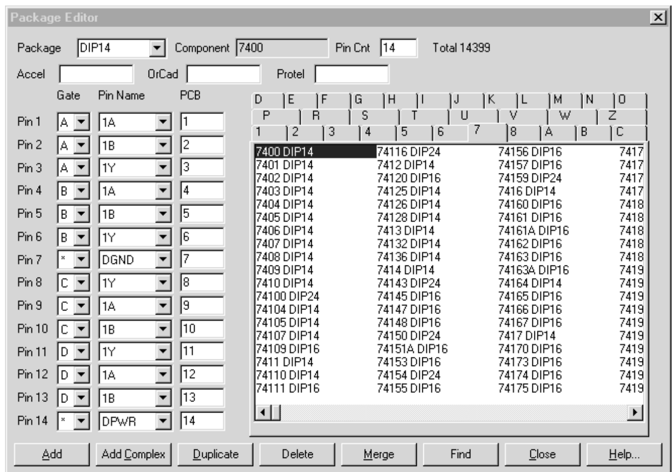


Figure 5-3 The Package editor settings for the 7400

Adding a complex package uses the following procedure:

- Click on the Add Complex command button.

- In the Find Component dialog box, specify the component name such as 7400.

- Define the package type in the Package field such as DIP14.

- Define the number of pins in the Pin Cnt field such as 14.

- Define the Pin Name fields with their corresponding Gate and PCB fields. Click on the drop-down list of a Pin Name field to view the available pin names. For example, in TI's data sheet for the 7400, pin 4 on the package is specified as the 2A pin which is the 1A pin for the second instance of the 7400 component in the package. In the Package editor, the Pin Name field should be set to 1A, the Gate field to B to denote the second instance of the component, and the PCB field to 4.

Chapter 6 Transient Analysis

What's in this chapter

Transient analysis requires the repeated iterative solution of a set of nonlinear time domain equations. The equations are derived from the time-domain models for each of the components in the circuit. The device models are covered in a later chapter.

The principal topics described in this chapter include:

- What happens in transient analysis
- Transient Analysis Limits dialog box
- Transient menu
- Initialization
- Using the P key
- Numeric output

Features new in Micro-Cap 7

- Optimizer
- Watch window
- Breakpoints window
- Range format <*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold spacing*>]

What happens in transient analysis

Transient analysis predicts the time-domain behavior of a circuit. It tries to predict what would happen if you built the circuit in the lab, hooked up power supplies and signal sources, and looked at the curves with a scope or a logic analyzer.

The program assumes that the circuit is fully nonlinear, though it can handle linear circuits as well. It begins by first constructing a set of nonlinear, time-varying, differential equations to represent the circuit. The remaining process is comprised of three steps:

- Initialization of state variables
- DC operating point (optional)
- Main transient analysis

Initialization of state variables:
The initialization process is explained in more detail later in this chapter. State variables include node voltages, inductor currents, and digital node states.

Optional DC operating point:
The purpose of the operating point is to establish, by iterative calculation, a stable set of state variable values that represent the steady state condition the circuit is assumed to have for the Time = 0 starting point of the transient analysis. The operating point is calculated by treating capacitors as open circuits and inductors as shorts. Using a DC nonlinear model for the other devices in the circuit, the program linearizes the model about the last set of state variable values. Linearizing means replacing the nonlinear model with simple numeric constants that express a linear relationship between the terminal voltages and currents of the device. These numeric constants are usually obtained by differentiating the state variables with respect to their controlling variables. The linear model is assumed to hold over the interval of one iteration. The program then solves for the incremental voltages and currents. It adds these increments to the prior state values, and checks to see if they have stabilized, or converged. When all state variables have converged, the operating point is complete and the program starts the main transient analysis.

Main transient analysis:
The main transient analysis begins with the state variables computed during the operating point, or the initialized values if the operating point was skipped. Using a standard nonlinear time-domain model for each device in the circuit, the program linearizes the models about the last set of state variable values. It then solves a

set of linear equations for incremental voltages and currents. It adds these linear increments to the prior state variable values and checks to see if they have stabilized. When all of the state variables have stabilized, convergence at this data point is achieved and the data point is evaluated to see if the local truncation error (LTE) is acceptable. If so, the time point is accepted and added to the plot, time is incremented, and the next data point is attempted. If the LTE is not acceptable, the data point is discarded, the time step is reduced, and a new data point is attempted. This process continues until the time variable equals the specified tmax.

To summarize, the basic conceptual sequence for Transient analysis is as follows:

1. Initialize state variables.
2. Optionally calculate the DC operating point.
3. Set Tlast = T = tmin and DT = minimum time step.
4. Solve for all state variables.
5. If variables have converged go to step 6, else go to step 4.
6. If LTE of state variables is acceptable go to step 8.
7. Discard time point: DT = DT/2, set T = Tlast , and go to step 4.
8. Plot or print requested variables. Set Tlast = T.
9. If Time equals tmax, quit.
10. Time = Time + time step.
11. Go to step 4.

The Transient Analysis Limits dialog box

Load the MIXED4 circuit file and select **Transient** from the **Analysis** menu. MC7 extracts the necessary circuit information directly from the schematic. More information is needed before the analysis can begin, and that information is supplied by the Analysis Limits dialog box.

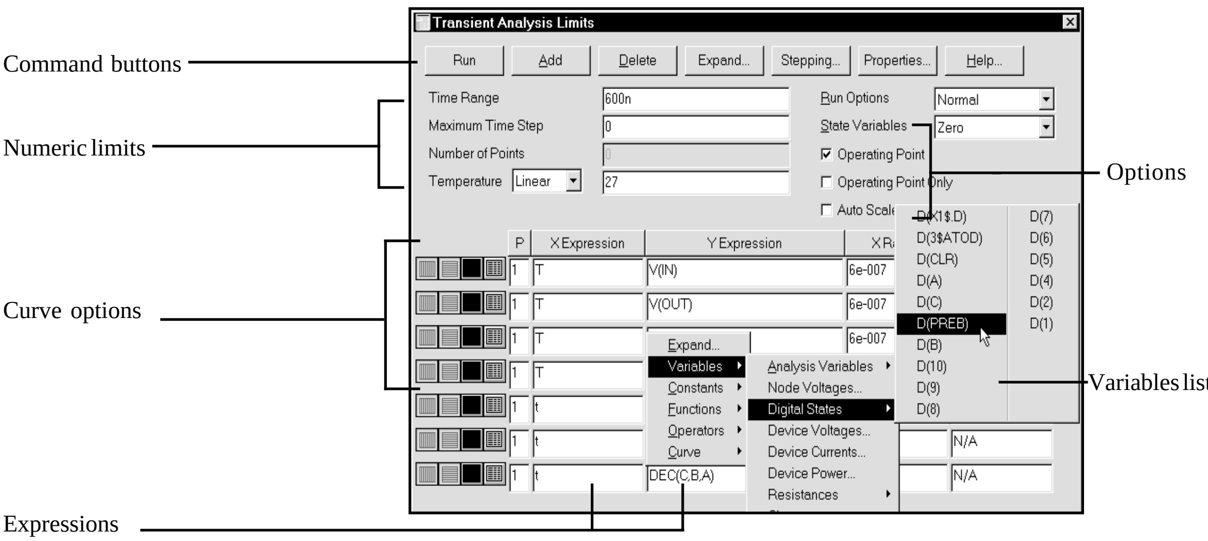


Figure 6-1 The Analysis Limits dialog box

The Analysis Limits dialog box is divided into five areas: the Command buttons, Numeric limits, Curve options, Expression fields, and Options.

Command buttons

The Command buttons, located just above the Numeric limits field, contain seven commands.

Run: This command starts the analysis run. Clicking the Tool bar Run button ▶ or pressing F2 will also start the run.

Add: This command adds another Curve options field and Expression field line after the line containing the text cursor. The scroll bar to the right of the Expression field scrolls through the curves when there are more than can be displayed.

Delete: This command deletes the Curve option field and Expression field line where the text cursor is.

Expand: This command expands the text field where the text cursor is into a large dialog box for editing or viewing. To use the feature, click the mouse in the desired text field, and then click the Expand button.

Stepping: This command invokes the Stepping dialog box. Stepping is reviewed in a separate chapter.

Properties: This command invokes the Properties dialog box which lets you control the analysis plot window and the way curves are displayed.

Help: This command invokes the Help screen which provides information by index and topic.

Numeric limits

The Numeric limits field provides control over the analysis time range, time step, number of printed points, and the temperature(s) to be used.

• **Time Range:** This field determines the start and stop time for the analysis. The format of the field is:

<*tmax*> [,<*tmin*>]

The run starts with time set equal to <*tmin*>, which defaults to zero, and ends when time equals <*tmax*>.

• **Maximum Time Step:** This field defines the maximum time step that the program is allowed to use. The default value, (<*tmax*>-<*tmin*>)/50, is used when the entry is 0.

• **Number of Points:** The contents of this field determine the number of printed values in the numeric output. The default value is 51. Note that this number is usually set to an odd value to produce an even print interval. The print interval is the time separation between successive printouts. The print interval used is:

print interval = (<*tmax*> - <*tmin*>)/([*number of points*] - 1)

• **Temperature:** This field specifies the global temperature of the run. This temperature is used for each device unless individual device temperatures are specified. If the Temperature list box shows Linear the format is:

<*high*> [, <*low*> [, <*step*>]]

The values are in degrees Celsius. The default value of <*low*> is <*high*>, and the default value of <*step*> is <*high*> - <*low*>.

If the Temperature list box shows List the format is:

<*t1*> [, <*t2*> [, <*t3*>] [,...]]
where *t1, t2,*.. are individual values of temperature.

All values are in degrees Celsius. One analysis is done at each specified temperature, producing one curve branch for each run.

Curve options

The Curve options field is located below the Numeric limits field and to the left of the Expressions field. Each curve option affects only the curve in its row. The options function as follows:

The first option toggles the X-axis between a linear and a log plot. Log plots require positive scale ranges.

The second option toggles the Y-axis between a linear and a log plot. Log plots require positive scale ranges.

The option activates the color menu. There are 64 color choices for an individual curve. The button color is the curve color.

The option prints a table showing the numeric value of the curve. The number of values printed is set by the Number of Points value. The table is printed to the Output window and saved in the file CIRCUITNAME.TNO.

A number from 1 to 9 in the (P) column selects the plot group the curve will be plotted in. All curves with like numbers are placed in the same plot group. If the P column is blank, the curve is not plotted.

Expressions

The X Expression and Y Expression fields specify the horizontal (X) and vertical (Y) expressions. MC7 can evaluate and plot a wide variety of expressions for either scale. Usually these are single variables like T (time), V(10) (voltage at node 10), or D(OUT) (digital state of node OUT), but the expressions can be more elaborate like V(2,3)*I(V1)*sin(2*PI*1E6*T).

Variables list

Clicking the *right* mouse button in the Y expression field invokes the Variables list which lets you select variables, constants, functions, operators, and curves, or expand the field to allow editing long expressions. Clicking the *right* mouse button in the other fields invokes a simpler menu showing suitable choices.

The X Range and Y Range fields specify the numeric scales to be used when plotting the X and Y expressions.

The format is:

<*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold grid spacing*>]

<*low*> defaults to zero. [,<*grid spacing*>] sets the spacing between grids. [,<*bold grid spacing*>] sets the spacing between bold grids. Placing "AUTO" in the X or Y range calculates the range automatically. The Auto Scale Ranges option calculates scales for all ranges during the simulation run and updates the X and Y Range fields. The Auto Scale (F6) command immediately scales all curves, without changing the range values, letting you restore them with CTRL + HOME if desired. Note that <*grid spacing*> and <*bold grid spacing*> are used only on linear scales. Logarithmic scales use a natural grid spacing of 1/10 the major grid values and bold is not used. The Auto Scale command uses the **Preferences / Auto Scale Grids** value to set the grid spacing.

Transient options

These options include the following:

- **Run Options**

  - **Normal:** This runs the simulation without saving it.

  - **Save:** This runs the simulation and saves it to disk, using the same format as in Probe. The file name is NAME.TSA.

  - **Retrieve:** This loads a previously saved simulation and plots and prints it as if it were a new run. The file name is NAME.TSA.

- **State Variables**

  These options determine the state variables at the start of the next run.

  - **Zero:** This sets the state variable initial values (node voltages, inductor currents, digital states) to zero or X.

  - **Read:** This reads a previously saved set of state variables and uses them as the initial values for the run.

- **Leave:** This leaves the current values of state variables alone. They retain their last values. If this is the first run, they are zero. If you have just run an analysis without returning to the Schematic editor, they are the values at the end of the run. If the run was an operating point only run, the values are the DC operating point.

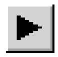
- **Operating Point:** This calculates a DC operating point. It uses the initial state variables as a starting point and calculates a new set that represents the DC steady state response of the circuit to the T=0 values of all sources.

- **Operating Point Only:** This calculates a DC operating point only. No transient run is made. The state variables are left with their final operating point values.

- **Auto Scale Ranges:** This sets the X and Y range to AUTO for each new analysis run. If it is not enabled, the existing scale values from the X and Y Range fields are used.

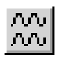
The Run, State Variables, and Analysis options affect the simulation results. To see the effect of changes of these options you must do a run by clicking on the Run command button or pressing F2.

The Transient menu

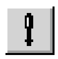


Run: (F2) This starts the analysis run.

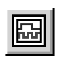
Limits: (F9) This accesses the Analysis Limits dialog box.



Stepping: (F11) This accesses the Stepping dialog box.



Optimize: (CTRL + F11) This accesses the Optimize dialog box.



Analysis Window: (F4) This command displays the analysis plot.

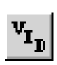
Watch: (CTRL + W) This displays the Watch window where you define expressions or variables to watch during a breakpoint invocation.

Breakpoints: (ALT + F9) This accesses the Breakpoints dialog box. Breakpoints are Boolean expressions that define when the program will enter single-step mode so that you can watch specific variables or expressions. Typical breakpoints are T>=100ns AND T<=10ns, or V(OUT)>5.5.

3D Windows: This lets you add or delete a 3D plot window. It is enabled only if more than one run is done.

Performance Windows: This lets you add or delete a performance plot window. It is enabled only if there is more than one run.

Numeric Output: (F5) This shows the Numeric Output window.



State Variables editor: (F12) This accesses the State Variables editor.

DSP Parameters: This opens the DSP dialog box. It lets you specify the time window lower and upper limits and the number of sample points to use for the DSP functions. See Chapter 24, "Fourier Analysis and Digital Signal Processing", for more details on DSP functions.

Reduce Data Points: This item invokes the Data Point Reduction dialog box. It lets you delete every n'th data point. This is useful when you use very small time steps to obtain very high accuracy, but do not need all of the data points produced. Once deleted, the data points cannot be recovered.

Exit Analysis: (F3) This exits the analysis.

Initialization

State variables define the state or condition of the mathematical system that represents the circuit at any instant. These variables must be initialized to some value prior to starting the analysis run. Here is how MC7 does the initialization:

Setup initialization:
When you first select a transient, AC, or DC analysis, all state variables are set to zero and all digital levels to X. This is called the *setup initialization*.

Run initialization:
Each new run evokes the *run initialization* based upon the State Variables option from the Analysis Limits dialog box. This includes every run, whether initiated by pressing F2, clicking on the Run button, stepping parameters, using Monte Carlo, or stepping temperature. There are three choices:

Zero: The analog state variables, node voltages, and inductor currents are set to 0. Digital levels are set to X, or in the case of flip-flop Q and QB outputs, set to '0', '1', or 'X' depending upon the value of the global value DIGINITSTATE. This value is defined in the Global Settings dialog box. This is the only option in DC analysis.

Read: MC7 reads the variables from the file CIRCUITNAME.TOP. The file itself is created by the State Variables editor Write command.

Leave: MC7 does nothing to the state variables. It simply leaves them alone. There are three possibilities:

First run: If the variables have not been edited with the State Variables editor, they still retain the setup initialization values.

Later run: If the variables have not been edited with the State Variables editor, they retain the ending values from the last run.

Edited: If the variables have been edited with the State Variables editor, they are the values shown in the editor.

After the State Variables option has been processed, .IC statements are processed. Device IC statements, such as those for inductor current and capacitor voltage, override .IC statements if they are in conflict.

Note that .IC statements specify values that persist throughout the initial bias point calculation.

They are more resilient than simple initial values which can (and usually do) change after the first iteration of the bias point. This may be good or bad depending upon what you are trying to achieve.

Using these initial values, an optional operating point calculation may be done and the state variables may change. If no operating point is done, the state variables are left unchanged from the initialization procedure. If an operating point only is done, the ending state variable values are equal to the DC operating point values.

Using the P key

During a simulation run, the value of the expressions for each curve can be seen by pressing the 'P' key. This key toggles the printing of the numeric values on the analysis plot adjacent to the expressions. This is a convenient way to check the course of a new, lengthy simulation when the initial plot scales are unknown. This feature may significantly slow the simulation, so only use it to "peek" at the numeric results, then toggle it off with the 'P' key.

The State Variables editor

The State Variables editor is for reviewing or editing state variables. It looks likes this:

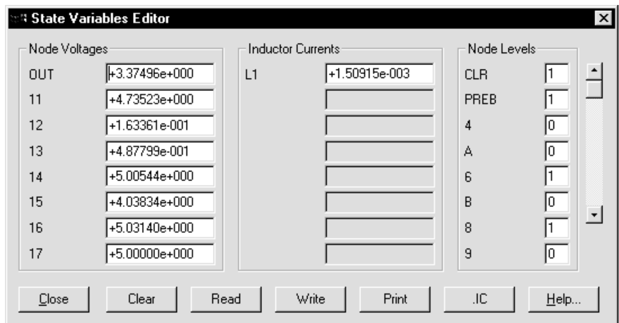


Figure 6-2 The State Variables editor

The editor displays node voltages, inductor currents, and digital node levels. The scroll bars may be used to review values not visible on the display. Any value may be edited.

The command buttons function as follows:

- **Close:** This exits the dialog box.

- **Clear:** This immediately sets all analog values to zero. Digital node levels are set to 'X'.

- **Read:** This immediately reads a new set of values from a disk file, after prompting for a file name.

- **Write:** This immediately writes the displayed values to a disk file using a user-supplied file name. This file is created for use when the Analysis Limits State Variables Read option is selected. The Read option uses the values stored in this file at the Run initialization stage.

- **Print:** This copies the values to a text file called CIRCUITNAME.SVV.

- **.IC:** This command translates the existing state variables into .IC statements and saves them in the circuit's text area. The following translations are made:

State variable	IC statement
Node voltage	.IC V(Node name) = Node voltage
Inductor current	.IC I(Inductor name) = Inductor current
Digital node state	.IC D(Digital node name) = Digital node state

The purpose of this command is to provide a handy way to create .IC statements, which some prefer as a means of initializing state variables.

It is important to note that the clear and read commands, and all manual edits result in immediate changes, as opposed to the delayed changes made by the Analysis Limits options. The Zero, Read, and Leave options from the Analysis Limits dialog box affect the values at the start of the simulation (Run initialization).

- **Help:** This accesses help topics for the State Variables editor.

Numeric output

Numeric output may be obtained for each curve by clicking the Numeric Output button in the curve row. Numeric output is of several types:

Analysis Limits Summary: This is a printout of the main analysis limits from the Analysis Limits, Stepping, and Monte Carlo dialog boxes.

Operating point information: This is a printout of the DC operating point values for each device in the circuit. It includes a selection of device currents, voltages, conductances, and capacitances.

Curve tables: This is a tabular printout of the value of each X and Y expression of each selected curve. The values are interpolated from the actual data points. The Number of Points value from the Analysis Limits dialog box determines the number of points printed in the table.

Digital warnings: These include digital hazard and constraint violations.

Output is saved in the file CIRCUITNAME.TNO and printed to the Numeric Output window, which is accessible after the run by clicking on the button. Here is a typical output window.

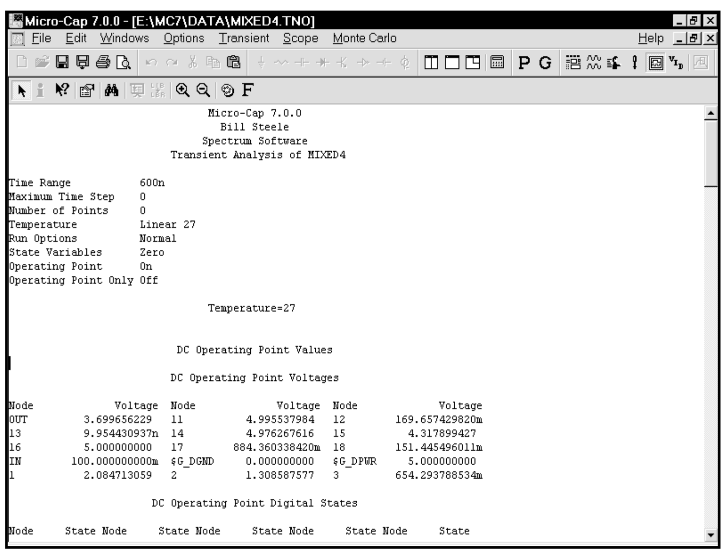


Figure 6-3 Numeric output

Chapter 7 AC Analysis

What's in this chapter

This chapter describes the AC analysis routines. AC analysis is a linear, small signal analysis. Before the main AC analysis is run, linearized small signal models are created for nonlinear components based upon the operating point bias.

Features new in Micro-Cap 7

- Smith charts
- Polar plots
- Optimizer
- Watch window
- Breakpoints window
- Range format <*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold spacing*>]

What happens in AC analysis

AC analysis is a type of small-signal or linear analysis. This means that all circuit variables are assumed to be linearly related. Double one voltage, and you double any related quantities. When you plot or print V(1) you are seeing the small-signal linear voltage between node 1 and the ground node.

Using a small-signal model of each device in the circuit, MC7 constructs a set of linear network equations and solves for every voltage and current in the circuit over the specified frequency range. The program obtains the small-signal models by linearizing the devices about the state variable values. These are usually the result of an operating point calculation, but may also result from a read, from edits in the State Variables editor, or may simply be left over from the last run.

Linearizing means replacing a device's nonlinear model with a simple constant that expresses a linear relationship between the terminal voltages and currents of the device. The linear model is assumed to hold for small signal changes about the point at which the linearization takes place, the DC operating point. Digital parts are treated as open circuits during the linearization process. For linear resistors, capacitors, and inductors, the linearized AC value is the same as the constant time-domain value. For nonlinear passive components whose value changes with bias, the linearized AC value is the time-domain resistance, capacitance, or inductance computed at the operating point. This means that in a resistor with a value expression like 1+2*V(10), the V(10) refers to the time-domain V(10), not the AC small signal V(10). If the operating point calculation produces a DC value of 2.0 volts for V(10), then the value of this resistor during AC analysis will be 1+2*2 = 5 Ohms. During the small-signal AC analysis, the resistance, capacitance, or inductance does not change. There is an exception to this rule however.

If the component's FREQ attribute is specified, then the FREQ expression *is* the AC resistance, capacitance, or inductance, and is allowed to be a function of frequency. See Chapter 22 for the specifics on the use of the FREQ attribute with resistors, capacitors, inductors, and NFV and NFI function sources.

For nonlinear components like diodes, JFETs, MOSFETs, and bipolar transistors, the conductances, capacitances, and controlled sources that comprise the AC model are obtained by partial derivatives evaluated at the operating point value. These values are also constant during AC small-signal analysis.

The only devices whose transfer function or impedance can change during AC small-signal analysis are the Z transform, Laplace function and table sources, and when their FREQ attribute is specified, resistors, capacitors, inductors, and NFV and NFI function sources. Z transform and Laplace sources use the complex frequency variable S (j*2*PI*frequency) in their transfer function, so their transfer function must change with frequency during the run.

For curve sources, the small signal model is simply an AC voltage or current source. The AC value of the source is determined from the parameter line for SPICE components and from the attribute value for schematic components.

SPICE V or I sources:
The AC magnitude value is specified as a part of the device parameter. For example, a source with the value attribute "DC 5.5 AC 2.0" has an AC magnitude of 2.0 volts.

Pulse and sine sources:
These sources have their AC magnitude fixed at 1.0 volt.

User sources: User sources provide a signal comprised of the real and imaginary parts specified in their files.

Function sources: These sources create an AC signal only if a FREQ expression is specified.

Because AC analysis is linear, it doesn't matter whether the AC amplitude of a single input source is 1 volt or 500 volts. If you are interested in relative gain from one part of the circuit to another, then plot V(OUT)/V(IN). This ratio will be the same regardless of the value of V(IN). If V(IN) is 1, then it is not necessary to plot ratios, since V(OUT)/V(IN) = V(OUT)/1 = V(OUT). If there is more than one input source with a nonzero AC value, then you can't meaningfully look at gains from one of the source nodes to another node.

The basic sequence for AC analysis is this:

1. Optionally calculate the DC operating point.
2. Compose the linear equivalent AC model for every device.
3. Construct a set of linearized circuit equations.
4. Set frequency to fmin.
5. Solve for all voltages and currents in the linearized model.
6. Plot or print requested variables.
7. Increment frequency.
8. If frequency exceeds fmax, quit, else go to step 5.

The AC Analysis Limits dialog box

The Analysis Limits dialog box is divided into five principal areas: the Command buttons, Numeric limits, Curve options, Expressions, and Options.

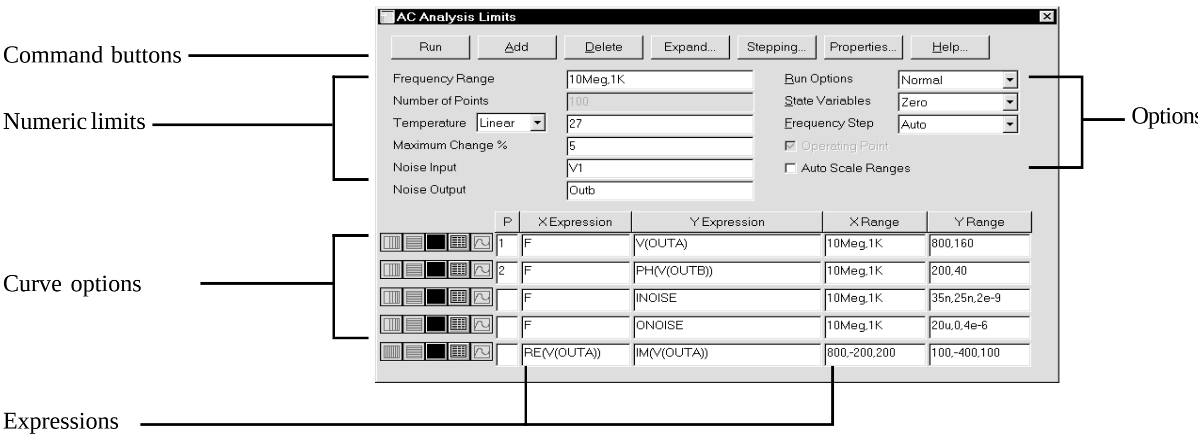
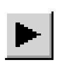


Figure 7-1 The Analysis Limits dialog box

The command buttons are located just above the Numeric limits.

Run: This command starts the analysis run. Clicking the Tool bar Run  button or pressing F2 will also start the run.

Add: This command adds another Curve options field and Expression field line after the line containing the cursor. The scroll bar to the right of the Expression field scrolls the curve rows when needed.

Delete: This command deletes the Curve option field and Expression field line where the text cursor is.

Expand: This command expands the working area for the text field where the text cursor currently is. A dialog box is provided for editing or viewing. To use the feature, click in an expression field, then click the Expand button.

Stepping: This command calls up the Stepping dialog box. Stepping is reviewed in a separate chapter.

Properties: This command invokes the Properties dialog box which lets you control the analysis plot window and the way curves are displayed.

Help: This command calls up the Help system which provides information by index and topic.

The definition of each item in the Numeric limits field is as follows:

• **Frequency Range:** This field controls the frequency range for the analysis. The syntax is <*Highest Frequency*> [, <*Lowest Frequency*>]. If <*Lowest Frequency*> is unspecified, the program calculates a single data point at <*Highest Frequency*>.

• **Number of Points:** This determines the number of data points printed in the Numeric Output window. It also determines the number of data points actually calculated if Linear or Log stepping is used. If the Auto Step method is selected, the number of points actually calculated is controlled by the <*Maximum change %*> value. If Auto is selected, interpolation is used to produce the specified number of points. The default value is 51. This number is usually set to an odd value to produce an even print interval.

For the Linear method, the frequency step and the print interval are:

(<*Highest Frequency*> - <*Lowest Frequency*>)/(<*Number of points*> - 1)

For the Log method, the frequency step is:

$$(\textit{<Highest Frequency>} \,/\, \textit{<Lowest Frequency>})^{1/(\textit{<Number of points>} - 1)}$$

• **Temperature:** This field controls the temperature of the run. If the Temperature list box shows Linear the format is:

<*high*> [, <*low*> [, <*step*>]]

The default value of <*low*> is <*high*>, and the default value of <*step*> is <*high*> - <*low*>.

If the Temperature list box shows List the format is:

<*t1*> [, <*t2*> [, <*t3*>] [,...]]

where *t1*, *t2*,.. are individual values of temperature.

All values are in degrees Celsius. One analysis is done at each specified temperature, producing one curve branch for each run.

• **Maximum Change %:** This value controls the frequency step used when Auto is selected for the Frequency Step method.

• **Noise Input:** This is the name of the input source to be used for noise calculations. If the INOISE and ONOISE variables are not used in the expression fields, this field is ignored.

• **Noise Output:** This field holds the name(s) or number(s) of the output node(s) to be used for noise calculations. If the INOISE and ONOISE variables are not used in the expression fields, this field is ignored.

The Curve options are located below the Numeric limits and to the left of the Expressions. Each curve option affects only the curve in its row. The options function as follows:

The first option toggles the X-axis between a linear and a log plot. Log plots require positive scale ranges.

The second option toggles the Y-axis between a linear and a log plot. Log plots require positive scale ranges.

The option activates the color menu. There are 64 color choices for an individual curve. The button color is the curve color.

The option prints a table showing the numeric value of the curve. The table is printed to the Output window and saved in the file CIRCUITNAME.ANO.

The option selects the basic plot type. In AC analysis there are three types available, rectangular, polar, and Smith chart.

A number from 1 to 9 in the P column is used to group the curves into different plot groups. All curves with like numbers are placed in the same plot group. If the P column is blank, the curve is not plotted.

The Expressions field specifies the horizontal (X) and vertical (Y) scale ranges and expressions. The expressions are treated as complex quantities. Some common expressions are F (frequency), db(v(1)) (voltage in decibels at node 1), and re(v(1)) (real voltage at node 1). *Note that while the expressions are evaluated as complex quantities, only the magnitude of the Y expression vs. the magnitude of the X expression is plotted.* If you plot the expression V(3)/V(2), MC7 evaluates the expression as a complex quantity, then plots the magnitude of the final result. It is not possible to plot a complex quantity directly versus frequency. You can plot the imaginary part of an expression versus its real part (Nyquist plot), or you can plot the magnitude, real, or imaginary parts versus frequency (Bode plot).

The scale ranges specify the scales to use when plotting the X and Y expressions. The range format is:

<*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold grid spacing*>]

<*low*> defaults to zero. [,<*grid spacing*>] sets the spacing between grids. [,<*bold grid spacing*>] sets the spacing between bold grids. Placing "AUTO" in the X or Y scale range calculates its range automatically. The Auto Scale Ranges option calculates scales for all ranges during the simulation run and updates the X and Y Range fields. The Auto Scale (F6) command immediately scales all curves, without changing the range values, letting you restore them with CTRL + HOME if desired. Note that <*grid spacing*> and <*bold grid spacing*> are used only on linear scales. Logarithmic scales use a natural grid spacing of 1/10 the major grid values and bold is not used. The Auto Scale command uses the **Preferences / Auto Scale Grids** value to set the grid spacing.

Clicking the *right* mouse button in the Y expression field invokes the Variables list. It lets you select variables, constants, functions, and operators, or expand the field to allow editing long expressions. Clicking the *right* mouse button in the other fields invokes a simpler menu showing suitable choices.

The Options group includes:

- **Run Options**
  - **Normal:** This runs the simulation without saving it to disk.
  - **Save:** This runs the simulation and saves it to disk.
  - **Retrieve:** This loads a previously saved simulation and plots and prints it as if it were a new run.
- **State Variables:** These options determine what happens to the time

domain state variables (DC voltages, currents, and digital states) prior to the optional operating point.

- **Zero:** This sets the state variable initial values (node voltages, inductor currents, digital states) to zero or X.

- **Read:** This reads a previously saved set of state variables and uses them as the initial values for the run.

- **Leave:** This leaves the current values of state variables alone. They retain their last values. If this is the first run, they are zero. If you have just run an analysis without returning to the Schematic editor, they are the values from that run.

- **Frequency step**
  - **Auto:** This method uses the first plot of the first group as a pilot plot. If, from one frequency point to another, the plot has a vertical change of greater than *Maximum change %* of full scale, the frequency step is reduced, otherwise it is increased. *Maximum change %* is the value from the fourth numeric field of the AC Analysis Limits dialog box. Auto is the standard.

  - **Linear:** This method produces a frequency step such that, with a linear horizontal scale, the data points are equidistant horizontally.

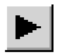
  - **Log:** This method produces a frequency step such that with a log horizontal scale, the data points are equidistant horizontally.

  - **List:** This method uses a comma-delimited list of frequency points from the Frequency Range, as in 1E8, 1E7, 5E6.

- **Operating Point:** This calculates a DC operating point, changing the time-domain state variables as a result. If an operating point is not done, the time domain variables are those resulting from the initialization step (zero, leave, or read). The linearization is done using the state variables after this optional operating point. In a nonlinear circuit where no operating point is done, the validity of the small-signal analysis depends upon the accuracy of the state variables read from disk, edited manually, or enforced by device initial values or .IC statements.

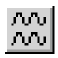
- **Auto Scale Ranges:** This sets all ranges to Auto every time a simulation is run. If disabled, the values from the range fields are used.

The AC menu

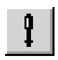


Run: (F2) This starts the analysis run.

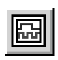
Limits: (F9) This accesses the Analysis Limits dialog box.



Stepping: (F11) This accesses the Stepping dialog box.



Optimize: (CTRL + F11) This accesses the Optimize dialog box.



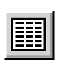
Analysis Window: (F4) This command displays the analysis plot.

Watch: (CTRL + W) This displays the Watch window where you define expressions or variables to watch during a breakpoint invocation.

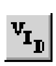
Breakpoints: (ALT + F9) This accesses the Breakpoints dialog box. Breakpoints are Boolean expressions that define when the program will enter single-step mode so that you can watch specific variables or expressions. Typical breakpoints are F>=10Meg AND F<=110Meg, or V(OUT)>2.

3D Windows: This lets you add or delete a 3D plot window. It is enabled only if more than one run is done.

Performance Windows: This lets you add or delete a performance plot window. It is enabled only if there is more than one run.



Numeric Output: (F5) This shows the Numeric Output window.



State Variables editor: (F12) This accesses the State Variables editor.

DSP Parameters: This opens the DSP dialog box. It lets you specify the frequency window lower and upper limits and the number of sample points to use for the DSP functions. See Chapter 24, "Fourier Analysis and Digital Signal Processing", for more details on DSP functions.

Reduce Data Points: This item invokes the Data Point Reduction dialog box. It lets you delete every n'th data point. This is useful when you use very small time steps to obtain very high accuracy, but do not need all of the data points produced. Once deleted, the data points cannot be recovered.

Exit Analysis: (F3) This exits the analysis.

Numeric output

Numeric output may be obtained for the X and Y expressions of each curve by clicking the button in the curve row. Numeric output includes:

> *Analysis Limits Summary:* This is a printout of the main analysis limits from the Analysis Limits, Stepping, and Monte Carlo dialog boxes.
>
> *Operating point information:* This is a printout of the DC operating point values for each device in the circuit. It includes a selection of device currents, voltages, conductances, and capacitances.
>
> *Curve tables:* These are tabular printouts of the value of each X and Y expression of selected curves. Duplicated X expressions (like F for frequency) are eliminated.

Output is saved to the text file CIRCUITNAME.ANO and printed to the window, which is accessible after the run by clicking on the button. Here is a typical result.

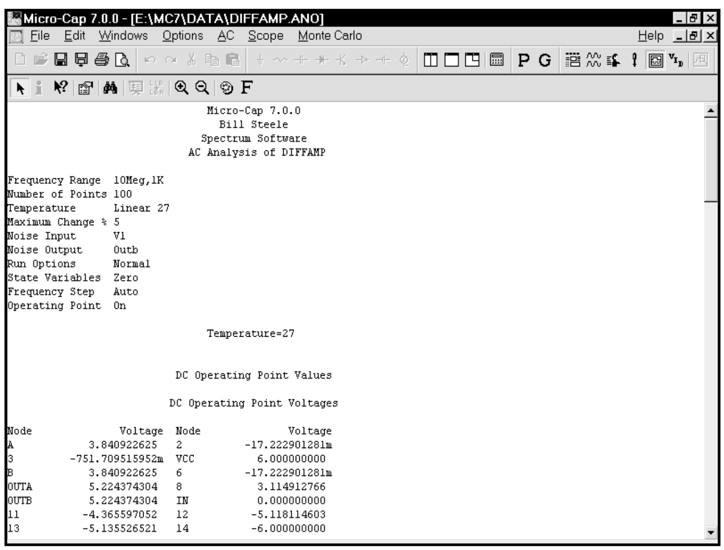


Figure 7-2 Numeric output

Noise

MC7, like SPICE, models three types of noise:

- Thermal noise
- Shot noise
- Flicker noise

Thermal noise, produced by the random thermal motion of electrons, is always associated with resistance. Discrete resistors and the parasitic resistance of active devices contribute thermal noise.

Shot noise is caused by a random variation in current, usually due to recombination and injection. Normally, shot noise is associated with the dependent current sources used in the active device models. All semiconductor devices generate shot noise.

Flicker noise originates from a variety of sources. In BJTs, the source is normally contamination traps and other crystal defects that randomly release their captured carriers.

Noise analysis measures the contribution from all of these noise sources as seen at the input and output. Output noise is calculated across the output node(s) specified in the Noise Output field. To plot or print it, specify 'ONOISE' as the Y expression. Similarly, input noise is calculated across the same output nodes, but is divided by the gain from the input node to the output node. To plot or print input noise, specify "INOISE" as the Y expression.

Because the network equations are structured differently for noise, it is not possible to simultaneously plot noise variables and other types of variables such as voltage and current. If you attempt to plot both noise and other variables, the program will issue an error message.

Because noise is an essentially random process, there is no phase information. Noise is defined as an RMS quantity so the phase (PH) and group delay (GD) operators should not be used.

Noise is measured in units of Volts / $Hz^{1/2}$.

AC analysis tips

Here are some useful things to remember for AC analysis:

Unexpected output when bypassing the operating point:
If your circuit is producing strange results and you are not doing an operating point, make sure that the initial conditions you've supplied are correct. You must supply all node voltages, inductor currents, and digital states if you skip the operating point in order to get meaningful results.

Zero output:
If your circuit is producing zero voltages and currents it is probably because the sources are absent or have zero AC magnitudes. Only the PULSE and SIN sources have a nonzero 1.0 default AC magnitude. The User source supplies what its file specifies. Function sources generate an AC excitation equal to the value of their FREQ expression, if any. The other sources either have no AC magnitude at all or else have a 0.0 volt default AC magnitude.

Simultaneous Auto Frequency step and Auto Scale:
If you select both Auto frequency stepping and Auto Scale, the first plot will probably be coarse. Auto frequency stepping uses the plot scale employed during the run to make dynamic decisions about a suitable frequency step. If Auto Scale is in effect, the plot scale employed during the run is preset to a very coarse scale. The frequency steps are not referenced to the actual range of the curve, since it is only known after the run. The solution is to make two runs, and turn off the Auto Scale option after the first run. The second run will have the benefit of knowing the true range of the curve and will produce a smoother plot.

Flat curves:
If your circuit is a very narrow band reject filter and your gain plot is flat, it is probably because the frequency step control is not sampling in the notch. This can happen when the sweep starts at a frequency far to the left of the notch. Since the plot is very flat, the frequency step is quickly increased to the maximum. By the time the frequency nears the notch, the step exceeds the notch width and the sweep jumps over the notch. This is like a car moving so fast that is doesn't notice a narrow pothole that would have been apparent at a lower speed. The solution is to set the *fmin* much nearer the expected notch, or use one of the fixed frequency step methods. If you use a fixed step, make sure the frequency step is smaller than the notch. This can be

accomplished by increasing the Number of Points value or decreasing the frequency range and centering it near the expected notch frequency.

Plotting reactive power:
To plot reactive power through a capacitor C1, plot V(C1)*I(C1). To plot reactive power through an inductor L1, plot V(L1)*I(L1). For a port whose input nodes are A and B, plot (V(A,B))*I(V1), where V1 is a zero-volt independent voltage source in series with one of the port leads.

Using the P key

During an AC analysis run, the value of the expressions for each curve can be seen by pressing the 'P' key. This key toggles the printing of the numeric values on the analysis plot adjacent to the expressions. This is a convenient way to check the course of a new, lengthy simulation when the initial plot scales are unknown. This feature may significantly slow the simulation, so only use it to "peek" at the numeric results, then toggle it off with the 'P' key.

Chapter 8 DC Analysis

What's in this chapter

This chapter describes the features of DC analysis. DC is an acronym for direct current, one of two competing power transmission strategies early in the development of electrical engineering technology. For our purposes, DC means simply that the sources are constant and do not vary with time.

During a DC analysis, capacitors are open-circuited, inductors are short-circuited, and sources are set to their time-zero values. DC analysis sweeps one or two input independent variables (such as current or voltage source values) over a specified range. At each voltage or current step, MC7 performs an operating point calculation.

Transfer characteristics are a typical application for DC analysis.

Features new in Micro-Cap 7

- Optimizer
- Watch window
- Breakpoints window
- Range format <*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold spacing*>].

The DC Analysis Limits dialog box

The Analysis Limits dialog box is divided into five major areas: the Command buttons, Numeric limits, Curve options, Expression fields, and Options.

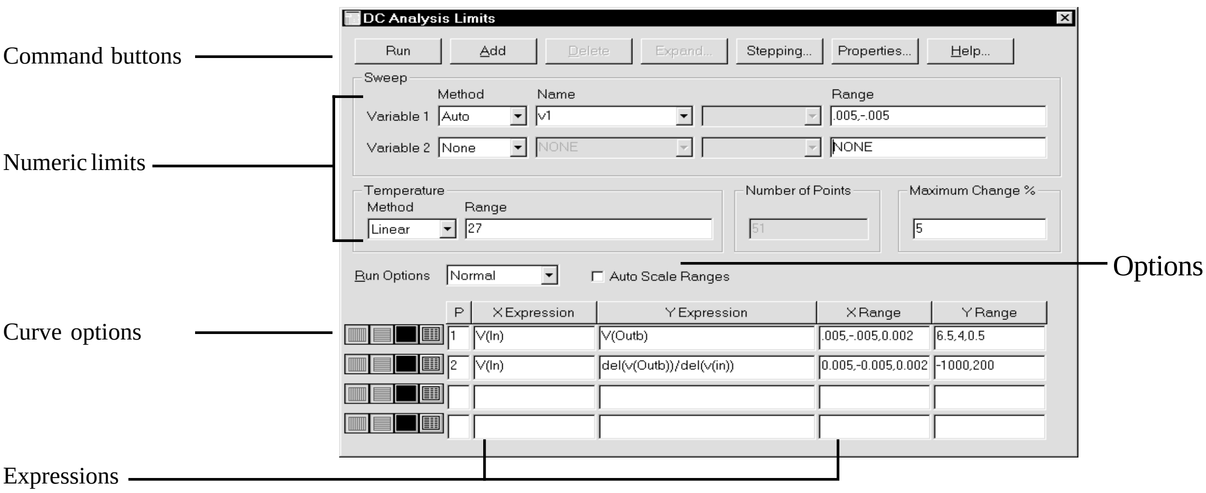


Figure 8-1 The Analysis Limits dialog box

Command buttons provide these commands.

Run: This command starts the analysis run. Clicking the Tool bar Run button or pressing F2 will also start the run.

Add: This command adds another Curve options field and Expressions field line after the line containing the cursor. The scroll bar to the right of the Expressions field scrolls through the curves when needed.

Delete: This command deletes the Curve option field and Expressions field line where the text cursor is.

Expand: This command expands the working area for the text field where the text cursor currently is. A dialog box is provided for editing or viewing. To use the feature, click in an expression field, then click the Expand button.

Stepping: This command calls up the Stepping dialog box. Stepping is reviewed in a separate chapter.

Properties: This command invokes the Properties dialog box which lets you control the analysis plot window and the way curves are displayed.

Help: This command calls up the Help system.

The definition of each field in the Numeric limits are as follows:

• **Variable 1:** This row specifies the Method, Name, and Range fields for variable 1. Its value is usually plotted along the X axis. Each value produces a minimum of one data point per curve. There are four column fields for this variable.

• **Method:** This field specifies one of four methods for stepping the variable: Auto, Linear, Log, or List.

• **Auto:** In Auto mode the rate of step size is adjusted to keep the point-to-point change less than Maximum Change % value.

• **Linear:** This mode uses the following syntax from the Range column for this row:

<*end*> [,<*start*> [,<*step*>]]

Start defaults to 0.0. *Step* defaults to (*start* - *end*)/50. Variable 1 starts at *start*. Subsequent values are computed by adding *step* until *end* is reached.

• **Log:** Log mode uses the following syntax from the Range column for this row:

<*end*> [,<*start*> [,<*step*>]]

Start defaults to *end/10*. *Step* defaults to *exp(ln(end/start)*/10). Variable 1 starts at *start*. Subsequent values are computed by multiplying by *step* until *end* is reached.

• **List:** List mode uses the following syntax from the Range column for this row:

<*v1*> [,<*v2*> [,<*v3*>] ...[,<*vn*>]]

The variable is simply set to each of the values *v1, v2, .. vn.*

• **Name:** This field specifies the name of variable 1. The variable itself may be a source value, temperature, a model parameter, or a symbolic parameter (one created with a .DEFINE statement). Model parameter stepping requires both a model name and a model parameter name.

• **Range:** This field specifies the numeric range for the variable. The range syntax depends upon the Method field described above.

• **Variable 2:** This row specifies the Method, Name, and Range fields for variable 2. The syntax is the same as for variable 1, except that the stepping options include None and exclude Auto. *Step* defaults to (*start* - *end*)/10. Each value of variable 2 produces a separte branch of the curve.

• **Temperature:** This controls the temperature of the run. The fields are:

• **Method:** This field specifies one of two methods for stepping the temperature: Linear or List.

• **Linear:** This mode uses the following syntax from the Range column for this row:

<*end*> [,<*start*> [,<*step*>]]

Start defaults to *end*. *Step* defaults to *start* - *end*. Temperature begins at *start*. Subsequent values are computed by adding *step* until *end* is reached.

• **List:** List mode uses the following syntax from the Range column for this row:

<*v1*> [,<*v2*> [,<*v3*>] ...[,<*vn*>]]

Temperature is simply set to each of the values *v1, v2, .. vn.*

One run is produced for each temperature. *Note that when temperature is selected as one of the stepped variables (variable1 or variable 2) this field is not available.*

• **Range:** This field specifies the range for the temperature variable. The range syntax depends upon the Method field described above.

- **Number of Points:** This is the number of data points to be interpolated and printed if numeric output is requested. This number defaults to 51 and is often set to an odd value to produce an even print interval. Its value is:

  (<*final1*> - <*initial1*>) /(<*Number of points*> - 1)

  <*Number of points*> values are printed for each value of the Variable 2 source.

- **Maximum Change %:** This value is only used if Auto is selected as the step method for Variable 1.

The Curve options are located below the Numeric limits and to the left of the Expressions. Curve options affect the curve in the same row. The definition of each option is as follows:

The first option toggles the X-axis between a linear and a log plot. Log plots require positive scale ranges.

The second option toggles the Y-axis between a linear and a log plot. Log plots require positive scale ranges.

The option activates the color menu. There are 64 color choices for an individual curve. The button color is the curve color.

The option prints a table showing the numeric value of the curve. The number of values printed is set by the Number of Points value. The table is printed to the Output window and saved in the file CIRCUITNAME.DNO.

A number from 1 to 9 in the Plot (P) column places the curve into a plot group. All curves with like numbers are placed in the same plot group. If the P column is blank, the curve is not plotted.

The Expressions field is used to specify the horizontal (X) and vertical (Y) scale ranges and expressions. Some common expressions used are VCE(Q1) (collector to emitter voltage of transistor Q1) or IB(Q1) (base current of transistor Q1).

The X Range and Y Range fields specify the scales to be used when plotting the X and Y expressions.

The range format is:

<*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold grid spacing*>]

<*low*> defaults to zero. [,<*grid spacing*>] sets the spacing between grids. [,<*bold grid spacing*>] sets the spacing between bold grids. Placing "AUTO" in the scale range calculates that individual range automatically. The Auto Scale Ranges option calculates scales for all ranges during the simulation run and updates the X and Y Range fields. The Auto Scale (F6) command immediately scales all curves, without changing the range values, letting you restore them with CTRL + HOME if desired. Note that <*grid spacing*> and <*bold grid spacing*> are used only on linear scales. Logarithmic scales use a natural grid spacing of 1/10 the major grid values and bold is not used. The Auto Scale command uses the **Preferences / Auto Scale Grids** value to set the grid spacing.

Clicking the *right* mouse button in the Y expression field invokes the Variables list which lets you select variables, constants, functions, and operators, or expand the field to allow editing long expressions. Clicking the *right* mouse button in the other fields invokes a simpler menu showing suitable choices.

The Options area is below the Numeric limits. The Auto Scale Ranges option has a check box. Clicking the mouse in the box will toggle the options on or off with an X in the box showing that the option is enabled.

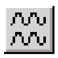
The options available from here are:

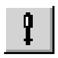
- **Run Options**
  - **Normal:** This runs the simulation without saving it to disk.
  - **Save:** This runs the simulation and saves it to disk.
  - **Retrieve:** This loads a previously saved simulation and plots and prints it as if it were a new run.
- **Auto Scale Ranges:** This sets the X and Y range to auto every time a simulation is run. If it is disabled, the values from the range fields will be used.

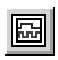
The DC menu


Run: (F2) This starts the analysis run.


Limits: (F9) This accesses the Analysis Limits dialog box.


Stepping: (F11) This accesses the Stepping dialog box.


Optimize: (CTRL + F11) This accesses the Optimize dialog box.


Analysis Window: (F4) This command displays the analysis plot.

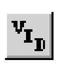
Watch: (CTRL + W) This displays the Watch window where you define expressions or variables to watch during a breakpoint invocation.

Breakpoints: (ALT + F9) This accesses the Breakpoints dialog box. Breakpoints are Boolean expressions that define when the program will enter single-step mode so that you can watch specific variables or expressions. Typical breakpoints are V(A)>=1 AND V(A)<=1.5, and V(OUT)>5.5.

3D Windows: This lets you add or delete a 3D plot window. It is enabled only if more than one run is done.

Performance Windows: This lets you add or delete a performance plot window. It is enabled only if there is more than one run.


Numeric Output: (F5) This shows the Numeric Output window.


State Variables editor: (F12) This accesses the State Variables editor.

Reduce Data Points: This item invokes the Data Point Reduction dialog box. It lets you delete every n'th data point. This is useful when you use very small time steps to obtain very high accuracy, but do not need all of the data points produced. Once deleted, the data points cannot be recovered.

Exit Analysis: (F3) This exits the analysis.

Numeric output

Numeric output may be obtained for the X and Y expressions of each curve by clicking the Numeric Output button  in the curve row. Numeric output includes:

Analysis Limits Summary: This is a printout of the main analysis limits from the Analysis Limits, Stepping, and Monte Carlo dialog boxes.

Expression tables: This is a tabular printout of the value of each X and Y expression of each selected curve. Duplicated X expressions are eliminated. The values are interpolated from the actual data points. The Number of Points value from the Analysis Limits dialog box determines the number of points printed in the table.

Output is saved to the text file CIRCUITNAME.DNO and printed to the Numeric Output window. This window is accessible after the run from the DC menu. Here is a typical numeric output window.

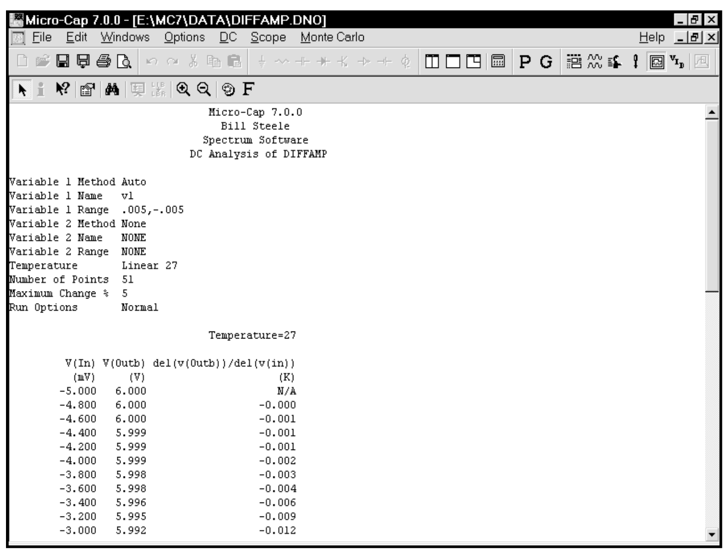


Figure 8-2 Numeric output

Using the P key

During a DC analysis run, the value of the expressions for each curve can be seen by pressing the 'P' key. This key toggles the printing of the numeric values on the analysis plot adjacent to the expressions. This is a convenient way to check the course of a new, lengthy simulation when the initial plot scales are unknown. This feature may significantly slow the simulation, so only use it to "peek" at the numeric results, then toggle it off with the 'P' key.

Troubleshooting tips

Here are some useful things to remember for DC analysis:

IV curves:
To generate IV curves for a device, place a voltage source across the output leads and sweep its voltage using the Variable 1 source. Place a voltage or current source at the base or gate and step its voltage or current with the Variable 2 source. Look at the IVBJT sample circuit for an example of how to do this.

No convergence during a sweep:
DC analysis is the most difficult of all the analysis modes, because it does not benefit from the converging effects of capacitors and inductors, as transient analysis does. If convergence fails, especially during a sweep, try changing the starting and/or step value to avoid the problem area.

Chapter 9 Dynamic DC Analysis

What's in this chapter

This chapter describes the features of Dynamic DC analysis. This analysis mode is designed to dynamically display the DC voltage, current, power, and device conditions, as the circuit responds to user changes.

What happens in Dynamic DC analysis

Dynamic DC is an interactive process in which the user modifies the circuit and the program calculates the DC response immediately and displays one or more measures of the DC state. The process looks like this:

- User modifies the circuit
- MC7 finds the DC solution
- The schematic display is updated

The schematic has four optional display buttons for displaying the circuit's time domain quantities. Each can be individually enabled.

 Voltages/states

 Device pin currents

 Device power

 Device condition (ON, OFF, SAT, LIN, etc.)

When the Dynamic DC mode is invoked, the Voltages/states button is enabled, so that, at a minimum, the schematic shows these values. To display current, power, or condition you must click these buttons separately.

You can make any kind of change to the schematic and the program will respond by calculating the new DC state. You can rewire, add or delete components, change parameter values or any edit you wish and the display will show the updated values.

The battery, V source, I source, and resistor values can be adjusted in one of two ways:

- By selecting a device and dragging the slider which appears next to it. The presence of the slider can be enabled or disabled from the Preferences dialog box (SHIFT + CTRL + P). By default it is off. The attributes SLIDER_MIN and SLIDER_MAX control the slider range for the part. When a part is first placed in a schematic or edited, its SLIDER_MIN attribute is set to zero and its SLIDER_MAX attribute is set equal to the VALUE attribute.

- By selecting a device (clicking on it) you can use the UP ARROW and DOWN ARROW keys to increase or decrease the source value. One or more parts may be selected for simultaneous manipulation in this way. Each key press changes the VALUE attribute by a convenient portion of the difference between SLIDER_MAX and SLIDER_MIN.

Note that the display buttons show the indicated quantities for all analysis modes, not just Dynamic DC. After a transient analysis, the buttons display the ending conditions of the transient analysis, which typically is not the DC operating point but the last transient analysis time point. It would be the result of the transient analysis operating point if the Operating Point Only option has been selected.

After an AC analysis, the buttons show the results of the last DC operating point performed during the analysis, if one was performed. The DC operating point is recommended but optional in AC.

After a DC analysis, the buttons show the results of the last DC sweep point.

When the Dynamic DC mode is exited, either by deselecting the Dynamic DC button on the Analysis menu, or by selecting another analysis type from that menu, the program lets you optionally restore the circuit edits made during the Dynamic DC.

Note that if you drag a component away from a circuit and drop it in an empty space, MC7 will normally object during the analysis setup since this creates one or more nodes with no DC path to ground. Since this is likely to temporarily occur during Dynamic DC,

MC7 enables the Add DC Path to Ground option during Dynamic DC.

After the Dynamic DC mode is exited, the option's original status is restored.

A sample of Dynamic DC analysis

To illustrate Dynamic DC analysis, load the file TTLINV. Select **Dynamic DC** from the **Analysis** menu. The display should look like this:

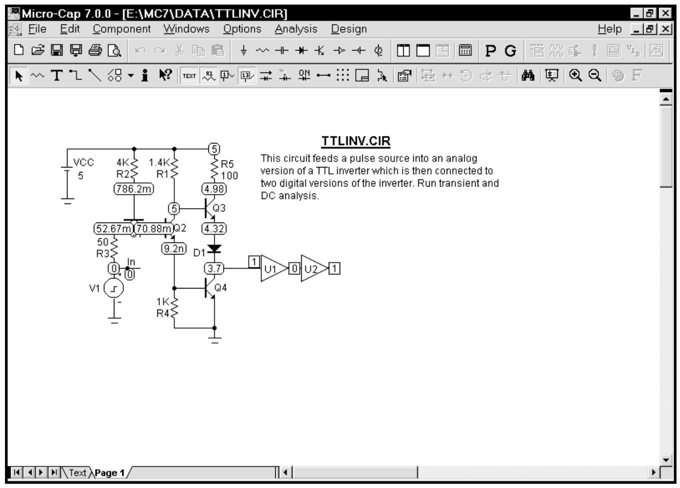


Figure 9-1 Display of initial node voltages

MC7 solved for the DC voltages in the circuit and displayed the voltages and digital states on the screen. Select the battery by clicking on it, and press the DOWN ARROW key. Each time you press the key, the battery voltage goes down by .1 volts so that by the seventeenth key press, the display looks like this:

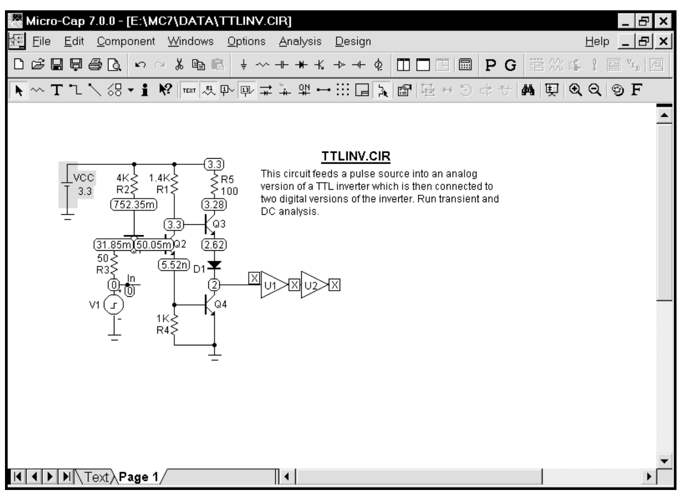


Figure 9-2 Display of voltages at the switching point

The battery voltage has declined to about 3.3 volts and the output of the analog stage and has fallen to about 2 volts. The digital states have all changed to X. The dynamic control can be used to adjust DC voltages precisely to show the switching point of the circuit.

Now disable the voltage button, and enable the button. This produces a display like this:

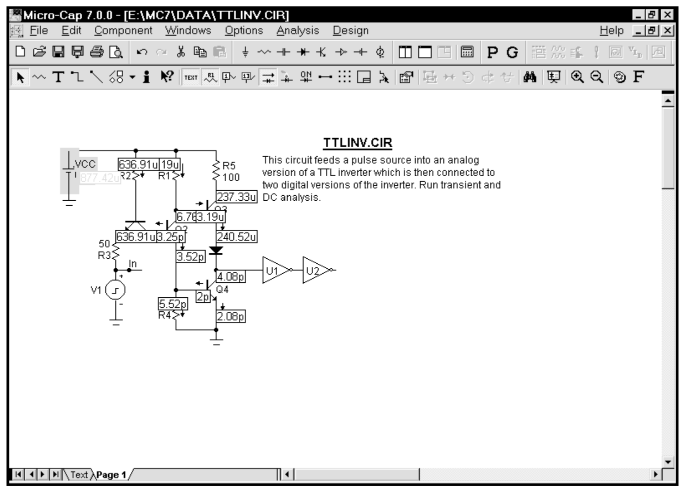
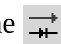


Figure 9-3 Display of device currents

The display now shows each of the device pin currents. Redundant currents have been removed for clarity.

The display shows voltages, currents, and power values using a numeric format specified in **Preferences / Format / Schematic Voltages/Current/Power**. You can change this format to display more digits if you like. The numeric format is explained in Chapter 2. More digits produce more numbers on the screen and often less clarity, so there is a trade-off.

You can also display power terms. Depending upon the device, there may be generated power, dissipated power, or stored power. In general, active devices have both stored and dissipated power terms. In a DC operating point calculation, however, the stored power term will be zero. Sources generally have only generated power terms.

To see what the power displays look like, disable the  button, and enable the  button. This produces a display like this:

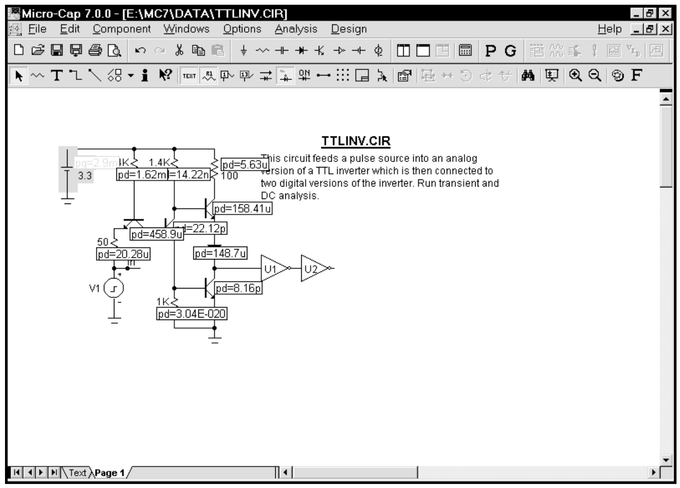


Figure 9-4 Display of device power terms

Finally, to see what the condition display looks like, disable the  button, and enable the  button. This produces a display like this:

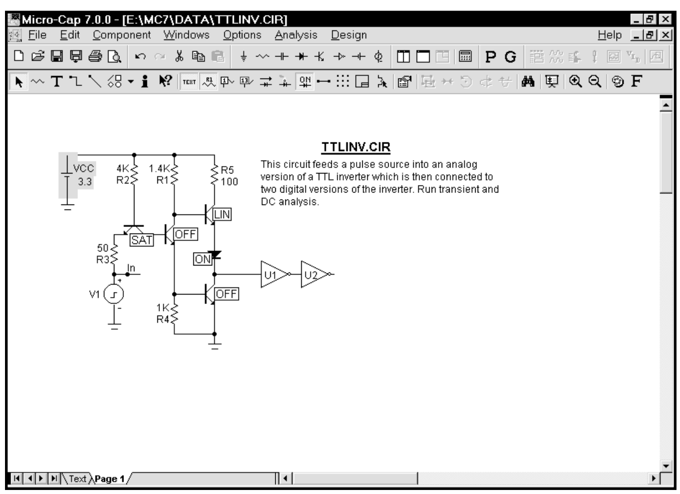


Figure 9-5 Display of device conditions

Chapter 10 Transfer Function Analysis

What's in this chapter

This chapter describes the features of Transfer Function analysis. This analysis mode is designed to calculate the DC transfer function from a specified input source to a specified output expression.

The principal topics described in this chapter include:

- What happens in Transfer Function analysis
- The Transfer Function analysis dialog box
- A Sample of Transfer Function analysis

What happens in transfer function analysis

Transfer function analysis calculates the small-signal DC transfer function from a specified input source to a specified output expression. Depending upon the input source and the output expression the transfer function calculated can be:

- Voltage gain: Input voltage source and output expression = V(OUT)
- Current gain: Input current source and output expression = I(RL)
- Transconductance: Input voltage source and output expression = I(RL)
- Transadmittance: Input current source and output expression = V(OUT)

This analysis mode also calculates the small-signal input and output impedances.

To measure the transfer function, the program makes a very small change in the input source DC value and measures the resulting change in the specified output expression value. The ratio of these two quantities produces the transfer function.

To measure the input impedance, the program makes a very small change in the input source DC value and measures the resulting change in the input current or voltage value. The ratio of these changes produces the input impedance.

To measure the output impedance, the program first adds a test voltage source across the node set implicit in the output expression. For example, an output expression such as "V(10,20)" would result in a voltage source between the nodes 10 and 20. If the output expression has no implicit output node set specified, as would be true with an expression like "IB(Q1)", the output impedance will not be calculated and the result N/A (meaning not available) supplied for the answer. If the output nodes fall across a battery, inductor, or other voltage-defined device, the answer 0.0 will be returned, since the DC resistances of these are all zero. Finally a small DC change is made in the output test source and the resulting change in its current noted. The ratio of these two quantities produces the output impedance result.

The Transfer Function Analysis Limits dialog box

To illustrate how this type of analysis works, load the file DIFFAMP. Select **Transfer Function** from the **Analysis** menu. The dialog box looks like this:

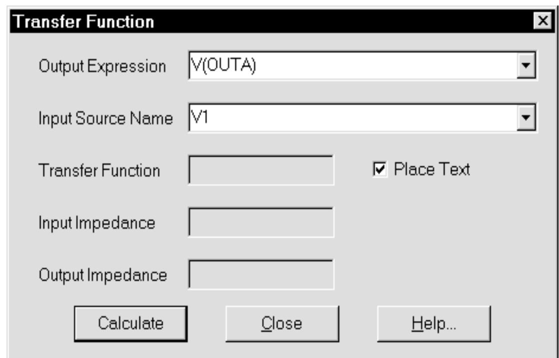


Figure 10-1 The Transfer Function dialog box

The dialog box provides the following input fields:

- Output Expression: This is where you specify the desired output expression. It can be any legal expression involving any number of DC time-domain variables and functions. Usually it's a simple expression like V(A,B) or I(R1).

- Input Source Name: This is the part name of the input source.

The results are placed in these fields:

- Transfer Function: This is the field where the program places the result of the transfer function calculation.

- Input Impedance: This is the field where the program places the result of the input impedance calculation.

- Output Impedance: This is the field where the program places the result of the output impedance calculation.

There is also a field called "Place Text". If this box is checked, then the numeric results are placed into the schematic as grid text.

A sample of transfer function analysis

To illustrate, make sure the Place Text option is selected, then click on the Calculate button. The display should look like this:

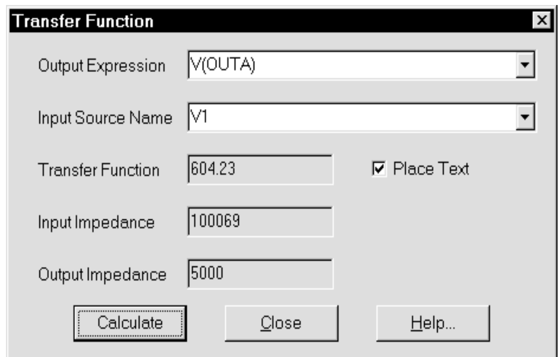


Figure 10-2 The analysis results

MC7 solved for the transfer function and impedances and printed the results in the dialog box. If you run DC analysis and manually calculate the small-signal DC transfer function, you will get the same result, though it will take longer to set up. Click on the Close button. Notice that the program added a piece of grid text describing the results of the transfer function analysis to the schematic

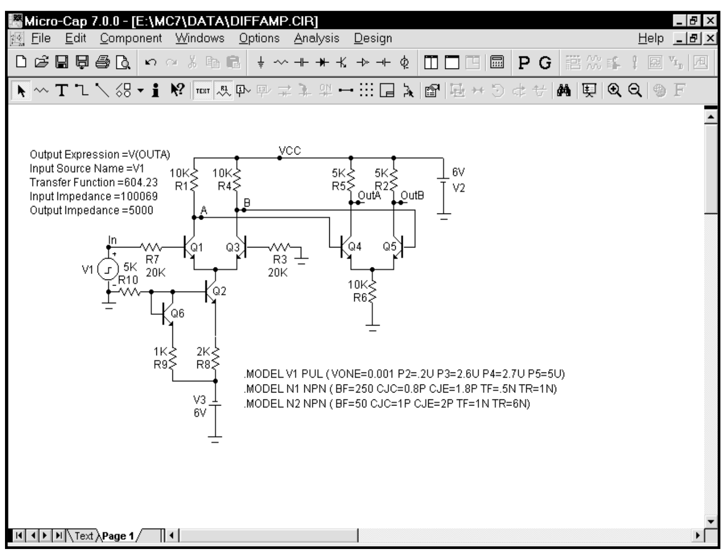


Figure 10-3 The annotated schematic

Chapter 11 Sensitivity Analysis

What's in this chapter

This chapter describes the features of Sensitivity analysis. This analysis mode calculates the DC sensitivity of one or more output expressions to one or more circuit parameters.

The principal topics described in this chapter include:

- What happens in Sensitivity analysis
- The Sensitivity analysis dialog box
- A Sample of Sensitivity analysis

What happens in sensitivity analysis

Sensitivity analysis calculates the small-signal DC sensitivity of one or more output expressions to one or more input variables. Sensitivity is defined as:

Change in an output expression / Small change in an input variable

The "Small change" part is important, as the intention is to approximate the value of the derivative at the nominal operating point. Accordingly, a change of 1E-6*Value, or 1U if the value is zero, is used.

Sensitivity analysis is like transfer function analysis, except that it calculates the sensitivity of almost any DC expression to any variable that can be stepped. Transfer function analysis, by contrast, only calculates the DC sensitivity of expressions to the value of input DC source values.

Sensitivity analysis can calculate a great many quantities depending upon the choices made in the dialog box. You can choose one input parameter or you can choose many. If you choose many, and opt for all parameters, the program may grind away for a long time, so choose wisely. If you select the All On and Model options, for example, each MOSFET level 1-3 model will require 51 operating point calculations, and the MOSFET level 5-8 will require several hundred.

The Sensitivity Analysis Limits dialog box

To illustrate how this type of analysis works, load the circuit DIFFAMP. Select **Sensitivity** from the **Analysis** menu. The dialog box looks like this:

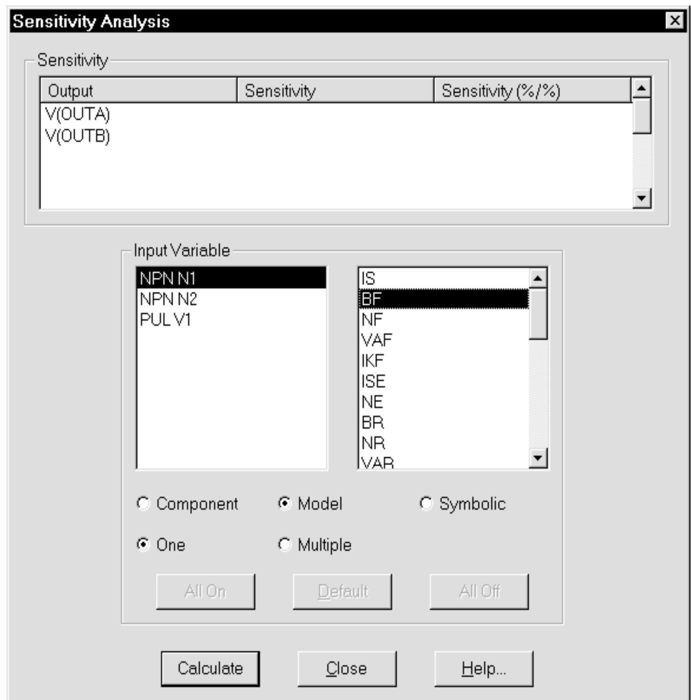


Figure 11-1 The Sensitivity dialog box

The dialog box provides the following input fields:

- Sensitivity group: This is where you specify the desired output expressions and it is also where the sensitivity answer is returned. There are three fields:
  - Output: This is where you specify one or more output expressions. Each expression is placed on a new line. To edit an existing expression click on it. To add a new output, click on a blank line and type in the new expression.
  - Sensitivity: This is the absolute sensitivity expressed as a pure ratio.

The result is printed here if only one input variable is selected. If more than one parameter is selected, then the results are placed in a text output file called CIRCUITNAME.SEN.

- Sensitivity (%/%): This is sensitivity expressed as the percentage change in the output expression divided by the percentage change in the input parameter.

- Input Variable group: This group is used to specify the input parameter. The fields are the same as in the Stepping dialog box, as the same parameters are available in both cases. There are several buttons that select the input parameter(s):

  - Component: This specifies a single instance of a part's parameter.

  - Model: This specifies a model parameter, affecting all parts that use the model name.

  - Symbolic: This specifies a symbolic parameter (one created with a .define statement)

  - One: This selects a single parameter for testing.

  - Multiple: This selects multiple parameters for testing. Specifically, it specifies all those parameters shown as selected in the Input Variable group.

  To specify which parameters to test when doing multiple input parameters, you can use the buttons as follows:

    - All On: This selects all parameters for all devices, creating lots of data.

    - Default: This selects a special subset of parameters for all devices. The set has been chosen to reflect common usage.

    - All Off: This deselects all parameters for all devices.

  You can manually select the parameters you want by using the CTRL + click method to select a list of preferred parameters.

To start the sensitivity calculation use the Calculate button. The Close button closes the dialog box without saving any changes.

Select NPN N1 and its BF parameter. Click on the Calculate button and the dialog box shows the results.

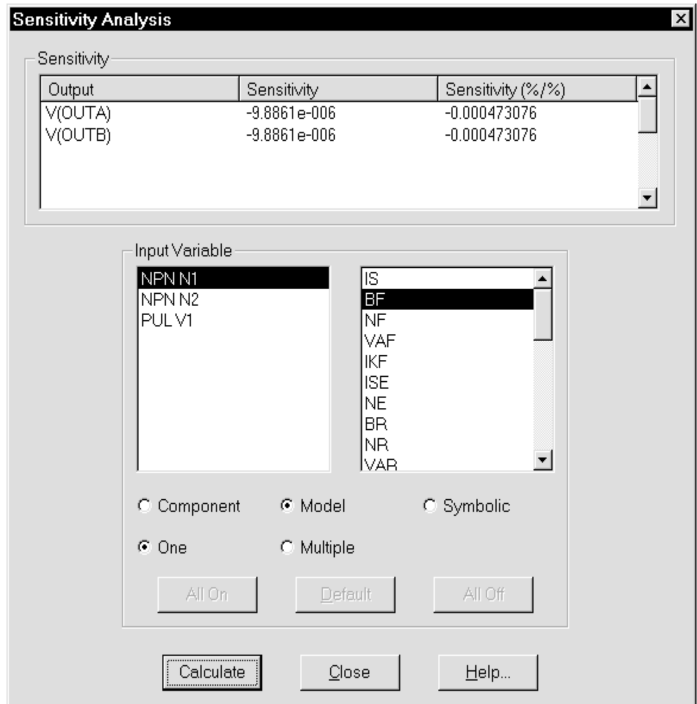


Figure 11-2 The Sensitivity dialog box

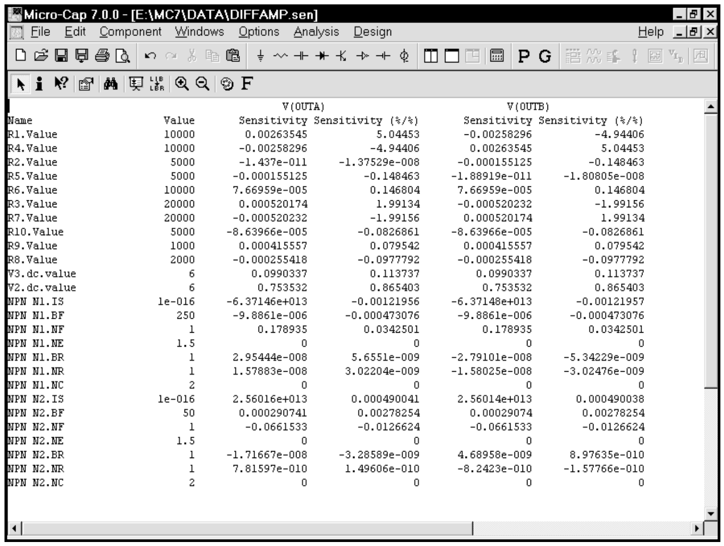
The Sensitivity fields show the raw and percentage relative sensitivity of V(OUTA) and V(OUTB) to the BF parameter.

Because we elected to calculate for only one parameter, the results are shown in the dialog box.

To see the effect of doing sensitivity for many parameters, click on the Model and Multiple options and the Default button.

Click on the Calculate button and the program computes the sensitivity of the two output expressions to the default model parameters and presents the results in a text page called DIFFAMP.SEN. It looks like Figure 11-3.

The file shows the two sensitivity measures for each checked model parameter in a tabular format as show below:



		V(OUTA)		V(OUTB)	
Name	Value	Sensitivity	Sensitivity (%/%)	Sensitivity	Sensitivity (%/%)
R1.Value	10000	0.00263545	5.04453	-0.00258296	-4.94406
R4.Value	10000	-0.00258296	-4.94406	0.00263545	5.04453
R2.Value	5000	-1.437e-011	-1.37529e-008	-0.000155125	-0.148463
R5.Value	5000	-0.000155125	-0.148463	-1.88919e-011	-1.80805e-008
R6.Value	10000	7.66959e-005	0.146804	7.66959e-005	0.146804
R3.Value	20000	0.000520174	1.99134	-0.000520232	-1.99156
R7.Value	20000	-0.000520232	-1.99156	0.000520174	1.99134
R10.Value	5000	-8.63966e-005	-0.0826861	-8.63966e-005	-0.0826861
R9.Value	1000	0.000415557	0.079542	0.000415557	0.079542
R8.Value	2000	-0.000255418	-0.0977792	-0.000255418	-0.0977792
V3.dc.value	6	0.0990337	0.113737	0.0990337	0.113737
V2.dc.value	6	0.753532	0.865403	0.753532	0.865403
NPN N1.IS	1e-016	-6.37146e+013	-0.00121956	-6.37148e+013	-0.00121957
NPN N1.BF	250	-9.8861e-006	-0.000473076	-9.8861e-006	-0.000473076
NPN N1.NF	1	0.178935	0.0342501	0.178935	0.0342501
NPN N1.NE	1.5	0	0	0	0
NPN N1.BR	1	2.95444e-008	5.6551e-009	-2.79101e-008	-5.34229e-009
NPN N1.NR	1	1.57883e-008	3.02204e-009	-1.58025e-008	-3.02476e-009
NPN N1.NC	2	0	0	0	0
NPN N2.IS	1e-016	2.56016e+013	0.000490041	2.56014e+013	0.000490038
NPN N2.BF	50	0.000290741	0.00278254	0.00029074	0.00278254
NPN N2.NF	1	-0.0661533	-0.0126624	-0.0661533	-0.0126624
NPN N2.NE	1.5	0	0	0	0
NPN N2.BR	1	-1.71667e-008	-3.28589e-009	4.68958e-009	8.97635e-010
NPN N2.NR	1	7.81597e-010	1.49606e-010	-8.2423e-010	-1.57766e-010
NPN N2.NC	2	0	0	0	0

Figure 11-3 Multiple parameter sensitivity results

Chapter 12 Filter Design

What's in this chapter

MC7 provides a filter design function that lets you create filter circuits. You can select the filter type, response, and circuit implementation. MC7 will then create a schematic of the filter circuit for you. This chapter shows you how to use it.

There are two types of filter design available, active filter design and passive filter design. Both are accessible from the Design menu.

Features new in Micro-Cap 7

- Circuits visible in background during filter design.

How the active filter designer works

The MC7 active filter designer is selected from the Design menu. It lets you select the filter type, specifications, response, and circuit implementation, then creates the required filter circuit.

The basic filter types include:

- Low pass
- High pass
- Bandpass
- Notch
- Delay

The first four are defined by their Bode plot characteristics. Delay filters are characterized by the time delay specification.

The available filter responses include:

- Butterworth
- Chebyshev
- Bessel
- Elliptic
- Inverse-Chebyshev

Not all of these responses are available for every filter type. Bessel, for example, is available for delay filters only.

The implementations, or circuits, may be different for each stage and include:

- Sallen-Key
- MFB (Multiple Feedback)
- Tow-Thomas
- Fleischer-Tow
- KHN
- Acker-Mossberg
- Tow-Thomas 2
- DABP (Dual Amplifier Band Pass)

Not all of these circuits are available for all responses, since some of the circuits cannot achieve some responses. Available circuits range from three to eight.

The Active Filter dialog box

The active filter designer is located on the Design menu. Selecting this menu invokes the following dialog box:

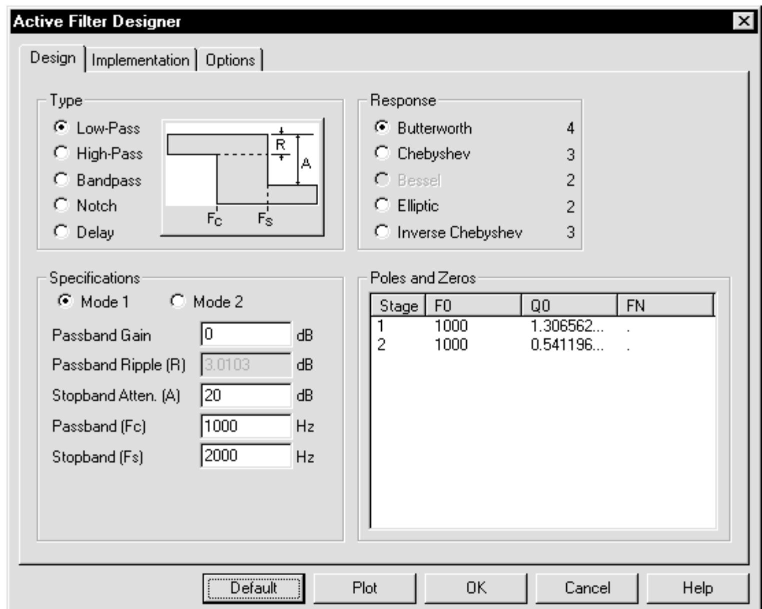


Figure 12-1 The Active Filter dialog box

This dialog box has three main panels, accessible by clicking on a named tab.

Design

This panel lets you pick the filter type, specifications, and the response characteristic. Each time you change one of these selections the number of stages, the pole location, Q value, and, depending upon the response type, the zeros of each stage are calculated and shown in the Poles and Zeros display. You can edit the F0 (pole frequency), Q0 (Q value), and FN (zero frequency) values to change the shape of the response.

Implementation

This panel lets you make the implementation decisions, including which circuit to use, whether to use exact passive component values or to select them from specified lists, which OPAMPs to use, and how to build the odd order stages.

Options

In this panel you choose the number of digits of precision to be used in the component values, what to plot, whether to create a macro or a circuit, and whether to use an existing circuit or a new one.

Design Panel: The main panel contains three sections:

Type: This section lets you select one of the five basic types of filters:

- Low pass
- High pass
- Bandpass
- Notch
- Delay

Response: This section lets you select the mathematical approximation to the ideal filter.

- Butterworth
- Chebyshev
- Bessel
- Elliptic
- Inverse-Chebyshev

Different responses provide different design trade-offs. Butterworth filters require much more circuitry for a given specification, but have a flatter time delay response. Chebyshev and inverse-Chebyshev responses require fewer stages, but have a more pronounced time delay variation. Elliptic responses require the fewest number of stages, but generate the greatest delay variation. Bessel filters are low pass filters with a very flat time delay curve and are really suitable only for delay types. The number of stages to implement the current design is shown to the right of each response.

Specifications: This is where you enter the numerical specifications of the filter. There are two ways to specify a filter, Mode 1 and Mode 2. In Mode 1, you specify the functional characteristics of the filter, like passband gain, cutoff frequency, stop frequency, and attenuation. You specify what you want and the program figures out the number of stages required to achieve it, using the specified response approximation. Mode 2, on the other hand, lets you directly specify the main design values and the number of stages directly.

Poles and Zeros: This section shows the numeric values of the poles, zeros, and Qs of the response polynomial. It essentially shows the mathematical design of the filter. When you make a change to the Type, Response, or Specifications fields, the program redesigns the polynomial coefficients and updates the numbers in this section. If the Plot button has been clicked, the program also redraws the response plot. It may be a Bode plot with gain, phase, and group delay shown, or it may be just a delay plot, depending upon the selections in the Options panel.

The plot is idealized because it is based on a standard polynomial formula for the selected response and the calculated or edited F0, Q0, and QN values. Its plot can only be achieved exactly with perfect components. The actual filter, made from real components, may behave differently. When the circuit is completed, you can run an analysis and see how well it fares. The actual circuit can be constructed from any opamp in the library ranging from ideal to ordinary and from resistors and capacitors that are either exact or chosen from a list of standard values. Real OPAMPs and approximate component values can have a profound effect on the response curves.

You can edit the values, to see their effect on the plot. You can even create your filter with modified values. Though this will not produce an ideal filter, the circuit created will approximate the plot shown. Note that editing any item in the Type, Response, or Specifications areas will cause the program to recalculate the values in the Poles and Zeros section, overwriting your edits.

The exact form of the approximating polynomials for each stage is shown below. Note that U is the complex frequency variable S, normalized to the specified FC.

Definitions

Symbol	**Definition**	
F	Frequency variable	
S	J*2*PI*F	Complex frequency
U	S / (2*PI*FC) = J*F/FC	Normalized complex frequency
F0	Pole location in Hz	
Q0	Q value	
FN	Zero location in Hz (Elliptic and inverse Chebyshev filters only)	

FC Stopband frequency for low-pass filters. Passband frequency for high-pass filters. Center frequency for bandpass and notch filters. The specified center frequency may be changed slightly to produce a symmetrical band pass/reject region. The adjusted F0 can be seen in the define statement. For example, a Butterworth bandpass filter using the default center frequency of 1000, and bandwidth of 100, is actually designed with a center frequency of 998.75 Hz.

FCI Passband frequency for low-pass filters. Stopband frequency for high-pass filters. For bandpass and notch filters FCI = FC

W0 F0/FC Normalized pole frequency

W0I F0/FCI Inverse Chebyshev normalized pole frequency

WN FN/FC Normalized zero frequency

WNI FN/FCI Inverse Chebyshev normalized zero frequency

Low Pass and Delay

Butterworth	$F(U) = 1 / (U^2 + U/Q0 + 1)$
Chebyshev	$F(U) = 1 / (U^2 + U*W0/Q0 + W0^2)$
Elliptic	$F(U) = (U^2+WN^2) / (U^2 + U*W0/Q0 + W0^2)$
Inv. Chebyshev	$F(U) = (U^2+WNI^2) / (U^2 + U*W0I/Q0 + W0I^2)$

High Pass

Butterworth	$F(U) = U^2 / (U^2 + U/Q0 + 1)$
Chebyshev	$F(U) = U^2 / (U^2 + U/(W0*Q0) + 1/W0^2)$
Elliptic	$F(U) = (U^2+WN^2) / (U^2 + U/(W0*Q0) + 1/W0^2)$
Inv. Chebyshev	$F(U) = (U^2+WNI^2) / (U^2 + U/(W0I*Q0) + 1/W0I^2)$

Bandpass

Butterworth	$F(U) = U / (U^2 + U/(W0*Q0) + 1/W0^2)$
Chebyshev	$F(U) = U / (U^2 + U/(W0*Q0) + 1/W0^2)$
Elliptic	$F(U) = (U^2+WN^2) / (U^2 + U/(W0*Q0) + 1/W0^2)$
Inv. Chebyshev	$F(U) = (U^2+WNI^2) / (U^2 + U/(W0I*Q0) + 1/W0I^2)$

Notch

Butterworth	$F(U) = (U^2+1) / (U^2 + U/(W0*Q0) + 1/W0^2)$
Chebyshev	$F(U) = (U^2+1) / (U^2 + U/(W0*Q0) + 1/W0^2)$
Elliptic	$F(U) = (U^2+WN^2) / (U^2 + U/(W0*Q0) + 1/W0^2)$
Inv. Chebyshev	$F(U) = (U^2+WNI^2) / (U^2 + U/(W0I*Q0) + 1/W0I^2)$

Implementation Panel: This panel lets you decide how to build or implement the filter design. It has several major sections:

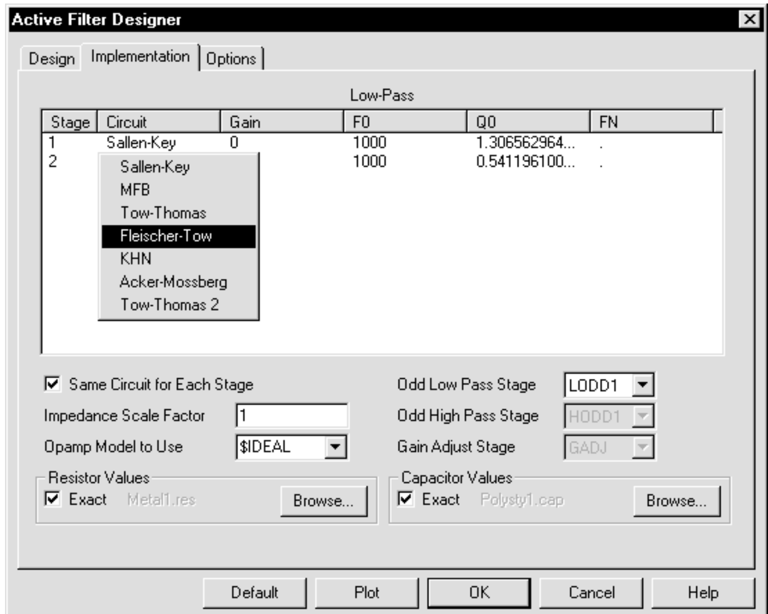


Figure 12-2 The Implementation panel

Stage Values: This section lets you specify, *stage by stage*, the type of circuitry to use and the gain to allocate to each stage. It even lets you edit the poles, Q's, and zeros, by stage. You can swap the pole/Q sets with other stages by clicking in the F0 or Q0 fields with the *right* mouse button. A pop-up menu lets you select another row to interchange F0/Q sets with. Why would you want to? You may want to optimize signal swing and noise sensitivity. Some circuit implementations can handle larger swings and are less noise sensitive. The zeros are fixed and can't be swapped.

To change the stage, click the *left* mouse button in the Circuit column at the row where you wish to change the stage. This will display a list of the available circuit implementations for the selected filter type and response. This list will have a maximum of eight circuits, but may have as little as three circuits, since not all circuits types can realize the requisite transfer function.

When the filter circuit is created, the stages are added and numbered from one on the left to N on the right. The input is always on the left and the output or last stage is always on the right.

Same Circuit for Each Stage: This option forces all stages to use the same circuit. When disabled, you can specify different circuits for each stage.

Impedance Scale Factor: This option lets you specify a scale factor to apply to all passive component values. The factor multiplies all resistor values and divides all capacitor values. It does not change the shape of the response curve, but shifts component values to more suitable or practical values.

Opamp Model to Use: By default, the model is set to $IDEAL. This model is a voltage controlled current source with a small output resistance. It has a very large gain, infinite bandwidth, and no leakage. Its main purpose is to demonstrate how the filter will behave with a nearly perfect device. You can select a different model from the list. It contains hundreds of popular OPAMP models. Vendor supplied OPAMP subcircuit models are not included in the list.

Resistor Values: This option determines how resistor values are chosen. During the implementation phase, when the circuit is being created, MC7 calculates the exact resistor values required by the design. Of course, resistors precise to 16 digits are impractical, so you must choose whether you want to create the circuit using the *exact* values, or to use *closest fit* values chosen from a list of standard parts. The choice is compounded somewhat in that you may be using mostly standard part values, but trimming some. There are several lists of standard parts, and you may add to them or make new lists to special requirements. See the section on list format at the end of this chapter. The Browse button lets you select the file containing the resistor values to use when the Exact check box is disabled.

Capacitor Values: This option lets you decide how the capacitor values will be determined. It works as described in the Resistor Values section.

Odd Low Pass Stage: This option lets you select the last stage to use for low pass filters. There are several different kinds. LODD1 is a simple RC filter. LODD2 is an RC filter, buffered with a non-inverting unity-gain amplifier. LODD3 is an RC filter, buffered with an inverting unity-gain amplifier.

Odd High Pass Stage: This option lets you select the last stage to use for high pass filters. There are three choices. HODD1 is a simple RC filter. HODD2 is an RC filter, buffered with a non-inverting unity-gain amplifier. HODD3 is an RC filter, buffered with an inverting unity-gain amplifier.

Gain Adjust Stage: This option lets you select the stage to use when a gain adjustment is required. There are two choices. NULL adds no stage, which means the gain specification will be ignored. GADJ is a simple inverting amplifier.

Options Panel: This panel lets you set several options. It looks like this:

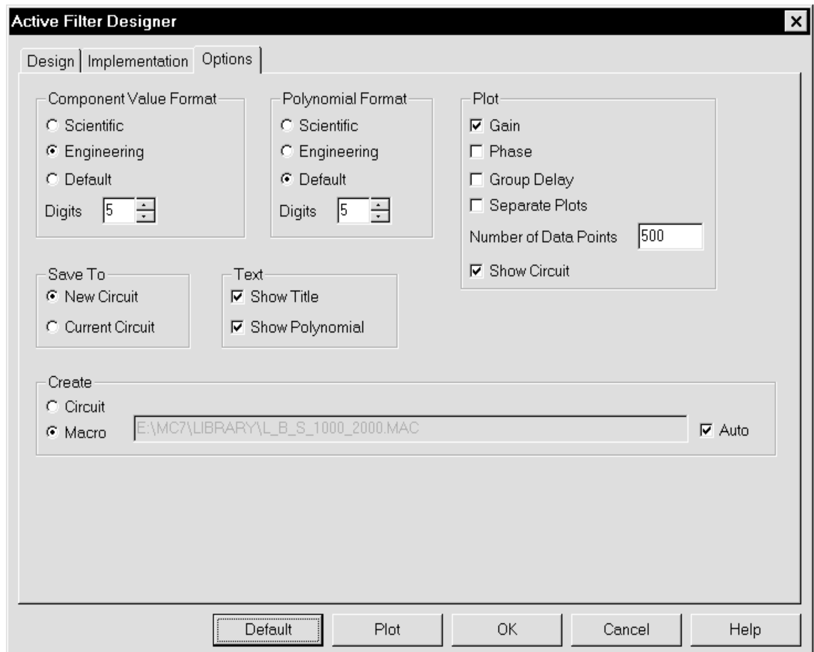


Figure 12-3 The Options panel

There are several options:

Component Value Format: This option lets you select whether the component values are to be specified in scientific, engineering, or default notation. You can also set the number of digits to use when specifying the value. These choices are mainly cosmetic, but for some very high order filters, increasing the number of digits from the default of 5 may be necessary to get an exactly correct Bode plot.

Polynomial Format: This option lets you select whether the polynomial coefficient values are to be specified in scientific, engineering, or default notation. It works the same way as the component values format. The polynomial values are used to define the transfer functions, LP, HP, BP, and BR. These functions are used in AC analysis to plot the ideal transfer function alongside the actual filter transfer function for comparison.

Plot: This option lets you select what to plot. You can select:

- Gain
- Phase
- Group delay
- Separate plots

These options affect several things.

First, it affects the plot displayed when the Plot button in the lower part of the dialog box is clicked. This plot is simply a graph of the ideal complex frequency transfer function. It shows you how the circuit would perform if built from ideal components.

Second, if the Save To / New Circuit option is selected, it changes the AC analysis setup when the circuit is created. It sets up the analysis plot expressions so that you need only select AC analysis and press F2 to see an actual analysis of both the circuit and the ideal transfer function, plotting the variables selected under the Plot option.

Number of Data Points
You can also set the number of data points to be calculated in the plot by editing this field. The internal plot and the AC analysis plot both use fixed log frequency steps rather than auto. This field determines how many data points will be plotted. The default of 500 is usually sufficient, but for very high order filters, you may need to increase the number of points to retain fidelity near steep band edges.

Show Circuit
When checked, this option shows the filter circuit in the background, responding to user changes to specifications.

Save To: This option lets you select where the filter is placed.

- New Circuit: In this option the filter is placed in a new circuit.

- Current Circuit: In this option the filter is placed in the currently selected circuit.

Create: This option lets you select how the filter is created.

- Circuit: In this option the filter is created and placed into either a new circuit or the current circuit.

• Macro: In this option the filter is created as a macro and placed into either a new or current circuit. The macro consists of a number of stages as in the Circuit option, except that they are contained in a macro circuit which is stored on disk. The macro component is entered into a separate component library file called FILTERS.CMP and is available for use by other circuits.

Text: This lets you decide whether to include several optional pieces of text:

• Show Title: This is a self-documenting block of text formed as a title. It identifies the main specifications of the filter.

• Show Polynomials: The polynomial functions that comprise the design are available in a series of .DEFINE statements that can optionally be included with the circuit. This is sometimes handy as a reference. The polynomial function is a symbolic variable and can be plotted versus frequency as a standard against which the actual circuit output can be compared. The names of the polynomials are as follows:

Type	Symbolic Polynomial Name
Low Pass	LP
High Pass	HP
Bandpass	BP
Notch	BR
Delay	LP

Buttons: There are several buttons at the bottom of the dialog box:

Default: This restores default values to all data fields and option buttons.

Plot: This option displays a plot of the selected characteristic(s). These include gain, phase, and/or group delay, depending upon which of these items have been enabled from the Options panel. The plot is a graph of the ideal transfer function for the selected filter. It is not a plot of the actual transfer function that can be achieved with the chosen circuit(s). For that, you must run AC analysis on the completed circuit. If the specification mode is set to Mode 1, the plot includes a design polygon to show the allowed region for the curve, based upon the design specifications.

Here is a sample filter plot showing the design polygon.

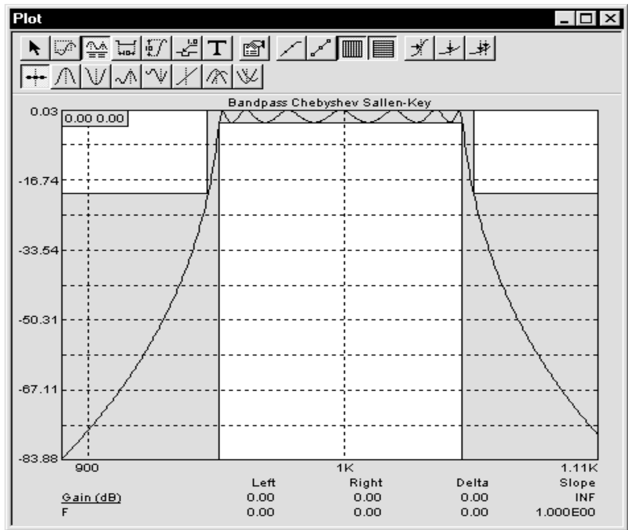


Figure 12-4 A plot of the filter characteristics

OK: This option builds the circuit to meet the chosen design, using the specified circuits for each stage. If the New Circuit option is enabled, it also sets up the AC analysis dialog box to be ready to plot the circuit output and the theoretical transfer function output over a suitable frequency or time scale. If the specification mode is set to Mode 1, it adds a design polygon to show the allowed region for the curve, based upon the design specifications. After creating the filter circuit, the program exits the dialog box.

Cancel: This option exits the dialog box without saving any changes.

Help: This accesses the Help system.

Component lists

Circuits can be constructed from the exact values required to fit the specification exactly, or they can be constructed from nearest-fit values obtained from standard lists. The lists are kept in ASCII text files that use the extensions CAP for capacitors, IND for inductors (for passive filters), and RES for resistors. The format of these files is as follows:

TOLERANCE
<*tolerance*>[%]

DIGITS
<*digit 1*>
<*digit 2*>
…
<*digit n*>

MULTIPLIERS
<*multiplier 1*>
<*multiplier 2*>
…
<*multiplier m*>

ADD
<*value 1*>
<*value 2*>
…
<*value p*>

REMOVE
<*value 1*>
<*value 2*>
…
<*value q*>

The <*tolerance*> value is used when setting up the model statement for the components and affects only Monte Carlo analysis.

The program creates a list of acceptable values by forming all possible products of digits and multipliers, then adding any values found under the ADD keyword, and removing any values found under the REMOVE keyword.

For example:

TOLERANCE
1%

DIGITS
10
50
80

MULTIPLIERS
1
10
100

ADD
26
135

REMOVE
8000

In this example the program would first compile a tentative list from the DIGITS and MULTIPLIERS groups containing these elements:

10, 50, 80, 100, 500, 800, 1000, 5000, 8000

The program would then add the values 26 and 135 and remove the 8000 value to produce the final list:

10, 26, 50, 80, 100, 135, 500, 800, 1000, 5000

The list is constructed in this way to make specification of standard component values easier. Standard values frequently are created from a small list of values, multiplied by various powers of ten. There are often exceptions, usually at the high and low ends. DIGITS and MULTIPLIER keep the list specification concise, while ADD and REMOVE help with the exceptions. All keyword items are optional, so you could create a list using only the ADD elements.

Filter support files

The filter design program uses a file called FILTER.BIN, which is resident on the same directory as the MC7.EXE program file. It must not be removed, for it contains the generic circuit stages for each of the different circuit implementations used to build complete filter circuits. The filter program will be able to do only the mathematical portion of the design if this file is missing. That is, the program will not be able to create actual filter circuits without FILTER.BIN.

MC7 also saves your last used settings from the dialog box in the ACTIVE.FLT file. These settings can be restored to the original default settings by clicking on the Default button at the bottom of the dialog box.

When you press OK, the program creates a circuit to implement the specified filter. It names it CIRCUIT2. Each subsequent circuit becomes CIRCUIT3, then CIRCUIT4, and so on. The circuits can be renamed when they are saved to disk.

Filter specification: mode 1

The active filter program uses two modes to define filter specifications, Mode 1, and Mode 2. Mode 1 includes the following:

Low Pass Filters:
Low pass filter specs are defined relative to the figure below:

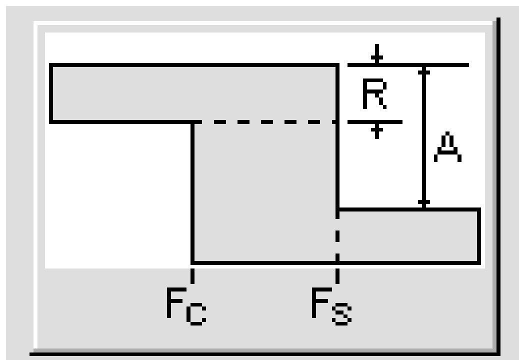


Figure 12-5 Low pass filter specifications

Passband Gain
This is the low frequency gain in dB.

Passband Ripple (R)
This is the variation in gain (in dB) across the passband.

Stopband Attenuation (A)
This is the maximum gain in dB in the passband minus the maximum gain in dB at the stopband. Attenuation is a positive number.

Passband Frequency (Fc)
Below Fc the gain is equal to the passband gain +- ripple.

Stopband Frequency (Fs)
Above Fs the gain is less than or equal to the passband gain +- ripple less the stopband attenuation.

For Chebyshev and elliptic filters the passband gain varies with the order of the filter as follows:

Order	Gain at DC	Gain at passband edge
Even	Passband Gain	Passband Gain + Ripple
Odd	Passband Gain	Passband Gain - Ripple

Butterworth and inverse-Chebyshev passband gain varies as follows:

Type	Gain at DC	Gain at passband edge
Butterworth	Passband Gain	Passband Gain - Ripple
Inverse-Chebyshev	Passband Gain	< Passband Gain - Ripple

Inverse-Chebyshev filters meet the passband spec with margin and meet the stopband spec exactly, with no margin. Ideal realizations of the other filters do just the opposite. They meet the stopband spec with margin and meet the passband specs exactly, with no margin.

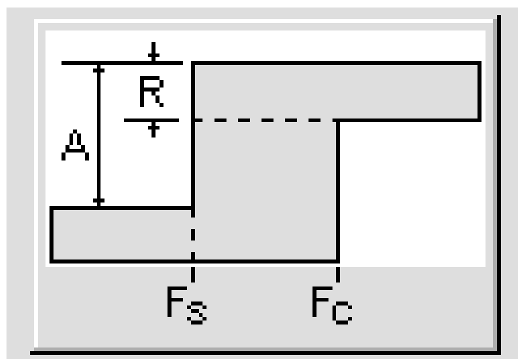


Figure 12-6 High pass filter specifications

High Pass Filters:
High pass filter specs are very similar to their low pass cousins and are defined relative to Figure 12-6.

Passband Gain
This is the high frequency gain in dB.

Passband Ripple (R)
This is the variation in gain (in dB) across the passband.

Stopband Attenuation (A)
This is the maximum gain in dB in the passband minus the maximum gain in dB at the stopband. Attenuation is a positive number.

Passband Frequency (Fc)
Above Fc the gain is equal to the passband gain +- ripple.

Stopband Frequency (Fs)
Below Fs the gain is less than or equal to the passband gain +- ripple less the stopband attenuation.

For Chebyshev and elliptic filters the passband gain varies as follows:

Order	**Gain at DC**	**Gain at passband edge**
Even	Passband Gain	Passband Gain + Ripple
Odd	Passband Gain	Passband Gain - Ripple

Butterworth and inverse-Chebyshev passband gain varies as follows:

Type	**Gain at DC**	**Gain at passband edge**
Butterworth	Passband Gain	Passband Gain - Ripple
Inverse-Chebyshev	Passband Gain	< Passband Gain - Ripple

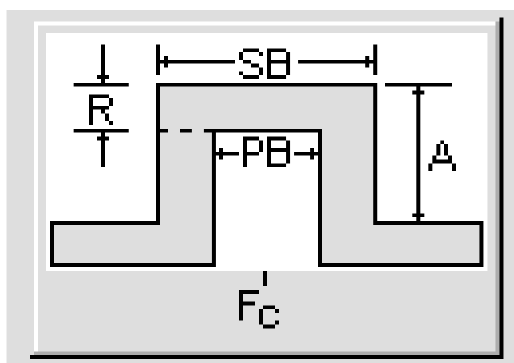


Figure 12-7 Bandpass filter specifications

Inverse-Chebyshev filters meet the passband spec with margin and meet the stopband spec exactly, with no margin. Ideal realizations of the other filters do just the opposite. They meet the stopband spec with margin and meet the passband specs exactly, with no margin.

Bandpass Filters:
Bandpass filter specs are defined relative to Figure 12-7

Passband Gain
This is the maximum gain in dB in the passband.

Passband Ripple (R)
This is the variation in gain in dB across the passband.

Stopband Attenuation (A)
This is the maximum gain in dB in the passband minus the maximum gain in dB at the stopband. Attenuation is a positive number.

Center Frequency (Fc)
This is the center frequency of the passband.

Passband (PB)
This is the band of frequencies where the filter gain is equal to the passband gain (plus or minus the ripple).

Stopband (SB)
This is the band of frequencies which includes the PB plus the two transition regions above and below the passband. It is also the frequency where the filter gain is larger than the total of passband gain + attenuation (plus or minus the ripple).

For Chebyshev and elliptic filters the passband gain varies with the order of the filter as follows:

Order	**Gain at DC**	**Gain at passband edge**
Even	Passband Gain	Passband Gain + Ripple
Odd	Passband Gain	Passband Gain - Ripple

Butterworth and inverse-Chebyshev passband gain varies as follows:

Type	**Gain at DC**	**Gain at passband edge**
Butterworth	Passband Gain	Passband Gain - Ripple
Inverse-Chebyshev	Passband Gain	< Passband Gain - Ripple

Inverse-Chebyshev filters meet the passband spec with margin and meet the stopband spec exactly, with no margin. Ideal realizations of the other filters do just the opposite. They meet the stopband spec with margin and meet the passband specs exactly, with no margin.

Notch Filters:
Notch or band reject filter specs are defined relative to Figure 12-8:

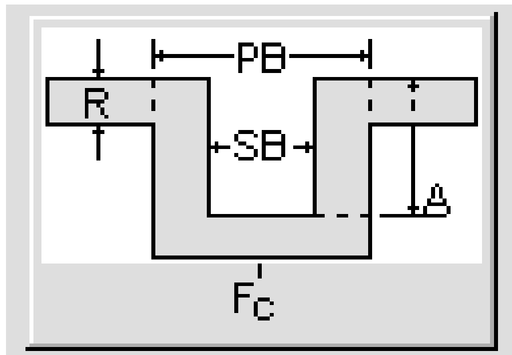


Figure 12-8 Notch filter specifications

Passband Gain
This is the maximum gain in dB outside the passband.

Passband Ripple (R)
This is the variation in gain (in dB) outside the passband.

Stopband Attenuation (A)
This is the maximum gain in dB outside the passband minus the maximum gain in dB in the stopband. Attenuation is a positive number.

Center Frequency (Fc)
This is the center frequency of the stopband.

Passband (PB)
This is the band of frequencies which includes the SB plus the two transition regions above and below the stopband.

Stopband (SB)
This is the band of frequencies where the filter gain is smaller than the total of passband gain minus stopband attenuation (A), plus or minus the ripple.

For Chebyshev and elliptic filters the passband gain varies with the order of the filter as follows:

Order	Gain at DC	Gain at passband edge
Even	Passband Gain	Passband Gain + Ripple
Odd	Passband Gain	Passband Gain - Ripple

Butterworth and inverse-Chebyshev passband gain varies as follows:

Type	Gain at DC	Gain at passband edge
Butterworth	Passband Gain	Passband Gain - Ripple
Inverse-Chebyshev	Passband Gain	< Passband Gain - Ripple

Inverse-Chebyshev filters meet the passband spec with margin and meet the stopband spec exactly, with no margin. Ideal realizations of the other filters do just the opposite. They meet the stopband spec with margin and meet the passband specs exactly, with no margin.

Filter specification: mode 2

Mode 2 formats allow direct specification of the filter order. Its formats are as follows:

Low (High) Pass Filters:

Gain
This is the low (high) frequency gain in dB.

Passband Frequency
This is the frequency below (above) which the gain is equal to Gain.

Ripple
This is the variation in gain (in dB) across the passband.

Order
This is the filter order.

Bandpass and Notch Filters:

Gain
This is the center frequency gain in dB (bandpass) or the low / high frequency gain in dB (notch).

Center Frequency
This is the frequency at which the maximum (bandpass) or minimum (notch) gain is achieved.

Ripple
This is the variation in gain in dB across the passband (bandpass) or outside the passband (notch).

Order
This is the filter order.

Q
This is the filter Q factor. Q is a measure of the resonance near the center frequency.

Delay Filters:

Gain
This is the low frequency gain in dB.

Delay
This is the filter delay in seconds.

How the passive filter designer works

The passive filter design function operates in a manner similar to the active filter function. It is selected from the Design menu. Like its active filter counterpart, it lets you select the filter type, specifications, response, and circuit implementation, then creates the required filter circuit.

The basic filter types include:

- Low pass
- High pass
- Bandpass
- Notch

The filter responses include:

- Butterworth
- Chebyshev

The implementations are of two types:

- Standard
- Dual

The Passive Filter dialog box

The passive filter design function is accessed from the Design menu. Selecting it invokes the following dialog box:

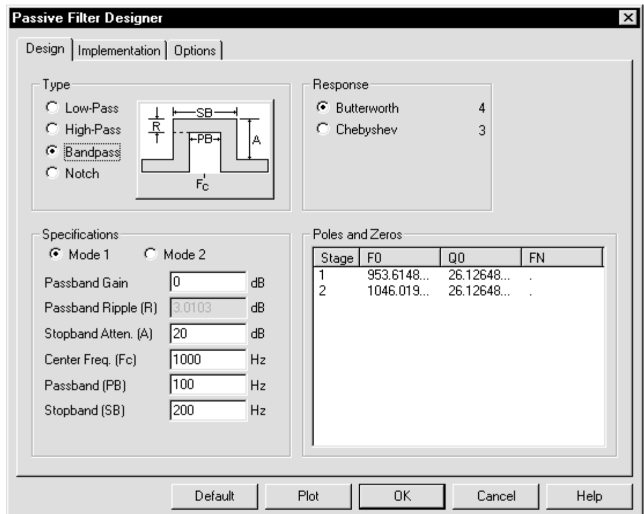


Figure 12-9 The passive filter design panel

This dialog box has three main panels, accessible by clicking on a named tab.

Design

This panel lets you pick the filter type, specifications, and the response characteristic. Each time you change one of these selections, the number of stages, the pole location, and the Q value change as shown in the Poles and Zeros display. You can edit the F0 (pole frequency) and Q0 (Q value) values to change the shape of the response.

Implementation

This panel lets you make the implementation decisions, including which circuit to use, whether to use exact passive component values or to select them from specified lists, whether to scale the values or not, and the source and load resistances.

Options

In this panel you choose the number of digits of precision to be used in the component values, what to plot, whether to create a macro or a circuit, and whether to use an existing circuit or a new one.

Design Panel
The main panel contains three sections:

Type: This section lets you select one of the filter types:

- Low pass
- High pass
- Bandpass
- Notch

Response: This section lets you select the mathematical approximation to the ideal filter.

- Butterworth
- Chebyshev

Different responses provide different design trade-offs. Butterworth filters require much more circuitry for a given specification, but have a flatter time delay response. Chebyshev responses require fewer stages, but have a more pronounced time delay variation. The number of stages to implement the current design is shown to the right of each response.

Specifications: This is where you enter the numerical specifications of the filter. There are two ways to specify a filter, Mode 1 and Mode 2. In Mode 1, you specify the functional characteristics of the filter, like passband gain, cutoff frequency, stop frequency, and attenuation. You specify what you want and the program figures out the number of stages required to achieve it, using the specified response approximation. Mode 2, on the other hand, lets you directly specify the main design values and the number of stages.

Poles and Zeros: This section shows the numeric values of the poles and Qs of the response polynomial. It essentially shows the mathematical design of the filter. When you make a change to the Type, Response, or Specifications fields, the program redesigns the polynomial coefficients and updates the numbers in this section. If the Plot button has been clicked, the program also redraws the response plot. It may be a Bode plot with gain, phase, and group delay show, or it may be just a delay plot, depending upon the selections in the Options panel.

The plot is idealized because it is based on a standard polynomial formula for the selected response and the calculated or edited F0 and Q0 values. Its plot can only be achieved exactly with perfect components. The actual filter, made from real components, may behave differently. When the circuit is completed, you can run an analysis and see how it behaves. The actual circuit can be constructed from a finite list of standard inductors and capacitors. Approximate component values can have a big effect on the response curves.

You can edit the values, to see their effect on the plot. You can even create your filter with modified values. Though this will not produce an ideal filter, the circuit created will approximate the plot shown. Note that editing any item in the Type, Response, or Specifications areas will cause the program to recalculate the values, overwriting your edits.

The exact form of the approximating polynomials for each stage is shown below. Note that U is the complex frequency variable S, normalized to the specified FC.

Definitions

Symbol	Definition	
F	Frequency variable	
S	J*2*PI*F	Complex frequency
U	S / (2*PI*FC) = J*F/FC	Normalized complex frequency
F0	Pole location in Hz	
Q0	Q value	
FC	Stopband frequency for low-pass filters. Passband frequency for high-pass filters. Center frequency for bandpass and notch filters. Note that the specified center frequency may be changed slightly by the program to produce a symmetrical band pass/reject region. The adjusted F0 can be seen in the define statement. For example, a Butterworth bandpass filter using the default center frequency of 1000, and bandwidth of 100, is actually designed with a center frequency of 998.75 Hz.	
W0	F0/FC	Normalized pole frequency

Low Pass Transfer Functions

Butterworth $F(U) = 1 / (U^2 + U/Q0 + 1)$

Chebyshev $F(U) = 1 / (U^2 + U*W0/Q0 + W0^2)$

High Pass Transfer Functions

Butterworth $F(U) = U^2 / (U^2 + U/Q0 + 1)$

Chebyshev $F(U) = U^2 / (U^2 + U/(W0*Q0) + 1/W0^2)$

Bandpass Transfer Functions

Butterworth $F(U) = U / (U^2 + U/(W0*Q0) + 1/W0^2)$

Chebyshev $F(U) = U / (U^2 + U/(W0*Q0) + 1/W0^2)$

Notch Transfer Functions

Butterworth $F(U) = (U^2+1) / (U^2 + U/(W0*Q0) + 1/W0^2)$

Chebyshev $F(U) = (U^2+1) / (U^2 + U/(W0*Q0) + 1/W0^2)$

Implementation Panel

This panel lets you decide how to build or implement the filter design.

It has several major sections:

Circuit: This section lets you specify whether to use the standard LC passive circuit, or its dual circuit.

Resistor Values: This section lets you specify whether to use exact values or to select them from a specified file. These choices are used only for the source and load resistors.

Capacitor Values: This section lets you specify whether to use exact capacitor values or to select them from a specified file.

Inductor Values: This section lets you specify whether to use exact inductor values or to select them from a specified file.

Impedance Scale Factor: This option lets you specify a scale factor to apply to all passive component values. The factor multiplies all resistor and inductor values and divides all capacitor values. It does not change the shape of the response curve, but is used to shift component values to more suitable or practical values.

Source/Load Resistor: This option lets you specify the desired source and load resistor value. If the Exact option is selected, this value, after multiplication by the impedance scale factor, will be used for the source and load resistors in the final circuit. If the Exact option is not checked and a resistor file (*.res) is selected, the value in the resistor file nearest the desired value will be used instead.

Options Panel

The options panel has the same content and use as in the active filter design function. It lets you set component value and polynomial numeric format, determine what to plot and how to plot it, whether to create a circuit or a macro, whether to place the filter in the current or a new circuit, and whether to add a title and polynomial text.

Chapter 13 Scope

What's in this chapter

Scope is a term for the collection of tools for displaying, analyzing, and annotating the analysis plot and its curves. Available in transient, AC, and DC analysis, it lets you expand, contract, pan, and manipulate curves, and display their numeric values. Cursors are available with commands to locate local curve peaks, valleys, maxima, minima, slopes, inflection points, and specific values. Tags, text, and graphics are available to annotate and document the plot.

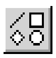
Features new in Micro-Cap 7

- Label Data Points command.
- Label Branches command.
- Mouse or Go To Branch command to select individual branches.
- Top and Bottom to find branch extremes.
- Even Decimal Values Cursor Positioning Mode.
- Incremental auto-ranging.
- New Grid Type (+).
- New range format <*high*> [,<*low*>] [,<*grid spacing*>] [,<*bold spacing*>]
- Plot Grids Quantity, Width, and Pattern Control.
- Thumb Nail Plot.

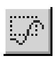
Analysis plot modes

After an AC, DC, or transient analysis is over, there are several modes for reviewing, analyzing, and annotating the analysis plot.

• **Select: (CTRL + E)** In this mode, the left mouse button is used to select text, tags, and graphic objects for moving and editing.



• **Graphics:** Clicking on this button lets you select a graphic object (line, ellipse, rectangle, diamond, arc, pie, or polygon) for placement on the plot. Once the object has been selected you drag the mouse to create the object.

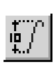


• **Scale: (F7)** In this mode, you can drag the left mouse button to define a plot region to magnify.

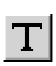
• **Cursor: (F8)** In this mode, the display shows the value of each curve at each of two numeric cursors. The left mouse button controls the left cursor and the right button controls the right cursor. The left cursor is initially placed on the first data point. The right cursor is initially placed on the last data point. The LEFT ARROW or RIGHT ARROW keys move the left cursor (or right cursor if SHIFT is also pressed) to a point on the selected curve. Depending on the cursor positioning mode, the point may be the next local peak or valley, global high or low, inflection point, or simply the next data point.

• **Point Tag:** In this mode, the left mouse button is used to tag a data point with its numeric (X,Y) value. The tag will snap to the nearest data point.

• **Horizontal Tag:** In this mode, the left mouse button is used to drag between two data points to measure the horizontal delta.



• **Vertical Tag:** In this mode, the left mouse button is used to drag between two data points to measure the vertical delta.



• **Text: (CTRL + T)** This mode lets you place text on an analysis plot. You can place relative or absolute text. Relative text maintains its position relative to the curve when the plot scale is changed. Absolute text maintains its position relative to the plot frame. You can also control text border and fill colors, orientation, font, style, size, and effects.

Panning the plot

Panning means to change the plot view without changing the scale. It is usually employed after zooming in. There are two ways to pan the plot:

Keyboard:

Use CTRL + LEFT ARROW to pan the plot to the left.
Use CTRL + RIGHT ARROW to pan the plot to the right.
Use CTRL + UP ARROW to pan the plot up.
Use CTRL + DOWN ARROW to pan the plot down.

Keyboard panning is available from any mode.

Mouse:

Cursor mode: Press and hold the CTRL key down. Place the mouse in the plot window and drag the *right* mouse button in the desired direction.

Other modes: Place the mouse in the plot window and drag the *right* mouse button in the desired direction.

Panning moves only the curves in the selected plot group. Click in another plot group, click on a plot expression in another plot group, or use the Tab key to select a different plot group.

Scaling the plot

Scaling means to shrink or magnify the analysis plot after the analysis is complete. Magnifying enlarges a small region of the plot for inspection. Shrinking brings the plot scale back to a smaller size for a more global view of the plot. There are several scaling commands:

Auto Scale: (F6) This command immediately scales the selected plot group. The selected group is the plot group containing the selected or underlined curve.

Restore Limit Scales: (CTRL + Home) This command draws all plots using the existing scale ranges from the Analysis Limits dialog box.

Zoom-Out: (CTRL + Numeric pad -) This command shrinks the image size of the selected plot group. Zoom-Out  does the same thing.

Zoom-In: (CTRL + Numeric pad +) This command enlarges the image size of the selected plot group. Zoom-In  does the same thing.

Mouse:

Scale mode: Place the mouse near one corner of the region to be magnified and drag the *left* mouse button to the other corner.

Cursor mode: Press and hold the CTRL key. Place the mouse near one corner of the region to be magnified and drag the *left* mouse button to the other corner. This is the same as in Scale mode, but with the CTRL key held down during the drag.

Other modes: There is no mouse scaling for all other modes.

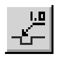
Properties dialog box: (F10) This dialog box controls the characteristics of the front window. When the front window is an analysis plot, the dialog box contains a Scales and Format panel which lets you change the scales of individual curves after the analysis run.

Undo: (CTRL + Z) This command restores the prior scale.

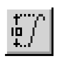
Redo: (CTRL + Y) This command undoes the last scale undo.

Tagging the plot

Tagging is a way of both measuring and documenting a data point or difference between two data points. Tagging can be done on a single curve, between two points on a single curve, or between two points on two curves. There are three tagging modes:



Point Tag mode: This mode lets you tag a data point on a curve. It shows the X expression and Y expression values at the data point. This mode is very useful for measuring or documenting the exact time of a digital event, or the peak or valley of an analog curve.



Vertical Tag mode: This mode lets you drag a tag between two data points on one or two curves. It shows the vertical difference between the Y values at the two data points.

Horizontal Tag mode: This mode lets you drag a tag between two data points on a single curve or two different curves. It shows the horizontal difference between the X expression values at two data points. It is most useful for measuring the time difference between two digital events such as the width of a pulse or the time delay between two events.

There are also several immediate commands for tagging the data points at the numeric cursors.

- Tag Left Cursor: (CTRL + L) This command attaches a tag to the left cursor data point on the selected curve.

- Tag Right Cursor: (CTRL + R) This command attaches a tag to the right cursor data point on the selected curve.

- Tag Horizontal: (SHIFT + CTRL + H) This command attaches a horizontal tag from the left to the right cursor, showing the difference between the two X expression values.

- Tag Vertical: (SHIFT + CTRL + V) This command attaches a vertical tag from the left to the right cursor, showing the difference between the two Y expression values.

The numeric format of the tag numbers is determined by the numeric format set at **Options / Preferences / Format / Analysis Plot Tags**.

Adding graphics to the plot

You can add graphic symbols to the analysis plot when in the Graphics mode. To invoke the Graphics mode, click on the graphics button,  and select one of the objects from the menu that pops up. To add one of the graphic objects, click and drag in the plot area. The objects are:

Rectangle
Line
Ellipse
Diamond
Arc
Pie
Polygon

Each of these objects can be edited after creation by double-clicking on them. This invokes a dialog box that lets you change their border and fill characteristics.

The polygon object allows direct numerical editing of the polygon vertices. It is intended as a design template, a region describing the area that a curve or curve may occupy and still be within specification. The filter designer adds a polygon to the AC plot to indicate the acceptable region for the Bode plot from the user's filter specs. You can use the constants MIN and MAX to specify the plot minimum and maximum coordinates easily.

To see an example of a design polygon, create a filter circuit using the active or passive filter functions from the **Design** menu and run an AC analysis. Enter Select mode and then double-click on the yellow polygon to see its properties, including its border, fill, and vertex structure.

Scope menu

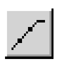
The **Scope** menu provides these options:

- **Delete All Objects:** This command removes all objects (shapes, tags, or text) from the analysis plot. To delete an object, select it by clicking on it, then press CTRL + X or the DELETE key. To delete all objects use CTRL + A to select all objects, then press CTRL + X or the DELETE key.

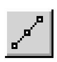
- **Auto Scale:** This command scales the plot group containing the selected curve. F6 may also be used. The selected curve is the one whose Y expression is underlined.

- **Restore Limit Scales:** This command restores the range scales to the values in the Analysis Limits dialog box. CTRL + HOME may also be used.

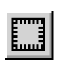
- **View:** The view options only affect the display of simulation results, so you may change these after a run and the screen is redrawn accordingly. These options may also be accessed through Tool bar icons shown below.



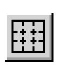
  - **Data Points:** This marks the actual points calculated by the program on the curve plot. All other values are linearly interpolated.



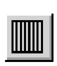
  - **Tokens:** This adds tokens to each curve plot. Tokens are small graphic symbols that help identify the curves.



  - **Ruler:** This substitutes ruler tick marks for the normal full screen X and Y axis grid lines.

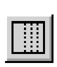


  - **Plus Mark:** This replaces continuous grids with "+" marks at the intersection of the X and Y grids.

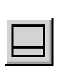


  - **Horizontal Axis Grids:** This adds grids to the horizontal axis.

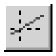
  - **Vertical Axis Grids:** This adds grids to the vertical axis.



  - **Minor Log Grids:** This adds minor log grid lines between the major grids to any axis which uses the log option.



  - **Baseline:** This adds a plot of the value 0.0 for use as a reference.



• **Horizontal Cursor:** This adds a horizontal cursor intersecting each of the two vertical numeric cursors at their respective data point locations.

• **Trackers:** These options control the display of the cursor, intercept, and mouse trackers, which are little boxes containing the numeric values at the cursor data point, its X and Y intercepts, or at the current mouse position.

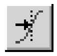
• **Cursor Functions:** These control the cursor positioning in Cursor mode. Cursor positioning is described in more detail in the User's Guide.

• **Label Branches:** This accesses a dialog box which lets you select from the automatic and user-specified X location method for labeling multiple branches of a curve. This option appears when multiple branches are created by Monte Carlo, parameter stepping, or temperature stepping.

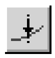
• **Label Data Points:** This accesses a dialog box which lets you specify a set of time, frequency, or input sweep data points, in transient, AC, or DC analysis, respectively, which are to be labelled. This command is mostly used to label frequency data points on polar and Smith charts.

• **Animate Options:** This option displays the node voltages/states, and optionally, device currents, power, and operating conditions (ON, OFF, etc.), on the schematic as they change from one data point to the next. To use this option, click on the  button, select one of the wait modes, then click on the Tile Horizontal  button, or Tile Vertical  button, then start the run. If the schematic is visible, MC7 prints the node voltages on analog nodes and the node states on digital nodes. To print device currents click on . To print device power click on . To print the device conditions click on . It lets you see the evolution of states as the analysis proceeds. To restore a full screen plot, click on the  button.

• **Normalize at Cursor: (CTRL + N)** This option normalizes the selected (underlined) curve at the current active cursor position. The active cursor is the last cursor moved, or if neither has been moved, then the left cursor. Normalization divides each of the curve's Y values by the Y value of the curve at the current cursor position, producing a normalized value of 1.0 there. If the Y expression contains the dB operator and the X expression is F (Frequency), then the normalization subtracts the value of the Y expression at the current data point from each of the curve's data points, producing a value of 0.0 at the current cursor position.

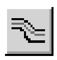


• **Go to X: (SHIFT + CTRL + X)** This command lets you move the left or right cursor to the next instance of a specific value of the X expression of the selected curve. It then reports the Y expression value at that data point.



• **Go to Y: (SHIFT + CTRL + Y)** This command lets you move the left or right cursor to the next instance of a specific value of the Y expression of the selected curve. It then reports the X expression value at that data point.

• **Go to Performance:** This command calculates performance function values on the Y expression of the selected curve. It also moves the cursors to the measurement points. For example, you can measure pulse widths and bandwidths, rise and fall times, delays, periods, and maxima and minima.



• **Go to Branch:** This command lets you select which branch of a curve to place the left and right cursors on. The left cursor branch is colored in the primary select color and the right cursor branch in the secondary select color.

• **Tag Left Cursor: (CTRL + L)** This command attaches a tag to the left cursor data point on the selected curve.

• **Tag Right Cursor: (CTRL + R)** This command attaches a tag to the right cursor data point on the selected curve.

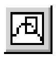
• **Tag Horizontal: (SHIFT + CTRL + H)** This command places a horizontal tag between the left and the right cursor, showing the difference between the two X expression values.

• **Tag Vertical: (SHIFT + CTRL + V)** This command places a vertical tag between the left and the right cursor, showing the difference between the two Y expression values.

• **Align Cursors:** This option, available only in Cursor mode, forces the numeric cursors of different plot groups to stay on the same data point.

• **Keep Cursors on Same Branch:** When stepping produces multiple runs, one line is produced for each run for each plotted expression. These lines are called branches of the curve or expression. This option forces the left and right cursors to stay on the same branch of the selected curve when the UP ARROW and DOWN ARROW cursor keys are used to move the numeric cursors among the several branches. If this option is disabled the cursor keys will move only the left or right numeric cursors, not both, allowing them to occupy positions on different branches of the curve.

• **Same Y Scales:** This option forces all curves in a plot group to use the same Y scale. If the curves have different Y Range values, one or more Y scales will be drawn, which may lead to crowding of the plot with scales.



• **Thumb Nail Plot:** This option draws a small guide plot which gives a global view of where the current plot(s) are on the whole curve. You can also select different views by dragging a box over the curve. Dragging the cursors on the guide plot drags them on the main plot.

The Plot Properties dialog box

Curve display and format can be changed after the run with the Properties dialog box. It looks like this:

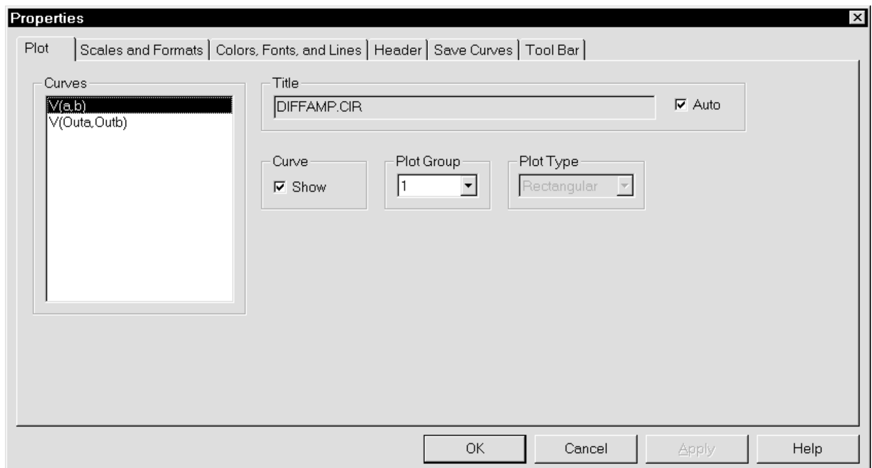


Figure 13-1 The Plot Properties dialog box

The Plot Properties dialog box lets you control the analysis plot window. It can be used for controlling curve display after or even before the plot. Before a plot exists, you can access the dialog box by clicking on the Properties button in the Analysis Limits dialog box. The dialog box provides the following choices:

- **Plot**

  Curves: This lets you select the curve that Plot and Graph fields apply to.

  Title: This lets you specify what the plot title is to be. If the Auto button is checked, the title is automatically created from the circuit name and analysis run details.

  Curve: This check box controls the plotting of the selected curve. To hide the curve, remove the check mark by clicking in the box.

  Plot Group: This controls the plot group number.

  Plot Type: This controls the plot type, rectangular, polar, or Smith chart. The latter two are available only on AC analysis.

- **Scales and Formats**

  Curves: This lets you select the curve that the other fields apply to.

  X: This group includes:

  > *Range Low:* This is the low value of the X range used to plot the selected curve.
  >
  > *Range High:* This is the high value of the X range used to plot the selected curve.
  >
  > *Grid Spacing:* This is the distance between X grids.
  >
  > *Bold Grid Spacing:* This is the distance between bold X grids.
  >
  > *Scale Format:* This lets you specify the numeric format used to print the X axis scale.
  >
  > *Value Format:* This lets you specify the numeric format used to print the X value in the table below the plot in Cursor mode and in the tracker boxes.
  >
  > *Auto Scale:* This command scales the X Range and places the numbers into the Range Low and Range High fields. The effect on the plot can be seen by clicking the Apply button.
  >
  > *Log:* If checked this makes the X scale log.

  Y: This group provides complementary commands for the Y axis group:

  > *Range Low:* This is the low value of the Y range used to plot the selected curve.
  >
  > *Range High:* This is the high value of the Y range used to plot the selected curve.
  >
  > *Grid Spacing:* This is the distance between Y grids.
  >
  > *Bold Grid Spacing:* This is the distance between bold Y grids.
  >
  > *Scale Format:* This is the numeric format used to print the Y axis scale.

Value Format: This lets you specify the numeric format used to print the Y value in the table below the plot in Cursor mode and in the tracker boxes.

Auto Scale: This command scales the Y Range and places the numbers into the Range Low and Range High fields. The effect on the plot can be seen by clicking the Apply button.

Log: If checked this makes the Y scale log.

Same Y Scales: Enabling this check box forces the Auto Scale command to use a single common scale for all plots within a graph group. If the box is not checked, the Auto Scale command determines individual scales for each curve.

Save Range Edits: Enabling this check box causes any edits to the range fields to be copied to the appropriate range fields of the Analysis Limits dialog box, making them permanent.

Slope Calculation: This list box lets you select the Normal, dB/Octave, or dB/Decade method of calculating slopes. The latter two are most useful in AC analysis.

Use Common Formats: Clicking this button copies the X and Y formats of the selected curve to the format fields of all curves.

- **Colors, Fonts, and Lines**

  Objects: This list box lets you select the object that the other commands (color, font, lines) apply to. These include:

  General Text: This is text used for axis scales, titles, cursor tables, and curve name.

  Grid: This is the analysis plot grid.

  Graph Background: This is the plot background.

  Window Background: This is the window background.

  Select: This is the color of a selected object.

  Select Box: This is the Select mode box.

Initial Object: This governs the initial properties of analysis text added to the plot, graphical objects, and numeric tags. Object properties can be changed after they are added to a plot by double-clicking on them.

Tracker: This sets the text and color properties of trackers.

Select Color Primary: This sets the color of the branch that the Go To Branch Left button selects.

Select Color Secondary: This sets the color of the branch that the Go To Branch Right button selects.

Data Point Labels: This sets the text and color properties of data point labels.

Plot All: This sets the text, line, and color properties of all curves simultaneously.

Curve Names: This sets the text, line, and color properties of the individual curves.

Variable Name (Color): This group lets you change the color of the selected object. The group name changes to reflect the object chosen.

Curve Line: This group lets you change the color, width, and pattern of the curve. The Rainbow option assigns a spectrum of colors to each branch of a stepped curve.

Font: This field lets you change the font of the selected object.

Size: This field lets you change the text size of the selected object.

Font style: This field lets you change the text style of the selected object.

Effects: This field lets you change text effects of the selected object.

Sample: This field shows a sample of the selected object using current text, line, color, width, and pattern properties.

- **Header:** This group controls the header format for text numeric out.

Left: This group lets you add text to the left side of the text output.

Center: This group lets you add text to the center of the text output.

Right: This group lets you add text to the right side of the text output.

In each case the following formats are available:

$MC	Prints Micro-Cap
$User	Prints user name
$Company	Prints company name
$Analysis	Prints analysis type (Transient, AC, DC)
$Name	Prints circuit name

You can use these or any other text in the left, center, or right.

Delimiters: This group lets you select the delimiter that will be placed between items in the curve tables of the numeric output. The choices are Tab, Semicolon, Comma, Space, and Other.

• **Save Curves:** This group lets you save one or more curves for later display or use in a User source. It provides these fields:

Curves: This lets you select the curve that the other fields apply to.

Temperature: If the analysis run stepped temperature and produced multiple curves, this field lets you select which to save. This field will be grayed out if temperature was not stepped.

Stepped Variable: If the analysis run stepped a variable and produced multiple curves, this field lets you select which to save. This field will be missing if nothing was stepped.

Save Curve: This field is a copy of the selected curve name.

As (New Name): This lets you specify the name to save the curve under. It is the name you later use to select the curve for display or use in a source.

In File: This lets you specify the file name to save the curve under.

Browse: This command lets you browse directories for the file name you want to save the curve under or to delete.

Save: This command saves the selected curve using the specified curve name and file name. Note that when curves are saved to existing files, they are added to the file. If the curve already exists in the file, it is overwritten. All other curves in the file are unaffected.

Delete: This command lets you delete the specified curve name.

• **Tool Bar:** This page lets you select the buttons that will appear in the local tool bar area below the Main tool bar.

Tool Bar: This list box lets you select the different local tool bars.

Buttons: This box lets you select the buttons that are to appear in the selected local tool bar.

Show Button: If checked, the selected button is shown in the analysis plot tool bar.

Top: If enabled, the tool bar is placed at the top part of the window.

Left: If enabled, the tool bar is placed at the left part of the window.

All On: This command places all buttons in the tool bar.

All Off: This command places no buttons in the tool bar.

Default: This command places the default set of buttons in the tool bar.

The four buttons at the bottom have the following function:

OK: This button accepts all changes, exits the dialog box, and redraws the analysis plot. Subsequent runs will use the changed properties, and they will be retained in the circuit file, if it is later saved.

Cancel: This button rejects all changes, exits the dialog box, and redraws the analysis plot using the original properties.

Apply: This button displays the analysis plot using the current settings in the dialog box to show how the display would be affected by the changes. The changes are still tentative, until the OK button is clicked.

Help: This button accesses the local help files.

Chapter 14 Probe

What's in this chapter

This chapter describes the use of Probe. Probe is a display tool for transient, AC, and DC analysis. When one of the Probe options is selected and an analysis is subsequently run, the simulation results are saved to disk. Probe then lets the user review the results by probing the schematic with the mouse.

Features new in Micro-Cap 7

- Adding or deleting curves with CTRL + click in One Curve mode
- Probing differential voltages with SHIFT + click
- Probing SPICE files
- Probing energy terms

How Probe works

Probe is another way to view simulation results. It lets you point to a location in a schematic and see one or more curves associated with the node or component at that point. It functions exactly like a normal simulation, but accesses all of the variables for each solution point from a disk file. When you first invoke Probe, the program determines if there is an up-to-date simulation file in the working data directory. If not, it runs the analysis and creates the simulation file. When you click on the schematic, Probe determines where the mouse pointer is, extracts from the file the appropriate variable for both the vertical and horizontal axes, and plots the resulting curve.

The simulation or analysis run is conducted according to the values set in the Analysis Limits dialog box. For example, in transient analysis, one of the important values is Time Range, which determines how long the analysis will run. To edit this or any other value, press F9 to access the dialog box. This will present an abbreviated version of the standard dialog box. You can edit the fields as needed. Press F2 to rerun the analysis, prior to probing.

Plots are constructed using the properties from **Options / Default Properties for New Circuits / Analysis Plots**. Numeric scales and Cursor mode values are formatted using the **Scales and Formats** settings. Plot text, line properties, and colors are taken from the **Colors, Fonts, and Lines** settings. Tool bar choices are taken from the **Tool Bar** settings.

You can temporarily change plot properties from the Properties dialog box (F10). These changes are used during the current Probe session and are discarded after exiting Probe. Subsequent invocations of Probe will start again with the settings from **Options / Default Properties for New Circuits / Analysis Plots**.

Probe menu

• **New Run (F2):** This option forces a new run. Probe automatically does a new run when the time of the last saved run is earlier than the time of the last edit to the schematic. However, if you have changed RELTOL or some other Global Settings value or option that can affect a simulation run, you may want to force a new run using the new value.

• **Limits:** This lets you edit the analysis limits for the run.

• **Add Curve:** This option lets you add a plot defined by a literal expression using any circuit variable. For example you might enter VCE(Q1)*IC(Q1) to plot a transistor's collector power.

• **Delete Curves:** This option lets you selectively remove curves.

• **Delete All Curves: (CTRL + F9)** This removes all curves from the plot.

• **Separate Analog and Digital:** This puts analog and digital curves in separate plot groups, overriding the P setting.

• **One Curve:** In this mode, only one curve is plotted. Each time the schematic is probed, the old curve is replaced with the new one. *You can also add more than one trace with this mode by holding the CTRL key down while clicking on an object. This adds the curve if not already plotted or deletes it if it is already plotted.*

• **Many Curves:** In this mode, new curves do not replace old ones, so many curves are plotted together using one or more vertical scales.

• **Save All:** This option forces Probe to save all variables. Use it only if you need to display charge, flux, capacitance, inductance, B field, or H field.

• **Save V and I Only:** This option saves space and lowers access time by saving only time, frequency, digital states, voltage, and current variables. It discards the remainder, including charge, flux, capacitance, inductance, resistance, power, and magnetic field values.

• **Plot Group:** This lets you pick the plot group to place the next curve in.

• **Exit Probe:** This exits Probe. F3 also works.

Transient analysis variables

The **Vertical** menu selects the vertical variable and the **Horizontal** menu selects the horizontal variable. When the user clicks the mouse in a schematic, Probe determines whether the object at the mouse tip is a node or a component and whether it is analog or digital.

If the object is a digital node, Probe plots the state curve of the node.

If the object is an analog macro or subcircuit, or a digital component, Probe presents a list showing its pin names and the associated node names. You can select one or more analog node voltages or digital state curves by clicking on the pin name or node name. After clicking the OK button, Probe plots the selected curve.

If the object is either an analog node or analog component (other than a macro or a subcircuit), Probe extracts the vertical and horizontal variables specified by these menus and uses them to plot the analog curve.

• **Voltage:** If the mouse probes on a node, a node voltage is selected. If the mouse probes on the shape of a two-lead component, the voltage across the component is selected. If the mouse probes between two leads of a three or four lead active device, the lead-to-lead difference voltage is selected.

Press the SHIFT key and click on two nodes and you'll get the differential voltage across the two nodes.

• **Current:** If the mouse probes on the shape of a two-lead component, the current through the component is selected. If the mouse probes on a lead of a three or four lead active device, the current into the lead is selected.

• **Energy:** If the mouse probes on a component, it plots the energy dissipated (ED), generated (EG), or stored (ES) in that component. If the component has more than one of these, a list appears allowing selection. Clicking off a component lets you select one of the total energy terms, EGT (total generated energy), EST (total stored energy), or EDT (total dissipated energy).

• **Power:** If the mouse probes on a component, it plots the power dissipated (PD), generated (PG), or stored (PS) in that component. If the component has more than one of these, a list appears allowing selection. Clicking off a component lets you select one of the total power terms, PGT (total generated power), PST (total stored power), or PDT (total dissipated power).

• **Resistance:** If the mouse clicks on a resistor, this selects the resistance.

• **Charge:** If the mouse clicks on a capacitor, this selects its charge. If the probe occurs between the leads of a semiconductor device, this selects the charge of the internal capacitor between the two leads, if any. For example, a click between the base and emitter of an NPN selects the CBE charge stored in the diffusion and junction capacitance.

• **Capacitance:** If you click on a capacitor, this selects its capacitance. If the probe occurs between the leads of a semiconductor device, this selects the capacitance of the internal capacitor between the two leads, if any. For example, a click between the base and emitter of an NPN selects the diffusion and junction capacitance of the base-emitter region.

• **Flux:** If the mouse clicks on an inductor, this selects the flux.

• **Inductance:** If the mouse clicks on an inductor, this selects its inductance.

• **B Field:** If the mouse clicks on an inductor which is referenced in a K (coupling) device with a nonlinear core model specified, this selects the B field of the core.

• **H Field:** If the mouse clicks on an inductor which is referenced in a K (coupling) device with a nonlinear core model specified, this selects the H field of the core.

• **Time:** This selects the transient analysis simulation time variable.

• **Linear:** This selects a linear scale.

• **Log:** This selects a log scale.

AC analysis variables

Only analog variables and operators are available in AC analysis. They include:

- **Voltage:** If the object is a node, a complex node voltage is selected. If the object is a two-lead component, the complex voltage across the component is selected. If the mouse probes between two leads of a three or four lead active device, the lead-to-lead differential complex voltage is selected. Hold the Shift key down and probe on two nodes to get differential voltage.

- **Current:** If the object is a two-lead component, the complex current through the component is selected. If the mouse probes on a lead of a three or four lead active device, the complex current into the lead is selected.

- **Inoise:** This selects a plot of noise, regardless of where the mouse is clicked. Inoise is referenced to the input source specified in the Noise Input field of the Analysis Limits dialog box (F9).

- **Onoise:** This selects a plot of noise, regardless of where the mouse is clicked. Onoise is referenced to the output node name specified in the Noise Output field of the Analysis Limits dialog box (F9).

- **Frequency:** This selects the sweep frequency variable.

- **Magnitude:** This plots the magnitude of the probe variable.

- **Magnitude(dB):** This plots the decibel magnitude of the probe variable. It is the default operator.

- **Phase:** This plots the phase of the probe variable.

- **Group Delay:** This plots the group delay of the probe variable.

- **Real Part:** This plots the real part of the probe variable.

- **Imag Part:** This plots the imaginary part of the probe variable.

- **Linear:** This selects a linear scale.

- **Log:** This selects a log scale.

DC analysis variables

Both analog and digital variables are available in DC analysis. They include:

- **Voltage:** If the mouse probes on a digital node, the digital state of the node is selected. If the mouse probes on an analog node, its node voltage is selected. If the mouse probes on the shape of a two-lead component, the voltage across the component is selected. If the mouse probes between two leads of a three or four lead active device, the lead-to-lead differential voltage is selected. Hold Shift down and probe on two nodes to get differential voltage.

- **Current:** If the mouse probes on a digital node, the digital state of the node is selected. If the mouse probes on the shape of a two-lead component, the current through the component is selected. If the mouse probes on a lead of a three or four lead active device, the current into the lead is selected.

- **Power:** If the mouse probes on a component, it plots the power dissipated (PD), generated (PG), or stored (PS) in that component. If the component has more than one of these, a list appears allowing selection. Clicking off a component lets you select one of the total power terms, PGT (total generated power), PST (total stored power), or PDT (total dissipated power).

- **Linear:** This selects a linear scale.

- **Log:** This selects a log scale.

The default horizontal variable is the value of the specified Variable 1 of the DC Analysis Limits dialog box. For example, if the source is a voltage source, the horizontal value is the voltage across the source, or if the source is a current source, the horizontal value is the current through the source.

This default can be changed through the Plot item in the Properties dialog box. The Properties dialog box is invoked by clicking in the Probe plot, then pressing F10. A plot must be present before an X variable change can be made.

Probe analog variables

The Horizontal and Vertical menus are used to select the curve variables and operators which will be displayed by Probe. The curves displayed for each variable or operator are dependent upon the component selected. The variables and operators are shown in the following tables.

Component Variables							
Component	Voltage	Current	Capacitance / Inductance	Charge / Flux	Energy / Power Generated	Energy / Power Stored	Energy / Power Dissipated
Sources	V	I	NA	NA	EG / PG	NA	NA
Resistor	V	I	NA	NA	NA	NA	ED / PD
Capacitor	V	I	C	Q	NA	ES / PS	NA
Inductor	V	I	L	X	NA	ES / PS	NA
Diode	V	I	C	Q	NA	ES / PS	ED / PD
Transmission Line	VAP, VAM, VBP VBM	IAP, IAM IBP, IBM	NA	NA	NA	NA	NA
BJT	VB, VC, VE, VBE, VBC, VEB VEC, VCB, VCE	IB, IE, IC	CBE, CBC	QBE, QBC	NA	ES / PS	ED / PD
BJT4	VB, VC, VE, VS, VBE, VBC, VBS VEB VEC, VES VCB, VCE, VCS VSB, VSE, VSC	IB, IE, IC IS	CBE, CBC CCS	QBE, QBC QCS	NA	ES / PS	ED / PD
MOSFET: LEV 1-3	VG, VS, VD, VB, VGS, VGD, VGB VDS, VDG, VDB VSG, VSD, VSB VBG, VBD, VBS	IG, IS, ID IB	CGS, CGD CGB, CBD CBS	QGS, QGD QGB, QBD QBS	NA	ES / PS	ED / PD
MOSFET:LEV 4,5,8	VG, VS, VD, VB, VGS, VGD, VGB VDS, VDG, VDB VSG, VSD, VSB VBG, VBD, VBS	IG, IS, ID IB	NA	NA	NA	NA	ED / PD
OPAMP	VP, VM, VOUT, VPM, VCC, VEE	NA	NA	NA	NA	NA	NA
JFET	VG, VD, VS, VGS, VGD, VSG VSD, VDG, VDS	IG, ID, IS	CGS, CGD	QGS, QGD	NA	ES / PS	ED / PD
GaAsFET	VG, VD, VS, VGS, VGD, VSG VSD, VDG, VDS	IG, ID, IS	CGS, CGD	QGS, QGD	NA	ES / PS	ED / PD
Variables that are mere permutations of the leads are not shown. For example CGS and CSG produce the same plot as do QGS and QSG.							

Table 14-1 General syntax for Probe variables

Resistance is available only for resistors. Inductance, B field, and H field are available only for inductors.

Probe regions

When probing for node voltages or digital states, you click the mouse on one of the round dots shown on the node, or on any portion of any wire connecting to a node. If there is more than one dot on a node, it doesn't matter which you pick.

Probe can plot many internal device variables. For example, the internal charge and capacitance of a MOSFET (Level 1-3 only) can be accessed. The probe regions for devices with three or more pins are more complicated than for nodes and other devices. For example, the variable selected for a transistor is determined by the nearest device pin line segment. These line segments are illustrated below for the case of a MOSFET. When the mouse is clicked near a device, the distance from each line segment to the point where the mouse is clicked is determined. The closest line segment determines the two leads. The Vertical and Horizontal menu choices determine the variable type.

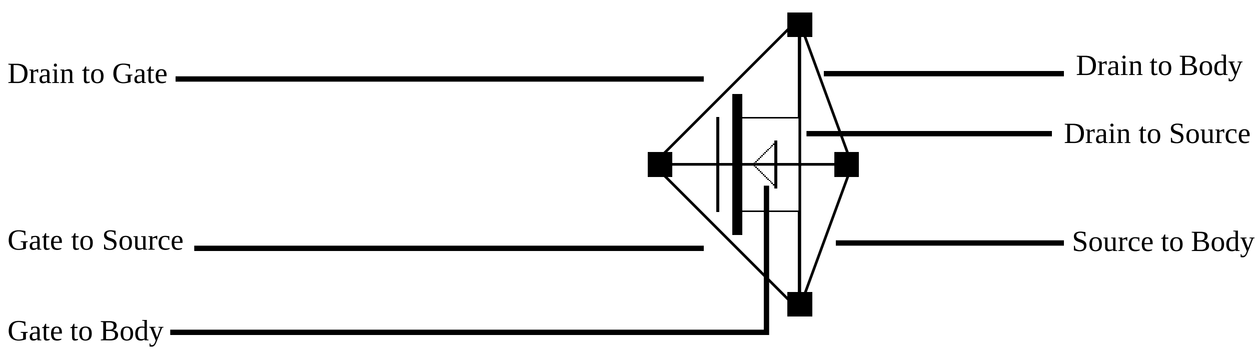


Figure 14-1 MOSFET line segment diagram

Probing a SPICE file

MC7 can also probe SPICE files. Clicking on two-terminal devices like sources, resistors, and diodes plots the voltage across or current through the device, depending upon whether the Vertical option is set to Voltage or Current.

Clicking on three-terminal devices, presents a dialog box where you can select the desired voltage or current variable.

Clicking on a node number plots the node voltage.

Chapter 15 Stepping

What's in this chapter

Stepping is the process of varying a parameter value to see its effect on circuit behavior. Transient, AC, and DC analysis all provide the capability of stepping parameter values.

This feature cannot be used simultaneously with Monte Carlo analysis. If both are enabled, MC7 will enable the last one turned on by the user, and disable the other.

Stepped waveforms are plotted on the same graph. To distinguish one from the other, use Cursor mode and the UP CURSOR ARROW and DOWN CURSOR ARROW keys to switch between branches. As the cursor keys change the selected branch (step value) of the waveform, the window title changes showing the value of the stepped parameter(s). You can also label the individual branches and use the mouse or the Go To Branch feature to select individual branches.

Features new in Micro-Cap 7

- Text stepping using symbolic parameters and the list method.
- Stepping symbolic parameters created with .PARAM.
- Ability to identify curve branches by labeling them and by selecting them with the ALT key and a mouse click or the Go To Branch feature.

How stepping works

Stepping systematically alters the value of one or more parameters of one or more components and then runs the analysis, drawing multiple branches for each curve. Earlier versions of the program placed restrictions on changing some variables, such as the model level and parasitic resistances. Most parameter types can now be stepped, including attribute parameters like the resistance of a resistor, model parameters like a transistor beta, and symbolic parameters created with a .define or a .param command. If the parameter changes the equation matrix, MC7 simply recreates the equations. For each parameter set, a run is made and the specified waveforms plotted.

With parameter stepping enabled, 3D plots can be used to display performance function dependence on parameter values.

What can be stepped?

There are three basic types of variables that can be stepped:

Attribute parameters
Model parameters
Symbolic parameters (those created with a .DEFINE statement)

Some components such as the simple dependent sources, IOFI, IOFV, VOFI, and VOFV, are characterized by a single attribute parameter called 'VALUE'. In this type of component, this is the only parameter that can be stepped.

Some components have only model parameters, and these are the only parameters that can be stepped.

Some components have both attribute and model parameters and both types of parameters can be stepped.

Symbolic parameters used in model statements or attributes can be stepped.

Using a symbolic variable, you can also step text using the list option. This is handy for changing models, stimulus files, or other text-based parameters.

The Stepping dialog box

Parameter stepping is controlled by the Stepping dialog box. It looks like this:

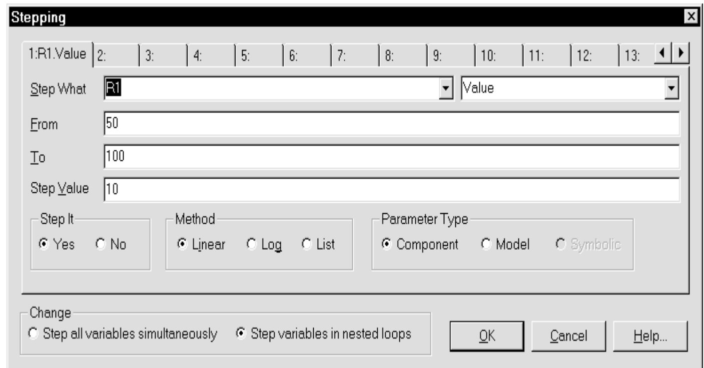


Figure 15-1 The Stepping dialog box

The dialog box is divided into several areas:

- **Parameter Panels**
  The dialog box provides tabs for up to twenty parameters, although two or three is a practical maximum. Each tab accesses a panel which controls a single parameter. Stepping for each parameter is enabled when its Step It option is set to Yes.

  - **Step It:** Set this option to Yes to enable stepping for the parameter.

  - **Step What**
    The left Step What list box in each parameter panel specifies the name of the model parameter, component, or symbolic variable value to be stepped. Since dissimilar parts may share the same model name, the electrical definition is shown along with the name. Clicking on the list box displays a list of the items available for stepping. To select one, click on it. The right Step What list box in each panel specifies the name of the parameter to be stepped. Clicking on the list box displays a list of the available parameters. To select a parameter, click on it.

  - **From:** This field specifies the starting value of the parameter.

  - **To:** This field specifies the ending value of the parameter.

• **Step Value:** This field specifies the step value of the parameter.

• **Method:** The Method option in each panel controls how the Step Value affects the parameter value.

  • **Linear:** Linear stepping *adds* the Step Value.

  • **Log:** Log stepping *multiplies* by the Step Value.

  • **List:** List stepping lets you enter a comma-delimited set of specific values in the From field.

• **Parameter Type:** This option in each panel specifies whether the Step What field refers to a model, component, or symbolic name.

  • **Component:** This is used for passive parameter values, as for example, the resistance of a resistor.

  • **Model:** This is used for model parameters, as for example, the BF of a BJT, or a delay parameter for a digital device.

  • **Symbolic:** Symbols are variables created with a .DEFINE or a .PARAM statement. They can be used in component value parameters or in model parameters. Stepping them is a powerful way of controlling many parameter values. For example, you can control the W and L of many MOSFET devices with this:

```
.DEFINE W1 2U
.DEFINE L1 0.3U

.MODEL NM1 NMOS (W=W1 L=L1...)
```

  This lets you selectively step only the W and L of the devices that use the symbols W1 and L1.

You can step an individual instance or all instances of a model parameter. In Component mode, stepping affects one parameter of one device only if the PRIVATEANALOG and PRIVATEDIGITAL options (Global Settings) are enabled. In Model mode, stepping affects one parameter of all devices that use that model name. This is true regardless of the state of the PRIVATEANALOG or PRIVATEDIGITAL flags.

• **Change:** The Change option only comes into play if you are stepping multiple parameters. It controls whether the parameter changes are to be nested or simultaneous.

  • **Step all variables simultaneously:** In this type of change, all parameters change value simultaneously, so you get a small set of matched parameter values.

  • **Step variables in nested loops:** In this type of change, each parameter changes independently, so you get all combinations of the specified values.

  For example, suppose you step L1 through the values 1u and 2u and you step C1 from 1n to 2n. Simultaneous stepping, will produce two runs, and nested stepping will produce four runs as shown below.

Nested	Simultaneous
L1=1u C1=1n	L1=1u C1=1n
L1=1u C1=2n	L1=2u C1=2n
L1=2u C1=1n	
L1=2u C1=2n	

  Simultaneous stepping requires an equal number of steps for each parameter. If the parameter panels specify different numbers of steps, an error message will be issued. Use nested stepping when you want all combinations of parameter variation. Use simultaneous stepping when you want only specific combinations.

• **OK:** This button accepts the changes made and exits the dialog box.

• **Cancel:** This button ignores any changes and exits the dialog box.

• **Help:** This button accesses the Help system.

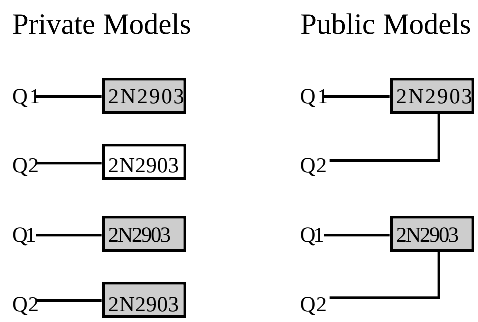
Public vs. private libraries

To illustrate the stepping possibilities with the four combinations of public or private libraries and component or model stepping, imagine a circuit with two transistors, each having a MODEL attribute of "2N2903".

This circuit uses two transistors with component names Q1 and Q2. They both use the 2N2903 model. This diagram illustrates the four possibilities.

Component Mode Stepping
Case 1: Step Q1.BF

Model Mode Stepping
Case 2 : Step 2N2903.BF



The shaded models are changed by stepping. In case 1, we are stepping Q1.BF. In this case we step the model that Q1 points to. When the libraries are private, then the model pointed to by Q1 would be stepped and that pointed to by Q2 would not be stepped. They point to distinct model locations, even though they use the same model name. When the libraries are public, stepping the model pointed to by Q1 also steps the model pointed to by Q2 since they point to the same model location.

In case 2, we are stepping 2N2903.BF. In this case we step all of the instances of the 2N2903 model. It doesn't matter whether models are private or public, since all instances of the model are changed.

Only component stepping and private libraries selectively step individual instances of model parameters. All other cases step all instances of model parameters.

Stepping summary

The most important things to remember when using stepping include:

1. The parameters of these components may not be stepped:

   - Transformer
   - User sources
   - Laplace sources
   - Function sources
   - SPICE dependent sources (E,F,G,H sources)
   - Old switches (S and W switch parameters can be stepped)

   The behavior of User, Laplace, Function, and SPICE sources are embodied in algebraic formulas and numeric tables, and thus have no parameters to be stepped. You can step symbolic parameters which are used in Laplace, Function, and SPICE source functions. For example, you could use

   .DEFINE TAU 5

   A Laplace function source whose LAPLACE attribute is:

   1/(1+TAU*S)

   could then be changed by stepping TAU.

   Similarly in a Laplace table source like

   .DEFINE RVAL 2.0

   .DEFINE TAB (1k,0,RVAL)

   you could change the behavior of the source by stepping RVAL.

   The User source gets its data from an external data file and has no parameter that can be stepped.

2. In Component mode, stepping affects one parameter of one device only if the PRIVATEANALOG and PRIVATEDIGITAL options (Global Settings) are enabled. In Model mode, stepping affects one parameter of all devices that use that model name. Thus you are potentially affecting many devices.

This is true regardless of the state of the PRIVATEANALOG or PRIVATEDIGITAL flags. In model stepping, AKO models track stepped parameters in the parent model and all of the temperature model parameters are available for stepping.

3. The MOSFET Level model parameter may be stepped, but an error will occur if the model statement has been created with parameters for level 1, 2, or 3 and the level changes to a BSIM model (Level 4, 5, or 8). Each BSIM model uses model parameter names that are different from the other BSIM models as well as the level 1, 2, and 3 models, so the parameter names would be meaningless.

4. Linear stepping starts with the From value and adds the Step Value until it reaches the To value. Log stepping starts with the From value and then multiplies by the Step Value until it reaches the To value. A Step Value of 2 is often convenient and is called octave stepping. A Step Value of 10 is sometimes referred to as decade stepping. List stepping simply uses the values specified in the From field.

5. If multiple parameters are to be *simultaneously* stepped, they must each specify the same number of steps. If there is a mismatch, an error message is generated.

6. At least two parameters must be varied to create 3D plots where a performance function is chosen for the Y axis.

7. Stepping a resistor, capacitor, or inductor that uses an expression for its value, replaces the value calculated from the expression with the step value. In other words, the step value takes precedence over the calculated value.

Chapter 16 Monte Carlo Analysis

What's in this chapter

After successfully simulating a circuit, a user may want to know how the circuit performance is affected by parameter variation. Monte Carlo analysis provides a means of answering that question.

During Monte Carlo analysis, multiple runs are performed. For each run, a new circuit is generated from components whose numerical parameter values are randomly selected. The selection process is based upon user-specified parameter tolerances and distribution types. MC7 extracts performance characteristics from each run and displays the information graphically in the form of histograms and numerically in the form of statistical parameters.

Monte Carlo-related features new in Micro-Cap 7

- Ability to identify curve branches by labeling them and by selecting them with the ALT key and a mouse click or the Go To Branch feature.
- Ability to use performance function expressions in histograms instead of single performance functions.

How Monte Carlo works

Monte Carlo works by analyzing many circuits. Each circuit is constructed of components randomly selected from populations matching the user-specified tolerances and distribution type.

Tolerances are applied to a component's numeric model parameters. *Only model parameters and symbolic parameters can be toleranced.* Tolerances are specified as an actual value or as a percentage of the nominal parameter value.

Both absolute (LOT) and relative (DEV) tolerances can be specified. A LOT tolerance is applied absolutely to each device. A DEV tolerance is then applied to the first through last device relative to the LOT toleranced value originally chosen for the first device. In other words, the first device in the list receives a LOT tolerance, if one was specified. All devices, including the first, then receive the first device's value plus or minus the DEV tolerance. DEV tolerances provide a means for having some devices track in their critical parameter values.

Both tolerances are specified by including the key words LOT or DEV after the model parameter:

[LOT[*t&d*]=<*value*>[%]] [DEV[*t&d*]=<*value*>[%]]

For example, this model statement specifies a 10% absolute tolerance to the forward beta of the transistor N1:

.MODEL N1 NPN (BF=300 LOT=10%)

In this example, for a worst case distribution, each transistor using the N1 model statement has a forward beta of either 270 or 330. If a Gaussian distribution were used, a random value would be selected from a Gaussian distribution having a standard deviation of 30. If a uniform distribution were used, a random value would be selected from a uniform distribution having a half-width of 30.

This example specifies a 1% relative tolerance to the BF of the N1 model:

.MODEL N1 NPN (BF=300 DEV=1%)

The DEV value specifies the relative percentage variation of a parameter. A relative tolerance of 0% implies perfect tracking. A 1.0% DEV tolerance implies that the BF of each N1 device is the same to within +- 1.0% for a worst case distribution. DEV tolerances require the use of private libraries, regardless of the state of the PRIVATEANALOG or PRIVATEDIGITAL flags. These flags are set in **Options / Global Settings**.

This sample specifies a 10% absolute and 1% relative BF tolerance:

.MODEL N1 NPN (BF=300 LOT=10% DEV=1%)

In this example, assuming a worst case distribution, the first N1 model is randomly assigned one of the two values 270 or 330. These two values are calculated from the mean value of 300 and 10% LOT tolerance as follows:

BF = 270 = 300 - .1·(300)
BF = 330 = 300 +.1·(300)

Suppose that the LOT toleranced BF value was randomly chosen to be 330. Then all N1 transistors, including the first, are randomly given one of these values, based upon the 1% DEV tolerance:

327 = 330 - .01·300
333 = 330 + .01·300

If the LOT toleranced BF value had been randomly chosen to be 270, then all N1 transistors, including the first, would be randomly given one of these values, based upon the 1% DEV tolerance:

267 = 270 - .01·300
273 = 270 + .01·300

Assuming a worst case distribution, all BF values in any particular run would be chosen from the set {267, 273, 327, 333}.

Resistors, capacitors, and inductors must be toleranced through their multiplier model parameter. This example provides a 10% LOT tolerance and a 1% DEV tolerance for a resistor.

.MODEL RMOD RES (R=1 LOT=10% DEV=1%)

Any resistor that uses the RMOD model will be toleranced, since the toleranced R value will multiply its resistor value.

[*t&d*] specifies the tracking and distribution, using the following format:

[/<*lot#*>][/<*distribution name*>]

These specifications must follow the keywords DEV and LOT without spaces and must be separated by "/".

<*lot#*> specifies which of ten random number generators, numbered 0 through 9, are used to calculate parameter values. This lets you correlate parameters of an individual model statement (e.g. RE and RC of a particular NPN transistor model) as well as parameters between models (e.g. BF of NPNA and BF of NPNB). The DEV random number generators are distinct from the LOT random number generators. Tolerances without <*lot#*> get unique random numbers.

<*distribution name*> specifies the distribution. It can be any of the following:

Keyword	**Distribution**
UNIFORM	Equal probability distribution
GAUSS	Normal or Gaussian distribution
WCASE	Worst case distribution

If a distribution is not specified in [*t&d*], the distribution specified in the Monte Carlo dialog box is used.

To illustrate the use of <*lot#*>, suppose we have the following circuit:

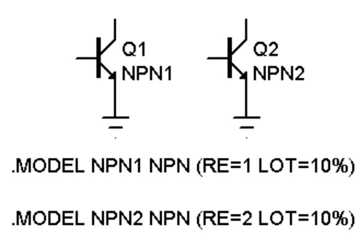


Figure 16-1 Uncorrelated RE values

In this example, Q1's RE value will be uncorrelated with Q2's RE value. During the Monte Carlo runs, each will receive random uncorrelated tolerances.

Now consider this circuit:

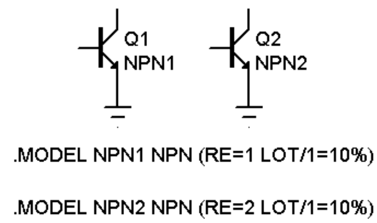


Figure 16-2 Using <*lot#*> to correlate RE values

Here, the presence of LOT/1 in both RE tolerance specs forces the LOT tolerance of the RE values to be the same. The values themselves won't be the same since their nominal values (1.0 and 2.0) are different.

DEV can also use [*t&d*] specifications. Consider this circuit.

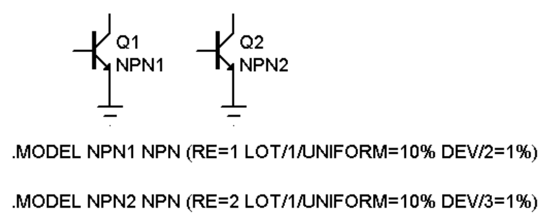


Figure 16-3 Using <*lot#*> in DEV and LOT

Here the LOT toleranced RE values will track perfectly, but when the DEV tolerance is added, the values will be different due to the use of different generators for DEV.

Tolerancing symbolic parameters

Symbolic parameters, those created with a .DEFINE statement, may also be toleranced. The format is as follows:

.DEFINE [{*lotspec devspec*}] <*varname*> <expr>

where the format of *lotspec* is the same as for other parameters:

[LOT[*t&d*]=<*value*>[%]]

and the format of *devspec* is similar:

[DEV[*t&d*]=<*value*>[%]]

[*t&d*] specifies the tracking and distribution, using the usual format:

[/<*lot#*>][/<*distribution name*>]

For example,

.DEFINE {LOT/1/GAUSS=10%} RATE 100

This defines a variable called RATE that has a Gaussian distribution with a LOT tolerance of 10% and its tolerances are based on random number generator 1.

Here is another example:

.DEFINE {LOT/1/GAUSS=10% DEV/2/UNIFORM=2%} VOLTAIRE 100

This defines a variable called VOLTAIRE that has a Gaussian distribution with a LOT tolerance of 10% and a uniform distribution with a DEV tolerance of 2%. Its LOT tolerance is based on random number generator 1 and its DEV tolerance is based on random number generator 2.

Tolerances and public vs. private libraries

In order to accomplish relative DEV tolerancing, it is necessary to have private libraries. Consider this case:

.MODEL N1 NPN (BF=300 LOT=10% DEV=1%)

Because of the DEV tolerance, each instance of a BJT using the N1 model will have a unique BF. This can only happen with private libraries. Thus, when a model uses a DEV tolerance, the PRIVATEANALOG or PRIVATEDIGITAL flags are enabled, forcing the use of a private library for all parts which use the model. The flags are not affected for other models.

To summarize, for all parts which use the model statement:

If DEV is used, then all devices have their own private model parameter set, regardless of the PRIVATEANALOG or PRIVATEDIGITAL flag settings, and may have different parameter values if the tolerances are not zero.

If DEV is not used, and PRIVATEANALOG or PRIVATEDIGITAL are disabled, then all devices that use the same model name will have the same parameter values, since public libraries are being used.

If DEV is not used, and PRIVATEANALOG or PRIVATEDIGITAL are enabled, then all devices that use the same model name may have different parameter values, if the tolerances are not zero.

This table summarizes how the parameters of two parts using the same model vary depending upon DEV use and the PRIVATE flags.

	PRIVATE	**PUBLIC**
DEV USED	UNIQUE	UNIQUE
DEV NOT USED	UNIQUE	SAME

This table summarizes how the parameters of two parts using the same model vary depending upon LOT use and the PRIVATE flags.

	PRIVATE	**PUBLIC**
LOT USED	UNIQUE	SAME
LOT NOT USED	SAME	SAME

Distributions

The actual values assigned to a toleranced parameter depend not only on the tolerance, but on the distribution as well.

A worst case distribution places all values at the extremes of the tolerance band. There are only two values:

Min = Mean - Tolerance
Max = Mean + Tolerance

The mean value is the model parameter value.

A uniform distribution places values equally over the tolerance band. Values are generated with equal probability over the range:

From *Mean* - *Tolerance* to *Mean* + *Tolerance*

A Gaussian distribution produces a smooth variation of parameters around the mean value. Values closer to the mean are more likely than values further away. The standard deviation is obtained from the tolerance by this formula:

Standard Deviation = Sigma = (Tolerance/100)·Mean/ SD

SD (from the Global Settings dialog box) is the number of standard deviations in the tolerance band. Thus, the standard deviation is calculated directly from the tolerance value. The value chosen depends upon how much of a normal population is to be included in the tolerance band. Here are some typical values:

Standard deviations	Percent of population
1.0	68.0
1.96	95.0
2.0	95.5
2.58	99.0
3.0	99.7
3.29	99.9

If a supplier guarantees that 99.9% of all 10% resistors are within the 10% tolerance, you would use the value 3.29. Using a Gaussian distribution, a 1K -10% resistor may have a value below 900 ohms or above 1100 ohms. The probability would be less than 0.1% for an SD of 3.29, but it could happen.

Performance functions

MC7 saves all X and Y expression values of each plotted expression at each data point for each run, so you can create histograms using expressions comprised of the functions after the runs are done. For example if you plotted the curve V(OUT), you could do a histogram of the expression:

Rise_Time(V(3),1,1,1,2) + Fall_Time(V(3),1,1,1,2)

Performance function expressions reduce an entire curve to a *single* number that captures an important behavioral characteristic for one particular run. Individual numbers are then combined to form a population which is statistically analyzed and its histogram is displayed. Ideally, the histograms and population statistics will reveal expected variations in the performance function expression when the circuit is manufactured.

The performance functions are described in greater detail in Chapter 26, "Performance Functions."

Options

The Monte Carlo options dialog box provides these choices:

- **Status:** Monte Carlo analysis is enabled by selecting the On button. To disable it, click the Off button.

- **Number of Runs:** The number of runs determines the confidence in the statistics produced. More runs produce a higher confidence that the mean and standard deviation accurately reflect the true distribution. Generally, from 30 to 300 runs are needed for a high confidence. The maximum is 30000 runs.

- **Report When:** This field specifies when to report a failure. The routine generates a failure report in the numeric output file when the Boolean expression in this field is true. The field must contain a performance function specification. For example, this expression

  rise_time(V(1),1,2,0.8,1.4)>10ns

  would generate a report when the specified rise time of V(1) exceeded 10ns. The report lists the model statement values that produced the failure. These reports are included in the numeric output file (NAME.TNO for transient, NAME.ANO for AC, and NAME.DNO for DC) and can be viewed directly. They are also used by the **Load MC File** item in the **File** menu to recreate the circuits that created the performance failure.

- **Distribution to Use:** This specifies the default distribution to use for all LOT and DEV tolerances that do not specify a distribution with [*t&d*].

  - *Gaussian distributions* are governed by the standard equation:

    $$f(x) = e^{-.5 \cdot s \cdot s}/\sigma(2 \cdot \pi)^{.5}$$

    Where $s = x\text{-}\mu/\sigma$ and μ is the nominal parameter value, σ is the standard deviation, and x is the independent variable.

  - *Uniform distributions* have equal probability within the tolerance limits. Each value from minimum to maximum is equally likely.

  - *Worst case distributions* have a 50% probability of producing the minimum and a 50% probability of producing the maximum.

Chapter 17 The MODEL Program

What's in this chapter

This chapter describes the operation of the MODEL program.

MODEL is designed to make the creation of device model parameters from data sheet values fast, easy, and accurate. It is an interactive, optimizing curve fitter that takes numbers from data sheet graphs or tables and produces an optimized set of device model parameters. MODEL uses Powell's direction-set method for optimization. This robust algorithm is well suited to the type of error minimization required in device modeling. For those interested in the algorithm, see *Numerical Recipes in C, The Art of Scientific Computing*, published by Cambridge University Press.

As a general guideline, two to five data pairs should be taken from the appropriate data sheet graphs. If the graphs are not available, use a single data pair from the specification tables. If the specification table is missing from the data sheet, use the default values supplied. Use typical, room temperature values. After entering the data, use the **Initialize** option under the **Run** menu for a first estimate, then use the **Optimize** option under the **Run** menu.

How to start MODEL

MODEL is a stand alone program and can be started by double-clicking on its icon from the Windows Program Manager. It may also be invoked from within MC7 using the **Windows** menu.

MODEL may also be invoked with the name of a data file:

MODEL datafilename

This form runs the MODEL program and loads the specified data file.

The MODEL window

The MODEL window houses all of the functions necessary to create and access data files and to produce optimal models. Its display looks like this:

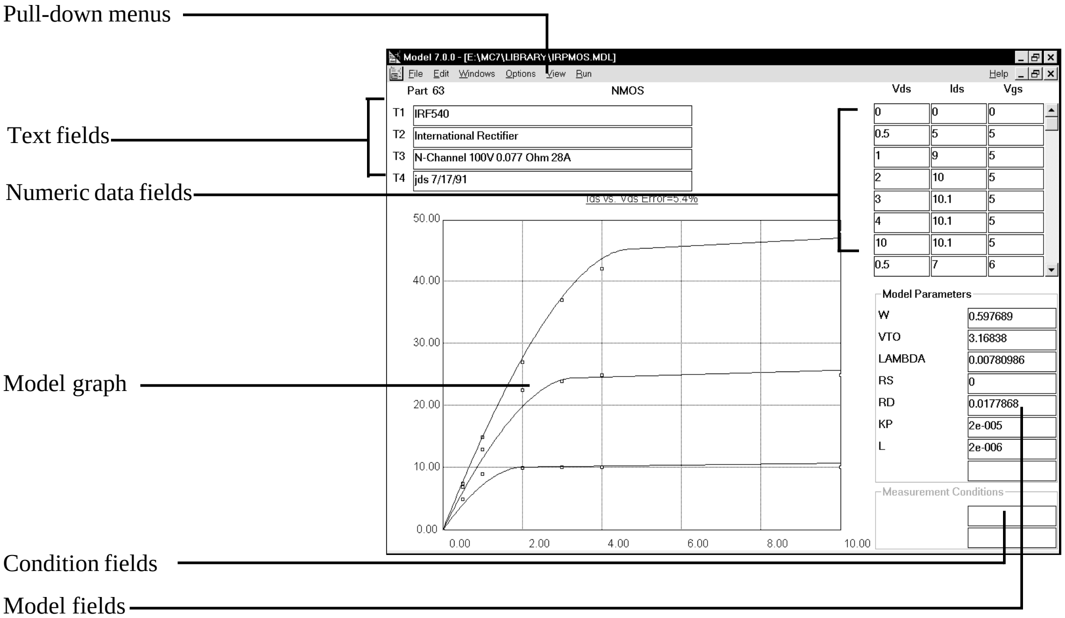


Figure 17-1 The Main Display

The principal components of the MODEL window are as follows:

Text fields: There are four Text fields: 'T1', 'T2', 'T3', and 'T4'. The 'T1' and 'T3' fields are imported to MC7 model libraries. The 'T1' field holds the part name and is used in sorting. The other text fields serve only as additional documentation.

Numeric data fields: There are from one to three data fields, depending upon the device type and graph. From one to fifty data sets may be entered in the fields. The data is usually obtained from a data sheet graph. If the graph is not available, a single data point may be taken from the specification tables. If the specification value is not available, no data is entered and default values are used for the corresponding model parameters.

Model Graph: The model graph shows a plot of the curve for the model parameters in the Model fields. It also plots the numeric data points, if any are entered by the user. The quality of the fit can be judged by noting how close the data points match the model curve. A more rigorous estimate is shown in the error display just above the text fields. This error figure is the average percentage error of all data points.

Model fields: Model parameters are changed by initialization or optimization, but may also be directly edited. This is sometimes useful to gauge the effect of a model parameter change. Occasionally, it is useful to manually initialize the model parameter values to obtain a better fit than can be obtained by using initialization.

Condition fields: These fields hold the value of any external test condition used in the measurement from which the data points were obtained.

The File menu

MODEL uses data files with the extension "MDL". This menu provides functions for managing the data files, SPICE text model files, and MC7 device model files.

- **New: (CTRL + N)** This creates a new data file containing a single device.

- **Open: (CTRL + O)** This loads an existing data file from disk.

- **Save: (CTRL + S)** This saves the current data file to disk.

- **Save As:** This lets you save the current data file to disk under a new name.

- **Paths...:** This lets you specify the default path(s) where the program loads and saves its files.

- **Create SPICE File:** This saves the parameter values as model statements in a text file, using a file name extension of "LIB". They can be accessed in this form by the MC7 simulator, but not by the MC7 Model Editor.

- **Create Model Library:** This saves the parameter values as a binary data file, using a file name extension of "LBR". They may be accessed in this form by both the MC7 simulator and by the MC7 Model Editor. This form is the preferred form for use by MC7, since it provides faster access to the part model values when MC7 is searching the libraries in preparation for a run.

- **Revert:** This restores the currently displayed file to the version stored on disk. If you have made many edits to the current file, and want to abandon them and start with a fresh copy of the file, this is the command to use.

- **Close (CTRL+ F4):** This command closes the current data file.

- **Merge:** This merges the current data file with one from the disk. The results are displayed in the current data file, but are only saved to disk if requested by the user with a Save or Save As command or when the file is unloaded.

- **Sort:** This option sorts the current data file using the T1 field.

- **File Name:** Recently opened files are listed here.

- **Exit: (ALT+ F4)** This command exits the MODEL program.

The Edit menu

This menu provides the commands for editing the data file:

- **Undo: (CTRL + Z)** Any change to a text field may be reversed with the Undo command. Undo will only reverse the last change.

- **Cut: (CTRL + X)** This command deletes the selected text and copies it to the clipboard.

- **Copy: (CTRL + C)** This command copies selected text to the clipboard.

- **Paste: (CTRL + V)** This command copies the contents of the clipboard starting at the current cursor position.

- **Clear: (DELETE)** This command deletes the selected items without copying them to the clipboard.

- **Select All: (CTRL + A)** This command selects all text in the field where the text cursor is.

- **Copy Front Window to Clipboard:** This copies all or part of the visible portion of the front window in bitmap graphics format to the clipboard.

- **Change Polarity:** This command lets you change the polarity of the displayed device. For example, you can change the polarity of a bipolar transistor from NPN to PNP, or MOSFET from PMOS to NMOS.

- **Add Part:** This option lets you add a new part to the current file. The part type is selected from a submenu and added to the end of the file.

- **Delete Data: (CTRL + D)** This option lets you delete the data pair or data triplet in the data field where the cursor is presently located. It only affects the Numeric Data fields, not the text fields. It is enabled only when the text cursor is located in one of the Numeric Data fields. Note that there is no Copy Part or Delete Part command on this menu. These functions are part of the Parts List dialog box. You can delete one or more parts by selecting them, then cutting or clearing them. You can copy one or more parts by selecting them, copying them to the clipboard, then pasting them into the file.

The Windows menu

This menu provides commands for arranging the open files:

- **Cascade: (SHIFT + F5)** This command cascades open files in an overlapping manner.

- **Tile Vertical: (SHIFT + F4)** This command vertically tiles the open files in a non-overlapping manner.

- **Tile Horizontal:** This command horizontally tiles the open files in a non-overlapping manner.

- **Arrange Icons:** This command neatly arranges any minimized file icons.

- **File names:** This lists the open files by name. Choosing one of the files makes it the active or front window.

The Options menu

This menu provides the following choices:

- **Help Bar:** This toggles the display of the Help Bar, a one line window which provides short descriptions of the item at the current mouse location.

- **Preferences:** These are simple operational preferences.

    - **File Warning:** This provides a warning when closing edited files.

    - **Sound:** This controls the use of sound for warnings.

    - **Quit Warning:** This provides a warning when quitting.

    - **Time Stamp:** This adds a time stamp at the top of the SPICE model statement output file.

    - **Date Stamp:** This adds a date stamp at the top of the SPICE model statement output file.

- **Global Settings:** The optimizer must achieve these limits before stopping.

    - **Maximum Relative Per-iteration Error:** If the relative difference in the RMS error function from one iteration to the next drops below this value, the optimization will stop. This value is typically 1u to 1m.

    - **Maximum Percentage Per-iteration Error:** If the percentage difference in the RMS error function from one iteration to the next drops below this value, optimization will stop. This value is typically 1u to 1m.

    - **Maximum Percentage Error:** If the actual percentage error of the RMS error function drops below this value, optimization will stop. This value is typically 0.1 to 5.0.

- **Model Defaults**
  This accesses an editor which lets you change the minimum, initial, and maximum values for all model parameters. The minimum and maximum serve as limits on the value of optimized parameters during the optimization process. The initial value is used for initialization prior to optimization.

- **Color Preferences**

This lets you select the color of the curve, data point icons, graph grid, graph background, and numeric scale.

- **Auto Scale (F6)**

This command automatically scales the plot.

- **Manual Scale (F9)**

This option lets you manually change the plot scale.

- **Step Model Parameters**

This option lets you step any of the model parameters to see their effect on the curve. It is a tool for exploring the effect of different model parameters.

The View menu

This menu lets you access the different graphs for each part and the parts themselves. The data file can be viewed as a two-dimensional structure. The graphs are arranged from left to right. The parts are arranged from top to bottom. The CTRL + ARROW keys provide a kind of two-dimensional access to the file:

Previous Graph:	CTRL + LEFT ARROW shows the graph to the left.
Next Graph:	CTRL + RIGHT ARROW shows the graph to the right.
Previous Part:	CTRL + UP ARROW shows the part above.
Next Part:	CTRL + DOWN ARROW shows the part below.

The graphs and parts are accessed with these options:

- **Parts List (CTRL + L):** This command displays a list box showing all of the parts in the current file. Double-clicking on one of the part names displays the part and its data. The menu lets you cut, copy, paste, and delete parts.

  To select a part, click on it. To select more than one part, drag the mouse over the part names, or use CTRL + mouse click to toggle the selection state of single parts, or SHIFT + mouse click to toggle the selection state of many parts.

  To delete one or more parts, first select them, then cut (CTRL + X) or clear (Del) them.

  To copy one or more parts, first select them, then copy them to the clipboard (CTRL + C), then paste (CTRL + V) them to the end of the file.

- **Find Part (CTRL + F):** This searches the current file for a part name in the T1 field.

- **Previous Part (CTRL + UP ARROW):** This shows the prior part.

- **Next Part (CTRL + DOWN ARROW):** This shows the next part.

- **First Part (CTRL + HOME):** This shows the first part in the data file.

- **Last Part (CTRL + END):** This shows the last part in the data file.

- **Previous graph (CTRL + LEFT ARROW):** This shows the prior graph.

- **Next Graph (CTRL + RIGHT ARROW):** This shows the next graph.

- **First Graph (CTRL + SHIFT + LEFT ARROW):** This shows the first graph of the current part.

- **Last Graph (CTRL + SHIFT + RIGHT ARROW):** This shows the last graph of the current part.

- **All Graphs:** This mode shows all graphs for the part. When you select this option, all graphs are displayed. A list of the graphs is presented at the end of the menu, together with a check mark. You can selectively add or delete graphs by clicking on the graph name. When more than one graph is shown, the data fields shown are for the selected graph. The selected graph is the one whose title is underlined. You can change the selected graph by clicking on the title bar. When you do, the screen shows the appropriate data fields for the graph.

- **One Graph at a Time:** This mode shows only one graph at a time.

- **Graph Names:** This portion lists each of the graphs for the part. If the **All Graphs** option is enabled, clicking on one of the graph names toggles its display status. If the **One Graph at a Time** option is enabled, clicking on one of the graph names selects it for display.

The Run menu

This menu provides control over the initialization and optimization functions:

- **Initialize: (CTRL+I)** This initializes the model parameter values for the displayed graph of the part. This is usually the prelude to optimizing.

- **Optimize: (CTRL+T)** This optimizes the model parameter values to fit the supplied data points. Optimizing is done for the selected graph of the current part. Optimization is done by minimizing the RMS difference between the data points and the plot values predicted by a particular set of parameters.

- **Initialize and Optimize All:** This first initializes and then optimizes each part in the data file. If the parts have never been optimized, this is a convenient way of optimizing all parts in the data file.

- **Optimize All:** This optimizes each part in the file without initialization. If the parts have been optimized one or more times, and you wish to tighten the optimization tolerances to produce a better fit, or possibly you have changed a few data values in the file, but can't remember which have been changed, this option provides a convenient way to re-optimize the entire data file.

A bipolar transistor example

To illustrate the use of MODEL, we'll use the 2N3903 transistor. The example is based upon the product data sheet which begins on page 2-3 in the data book, *Motorola Small-Signal Transistors, FETs, and Diodes Device Data Rev 5*.

Begin by creating a new file. Select **New** from the **File** menu. Use the name "NEW1.MDL". Click on the NPN type option. Click OK. The program creates a new file with one part of the selected type. The cursor is initially in the T1 text field. Type "2N3903". Enter any text you wish in the remaining three text fields.

Now you're ready to enter data. Locate Figure 17, the "ON" VOLTAGES graph, on page 2-7 of the Motorola handbook. From the two Vbe vs. Ic plots choose the Vbe(sat) @ Ic/Ib = 10 curve. Press the Tab key or use the mouse to move the cursor to the Ic data field. Enter these data sets from the graph.

Ic	**Vbe**
.001	.65
.01	.74
.025	.80
.1	.93

Press CTRL + I to initialize and CTRL + T to optimize the values. The results look like this:

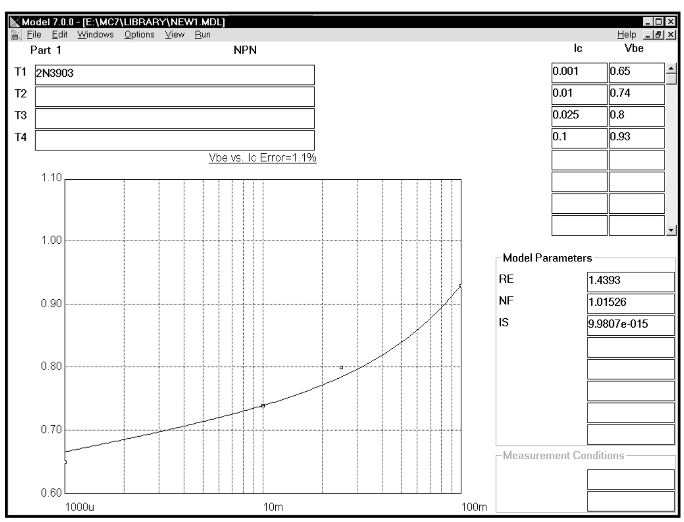


Figure 17-2 The Vbe vs. Ic plot

The model parameters RE, NF, and IS are optimized to produce a fit to the Vbe vs. Ic curve of about 1%. This is a slightly better than average error for this type of graph.

Press CTRL + RIGHT ARROW. This displays the Hoe vs. Ic graph. Use the maximum value of 40 umhos from the table for Hoe.

Press CTRL + RIGHT ARROW. This displays the Beta vs. Ic graph. Locate the graph, Figure 15, DC Current Gain, on page 2-6. Select the 25° plot. Enter the following data points from the graph:

Ic	Beta	Ic	Beta
.0001	44	.030	72
.001	77	.050	50
.005	98	.100	27
.010	100		

Move the cursor to the Measurement Conditions field and enter 1.0 for Vce. Press CTRL + I to initialize and CTRL + T to optimize. The results look like this:

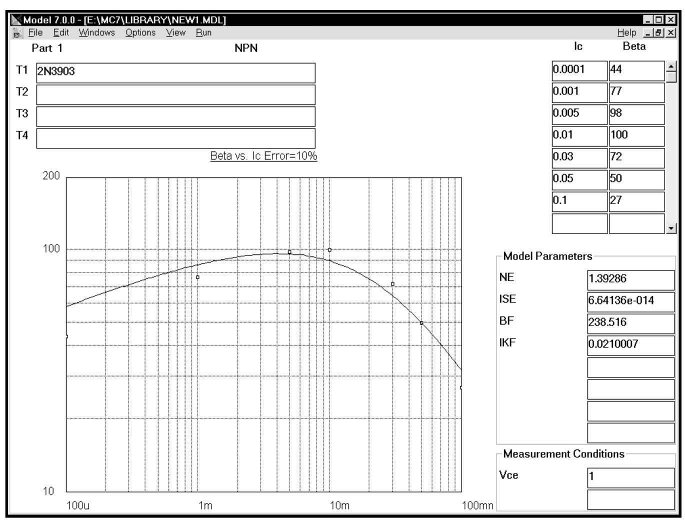


Figure 17-3 The Beta vs. Ic Plot

The model parameters NE, ISE, BF, and IKF are optimized to fit the plot to within an error of about 11%. A typical error range for this plot is 1% to 20%. Press CTRL + RIGHT ARROW and MODEL will display the next graph, Vce vs. Ic.

From the same "ON" VOLTAGES graph used earlier, select the Vce(Sat) curve. From the curve enter the following data points:

Ic	**Vce**
.001	.1
.010	.11
.05	.2
.10	.35

Move the cursor to the Measurement Conditions field and enter 10 for Ic/Ib. Initialize and optimize the values. The results look like this:

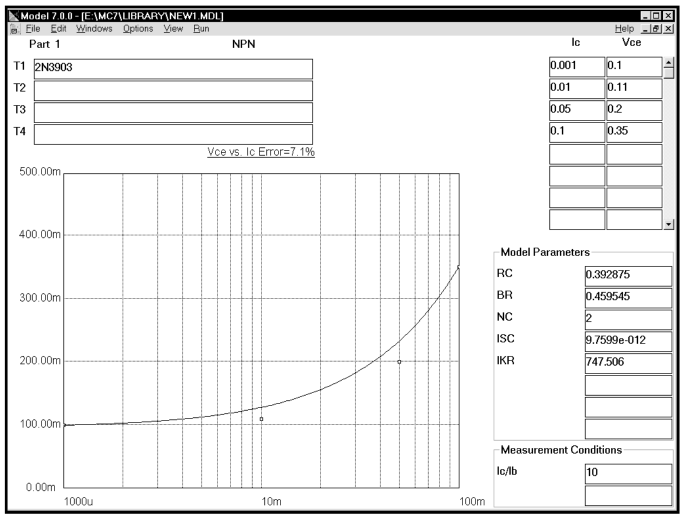


Figure 17-4 The Vce vs. Ic Plot

The model parameters RC, BR, NC, ISC, and IKR are optimized to produce a fit to the Vce vs. Ic plot of about 7%. This is a good fit for the Vce plot. Generally an error range of 5% to 25% is to be expected here.

Press CTRL + RIGHT ARROW to select the next plot, Cob vs. Vcb. From Cobo plot in Figure 3, CAPACITANCE graph, on page 2-4 enter these values.

Vcb	**Cob**
0.10	3.5pf
1.00	2.7pf
10.0	1.7pf

Initialize and optimize and the results look like Figure 17-5.

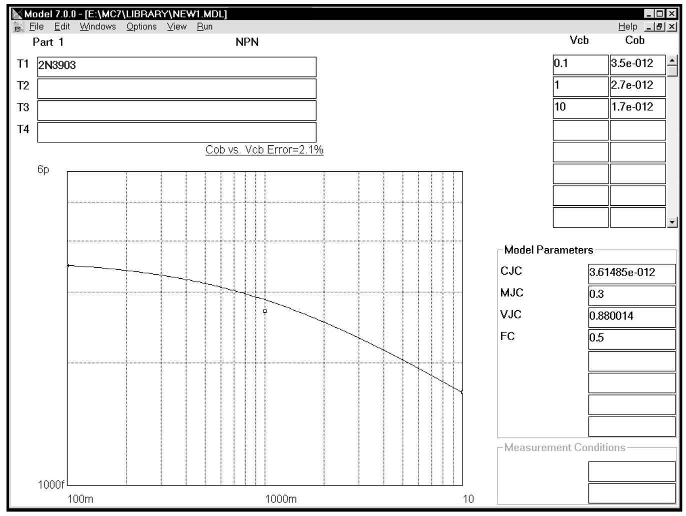


Figure 17-5 The Cob vs. Vcb Plot

Press CTRL + RIGHT ARROW to select the next plot Cib vs. Veb. From Cibo plot in Figure 3, CAPACITANCE graph, on page 2-4, enter these values.

Veb	Cib	Veb	Cib	Veb	Cib
.10	4.2pf	1.0	3.3pf	5.0	2.5pf

Initialize and optimize and the results look like this:

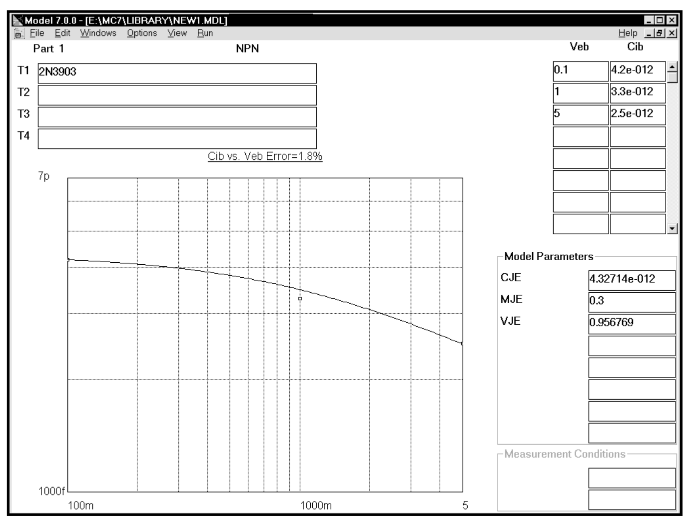


Figure 17-6 The Cib vs. Veb Plot

Press CTRL + RIGHT ARROW to select the next plot, TS vs. Ic. From the Ic/Ib = 10 curve in Figure 7, STORAGE TIME, graph on page 2-5, enter these values for collector current and storage time:

Ic	**TS**
1m	100n
10m	130n
200m	53n

Set the Measurement Condition field to 10. Initialize and optimize and the results look like this:

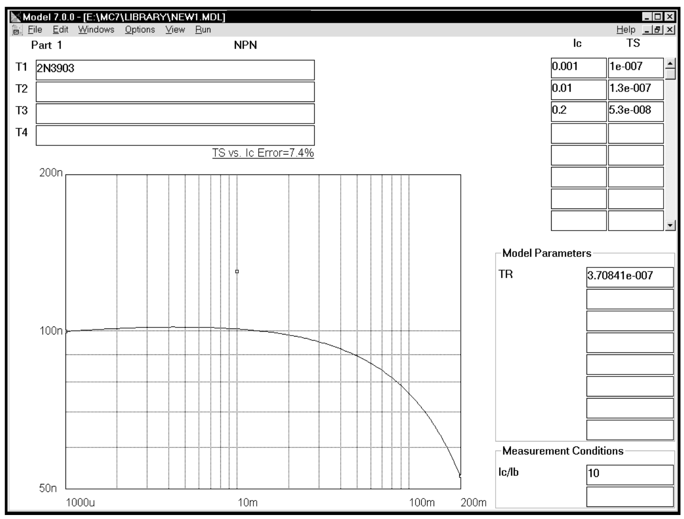


Figure 17-7 The TS vs. Ic Plot

The model parameter TR is optimized to produce an average error of under 10%.

Press CTRL + RIGHT ARROW to select the next plot, FT vs. Ic. From the Small-signal specification tables enter this data point.

Ic	**FT**
10m	250E6

Enter 10.0 for Vce in the Measurement Conditions field. Initialize and optimize and the results look like this:

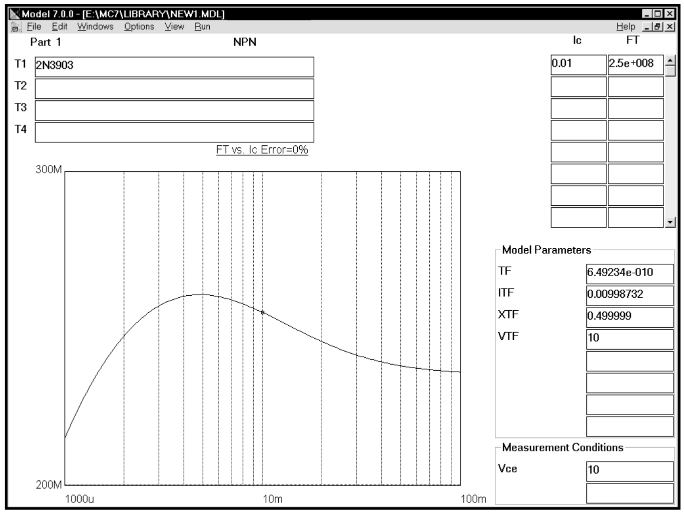


Figure 17-8 The FT vs. Ic Plot

The model parameters TF and ITF are optimized to produce a near perfect fit to the single data point. Standard, unoptimized XTF and VTF values are used.

This completes the example for the 2N3903. Probably because it is such a popular device, it has a fairly good selection of graphs and specification values to use. Other parts may not be well documented. In these cases, you have three choices:

1. Measure the data sheet values on a sample of actual parts.
2. Use the default model parameter values.
3. Use a part from a different manufacturer with better documentation.

Save the results in a model file using the **Save** option from the **File** menu.

The final step is to create a model library file that MC7 can use. Select the **Create Model Library** option from the **File** menu. The program will present a Save As dialog box and let you specify the path and name of the model library file to use. Click on the OK button to accept the default name NEW1.LBR. The library file is now saved and ready for use by MC7. Finally, be sure to enter the line .LIB "NEW1.LBR" into the NOM.LIB text file to tell MC7 about the new file.

Parameters for other device types are created in a similar fashion. A summary of each graph and some guidelines are included in the following pages.

Diode graphs

Title	**Forward current vs. Forward voltage**
Purpose	This screen estimates IS, N, and RS.
Input	One or more pairs of If and Vf values.
Output	Model values for IS, N, and RS.
Equations	$Vf = VT \cdot log(If/IS)+If \cdot RS$
Guidelines	Use data from the If vs. Vf graphs. If unavailable, use typical values from the tables. Use data from both the low and high current ranges. The low-current data will determine the value of IS and NF, and the high-current data points will determine RS.

Title	**Capacitance C vs. Reverse voltage**
Purpose	This screen estimates CJO, M, VJ, and FC.
Input	One or more pairs of Cj and Vr values.
Output	Model values for CJO, M, VJ, and FC.
Equations	$C = CJO/(1+VR/VJ)^{M}$
Guidelines	Use data from the C vs. Vr graphs. Vr is the value of the reverse voltage and is always positive.

Title	**Id vs Vrev**
Purpose	This screen estimates RL.
Input	One or more pairs of Irev and Vrev values.
Output	Model value for RL.
Equations	Irev = Vrev/RL (the breakdown portion is ignored)
Guidelines	Use data from the Irev vs. Vrev graphs. If unavailable, use typical values from the tables. RL models the main reverse leakage current component. BV is not optimized.

Title	**Trr vs. Ir/If ratio**
Purpose	This screen estimates TT, the transit time parameter.
Input	One or more pairs of Trr and Ir/If values. Ir/If is the ratio of the forward and reverse base currents used to measure Trr.
Output	Model value for TT.
Equations	$trr = tt \cdot log10(1.0+1.0/ratio)$
Guidelines	Use data from the Trr vs. Ir/If ratio graphs. If unavailable, use typical values from the tables. If the typical value is not available, use an average of the min and max values.

Bipolar transistor graphs

Title	**Vbe vs. Ic**
Purpose	This screen estimates IS, NF, and RE.
Input	One or more pairs of Vbe and Ic values.
Output	Model values for IS, NF, and RE.
Equations	Vbe=VT·NF·log(Ic/IS)+Ic·RE
Guidelines	Use data from the VbeSat vs. Ic graphs. If unavailable, use typical values.

Title	**Hoe vs. Ic**
Purpose	This screen estimates the forward Early voltage, VAF.
Input	One or more pairs of Hoe and Ic values.
Conditions	The value of Vce.
Output	Model values for VAF.
Equations	Hoe = Ic / (VAF+Vce-0.7)
Guidelines	Use the Hoe vs. Ic graphs. If unavailable, use typical values.

Title	**Beta vs. Ic**
Purpose	This screen estimates the parameters, NE, ISE, BF, and IKF. These parameters model the low-current recombination and high-level injection effects that produce a drop-off in the forward beta.
Input	One or more pairs of Beta and Ic values.
Conditions	The value of Vce.
Output	Model values NE, ISE, BF, and IKF.
Equations	BF= f(Ic) = simulated tabular function of BF vs. Ic
Guidelines	Use the Beta vs. Ic graph. If unavailable, use typical table values.

Title	**Vce vs. Ic**
Purpose	This screen estimates NC, ISC, BR, IKR, and RC. These model the low-current recombination and high-level injection effects that cause reverse beta drop-off. The collector resistance is also estimated.
Input	One or more pairs of Vce and Ic values.
Conditions	The value of the Ic/Ib ratio used in the measurement.
Output	Model values NC, ISC, BR, IKR, and RC.
Equations	Vce = (simulated tabular function of Vce vs. Ic)+Ic·(RC+RE)
Guidelines	Use the Vce vs. Ic graphs. If unavailable, use typical table values.

Title	**Cob vs. Vcb**
Purpose	This screen estimates CJC, MJC, VJC, and FC.
Input	One or more pairs of Cob and Vcb values.
Output	Model values for CJC, MJC, VJC, and FC.
Equations	$Cob = CJC/(1+Vcb/VJC)^{MJC}$
Guidelines	Use the Cob vs. Vcb graphs. Vcb is the value of the collector-base voltage and is always positive.

Title	**Cib vs. Veb**
Purpose	This screen estimates CJE, MJE, and VJE.
Input	One or more pairs of Cib and Veb values.
Output	Model values for CJE, MJE, and VJE.
Equations	$Cib = CJE/(1+Veb/VJE)^{MJE}$
Guidelines	Use the Cib vs. Veb graphs. Veb is the value of the emitter-base voltage and is always positive.

Title	**TS vs. Ic**
Purpose	This screen estimates Tr, the reverse transit time value.
Input	One or more pairs of TS and Ic values.
Conditions	The value of the Ic/Ib ratio used in the measurement.
Output	Model value for Tr.
Equations	$ar = br/(1.0+br)$, $af=bf/(1.0+bf)$ $k1 = (1.0-af\cdot ar)/ar$, $k2=(af/ar)\cdot TF$ $TS = ((Tr+k2)/k1)\cdot \ln(2.0/((Ic/Ib)/bf+1.0))$
Guidelines	Use the TS vs. Ic graphs. Use a typical value. If the typical value is unavailable, use an average of the min and max values.

Title	**Ft vs. Ic**
Purpose	This screen estimates TF, ITF, XTF, and VTF.
Input	One or more pairs of Ft and Ic values.
Conditions	The value of Vce.
Output	Model values for TF, ITF, XTF, and VTF.
Equations	$vbe=VT\cdot N\cdot \ln(Ic/ISS)$, $vbc = vbe - Vce$ $atf=1+XTF\cdot(Ic/(Ic+ITF))^2\cdot e^{(vbc/(1.44\cdot VTF))}$ $tf =TF\cdot(atf+2\cdot(atf-1)\cdot ITF/(Ic+ITF)+VT\cdot N\cdot(atf-1)/(1.44\cdot VTF))$ $fa =(1-vbc/VAF)\cdot(1-vbc/VAF)$ $Ft =1/(2\cdot PI\cdot(tf/fa+VT\cdot N\cdot(cje+cjc\cdot(1+Ic\cdot RC/(VT\cdot N)))/Ic))$
Guidelines	Use data from the Ft vs. Ic graphs. If unavailable, use typical values from the tables. If the typical value is unavailable, use an average of the min and max values.

JFET graphs

Title	**Id vs. Vgs**
Purpose	This screen estimates the value of BETA, VTO, and RS.
Input	Values for Vgs and Id.
Output	Model values for BETA, VTO, and RS.
Equations	$Vgs = RS \cdot Id - VTO - sqrt(Id/BETA)$

Title	**Gos vs. Id**
Purpose	This screen estimates the value of LAMBDA.
Input	Enter values for Gos and Id.
Output	Model value for LAMBDA.
Equations	$Gos = Id \cdot LAMBDA$

Title	**Crss vs. Vgs**
Purpose	This screen estimates the value of CGD, PB, and FC.
Input	Enter values for Crss and Vgs.
Conditions	The value of Vds at which the capacitance was measured.
Output	Model values for CGD, PB, FC.
Equations	$Crss=CGS/(1-(Vds-Vgs)/PB)^{.5}$ $\{(Vds-Vgs)< FC \cdot PB\}$ $Crss=CGS/(1-FC)^{1.5} \cdot (1-FC \cdot 1.5+.5 \cdot (Vds-Vgs)/PB)$ $\{(Vds-Vgs)>=FC \cdot PB\}$

Title	**Ciss vs. Vgs**
Purpose	This screen estimates the value of CGS.
Input	Enter values for Ciss and Vgs.
Conditions	The value of Vds at which the capacitance was measured.
Output	Model value for CGS.
Equations	$Crss=Ciss+CDS/(1-Vgs/PB)^{.5}$ $\{Vgs< FC \cdot PB$ $Crss=Ciss+CDS/(1-FC)^{1.5} \cdot (1-FC \cdot 1.5+.5 \cdot Vgs/PB)$ $\{Vgs>=FC \cdot PB\}$

Title	**Noise**
Purpose	This screen estimates the value of KF and AF.
Input	Enter the values of En and frequency.
Conditions	The value of Ids at which the measurement is made.
Output	Model values for KF and AF.
Equations	$vgs = VTO + Id \cdot RS + sqrt(Id/BETA)$ $gm = 2.0 \cdot BETA \cdot (vgs - VTO)$ $En = sqrt((8 \cdot k \cdot T \cdot gm)/3 + (KF \cdot ID^{AF}) / freq)/gm$

MOSFET graphs

All voltage and current values are entered as positive quantities for N-channel devices and negative quantities for P-channel devices.

Title	**Transconductance vs. Ids graph**
Purpose	This screen estimates KP, W, L, VTO, and RS.
Input	One or more pairs of Gfs and Ids values.
Output	Model values KP, RS, W, VT, L.
Equations	$beta = KP \cdot W/L$ $t1 = (2 \cdot Ids \cdot beta)^{1/2}$ $Gfs = t1 / (1+RS \cdot t1)$
Guidelines	Use data from the Gfs vs. Id curves. If unavailable, use typical values from the specification tables. Use data points from the highest current values to get the most accurate value of RS.

Title	**Static drain-source on resistance vs. Drain current**
Purpose	This screen estimates RD from the Ron vs. Id curves.
Input	One or more pairs of Ron and Id values.
Conditions	The value of Vgs.
Output	Model value for RD.
Equations	$beta = KP \cdot W/L$ $vgst = Vgs - VTO - Id \cdot RS$ $vds = vgst - (vgst^2 - 2 \cdot Id/beta)^{1/2}$ $RON = RD + RS + 1/(beta \cdot (vgst - vds))$
Guidelines	Use data from the Ron vs. Id curves. If unavailable, use typical values from the tables. Use low current values for the best results.

Title	**Output Characteristic Curves**
Purpose	This screen estimates all of the principal model values except the capacitance values. It uses the already estimated values of W, VTO, RD, RS, LAMBDA, KP, and L as an estimate and optimizes the fit of the characteristic Id vs Vds curves to the user data points.
Input	Triplets of Ids, Vds, and Vgs values.
Output	Model values for W, VTO, RD, RS, LAMBDA, KP, and L. KP and L are not optimized but are used in the calculation.
Equations	$Ids=0.0 \quad Vgs<VTO$ $Ids=(KP \cdot W/L) \cdot (Vgs-VTO-.5 \cdot Vds) \cdot Vds \cdot (1+LAMBDA \cdot Vds) \quad Vgs-Vth>Vds$ $Ids=(.5 \cdot KP \cdot W/L) \cdot (Vgs-VTO)^2 \cdot (1+LAMBDA \cdot Vds) \quad Vgs-Vth<Vds$

Guidelines	If the previous screens have been used, do not initialize before optimizing. Otherwise, use the initialize option prior to optimizing. If the Output Characteristic Curves are not available, skip this screen and use the model values from the prior screens.
Title	**Idss vs Vds**
Purpose	This screen estimates RDS, the fixed resistor connected from drain to source. It models the drain-source leakage.
Input	One pair of Idss and Vds values.
Output	Model value for RDS.
Equations	RDS=Vds/Idss
Guidelines	Use data from the specification tables or from the graphs.
Title	**Cds vs Vds**
Purpose	This screen estimates the values of CBD, PB, and MJ.
Input	The values of Ciss, Coss, and Crss.
Output	Model values for CBD, PB, FC, and MJ.
Equations	$Cds= CBD / (1-Vds/PB)^{MJ}$
Guidelines	Use data from the specification tables or from the graphs.
Title	**Vgs vs Qg**
Purpose	This screen estimates the values of CGSO and CGDO.
Input	Enter Q1 and Q2. Q1 is the gate charge at the first breakpoint in the graph. Q2 is the gate charge at the second breakpoint.
Conditions	The values of VDS(or VDD) and ID for the measurement.
Output	Model values for CGSO and CGDO.
Equations	The program runs a circuit simulation and measures Vgs and Qgs.
Guidelines	Use data from the specification tables or from the graphs.
Title	**Gate Resistance**
Purpose	This screen estimates the value of RG, the gate resistance.
Input	Enter the value of Tf, the 90% to 10% fall time.
Conditions	The values of VDD and ID at which the measurement is made.
Output	Model value for RG.
Equations	The program runs a circuit simulation, measures the fall time and adjusts RG to fit the specified value of Tf.
Guidelines	Use data from the specification tables or from the graphs.

Opamp graphs

Title	**Screen 1**
Purpose	This screen provides for direct entry of the model parameters from the data sheets. The parameters are: LEVEL :Model level (1,2,3). Always use level 3. TYPE :1=NPN, 2=PNP, 3=NJFET input C :Compensation capacitor A :DC open-loop voltage gain ROUTAC :AC output resistance ROUTDC :DC output resistance VOFF :Offset voltage
Input	Values for LEVEL, TYPE, C, A, ROUTAC, ROUTDC, and VOFF.
Output	Values for LEVEL, TYPE, C, A, ROUTAC, ROUTDC, and VOFF.

Title	**Screen 2**
Purpose	This screen provides for direct entry of the model parameters from the data sheets. The parameters are: IOFF :Input offset current SRP :Positive slew rate (V/Sec) SRN :Negative slew rate (V/Sec) IBIAS :Input bias current VEE :Negative power supply VCC :Positive power supply VPS :Positive voltage swing
Input	Values for IOFF, SRP, SRN, IBIAS, VEE, VCC, and VPS.
Output	Values for IOFF, SRP, SRN, IBIAS, VEE, VCC, and VPS.

Title	**Screen 3**
Purpose	This screen provides for direct entry of the model parameters from the data sheets. The parameters are: VNS :Negative voltage swing CMRR :Common mode rejection ratio GBW :Gain bandwidth PM :Phase margin PD :Power dissipation IOSC :Output short-circuit current
Input	Values for VNS, CMRR, GBW, PM, PD, and IOSC.
Output	Values for VNS, CMRR, GBW, PM, PD, and IOSC.

Core graph

Title	**Core B-H**
Purpose	This screen estimates the nonlinear magnetic core model values MS, ALPHA, A, C, and K. Model values for Area, Path, and Gap are entered directly from the data sheet table.
Input	Triplets of H, B, and Region values.
Output	Model values for MS, ALPHA, A, C, and K.
Equations	Jiles-Atherton state equations.
Guidelines	Enter the data from the B-H graph. H is entered in Oersteds, B in Gauss, and the Region value is 1, 2, or 3.

Region	Value	Otherwise known as:
H = 0 to Hmax	1.0	Initial permeability curve
H = Hmax to - Hmax	2.0	Top B-H curve
H = - Hmax to Hmax	3.0	Bottom B-H curve

If the Initial permeability curve is unavailable, skip it and enter values for regions 2 and 3. For the best results, enter an equal number of data points for each region. Sometimes only the top part of the B-H curve is given. In this case, select an equal number of data points from each of the given parts of the three regions.

Enter the data for each core material first. Once a particular material is modeled, copy the part and use the copy as a template to model parts that use the same material, but have different values of Area, Path, or Gap. This avoids duplication of the same B-H curve.

Chapter 18 Convergence

What's in this chapter

This chapter describes convergence, or the lack thereof. It explains the source of many non-convergence error messages and some of the remedies available.

Convergence defined

To do its work, Micro-Cap 7 must solve nonlinear equations. Neither people nor computers are able to solve these equations analytically, so they must be solved numerically. There are many techniques for numerically solving equations, but they all rely upon a rule that tells the algorithm when to stop. Usually it is embodied in a piece of code like this:

```
while (error > RELTOL*V + VNTOL and iterations < MAXITERATIONS)
{
    error=Solve();
    iterations = iterations +1;
}
```

This code says to continue iterating the solution while the error is greater than some tolerance and we have not yet exceeded the specified maximum number of iterations. The error itself is defined as the difference in successive estimates of the final answer. Thus, if we get the same answer from one iteration to the next, or at least the difference between two successive answers is less than some acceptable tolerance then we say the solution converged, and the answer at this one data point is accepted as correct.

This criteria is checked for every nonlinear variable in the circuit. If any of these variables fails to converge, then the infamous message,

"Internal time step too small",

or one of its many cousins, is issued.

Convergence checks are applied during each nonlinear analysis operation, including the transient and AC analysis operating point calculations, all transient analysis data points, and all DC transfer analysis data points. The linear part of AC analysis is the only part when convergence checking is not necessary.

Convergence is the agreement of successive approximations to an answer.

What causes convergence problems

Many things can cause non-convergence. Here are the usual suspects:

Model discontinuities: Sometimes the model produces a discontinuity in a conductance, transconductance, or capacitance term. When the solution traverses the discontinuity, a disproportionate result is obtained, and the solution iterates around the discontinuity until the iteration limit is reached. There is little the user can do about this cause, except to avoid the model region where the discontinuity occurs.

Bistable or even unstable circuits: If a circuit is unstable, or has multiple stable states, the routines will sometimes iterate between one stable state and another. Because they never converge to one stable answer, convergence fails. This is usually a problem only in the DC operation point and transfer function analyses. To solve this problem, the DC operating point is often bypassed or achieved by ramping the power supplies and transient analysis is often substituted for DC transfer analysis.

Incorrect modeling: This is the most common source of trouble. Unrealistic impedances or source values, zero-valued capacitors, dangling nodes, and unintentional shorts are some of the more common problems. The most common problem in this category is misconnected components. If you take a circuit that converges and runs through every analysis perfectly and randomly alter its topology by breaking or adding connections, it will frequently fail to converge. The second most common problem is zero-valued capacitance. Capacitors act like shock absorbers in transient analysis. They harmonize with the numerical routines to produce more realistic, more convergeable solutions. Model capacitances should rarely be set to zero. Use small values if you must, but don't make them zero. The third most common problem occurs when a diode or a switch is placed in series with an inductor. As experienced auto mechanics know, very high voltages are produced by interrupting the flow of current through an inductor. An ideal diode does this very neatly. When this happens, the time step routines reduce the time step to very small values and sometimes produce non-convergence. To mitigate this problem, use a medium value resistor in parallel with the diode or switch to absorb some of the current. Make the resistance large enough so that it doesn't interfere with normal circuit operation, but small enough to absorb the current when the diode shuts off. A practical value is 1E6. Even if non-convergence isn't a problem, this technique will usually speed up the run.

Convergence checklist

Here are the things you should try when convergence problems arise:

Check circuit topology:

Path to ground: Every node must have a DC path to the ground node. The classic example of this is a circuit containing two capacitors in series, with nothing else connected to the junction of the two capacitors. From the junction, there is no DC path to ground and the circuit will fail to find a DC operating point. The solution is to replace the series combination with an equivalent capacitor, or put a high-valued resistor like 1E12 from the junction to ground. Another circuit configuration that creates a very similar problem is a set of cascaded, ungrounded, transmission lines. There is no DC path from the output of a transmission line to the input, so unless the junction between the lines is grounded or has a path to ground through some other component, an operating point convergence failure usually occurs. The solution is to place a high-valued resistor like 1E12 from the junction to ground.

Current sources in series: Current sources in series with different values are a logical absurdity and may produce convergence errors. Eliminate them or add a large resistance in parallel with each source.

Voltage sources or inductor loops: Both voltage sources and inductors are voltage-defined branches. A loop with only voltage-defined branches is a source of much trouble since it allows a possibly nonzero sum of voltages around the loop. MC7 will check for these, but if you eliminate them by adding a very small resistance within the loop to absorb the net loop voltage, be careful not to make it too small.

Shorts and opens: The easiest way to see short circuits and open circuits is to turn on the Node numbers display option. If two nodes are shorted to form one node there will be a single node number. If two nodes are distinct there will be a unique node number on each.

Floating nodes: Floating nodes sometimes cause problems. Make sure the Floating Nodes option (Options / Preferences / Common Options) is enabled. This eliminates floating nodes by checking for the presence of at least two connections for every node. The check procedure is done when an analysis is selected.

Check circuit modeling:

Nonzero diode series resistance: The RS of the diode serves to self-limit the possibly exponential current that can flow if an external voltage source is placed across the diode. The same argument applies to the BJT transistor lead resistances.

Nonzero diode parallel resistance: The RL of the diode serves to limit the reverse voltages that can occur when an inductor is in series with the diode. Normal reverse resistance due to IS alone is on the order of 1e15 ohms. This is unrealistically high for most diodes. The RL parameter provides a convenient way to specify a realistic leakage resistance of 1E6 to 1E9. *High reverse diode resistance in the presence of series inductors not only leads to the most common cause of "time step too small" errors, it is also a common cause of slow simulations, as it forces very small time steps.*

Nonzero capacitors: Do all capacitors have nonzero values? If not, set them to a small value relative to other capacitors in the circuit.

Switches and inductors in series: If these are present and are causing problems, add parallel resistors of 10K or larger to minimize convergence problems. Any kind of a switch in series with an inductor can cause a problem. This includes diodes, switches, and active devices.

MOSFET drain-source conductance: A small nonzero value, like .01 to .001, for the LAMBDA model parameter, produces a finite output conductance and can frequently solve convergence problems with MOSFETs. Alternatively, adding a 1K to 10K resistor between the drain and source will accomplish the same thing. The value of the resistance should be large enough to avoid loading the drain circuit.

Tweak the Global Settings:

Increase Gmin: Gmin is the minimum branch conductance. Increasing it sometimes helps a DC operating point to converge.

Increase RELTOL: RELTOL is the maximum relative error tolerance. Sometimes increasing it from the default value of .001 to .01 will converge a difficult circuit.

Increase ABSTOL or VNTOL: These Global Settings values specify the maximum absolute error tolerance in current variables and voltage variables, respectively. The default values for these parameters are:

ABSTOL = 1E-12
VNTOL = 1E-6

These values are suitable for typical integrated circuits where the currents are typically 10 mA and the voltages are typically 10 volts. If your circuit contains especially large currents or voltages, try increasing these values proportionate to the size of the expected maximum current and voltage values in your circuit. Thus, if your circuit produces 1000 amps, increase ABSTOL by 1E5 (1000/10mA), to 1E-7 (1E5*1E-12).

Increase ITL1: If the non-convergence occurs in a DC operating point, then increasing ITL1 from its customary 100 to a larger value sometimes helps. There are circuits that converge after 150, or even 300 iterations. Convergence Assist, if enabled, will increase ITL1 for you automatically.

Increase ITL4: If the non-convergence occurs during a transient analysis, increase ITL4 to 20 or even 50. ITL4 is the limit on the number of iterations at each timepoint beyond which the program discards the solution, reduces the timestep, and tries again with the smaller timestep. Convergence Assist, if enabled, will do this for you automatically.

Turn off the operating point:

If the problem is lack of convergence in the operating point, try eliminating it and letting the circuit simply stabilize at its operating point the way a real circuit would do if you simply turned on the power. If you anticipate many simulations, you might want to run the circuit with all waveform sources set to their Time= 0 values to let it stabilize at an operating point and then save the operating point values from the State Variables editor. Later runs would then be set to read the operating point values from the file. This is done by choosing the Read option in the State Variables list box in the Analysis Limits dialog box. On the subsequent runs you may need to choose the Operating Point option if the circuit has changed enough to change the operating point values. The initial approximation from the file should substantially aid convergence.

Try something drastic:

Ramp the supplies slowly: Replace all DC voltage and current sources in the circuit with pulse source equivalents. Then set the timing parameters of the pulse sources to gradually ramp up the values. Use 0.0 for all initial source values, or bypass the DC operation point completely by disabling the Operating Point option in the Analysis Limits dialog box. This option is only available in transient analysis.

Use Nodeset commands: Nodeset commands are used to specify an initial guess at the value of a node voltage, prior to beginning the operating point. If the guess is close, it sometimes helps convergence.

Use the Off keyword: Turn active devices off with the OFF keyword. Especially if the device is involved in the area of the circuit that is not converging, this can be a very useful technique.

Use the IC device options: These options let you specify the initial conditions for the active devices. If the device is the one not converging, this can help. It is necessary to estimate the initial condition (voltage or current) to use this method.

Check the numeric output file: If convergence fails, look at the numeric output (F5). It includes information on the devices that failed to converge. Sometimes you can get a hint of what the problem is by seeing which devices failed.

Enable Convergence Assist: This option, enabled from **Options / Global Settings / Common Options / Analysis** is sometimes useful in getting difficult circuits to converge. It tries several combinations of the Global Settings simulation parameters in an attempt to get the circuit to converge. If it succeeds, it places a .OPTIONS statement with the successful parameters in the circuit so that future runs will converge more readily.

Chapter 19 Expressions

What's in this chapter

Expressions are used in certain component parameters and in plotting and printing simulation results. A clear understanding of expressions is essential to get the most from MC7. This chapter covers these topics:

- What are expressions?
- Numbers
- Constants
- Variables
- Component variables
- Subcircuit and Macro Variables
- Model variables
- Sample variables
- Mathematical operators and functions
- Sample expressions
- Rules for using operators and variables

Features new in Micro-Cap 7

- All trigonometric and hyperbolic functions and their inverses are now available and all accept complex arguments and return complex answers.
- Complex IF, MIN, MAX, and LIMIT functions.
- New energy terms (ED,ES,EG) similar to power terms.
- Run-invariant expressions are now allowed in model parameters. For example, VTO=2+TEMP/100.

What are expressions?

Expressions are text strings entered by the user that contain numbers, constants, variables, and mathematical operators. All curves to be printed or plotted are defined in the Analysis Limits dialog box with expressions. The numeric behavior of resistors, capacitors, inductors, Laplace sources, and Function sources is defined through the use of expressions.

Case is ignored in expressions, so RLOAD is the same variable as Rload. The expression value is updated whenever any of the constituent variables change.

Here are some typical expressions:

10.0
V(10)
V(OUT)*I(L1)
VCE(Q1)*IC(Q1)

Numbers

Numbers are expressed in one of three formats:

- **Real numbers:**
  1.0, 6.7

- **Floating point numbers**: These use standard scientific notation.
  1.87E-12, 23E3

- **Engineering notation:**
  2.7K, 12pF, 10.5ma, 1MEGHz

  This notation uses standard engineering abbreviations. The letters for units are optional (F, a, Hz). No blank space is allowed between the number and letter(s). Engineering notation uses these abbreviations:

Abbreviation	Name	Value
F	Femto	1E-15
P	Pico	1E-12
N	Nano	1E-9
U	Micro	1E-6
M*	Milli	1E-3
K	Kilo	1E3
MEG*	Mega	1E6
G	Giga	1E9
T	Tera	1E12

* To conserve space, the X and Y scales on MC7 graphs use the small letter 'm' to refer to milli and the large letter 'M' to refer to Mega.

Constants and analysis variables

MC7 provides these constants and analysis variables.

Symbol	Value
T	Time in seconds
F	Frequency in Hz
DCINPUT1	Value of Variable1 in DC analysis
E	EXP(1) = 2.718281828459045
PI	3.141592653589793
S	Complex frequency = 2*PI*J
J	Imaginary unit value used in complex numbers. For example 1+J, or 10+23*J
TEMP	Analysis temperature and default device temperature in degrees Celsius
VT	1.3806226e-23*(273.15 + TEMP)/1.6021918e-19 = 2.586419e-2 at TEMP=27 °C
GMIN	Minimum conductance across junctions
TMIN	Starting transient analysis time
TMAX	Ending transient analysis time
DT	Transient analysis time step
FMIN	Starting AC analysis frequency
FMAX	Ending AC analysis frequency
INOISE	Input noise in AC analysis
ONOISE	Output noise in AC analysis
ANALYSIS	= _TRANSIENT in transient analysis = _AC in AC analysis = _DC in DC analysis = _DYNAMICDC in Dynamic DC analysis = _TF in Transfer Function analysis = _SENS in Sensitivity analysis For example, the expression ANALYSIS=_AC would be TRUE (1.0) in AC analysis and FALSE (0.0) elsewhere.
PGT	Total power generated in the circuit
PST	Total power stored in the circuit
PDT	Total power dissipated in the circuit
EGT	Total energy generated by sources in the circuit
EST	Total energy stored in the circuit
EDT	Total energy dissipated in the circuit

Variables

In the variable definitions that follow, the symbols A and B represent node names. A node name is one of the following:

1. A node number assigned by MC7.
2. A text node name (a piece of grid text placed on the node).

Text node names consist of a number, letter, special character (+,-,*,/,$,%), or an underscore followed by at most 50 alphanumeric characters.

Node names may consist of numbers only, but this is not recommended due to the likely confusion between node numbers assigned by the program and integer node names assigned by the user. In case of conflict, MC7 gives priority to the node number. So if you place the text "1" on node number 2 and try to plot V(1) you'll get the voltage on node number 1, not the voltage on the node labeled "1".

Reserved variable names and some mathematical operator names may not be used. Spaces are not allowed in text node names. For example, T, S, F, TEMP, VT, GMIN, J, E, and PI are all invalid because they use reserved variable names. Function names that use parentheses such as SIN, COS, and TAN are legal, whereas function names that do not use parentheses such as MOD, DIV, and RND are illegal.

A1, Out, _721, +, -, and Reset are valid node names. B&&4 is invalid because it uses the non-alphanumeric character &. T1 is valid but T is not because T is a reserved variable name.

Global nodes are nodes whose names are globally available to all parts of the circuit, including the top level circuit and all macros and subckts used by it. Global nodes are always prefaced by $G_. For example if a circuit uses a subckt that has a node named $G_ABC, then you can plot its voltage with V($G_ABC). As another example, if a digital part that uses the $G_DPWR is present in the circuit, plotting V($G_DPWR) plots the default TTL power supply node voltage waveform (a flat line at 5.0 volts)

The general list of variables is as follows:

D(A)	Digital state of node A
V(A)	Voltage at node A
V(A,B)	Voltage at node A minus voltage at node B
V(D1)	Voltage across the device D1
I(D1)	Current through the device D1
I(A,B)	Current through the device using nodes A and B
IR(Q1)	Current into the R lead of the device Q1
VRS(Q1)	Voltage across the leads R and S of the device Q1
CRS(Q1)	Capacitance between leads R and S of the device Q1
QRS(Q1)	Capacitor charge between leads R and S of device Q1
R(R1)	Resistance of the resistor R1
C(X1)	Capacitance (in farads) of the capacitor or diode X1
Q(X1)	Charge (in coulombs) stored in the capacitor or diode X1
L(L1)	Inductance (in henrys) of the inductor L1
X(L1)	Flux (in webers) in the inductor L1
B(L1)	B field (in gauss) of the core material of inductor L1
H(L1)	H field (in oersteds) of the core material of inductor L1
T	Time
F	Frequency
S	Complex frequency = 2*π*F*j
RND	Random number generator (0 - rnd -1)
ONOISE	Noise voltage at the output node
INOISE	Noise voltage referred to the input = ONOISE / gain
EG(V1)	Energy generated by source V1
ES(Q1)	Energy stored in device Q1
ED(D1)	Energy dissipated in device D1
PG(V1)	Power generated by source V1
PS(Q1)	Power stored in device Q1
PD(D1)	Power dissipated in device D1

D1 represents any two-terminal device or controlled source. Q1 represents all active devices and transmission lines. The lead name abbreviations, R and S, are chosen from the following table.

Device	**Abbreviations**	**Lead name**
MOSFET	D,G,S,B	Drain, Gate, Source, Bulk
JFET	D,G,S	Drain, Gate, Source
GaAsFET	D,G,S	Drain, Gate, Source
BJT	B,E,C,S	Base, Emitter, Collector, Substrate

Component variables

The variables available for each component are shown in the following tables.

Component Variables							
Component	Voltage	Current	Capacitance/ Inductance	Charge / Flux	Power/Energy Generated	Power/Energy Stored	Power/Energy Dissipated
Sources	V	I	NA	NA	PG / EG	NA	NA
Resistor	V	I	NA	NA	NA	NA	PD / ED
Capacitor	V	I	C	Q	NA	PS / ES	NA
Inductor	V	I	L	X	NA	PS / ES	NA
Diode	V	I	C	Q	NA	PS / ES	PD / ED
Transmission Line	VAP, VAM, VBP VBM	IAP, IAM IBP, IBM	NA	NA	NA	NA	NA
BJT	VB, VC, VE, VBE, VBC, VEB VEC, VCB, VCE	IB, IE, IC	CBE, CBC	QBE, QBC	NA	PS / ES	PD / ED
BJT4	VB, VC, VE, VS, VBE, VBC, VBS VEB VEC, VES VCB, VCE, VCS VSB, VSE, VSC	IB, IE, IC IS	CBE, CBC CCS	QBE, QBC QCS	NA	PS / ES	PD / ED
MOSFET: LEV 1-3	VG, VS, VD, VB, VGS, VGD, VGB VDS, VDG, VDB VSG, VSD, VSB VBG, VBD, VBS	IG, IS, ID IB	CGS, CGD CGB, CBD CBS	QGS, QGD QGB, QBD QBS	NA	PS / ES	PD / ED
MOSFET:LEV 4,5,8	VG, VS, VD, VB, VGS, VGD, VGB VDS, VDG, VDB VSG, VSD, VSB VBG, VBD, VBS	IG, IS, ID IB	NA	NA	NA	NA	PD / ED
OPAMP	VP, VM, VOUT, VPM, VCC, VEE	NA	NA	NA	NA	NA	NA
JFET	VG, VD, VS, VGS, VGD, VSG VSD, VDG, VDS	IG, ID, IS	CGS, CGD	QGS, QGD	NA	PS / ES	PD / ED
GaAsFET	VG, VD, VS, VGS, VGD, VSG VSD, VDG, VDS	IG, ID, IS	CGS, CGD	QGS, QGD	NA	PS / ES	PD / ED
Variables that are mere permutations of the leads are not shown. For example CGS and CSG produce the same plot as do QGS and QSG.							

Table 19-1 Syntax for common variables

Component Variables					
Component	Resistance	Flux	Inductance	B field	H field
Resistor	R	NA	NA	NA	NA
Inductor	NA	X	L	B	H
The complete variable name includes the appropriate device name. For example, the B field of a device L1 is referenced as B(L1).					

Table 19-2 Syntax for resistance, flux, inductance, and B / H field variables

Subcircuit and macro variables

To reference a node name or a part name of an object within a macro or subcircuit, use the following dot notation:

Subcircuit part name + "." + node name or part name

For example, to reference node 10 in subcircuit X41, use the expression:

X41.10

To reference the node voltage on that node you would use:

V(X41.10)

To reference the current in the diode DSTUB in subcircuit CHOPPER4, you would use the expression:

I(CHOPPER4.DSTUB)

To reference the charge in the base-emitter junction of an NPN N3 in subcircuit AMP1, you would use the expression:

QBE(AMP1.N3)

If the node or part is nested inside more than one macro or subcircuit, simply concatenate the macro or subcircuit names. For example:

V(X1.X2.X3.10)

This specifies the voltage on node 10 in macro X3, in macro X2, in macro X1.

To see more examples, load the circuit SUBCKT1, run transient analysis, and click the right mouse button in the Y Expression field. This pops up a variable menu which shows all of the circuit variables available in the entire circuit. It nicely demonstrates many examples of subcircuit variables.

Model parameter variables

You can print or plot a part's model parameter by using the following syntax:

PART_NAME.MODEL_PARAMETER_NAME

Here are some examples:

Q1.BF	Forward beta of BJT Q1
M1.GAMMA	GAMMA parameter of MOSFET M1
J1.VTO	VTO of JFET J1

Since model parameters do not vary during an analysis, plots of these variables will produce straight lines. Why bother plotting them? If you are stepping a parameter, or running a Monte Carlo analysis, and you want to check the value of a parameter of a particular part for a particular run, plotting its model parameter value is a quick way to be sure it is being stepped through the values you want.

Sample variables

Here are some sample variables.

T	Time in seconds.
F	Frequency in Hz.
D(A)	Digital state of node A.
HEX(A1,A2,A3,A4)	Hex value of the nodes A1, A2, A3, and A4.
BIN(A1,A2,A3,A4)	Binary value of the nodes A1, A2, A3, and A4.
OCT(A1,A2,A3)	Octal value of the nodes A1, A2, and A3.
DEC(A1,A2,A3,A4)	Decimal value of the nodes A1, A2, A3, and A4.
V(16,4)	Voltage at node 16 minus the voltage at node 4.
V(A,B)	Voltage at node A minus the voltage at node B.
I(R1)	Current flowing through the resistor R1.
I(2,3)	Current flowing through the resistor, capacitor, source, or inductor between nodes 2 and 3.
B(L1)	B field in the inductor L1.
H(L1)	H field in the inductor L1.
X(L2)	Flux in the inductor L2.
IB(Q1)	Base current into the device Q1.
VBE(Q1)	Base-emitter voltage for the device Q1.
IG(M1)	Gate current into the device M1.
VGS(M1)	Gate-source voltage for the device M1.
QBE(Q1)	Charge stored in Q1's base-emitter capacitance.
VAP(T1)	Voltage at the positive pin of the input port of transmission line T1.
ID(J1)	Drain current into the device J1.
I(D1)	Current into the diode D1.
L(L1)	Inductance of the inductor L1.
C(C2)	Capacitance of the capacitor C2.
R(R7)	Resistance of the resistor R7.
I(R1)	Current through the resistor R1.
I(Lap1)	Current through the Laplace source Lap1.
I(V1)	Current through the waveform source V1.
V(F1)	Voltage across the Function source F1.
V(X1.MID)	Voltage on node MID in subcircuit X1.
IB(G3.Q1)	Base current of Q1 NPN in macro circuit G3.
V(G1.G2.N)	Voltage on node N in macro G2, in macro G1.
ES(C1)	Power stored in capacitor C1
PS(D1)	Power stored in diode D1
PG(V1)	Power generated by source V1
PD(Q1)	Power dissipated in transistor Q1

Mathematical operators and functions

In the definitions, the following symbol conventions are used:

Symbol	Represents
n, m	Integers
dt	The DSP timestep.
x, y, u	Real expression. For example 26.5, T in transient, V(10) in DC.
z	Complex quantity. z= x + i • y. For example, V(1) in AC.
S	Spectrum generated by one of the signal processing operators.
D1, D2	Digital node states. For example D(1), D(QB).

Arithmetic

+	Addition
-	Subtraction
*	Multiplication
/	Division
MOD	Modulus (remainder after integer division)
DIV	Integer division

Transcendental (x and y are real, z is complex, z = x + i • y)

SIN(z)	Sine function
COS(z)	Cosine function
TAN(z)	Tangent function
COT(z)	Cotangent function
SEC(z)	Secant function
CSC(z)	Cosecant function
ASIN(z)	Inverse sine function
ACOS(z)	Inverse cosine function
ATAN(z)	Inverse tangent function
ATN(z)	Inverse tangent function
ARCTAN(z)	Inverse tangent function
ATAN2(y,x)	Inverse tangent function = ATN(y/x)
ACOT(z)	Inverse cotangent function
ASEC(z)	Inverse secant function
ACSC(z)	Inverse cosecant function
SINH(z)	Hyperbolic sine
COSH(z)	Hyperbolic cosine
TANH(z)	Hyperbolic tangent

COTH(z)	Hyperbolic cotangent
SECH(z)	Hyperbolic secant
CSCH(z)	Hyperbolic cosecant
ASINH(z)	Inverse hyperbolic sine
ACOSH(z)	Inverse hyperbolic cosine
ATANH(z)	Inverse hyperbolic tangent
ACOTH(z)	Inverse hyperbolic cotangent
ASECH(z)	Inverse hyperbolic secant
ACSCH(z)	Inverse hyperbolic cosecant
LN(z)	Natural log: $\log_e(\|x + i \cdot y\|) + i \cdot \tan^{-1}(y / x)$
LOG(z)	Common log: $\log_{10}(\|x + i \cdot y\|) + i \cdot \tan^{-1}(y / x) / \log_e(10)$
LOG10(z)	Common log: $\log_{10}(\|x + i \cdot y\|) + i \cdot \tan^{-1}(y / x) / \log_e(10)$
EXP(z)	Exponential: $e^x \cdot (\cos(y) + i \cdot \sin(y))$
POW(z,x)	Complex exponentiation function= $z^x = e^{x \ln(z)}$ For example, POW(-1+j,2) = -2j, POW(2,2) = 4
^, or **	Same as PWW(z,x). z^x = z**x = POW(z,x) For example, (-1+j,2)**2 = -2j, j^2 = -1
PWR(y,x)	Real power function = y^x. For example PWR(-2,3) = -8, PWR(-2,2) = 4
PWRS(y,x)	Real signed power function: if y<0 PWRS(y,x) = $-\|y\|^x$ if y>0 PWRS(y,x) = $\|y\|^x$ For example PWRS(-2,2) = -4, PWRS(2,2) = 4
DB(z)	20 * LOG(|z|)
RE(z)	Real part of z
IM(z)	Imaginary part of z. IMAG() and IMG() also work.
MAG(z)	Magnitude of z. M() also works.
PH(z)	Phase of z in degrees. PHASE() and P() also work.
GD(z)	Group delay=∂(Phase(z in radians))/∂(radian frequency)

Digital

A is the MSB. D is the LSB. These operators are designed for use in plotting and printing logic expression waveforms.

D(A)	Digital state on node A.
HEX(A,B,C,D)	Hex value of the digital states of nodes A, B, C, D.
BIN(A,B,C,D)	Binary value of the digital states of nodes A, B, C, D.

DEC(A,B,C,D)	Decimal value of the digital states of nodes A, B, C, D.
OCT(A,B,C,D)	Octal value of the digital states of nodes A, B, C, D.
+	Sum of two binary, octal, hex, decimal values.
-	Difference of two binary, octal, hex, decimal values.
MOD	Modulus operator (integer division remainder) of two binary, octal, hex, decimal values.
DIV	Integer division of two binary, octal, hex, decimal values.
&	Bitwise AND of two digital node states.
|	Bitwise OR of two digital node states.
^	Bitwise XOR of two digital node states.
~	Bitwise NOT of a digital node state.

Analog Boolean and Relational

AND	And operator
NAND	Nand operator
OR	Or operator
NOR	Nor operator
XOR	Exclusive-Or operator
NOT	Negation operator
<	Less than operator
>	Greater than operator
<=	Less than or equal operator
>=	Greater than or equal operator
!=	Not equal to operator
<>	Not equal to operator
==	Equal to operator
MIN(z1,z2)	Minimum of real and imaginary parts of z1 and z2
MAX(z1,z2)	Maximum of real and imaginary parts of z1 and z2
LIMIT(z,z1,z2)	Returns z, with its real part limited to the range of RE(z1) to RE(z2) and the imaginary part limited to the range IM(z1) to IM(z2)
IF(b,z1,z2)	If b is true, the function returns z1, else it returns z2.

Signal processing

HARM(u)	Harmonics of waveform u
THD(S[,F])	Total harmonic distortion of spectrum S as a percent of the value at the reference frequency F. If F is missing, it is set to the first harmonic (1/tmax in transient analysis)

IHD(S[,F])	Individual harmonic distortion of spectrum S as a percent of the value at F. Similar to THD but not cumulative.
FFT(u)	Forward Fourier transform of waveform u
IFT(S)	Inverse Fourier transform of spectrum S
CONJ(S)	Conjugate of spectrum S
CS(u,v)	Cross spectrum = CONJ(FFT(v))*FFT(u)*dt*dt
AS(u)	Auto spectrum of waveform u = CS(u,u)
CC(u,v)	Cross correlation of u and v = IFT(CS(u,v))/dt
AC(u)	Auto correlation of waveform u is = IFT(AS(u))/dt
COH(u,v)	Coherence of u and v =CC(u,v)/sqr(AC(u(0))*AC(v(0)))
REAL(S)	Real part of spectrum S produced by FFT
IMAG(S)	Imaginary part of spectrum S produced by FFT
MAG(S)	Magnitude of spectrum S produced by FFT
PHASE(S)	Phase of spectrum S produced by FFT

Numeric integration and differentiation: These are for analysis plots only.

With respect to any variable

DER(u,x)	Calculates the derivative of u W.R.T x.
SUM(y,x[,start])	Running integral of y with respect to x, with optional start parameter. Integral begins at x=start. Start defaults to the analysis variable minimum (tmin, fmin, or dcmin), or 0, depending upon the integration variable, x.

With respect to the analysis variable (T, F, or DCINPUT1)

SD(y[,start])	Running integral of y with respect to T in transient, F in AC, or DCINPUT1 in DC, with an optional start parameter. Integral begins at start. Start defaults to tmin, fmin, dcmin, according to the analysis type
DD(y)	Numerical derivative of y with respect to T in transient, F in AC, or DCINPUT1 in DC
RMS(y[,start])	Running root-mean-square of y with respect to F in AC, T in transient, or DCINPUT1 in DC, with an optional start parameter. The integral begins at start. Start defaults to tmin, fmin, or dcmin, according to the analysis type.
AVG(y[,start])	Running average of y with respect to T in transient, F in AC, or DCINPUT1 in DC. The optional start parameter defaults to tmin, fmin, dcmin.

With respect to T (Time) only

SDT(y)	Running integral of y with respect to T (Time). Integral begins at T = tmin
DDT(y)	Numerical derivative of y with respect to T (Time)
DEL(y)	Change in y from the prior data point to the current point. A numerical derivative is formed by the ratio of two operators. For example, DEL(y)/DEL(t) approximates the numerical time derivative of y.

Other functions

ABS(z)	Absolute value function = $(\|z\|^2)^{0.5}$
DIFA(u,v[,d])	DIFA reports differences between two analog curves. It compares the u expression with the v expression at every analysis point, and returns 1 if the absolute value of the result is more than d. Otherwise, it returns 0. U and v may be imported curves. See the IMPORT function. D is optional and defaults to 0.
DIFD(u,v[,d])	DIFD reports differences between two digital curves. It compares the u level with the v level at every analysis point, and returns 1 if they differ for a time exceeding d. Otherwise, it returns 0. U and v may be imported curves. D is optional and defaults to 0.
FACT(n)	Factorial of the integer n.
JN(n,z[,m])	N'th order Bessel function of the first kind of the complex expression z, compiled from the series using m terms. M defaults to 10.
J0(z)	Zero'th order Bessel function of the first kind of the complex expression z. Same as JN(0,z,10)
J1(z)	First order Bessel function of the first kind of the complex expression z. Same as JN(1,z,10)
RND	Returns a random value between 0 and 1.

SERIES(n,n1,n2,z)	Calculates the summation of the series of the complex expression z = z(n), for n = n1 to n = n2.
SGN(y)	+1 (if y>0), 0 (if y=0), -1 (if y<0)
SQRT(z)	Complex square root = $z^{0.5}$
STP(x)	Step function of amplitude 1.0 starting at T >= x.
TABLE($x,x_1..x_n,y_n$)	This function performs a table lookup. It returns a value for y associated with the value of x, interpolated from the table. X values less than x_1 generate an answer of y_1. X values greater than x_n generate an answer of y_n.
IMPORT(f,y)	Imports curve y from the file f. The file must be a SPICE or MC7 output text file with a table of values that includes the value F(frequency), T(Time), V(source voltage), or I(current source), and the value of the expression y. The expression y must be typed exactly as shown in the file and must contain an even number of parentheses.
IMPULSE(y)	Impulse function of amplitude y and area of 1.0.
CURVEY("file","W")	Imports the Y component of curve W from the User source filename. The file must be saved in standard format. MC7 can save curves for you using the Save Curves section of the Plot Properties dialog box. See User source in Chapter 22 for the file format.
CURVEX("file","W")	Imports the X component of curve W from the User source filename.
YN(n,z[,m])	N'th order Bessel function of the second kind of the complex expression z, compiled from the series using m terms. M defaults to 10.
Y0(z)	Zero'th order Bessel function of the second kind of the complex expression z. Same as YN(0,z,10)
Y1(z)	First order Bessel function of the second kind of the complex expression z. Same as YN(1,z,10)

Sample expressions

Digital

D(1) & D(2)	AND of the digital state of node 1 with the digital state of node 2.
D(1) | D(2)	OR of the digital state of node 1 with the digital state of node 2.
Hex(A,B,C,D)+Hex(R,S,T,U)	Hex sum of two hex values. The first term is the hex value of the states on nodes A, B, C, D. The second term is the hex value of the states on nodes R, S, T, U. The result is the hex sum of these two hex values.

Laplace source transfer functions

1.0/(1.0+.001*s)	Transfer function of a low pass filter
1.0/(1.0+.001*s+1e-12*s*s)	Transfer function of a second order filter
exp(-(s*C*(R+s*L))**0.5)	Transfer function equation for a lossy transmission line (R, L, and C are the per unit length values)

Function sources

exp(-T/.5)*sin(2*π*10*T)	An exponentially damped 10 Hz sine wave
-k*(v(p)-v(c)+mu*(v(g)-v(c)))**1.5	The current equation for a vacuum triode. The terms p, g, and c are the plate, grid, and cathode names. v(p), v(g), and v(c) are the voltage at the plate, grid, and cathode respectively.

Capacitance

2pf/((1-v(p,n)/.7)**.5)	Typical junction capacitance expression
5.0pf*(1+2e-6*T)	A time-dependent capacitor

Resistance

5*(1+2*(TEMP - 273)**2)	A temperature-dependent resistance
4.7K*(1+.3*V(P,M))	A voltage-dependent resistance

Inductance	
1uh*coth(I(L1)/1ma)	A nonlinear inductance
2.6uh*(1+(t-1e-7)**2.0)	A time-dependent inductance
Power and energy	
V(VCC)*I(VCC)	Instantaneous power from the source VCC
PD(R1)	Power dissipated in resistor R1
PS(Q1)	Power stored in transistor Q1
PG(V1)	Power generated by source V1
ED(D3)	Energy dissipated in diode D3
ES(C2)	Energy stored in capacitor C2
ES(L4)	Energy stored in inductor L4
EG(VCC)	Energy generated by source VCC
EST	Total stored energy in the circuit
EDT	Total dissipated energy in the circuit
EGT	Total generated energy in the circuit
PST	Total power stored in the circuit
PDT	Total power dissipated in the circuit
PGT	Total power generated in the circuit
SUM(V(VCC)*I(VCC),T)	Energy provided by the source VCC
Miscellaneous	
FFT(V(A)+V(B))	Forward Fourier transform of V(A)+V(B)
IFT(2*fmax*V(Out))	In AC analysis, using this Y expression and T as the X expression generates the impulse response of a network. V(Out) is the complex output voltage and fmax is the maximum frequency of the run.
CC(V(1),V(2))	Cross correlation of V(1) and V(2)
DEL(I(L1))/DEL(T)	Numeric derivative of the current flowing in L1
SUM(V(Out),T)	Numeric integral of the voltage curve V(Out)

SUM(V(Out),T,5n)	Numeric integral of the voltage waveform at the node Out with respect to time, from T = 5ns to end of *tmax*.
SUM(V(A),V(B),2)	Numeric integral of the voltage waveform at the node A with respect to V(B), from V(B) >= 2 to the last value of V(B) at end of run.
RMS(V(Out))	Running RMS value of the expression V(Out)
5*(T>10ns AND T<20ns)	A non-periodic, 5V pulse, from 10ns to 20ns
5*((T mod 50)>10 AND (T mod 50)<20)	A 5V pulse, from 10s to 20s, with period of 50s
IM(V(7))	Imaginary part of the complex voltage V(7)
VCE(Q1)*IC(Q1)	Complex AC power in collector of device Q1
TABLE(V(1),-10,-1,10,1)	This table function returns -1 if V(1) is less than -10, 0.1*V(1) if V(1) is between -10 and +10, and +1 otherwise.
IMPORT(A.OUT,V(1))	Imports waveform V(1) from the file A.OUT.
CURVEX("T1","I(V1)")	Imports the X component of the curve "I(V1)" from the user source file "T1".
CURVEY("T2","I(V1)")	Imports the Y component of the curve "I(V1)" from the user source file "T2".
FACT(5)	Computes factorial(5) = 120.
JN(5,1+J,6)	Computes the fifth order Bessel function of the first kind, using the first 6 terms of the series.
.DEFINE _EXP(X) (1+SERIES(N,1,10,POW(X,1)/FACT(N)))	Creates a macro to compute exp(x) from its series. _exp(1.0) returns 2.718282 = e.

Rules for using operators and variables

Here are some important rules to keep in mind when using expressions.

1. The relational and Boolean operators return 1.0 if true and 0.0 if false.

2. The operators that use integration and differentiation, such as RMS, AVG, SUM, SD, DD, DEL, SDT, and DDT operators may be used only for printing and plotting. They can't be used in device parameter expressions.

3. ONOISE and INOISE should only be used in AC analysis and never mixed with other variables like V(somenode).

4. In AC analysis, all intermediate calculations are performed on complex values. After the expression is completely evaluated, the magnitude of the complex result is printed or plotted. For example, V(1)*V(2) prints or plots the magnitude after the complex multiplication. To print the imaginary part use IM(V(1)*V(2)). To print the real part use RE(V(1)*V(2)). To plot the magnitude use V(1)*V(2), or if you prefer, MAG(V(1)*V(2)).

5. The value of the time variable, T, is set to zero in AC and DC analysis. The value of the frequency variable, F, is set to zero in transient and DC analysis.

6. Transfer function expressions for a Laplace source use only the complex frequency variable, S. No other variables should be used. If the S is missing, an error will result. Do not use Laplace sources for constant gain blocks. Use independent sources, SPICE poly sources, or function sources.

7. You can use symbolic variables in model statements, if the variable is used only for the parameter value, and not for the parameter name. For example, this will work:

```
.define VALUE 111
.model Q1 NPN (BF=VALUE ...)
```

After expansion, the model statement becomes

```
.model Q1 NPN (BF=111 ...)
```

This is syntactically correct and meaningful.

8. Define statements expand literally. For example:

```
.define a       4+c
.define b       a*x
```

You might expect that b would expand to (4+c)*x but it actually expands to 4+c*x. That is because the expansion of define statements is literally a text replacement. It replaces the text 'a' with the text '4+c'.

To avoid such problems, use parentheses around the defined quantity, such as in the following define statements:

```
.define a       (4+c)
.define b       a*x
```

Now b expands to (4+c)*x.

Chapter 20 Command Statements

What's in this chapter

Command statements are text strings which begin with a period. They are implemented by placing grid text in a schematic page or in the text area, or ordinary text in a SPICE circuit.

Features new in Micro-Cap 7

- .TR command
- .FUNC command
- .HELP command

.AC

General Form (SPICE files only)
.AC [[DEC] | [OCT] | [LIN]] <*data points*> <*fmin*> <*fmax*>

Examples
.AC DEC 30 20 20K
.AC LIN 10 100 200

The .AC command arguments are mapped into the appropriate MC7 Analysis Limits dialog box fields upon selecting **AC** from the **Analysis** menu. DEC (decade), OCT (octal), and LIN (linear) specify the type of fixed frequency step to be used during the AC analysis. DEC and OCT select the Log Frequency Step option and LIN selects the Linear Frequency Step option on the AC Analysis Limits dialog box.

<*data points*> specifies the number of data points per decade for the decade option or the total number of data points for the linear option. It is mapped into the Number of Points field. <*fmin*> specifies the first frequency of the run and <*fmax*> specifies the last. These are mapped into the Frequency Range field.

A waveform source need not be present in the circuit, but if there is no source, all output will be zero. For waveform sources, the small signal model is simply an AC voltage or current source. The AC value of the source is determined from the parameter line for SPICE components and from the attribute value for schematic components.

SPICE V or I sources:
The AC magnitude value is specified as a part of the device parameter. For example, a source with the value attribute "DC 5.5 AC 2.0" has an AC magnitude of 2.0 volts.

Pulse and sine sources:
These sources have their AC magnitude fixed at 1.0 volt.

User sources: User sources provide a signal comprised of the real and imaginary parts specified in their files.

Function sources: These sources create an AC signal only if a FREQ expression is specified.

.DC

General Form (SPICE files only)

Linear sweep type
.DC [LIN] *<v1> <start1> <end1> <increment1>*
+*<v2> <start2> <end2> <increment2>*]

Log sweep type
.DC <OCT | DEC>
+ *<v1> <start1> <end1> <points1 per octave or decade>*
+ [*<v2> <start2> <end2> <points2 per octave or decade>*]

List sweep type
.DC *<v1>* LIST *<value>** [*<v2>* LIST *<value>**]

<v1> and *<v2>* must be one of the following:

1. The name of an independent source in the circuit. e.g. VCC
2. A model parameter name in the form:
   <model type> <model name(parameter name)> e.g. NPN QF(BF)
3. Operating temperature in the form: TEMP
4. A symbolic parameter in the form:
   PARAM <symbolic parameter name> e.g. PARAM temp1

Sweep specifications can be different for *<v1>* and *<v2>*. That is, you can use linear sweeping for one variable and log for the other.

Examples
.DC VIN1 -.001 .001 1U
.DC VCC 0 5 0.1 IB 0 0.005 0.0005
.DC DEC RES RMOD(R) 1m 100 5
.DC PARAM FILTER_Q 10 20 1
.DC VCC LIST 4.0 4.5 5.0 5.5 6.5 VEE LIST 24 25 26

.DEFINE

General Form1 (Schematics only)
.DEFINE [{LOT[*t&d*]=<*n*>[%]}] <*text1*> <*text2*>

where *t&d* is [[/<*lot#*>][/GAUSS|UNIFORM|WCASE]]

Examples
.Define V1 (2*t*sin(2*pi*T))
.DEFINE RVAL 128.5K
.DEFINE {LOT=10%} LVAL 1200MH
.DEFINE {LOT/1/GAUSS=10%} CVAL 1200NF

This form of the statement is used to create and define the value of a symbolic variable. It substitutes <*text2*> for <*text1*> everywhere except for text from the PART attribute Value field and model parameter name text. While you can't use this statement to change the name of a part or a model parameter name, you can use it to change the name of the model itself.

General Form2 (Schematics only)
.DEFINE <*name*(<*p1*>[,<*p2*>][...,<*pn*>])> f(<*p1*>[,<*p2*>][...,<*pn*>])

where f() is some expression involving the parameters <*p1*>[,<*p2*>][.. .,<*pn*>]

This form of the statement works exactly like C language macros, substituting in at run time the values of <*p1*>[,<*p2*>][...,<*pn*>]. It is also similar to the SPICE .FUNC statement.

Examples:

Macro formulas
.DEFINE F1(A,B) (A+2*B)
.DEFINE THREE(Y,A,B,C) IF(Y>C,C,IF(Y>B,B,A))
.DEFINE DISTANCE(X,Y,Z) POW(X*X+Y*Y+Z*Z,0.5)

String replacement
A good example of string replacement occurs in digital STIM devices. These devices require a COMMAND attribute string to describe their behavior. A typical command might be:

```
.define SQUAREWAVE
+ 0NS 0
+ LABEL=START
+ +10NS 1
+ +10NS 0
+ +10NS GOTO START 10 TIMES
```

In the STIM COMMAND attribute we enter "SQUAREWAVE ". Later when an analysis is run, MC7 substitutes the lengthy text.

.DEFINE statements are frequently used for the STIM COMMAND attribute, PLA DATA attribute, Nonlinear Table Source TABLE attribute, and Laplace Table Source FREQ attribute.

Symbolic variables
The following lets you globally assign or even step the W and L of all MOSFETs that use the MX model statement.

```
.DEFINE W1 2U
.DEFINE L1 .3U

.MODEL MX NMOS (W=W1 L=L1....)
```

User functions
This function lets you easily calculate collector power in a transistor.

```
.DEFINE PC(Q)   VCE(Q)*IC(Q)
```

With this definition, PC(Q10) would plot the collector power in transistor Q10.

The HOT function below flags excessive instantaneous transistor collector power by returning a 1.

```
.DEFINE HOT(Q,MAX)   IF((VCE(Q)*IC(Q)>MAX),1,0)
```

With this definition, HOT(QX3,100MW) would plot a 1 if QX3's instantaneous transistor collector power exceeded 100 mW.

Define statements within a circuit are local to that one circuit. However, the define statements in the MCAP.INC file are globally available to all circuits. This file is accessed from the **User Definitions** item on the **Options** menu. It can be edited by the user.

.END

General Form (SPICE files only)

.END

Examples

.END

This statement specifies the end of the circuit definition. All circuit descriptions and commands must come before the .END statement.

.ENDS

General Form (SPICE files only and schematic text areas only)
.ENDS [<*subcircuit name*>]

Examples
.ENDS
.ENDS FILTER

This statement terminates a subcircuit description. The optional subcircuit name label is used only for clarification.

.FUNC

General Form (SPICE or schematics)

.FUNC <*name*(<*p1*>[,<*p2*>][...,<*pn*>])> f(<*p1*>[,<*p2*>][...,<*pn*>])

This command is similar to the macro form of the .DEFINE command and is included as a separate command because it is sometimes used in commercial models. The function name must not be the same as any of the predefined functions such as sin, cos, exp, etc.

Examples

```
.FUNC MAX3(A,B,C) MAX(MAX(A,B),C)
.FUNC QUAD(A,B,C,X) A*X^2+B*X+C
.FUNC DIVIDER(A,B,C) V(B,C)/V(A,C)
```

.HELP

General Form (Schematics only)

.HELP <*parameter name*> <"*help text*">

This command places parameter help text in a macro schematic. The *help text* is displayed in the Status bar of the Attribute dialog box when a macro is placed or edited and the cursor is over *parameter name*.

<*parameter name*> should be one of the macro parameters listed in the .PARAMETERS statement.

Here are several examples:

.HELP VP "Peak magnitude of the output signal"

.HELP KF "Frequency sensitivity in Hz/Volt"

.IC

General Forms (SPICE or schematics)
Analog nodes
.IC <V(<*analog node1*>[,<*analog node2*>]) = <*voltage value*>>*

Inductors
.IC <I(<*inductor*>) = <*current value*>>*

Digital nodes
.IC <D(<*digital node*>) = <*digital value*>>*

Examples
.IC V(VOUT)=2.0
.IC I(L1)=6.0 V(3)=2
.IC D(1440)=0
.IC D(DIN)=X D(12)=1

The .IC statement assigns initial voltages, inductor currents, and digital logic states during the AC and transient analysis operating point calculation, and during the first data point in DC analysis. It assigns the analog or digital value to the node or branch and holds the value during the entire operating point calculation. After the operating point calculation, the node is released. If a .NODESET and an .IC statement are both present in the circuit, the .IC statement takes precedence. That is, the .NODESET statement is ignored.

Note, using .IC statements to set the voltage across inductors or voltage sources is futile. During the operating point calculation inductor voltages are set to zero, and voltage sources assume their TIME=0 value.

The IC statement works as follows in AC and transient analysis:

1. If transient analysis is being run and the Operating Point option is enabled or if AC analysis is being run, a DC operating point is calculated. The values specified in the .IC statement are fixed during the operating point calculation.

2. If transient analysis is being run and the Operating Point option is disabled: A DC operating point is not calculated. The initial condition appearing in the .IC statement and device initializations are assigned, and the first time point of transient analysis begins using these initial values.

.INCLUDE

General Form (SPICE or schematics)
.INC[LUDE] <"Filename">

Examples
.INCLUDE "C:\MC7\DATA\EX1DEF.TXT"
.INC "C:\MC7\DATA\MYSMALL.LIB"

This statement copies statements from an external text file into a schematic or SPICE file prior to an analysis. This is useful if you have an external text file library of model statements or you want all your statements contained in a separate DOS file. Use the Text tool in the Schematic Editor to add the statement to a schematic. <"Filename"> may include a path. The quotation marks are optional.

This command includes all of the text in the file and can quickly exhaust memory if the included file is large. It should only be used with small text files. For large files, use the .LIB command.

.LIB

General Form (SPICE or schematics)
.LIB ["Filename"]

Examples
.LIB
.LIB "C:\MC7\DATA\BIPOLAR.LIB"

The .LIB command is both an alternative and a supplement to placing model statements in a schematic or SPICE file. It accesses device models from binary library files (*.LBR) or .MACRO, .MODEL, or .SUBCKT statements from text files (*.LIB). "Filename" is any legal file name and may include a path. Quotation marks are optional. There is no default extension, so you must include the file name extension. .LIB files may contain .MODEL, .SUBCKT, .MACRO, .ENDS, .PARAM, or .LIB statements. Other statements are ignored. Lines are nullified with a "*" at the start of the line. A ";" nullifies the remaining portion of the line.

"Filename" defaults to NOM.LIB. The original NOM.LIB supplied with MC7 accesses the entire Model library by listing each of the constituent model library files. *The default command .LIB NOM.LIB is automatically applied to every circuit and is the main access mechanism to the MC7 Model library.*

Whenever MC7 needs model information from a macro statement, model statement, or a subcircuit it will look in these places in the order shown:

- If the circuit is a schematic:
  - In the grid text or text area.
  - In the file named in the File attribute (if the device has one).
  - In any file listed in a .LIB statement contained within the circuit.
  - In any file named in the master NOM.LIB file.

- If the circuit is a SPICE text file:
  - In the circuit description text.
  - In any file listed in a .LIB statement contained within the circuit.
  - In any file named in the master NOM.LIB file.

When searching for model library files, MC7 scans the library folders specified at **File / Paths**. If more than one folder is specified, it searches in left to right order. Should the search fail, an error message is issued. Generally, MC7 first looks for model information locally within the circuit, then globally in the library folders.

.LOADOP

General Form (Schematics only)

.LOADOP

Example

.LOADOP

This statement loads a previously saved State Variables file prior to the operating point calculation in AC analysis. The contents of the file are expected to provide a good approximation to a stable operating point.

The command lets you use transient analysis to produce an operating point, save it with the State Variables Editor, then use it as an initial estimate for an AC analysis operating point. The name of the file loaded is:

CIRCUITNAME.TOP

This command is equivalent to a set of .NODESET commands for each node in the circuit.

.MACRO

General Form (Schematics)

.MACRO <*alias*> <*macro circuit name*(*parameter list*)>

Examples

.MACRO MCR3818_2 SCR(50m,40m,1u,1,50,50MEG,20u,.5,1)
.MACRO MAC320_4 TRIAC(6m,50m,1.5u,1.4,200,50MEG,0,1,1)
.MACRO KDS_049_S XTAL(4.9152MEG,120,30K)

This statement functions much like a .DEFINE statement. It provides a way to replace lengthy macro parameter calls in a macro's VALUE attribute with short model names that compactly describe the behavior of the macro.

For an example of how this works, load the library file THY_LIB.LIB and search for B25RIA10. You will find the following .MACRO statement :

.macro B25RIA10 SCR(100m,60m,.9u,.9,100,100MEG,110u,1,1)

When you run an analysis on a circuit that uses this part, the FILE attribute B25RIA10 is replaced by SCR and the parameters are assigned values from the macro statement (100m,60m,.9u,.9,100,100MEG,110u,1,1) in the same order as defined in the .PARAMETERS statement within the SCR macro itself.

The SCR macro's statement is:

.PARAMETERS(IH=50mA, IGT=40mA,TON=1uS, VTMIN=1V, VDRM=50V, DVDT=50Meg,TQ=20Us,K1=1, K2=1)

When a B25RIA10 is used, its IH parameter is set to 100m, its IGT parameter is set to 60m, its TON parameter is set to .9u, and so on.

.MODEL

General Form (SPICE or schematics)

.MODEL <*model name*> [AKO:<*reference model name*>] <*model type*>
+ ([<*parameter name*>=<*value*>]
+ [LOT[*t&d*]=<*value*>[%]] [DEV[*t&d*]=<*value*>[%]])

Examples

.MODEL Q1 NPN (IS=1e-15)
.MODEL VIN1 PUL (Vone=10V p1=0 p2=.1u p3=10u p4=10.1u p5=15u)
.MODEL M1 NMOS (Level=3 VTO=2.5 LOT=30% DEV=1%)
.MODEL R1 RES (R=2.0 TC1=.015)
.MODEL 2N2222A AKO:2N2222 NPN (BF=55 ISE=10F)
.MODEL NPN_A NPN (RE=12 LOT/1/GAUSS=30% DEV/2/UNIFORM=2%)

The model statement is one way to define the electrical behavior of a device. Others include binary model libraries created by the Model Editor or the MODEL program. <*model name*> is the name used to reference or access a particular model. <*value*> may contain simple run-invariant expressions as in this example:

.MODEL M2 NMOS (VTO=3.5+TEMP*.0015...

This is legal because TEMP (operating temperature) is constant during the run.

The AKO (an acronym for A Kind Of) option lets you clone new models from existing ones. All of the AKO parameters except LOT and DEV tolerances are identical to the parent, except where overridden by the specified model values. In the following example, the 1N914A has the same parameters as the 1N914 except RS, which is 10.

.MODEL 1N914A AKO:1N914 D (RS=10)

Tolerances may be specified as an actual value or as a percentage of the nominal parameter value. Both absolute tracking tolerances (LOT) and relative (DEV) tolerances may be specified.

Both types of tolerance are specified by placing a keyword after the parameter:

[LOT[*t&d*]=<tol1>[%]] [DEV[*t&d*]=<tol2>[%]]

This example specifies a 10% tolerance for the forward beta of Q1.

.MODEL Q1 NPN (BF=100 LOT=10%)

[*t&d*] specifies the tracking and distribution, using the following format:

[/<*lot#*>][/<*distribution name*>]

These specifications must follow the keywords DEV and LOT without spaces and must be separated by "/".

<*lot#*> specifies which of ten random number generators, numbered 0 through 9, are used to calculate parameter values. This lets you correlate parameters of an individual model statement (e.g. RE and RC of a particular NPN transistor model) as well as parameters between models (e.g. BF of NPNA and BF of NPNB). The DEV random number generators are distinct from the LOT random number generators. Tolerances without <*lot#*> get unique random numbers.

<*distribution name*> specifies the distribution. It can be any of the following:

Keyword	**Distribution**
UNIFORM	Equal probability distribution
GAUSS	Normal or Gaussian distribution
WCASE	Worst case distribution

If a distribution is not specified in [*t&d*], the distribution specified in the Monte Carlo dialog box is used.

The model statements of the capacitor, inductor, resistor, diode, GaAsFET, JFET, MOSFET, and BJT devices can individually control the two temperatures:

Measurement temperature: This is the temperature at which the model parameters are assumed to have been measured. It serves as a reference point for temperature adjusting the parameter values. The default value is the value set by a .OPTIONS TNOM statement, if present, or if not then the TNOM Global Settings value (which defaults to 27° C).

Device operating temperature: This is the temperature used to adjust the model parameters from their measured values.

To modify the measurement temperature, specify a value for the model parameter, T_MEASURED. For example,

.Model M710 NMOS (Level=3 VTO=2.5 T_MEASURED=35)

There are three ways to modify the device operating temperature:

Keyword	**Device operating temperature**
T_ABS	T_ABS
T_REL_LOCAL	T_REL_LOCAL + T_ABS(of AKO parent)
T_REL_GLOBAL	T_REL_GLOBAL + global temperature

Global temperature is determined as follows:

SPICE circuits
In SPICE circuits, global temperature is set by the .TEMP statement, if present, or a .OPTIONS TNOM=XXX, if present, or by the TNOM Global Settings value. When you select an analysis type, global temperature is determined and placed in the Temperature field of the Analysis Limits dialog box. When this dialog box comes up, you can change the temperature prior to starting the analysis run.

Schematics
In schematics, global temperature is the value in the Temperature field of the Analysis Limits dialog box. .TEMP statements have no effect.

Examples
In this example, the operating temperature of N1 is 47° C:

```
.TEMP 47
.MODEL N1 NPN(BF=50)
```

In this example, the operating temperature is 35° C:

```
.MODEL N1 NPN(BF=50 T_ABS=35)
```

In this example, the operating temperature of N1 is 30° C and of N2 is 55° C:

```
.MODEL N1 NPN(BF=50 T_ABS=30)
.MODEL N2 AKO:N1 NPN(T_REL_LOCAL=25)
```

In this example, the operating temperature of N1 is 75° C:

```
.TEMP 35
.MODEL N1 NPN(BF=50 T_REL_GLOBAL=40)
```

See Chapter 22, "Analog Devices" for more information on the exact effects of device operating temperature on particular device parameters.

.NODESET

General Forms (SPICE or schematics)
Analog nodes
.NODESET <V(<*analog node1*>[,<*analog node2*>]) = <*voltage value*>>*

Inductors
.NODESET <I(<*inductor*>) = <*current value*>>*

Digital nodes
.NODESET <D(<*digital node*>) = <*digital value*>>*

Examples
.NODESET V(IN1)=45UV V(OUT)=1.2MV
.NODESET V(7)=4 D(H1)=1
.NODESET I(L10)=3.5ma

The .NODESET statement provides an initial guess for node voltages, currents, and digital logic states during the AC and transient analysis operating point calculation. It assigns the analog or digital value to the node but, unlike the .IC statement, it does not hold the value during the entire operating point calculation. If both an .IC and a .NODESET statement are present in the circuit, the .IC statement takes precedence (the .NODESET statement is ignored).

Note that using .NODESET statements to set the voltage across inductors or voltage sources is futile as their initial voltages are predetermined.

This command can affect transient analysis even when the operating point is skipped, because it sets state variables.

.NOISE

General Form (SPICE files only)
.NOISE V(<*node1*>[,<*node2*>]) <*source name*> [<*interval value*>]

Examples
.NOISE V(10) V1
.NOISE V(4,3) VAC1 50
.NOISE V(10,12) I1 100

The noise command arguments are placed into the proper MC7 Analysis Limits dialog box fields upon selecting **AC** from the **Analysis** menu. The user should then select either ONOISE, INOISE, or both, for the X or Y expression of one of the plots. Simply selecting one of these variables enables the Noise analysis mode.

V(<node1>[,<node2>]) is any valid AC voltage variable and defines the output at which to measure the noise value. <source name> can be either an independent current or voltage source. It defines the input node(s) at which to calculate the equivalent input noise (inoise).

Noise analysis calculates the RMS value of the noise at the output specified by V(<node1>[,<node2>]), arising from the contributions of all noise sources within the circuit. All semiconductors and resistors contribute noise.

Note that selecting one of the noise variables precludes plotting normal AC small signal variables, like node voltages. You can't, for instance, plot INOISE and V(1) at the same time.

.OP

General Form (SPICE files only)
.OP

Example
.OP

In a SPICE file, this command normally specifies that the results of the operating point calculation be printed. Since these are always printed to the Numeric Output window and the CIRCUITNAME.TNO or CIRCUITNAME.ANO file anyway, this command is redundant and has no effect.

.OPT[IONS]

General Form (SPICE or schematics)
.OPTIONS [<*option name*>]* [<*option name*>=<*option value*>]*

Examples
.options GMIN=1e-9 VNTOL =1n ABSTOL=1n DEFAS=.1u
.options NOOUTMSG

This command lets you change the value of one or more constants for a particular circuit. The Options statement overrides Global Settings values. Individual options are described in detail in the Global Settings section of Chapter 2, "The Circuit Editor".

.PARAM

General Form (SPICE or schematics)
.PARAM <<*name*> = {<*expression*>}>*

Examples
.PARAM VSS = 5 VEE = -12
.PARAM RISETIME={PERIOD/10}

The .PARAM statement is similar to the .DEFINE statement. It is available for use in both circuit files and libraries. It is provided mainly to ensure compatibility with some commercial vendor libraries.

Curly brackets, "{" and "}" must enclose the definition and usage of the variable. Brackets are not required in the .PARAM statement if the variable is a constant, as in the first example above.

For example, if BF1 were defined as follows:

.PARAM BF1={100+TEMP/20}

Then you could use it in a model statement like so:

.MODEL Q1 NPN (BF={BF1})

Here is the equivalent .DEFINE statement:

.DEFINE BF1 100+TEMP/20

Here is the equivalent usage:

.MODEL Q1 NPN (BF=BF1)

Essentially, the .PARAM statement requires curly brackets in definition and use and the .DEFINE does not.

As with the .DEFINE statement, <*name*> can't be a reserved variable name, such as T (Time), F (Frequency), or S (complex frequency), or a reserved constant name such as VT, TEMP, PI, or GMIN.

.PARAMETERS

General Form (Schematics only)
.PARAMETERS (<*parameter*[=<value>]> [, <*parameter*[=<value>]>]*)

Examples
.parameters(GBW, Slew, Iscp, F1=1K, F2=1.1K)
.Parameters(Gain,ROUT=50)

The .PARAMETERS statement is placed in a macro circuit as grid text or in the text area and declares the names of parameters to be passed to it from the calling circuit. A parameter is a numeric value that you pass to a macro circuit from the macro's VALUE attribute (or an equivalent .MACRO statement). It can be used in the macro circuit as, for example, the VALUE attribute of a resistor, or as a model parameter value, such as the BF of a BJT transistor.

The optional default [=<value>] specifies a parameter's default value. This value is assigned to the parameter when the part is placed in the schematic. The value can be edited from the Attribute dialog box. If the parameter's default value is not specified in the .PARAMETERS statement within the macro, its value must be specified when the macro is placed in a circuit.

See the macro circuits SCR, XTAL, PUT, or TRIAC for examples of this command.

.PLOT

General Form (SPICE files only)

.PLOT <*analysis type*> [<*output variable*>]*
+ ([<*lower limit value*>,<*upper limit value*>])*

Examples

.PLOT AC V(10) V(1,2) (0,10)
.PLOT TRAN V(1) D(2)

This statement specifies what is to be plotted in the analysis plot. It enters the specified variables as simple waveform expressions in the Analysis Limits Y expression fields. <analysis type> is one of the following AC, DC, NOISE, or TRAN. Output variables can be any valid node voltage, source current, or digital state.

.PRINT

General Form (SPICE files only)
.PRINT *<analysis type>* [*<output variable>*]*

Examples
.PRINT AC V(1) V(3)
.PRINT TRAN V(1) D(10)

This statement specifies what is to be printed in the Numeric Output window. It enters the specified variables as simple waveform expressions in the Analysis Limits Y expression fields. <analysis type> is one of the following: AC, DC, NOISE, or TRAN. Output variables include node voltages, source currents, and digital states. Numeric Output window contents are also saved in one of the following files:

CIRCUITNAME.TNO	Transient analysis
CIRCUITNAME.ANO	AC analysis
CIRCUITNAME.DNO	DC analysis

.SENS

General Form (SPICE files only)
.SENS <*output expression*> [<*output expression*>]*

Examples
.SENS V(1) V(3)
.SENS V(D1)*I(D1)

This statement controls the sensitivity analysis feature. It tells the program to calculate the DC sensitivity of each specified <*output expression*> to the default parameters. Default parameters include a subset of all possible model parameters. These can be changed from the Sensitivity analysis dialog box.

.SUBCKT

General Form (SPICE files and schematic text areas only)
.SUBCKT <*subcircuit name*> [<*node*>]*
+ [OPTIONAL:<<*node*>=<*default value*>>*]
+ [PARAMS:<<*parameter name*>=<*parameter default value*>>*]
+ [TEXT:<<*text name*>=<*text default value*>>*]

Examples
.SUBCKT LT1037 1 2 3 50 99
.SUBCKT CLIP IN OUT PARAMS: LOW=0 HIGH=10
.SUBCKT 7400 D1 D2 Y1
+ OPTIONAL: DPWR=44 DGND=55

This statement marks the beginning of a subcircuit definition. The subcircuit definition ends with the .ENDS statement. All statements between the .SUBCKT and the .ENDS statements are included in the subcircuit definition.

<*subcircuit name*> is the name of the subcircuit and is the name used when the subcircuit is called or used by another circuit.

[<*node*>]* are the node numbers passed from the subcircuit to the calling circuit. The number of nodes in the subcircuit call must be the same as the number of nodes in the .SUBCKT statement. When the subcircuit is called, the nodes in the call are substituted for the nodes in the body of the subcircuit in the same order as in the .SUBCKT statement. Consider this example:

```
X1 1 2 BLOCK

.SUBCKT BLOCK 10 20
R1 10 0 1K
R2 20 0 2K
.ENDS
```

In this example, the resistor R1 is connected between node 1 and node 0 and R2 is connected between node 2 and node 0.

The OPTIONAL keyword lets you add one or more nodes to the subcircuit call. If the nodes are added, they override the default node values. This option is often used to override the default digital global power pins. In the subcircuit call, you may specify one or more of the optional nodes in the call, but you must specify all nodes prior to and including the last one you want to specify. You can't skip any nodes, since the parser can't tell which nodes are being skipped. Consider this:

```
.SUBCKT GATE 1 2
+ OPTIONAL: A=100 B=200 C=300
```

Any of the following are legal calls:

```
X1 1 2 GATE              ; results in A=100 B=200 C=300
X2 1 2 20 GATE           ; results in A=20 B=200 C=300
X3 1 2 20 30 GATE        ; results in A=20 B =30 C=300
X4 1 2 20 30 40 GATE     ; results in A=20 B =30 C=40
```

The keyword PARAMS lets you pass multiple numeric parameters to the subcircuit. <*parameter name*> defines its name and <*parameter default value*> defines the value it will assume if the parameter is not included in the subcircuit call. For example:

```
.SUBCKT BAND 10 20 30
+ PARAMS: F0=10K BW=1K
```

Any of these calls are legal:

```
X1 10 20 30 BAND                           ;Yields F0=10K BW=1K
X2 10 20 30 BAND PARAMS: F0=50K BW=2K      ;Yields F0=50K BW=2K
X3 10 20 30 BAND PARAMS: BW=2K             ;Yields F0=10K BW=2K
```

The keyword TEXT lets you pass text parameters to the subcircuit. <text name> defines the name of the text parameter and <text default value> defines the value it will assume if the parameter is not included in the subcircuit call. For example:

```
.SUBCKT PLA 1 2 3 4
+ TEXT: FILE="JEDEC_10"
```

Either of these calls are legal:

```
X1 10 20 30 40 PLA                        ;Yields FILE="JEDEC_10"
X2 10 20 30 40 PLA TEXT:FILE="JE20"       ;Yields FILE="JE20"
```

The text parameter may be used:

- To specify a JEDEC file name of a PLD component.
- To specify a stimulus file name for a FSTIM device.
- To specify a text parameter to a subcircuit.
- As a part of a text expression in one of the above.

.TEMP

General Form (SPICE files only)
.TEMP <*temperature value*>

Examples
.TEMP 50
.TEMP 0 50 100

The .TEMP statement specifies the operating temperature at which the circuit will be analyzed. The default value is 27 ° Centigrade. Temperature dependent parameters are a function of the difference between the operating temperature and the measurement temperature.

The temperature at which device parameters are assumed to have been measured is called the measurement temperature or TNOM. It is obtained from a .OPTIONS TNOM=XX statement. If this statement is not present in the circuit, the Global Settings TNOM value is used instead.

.TF

General Form (SPICE files only)
.TF <*output expression*> <*input source name*>

Examples
.TF V(OUT) V1
.TF VBE(Q1)*IB(Q1)   VIN

In this type of analysis, the program calculates the small-signal DC transfer function from the specified <*input source name*> to the specified <*output expression*>. It also calculates the small-signal DC input and output resistances.

.TIE

General Form (Schematics)

.TIE <*part name*> <*pin name*>

Examples

.TIE JKFF CLKB
.TIE LF155 VCC

The .TIE statement connects together all of the specified <*pin name*> pins of the specified <*part name*> parts. This is a convenient way of simultaneously connecting many common pins. It is normally used for power, clock, reset, and preset pins. The first example above specifies that the CLKB pins of all JKFF parts are to be connected together.

Note that <*part name*> is the general part name from the Component library, not a schematic part name. For example, in a circuit with three JKFF parts, named U1, U2, and U3, the CLKB pins would be connected with .TIE JKFF CLKB.

.TR

General Form (Schematics)
.TR <s1 t1> [s2 t2...sn-1 tn-1 sn tn]

The .TR statement lets you set the maximum time step during different parts of transient analysis.

Examples
.TR 1n 100n .1n 200n 10n 1u

In this example the time step is limited to 1n from tmin to 100n. Between 100ns and 200ns it is limited to .1n. Between 200ns and 1us it is limited to 10n.

The purpose of this command is to provide some flexibility in simulating difficult circuits. It is mainly used when the circuit needs a more conservative (smaller) maximum time step during a critical part of the simulation but would have a very long run time if the smaller maximum time step were specified for the entire run.

Without this command, the Maximum Time Step parameter specified in the Analysis Limits dialog box controls the time step for the entire run.

.TRAN

General Form (SPICE files only)
.TRAN <*printstep*> <*run stop time*>
+ [<*print start time*> [<*max time step*>]] [UIC]

Examples
.TRAN 10ps 110ns
.TRAN 1ns 1us 500ns .5ns UIC

The .TRAN command arguments control the transient analysis run parameters. They are copied into the appropriate Analysis Limits dialog box fields upon selecting **Transient** from the **Analysis** menu. Here is the conversion:

.TRAN value	MC7 effect or assignment
<*printstep*>	Number of Points = 1 + <*run stop time*> / <*printstep*>
<*run stop time*>	Time Range = <*run stop time*>
<*print start time*>	No effect
<*max time step*>	Maximum Time Step = <*max time step*>
UIC	Disables the Operating Point option

The <*printstep*> value declares the time interval between numeric printouts. The <*run stop time*> value specifies the last time point to be simulated. The UIC keyword, an acronym for Use Initial Conditions, instructs the simulator to skip the usual operating point calculation and use the initial conditions as specified in the .IC statements and device initial conditions.

Chapter 21 Analog Behavioral Modeling

What's in this chapter

This chapter describes the analog behavioral modeling capability and macro circuit blocks. These macro circuit files are provided with the MC7 package:

- Absolute value (full wave rectifier) function (ABS)
- Amplifier (AMP)
- Center-tapped transformer (CENTAP)
- Clipping limiter circuit (CLIP)
- Delay (DELAY)
- Differentiator (DIF)
- Divider (DIV)
- Linear transfer function (F)
- Frequency-shift keyer (FSK)
- Gyrator for impedance transformation (GYRATOR)
- Integrator (INT)
- Two-input weighted multiplier (MUL)
- Time domain noise source (NOISE)
- Potentiometer (POT)
- Phase-shift keyer (PSK)
- Programmable Unijunction Transistor (PUT)
- Relays (RELAY1 AND RELAY2)
- Resonant circuit (RESONANT)
- Schmitt trigger circuit (SCHMITT)
- Silicon-controlled rectifier (SCR)
- Slip or hysteresis function (SLIP)
- Spark gap device (SPARKGAP)
- Two-input weighted subtracter (SUB)
- Two-input weighted summer (SUM)
- Three-input weighted summer (SUM3)
- Triac (TRIAC)
- Trigger (TRIGGER)
- Triode vacuum tube model (TRIODE)
- Voltage-controlled oscillator (VCO)
- Wideband transformer (WIDEBAND)
- Crystal model (XTAL)
- 555 Timer (555)

Laplace sources

Laplace sources are controlled sources whose transfer function is a function of the complex frequency variable, S= 2*PI*J*F. There are eight basic types:

Expression-defined sources:

Current-controlled current source	LFIOFI
Current-controlled voltage source	LFVOFI
Voltage-controlled voltage source	LFVOFV
Voltage-controlled current source	LFIOFV

Table-defined sources:

Current-controlled current source	LTIOFI
Current-controlled voltage source	LTVOFI
Voltage-controlled voltage source	LTVOFV
Voltage-controlled current source	LTIOFV

The presence of the complex frequency variable, S, lets you describe more than simple gain blocks. This type of source lets you define an arbitrary linear transfer function block with any combination of poles and zeros. In fact, you can define virtually any linear S domain function that can be expressed either as an algebraic formula or as a table of ordered triplets (frequency, magnitude, phase).

For transient analysis, MC7 first determines the impulse response of the function. The impulse response is obtained by performing an inverse Fourier transform on the transfer function. During the transient analysis, the output of the source is obtained from the convolution of the actual waveform at the source input nodes and the stored impulse response waveform. This allows the Laplace source to accurately respond to any input waveform, not just simple, predefined waveforms.

In AC analysis, the transfer function value is computed from the expression involving S, where $S = 2 \cdot \pi \cdot \text{frequency} \cdot j$, or interpolated from the given table.

For DC analysis, the transfer function value is computed from the expression with S = 0, or obtained from the table using the lowest frequency data point.

Here are some examples:

1/(1+.001*S)	A low pass filter
.01*S/(1+.01*S)	A high pass filter
exp(-pow((C*S*(R+S*L)),.5))	Simple lossy, transmission line

Function sources

Function sources are controlled sources whose transfer function is a function of everything but the complex frequency variable. Like Laplace sources, there are four basic types; current and voltage dependent and current and voltage output. There are both expression and table-defined versions.

The table-defined sources come in the usual four varieties:

Current-controlled current source	NTIOFI
Current-controlled voltage source	NTVOFI
Voltage-controlled voltage source	NTVOFV
Voltage-controlled current source	NTIOFV

A table of ordered pairs (in,out) describes the time domain input-output relationship. For AC analysis, the source is linearized about the operating point, and the resulting linear real transfer function is used.

The algebraic sources come in two varieties:

Voltage sources	NFV
Current sources	NFI

Because you define the output value directly in an expression, rather than describing the transfer function, the sources can be used as either dependent or independent sources. The 'input' is implicitly contained in the expression as a variable.

Here are some examples:

10*TANH(I(L1))	Nonlinear inductor expression
10PF/(1+V(D1)/.45)	Nonlinear varactor expression
10*SIN(2*PI*10*T)*EXP(-T/(R(R1)*5))	Damped Sine wave source

ABS

Sometimes referred to as a full-wave rectifier, the ABS block provides the absolute value of the input signal. Its definition is:

$$V_{Out}(t) = | V_{In}(t) |$$

The function is implemented with the ABS macro:

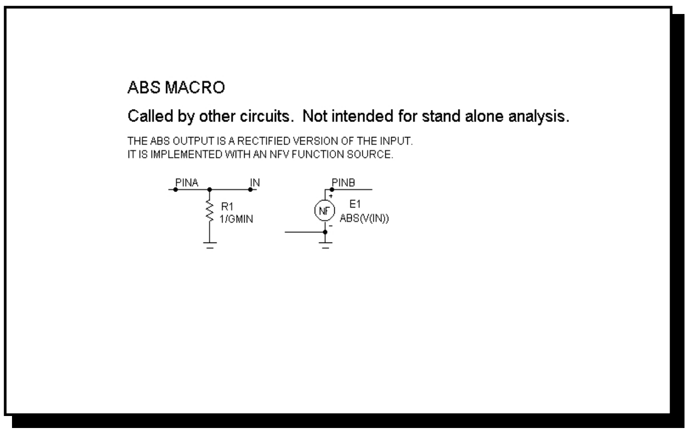


Figure 21-1 ABS macro equivalent circuit

There are no input parameters. This implementation uses an NFV Function source to provide the absolute value function. The macro block mainly serves to provide a more suitable symbol than the general source symbol of the NFV Function source.

See the circuit SYSTEM2 for an example of the use of this macro.

AMP

This block provides a simple linear amplifier. Its definition is:

$$V_{Out}(t) = gain\ V_{In}(t)$$

The function is implemented with the AMP macro:

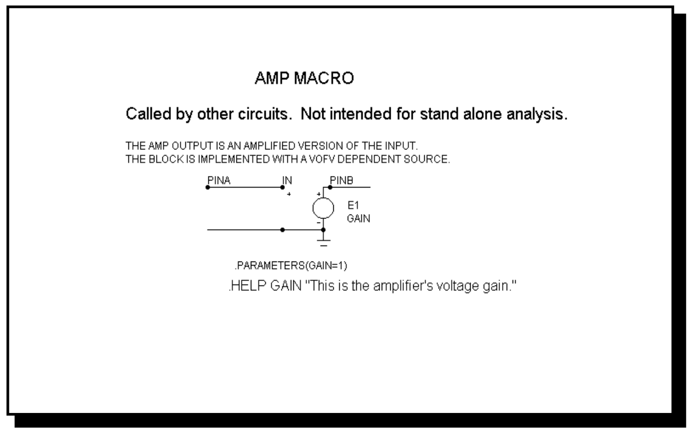


Figure 21-2 AMP macro equivalent circuit

The single input parameter, GAIN, multiplies the input to produce an amplified output. This implementation uses a simple linear dependent VOFV source. It could have been done with a Function source or a Spice poly source. In general, the simplest type of source that will perform the function is preferred.

CENTAP

The CENTAP macro is a center-tapped five terminal transformer with parameters defining the primary and two secondary inductances

This circuit is implemented with three linear inductors and one K (coupling) device.

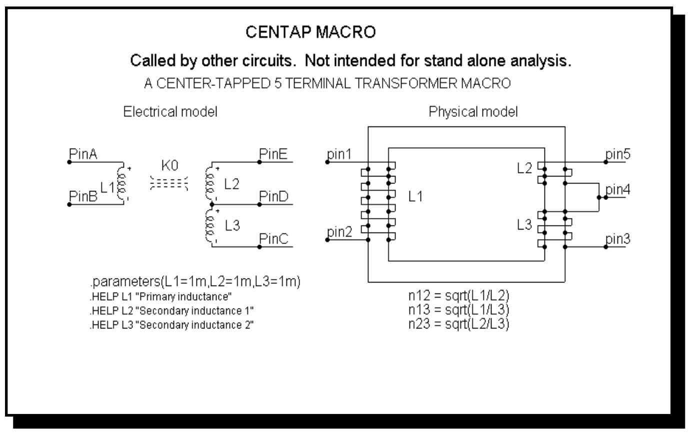


Figure 21-3 CENTAP macro equivalent circuit

Parameter	**Definition**
L1	Primary inductance
L2	Secondary inductance 1
L3	Secondary inductance 2

See the circuit TRANS for an example of the use of this macro.

CLIP

The clip macro can be used as a limiter, ideal OPAMP, or inverter. It provides an output that is a scaled copy of the input, but limited to the specified maximum and minimum levels.

This function is implemented with the CLIP macro:

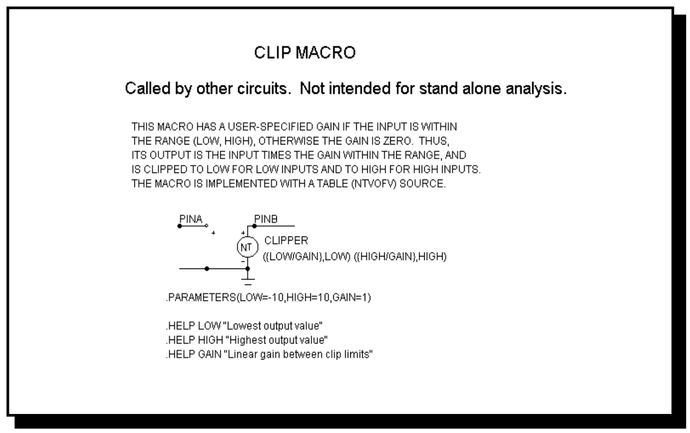


Figure 21-4 CLIP macro equivalent circuit

A pair of input parameters, LOW and HIGH, define the lowest value and highest value of the output. Between these limits, the output equals the input multiplied by the GAIN parameter. The block is constructed of a NTVOFV Function table source.

Parameter	**Definition**
GAIN	Linear gain between clip limits
LOW	Lowest output value
HIGH	Highest output value

See the circuit SYSTEM2 for an example of the use of this macro.

DELAY

The delay macro provides a programmable time delay.

$$V_{Out}(t+delay) = V_{Out}(t)$$

This function is implemented with the DELAY macro:

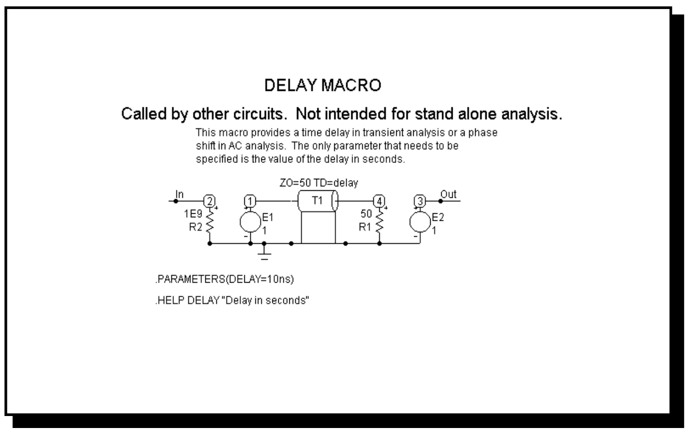


Figure 21-5 DELAY macro equivalent circuit

The single input parameter, DELAY, provides the specified time delay through a transmission line.

DIF

The differentiator is the inverse of the integrator. It provides an output which is a scaled version of the time derivative of the input signal:

$$V_{Out}(t) = scale\ \ d(V_{In}(t))/dt$$

This function is implemented with the DIF macro:

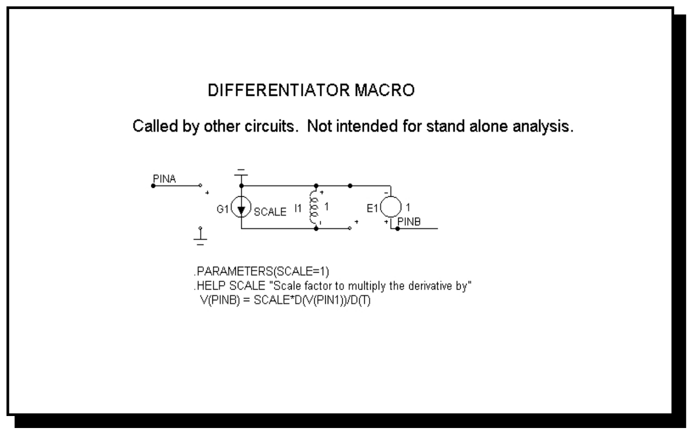


Figure 21-6 DIF macro equivalent circuit

The single input parameter, SCALE, multiplies or scales the derivative. This particular implementation has a buffered output, allowing it to drive very low impedance networks.

DIV

Occasionally, system blocks require a function that divides two analog signals. The desired function is:

$$V_{Out}(t) = scale\ V_a(t)/V_b(t)$$

This function is implemented with the DIV macro:

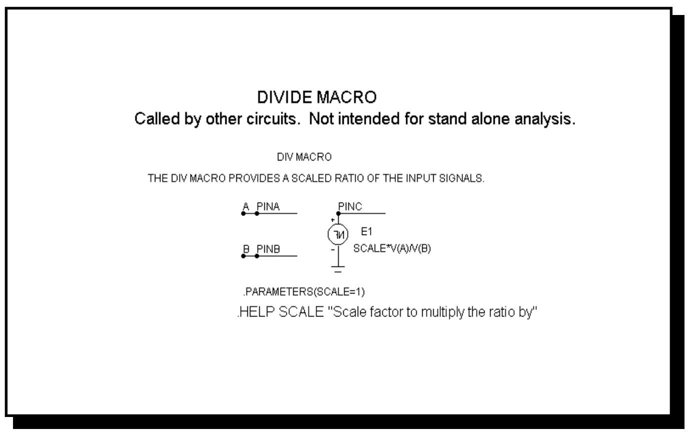


Figure 21-7 DIV macro equivalent circuit

The single input parameter passed to the macro by the calling circuit, SCALE, multiplies or scales the ratio of the two input waveforms at the output.

F

This system block merely provides a convenient shape to house a general linear transfer function, F(S). It is implemented with a Laplace LFVOFV source.

$$F(s) = V_{Out}(s) / V_{In}(s)$$

This function is implemented with the F macro:

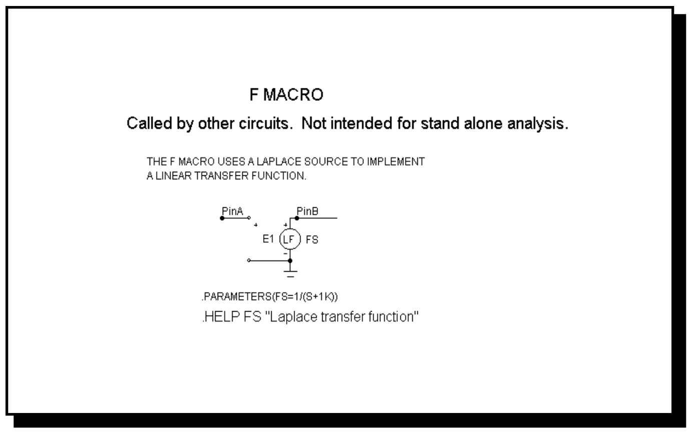


Figure 21-8 F macro equivalent circuit

The input parameter is an expression representing the complex frequency transfer function.

See the circuit SYSTEM2 for an example of the use of this macro.

FSK

This block provides a frequency-shift keyer encoder.

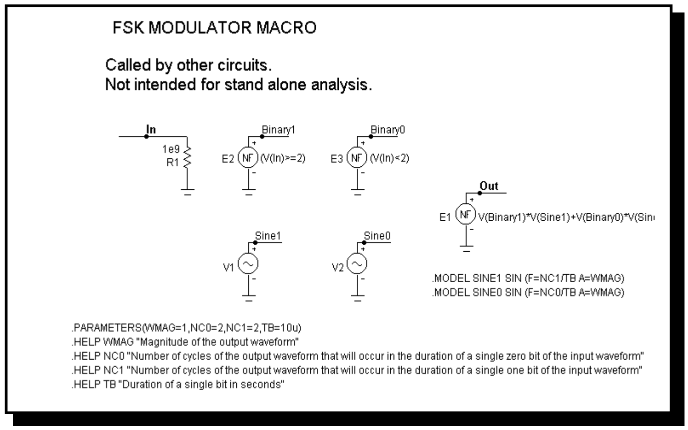


Figure 21-9 FSK macro equivalent circuit

The input parameters are as follows:

Parameter	**Definition**
WMAG	Magnitude of the output waveform
NC0	Number of cycles of the output waveform that will occur in the duration of a single zero bit of the input waveform
NC1	Number of cycles of the output waveform that will occur in the duration of a single one bit of the input waveform
TB	Duration of a single bit in seconds

See the circuit FSK2 for an example of the use of this macro.

GYRATOR

The gyrator can be used to scale a resistive impedance. It can also transform an inductive impedance to a capacitive impedance or a capacitive impedance to an inductive impedance.

This function is implemented with two cross-referenced linear dependent VOFI sources.

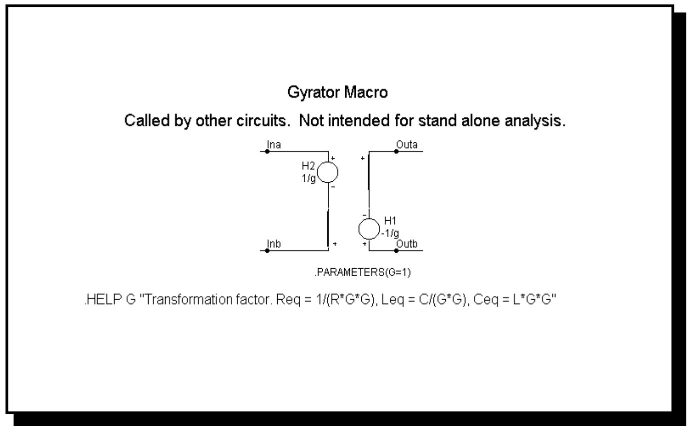


Figure 21-10 GYRATOR macro equivalent circuit

A single parameter, G, determines the impedance transformation as follows:

Req = 1/(R*G*G)

Leq = C/(G*G)

Ceq = L*G*G

See the circuit GYRTEST for an example of the use of this macro.

INT

One of the most useful functions for system modeling is the integrator. Ideally this function provides an output which is the integral of the input signal:

$$V_{Out}(t) = vinit + scale \int_0^t V_{In}(t)\, dt$$

This function is implemented with the INT macro:

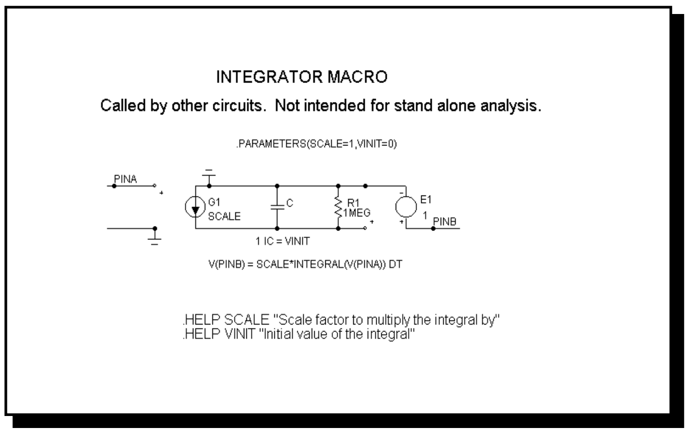


Figure 21-11 INT macro equivalent circuit

Two parameters are passed to the macro by the calling circuit: SCALE and VINIT. SCALE multiplies the integral and VINIT provides its initial value. This particular implementation has a buffered output, allowing it to drive very low impedance networks. It also has a voltage limiting resistor. The resistor keeps the output voltage finite when there is a DC voltage input. It should be large enough to avoid placing any practical limit on the frequency response.

Parameter	**Definition**
SCALE	Scale factor to multiply the integral by
VINIT	Initial value of the integral.

See the circuit SYSTEM1 for an example of the use of this macro.

MUL

Phase detectors and other system blocks require a function that multiplies two analog signals. The desired function is:

$$V_{Out}(t) = scale\ V_a(t)\ V_b(t)$$

This function is implemented with the MUL macro:

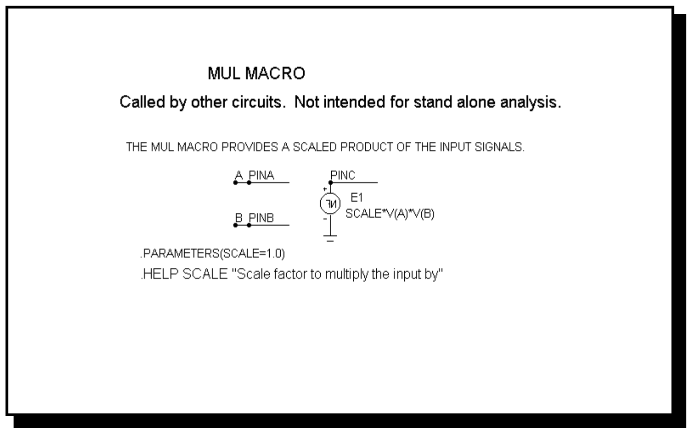


Figure 21-12 MUL macro equivalent circuit

The single input parameter passed to the macro by the calling circuit, SCALE, multiplies the product of the two input waveforms. The scaled product is provided at the output. This implementation is done with an NFV function source.

Parameter	**Definition**
SCALE	Scale factor to multiply the input by

NOISE

This macro implements a random noise generator to produce a noisy time-domain waveform.

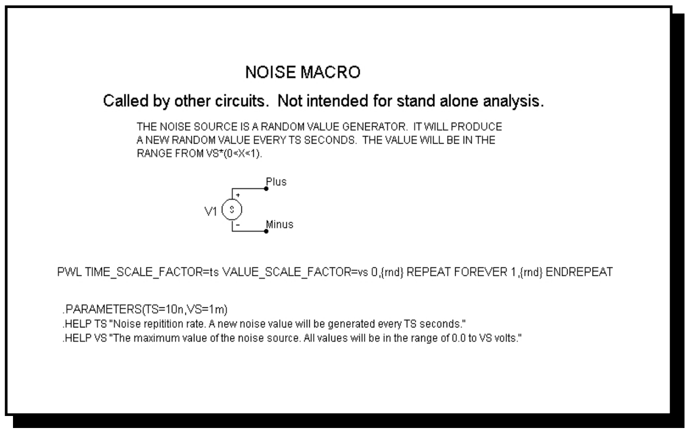


Figure 21-13 NOISE macro equivalent circuit

The parameter definitions are as follows:

Parameter	Definition
TS	Noise repetition rate. A new noise value will be generated every TS seconds.
VS	The maximum value of the noise source. All values will be in the range of 0.0 to VS volts.

POT

This macro implements a potentiometer using several resistor value expressions. The two parameters determine the initial pot setting. Percent is in % so 70% would be given as 70 and not .70.

To sweep the pot wiper arm, step the Value field of the macro resistor R1 from the Stepping dialog box. The stepped value will override the default percent parameter. See POTDEMO.CIR for an example.

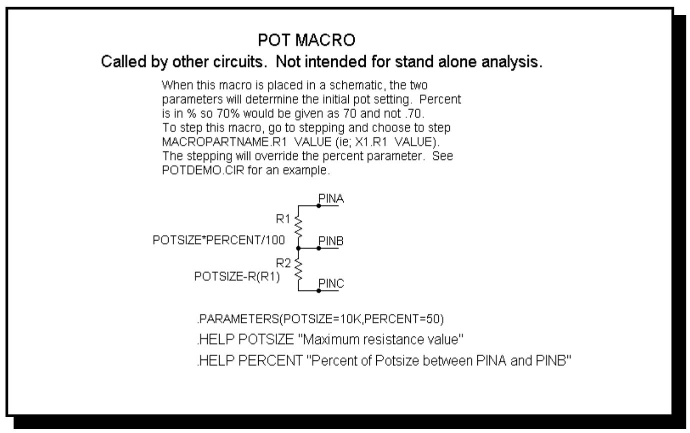


Figure 21-14 POT macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
Potsize	Maximum resistance value
Percent	Percent of Potsize between PINA and PINB

See the circuit POTDEMO for an example of the use of this macro.

PSK

This is a phase-shift keyer.

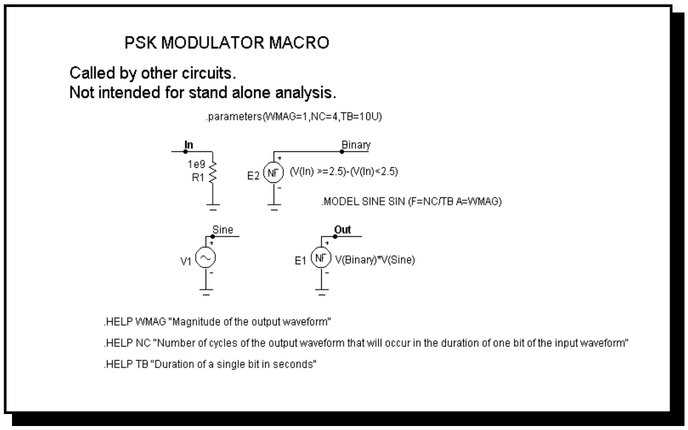


Figure 21-15 PSK macro equivalent circuit

The parameter definitions are as follows:

Parameter	Definition
WMAG	Magnitude of the output waveform
NC	Number of cycles of the output waveform that will occur in the duration of one bit of the input waveform
TB	Duration of a single bit in seconds

See the circuit PSK2 for an example of the use of this macro.

PUT

This circuit is a macro model for a PUT, a Programmable Unijunction Transistor. The macro equivalent circuit is as follows:

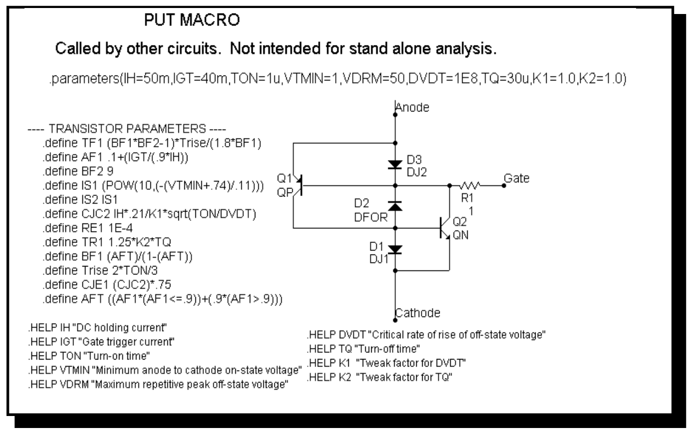


Figure 21-16 PUT macro equivalent circuit

The parameter definitions are as follows:

Parameter	Definition
IH	DC holding current
IGT	Gate trigger current
TON	Turn-on time
VTMIN	Minimum anode to cathode on-state voltage
VDRM	Maximum repetitive peak off-state voltage
DVDT	Critical rate of rise of off-state voltage
TQ	Turn-off time
K1	Tweak factor for DVDT
K2	Tweak factor for TQ

See the circuit THY1 for an example of the use of this macro.

RELAY1

This relay model includes a user-specified coil resistance and inductance. The coil current is sensed and converted to a voltage by H1 which drives a Schmitt macro to provide hysteresis between the ION and IHOLD currents. The output of the Schmitt drives a standard voltage controlled switch S1.

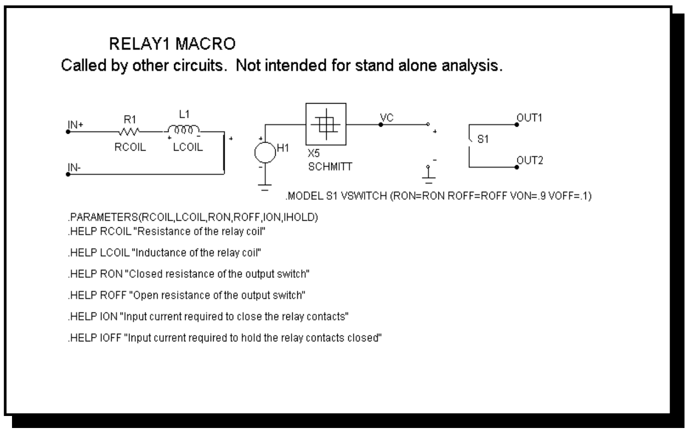


Figure 21-17 RELAY1 macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
RCOIL	Resistance of the relay coil
LCOIL	Inductance of the relay coil
RON	Closed resistance of the output switch
ROFF	Open resistance of the output switch
ION	Input current required to close the relay contacts
IOFF	Input current required to hold the relay contacts closed

See the circuit RELAY for an example of the use of this macro.

RELAY2

This relay model includes a flux circuit and derives a magnetizing force from the flux. It then algebraically sums the magnetizing, stop, friction and restoring spring forces acting on the relay plunger to arrive at a net force which is integrated once to get the plunger velocity and again to get the plunger position. This plunger position directly controls the switch contacts.

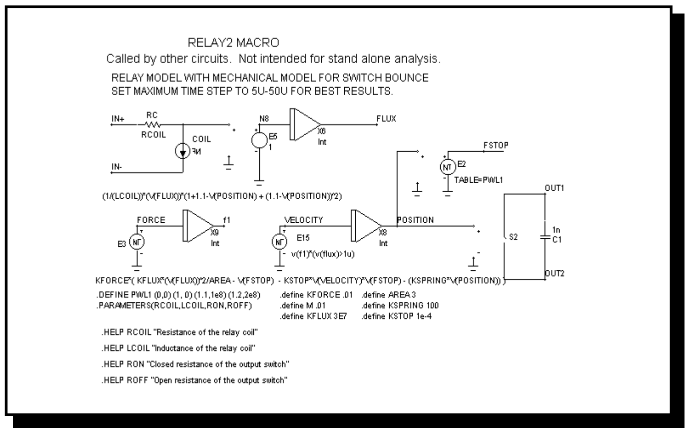


Figure 21-18 RELAY2 macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
RCOIL	Resistance of the relay coil
LCOIL	Inductance of the relay coil
RON	Closed resistance of the output switch
ROFF	Open resistance of the output switch

See the circuit RELAY for an example of the use of this macro.

RESONANT

This macro implements a resonant circuit with a resistor, capacitor, and inductor. The L value is entered as an input parameter, LIN. The C value is computed from the LIN and F0 input parameters. The resistor is computed from all three input parameters. The implementation is as follows:

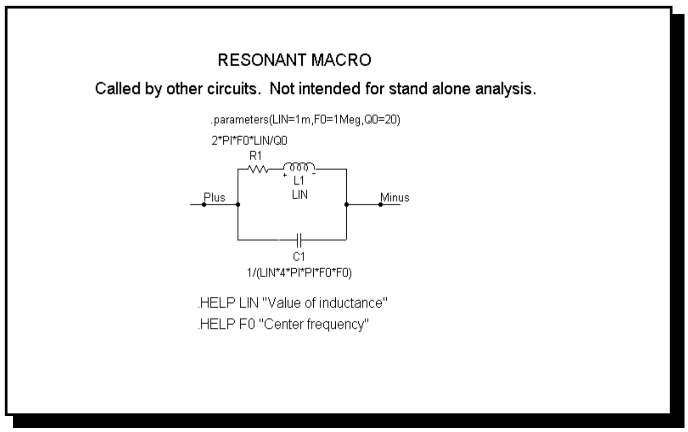


Figure 21-19 RESONANT macro equivalent circuit

The parameter definitions are as follows:

Parameter	Definition
LIN	Value of inductance
F0	Center frequency
Q0	Quality factor

SCHMITT

This circuit is a macro model for a Schmitt trigger, a circuit with a large number of uses, including noise filtering, hysteresis, and level shifting. The circuit looks like this:

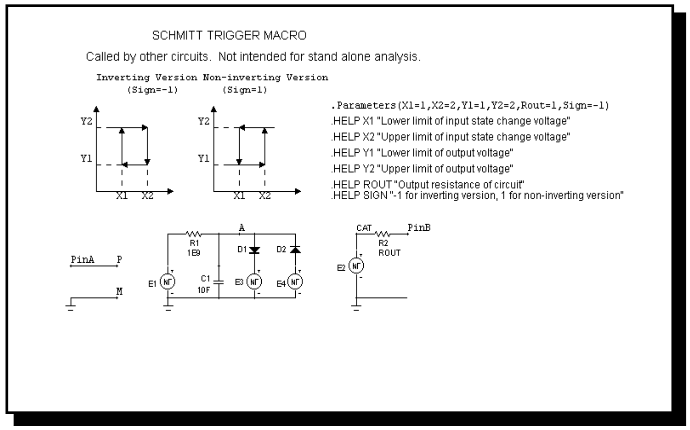


Figure 21-20 SCHMITT macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
X1	Lower limit of input state change voltage
X2	Upper limit of input state change voltage
Y1	Lower limit of output voltage
Y2	Upper limit of output voltage
ROUT	Output Resistance of circuit
SIGN	-1 for inverting version, 1 for non-inverting version

See the circuit OSC1 for an example of the use of this macro.

SCR

This circuit is a macro model for a silicon controlled rectifier or SCR. The macro equivalent circuit is as follows:

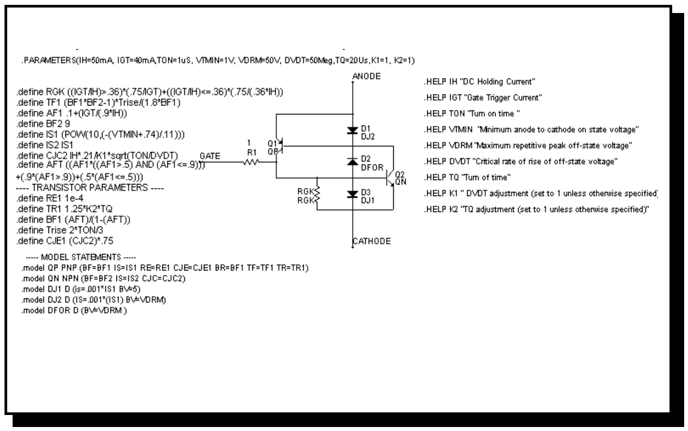


Figure 21-21 SCR macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
IH	DC holding current
IGT	Gate trigger current
TON	Turn-on time
VTMIN	Minimum anode to cathode on-state voltage
VDRM	Maximum repetitive peak off-state voltage
DVDT	Critical rate of rise of off-state voltage
TQ	Turn-off time
K1	Tweak factor for DVDT
K2	Tweak factor for TQ

See the circuit THY1 for an example of the use of this macro.

SLIP

The SLIP macro models hysteresis, or backlash. The output is zero within the slip zone, -DX to +DX. Outside of the hysteresis zone, the output is proportional to the input. The output is clipped to MAX.

This function is implemented with the SLIP macro:

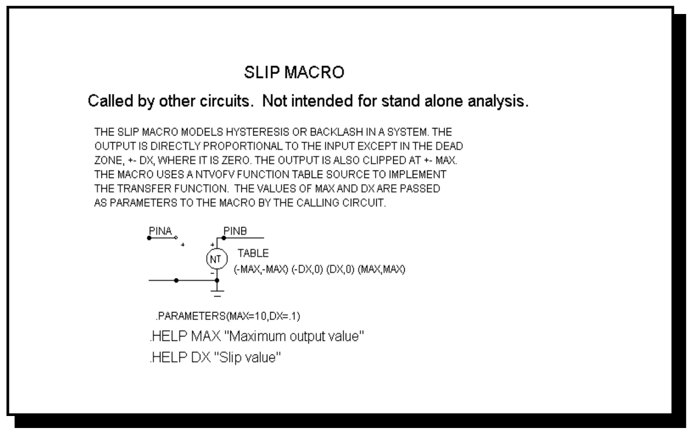


Figure 21-22 SLIP macro equivalent circuit

A pair of input parameters, DX and MAX, define the slip zone and the maximum output level. The function is constructed with an NTVOFV Function table source.

Parameter	**Definition**
DX	Slip value
MAX	Maximum value

See the circuit SYSTEM2 for an example of the use of this macro.

SPARKGAP

This is a macro circuit model for a spark gap arrestor.

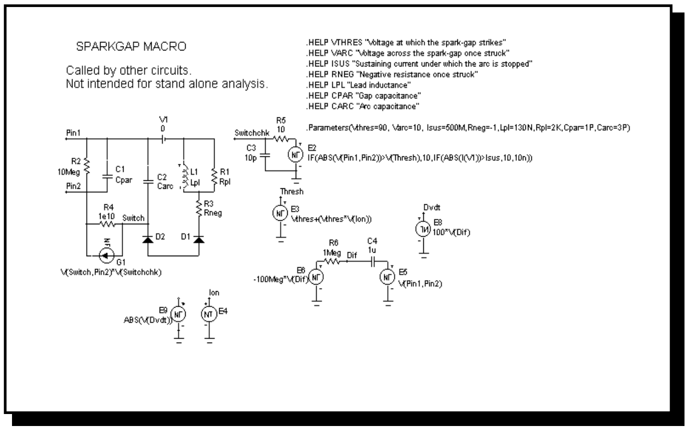


Figure 21-23 SPARKGAP macro equivalent circuit

Parameter	**Definition**
VTHRES	Voltage at which the spark-gap strikes
VARC	Voltage across the spark-gap once struck
ISUS	Sustaining current under which the arc is stopped
RNEG	Negative resistance once struck
LPL	Lead inductance
RPL	Flux loss associated with LPL
CPAR	Gap capacitance
CARC	Arc capacitance

See the circuit SPARK for an example of the use of this macro.

SUB

A common requirement is to subtract two analog signals. The desired function is:

$$V_{Out}(t) = ka\ V_a(t) - kb\ V_b(t)$$

This function is implemented with the SUB macro:

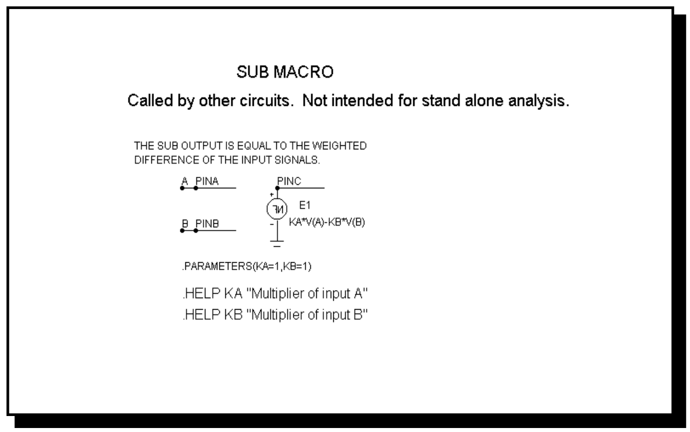


Figure 21-24 SUB macro equivalent circuit

The two input parameters, KA and KB, scale each input. The scaled input signals are then subtracted to produce the output. This implementation is done with an NFV function source.

Parameter	**Definition**
KA	Multiplier of input A
KB	Multiplier of input B

See the circuit SYSTEM2 for an example of the use of this macro.

SUM

Many systems simulations call for a function to perform the analog addition of two signals. The desired function is:

$$V_{Out}(t) = ka\ V_a(t) + \ kb\ V_b(t)$$

This function is implemented with the SUM macro:

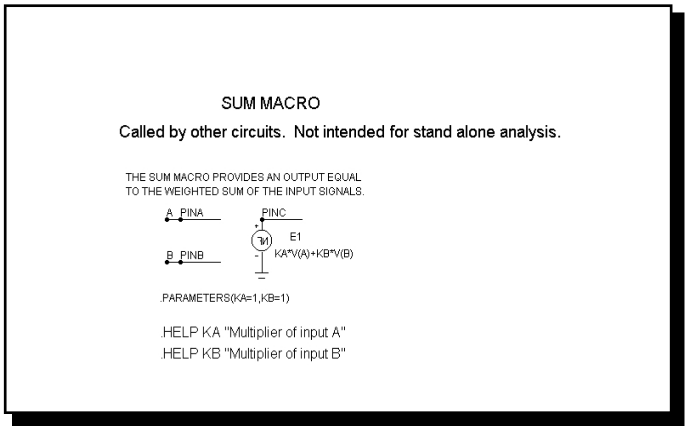


Figure 21-25 SUM macro equivalent circuit

The two input parameters, KA and KB, scale each input. The scaled input signals are then added to produce the output.

Parameter	**Definition**
KA	Multiplier of input A
KB	Multiplier of input B

See the circuit SYSTEM2 for an example of the use of this macro.

SUM3

A triple summer is sometimes convenient. The desired function is:

$$V_{Out}(t) = ka\ V_a(t) + \ kb\ V_b(t) + \ kc\ V_c(t)$$

This function is implemented with the SUM3 macro:

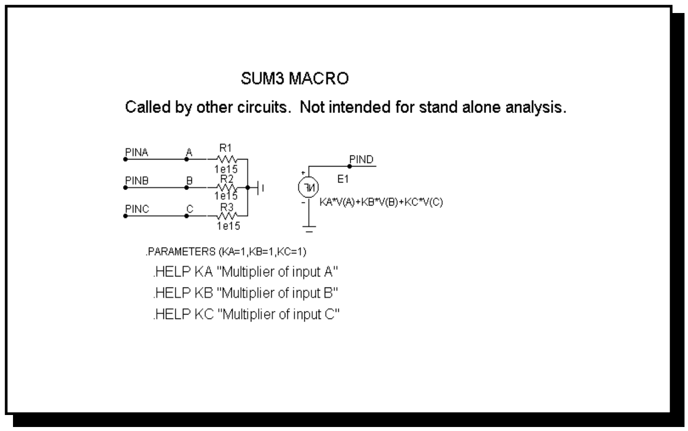


Figure 21-26 SUM3 macro equivalent circuit

The three input parameters, KA, KB, and KC, multiply each input. The three scaled input signals are then added to produce the output. This implementation is done with an NFV function source.

Parameter	**Definition**
KA	Multiplier of input A
KB	Multiplier of input B
KC	Multiplier of input C

See the circuit SYSTEM1 for an example of the use of the SUM3 macro.

TRIAC

This circuit is a macro model for a TRIAC. The macro equivalent circuit is as follows:

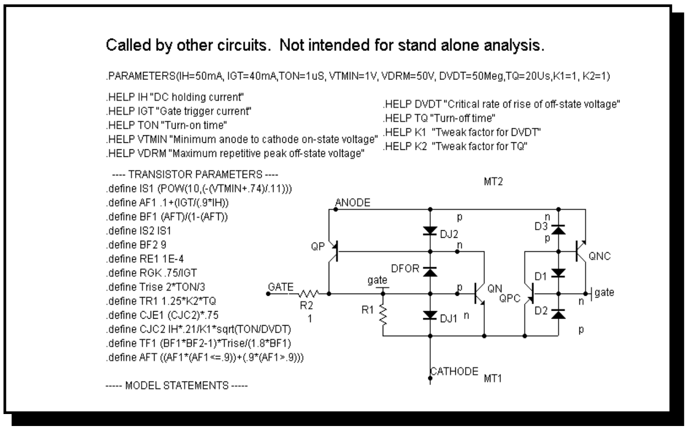


Figure 21-27 TRIAC macro equivalent circuit

The parameter definitions are as follows:

Parameter	**Definition**
IH	DC holding current
IGT	Gate trigger current
TON	Turn-on time
VTMIN	Minimum anode to cathode on-state voltage
VDRM	Maximum repetitive peak off-state voltage
DVDT	Critical rate of rise of off-state voltage
TQ	Turn-off time
K1	Tweak factor for DVDT
K2	Tweak factor for TQ

See the circuit THY1 for an example of the use of the TRIAC macro.

TRIGGER

This circuit is a macro model for a thyristor gate trigger circuit. The macro equivalent circuit is as follows:

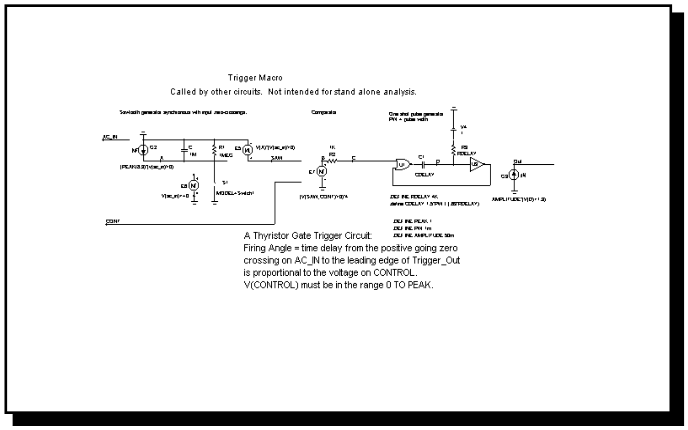


Figure 21-28 TRIGGER macro equivalent circuit

There are no parameters for the macro.

See the circuit CONVERTER3 for an example of the use of this macro.

TRIODE

The TRIODE is a macro model of a vacuum triode device. Its equivalent circuit is as shown below.

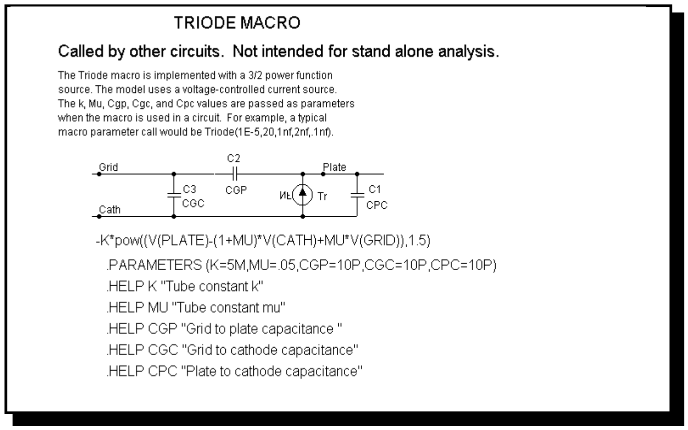


Figure 21-29 TRIODE macro equivalent circuit

The Triode macro is implemented with a 3/2 power function voltage-controlled current source. The K, MU, CGP, CGC, and CPC values are passed as parameters when the macro is used in a circuit.

Parameter	Definition
K	Tube constant k
MU	Tube constant mu
CGP	Grid to plate capacitance
CGC	Grid to cathode capacitance
CPC	Plate to cathode capacitance

See the circuit F4 for an example of the use of the TRIODE macro.

VCO

A voltage-controlled oscillator, or VCO, is an oscillator whose instantaneous frequency is dependent upon a time-varying voltage. The VCO macro has a voltage whose time dependence is given by:

$$V_{Out}(t) \;= vp\,\cos\,(2\pi(\,f0\;t\;+kf\int_0 V_{In}(t)\;dt))$$

In this form, f_0 is the center frequency and k_f is the frequency sensitivity in Hz/volt. This form of linear VCO is easily implemented as a macro:

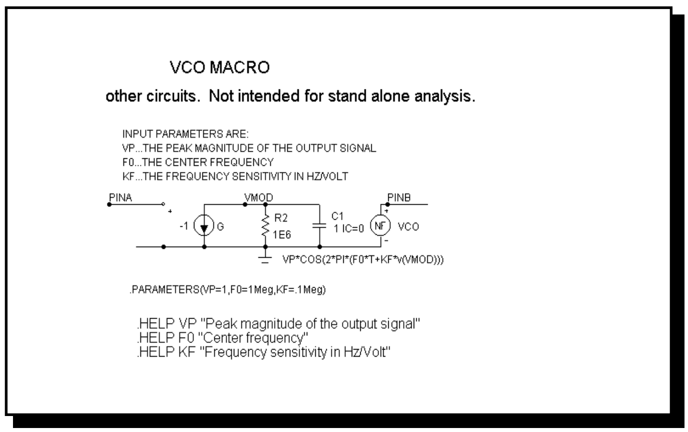


Figure 21-30 VCO macro equivalent circuit

The VCO uses a nonlinear function source, which uses the output of an integrator stage to control the frequency. The input parameters specify the magnitude, center frequency, and the frequency sensitivity.

Parameter	Definition
VP	Peak magnitude of the output signal
F0	Center frequency
KF	Frequency sensitivity in Hz/Volt

See the circuit F1 for an example of the use of this macro.

WIDEBAND

This is a wideband model of a transformer.

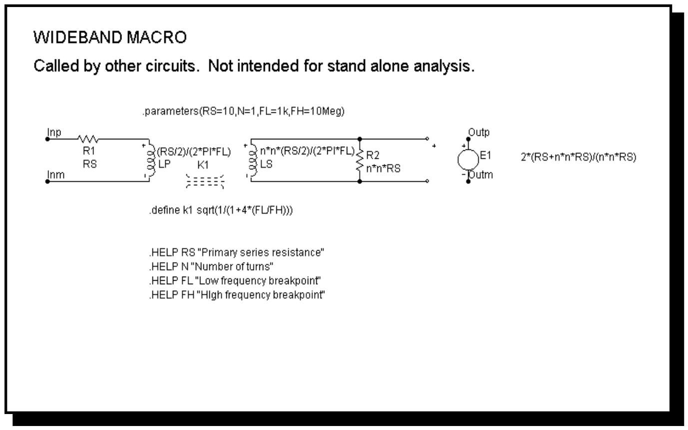


Figure 21-31 WIDEBAND macro equivalent circuit

Parameter	**Definition**
RS	Primary series resistance
N	Number of turns
FL	Low frequency breakpoint
FH	High frequency breakpoint

XTAL

XTAL is a macro model of a crystal. Its equivalent circuit is as shown below.

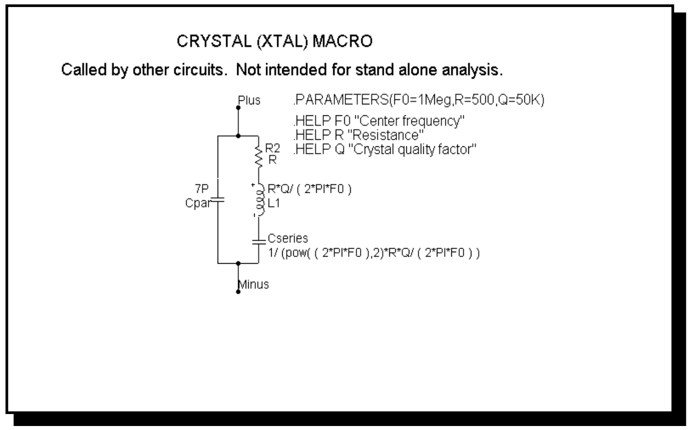


Figure 21-32 XTAL macro equivalent circuit

The XTAL macro is implemented with a standard tank circuit model for crystals.

Parameter	Definition
F0	Center frequency
R	Resistance
Q	Crystal quality factor

For examples of how to use the macro, see the circuit XTAL1, which shows a crystal oscillator application.

555

The 555 is a model of the ubiquitous 555 timer circuit. Its macro and equivalent circuit are as follows:

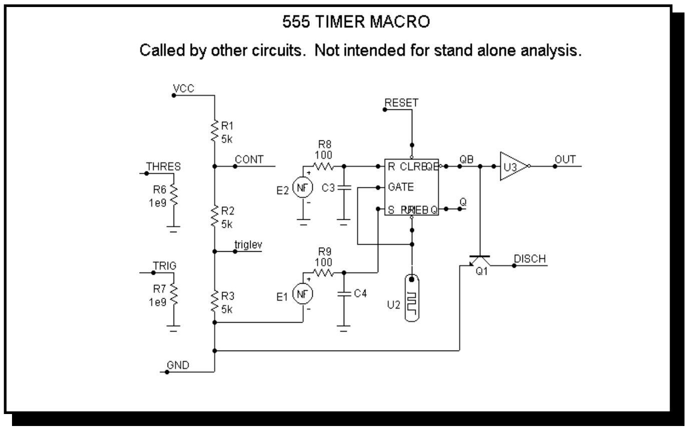


Figure 21-33 555 macro equivalent circuit

The 555 uses several nonlinear function sources to monitor the THRES and TRIG input voltage values. When they cross a certain threshold the sources switch to a high or low level and charge capacitors which drive the R and S inputs of a RS flip-flop. The flip-flop drives the output and an NPN which provides a discharge path. There are no input parameters.

For examples of how to use the macro, see the circuits 555MONO, which shows a monostable application, and 555ASTAB, which demonstrates how to use the 555 in an astable application.

You can change the power supply by using a .PARAM statement as follows:

```
.PARAM V555_VDD=<VDD value desired>
.PARAM V555_VSS=<VSS value desired>
```

The statement may be placed in the text area, the grid text, the User Definitions, or an appropriate text library file.

Chapter 22 Analog Devices

What's in this chapter

This chapter covers the parameter syntax, model statements, model parameters, and model equations used by each of the Micro-Cap analog devices.

Model statements describe the model parameters for the more complex devices.

Model parameters are the numeric values to be used in the model equations. They are obtained from *model statements* or binary model libraries (*.LBR).

Model equations use the numeric model parameter values in a set of mathematical equations that describe three aspects of a device's electrical behavior:

1. The static relationship between terminal currents and branch voltages.
2. Energy storage within the device.
3. Noise generation.

In the chapter that follows, each component is described in terms of:

1. The SPICE parameter and / or schematic attribute format.
2. The Model statement format (if any).
3. The Model parameters (if any).
4. The electrical model in terms of its schematic and model equations.

If the SPICE parameter format is not given, the component is available for use only in schematic circuits and not in SPICE text file circuits.

All devices have PACKAGE, COST, and POWER attributes. The PACKAGE attribute specifies the package to be used for PCB netlists. The optional COST and POWER attributes specify the cost and power contributions for the Bill of Materials report.

Features new in Micro-Cap 7

- S_PORT device
- COST, and POWER attributes.
- NLEV and GDSNOI noise parameters for MOSFETs.

References

This chapter provides a comprehensive guide to the device models used in MC7. It does not, however, teach the principles of analog or digital simulation, or the operation of semiconductor devices. The following references provide a good introduction to these and related topics:

Circuit design:

1. Paul R. Gray, and Robert G. Meyer
   Analysis and Design of Analog Integrated Circuits
   Second Edition.
   John Wiley and Sons, 1977, 1984

2. David A. Hodges, and Horace G. Jackson.
   Analysis and Design of Digital Integrated Circuits
   McGraw-Hill, 1983

3. Richard S. Muller and Theodore I. Kamins
   Device Electronics for Integrated Circuits
   Second Edition
   John Wiley and Sons, 1977, 1986

4. Adel Sedra, Kenneth Smith
   Microelectronic Circuits
   Fourth Edition
   Oxford, 1998

Analog Simulation:

5. A. Ruehli, Editor
   Circuit Analysis, Simulation and Design
   Advances in CAD for LSI, Part 3, Vol 1
   North-Holland 1986

6. J. Vlach, K. Singhal
   Computer Methods for Circuit Analysis and Design
   Van Nostrand Reinhold 1994

7. Kenneth Kundert
Designers Guide to SPICE & Spectre
Kluwer Academic Publishers, 1995

8. William J. McCalla
Fundamentals of Computer-Aided Circuit Simulation
Kluwer Academic Publishers, 1988

9. Andrei Vladimiresescu and Sally Liu
The Simulation of MOS Integrated circuits using SPICE2
University of California, Berkeley
Memorandum No. UCB / ERL M80/7

10. Lawrence Nagel.
SPICE2: A Computer Program to Simulate Semiconductor Circuits
University of California, Berkeley,
Memorandum No. ERL - M520

11. Andrei Vladimirescu
The SPICE Book
John Wiley & Sons, Inc., First Edition, 1994. ISBN# 0-471-60926-9

Device modeling:

12. H. Statz, P. Newman, I. W. Smith, R. A. Pucel, and H. A. Haus
GaAsFET Device and Circuit Simulation in SPICE
IEEE Transactions on Electron Devices, ED - 34, 160-169 (1987)

13. Graeme R. Boyle, Barry M. Cohn, Donald O. Petersen, and James E. Solomon
Macromodeling of Integrated Circuit Operational Amplifiers
IEEE Journal of Solid-State Circuits, Vol. SCV-9 No. 6 (1974)

14. W.R Curtice
A MESFET model for use in the design of GaAs integrated circuits
IEEE Transactions on Microwave Theory and Techniques
MTT - 28,448-456 (1980)

15. Ian Getreu
Modeling the Bipolar Transistor
Tektronix, 1979

16. *MOSPOWER Applications*
copyright 1984, Signetics, Inc.

17. Bing Jay Sheu
MOS Transistor Modeling and characterization for Circuit Simulation
University of California, Berkeley Memo No. UCB / ERL M85/85

18. Yannis P. Tsividis
Operation and Modeling of the MOS Transistor
McGraw-Hill, 1987

19. Paolo Antognetti, and Giuseppe Massobrio
Semiconductor Device Modeling with SPICE.
McGraw-Hill, 1988

20. D. C. Jiles, and D. L. Atherton
Theory of Ferromagnetic Hysteresis
Journal of Magnetism and Magnetic Materials, 61, 48 (1986)

21. Sally Liu
A Unified CAD Model for MOSFETs
University of California, Berkeley
Memorandum No. UCB / ERL M81/31

22. Y. Cheng, M. Chan, K. Hui, M. Jeng, Z. Liu, J. Huang, K. Chen, J. Chen, R. Tu, P. Ko, C. Hu
BSIM3v3.1 Manual
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

23. Daniel Foty
MOSFET Modeling with SPICE Principles and Practice
Prentice Hall, First Edition, 1997. ISBN# 0-13-227935-5

Filters:

24. Lawrence P. Huelsman
Active and Passive Filter Design, An Introduction
McGraw-Hill, 1993

25. Kendall L. Su
Analog Filters
Chapman & Hall, 1996

26. Arthur B. Williams
Electronic Filter Design Handbook
McGraw-Hill, 1981

Switched-Mode Power Supply Circuits:

27. Christophe Basso
Switch-Mode Power Supply SPICE Simulation Cookbook
McGraw-Hill, 2001

28. Steven M. Sandler
SMPS Simulation with SPICE 3
McGraw Hill, First Edition, 1997. ISBN# 0-07-913227-8

SPICE Modeling:

29. Ron Kielkowski
SPICE - Practical Device Modeling
McGraw Hill 1995. ISBN# 0-07-911524-1

30. Ron Kielkowski
Inside SPICE - Overcoming the Obstacles of Circuit Simulation
McGraw Hill 1993. ISBN# 0-07-911525-X

31. Connelly and Choi
Macromodeling with SPICE
Prentice Hall 1992. ISBN# 0-13-544941-3

RF Circuits:

27. Vendelin, Pavio, and Rhoda
Microwave Circuit Design
Wiley-Interscience, 1990

Battery

Schematic format

PART attribute
<*name*>

Example
V1

VALUE attribute
<*value*>

Examples
10
5.5V

The battery produces a constant DC voltage. It is implemented internally as the simplest form of SPICE independent voltage source. Its main virtue lies in its simplicity.

The battery provides simple constant voltage values. If you need a voltage source that is dependent on other circuit variables or time, use one of the dependent sources or the function source (NFV).

Bipolar transistor

SPICE format

Syntax
Q<*name*> <*collector*> <*base*> <*emitter*> [*substrate*]
+<*model name*> [*area*] [OFF] [IC=<*vbe*>[,*vce*]]

Examples
Q1 5 7 9 2N3904 1 OFF IC=0.65,0.35
Q2 5 7 9 20 2N3904 2.0
Q3 C 20 OUT [SUBS] 2N3904

Schematic format

PART attribute
<*name*>

Examples
Q1
BB1

VALUE attribute
[*area*] [OFF] [IC=<*vbe*>[,*vce*]]

Example
1.5 OFF IC=0.65,0.35

MODEL attribute
<*model name*>

Example
2N2222A

The initialization, 'IC=<*vbe*>[,*vce*]', assigns initial voltages to the junctions in transient analysis if no operating point is done (or if the UIC flag is set). Area multiplies or divides parameters as shown in the model parameters table. The OFF keyword forces the BJT off for the first iteration of the operating point.

Model statement forms

.MODEL <*model name*> NPN ([*model parameters*])
.MODEL <*model name*> PNP ([*model parameters*])
.MODEL <*model name*> LPNP ([*model parameters*])

Examples
.MODEL Q1 NPN (IS=1E-15 BF=55 TR=.5N)
.MODEL Q2 PNP (BF=245 VAF=50 IS=1E-16)
.MODEL Q3 LPNP (BF=5 IS=1E-17)

Name	**Parameter**	**Unit/s**	**Default**	**Area**
IS	Saturation current	A	1E-16	*
BF	Ideal maximum forward beta		100.0	
NF	Forward current emission coefficient		1.00	
VAF	Forward Early voltage	V	∞	
IKF	BF high-current roll-off corner	A	∞	*
ISE	BE leakage saturation current	A	0.00	*
NE	BE leakage emission coefficient		1.50	
BR	Ideal maximum reverse beta		1.00	
NR	Reverse current emission coefficient		1.00	
VAR	Reverse Early voltage	V	∞	
IKR	BR high-current roll-off corner	A	∞	*
ISC	BC leakage saturation current	A	0.00	*
NC	BC leakage emission coefficient		2.00	
NK	High current rolloff coefficient		0.50	
ISS	Substrate pn saturation current	A	0.00	*
NS	Substrate pn emission coefficient		1.00	
RC	Collector resistance	Ω	0.00	/
RE	Emitter resistance	Ω	0.00	/
RB	Zero-bias base resistance	Ω	0.00	/
IRB	Current where RB falls by half	A	∞	*
RBM	Minimum RB at high currents	Ω	RB	/
TF	Ideal forward transit time	S	0.00	
TR	Ideal reverse transit time	S	0.00	
XCJC	Fraction of BC dep. cap. to internal base		1.00	
MJC	BC junction grading coefficient		0.33	
VJC	BC junction built-in potential	V	0.75	
CJC	BC zero-bias depletion capacitance	F	0.00	*
MJE	BE junction grading coefficient		0.33	
VJE	BE junction built-in potential	V	0.75	
CJE	BE zero-bias depletion capacitance	F	0.00	*
MJS	CS junction grading coefficient		0.00	
VJS	CS junction built-in potential	V	0.75	
CJS	CS junction zero-bias capacitance	F	0.00	*
VTF	Transit time dependence on VBC	V	∞	
ITF	Transit time dependence on IC	A	0.00	*
XTF	Transit time bias dependence coefficient		0.00	
PTF	Excess phase		0.00	
XTB	Temperature coefficient for betas		0.00	

Name	Parameter	Units	Default	Area
EG	Energy gap	eV	1.11	
XTI	Saturation current temperature exponent		3.00	
KF	Flicker-noise coefficient	0.00		
AF	Flicker-noise exponent		1.00	
FC	Forward-bias depletion coefficient		0.50	
T_MEASURED	Measured temperature	°C		
T_ABS	Absolute temperature	°C		
T_REL_GLOBAL	Relative to current temperature	°C		
T_REL_LOCAL	Relative to AKO temperature	°C		
TRE1	RE linear temperature coefficient	$°C^{-1}$	0.00	
TRE2	RE quadratic temperature coefficient	$°C^{-2}$	0.00	
TRB1	RB linear temperature coefficient	$°C^{-1}$	0.00	
TRB2	RB quadratic temperature coefficient	$°C^{-2}$	0.00	
TRM1	RBM linear temperature coefficient	$°C^{-1}$	0.00	
TRM2	RBM quadratic temperature coefficient	$°C^{-2}$	0.00	
TRC1	RC linear temperature coefficient	$°C^{-1}$	0.00	
TRC2	RC quadratic temperature coefficient	$°C^{-2}$	0.00	

BJT model equations

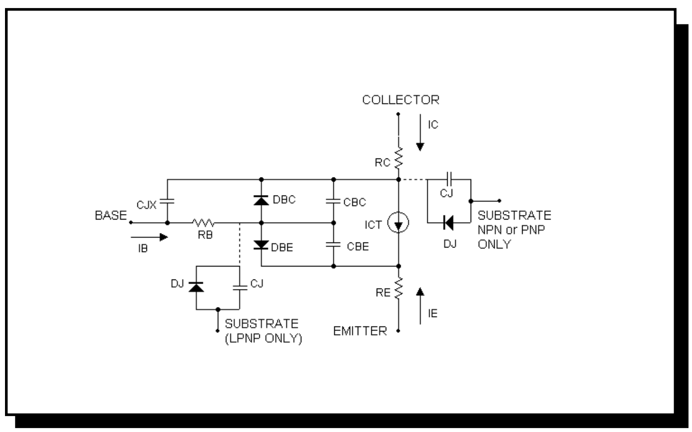


Figure 22-1 Bipolar transistor model

Definitions

The model parameters IS, IKF, ISE, IKR, ISC, ISS, IRB, CJC, CJE, CJS, and ITF are multiplied by [*area*] and the model parameters RC, RE, RB, and RBM are divided by [*area*] prior to their use in the equations below.

T is the device operating temperature and Tnom is the temperature at which the model parameters are measured. Both are expressed in degrees Kelvin. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may both be customized for each model by specifying the parameters T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. For more details on how device operating temperatures and Tnom temperatures are calculated, see the .MODEL section of Chapter 20, "Command Statements".

The substrate node is optional and if not specified, is connected to ground. If the substrate node is specified and is an alphanumeric name, it must be enclosed in square brackets.

The model types NPN and PNP are used for vertical transistor structures and the LPNP is used for lateral PNP structures. The isolation diode DJ and capacitance CJ are connected from the substrate node to the internal collector node for NPN and PNP model types, and from the substrate node to the internal base node for the LPNP model type.

When adding new four terminal BJT components to the Component library, use NPN4 or PNP4 for the Definition field.

When a PNP4 component is placed in a schematic, the circuit is issued an LPNP model statement. If you want a vertical PNP4, change the LPNP to PNP.

VT = k·T/q
VBE = Internal base to emitter voltage
VBC = Internal base to collector voltage
VCS = Internal collector to substrate voltage
In general, X(T) = Temperature adjusted value of parameter X

Temperature effects

$EG(T) = 1.16 - .000702 \cdot T^2/(T+1108)$

$IS(T) = IS \cdot e^{((T/Tnom-1) \cdot EG/(VT))} \cdot (T/Tnom)^{(XTI)}$

$ISE(T) = (ISE/(T/Tnom)^{XTB}) \cdot e^{((T/Tnom-1) \cdot EG/(NE \cdot VT))} \cdot (T/Tnom)^{(XTI/NE)}$

$ISC(T) = (ISC/(T/Tnom)^{XTB}) \cdot e^{((T/Tnom-1) \cdot EG/(NC \cdot VT))} \cdot (T/Tnom)^{(XTI/NC)}$

$BF(T) = BF \cdot (T/Tnom)^{XTB}$

$BR(T) = BR \cdot (T/Tnom)^{XTB}$

$VJE(T) = VJE \cdot (T/Tnom)-3 \cdot VT \cdot \ln((T/Tnom))-EG(Tnom) \cdot (T/Tnom)+EG(T)$

$VJC(T) = VJC \cdot (T/Tnom)-3 \cdot VT \cdot \ln((T/Tnom))-EG(Tnom) \cdot (T/Tnom)+EG(T)$

$VJS(T) = VJS \cdot (T/Tnom)-3 \cdot VT \cdot \ln((T/Tnom))-EG(Tnom) \cdot (T/Tnom)+EG(T)$

$CJE(T) = CJE \cdot (1+MJE \cdot (.0004 \cdot (T-Tnom) + (1 - VJE(T)/VJE)))$

$CJC(T) = CJC \cdot (1+MJC \cdot (.0004 \cdot (T-Tnom) + (1 -VJC(T)/VJC)))$

$CJS(T) = CJS\cdot(1+MJS\cdot(.0004\cdot(T-Tnom) + (1 - VJS(T)/VJS)))$

Current equations

$Q1 = 1/ (1 - VBC/VAF - VBE/VAR)$

$Q2 = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/IKF + IS(T)\cdot(e^{(VBC/(NR\cdot VT))}-1)/IKR$

$QB = Q1\cdot(1+(1+4\cdot Q2)^{0.5}) / 2$

Current source value

$ICT = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/QB - IS(T)\cdot(e^{(VBC/(NR\cdot VT))}-1)/QB$

Base emitter diode current

$IBE = ISE(T)\cdot(e^{(VBE/(NE\cdot VT))}-1)+IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/BF(T)$

Base collector diode current

$IBC = ISC(T)\cdot(e^{(VBC/(NC\cdot VT))}-1)+IS(T)\cdot(e^{(VBC/(NR\cdot VT))}-1)/QB/BR(T)$

Base terminal current

$IB = IBE + IBC$

$IB = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/BF(T)+ISE(T)\cdot(e^{(VBE/(NE\cdot VT))}-1)+$
$\quad IS(T)\cdot(e^{(VBC/(NR\cdot VT))}-1)/BR(T)+ISC(T)\cdot(e^{(VBC/(NC\cdot VT))}-1)$

Collector terminal current

$IC = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}- e^{(VBC/(NR\cdot VT))})/QB$
$\quad - IS(T)\cdot(e^{(VBC/(NR\cdot VT))}-1)/BR(T)- ISC(T)\cdot(e^{(VBC/(NC\cdot VT))} -1)$

Emitter terminal current

$IE = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-e^{(VBC/(NR\cdot VT))})/QB$
$\quad +IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/BF(T)+ISE(T)\cdot(e^{(VBE/(NE\cdot VT))}-1)$

Capacitance equations

Base emitter capacitance

GBE = base emitter conductance = ð(IBE) / ð(VBE)

If $VBE \leq FC\cdot VJE(T)$

$\quad CBE1 =CJE(T)\cdot(1 - VBE/VJE(T))^{-MJE}$

Else

$\quad CBE1 = CJE(T)\cdot(1 - FC)^{-(1+MJE)} \cdot (1 - FC\cdot(1+MJE)+MJE\cdot VBE/VJE(T))$

$R = IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)/(IS(T)\cdot(e^{(VBE/(NF\cdot VT))}-1)+ITF)$

$CBE2 = GBE\cdot TF\cdot(1+XTF\cdot(3\cdot R^2-2\cdot R^3)\cdot e^{(VBC/(1.44\cdot VTF))})$

$CBE = CBE1+CBE2$

Base collector capacitances
GBC = base collector conductance = $\partial(IBC) / \partial(VBC)$

If $VBC \leq FC \cdot VJC(T)$

$C = CJC(T) \cdot (1 - VBC/VJC(T))^{-MJC}$

Else

$C = CJC(T) \cdot (1 - FC)^{-(1+MJC)} \cdot (1 - FC \cdot (1+MJC)+MJC \cdot VBC/VJC(T))$

$CJX = C \cdot (1- XCJC)$
$CBC = GBC \cdot TR + XCJC \cdot C$

Collector substrate capacitance
If $VCS \leq 0$

$CJ = CJS(T) \cdot (1 - VCS/VJS(T))^{-MJS}$

Else

$CJ = CJS(T) \cdot (1 - FC)^{-(1+MJS)} \cdot (1 - FC \cdot (1+MJS)+MJS \cdot VCS/VJS(T))$

Noise

RE, RB, and RC generate thermal noise currents.

$Ie^2 = (4 \cdot k \cdot T) / RE$
$Ib^2 = (4 \cdot k \cdot T) / RB$
$Ic^2 = (4 \cdot k \cdot T) / RC$

Both the collector and base currents generate frequency-dependent flicker and shot noise currents.

$Ic^2 = 2 \cdot q \cdot Ic + KF \cdot ICB^{AF} / Frequency$
$Ib^2 = 2 \cdot q \cdot Ib + KF \cdot IBE^{AF} / Frequency$

where

KF is the flicker noise coefficient
AF is the flicker noise exponent

Capacitor

SPICE format

Syntax

C<*name*> <*plus*> <*minus*> [*model name*] <value> [IC=<*initial voltage*>]

Examples
C1 2 3 1U
C2 7 8 110P IC=2

<*plus*> and <*minus*> are the positive and negative node numbers. The polarity references are used to apply the initial condition.

Schematic format

PART attribute
<*name*>

Examples
C5
XC16

VALUE attribute
<*value*> [IC=<*initial voltage*>]

Examples
1U
110P IC=3
50U*(1+V(6)/100)

FREQ attribute
[*fexpr*]

Examples
1.2+10m*log(F)

MODEL attribute
[*model name*]

Examples
CMOD
MICA

VALUE attribute

<*value*> may be a simple number or an expression involving time-domain variables. It is evaluated in the time domain only. Consider the following expression:

1n+V(10)*2n

V(10) refers to the value of the voltage on node 10 during a transient analysis, a DC operating point calculation prior to an AC analysis, or during a DC analysis. It does not mean the AC small signal voltage on node 10. If the DC operating point value for node 10 was 2, the capacitance would be evaluated as 1n + 2*2n = 5n. The constant value, 5n, would be used in AC analysis.

FREQ attribute

If <*fexpr*> is used, it replaces the value determined during the operating point. <*fexpr*> may be a simple number or an expression involving frequency domain variables. The expression is evaluated during AC analysis as the frequency changes. For example, suppose the <*fexpr*> attribute is this:

1n + 1E-9*V(1,2)*(1+10m*log(f))

In this expression F refers to the AC analysis frequency variable and V(1,2) refers to the AC small signal voltage between nodes 1 and 2. Note that there is no time-domain equivalent to <*fexpr*>. Even if <*fexpr*> is present, <*value*> will be used in transient analysis.

Initial conditions

The initial condition assigns an initial voltage across the capacitor.

Stepping effects

The VALUE attribute and all of the model parameters may be stepped. If VALUE is stepped, it replaces <*value*>, even if it is an expression. The stepped value may be further modified by the quadratic and temperature effects.

Quadratic effects

If [*model name*] is used, <*value*> is multiplied by a factor, QF, a quadratic function of the time-domain voltage, V, across the capacitor.

$$QF = 1 + VC1 \cdot V + VC2 \cdot V^2$$

This is intended to provide a subset of the old SPICE 2G POLY keyword, which is no longer supported.

Temperature effects
The temperature factor is computed as follows:
If [*model name*] is used, <*value*> is multiplied by a temperature factor, TF.

$$TF = 1+TC1\cdot(T-Tnom)+TC2\cdot(T-Tnom)^2$$

TC1 is the linear temperature coefficient and is sometimes given in data sheets as parts per million per degree C. To convert ppm specs to TC1 divide by 1E6. For example, a spec of 1500 ppm/degree C would produce a TC1 value of 1.5E-3.

T is the device operating temperature and Tnom is the temperature at which the nominal capacitance was measured. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may be changed for each model by specifying values for T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL section of Chapter 20, "Command Statements", for more information on how device operating temperatures and Tnom temperatures are calculated.

Monte Carlo effects
LOT and DEV Monte Carlo tolerances, available only when [*model name*] is used, are obtained from the model statement. They are expressed as either a percentage or as an absolute value and are available for all of the model parameters except the T_parameters. Both forms are converted to an equivalent *tolerance percentage* and produce their effect by increasing or decreasing the Monte Carlo factor, MF, which ultimately multiplies the final value.

$$MF = 1 \pm \textit{tolerance percentage} /100$$

If *tolerance percentage* is zero or Monte Carlo is not in use, then the MF factor is set to 1.0 and has no effect on the final value.

The final capacitance used in the analysis, cvalue, is calculated as follows:

$$\textit{cvalue} = \textit{value} * C * QF * TF * MF$$

Model statement form
.MODEL <*model name*> CAP ([*model parameters*])

Examples
.MODEL CMOD CAP (C=2.0 LOT=10% VC1=2E-3 VC2=.0015)
.MODEL CEL CAP (C=1.0 LOT=5% DEV=.5% T_ABS=37)

Model parameters

Name	Parameter	Units	Default
C	Capacitance multiplier		1
VC1	Linear voltage coefficient	V^{-1}	0
VC2	Quadratic voltage coefficient	V^{-2}	0
TC1	Linear temperature coefficient	$°C^{-1}$	0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0
T_MEASURED	Measured temperature	°C	
T_ABS	Absolute temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO temperature	°C	

Noise effects

There are no noise effects included in the capacitor model.

Dependent sources (linear)

Schematic format

PART attribute
<*name*>

Example
DEP1

VALUE attribute
<*value*>

Example
10

These ideal, linear, two-port functions transform the input voltage or current to an output voltage or current. Input voltage is the voltage between the plus and minus input leads. Positive input current is defined as *into* the plus input lead.

Model equations

Part	**Equation**
IOFI	Iout(Iin) = *value*· Iin
IOFV	Iout(Vin) = *value*· Vin
VOFV	Vout(Vin) = *value*· Vin
VOFI	Vout(Iin) = *value*· Iin

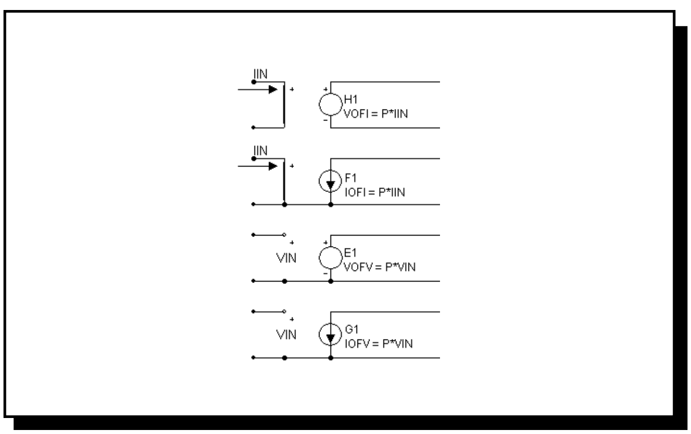


Figure 22-2 Dependent sources-linear

Dependent sources (SPICE E, F, G, H devices)

Standard SPICE formats:

Syntax of the voltage-controlled voltage source
E<*name*> <*plusout*> <*minusout*> [POLY(<*k*>)] *n1p n1m*
+ [*n2p n2m...nkp nkm*] *p0* [*p1...pk*] [IC=*c1*[,*c2*[,*c3*...[,ck]]]]

Syntax of the current-controlled current source
F<*name*> <*plusout*> <*minusout*> [POLY(<*k*>)] *v1* [*v2...vk*]
+ *p0* [*p1...pk*] [IC=*c1*[,*c2*[,*c3*...[,ck]]]]

Syntax of the voltage-controlled current source
G<*name*> <*plusout*> <*minusout*> [POLY(<*k*>)]
+*n1p n1m* [*n2p n2m...nkp nkm*] *p0* [*p1...pk*]
[IC=*c1*[,*c2*[,*c3*…[,*ck*]]]]

Syntax of the current-controlled voltage source
H<*name*> <*plusout*> <*minusout*> [POLY(<*k*>)] *v1* [*v2...vk*]
+ *p0* [*p1...pk*] [IC=*c1*[,*c2*[,*c3*...[,*ck*]]]]

Standard PSpice supported formats:

Extended syntax of the voltage-controlled voltage source
[E | G]<*name*> <*plusout*> <*minusout*> VALUE = {<*expression*>}

[E | G]<*name*> <*plusout*> <*minusout*> TABLE{<*expression*>} =
+ <<*input value*>,<*output value*>>*

[E | G]<*name*> <*plusout*> <*minusout*> LAPLACE {<*expression*>} =
+ {<*Laplace transfer function*>}

[E | G]<*name*> <*plusout*> <*minusout*> FREQ
+ {<*expression*>} = [KEYWORD]
+<<*frequency value*>,<*magnitude value*>,<*phase value*>>*

n1p is the first positive controlling node.
n1m is the first negative controlling node.
nkp is the k'th positive controlling node.
nkm is the k'th negative controlling node.
p0 is the first polynomial coefficient.
pk is the k'th polynomial coefficient.
v1 is the voltage source whose current is the first controlling variable.

vk is the voltage source whose current is the k'th controlling variable.
c1 is the 1'st initial condition.
ck is the k'th initial condition.

SPICE Examples
E2 7 4 POLY(2) 10 15 20 25 1.0 2.0 10.0 20.0
G2 7 4 POLY(3) 10 15 20 25 30 35 1.0 2.0 3.0 10.0 20.0 30.0
F2 7 4 POLY(2) V1 V2 1.0 2.0 10.0 20.0
H2 7 4 POLY(3) V1 V2 V3 1.0 2.0 3.0 10.0 20.0 30.0
E1 10 20 FREQ {V(1,2)} = (0,0,0) (1K,0,0) (10K,0.001,0)
L1 10 20 TABLE{V(5,6)*V(3)} = {(0,0,0) (1K,0,0) (2K,-20,0)}
L2 10 20 LAPLACE {V(5,6)} = {1/(1+.001*S+1E-8*S*S)}

Schematic format
The schematic attributes are similar to the standard SPICE format without the <*plusout*> and <*minusout*> node numbers. The TABLE, VALUE, LAPLACE, and FREQ features are not supported in the schematic versions of the E, F, G, and H devices. These features are supported in the Function and Laplace devices, described later in this chapter.

PART attribute
<*name*>

Example
G1

VALUE attribute
[POLY(<*k*>)] *n1p n1m* [*n2p n2m...nkp nkm*] *p0* [*p1...pk*]
+ [IC=*c1*[,*c2*[,*c3*...[,*ck*]]]]

[POLY(<*k*>)] *n1p n1m* [*n2p n2m...nkp nkm*] *p0* [*p1...pk*]
+ [IC=*c1*[,*c2*[,*c3*...[,*ck*]]]]

[POLY(<*k*>)] *v1* [*v2...vk*] *p0* [*p1...pk*] [IC=*c1*[,*c2*[,*c3*...[,*ck*]]]]

[POLY(<*k*>)] *v1* [*v2...vk*] *p0* [*p1...pk*] [IC=*c1*[,*c2*[,*c3*...[,*ck*]]]]

Examples
POLY(2) 10 15 20 25 1.0 2.0 10.0 20.0
POLY(3) 10 15 20 25 30 35 1.0 2.0 3.0 10.0 20.0 30.0
POLY(2) V1 V2 1.0 2.0 10.0 20.0
POLY(3) V1 V2 V3 1.0 2.0 3.0 10.0 20.0 30.0

Model equations

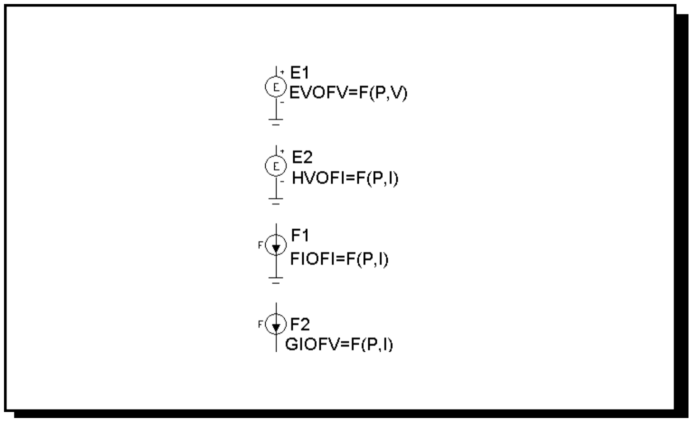


Figure 22-3 Dependent sources-SPICE poly

When the POLY keyword is not used, the general equation for the dependent source function is:

$$F = p_0 + p_1 \cdot V_1 + p_2 \cdot V_1^2 + p_3 \cdot V_1^3 + \ldots\ p_k \cdot V_1^k$$

F is the output value of the dependent voltage or current source.
V_1 is the k independent variable value.
$p_0, p_1, p_2 \ldots p_k$ are the k +1 polynomial coefficients.

When the POLY keyword is used, the polynomial is computed as follows

$$F = \sum_{j=0}^{n} p_j \prod_{k=1}^{n} V_k^{E^k}$$

The values of the exponents E_1, E_2, E_3, ..., E_n are chosen by a procedure which is best understood by referring to Table 22-1. For a description of how the LAPLACE and FREQ versions work, see the Laplace Sources section of this chapter and the intoduction to Chapter 21, "Analog Behavioral Modeling."

	Number of input variables (Order)									
	1	2		3			4			
Coefficient	E1	E1	E2	E1	E2	E3	E1	E2	E3	E4
P0	0	0	0	0	0	0	0	0	0	0
P1	1	1	0	1	0	0	1	0	0	0
P2	2	0	1	0	1	0	0	1	0	0
P3	3	2	0	0	0	1	0	0	1	0
P4	4	1	1	2	0	0	0	0	0	1
P5	5	0	2	1	1	0	2	0	0	0
P6	6	3	0	1	0	1	1	1	0	0
P7	7	2	1	0	2	0	1	0	1	0
P8	8	1	2	0	1	1	1	0	0	1
P9	9	0	3	0	0	2	0	2	0	0
P10	10	4	0	3	0	0	0	1	1	0
P11	11	3	1	2	1	0	0	1	0	1
P12	12	2	2	2	0	1	0	0	2	0
P13	13	1	3	1	2	0	0	0	1	1
P14	14	0	4	1	1	1	0	0	0	2
P15	15	5	0	1	0	2	3	0	0	0
P16	16	4	1	0	3	0	2	1	0	0
P17	17	3	2	0	2	1	2	0	1	0
P18	18	2	3	0	1	2	2	0	0	1
P19	19	1	4	0	0	3	1	2	0	0

Table 22-1 Polynomial exponents

The lightly shaded parts of the table mark the coefficients used for summing the input variables. The heavily shaded portions of the table mark the coefficients used for forming a product of the input variables. Other combinations of polynomial products are shown in the rest of the table.

For example, to create a voltage source whose value equals the sum of three input voltages, use this:

```
E1 4 0 POLY(3) 1 0 2 0 3 0 0 1 1 1
```

This creates a voltage source whose output is a third-order polynomial function of three sets of differential voltages.

Input variable 1 = V(1) - V(0)= V(1)= voltage on node 1
Input variable 2 = V(2) - V(0)= V(2)= voltage on node 2
Input variable 3 = V(3) - V(0)= V(3)= voltage on node 3

The exponents E_1=1, E_2=1, and E_3=1 are chosen from the p_1, p_2, and p_3 rows, respectively, of the 3'rd order column. The output of this source is:

$$V = p_0 \cdot V_1^0 \cdot V_2^0 \cdot V_3^0 + p_1 \cdot V_1^1 \cdot V_2^0 \cdot V_3^0 + p_2 \cdot V_1^0 \cdot V_2^1 \cdot V_3^0 + p_3 \cdot V_1^0 \cdot V_2^0 \cdot V_3^1$$
$$V = p_0 + p_1 \cdot V_1 + p_2 \cdot V_2 + p_3 \cdot V_3$$
$$V = 0 + 1 \cdot V_1 + 1 \cdot V_2 + 1 \cdot V_3$$
$$V = V_1 + V_2 + V_3$$

To create a current source whose value equals the product of the current flowing in two sources, use this:

```
F1 3 0 POLY(2) V1 V2 0 0 0 0 1
```

This creates a current source whose output is a second order polynomial function of the current flowing in the sources V_1 and V_2.

Input variable 1 = I1= current flowing through V_1
Input variable 2 = I2 = current flowing through V_2

The exponents E_1=1 and E_2=1 are chosen from the p_4 row and second order column. The output of this source is:

$$I = p_0 \cdot I_1^0 \cdot I_2^0 + p_1 \cdot I_1^1 \cdot I_2^0 + p_2 \cdot I_1^0 \cdot I_2^1 + p_3 \cdot I_1^2 \cdot I_2^0 + p_4 \cdot I_1^1 \cdot I_2^1$$
$$I = 0 \cdot I_1^0 \cdot I_2^0 + 0 \cdot I_1^1 \cdot I_2^0 + 0 \cdot I_1^0 \cdot I_2^1 + 0 \cdot I_1^2 \cdot I_2^0 + 1 \cdot I_1^1 \cdot I_2^1$$
$$I = I_1 \cdot I_2$$

Diode

SPICE format

Syntax
D<*name*> <*anode*> <*cathode*> <*model name*> [*area*] [OFF]
+ [IC=<*vd*>]

Example
D1 7 8 1N914 1.0 OFF IC=.001

Schematic format

PART attribute
<*name*>

Example
D1

VALUE attribute
[*area*] [OFF] [IC=<*vd*>]

Example
10.0 OFF IC=0.65

MODEL attribute
<*model name*>

Example
1N914

Both formats

[*area*] multiplies or divides model parameters as shown in the model parameters table. The presence of the OFF keyword forces the diode off during the first iteration of the DC operating point. The initial condition, [IC=<*vd*>], assigns an initial voltage to the junction in transient analysis if no operating point is done (or if the UIC flag is set).

Model statement form

.MODEL <*model name*> D ([*model parameters*])

Example
.MODEL 1N4434 D (IS=1E-16 RS=0.55 TT=5N)

Diode model parameters

The diode model is the standard PSpice™ diode model with an additional linear parallel resistance added to account for leakage effects.

Name	Parameter	Units	Def	Area
Level	Model level (1=SPICE2G, 2=PSpice)		1.0	
IS	Saturation current	A	1E-14	*
N	Emission coefficient		1.00	
ISR	Recombination current param.	A	0.00	*
NR	Emission coefficient for ISR		2.00	
IKF	High-injection knee current	A	∞	*
BV	Reverse breakdown knee voltage V	∞		
IBV	Reverse breakdown knee current A	1E-10	*	
NBV	Reverse breakdown ideality		1	
IBVL	Low-level reverse breakdown current	A	0	*
NBVL	Low-level reverse breakdown ideality		1	
RS	Parasitic series resistance	Ω	0	/
TT	Transit time	S	0.00	
CJO	Zero-bias junction cap.	F	0.00	*
VJ	Built-in potential	V	1.00	
M	Grading coefficient		0.50	
FC	Forward-bias depletion coefficient		0.50	
EG	Energy gap	eV	1.11	
XTI	Temperature exponent for IS		3.00	
TIKF	IKF temperature coefficient(linear)	°C^{-1}	0.00	
TBV1	BV temperature coefficient(linear)	°C^{-1}	0.00	
TBV2	BV temperature coefficient(quadratic)	°C^{-2}	0.00	
TRS1	RS temperature coefficient(linear)	°C^{-1}	0.00	
TRS2	RS temperature coefficient(quadratic)	°C^{-2}	0.00	
KF	Flicker noise coefficient		0.00	
AF	Flicker noise exponent		1.00	
RL	Leakage resistance	Ω	∞	
T_MEASURED	Measured temperature	°C		
T_ABS	Absolute temperature	°C		
T_REL_GLOBAL	Relative to current temperature	°C		
T_REL_LOCAL	Relative to AKO temperature	°C		

Model equations

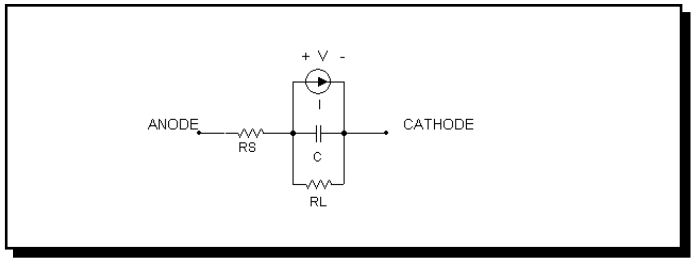


Figure 22-4 Diode model

Notes and Definitions

The model parameters IS, ISR, IKF, IBV, IBVL, and CJO are multiplied by [*area*] and the model parameter RS is divided by [*area*] prior to their use in the diode model equations below.

T is the device operating temperature and Tnom is the temperature at which the model parameters are measured. Both are expressed in degrees Kelvin. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may both be customized for each model by specifying the parameters T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL section of Chapter 20, "Command Statements", for more information on how device operating temperatures and Tnom temperatures are calculated.

Temperature Effects

$VT=k \cdot T / q = 1.38E\text{-}23 \cdot T / 1.602E\text{-}19$

$IS(T) = IS \cdot e^{((T/Tnom - 1) \cdot EG/(VT \cdot N))} \cdot (T/Tnom)^{(XTI/N)}$

$ISR(T) = ISR \cdot e^{((T/Tnom - 1) \cdot EG/(VT \cdot NR))} \cdot (T/Tnom)^{(XTI/NR)}$

$IKF(T) = IKF \cdot (1+TIKF \cdot (T - Tnom))$

$BV(T) = BV \cdot (1+TBV1 \cdot (T-Tnom)+TBV2 \cdot (T-Tnom)^2)$

$RS(T) =RS \cdot (1+TRS1 \cdot (T-Tnom)+TRS2 \cdot (T-Tnom)^2)$

$VJ(T) = VJ \cdot (T/Tnom)- 3 \cdot VT \cdot \ln(T/Tnom)- EG(Tnom) \cdot (T/Tnom)+EG(T)$

where $EG(T) = 1.17-.000702 \cdot T^2/(T+1108)$

$CJO(T) = CJO \cdot (1+M \cdot (.0004 \cdot (T-Tnom) + (1 - VJ(T)/VJ)))$

Current source equations

$I = Ifwd - Irev$

$Inrm = IS(T) \cdot (e^{(V/(VT \cdot N))} - 1)$

If $IKF > 0$

$\quad Kinj = (IKF /(IKF + Inrm))^{1/2}$

Else

$\quad Kinj = 1$

$Irec = ISR(T) \cdot (e^{(V/(VT \cdot NR))} - 1)$

$Kgen = ((1-V/VJ(T))^2 + 0.005)^{M/2}$

$Irev = IBV(T) \cdot (e^{-(V+BV)/(VT \cdot NBV)} - 1) + IBVL(T) \cdot (e^{-(V+BV)/(VT \cdot NBVL)} - 1)$

$Ifwd = Kinj \cdot Inrm + Kgen \cdot Irec$

Capacitance Equations

Transit Time capacitance

Gd = DC conductance of the diode

$CT = TT \cdot Gd$

If $V \leq FC \cdot VJ(T)$ then

$\quad CJ = CJO(T) \cdot (1 - V/VJ(T))^{-M}$

Else

$\quad CJ = CJO(T) \cdot (1 - FC)^{-(1+M)} \cdot (1 - FC \cdot (1+M) + M \cdot (V/VJ(T)))$

$C = CT + CJ$

Noise Equations

Flicker and shot noise is generated by the diode current, I. The resistors RS and RL generate thermal noise. The noise currents are computed as follows:

$\quad I_{RS} = (4 \cdot k \cdot T / RS)^{0.5}$

$\quad I_{RL} = (4 \cdot k \cdot T / RL)^{0.5}$

$\quad I_I = (2 \cdot q \cdot I + KF \cdot I^{AF} / Frequency)^{0.5}$

Function sources

Schematic format

PART attribute
<*name*>

Example
F1

VALUE attribute for formula (NFV and NFI) type
<*formula*>

Example of formula type
10*Sin(2*PI*1E6*T)*V(3)*I(L1)*EXP(-V(IN)/100NS)

FREQ attribute
[<*fexpr*>]

Example
1200*(1+sqrt(F/1e6))

NOISE EXPRESSION attribute for NFI only
[<*noise_expr*>]

Example
1200*(1+sqrt(F/1e6))

TABLE attribute for table (NTIOFI, NTIOFV, NTVOFV, NTVOFI) types
(<*x1,y1*>) (<*x2,y2*>) ... ([(*xk,yk*)])
Braces are required for expressions and optional for variables.

Examples of table type
(-1m,25) (1m,25) (2m,30)
({start - 1m}, {25*level}) (end,level) ({end+3m}, level2)

FREQ usage

If <*fexpr*> is used, it replaces the ordinary small-signal AC incremental value determined during the operating point. <*fexpr*> may be a simple number or an expression involving frequency domain variables. The expression is evaluated during AC analysis as the frequency changes. For example, suppose the <*fexpr*> attribute is this:

1+V(3)*(1+1e6/F)

In this expression, F refers to the AC analysis frequency variable and V(3) refers to the AC small-signal voltage from node 3 to ground. There is no time-domain equivalent to <*fexpr*>. Even if <*fexpr*> is present, only <*value*> will be used in transient analysis.

NOISE_EXPRESSION usage

If *noise_expr* is used, it generates a noise current equal to the expression. For example to simulate shot noise you might use an expression like this:

1E-16 * pow(6.5ma,1.1) / F

Note that the expression should contain only frequency (F) dependent variables. The feature is available only in the NFI source.

Function sources provide the principle time domain Analog Behavioral Modeling capability. The two basic types are distinguished by the way the value of the output variable (current or voltage) is calculated.

Formula type

The Formula type, which is similar to the SPICE3 B device, uses an algebraic formula, or expression, to compute the value of the output variable as a function of any set of valid time-domain variables. There are two versions of this source:

NFI	Function current source
NFV	Function voltage source

Here is an example of an expression that models a vacuum triode:

K* pow((V(Plate)-V(Cathode)+Mu*(V(Grid)-V(Cathode))),1.5)

Table type

The Table type, which is similar to the SPICE3 A device, uses a table of ordered data pairs which describe the output variable as a function solely of the input variable. The table describes a time-domain transfer function.

The input variable for a Table source may be either:

Current flowing into the positive input lead.
Voltage between the positive input lead and the negative input lead.

There are four basic types of Table source:

Source type	Input	Output	Definition
Current-controlled current source	I	I	NTIOFI
Current-controlled voltage source	I	V	NTVOFI
Voltage-controlled voltage source	V	V	NTVOFV
Voltage-controlled current source	V	I	NTIOFV

There are two rules for constructing the data pairs in the TABLE attribute.

1. The *x,y* pairs are separated by commas, pairs are enclosed in parentheses and are separated by spaces. The *x,y* values may be replaced by expressions containing constants or symbolic variables created with a .define statement. Expressions are evaluated only once, in the setup phase of the analysis, so they must not contain variables that vary during an analysis run, like V(1) or T, or even simulation control variables like tmin that are unknown when the expressions are evaluated.

2. Data pairs must be arranged in input ascending order.

$$x1 < x2 < \ldots xk$$

Output is calculated from the input value as follows:

1. The output value is constant at *y1* for input values below *x1*.

2. The output value is constant at *yk* for input values above *xk.*

3. Output values are interpolated for input values between table values.

For example:
(-.010,-10) (.010,10)

For an NTVOFV source, this describes an ideal amplifier having a gain of 1000 and with the output clipped to +-10 volts. The output value when the input is greater than .010 is limited to +10.0. Similarly, the output value when the input is less than -.010 is limited to -10.0.

See the sample circuit T1 for an example of table sources, and the sample circuits F1, F2, F3, and F4 for examples of formula sources.

GaAsFET

SPICE format

Syntax
B<*name*> <*drain*> <*gate*> <*source*> <*model name*>
+ [*area*] [OFF] [IC=<*vds*>[,*vgs*]]

Example
B1 5 7 9 2N3531 1 OFF IC=1.0,2.5

Schematic format

PART attribute
<*name*>

Example
B1

VALUE attribute
[*area*] [OFF] [IC=*vds*[,*vgs*]]

Example
1.5 OFF IC=0.05,1.00

MODEL attribute
<*model name*>

Example
GFX_01

The device is an n-channel device. There is no p-channel version. Level 1 specifies the Curtice model, level 2 specifies the Raytheon or Statz model, and level 3 specifies the Triquint model. The [OFF] keyword forces the device off for the first iteration of the operating point. The initial condition, [IC=*vds*[,*vgs*]], assigns initial voltages to the drain-source and gate-source terms. Additional information on the model can be found in references (14) and (15).

Model statement form

.MODEL <*model name*> GASFET ([*model parameters*])

Example
.MODEL B1 GASFET (VTO=-2 ALPHA=2 BETA=1E-4 LAMBDA=1E-3)

Model Parameters

Name	Parameter	Units	Def.	Level	Area
LEVEL	Model level (1, 2, or 3)		1	ALL	
VTO	Pinch-off voltage	V	-2.50	ALL	
ALPHA	Saturation voltage parameter	V^{-1}	2.00	ALL	
BETA	Transconductance coefficient	A/V^2	0.10	ALL	*
B	Doping tail extender	V^{-1}	0.30	2	
LAMBDA	Channel-length modulation	V^{-1}	0.00	ALL	
GAMMA	Static feedback parameter		0.00	3	
DELTA	Output feedback parameter	$(A\text{-}V)^{-1}$	0.00	3	
Q	Power law parameter		2.00	3	
RG	Gate ohmic resistance	Ω	0.00	ALL	/
RD	Drain ohmic resistance	Ω	0.00	ALL	/
RS	Source ohmic resistance	Ω	0.00	ALL	/
IS	Gate pn saturation current	A	1E-14	ALL	
N	Gate pn emission coefficient		1.00	ALL	
M	Gate pn grading coefficient		0.50	ALL	
VBI	Gate pn potential	V	1.00	ALL	
CGD	Zero-bias gate-drain pn cap.	F	0.00	ALL	*
CGS	Zero-bias gate-source pn cap.	F	0.00	ALL	*
CDS	Fixed drain-source cap.	F	0.00	ALL	*
FC	Forward-bias depletion coeff.		0.50	ALL	
VDELTA	Capacitance transition volt.	V	0.20	2,3	
VMAX	Capacitance limiting voltage	V	0.50	2,3	
EG	Bandgap voltage	eV	1.11	ALL	
XTI	IS temperature coefficient		0.00	ALL	
VTOTC	VTO temperature coefficient	V/°C	0.00	ALL	
BETATCE	BETA exp. temperature coeff.	%/°C	0.00	ALL	
TRG1	RG temperature coefficient	$°C^{-1}$	0.00	ALL	
TRD1	RD temperature coefficient	$°C^{-1}$	0.00	ALL	
TRS1	RS temperature coefficient	$°C^{-1}$	0.00	ALL	
KF	Flicker-noise coefficient		0.00	ALL	
AF	Flicker-noise exponent		1.00	ALL	
T_MEASURED	Measured temperature	°C		ALL	
T_ABS	Absolute temperature	°C		ALL	
T_REL_GLOBAL	Relative to current temp.	°C		ALL	
T_REL_LOCAL	Relative to AKO temperature	°C		ALL	

GaAsFET model equations

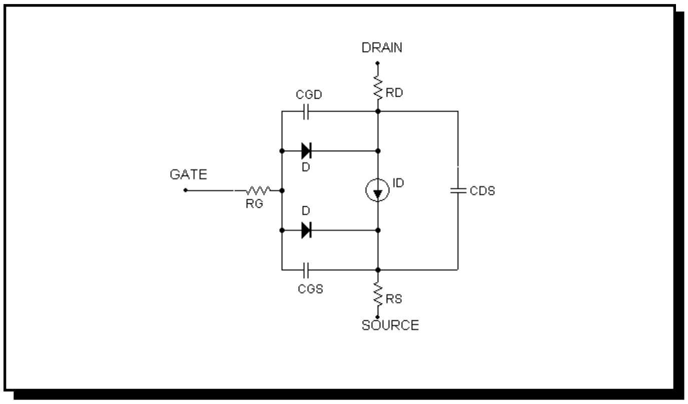


Figure 22-5 The GaAsFET model

Notes and Definitions

The model parameters BETA, CGS, CGD, and CDS are multiplied by [*area*] and the model parameters RG, RD, and RS are divided by [*area*] prior to their use in the equations below.

T is the device operating temperature and Tnom is the temperature at which the model parameters are measured. Both are expressed in degrees Kelvin. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may both be customized for each model by specifying the parameters T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL section of Chapter 20, "Command Statements", for more information on how device operating temperatures and Tnom temperatures are calculated.

Vgs = Internal gate to source voltage
Vds = Internal drain to source voltage
Id = Drain current
VT= k · T / q = 1.38E-23 ·T / 1.602E-19
In general, X(T) = Temperature adjusted value of parameter X

Temperature Dependence

$\text{BETA(T)} = \text{BETA}\cdot 1.01^{\text{BETATCE}\cdot(\text{T- Tnom})}$

$\text{EG(T)} = 1.16\text{-}.000702\cdot \text{T}^2/(\text{T}+1108)$

VTO(T) = VTO + VTOTC·(T - Tnom)

$$IS(T) = IS(Tnom) \cdot e^{((EG/(VT \bullet N))(T/Tnom -1))}$$
$$RG(T) = RG \cdot (1 + TRG1 \cdot (T - Tnom))$$
$$RD(T) = RD \cdot (1 + TRD1 \cdot (T - Tnom))$$
$$RS(T) = RS \cdot (1 + TRS1 \cdot (T - Tnom))$$
$$VBI(T) = VBI \cdot (T/Tnom)-3 \cdot VT \cdot \ln((T/Tnom))-EG(Tnom) \cdot (T/Tnom)+EG(T)$$
$$CGS(T) = CGS \cdot (1+M \cdot (.0004 \cdot (T-Tnom) + (1 - VBI(T)/VBI)))$$
$$CGD(T) = CGD \cdot (1+M \cdot (.0004 \cdot (T-Tnom) + (1 - VBI(T)/VBI)))$$

Current equations level 1

Cutoff Region : $Vgs \leq VTO(T)$

$Id = 0$

Linear and Saturation Region: $(Vgs - VTO(T)) > 0.0$

$$Id=BETA(T) \cdot (1+LAMBDA \cdot Vds) \cdot (Vgs- VTO(T))^2 \cdot \tanh(ALPHA \cdot Vds)$$

Current equations level 2

Cutoff Region : $Vgs \leq VTO(T)$

$Id = 0$

Linear and Saturation Region : $(Vgs - VTO(T))> 0.0$

If $0< Vds< 3/ALPHA$

$Kt = 1 - (1 - Vds \cdot ALPHA/3)^3$

Else

$Kt = 1$

$$Id=BETA(T) \cdot (1+LAMBDA \cdot Vds) \cdot (Vgs-VTO(T))^2 \cdot Kt/(1+B \cdot (Vgs-VTO(T)))$$

Current equations level 3

Cutoff Region : $Vgs \leq VTO(T)$

$Id = 0$

Linear and Saturation Region : $(Vgs - VTO(T))> 0.0$

If $0< Vds< 3/ALPHA$

$Kt = 1 - (1 - Vds \cdot ALPHA/3)^3$

Else

$Kt = 1$

$$Idso = BETA \cdot (Vgs-(VTO-GAMMA \cdot Vds)^Q \cdot Kt$$
$$Id = Idso/(1+DELTA \cdot Vds \cdot Idso)$$

Capacitance equations level 1

If $Vgs \le FC \cdot VBI(T)$

$Cgs = CGS/(1 - Vgs/VBI(T))^{M}$

Else

$Cgs = CGS \cdot (1 - FC)^{-(1-M)} \cdot (1 - FC \cdot (1+M)+M \cdot (Vgs/VBI(T)))$

If $Vds \le FC \cdot VBI(T)$

$Cgd = CGD/(1 - Vgd/VBI(T))^{M}$

Else

$Cgd = CGD \cdot (1 - FC)^{-(1-M)} \cdot (1 - FC \cdot (1+M)+M \cdot (Vgd/VBI(T)))$

Capacitance equations level 2 and level 3

$Ve = (Vgs +Vgd + ((Vgs - Vgd)^{2} + ALPHA -2)^{1/2})/2$

If $(Ve +VTO(T) +((Ve - VTO(T))^{2}+DELTA^{2})^{1/2})/2 < VMAX$

$Vn = (Ve +VTO(T) +((Ve - VTO(T))^{2}+DELTA^{2})^{1/2})/2$

Else

$Vn = VMAX$

$K1 = (1 + Ve - VTO(T))/((Ve - VTO(T))^{2}+DELTA^{2})^{1/2})/2$

$K2 = (1 + (Vgs - Vgd)/((Vgs - Vgd)^{2}+ALPHA^{-2})^{1/2})/2$

$K3 = (1 - (Vgs - Vgd)/((Vgs - Vgd)^{2}+ALPHA^{-2})^{1/2})/2$

$Cgs = CGS \cdot K2 \cdot K1/(1 - Vn/VBI(T))^{1/2} +CGD \cdot K3)$

$Cgd = CGS \cdot K3 \cdot K1/(1 - Vn/VBI(T))^{1/2} +CGD \cdot K2)$

Noise

The parasitic lead resistances, RG, RD, and RS, generate thermal noise currents.

$Ig^{2} = 4 \cdot k \cdot T / RG$

$Id^{2} = 4 \cdot k \cdot T / RD$

$Is^{2} = 4 \cdot k \cdot T / RS$

The drain current generates a noise current.

$I^{2} = 4 \cdot k \cdot T \cdot gm \cdot 2/3 + KF \cdot Id^{AF} / Frequency$

where $gm = \partial Id / \partial Vgs$ (at operating point)

Independent sources (V and I sources)

SPICE format

Syntax for the voltage source
Vname <*plus*> <*minus*> [[DC] *value*]
\+ [AC *magval* [*phaseval*]]
\+ [PULSE *v1 v2* [*td* [*tr* [*tf* [*pw* [*per*]]]]]]
OR [SIN *vo va* [*f0* [*td* [*df* [*ph*]]]]]
OR [EXP *v1 v2* [*td1* [*tc1* [*td2* [*tc2*]]]]]
OR [PWL *t1 v1 t2 v2* ...[*tn* , *vn*]]
OR [SFFM *vo va f0* [*mi* [*fm*]]]

Syntax for the current source
Iname <*plus*> <*minus*> [[DC] *value*]
\+ [AC *magval* [*phaseval*]]
\+ [PULSE *i1 i2* [*td* [*tr* [*tf* [*pw* [*per*]]]]]]
OR [SIN *io ia* [*f0* [*td* [*df* [*ph*]]]]]
OR [EXP *i1 i2* [*td1* [*tc1* [*td2* [*tc2*]]]]]
OR [PWL *t1 i1 t2 i2* ...[*tn* , *in*]]
OR [SFFM *io ia f0* [*mi* [*fm*]]]

The only difference between the voltage and current independent source is the use of V and I for the first character of the name.

Example
V1 10 20 DC 1 PULSE 0 1MA 12ns 8ns 110ns 240ns 500ns

Schematic format

These are the V and I sources from the **Analog Primitives / Waveform Sources** group of the **Component** menu.

PART attribute
<*name*>

Example
V1

VALUE attribute
<*value*> where *value* is identical to the SPICE format without the name and the plus and minus node numbers.

Example
DC 1 PULSE 0 1MA 12ns 8ns 110ns 240ns 500ns

Equations

The equations and sample waveforms that follow are for transient analysis only. AC analysis uses AC *magval* (volts) and AC *phaseval* (degrees) to set the amplitude and phase of the small signal stimulus. TSTEP is the print interval. TSTOP is the run time. These values are obtained from the Analysis Limits dialog box. For SPICE files, MC7 obtains these values from the .TRAN statement and copies them to the Analysis Limits dialog box.

EXP type

Name	Description	Units	Default
v1	Initial value	V or A	None
v2	Peak value	V or A	None
td1	Rise delay	S	0
tc1	Rise time constant	S	TSTEP
td2	Fall delay	S	*td1*+TSTEP
tc2	Fall time constant	S	TSTEP

The waveform value generated by the EXP option is as follows:

Time interval	Value
0 to *td1*	$v1$
td1 to *td2*	$v1+(v2-v1)\cdot(1-e^{-(TIME-td1)}/tc1)$
td2 to TSTOP	$v1+(v2-v1)\cdot((1-e^{-(TIME-td1)}/tc1)-(1-e^{-(TIME-td2)}/tc2))$

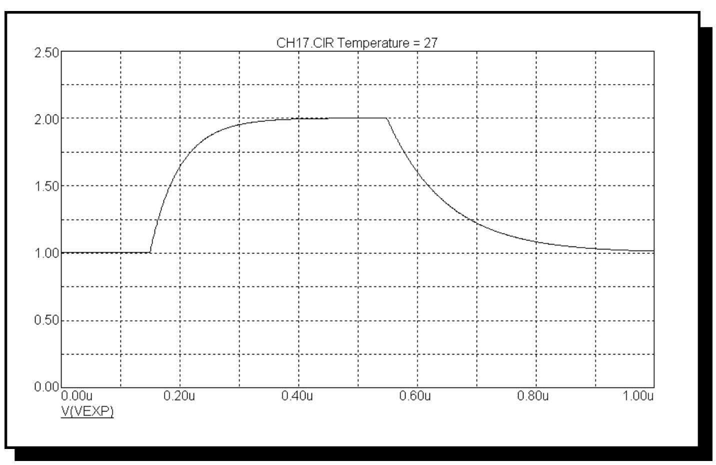


Figure 22-6 Waveform for "EXP 1 2 150n 50n 550n 100n"

PULSE type

Name	Description	Units	Default
v1	Initial value	V or A	None
v2	Pulse value	V or A	None
td	Delay	S	0
tr	Rise time	S	TSTEP
tf	Fall time	S	TSTEP
pw	Pulse width	S	TSTOP
per	Period	S	TSTOP

The waveform value generated by the PULSE option is as follows:

From	To	Value
0	*td*	*v1*
td	*td+tr*	$v1+((v2-v1)/tr)\cdot(T-td)$
td+tr	*td+tr+pw*	*v2*
td+tr+pw	*td+tr+pw+tf*	$v2+((v1-v2)/tf)\cdot(T-td-tr-pw)$
td+tr+pw+tf	*per*	*v1*

where From and To are T values, and T=TIME mod per. The waveform repeats every *per* seconds.

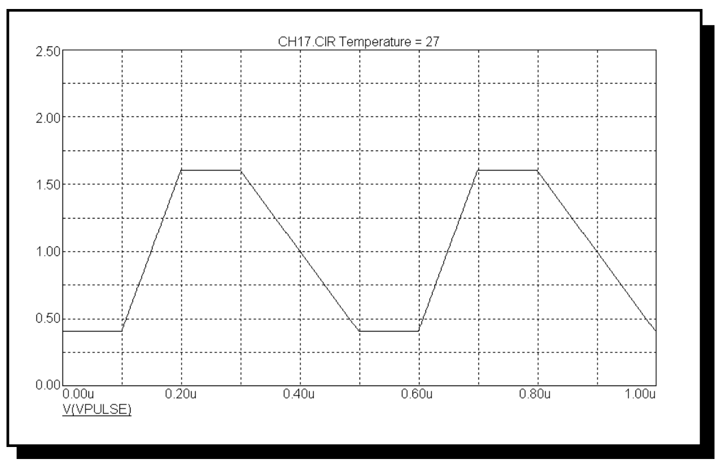


Figure 22-7 Waveform for "PULSE .4 1.6 .1u .1u .2u .1u .5u"

SFFM type

Name	Description	Units	Default
vo	Offset value	V or A	None
va	Peak amplitude	V or A	None
f0	Carrier frequency	Hz	1/TSTOP
mi	Modulation index		0
fm	Modulation freq.	Hz	1/TSTOP

The waveform value generated by the SFFM option is as follows:
$F = vo + va \cdot \sin(2 \cdot \pi \cdot f0 \cdot T + mi \cdot \sin(2 \cdot \pi \cdot fm \cdot T))$
where T = Transient analysis time

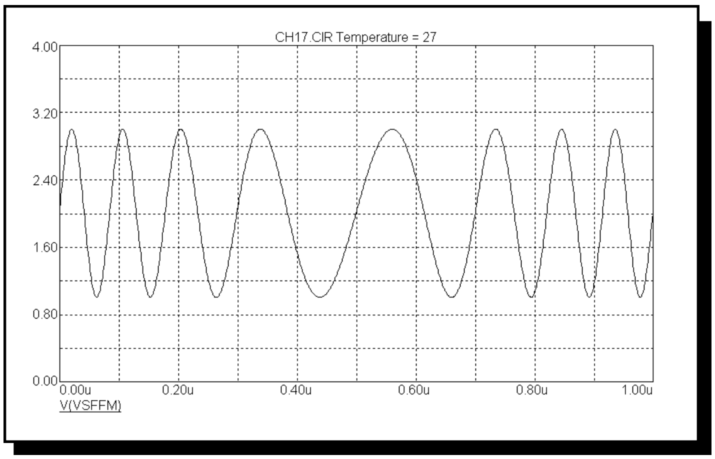


Figure 22-8 Waveform for "SFFM 2 1 8Meg 4 1Meg"

SIN type

Name	Description	Units	Default
vo	Offset value	V or A	None
va	Peak amplitude	V or A	None
f0	Frequency	Hz	1/TSTOP
td	Delay	s	0
df	Damping factor	s^{-1}	0
ph	Phase	degrees	0

The waveform value generated by the SIN option is as follows:

From	To	Value
0	*td*	*vo*
td	TSTOP	$vo+va\cdot\sin(2\cdot\pi\cdot(f0\cdot(T-td)+ph/360))\cdot e^{-(T-td)\cdot df}$ where T = Transient analysis time

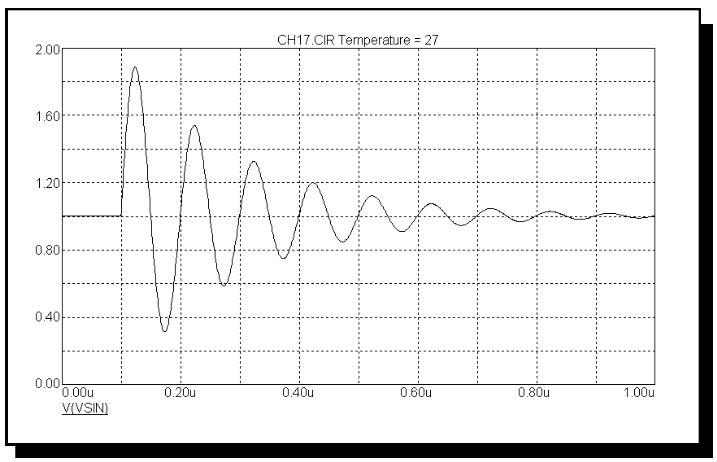


Figure 22-9 Waveform for "SIN 1 1 10Meg 100n 5E6 "

PWL type

General Form:
PWL
+ [TIME_SCALE_FACTOR=<*ts_value*>]
+ [VALUE_SCALE_FACTOR=<*vs_value*>]
+(*data_pairs*)

where the syntax of *data_pairs* is:
<*tin*>,<*in*>

ts_value, if present, multiplies all *tin* values and *vs_value*, if present, multiplies all *in* values.

Syntax for a single point on the waveform:
(<*tin*>,<*in*>)

Syntax for m points on the waveform:
(<*tin*$_1$>,<*in*$_1$>) (<*tin*$_2$>,<*in*$_2$>) ... (<*tin*$_m$>,<*in*$_m$>)

Syntax to repeat (*data_pairs*)* *n* times:
REPEAT FOR <*n*> (*data_pairs*)*
ENDREPEAT

Syntax to repeat (*data_pairs*)* forever:
REPEAT FOREVER (*data_pairs*)*
ENDREPEAT

Each data pair specifies one point on the waveform curve. Intermediate values are linearly interpolated from the table pairs.

There is no specific limit on the number of data pairs in the table. They may be added indefinitely until system memory is exhausted.

Examples:
The VALUE attribute for a 10 ns non repeating square wave:
PWL (0,0) (5n,0) (5n,5) (10n,5) (10n,0)

The VALUE attribute for another 10 ns non repeating square wave:
PWL TIME_SCALE_FACTOR=1n (0,0) (5,0) (5,5) (10,5) (10,0)

The VALUE attribute for a 10 ns non repeating square wave, with a high level of 5KV:
PWL VALUE_SCALE_FACTOR=1E3 (0,0) (5n,0) (5n,5) (10n,5) (10n,0)

The VALUE attribute for a 10 ns square wave, repeated 20 times:
PWL REPEAT FOR 20 (0,0) (5n,0) (5n,5) (10n,5) (10n,0) ENDREPEAT

The VALUE attribute for a 10 ns square wave, repeated forever:
PWL REPEAT FOREVER (0,0) (5n,0) (5n,5) (10n,5) (10n,0) ENDREPEAT

Here is an example of a PWL waveform:

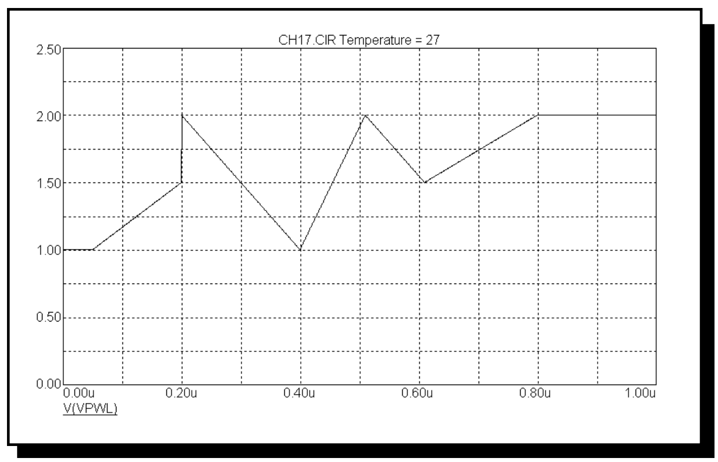


Figure 22-10 Waveform for
"PWL 0.05u,1 0.20u,1.5 0.20u,2 0.40u,1 0.51u,2 0.61u,1.5 0.80u,2"

Inductor

SPICE format

Syntax

L<*name*> <*plus*> <*minus*> [*model name*] <*value*> [*IC*=<*initial current*>]

Examples

L1 2 3 1U

L2 7 8 110P IC=2

<*plus*> and <*minus*> are the positive and negative node numbers. Positive current flows into the plus node and out of the minus node.

Schematic format

PART attribute

<*name*>

Examples

L5

L1

VALUE attribute

<*value*> [IC=<*initial current*>]

Examples

1U

110U IC=3

10U*(1+I(L10)/100)

FREQ attribute

[*fexpr*]

Examples

1.2mh+5m*(1+log(F))

MODEL attribute

[*model name*]

Examples

LM

L_MODEL

VALUE attribute

<*value*> may be a simple number or an expression involving time-domain variables. The expression is evaluated in the time domain only. Consider the expression:

100+I(L2)*2

I(L2) refers to the value of the L2 current, during a transient analysis, a DC operating point calculation prior to an AC analysis, or during a DC analysis. It does not mean the AC small signal L2 current. If the operating point value for L2 current was 2, the inductance would be evaluated as 100+2*2=104. The constant value, 104, is used in AC analysis.

FREQ attribute

If <*fexpr*> is used, it replaces the value determined during the operating point. <*fexpr*> may be a simple number or an expression involving frequency domain variables. The expression is evaluated during AC analysis as the frequency changes. For example, suppose the <*fexpr*> attribute is this:

10mh+I(L1)*(1+1E-9*f)/5m

In this expression, F refers to the AC analysis frequency variable and I(L1) refers to the AC small signal current through inductor L1. Note that there is no time-domain equivalent to <*fexpr*>. Even if <*fexpr*> is present, <*value*> will be used in transient analysis.

Initial conditions

The initial condition assigns an initial current through the inductor in transient analysis if no operating point is done (or if the UIC flag is set).

Stepping effects

Both the VALUE attribute and all of the model parameters may be stepped. If VALUE is stepped, it replaces <*value*>, even if it is an expression. The stepped value may be further modified by the quadratic and temperature effects.

Quadratic effects

If [*model name*] is used, <*value*> is multiplied by a factor, QF, which is a quadratic function of the time-domain current, I, through the inductor.

$$QF = 1 + IL1 \cdot I + IL2 \cdot I^2$$

This is intended to provide a subset of the old SPICE 2G POLY keyword, which is no longer supported.

Temperature effects
The temperature factor is computed as follows:

If [*model name*] is used, <*value*> is multiplied by a temperature factor, TF.

$$TF = 1+TC1\cdot(T-Tnom)+TC2\cdot(T-Tnom)^2$$

TC1 is the linear temperature coefficient and is sometimes given in data sheets as parts per million per degree C. To convert ppm specs to TC1 divide by 1E6. For example, a spec of 200 ppm/degree C would produce a TC1 value of 2E-4.

T is the device operating temperature and Tnom is the temperature at which the nominal inductance was measured. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may be changed for each model by specifying values for T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL section of Chapter 20, "Command Statements", for more information on how device operating temperatures and Tnom temperatures are calculated.

Monte Carlo effects
LOT and DEV Monte Carlo tolerances, available only when [*model name*] is used, are obtained from the model statement. They are expressed as either a percentage or as an absolute value and are available for all of the model parameters except the T_parameters. Both forms are converted to an equivalent *tolerance percentage* and produce their effect by increasing or decreasing the Monte Carlo factor, MF, which ultimately multiplies the final value.

MF = 1 ± *tolerance percentage* /100

If *tolerance percentage* is zero or Monte Carlo is not in use, then the MF factor is set to 1.0 and has no effect on the final value.

The final inductance, *lvalue*, is calculated as follows:

lvalue = <*value*> * L * QF * TF * MF

Model statement form

.MODEL <*model name*> IND ([*model parameters*])

Examples

.MODEL LMOD IND (L=2.0 LOT=10% IL1=2E-3 IL2=.0015)

.MODEL L_W IND (L=1.0 LOT=5% DEV=.5% T_ABS=37)

Model parameters

Name	**Parameter**	**Units**	**Default**
L	Inductance multiplier		1
IL1	Linear current coefficient	A^{-1}	0
IL2	Quadratic current coefficient	A^{-2}	0
TC1	Linear temperature coefficient	$°C^{-1}$	0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0
T_MEASURED	Measured temperature	°C	
T_ABS	Absolute temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO temperature	°C	

Noise effects

There are no noise effects included in the inductor model.

Isource

Schematic format

PART attribute
<*name*>

Examples
I1
CURRENT_SOURCE

VALUE attribute
<*value*>

Examples
1U
10

The Isource produces a constant DC current. It is implemented internally as a SPICE independent current source.

JFET

SPICE format

Syntax

J<*name*> <*drain*> <*gate*> <*source*> <*model name*>
+ [*area*] [OFF] [IC=<*vds*>[,*vgs*]]

Example

J1 5 7 9 2N3531 1 OFF IC=1.0,2.5

Schematic format

PART attribute

<*name*>

Example

J1

VALUE attribute

[*area*] [OFF] [IC=<*vds*>[,*vgs*]]

Example

1.5 OFF IC=0.05,1.00

MODEL attribute

<*model name*>

Example

JFET_MOD

The value of [*area*], whose default value is 1, multiplies or divides parameters as shown in the table. The [OFF] keyword turns the JFET off for the first operating point iteration. The initial condition, [IC= <*vds*>[,*vgs*]], assigns initial drain-source and gate-source voltages. Negative VTO implies a depletion mode device and positive VTO implies an enhancement mode device. This conforms to the SPICE 2G.6 model. Additional information on the model can be found in reference (2).

Model statement forms

.MODEL <*model name*> NJF ([*model parameters*])
.MODEL <*model name*> PJF ([*model parameters*])

Examples
.MODEL J1 NJF (VTO=-2 BETA=1E-4 LAMBDA=1E-3)
.MODEL J2 PJF (VTO= 2 BETA=.005 LAMBDA=.015)

Model Parameters

Name	**Parameter**	**Units**	**Def.**	**Area**
VTO	Threshold voltage	V	-2.00	
BETA	Transconductance parameter	A/V^2	1E-4	*
LAMBDA	Channel-length modulation	V^{-1}	0.00	
RD	Drain ohmic resistance	Ω	0.00	/
RS	Source ohmic resistance	Ω	0.00	/
CGS	Zero-bias gate-source junction cap.	F	0.00	*
CGD	Zero-bias gate-drain junction cap.	F	0.00	*
M	Gate junction grading coefficient		0.50	
PB	Gate-junction potential	V	1.00	
IS	Gate-junction saturation current	A	1E-14	*
FC	Forward-bias depletion coefficient		0.50	
VTOTC	VTO temperature coefficient	V/°C	0.00	
BETATCE	BETA exp. temperature coefficient	%/°C	0.00	
XTI	IS temperature coefficient		3.00	
KF	Flicker-noise coefficient		0.00	
AF	Flicker-noise exponent		1.00	
T_MEASURED	Measured temperature	°C		
T_ABS	Absolute temperature	°C		
T_REL_GLOBAL	Relative to current temperature	°C		
T_REL_LOCAL	Relative to AKO temperature	°C		

Model equations

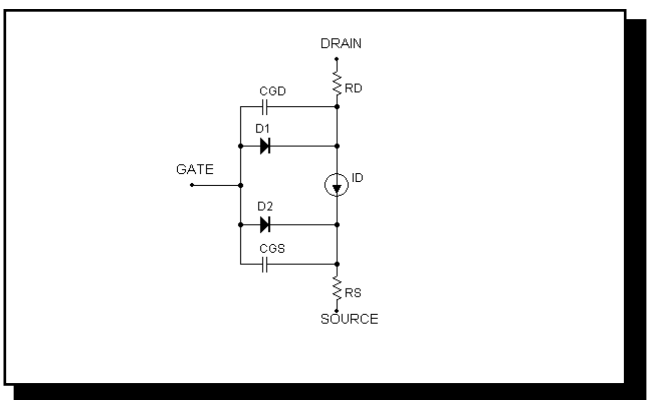


Figure 22-11 JFET model

Notes and Definitions

Parameters BETA, CGS, CGD, and IS are multiplied by [*area*] and parameters RD and RS are divided by [*area*] prior to their use in the equations below.

Vgs = Internal gate to source voltage
Vds = Internal drain to source voltage
Id = Drain current

Temperature Dependence

T is the device operating temperature and Tnom is the temperature at which the model parameters are measured. Both are expressed in degrees Kelvin. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. Both T and Tnom may be customized for each model by specifying the parameters T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL section of Chapter 20, "Command Statements", for more information on how device operating temperatures and Tnom temperatures are calculated.

$VTO(T) = VTO + VTOTC\cdot(T\text{-}Tnom)$
$BETA(T) = BETA\cdot 1.01^{BETACE\cdot(T-Tnom)}$
$IS(T) = IS\cdot e^{1.11\cdot(T/Tnom-1)/VT}\cdot(T/Tnom)^{XTI}$
$EG(T) = 1.16 - .000702\cdot T^2/(T+1108)$
$PB(T) = PB\cdot(T/Tnom)- 3\cdot VT\cdot ln((T/Tnom))-EG(Tnom)\cdot(T/Tnom)+EG(T)$
$CGS(T) = CGS\cdot(1+M\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB)))$
$CDS(T) = CDS\cdot(1+M\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB)))$

Current equations

Cutoff Region : $Vgs \le VTO(T)$

$Id = 0$

Saturation Region : $Vds > Vgs - VTO(T)$

$Id=BETA(T)\cdot(Vgs - VTO(T))^2\cdot(1+LAMBDA\cdot Vds)$

Linear Region : $Vds < Vgs - VTO(T)$

$Id=BETA(T)\cdot Vds\cdot(2\cdot(Vgs - VTO(T))- Vds)\cdot(1+LAMBDA\cdot Vds)$

Capacitance equations

If $Vgs \le FC \cdot PB(T)$ then

$$Cgs = CGS(T)/(1 - Vgs/PB(T))^{M}$$

Else

$$Cgs = CGS(T)\cdot(1 - FC\cdot(1+M)+M\cdot(Vgs/PB(T)))/ (1 - FC)^{(1-M)}$$

If $Vgd \le FC \cdot PB(T)$ then

$$Cgd = CGD(T)/(1 - Vgd/PB(T))^{M}$$

Else

$$Cgd = CGD(T)\cdot(1 - FC\cdot(1+M)+M\cdot(Vgd/PB(T)))/ (1 - FC)^{(1-M)}$$

Noise

The resistors RS and RD generate thermal noise currents.

$$Ird^2 = 4\cdot k\cdot T / RD$$
$$Irs^2 = 4\cdot k\cdot T / RS$$

The drain current generates a noise current.

$$I^2 = 4\cdot k\cdot T\cdot gm\cdot 2/3 + KF\cdot Id^{AF} / Frequency$$

where $gm = \partial Id / \partial Vgs$ (at operating point)

K (Mutual inductance / Nonlinear magnetics model)

SPICE formats

K<*name*> L<*inductor name*> <L<*inductor name*>>*
\+ <*coupling value*>

K<*name*> L<*inductor name*>* <*coupling value*>
\+ <*model name*>

Examples
K1 L1 L2 .98
K1 L1 L2 L3 L4 L5 L6 .98

Schematic format

PART attribute
<*name*>

Example
K1

INDUCTORS attribute
<*inductor name*> <*inductor name*>*

Example
L10 L20 L30

COUPLING attribute
<*coupling value*>

Example
0.95

MODEL attribute
[*model name*]

Example
K_3C8

If <*model name*> is used, there can be a single inductor name in the INDUCTORS attribute. If model name is not used, there must be at least two inductor names in the INDUCTORS attribute.

The K device specifies the linear mutual inductance between two or more inductors. You can optionally specify a nonlinear magnetic core.

Coupled linear inductors
In this mode, the K device provides a means to specify the magnetic coupling between multiple inductors. The equations that define the coupling are:

$$V_i = L_i \frac{dI_i}{dt} + M_{ij} \frac{dI_j}{dt} + M_{ik} \frac{dI_k}{dt} + \ldots$$

where I_i is the current flowing into the plus lead of the i'th inductor. For linear inductors, <*model name*> is not used.

Nonlinear magnetic core(s)
If a <*model name*> is supplied, the following things change:

1. The linear K device becomes a nonlinear magnetic core. The model for the core is a variation of the Jiles-Atherton model.

2. Inductors are interpreted as windings and each inductor <*value*> is interpreted as the number of turns for the winding. In this case, <*value*> must be a constant whole number. It may not be an expression.

3. The list of coupled inductors may contain just one inductor. Use this method to create a single magnetic core device, not coupled to another inductor.

4. A model statement is required to define the model parameters or <*model name*> must be in the model library referenced by .LIB statements.

The nonlinear magnetics model is based on the Jiles-Atherton model. This model is based upon contemporary theories of domain wall bending and translation. The anhysteretic magnetization curve is described using a mean field approach. All magnetic domains are coupled to the bulk magnetization and magnetic fields. The anhysteretic curve is regarded as the magnetization curve that would prevail if there were no domain wall pinning. Of course, such pinning does occur, mainly at defect sites. The hysteresis effect that results from this pinning is modeled as a simple frictional force, characterized by a single constant, K. The resulting state equation produces a realistic ferromagnetic model.

The Core is modeled as a state-variable nonlinear inductor. MC7 solves a differential equation for the B and H fields and derives the terminal current and voltage from these values. The B field (in Gauss) and the H field (in Oersteds) are available as output variables in transient analysis.

To place a single core in a circuit, use this procedure:

1. Place an inductor in the circuit with these attributes:

PART	L1
VALUE	1

The value 1 represents the number of turns.

2. Place a K device in the circuit with these attributes:

PART	K1
INDUCTORS	L1
COUPLING	1.0
MODEL	KCORE

Step 2 changes the inductor L1 from a standard linear model to a nonlinear core whose properties are controlled by the model statement. See the circuit file CORE for an example of a single core device and how to do BH loop plots.

To place two magnetically coupled cores in a circuit, use this procedure:

1. Place the first inductor in the circuit with these attributes:

PART	L1
VALUE	Number of primary turns

2. Place the second inductor in the circuit with these attributes:

PART	L2
VALUE	Number of secondary turns

3. Place a K device in the circuit with these attributes:

PART	K2
INDUCTORS	L1 L2
COUPLING	Coupling coefficient between L1 and L2 (0-1.0)
MODEL	KCORE

This procedure creates two coupled cores whose magnetic properties are controlled by the KCORE model statement. See the sample circuit file CORE3 for an example of multiple, coupled core devices.

Model statement form

.MODEL <*model name*> CORE ([*model parameters*])

Examples

.MODEL K1 CORE (Area=2.54 Path=.54 MS=2E5)

.MODEL K2 CORE (MS=2E5 LOT=25% GAP=.001)

Model parameters

Name	Parameter	Units	Default
Area	Mean magnetic cross-section	cm^2	1.00
Path	Mean magnetic path length	cm	1.00
Gap	Effective air gap length	cm	0.00
MS	Saturation magnetization	a/m	4E5
Alpha	Mean field parameter		2E-5
A	Shape parameter	a/m	25
C	Domain wall flexing constant		.001
K	Domain wall bending constant		25

Model Equations

Definitions

N= number of turns

Ma = Anhysteretic magnetization

H = Magnetic field intensity

HE = Effective magnetic field intensity

B = Magnetic flux density

M = Magnetization

I = Core current

V = Core voltage

H = (100·N·I-M·Gap)/Path

HE = H + ALPHA·Ma

Ma = MS · (coth(HE/A)-A/HE)

Sign = K if dH/dt > 0.0

Sign= - K if dH/dt <= 0.0

Equations

μ = dM/dH = (Ma-M) / ((Sign)·(1+C)) + (C / (1+C))·dMa/DH

B = μ_0 · (H +M)

L = $\mu \cdot 4 \cdot \pi \cdot$ 1E-9 · N^2 · AREA / PATH

V = L · dI/dt

To derive model parameters from data sheet values, use the MODEL program.

Doing it *manually* requires this procedure:

1. Many data sheets provide the value of Bsat in Gauss. To calculate the required value of MS in units of Amps/meter, multiply the Bsat value in Gauss by 79.577. This yields the required model value for MS in Amps/meter.

2. Run the sample circuit CORE.CIR and adjust the values of A, K, C, and Alpha, to fit the data sheet BH curve. The effect of increasing each parameter is as follows:

Parameter	µ	HC	BR
Alpha	+		-
A	-	+	+
K		+	+
C	+	-	-

where µ is the slope or permeability, HC is the coercive force value, and BR is the remanence.

Laplace sources

Schematic format

PART attribute
<*name*>

Examples
FIL1
LOW1

LAPLACE attribute of LFIOFI, LFIOFV, LFVOFV, LFVOFI
<*expression*>

Example
1/(1+.001*S+1E-8*S*S)

FREQ attribute of LTIOFI, LTIOFV, LTVOFV, LTVOFI
<*(f1,m1,p1) (f2,m2,p2)...(fn,mn,pn)*>

Example
(0.0,1.0,0.0)(1Meg,0.9,-10)(10Meg,0.2,-35)

KEYWORD attribute (for use with FREQ attribute)
[[DB | MAG] [DEG | RAD]] | [R_I]

Examples
DB RAD
MAG DEG
R_I

There is no SPICE version of this source. Use the Dependent source, E or G device.

The keywords DB, MAG, DEG, RAD, R_I are interpreted as follows:

DB: Magnitude value is expressed in decibels. (default)
MAG: Magnitude value is true magnitude.
DEG: Degrees value is expressed in degrees. (default)
RAD: Degrees value is expressed in radians.
R_I: The table contains real and imaginary parts.

Laplace sources are characterized by a linear transfer function. The two basic types are distinguished by the way the transfer function is calculated. The Formula type uses an algebraic expression to describe the transfer function in terms of the complex frequency variable, S. The Table type uses a table of ordered data triplets which describe the transfer function. Each data triplet comprises the frequency, magnitude, and phase of the transfer function.

In AC analysis, the value of the transfer function is computed from the algebraic expression involving S, where $S = 2 \cdot \pi \cdot \text{frequency} \cdot j$, or obtained by interpolation from the given table.

For DC analysis, the value of the transfer function is computed from the given algebraic expression involving S, where S = 0, or obtained from the table, using the lowest frequency data point supplied.

For transient analysis, it is necessary to first determine the impulse response of the function. The impulse response is obtained by performing an inverse Fourier transform on the transfer function. Then, during the transient run, the output of the source is obtained from the convolution of the waveform at the source input nodes and the impulse response waveform. This allows the source to accurately respond to any input waveform, not just simple, predefined waveforms.

The accuracy of this procedure is limited by the number of time points in the impulse response, or alternatively, by the bandwidth of the function. As a practical matter, no more than 8192 time points should be computed for the impulse response, due to memory and time limitations. The actual number of time points, N, is a logarithmic function of the value of RELTOL.

$$N = 2^{6 - \log_{10}(\mathrm{RELTOL})}$$

For example, for RELTOL= .001, 512 time points are computed.

As a general rule, Laplace sources will give the best transient analysis results on narrow band functions. Wide band functions, such as the differentiator, f(s)=s, and the integrator, f(s)=1/s, are best modeled by using discrete components. See the sample circuits INT (integrator macro) and DIF(differentiator macro).

Formula types

The input and output variables and definition names for the Laplace formula sources are as follows:

Source type	Input	Output	Definition
Current-controlled current source	I	I	LFIOFI
Current-controlled voltage source	I	V	LFVOFI
Voltage-controlled voltage source	V	V	LFVOFV
Voltage-controlled current source	V	I	LFIOFV

Here are some examples:

1/(1+.001*S)	a simple low pass filter.
1/(1+.001*s+1E-8*S*S)	a second order filter.
exp(-pow((C*S*(R+S*L)),.5))	equation of a simple lossy, transmission line. R, L, and C are .define constants.

For illustration, see the circuits L1, L2, and L3.

Table types

In a Table type, the transfer function is defined with a table. The table contains ordered triplets of numbers listing the frequency, magnitude or real value, and phase or imaginary value of the transfer function. The general form of the table entries is:

(F1,X1,Y1) (F2,X2,Y2) ... (FN,XN,YN)

Fi is the i'th frequency value in hertz.
Xi is the i'th magnitude value or the real value.
Yi is the i'th phase value or the imaginary value.

There are six rules for forming the table:

1. Values are separated by commas, triplets are enclosed in parentheses and are separated by spaces.

2. Data triplets must be arranged in order of ascending frequency.

3. The function is constant at X1,Y1 for inputs below F1.

4. The function is constant at XN,YN for inputs above FN.

5. The function is logarithmically interpolated for frequencies values between the table values.

6. The table should contain one data point at DC or zero frequency.

The table may be entered directly as the parameter string or indirectly using the .define statement. For illustration, see the circuit P1.

The input variable and output variables and definition names are as follows:

Source type	Input	Output	Definition
Current-controlled current source	I	I	LTIOFI
Current-controlled voltage source	I	V	LTVOFI
Voltage-controlled voltage source	V	V	LTVOFV
Voltage-controlled current source	V	I	LTIOFV

Macro

Schematic format

PART attribute
<*name*>

Example
2N5168

FILE attribute
<*macro circuit name*>

Example
SCR

Macros are the schematic equivalents of subcircuits. They are circuit building blocks that have been created and saved to disk for use in other circuits.

To create a macro:

1. Create a circuit. Place grid text on the nodes that you want to make available as macro pins. If you want to pass numeric parameters to the macro, use symbolic names for VALUE attributes and/or model parameter values and declare these names in a .parameters statement. Save the circuit to disk using the desired macro name.

2. Enter a component in the Component library as follows:

- Enter the macro file name for the Name field.
- Choose a suitable shape.
- Choose Macro for the Definition field.
- Place pins on the shape by clicking in the Shape drawing area. Name the pins with the same grid text names you placed on the nodes in the macro circuit.
- Add optional .MACRO statements to one of the *.LIB files to substitute long parameter lists for shorter names.

To use a macro:

Select the macro from the Component library. Place it in the circuit that will use it and edit its parameters, if it has any. You can also use an alias which, using a .macro statement, substitutes a short name like 2N5168 for the macro FILE name and a corresponding set of parameters.

The format of the macro command is:

.MACRO *<alias> <macro circuit name(parameter list)>*

This statement lets you store the parameters that turn a general macro for, say an SCR, into a specific model for a specific part like the 2N5168 SCR, and to access the part with a simple meaningful name, like 2N5168. For more information on the .MACRO statement see Chapter 20, "Command Statements".

When a macro is placed in a circuit, the program reads the macro circuit file, determines if it has parameters from the .PARAMETERS statement in the macro circuit file and shows these parameters and their default values in the Attribute dialog box. Edit the parameter values from their default values to the those you want.

MOSFET

SPICE format

Syntax

M<*name*> <*drain*> <*gate*> <*source*> <*bulk*> <*model name*>
[M=<*mval*>]
+ [L=<*length*>] [W=<*width*>] [AD=<*drainarea*>] [AS=<*sourcearea*>]
+ [PD=<*drainperiphery*>] [PS=<*sourceperiphery*>]
+ [NRD=<*drainsquares*>] [NRS=<*sourcesquares*>]
+ [NRG=<*gatesquares*>] [NRB=<*bulksquares*>]
+ [OFF][IC=<*vds*>[,*vgs*[,*vbs*]]]

Example

M1 5 7 9 0 IRF350 L=1.5E-6 W=0.25 OFF IC=25.0,8.0

Schematic format

PART attribute

<*name*>

Example

M1

VALUE attribute

[M=<*mval*>]
+ [L=<*length*>] [W=<*width*>] [AD=<*drainarea*>] [AS=<*sourcearea*>]
+ [PD=<*drainperiphery*>] [PS=<*sourceperiphery*>]
+ [NRD=<*drainsquares*>] [NRS=<*sourcesquares*>]
+ [NRG=<*gatesquares*>] [NRB=<*bulksquares*>]
+ [OFF][IC=<*vds*>[,*vgs*[,*vbs*]]]

Examples

M=20 NRD=10 NRS=25 NRG=5
L=.35u IC=.1, 2.00
L=.4u W=2u OFF IC=0.05,1.00

MODEL attribute

<*model name*>

Examples

IRF350
MM150

Both the original group of SPICE2 model levels 1, 2, and 3 and a BSIM group of models are supported. The BSIM models are supported as levels 4, 5, and 8. Level 4 is the original BSIM1 model. Level 2 is the improved BSIM2 model. Level 8 is the BSIM3 model version 3.3.2, derived from the UC Berkeley code dated 9/7/99. Levels 6 and 7 are not used.

<*width*> and <*length*> are the drawn device dimensions, before side diffusion, in meters. They can be specified in a circuit, or in a .MODEL or .OPTIONS statement. W and L VALUE attribute values supersede those in the .MODEL statement, which supersede those in the .OPTIONS statement.

The initialization [IC=<*vds*>[,*vgs*[,*vbs*]]] assigns initial voltages to the drain-source, gate-source, and body-source terms in transient analysis if no operating point is done (or if the UIC flag is set). The [OFF] keyword forces the device off during the first iteration of the DC operating point.

<*sourceperiphery*> and <*drainperiphery*> are the diffusion peripheries (m). <*sourcearea*> and <*drainarea*> are the diffusion areas (sq. m). Source and drain junction capacitances may be specified directly by the model parameters CBS and CBD. If absent, they are calculated from area and periphery terms.

The parasitic resistances may be specified directly with the model parameters RS, RD, RG, and RB. If unspecified, they are calculated from the product of the sheet resistivity, RSH, and the number of squares terms, <*drainsquares*>, <*sourcesquares*>, <*gatesquares*>, and <*bulksquares*>. If these terms are absent, or zero, and the model parameters RS, RD, RG, and RB are absent or zero, then the parasitic resistances are not included in the model.

<*drainsquares*> and <*sourcesquares*> default to 1.0. The other parameter line values default to zero. <*width*> and <*length*> default to DEFW and DEFL, defined in the Global Settings dialog box (SHIFT + CTRL + G).

<*mval*> is a multiplier (default = 1) that provides a way to simulate the effect of paralleling many devices. It multiplies the effective width, overlap, and junction capacitances, and the junction currents. It multiplies the drain and source areas, the device width, and the two peripheries, and divides the four resistances RS, RD, RG, and RB.

Model statement forms

.MODEL <*model name*> NMOS ([*model parameters*])
.MODEL <*model name*> PMOS ([*model parameters*])

Examples
.MODEL M1 NMOS (W=0.2 L=0.8U KP=1E-6 GAMMA=.65)
.MODEL M2 PMOS (W=0.1 L=0.9U KP=1.2E-6 LAMBDA=1E-3)

Common model parameters

These model parameters are common to all levels: All models except 8 share common default values. Level 8 default values are shown in the last column.

Name	**Parameter**	**Units**	**Default Values For Lev 1-5**	**Lev 8**
LEVEL	Model level		1	1
L	Channel length	m	DEFL	DEFL
W	Channel width	m	DEFW	DEFW
RDS	Drain-source shunt resistance	Ω	∞	∞
RD	Drain ohmic resistance	Ω	0.00	0.00
RS	Source ohmic resistance	Ω	0.00	0.00
RG	Gate ohmic resistance	Ω	0.00	0.00
RB	Bulk ohmic resistance	Ω	0.00	0.00
RSH	Source and drain sheet res.	Ω/sq	0.00	0.00
CGDO	Gate-drain overlap cap.	F/m	0.00	0.00
CGSO	Gate-source overlap cap.	F/m	0.00	0.00
CGBO	Gate-bulk overlap cap.	F/m	0.00	0.00
CBD	Bulk p-n zero-bias B-D cap.	F	0.00	0.00
CBS	Bulk p-n zero-bias B-S cap.	F	0.00	0.00
CJ	Bulk p-n zero-bias bot. cap.	F/m^2	0.00	5E-4
CJSW	Bulk p-n zero-bias s/w cap.	F/m	0.00	5E-10
MJ	Bulk p-n zero-bias bottom grad.		0.50	0.50
MJSW	Bulk p-n zero-bias s/w coeff.		0.33	0.33
TT	Bulk p-n transit time	S	0.00	0.00
IS	Bulk p-n saturation current	A	1E-14	1E-14
N	Bulk p-n emission coefficient		1.00	1.00
JS	Bulk p-n bot. current density	A/m^2	1E-8	1E-4
PB	Bulk p-n bottom potential	V	0.80	1.00
PBSW	Bulk p-n sidewall potential	V	PB	1.00
KF	Flicker-noise coefficient		0.00	0.00
AF	Flicker-noise exponent		1.00	1.00
FC	Forward-bias depletion coeff.		0.50	0.50
T_MEASURED	Measured temperature	°C		
T_ABS	Absolute temperature	°C		
T_REL_GLOBAL	Relative to current temp.	°C		
T_REL_LOCAL	Relative to AKO temperature	°C		

Model parameters for levels 1, 2, and 3

In addition to the 32 common parameters, the following table lists the additional parameters used in the level 1, 2, and 3 models.

Name	**Parameter**	**Units**	**Default**	**Level**
LD	Lateral diffusion length	m	0.00	1,2,3
WD	Lateral diffusion width	m	0.00	1,2,3
KP	Process transconductance	A/V^2	2E-5	1,2,3
VTO	Zero-bias threshold voltage	V	0.00	1,2,3
GAMMA	Body-effect coefficient	$V^{0.5}$	0.00	1,2,3
PHI	Surface inversion potential	V	0.60	1,2,3
LAMBDA	Channel-length modulation	V^{-1}	0.00	1,2
TOX	Thin oxide thickness	m	1E-7	1,2,3
UO	Surface mobility	$cm^2/V/s$	600	2,3
NEFF	Total channel charge coeff.		1.0	2
NSUB	Substrate doping density	cm^{-3}	None	2,3
NSS	Surface state density	cm^{-2}	None	2,3
NFS	Fast surface-state density	cm^{-2}	None	2,3
XJ	Metallurgical junction depth	m	0.00	2,3
VMAX	Max drift velocity of carriers	m/s	0.00	2,3
DELTA	Width effect on VTO		0.00	2,3
THETA	Mobility modulation	V^{-1}	0.00	3
ETA	Static feedback on VTO		0.00	3
KAPPA	Saturation field factor		0.20	3
TPG	Type of gate material		1.00	2,3
XQC	Coeff. of channel charge share		1.00	2,3
UCRIT	Mobility degrad. critical field	V/cm	1E4	2
UEXP	Mobility degradation exponent		0.00	2
UTRA	Mobility degrad. tr. field coeff.	m/s	0.00	2
GDSNOI	Channel shot noise coefficient		1.0	ALL
NLEV	Noise equation selector		2.0	ALL

Model equations for levels 1, 2, and 3

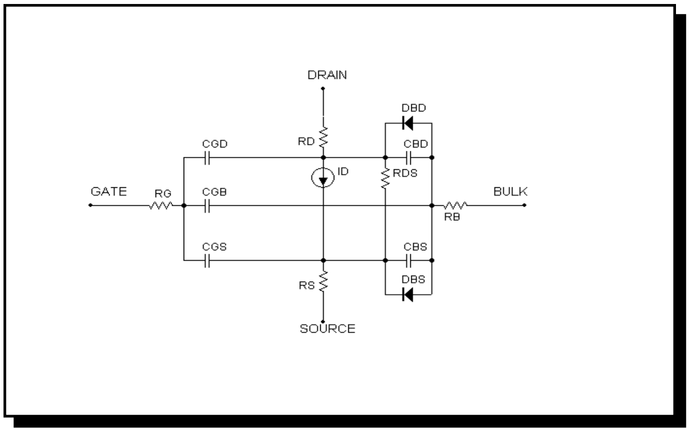


Figure 22-12 MOSFET model

Definitions

Vgs = Internal gate to source voltage

Vds = Internal drain to source voltage

Id = Drain current

VT = k·T/q

Temperature effects

T is the device operating temperature and Tnom is the temperature at which the model parameters are measured. Both are expressed in degrees Kelvin. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may be customized for each model by specifying the parameters T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. For details on how device temperatures and Tnom temperatures are calculated, see the .MODEL section of chapter 20 "Command Statements".

$EG(T) = 1.16 - .000702 \cdot T \cdot T/(T+1108)$

$IS(T) = IS \cdot e^{(EG(Tnom) \cdot T/Tnom - EG(T))/VT}$

$JS(T) = JS \cdot e^{(EG(Tnom) \cdot T/Tnom - EG(T))/VT}$

$JSSW(T) = JSSW \cdot e^{(EG(Tnom) \cdot T/Tnom - EG(T))/VT}$

$KP(T) = KP \cdot (T/Tnom)^{-1.5}$

$UO(T) = UO \cdot (T/Tnom)^{-1.5}$

$PB(T) = PB \cdot (T/Tnom) - 3 \cdot VT \cdot \ln((T/Tnom)) - EG(Tnom) \cdot (T/Tnom) + EG(T)$

$PBSW(T) = PBSW\cdot(T/Tnom)- 3\cdot VT\cdot ln((T/Tnom))-EG(Tnom)\cdot(T/Tnom)+EG(T)$
$PHI(T) = PHI\cdot(T/Tnom)- 3\cdot VT\cdot ln((T/Tnom))-EG(Tnom)\cdot(T/Tnom)+EG(T)$

$CBD(T) = CBD\cdot(1+MJ\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB)))$
$CBS(T) = CBS\cdot(1+MJ\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB)))$
$CJ(T) = CJ\cdot(1+MJ\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB)))$
$CJSW(T) = CJSW\cdot(1+MJ\cdot(.0004\cdot(T-Tnom) + (1 - PB(T)/PB))$

The parasitic lead resistances have no temperature dependence.

Current equations
Only the Level 1 drain equations are shown here. The Level 2 and Level 3 current equations are too complex for presentation in this manual. Interested users should consult reference (2) for more information.

$K = KP \cdot W/ (L - 2 \cdot LD)$
$VTH = VTO + GAMMA \cdot ((PHI - VBS)^{1/2} - \sqrt{(PHI)})$

Cutoff region: For Vgs < VT

$Id = 0.0$

Linear region: For Vgs > VTH and Vds < (Vgs - VTH)

$Id = K\cdot(Vgs - VTH - 0.5\cdot Vds)\cdot Vds\cdot(1+LAMBDA\cdot Vds)$

Saturation region: For Vgs > VTH and Vds > (Vgs - VTH)

$Id = 0.5\cdot K\cdot(Vgs - VTH)^2\cdot(1 + LAMBDA\cdot Vds)$

These equations are for an N-channel device.

Capacitance equations
Meyer model for gate capacitance
All levels use the SPICE 2G.6 capacitance model proposed by Meyer when XQC is specified or greater than 0.5. If XQC is specified and is less than or equal to .5, the Ward model is used. Meyer's model does not guarantee charge conservation, but is generally more robust than the Ward model.

The charges are modeled by three nonlinear capacitances, Cgb, Cgd, and Cgs.

$Cox = COX \cdot W \cdot Leff$

Accumulation region (Vgs < Von - PHI)
For Vgs < Von - PHI,
$Cgb = Cox + CGBO \cdot Leff$
$Cgs = CGSO \cdot W$
$Cgd = CGDO \cdot W$

Depletion region (Von - PHI < Vgs < Von)
$Cgb = Cox \cdot (Von - Vgs)/PHI + CGBO \cdot Leff$
$Cgs = 2/3 \cdot Cox \cdot ((Von - Vgs)/PHI + 1) + CGSO \cdot W$
$Cgd = CGDO \cdot W$

Saturation region (Von < Vgs < Von + Vds)
$Cgb = CGBO \cdot Leff$
$Cgs = 2/3 \cdot Cox + CGSO \cdot W$
$Cgd = CGDO \cdot W$

Linear region:
For Vgs > Von + Vds,
$Cgb = CGBO \cdot Leff$
$Cgs = Cox \cdot (1 - ((Vgs - Vds - Von)/(2 \cdot (Vgs - Von) - Vds))^{2)} + CGSO \cdot W$
$Cgd = Cox \cdot (1 - ((Vgs - Von)/(2 \cdot (Vgs - Von) - Vds))^2) + CGDO \cdot W$

Junction capacitance
The junction capacitance is modeled by two nonlinear capacitors, Cbs and Cbd.

If CBS=0 and CBD=0 then
$Cbs = CJ(T) \cdot AS \cdot f1(VBS) + CJSW(T) \cdot PS \cdot f2(VBS) + TT \cdot GBS$
$Cbd = CJ(T) \cdot AD \cdot f1(VBD) + CJSW(T) \cdot PD \cdot f2(VBD) + TT \cdot GBD$
else
$Cbs = CBS(T) \cdot f1(VBS) + CJSW(T) \cdot PS \cdot f2(VBS) + TT \cdot GBS$
$Cbd = CBD(T) \cdot f1(VBD) + CJSW(T) \cdot PD \cdot f2(VBD) + TT \cdot GBD$

GBS= DC bulk-source conductance = d(IBS)/d(VBS)
GBD= DC bulk-drain conductance = d(IBD)/d(VBD)

If $VBS \leq FC \cdot PB(T)$ then
$f1(VBS) = 1/(1 - VBS/PB(T))^M$
Else
$f1(VBS) = (1 - FC \cdot (1+M)+M \cdot (VBS/PB(T)))/ (1 - FC)^{(1-M)}$

If VBS $\leq$ FC $\cdot$ PBSW(T) then

$$f2(VBS) = 1/ (1 - VBS/PBSW(T))^{M}$$

Else

$$f2(VBS) = (1 - FC\cdot(1+M)+M\cdot(VBS/PBSW(T)))/ (1 - FC)^{(1-M)}$$

If VBD $\leq$ FC $\cdot$ PB(T) then

$$f1(VBD) = 1/(1 - VBD/PB(T))^{M}$$

Else

$$f1(VBD) = (1 - FC\cdot(1+M)+M\cdot(VBS/PB(T)))/ (1 - FC)^{(1-M)}$$

If VBD $\leq$ FC $\cdot$ PBSW(T) then

$$f2(VBD) = 1/ (1 - VBD/PBSW(T))^{M}$$

Else

$$f2(VBD) = (1 - FC\cdot(1+M)+M\cdot(VBD/PBSW(T)))/ (1 - FC)^{(1-M)}$$

Model parameters for level 4

These are the model parameters for the BSIM1 model, level 4. There are no default values. All parameter values must be specified.

Name	Parameter	Units
DL	Channel length reduction	μ
DW	Channel width reduction	μ
TOX	Gate oxide thickness	μ
VFB	Flat band voltage	V
VFBL	Length dependence of VFB	V/m
VFBW	Width dependence of VFB	V/m
PHI	Strong inversion surface potential	V
PHIL	Length dependence of PHI	V/m
PHIW	Width dependence of PHI	V/m
K1	Bulk effect coefficient 1	√V
K1L	Length dependence of K1	√V/m
K1W	Width dependence of K1	√V/m
K2	Bulk effect coefficient 2	
K2L	Length dependence of K2	1/m
K2W	Width dependence of K2	1/m
ETA	VDS dependence of threshold voltage	
LETA	Length dependence of ETA	1/m
WETA	Width dependence of ETA	1/m
X2E	VBS dependence of ETA	V^{-1}
LX2E	Length dependence of X2E	V^{-1}/m
WX2E	Width dependence of X2E	V^{-1}/m
X3E	VDS dependence of ETA	V^{-1}
LX3E	Length dependence of X3E	V^{-1}/m
WX3E	Width dependence of X3E	V^{-1}/m
MUZ	Mobility at vds=0,vgs=vth	cm^2/Vs
X2MZ	VBS dependence of MUZ	
LX2MZ	Length dependence of X2MZ	
WX2MZ	Width dependence of X2MZ	
MUS	Mobility at vds=vdd,vgs=vth	
LMUS	Length dependence of MUS	
WMUS	Width dependence of MUS	
X2MS	VBS dependence of MUS	
LX2MS	Length dependence of X2MS	
WX2MS	Width dependence of X2MS	
X3MS	VDS dependence of MUS	
LX3MS	Length dependence of X3MS	

Model parameters for level 4 (continued)

Name	Parameter	Units
WX3MS	Width dependence of X3MS	
U0	VGS dependence of mobility	
LU0	Length dependence of U0	
WU0	Width dependence of U0	
X2U0	VBS dependence of U0	
LX2U0	Length dependence of X2U0	
WX2U0	Width dependence of X2U0	
U1	VDS dependence of mobility	
LU1	Length dependence of U1	
WU1	Width dependence of U1	
X2U1	VBS dependence of U1	
LX2U1	Length dependence of X2U1	
WX2U1	Width dependence of X2U1	
X3U1	VDS dependence of U1	
LX3U1	Length dependence of X3U1	
WX3U1	Width dependence of X3U1	
N0	Subthreshold slope	
LN0	Length dependence of N0	
WN0	Width dependence of N0	
NB	VBS dependence of subthreshold slope	
LNB	Length dependence of NB	
WNB	Width dependence of NB	
ND	VDS dependence of subthreshold slope	
LND	Length dependence of ND	
WND	Width dependence of ND	
VDD	Supply voltage for MUS	
XPART	Channel charge partitioning flag	
DELL	Unused: Length reduction of S-D diff.	

Model parameters for level 5

These are the model parameters for the BSIM 2 model, level 5.

Name	Parameter	Units	Default
DL	Channel length reduction	μ	0.00
DW	Channel width reduction	μ	0.00
TOX	Gate oxide thickness	μ	0.00
VFB	Flat band voltage	V	-1.00
VFBL	Length dependence of VFB	V•μ	0.00
VFBW	Width dependence of VFB	V•μ	0.00
PHI	Strong inversion surface potential	V	0.75
PHIL	Length dependence of PHI	V•μ	0.00
PHIW	Width dependence of PHI	V•μ	0.00
K1	Bulk effect coefficient 1	√V	0.80
K1L	Length dependence of K1	√V•μ	0.00
K1W	Width dependence of K1	√V•μ	0.00
K2	Bulk effect coefficient 2		0.00
K2L	Length dependence of K2	μ	0.00
K2W	Width dependence of K2	μ	0.00
ETA0	VDS dependence of threshold voltage		0.00
LETA	Length dependence of ETA0	μ	0.00
WETA	Width dependence of ETA0	μ	0.00
ETAB	VBS dependence of ETA0	V^{-1}	0.00
LETAB	Length dependence of ETAB	V^{-1}•μ	0.00
WETAB	Width dependence of ETAB	V^{-1}•μ	0.00
MU0	Mobility at vds=0,vgs=vth	m^2/V•s	400
MU0B	VBS dependence of MU0	m^2/V^2•s	0.00
LMU0B	Length dependence of MU0B	μ•m^2/V^2•s	0.00
WMU0B	Width dependence of MU0B	μ•m^2/V^2•s	0.00
MUS0	Mobility at vds=vdd,vgs=vth	m^2/V•s	500
LMUS0	Length dependence of MUS0	μ•m^2/V•s	0.00
WMUS0	Width dependence of MUS0	μ•m^2/V•s	0.00
MUSB	VBS dependence of MUS	m^2/V^2•s	0.00
LMUSB	Length dependence of MUSB	μ•m^2/V^2•s	0.00
WMUSB	Width dependence of MUSB	μ•m^2/V^2•s	0.00
MU20	VDS dependence of MU in tanh term	m^2/V^2•s	1.5
LMU20	Length dependence of MU20	μ•m^2/V^2•s	0.00
WMU20	Width dependence of MU20	μ•m^2/V^2•s	0.00
MU2B	VBS dependence of MU2	m^2/V^3•s	0.00
LMU2B	Length dependence of MU2B	μ•m^2/V^3•s	0.00
WMU2B	Width dependence of MU2B	μ•m^2/V^3•s	0.00

Model parameters for level 5 (continued)

Name	Parameter	Units	Default
MU2G	VGS dependence of MU2	$m^2/V^3 \cdot s$	0.00
LMU2G	Length dependence of MU2G	$\mu \cdot m^2/V^3 \cdot s$	0.00
WMU2G	Width dependence of MU2G	$\mu \cdot m^2/V^3 \cdot s$	0.00
MU30	VDS dependence of MU in linear term	$m^2/V^2 \cdot s$	10.0
LMU30	Length dependence of MU30	$\mu \cdot m^2/V^2 \cdot s$	0.00
WMU30	Width dependence of MU30	$\mu \cdot m^2/V^2 \cdot s$	0.00
MU3B	VBS dependence of MU3	$m^2/V^3 \cdot s$	0.00
LMU3B	Length dependence of MU3B	$\mu \cdot m^2/V^3 \cdot s$	0.00
WMU3B	Width dependence of MU3B	$\mu \cdot m^2/V^3 \cdot s$	0.00
MU3G	VGS dependence of MU3	$m^2/V^3 \cdot s$	0.00
LMU3G	Length dependence of MU3G	$\mu \cdot m^2/V^3 \cdot s$	0.00
WMU3G	Width dependence of MU3G	$\mu \cdot m^2/V^3 \cdot s$	0.00
MU40	VDS dependence of MU in quad. term	$m^2/V^4 \cdot s$	0.00
LMU40	Length dependence of MU40	$\mu \cdot m^2/V^4 \cdot s$	0.00
WMU40	Width dependence of MU40	$\mu \cdot m^2/V^4 \cdot s$	0.00
MU4B	VBS dependence of MU4	$m^2/V^5 \cdot s$	0.00
LMU4B	Length dependence of MU4B	$\mu \cdot m^2/V^3 \cdot s$	0.00
WMU4B	Width dependence of MU4B	$\mu \cdot m^2/V^3 \cdot s$	0.00
MU4G	VGS dependence of MU4	$m^2/V^5 \cdot s$	0.00
LMU4G	Length dependence of MU4G	$\mu \cdot m^2/V^5 \cdot s$	0.00
WMU4G	Width dependence of MU4G	$\mu \cdot m^2/V^5 \cdot s$	0.00
UA0	Linear VGS dependence of mobility	$m^2/V^2 \cdot s$	0.20
LUA0	Length dependence of UA0	$\mu \cdot m^2/V^2 \cdot s$	0.00
WUA0	Width dependence of UA0	$\mu \cdot m^2/V^2 \cdot s$	0.00
UAB	VBS dependence of UA	$m^2/V^3 \cdot s$	0.00
LUAB	Length dependence of UAB	$\mu \cdot m^2/V^3 \cdot s$	0.00
WUAB	Width dependence of UAB	$\mu \cdot m^2/V^3 \cdot s$	0.00
UB0	Quadratic VGS dependence of mobility	$m^2/V^3 \cdot s$	0.00
LUB0	Length dependence of UB0	$\mu \cdot m^2/V^3 \cdot s$	0.00
WUB0	Width dependence of UB0	$\mu \cdot m^2/V^3 \cdot s$	0.00
UBB	VBS dependence of UB	$m^2/V^4 \cdot s$	0.00
LUBB	Length dependence of UBB	$\mu \cdot m^2/V^4 \cdot s$	0.00
WUBB	Width dependence of UBB	$\mu \cdot m^2/V^4 \cdot s$	0.00
U10	VDS dependence of mobility	$m^2/V^2 \cdot s$	0.10
LU10	Length dependence of U10	$\mu \cdot m^2/V^2 \cdot s$	0.00
WU10	Width dependence of U10	$\mu \cdot m^2/V^2 \cdot s$	0.00
U1B	VBS dependence of U1	$m^2/V^3 \cdot s$	0.00
LU1B	Length dependence of U1B	$\mu \cdot m^2/V^3 \cdot s$	0.00

Model parameters for level 5 (continued)

Name	Parameter	Units	Default
WU1B	Width dependence of U1B	$\mu \cdot m^2/V^3 \cdot s$	0.00
U1D	VDS dependence of U1	$m^2/V^3 \cdot s$	0.00
LU1D	Length dependence of U1D	$\mu \cdot m^2/V^3 \cdot s$	0.00
WU1D	Width dependence of U1D	$\mu \cdot m^2/V^3 \cdot s$	0.00
N0	Subthreshold slope at vds=0,vbs=0		1.40
LN0	Length dependence of N0	μ	0.00
WN0	Width dependence of N0	μ	0.00
NB	VBS dependence of N	√V	0.50
LNB	Length dependence of NB	μ•√V	0.00
WNB	Width dependence of NB	μ•√V	0.00
ND	VDS dependence of N	V^{-1}	0.00
LND	Length dependence of ND	$\mu \cdot V^{-1}$	0.00
WND	Width dependence of ND	$\mu \cdot V^{-1}$	0.00
VOF0	Threshold voltage offset at vds=0,vbs=0	V	1.80
LVOF0	Length dependence of VOF0	μ•V	0.00
WVOF0	Width dependence of VOF0	μ•V	0.00
VOFB	VBS dependence of VOF		0.00
LVOFB	Length dependence of VOFB	μ	0.00
WVOFB	Width dependence of VOFB	μ	0.00
VOFD	VDS dependence of VOF		0.00
LVOFD	Length dependence of VOFD	μ	0.00
WVOFD	Width dependence of VOFD	μ	0.00
AI0	Pre-factor of hot electron effect		0.00
LAI0	Length dependence of AI0	μ	0.00
WAI0	Width dependence of AI0	μ	0.00
AIB	VBS dependence of AI	V^{-1}	0.00
LAIB	Length dependence of AIB	$\mu \cdot V^{-1}$	0.00
WAIB	Width dependence of AIB	$\mu \cdot V^{-1}$	0.00
BI0	Exponential factor of hot electron effect		0.00
LBI0	Length dependence of BI	μ	0.00
WBI0	Width dependence of BI	μ	0.00
BIB	VBS dependence of BI	V^{-1}	0.00
LBIB	Length dependence of BIB	$\mu \cdot V^{-1}$	0.00
WBIB	Width dependence of BIB	$\mu \cdot V^{-1}$	0.00
VGHIGH	Upper bound of cubic spline function	μ•V	0.20
LVGHIGH	Length dependence of VGHIGH	μ•V	0.00
WVGHIGH	Width dependence of VGHIGH	μ•V	0.00
VGLOW	Lower bound of cubic spline function	V	-0.15

Model parameters for level 5 (continued)

Name	Parameter	Units	Default
LVGLOW	Length dependence of VGLOW	μ•V	0.00
WVGLOW	Width dependence of VGLOW	μ•V	0.00
VDD	Maximum VDS	V	5.00
VGG	Maximum VGS	V	5.00
VBB	Maximum VBS	V	5.00
XPART	Channel charge partitioning flag		0.00
DELL	Unused: length reduction of S-D diff.	m	0.00

Model parameters for level 8

These are the model parameters for the BSIM 3 model, level 8. This is the latest version 3.2 of the Berkeley BSIM3 model, dated 10/22/98.

Name	Parameter	Default
NJ	Emission coefficient (takes priority over N)	0.00
XTI	Junction current temperature exponent coefficient	0.00
TPB	Temperature coefficient of PB	0.00
TPBSW	Temperature coefficient of PBSW	0.00
PBSWG	S/D gate sidewall junction built-in potential	0.00
TPBSWG	Temperature coefficient of PBSWG	0.00
TCJ	Temperature coefficient of CJ	0.00
TCJSW	Temperature coefficient of CJSW	0.00
CJSWG	S/D gate sidewall junction capacitance	0.00
TCJSWG	Temperature coefficient of CJSWG	0.00
MJSWG	S/D gate sidewall junction capacitance grading coeff.	0.00
LMIN	Minimum channel length	0.00
LMAX	Maximum channel length	1.00
WMIN	Minimum channel width	0.00
WMAX	Maximum channel width	1.00
BINUNIT	Bin units selector	1.00
MOBMOD	Mobility model	1.00
CAPMOD	Flag for short channel cap. model	3.00
NQSMOD	Flag for NQS model	0.00
NOIMOD	Flag for noise model	1.00
PARAMCHK	Parameter check flag	0.00
VERSION	Version number 3.1 or 3.2	3.20
VTH0	VTH at vbs=0 for large channel length	0.00
LVTH0	Length dependence of VTH0	0.00
WVTH0	Width dependence of VTH0	0.00
PVTH0	Length and width product dependence of VTH0	0.00
K1	First-order body effect coefficient	0.00
LK1	Length dependence of K1	0.00
WK1	Width dependence of K1	0.00
PK1	Length and width product dependence of K1	0.00
K2	Second-order body effect coefficient	0.00
LK2	Length dependence of K2	0.00
WK2	Width dependence of K2	0.00
PK2	Length and width product dependence of K2	0.00
K3	Narrow width effect coefficient	80.00
LK3	Length dependence of K3	0.00

Model parameters for level 8 (continued)

Name	Parameter	
WK3	Width dependence of K3	0.00
PK3	Length and width product dependence of K3	0.00
K3B	Body effect coefficient of K3	0.00
LK3B	Length dependence of K3B	0.00
WK3B	Width dependence of K3B	0.00
PK3B	Length and width product dependence of K3B	0.00
W0	Narrow width effect parameter	2.5E-6
LW0	Length dependence of W0	0.00
WW0	Width dependence of W0	0.00
PW0	Length and width product dependence of W0	0.00
NLX	Lateral non-uniform doping coefficient	1.74E-7
LNLX	Length dependence of NLX	0.00
WNLX	Width dependence of NLX	0.00
PNLX	Length and width product dependence of NLX	0.00
DVT0	First coefficent of short-channel effect on VTH	2.20
LDVT0	Length dependence of DVT0	0.00
WDVT0	Width dependence of DVT0	0.00
PDVT0	Length and width product dependence of DVT0	0.00
DVT1	Second coefficent of short-channel effect on VTH	0.53
LDVT1	Length dependence of DVT1	0.00
WDVT1	Width dependence of DVT1	0.00
PDVT1	Length and width product dependence of DVT1	0.00
DVT2	Body bias coefficent of short-channel effect on VTH	-.032
LDVT2	Length dependence of DVT2	0.00
WDVT2	Width dependence of DVT2	0.00
PDVT2	Length and width product dependence of DVT2	0.00
DVT0W	First coefficient of narrow width effect on Vth for small channel length	0.00
LDVT0W	Length dependence of DVT0W	0.00
WDVT0W	Width dependence of DVT0W	0.00
PDVT0W	Length and width product dependence of DVT0W	0.00
DVTIW	Second coefficient of narrow width effect on Vth for small channel length	5.3E6
LDVT1W	Length dependence of DVT1W	0.00
WDVT1W	Width dependence of DVT1W	0.00
PDVT1W	Length and width product dependence of DVT1W	0.00
DVT2W	Body-bias coefficient of narrow width effect on Vth for small channel length	-.032

Model parameters for level 8 (continued)

Name	Parameter	Default
LDVT2W	Length dependence of DVT2W	0.00
WDVT2W	Width dependence of DVT2W	0.00
PDVT2W	Length and width product dependence of DVT2W	0.00
U0	Mobility at Temp = TNOM	0.00
LU0	Length dependence of U0	0.00
WU0	Width dependence of U0	0.00
PU0	Length and width product dependence of U0	0.00
UA	Linear gate dependence of mobility	2.25E-9
LUA	Length dependence of UA	0.00
WUA	Width dependence of UA	0.00
PUA	Length and width product dependence of UA	0.00
UB	Quadratic gate dependence of mobility	5.87E-19
LUB	Length dependence of UB	0.00
WUB	Width dependence of UB	0.00
PUB	Length and width product dependence of UB	0.00
UC	Body-bias dependence of mobility	0.00
LUC	Length dependence of UC	0.00
WUC	Width dependence of UC	0.00
PUC	Length and width product dependence of UC	0.00
VSAT	Saturation velocity at Temp = TNOM	8.0E4
LVSAT	Length dependence of VSAT	0.00
WVSAT	Width dependence of VSAT	0.00
PVSAT	Length and width product dependence of VSAT	0.00
A0	Bulk charge effect coefficient	1.00
LA0	Length dependence of A0	0.00
WA0	Width dependence of A0	0.00
PA0	Length and width product dependence of A0	0.00
AGS	Gate bias coefficient for channel length	0.00
LAGS	Length dependence of AGS	0.00
WAGS	Width dependence of AGS	0.00
PAGS	Length and width product dependence of AGS	0.00
B0	Bulk charge effect coefficient for channel width	0.00
LB0	Length dependence of B0	0.00
WB0	Width dependence of B0	0.00
PB0	Length and width product dependence of B0	0.00
B1	Bulk charge effect width offset	0.00
LB1	Length dependence of B1	0.00
WB1	Width dependence of B1	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
PB1	Length and width product dependence of B1	0.00
KETA	Body bias coefficient of the bulk charge effect	-.047
LKETA	Length dependence of KETA	0.00
WKETA	Width dependence of KETA	0.00
PKETA	Length and width product dependence of KETA	0.00
A1	First non-saturation effect parameter	0.00
LA1	Length dependence of A1	0.00
WA1	Width dependence of A1	0.00
PA1	Length and width product dependence of A1	0.00
A2	Second non-saturation effect parameter	1.00
LA2	Length dependence of A2	0.00
WA2	Width dependence of A2	0.00
PA2	Length and width product dependence of A2	0.00
RDSW	Parasitic resistance per unit width	0.00
LRDSW	Length dependence of RDSW	0.00
WRDSW	Width dependence of RDSW	0.00
PRDSW	Length and width product dependence of RDSW	0.00
PRWB	Body effect coefficient of RDSW	0.00
LPRWB	Length dependence of PRWB	0.00
WPRWB	Width dependence of PRWB	0.00
PPRWB	Length and width product dependence of PRWB	0.00
PRWG	Gate bias effect coefficient of RDSW	0.00
LPRWG	Length dependence of PRWG	0.00
WPRWG	Width dependence of PRWG	0.00
PPRWG	Length and width product dependence of PRWG	0.00
WR	Width offset from Weff for Rds calculation	1.00
LWR	Length dependence of WR	0.00
WWR	Width dependence of WR	0.00
PWR	Length and width product dependence of WR	0.00
WINT	Width offset fitting parameter from I-V without bias	0.00
LINT	Length offset fitting parameter from I-V without bias	0.00
DWG	Coefficient of Weff's dependence	0.00
LDWG	Length dependence of DWG	0.00
WDWG	Width dependence of DWG	0.00
PDWG	Length and width product dependence of DWG	0.00
DWB	Coefficient of Weff's substrate body bias dependence	0.00
LDWB	Length dependence of DWB	0.00
WDWB	Width dependence of DWB	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
PDWB	Length and width product dependence of DWB	0.00
VOFF	Offset voltage in the subthr. region for large W and L	-0.08
LVOFF	Length dependence of VOFF	0.00
WVOFF	Width dependence of VOFF	0.00
PVOFF	Length and width product dependence of VOFF	0.00
NFACTOR	Subthreshold swing factor	1.00
LNFACTOR	Length dependence of NFACTOR	0.00
WNFACTOR	Width dependence of NFACTOR	0.00
PNFACTOR	Length and width product dependence of NFACTOR	0.00
ETA0	DIBL coefficient in subthreshold region	0.08
LETA0	Length dependence of ETA0	0.00
WETA0	Width dependence of ETA0	0.00
PETA0	Length and width product dependence of ETA0	0.00
ETAB	Body bias coefficient for the subthr. DIBL coefficient	-0.07
LETAB	Length dependence of ETAB	0.00
WETAB	Width dependence of ETAB	0.00
PETAB	Length and width product dependence of ETAB	0.00
DSUB	DIBL coefficient exponent in subthreshold region	0.00
LDSUB	Length dependence of DSUB	0.00
WDSUB	Width dependence of DSUB	0.00
PDSUB	Length and width product dependence of DSUB	0.00
CIT	Interface trap capacitance	0.00
LCIT	Length dependence of CIT	0.00
WCIT	Width dependence of CIT	0.00
PCIT	Length and width product dependence of CIT	0.00
CDSC	Drain/source to channel coupling capacitance	2.4E-4
LCDSC	Length dependence of CDSC	0.00
WCDSC	Width dependence of CDSC	0.00
PCDSC	Length and width product dependence of CDSC	0.00
CDSCB	Body-bias sensitivity of CDSC	0.00
LCDSCB	Length dependence of CDSCB	0.00
WCDSCB	Width dependence of CDSCB	0.00
PCDSCB	Length and width product dependence of CDSCB	0.00
CDSCD	Drain-bias sensitivity of CDSC	0.00
LCDSCD	Length dependence of CDSCD	0.00
WCDSCD	Width dependence of CDSCD	0.00
PCDSCD	Length and width product dependence of CDSCD	0.00
PCLM	Channel length modulation parameter	1.30

Model parameters for level 8 (continued)

Name	Parameter	Default
LPCLM	Length dependence of PCLM	0.00
WPCLM	Width dependence of PCLM	0.00
PPCLM	Length and width product dependence of PCLM	0.00
PDIBLC1	1'st output resist. DIBL correction parameter	0.39
LPDIBLC1	Length dependence of PDIBLC1	0.00
WPDIBLC1	Width dependence of PDIBLC1	0.00
PPDIBLC1	Length and width product dependence of PDIBLC1	0.00
PDIBLC2	2'nd output resist. DIBL correction parameter	0.0086
LPDIBLC2	Length dependence of PDIBLC2	0.00
WPDIBLC2	Width dependence of PDIBLC2	0.00
PPDIBLC2	Length and width product dependence of PDIBLC2	0.00
PDIBLCB	Body coefficient of DIBL parameters	0.00
LPDIBLCB	Length dependence of PDIBLCB	0.00
WPDIBLCB	Width dependence of PDIBLCB	0.00
PPDIBLCB	Length and width product dependence of PDIBLCB	0.00
DROUT	L dep. coeff. of the DIBL corr. parameter in ROUT	0.56
LDROUT	Length dependence of DROUT	0.00
WDROUT	Width dependence of DROUT	0.00
PDROUT	Length and width product dependence of DROUT	0.00
PSCBE1	First substrate current body effect parameter	4.24E8
LPSCBE1	Length dependence of PSCBE1	0.00
WPSCBE1	Width dependence of PSCBE1	0.00
PPSCBE1	Length and width product dependence of PSCBE1	0.00
PSCBE2	Second substrate current body effect parameter	1E-5
LPSCBE2	Length dependence of PSCBE2	0.00
WPSCBE2	Width dependence of PSCBE2	0.00
PPSCBE2	Length and width product dependence of PSCBE2	0.00
PVAG	Gate dependence of Early voltage	0.00
LPVAG	Length dependence of PVAG	0.00
WPVAG	Width dependence of PVAG	0.00
PPVAG	Length and width product dependence of PVAG	0.00
DELTA	Effective Vds parameter	0.01
LDELTA	Length dependence of DELTA	0.00
WDELTA	Width dependence of DELTA	0.00
PDELTA	Length and width product dependence of DELTA	0.00
NGATE	Poly gate doping concentration	0.00
LNGATE	Length dependence of NGATE	0.00
WNGATE	Width dependence of NGATE	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
PNGATE	Length and width product dependence of NGATE	0.00
ALPHA0	Substrate current model parameter 1	0.00
LALPHA0	Length dependence of ALPHA0	0.00
WALPHA0	Width dependence of ALPHA0	0.00
PALPHA0	Length and width product dependence of ALPHA0	0.00
ALPHA1	Substrate current model parameter 2	0.00
LALPHA1	Length dependence of ALPHA1	0.00
WALPHA1	Width dependence of ALPHA1	0.00
PALPHA1	Length and width product dependence of ALPHA1	0.00
IJTH	Diode limiting current	0.10
BETA0	Substrate current model parameter 3	30.00
LBETA0	Length dependence of BETA0	0.00
WBETA0	Width dependence of BETA0	0.00
PBETA0	Length and width product dependence of BETA0	0.00
XPART	Channel charge partitioning flag	0.00
CGSL	Non-LDD region s-g overlap cap. per unit ch. length	0.00
LCGSL	Length dependence of CGSL	0.00
WCGSL	Width dependence of CGSL	0.00
PCGSL	Length and width product dependence of CGSL	0.00
CGDL	Non-LDD region d-g overlap cap. per unit ch. length	0.00
LCGDL	Length dependence of CGDL	0.00
WCGDL	Width dependence of CGDL	0.00
PCGDL	Length and width product dependence of CGDL	0.00
CKAPPA	Coefficient for lightly doped region overlap fringing field capacitance	0.60
LCKAPPA	Length dependence of CKAPPA	0.00
WCKAPPA	Width dependence of CKAPPA	0.00
PCKAPPA	Length and width product dependence of CKAPPA	0.00
CF	Fringing field capacitance	0.00
LCF	Length dependence of CF	0.00
WCF	Width dependence of CF	0.00
PCF	Length and width product dependence of CF	0.00
CLC	Constant term for short channel model	1E-7
LCLC	Length dependence of CLC	0.00
WCLC	Width dependence of CLC	0.00
PCLC	Length and width product dependence of CLC	0.00
CLE	Exponential term for short channel model	0.60
LCLE	Length dependence of CLE	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
WCLE	Width dependence of CLE	0.00
PCLE	Length and width product dependence of CLE	0.00
DLC	Length offset fitting parameter from C-V	0.00
LDLC	Length dependence of DLC	0.00
WDLC	Width dependence of DLC	0.00
PDLC	Length and width product dependence of DLC	0.00
DWC	Width offset fitting parameter from C-V	0.00
LDWC	Length dependence of DWC	0.00
WDWC	Width dependence of DWC	0.00
PDWC	Length and width product dependence of DWC	0.00
WLC	Width reduction parameter for CV	0.00
WWC	Width reduction parameter for CV	0.00
WWLC	Width reduction parameter for CV	0.00
LLC	Length reduction parameter for CV	0.00
LWC	Length reduction parameter for CV	0.00
LWLC	Length reduction parameter for CV	0.00
VFBCV	Flat-band voltage parameter (for capmod=0 only)	-1.00
LVFBCV	Length dependence of VFBCV	0.00
WVFBCV	Width dependence of VFBCV	0.00
PVFBCV	Length and width product dependence of VFBCV	0.00
NOFF	C-V turn-on/off parameter	1.00
LNOFF	Length dependence of NOFF	0.00
WNOFF	Width dependence of NOFF	0.00
PNOFF	Length and width product dependence of NOFF	0.00
MOIN	Coefficient for gate-bias dependent surface potential	15.00
LMOIN	Length dependence of MOIN	0.00
WMOIN	Width dependence of MOIN	0.00
PMOIN	Length and width product dependence of MOIN	0.00
VOFFCV	C-V lateral-shift parameter	0.00
LVOFFCV	Length dependence of VOFFCV	0.00
WVOFFCV	Width dependence of VOFFCV	0.00
PVOFFCV	Length and width product dependence of VOFFCV	0.00
ACDE	Exponential coefficient for finite charge thickness	1.00
LACDE	Length dependence of ACDE	0.00
WACDE	Width dependence of ACDE	0.00
PACDE	Length and width product dependence of ACDE	0.00
ELM	Elmore constant of the channel	5.00
LELM	Length dependence of ELM	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
WELM	Width dependence of ELM	0.00
PELM	Length and width product dependence of ELM	0.00
WL	Coefficient for length dependence of width offset	0.00
WLN	Power of length dependence of width offset	1.00
WW	Coefficient for width dependence of width offset	0.00
WWN	Power of width dependence of width offset	1.00
WWL	Coeff. of len. and width cross term for width off.	0.00
LL	Coefficient for length dependence for length offset	0.00
LLN	Power of length dependence for length offset	1.00
LW	Coefficient for width dependence for length offset	0.00
LWN	Power of width dependence for length offset	1.00
LWL	Coeff. of length and width cross term for length offset	0.00
TNOM	Temperature at which parameters are extracted	27.00
UTE	Mobility temperature exponent	-1.50
LUTE	Length dependence of UTE	0.00
WUTE	Width dependence of UTE	0.00
PUTE	Length and width product dependence of UTE	0.00
KT1	Temperature coefficient for threshold voltage	-0.11
LKT1	Length dependence of KT1	0.00
WKT1	Width dependence of KT1	0.00
PKT1	Length and width product dependence of KT1	0.00
KT1L	Channel length sensitivity of temperature coefficient for threshold voltage	0.00
LKT1L	Length dependence of KT1L	0.00
WKT1L	Width dependence of KT1L	0.00
PKT1L	Length and width product dependence of KT1L	0.00
KT2	Body bias coefficient of the threshold voltage temperature effect	0.022
LKT2	Length dependence of KT2	0.00
WKT2	Width dependence of KT2	0.00
PKT2	Length and width product dependence of KT2	0.00
UA1	Temperature coefficient for UA	4.31E-9
LUA1	Length dependence of UA1	0.00
WUA1	Width dependence of UA1	0.00
PUA1	Length and width product dependence of UA1	0.00
UB1	Temperature coefficient for UB	-7.61E-18
LUB1	Length dependence of UB1	0.00
WUB1	Width dependence of UB1	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
PUB1	Length and width product dependence of UB1	0.00
UC1	Temperature coefficient for UC	0.00
LUC1	Length dependence of UC1	0.00
WUC1	Width dependence of UC1	0.00
PUC1	Length and width product dependence of UC1	0.00
AT	Saturation velocity temperature coefficient	3.3E4
LAT	Length dependence of AT	0.00
WAT	Width dependence of AT	0.00
PAT	Length and width product dependence of AT	0.00
PRT	Temperature coefficient for RDSW	0.00
LPRT	Length dependence of PRT	0.00
WPRT	Width dependence of PRT	0.00
PPRT	Length and width product dependence of PRT	0.00
NOIA	Noise parameter A (PMOS=9.9E18)	1E20
NOIB	Noise parameter B (PMOS=2.4E3)	5E4
NOIC	Noise parameter C (PMOS=1.4E-12)	-1.4E-12
EM	Saturation field	4.1E7
EF	Frequency exponent for noise model = 2	1.00
TOX	Gate oxide thickness	1.5E-8
TOXM	Gate oxide thickness in meters	0.00
XJ	Junction depth	1.5E-7
LXJ	Length dependence of XJ	0.00
WXJ	Width dependence of XJ	0.00
PXJ	Length and width product dependence of XJ	0.00
GAMMA1	Body effect coefficient near the interface	0.00
LGAMMA1	Length dependence of GAMMA1	0.00
WGAMMA1	Width dependence of GAMMA1	0.00
PGAMMA1	Length and width product dependence of GAMMA1	0.00
GAMMA2	Body effect coefficient in the bulk	0.00
LGAMMA2	Length dependence of GAMMA2	0.00
WGAMMA2	Width dependence of GAMMA2	0.00
PGAMMA2	Length and width product dependence of GAMMA2	0.00
NPEAK	Channel doping concentration	0.00
LNPEAK	Length dependence of NPEAK	0.00
WNPEAK	Width dependence of NPEAK	0.00
PNPEAK	Length and width product dependence of NPEAK	0.00
NSUB	Substrate doping concentration	6E16
LNSUB	Length dependence of NSUB	0.00

Model parameters for level 8 (continued)

Name	Parameter	Default
WNSUB	Width dependence of NSUB	0.00
PNSUB	Length and width product dependence of NSUB	0.00
VBX	VBS at which the depletion width equals XT	0.00
LVBX	Length dependence of VBX	0.00
WVBX	Width dependence of VBX	0.00
PVBX	Length and width product dependence of VBX	0.00
VBM	Maximum applied body bias in Vth calculation	0.00
LVBM	Length dependence of VBM	0.00
WVBM	Width dependence of VBM	0.00
PVBM	Length and width product dependence of VBM	0.00
XT	Doping depth	0.00
LXT	Length dependence of XT	0.00
WXT	Width dependence of XT	0.00
PXT	Length and width product dependence of XT	0.00
JSSW	Bulk p-n sidewall saturation current density	0.00

Noise equations

Noise models for the BSIM3 device are defined in reference 22 at the beginning of this chapter and are used when NLEV is undefined. Noise models for all cases are as follows:

Parasitic lead resistor thermal noise

$I^2 = 4 \cdot k \cdot T / RG$

$I^2_{rd} = 4 \cdot k \cdot T / RD$

$I^2_{rs} = 4 \cdot k \cdot T / RS$

$I^2_{rb} = 4 \cdot k \cdot T / RB$

Channel and shot flicker noise

$Ichannel^2 = Ishot^2 + Iflicker^2$

Intrinsic flicker noise:

If NLEV = 0 $\quad Iflicker^2 = KF \cdot Idrain^{AF} / (COX \cdot Leff^2 \cdot f)$

If NLEV = 1 $\quad Iflicker^2 = KF \cdot Idrain^{AF} / (COX \cdot Weff \cdot Leff \cdot f)$

If NLEV = 2 or 3 $\quad Iflicker^2 = KF \cdot Idrain^{AF} / (COX \cdot Weff \cdot Leff \cdot f^{AF})$

Intrinsic shot noise:

If NLEV < 3 $\quad Ishot^2 = (8/3) \cdot k \cdot T \cdot gm$

If NLEV = 3 $\quad Ishot^2 = GDSNOI \cdot (8/3) \cdot k \cdot T \cdot gm \cdot beta \cdot (Vgs - Vth) \cdot (1+a+a^2)/(1+a)$

a = 1 - Vds/Vdsat if Vds <= Vdsat (linear region) and a = 0 elsewhere.

OPAMP

Schematic format

PART attribute
<*name*>

Examples
OP1

MODEL attribute
<*model name*>

Example
LF351

There are three model levels for the OPAMP. Each succeeding level provides increasingly more realistic models at the expense of increasingly more complex equivalent circuits.

Level 1 is a simple voltage-controlled current source with a finite output resistance and open loop gain.

Level 2 is a three stage, two pole model with slew rate limiting, finite gain, and output resistance.

Level 3 is an enhanced Boyle model, similar to those implemented in other SPICE programs as subcircuits. It is not, however, a macro or a subcircuit, but a fully internal MC7 device model. It models positive and negative slew rates, finite gain, AC and DC output resistance, input offset voltage and current, phase margin, common mode rejection, unity gain bandwidth, three types of differential input, and realistic output voltage and current limiting.

Model statement forms

.MODEL <*model name*> OPA ([*model parameters*])

Examples
.MODEL LM709 OPA (A=45K VOFF=.001 SRP=250K GBW=1E6)
.MODEL LF155 OPA (LEVEL=2 TYPE=1 A=50K SRP=330K)

Model parameters

Name	Parameter	Units	Def.	Level
LEVEL	Model level(1, 2, 3)		1	1,2,3
TYPE	1=NPN 2=PNP 3=JFET		1	3
C	Compensation capacitance	F	30E-12	3
A	DC open-loop gain		2E5	1,2,3
ROUTAC	AC output resistance	Ω	75	1,2,3
ROUTDC	DC output resistance	Ω	125	1,2,3
VOFF	Input offset voltage	V	0.001	3
IOFF	Input offset current	A	1E-9	3
SRP	Maximum positive slew rate	V/S	5E5	2,3
SRN	Maximum negative slew rate	V/S	5E5	2,3
IBIAS	Input bias current	A	1E-7	3
VCC	Positive power supply	V	15	3
VEE	Negative power supply	V	-15	3
VPS	Maximum pos. voltage swing	V	13	3
VNS	Maximum neg. voltage swing	V	-13	3
CMRR	Common-mode rejection ratio		1E5	3
GBW	Gain bandwidth		1E6	2,3
PM	Phase margin	deg.	60	2,3
PD	Power dissipation	W	.025	3
IOSC	Short circuit current	A	.02	3
T_MEASURED	Measured temperature	°C		
T_ABS	Absolute temperature	°C		
T_REL_GLOBAL	Relative to current temp.	°C		
T_REL_LOCAL	Relative to AKO temperature	°C		

VCC and VEE are the nominal power supply values at which VPS and VNS are specified. It is possible to operate the OPAMP at other supply voltages. VEE and VCC affect only power dissipation and the output saturation characteristics.

Model schematic and equations:

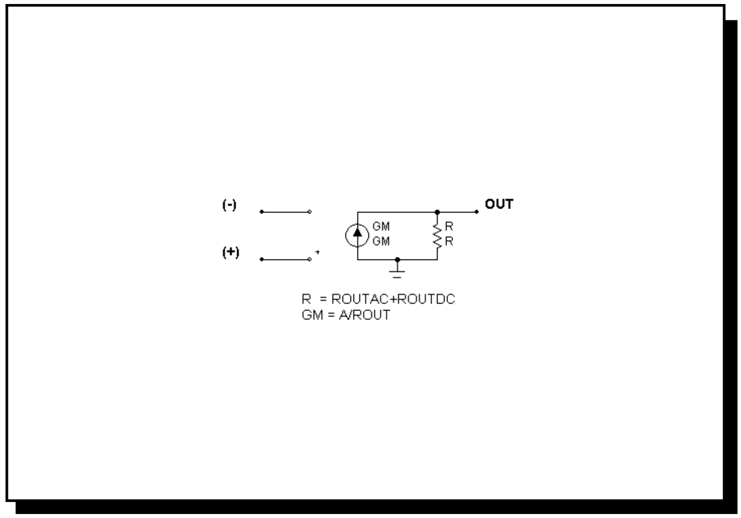


Figure 22-13 The level 1 opamp model

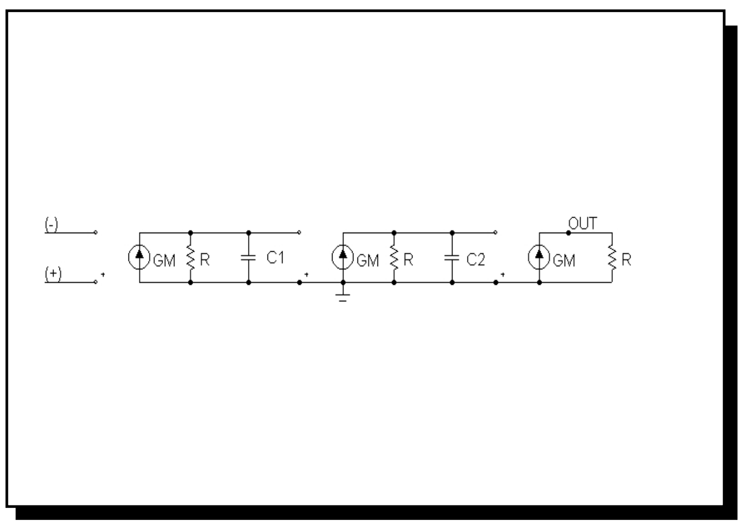


Figure 22-14 The level 2 opamp model

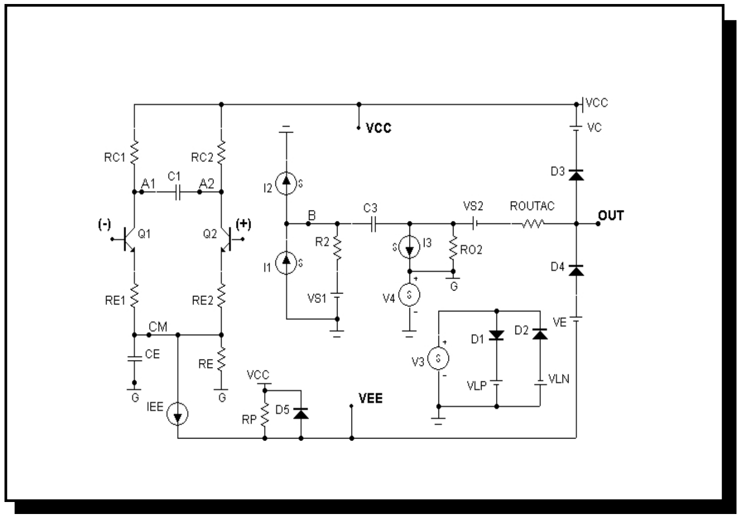


Figure 22-15 The level 3 opamp model with NPN inputs

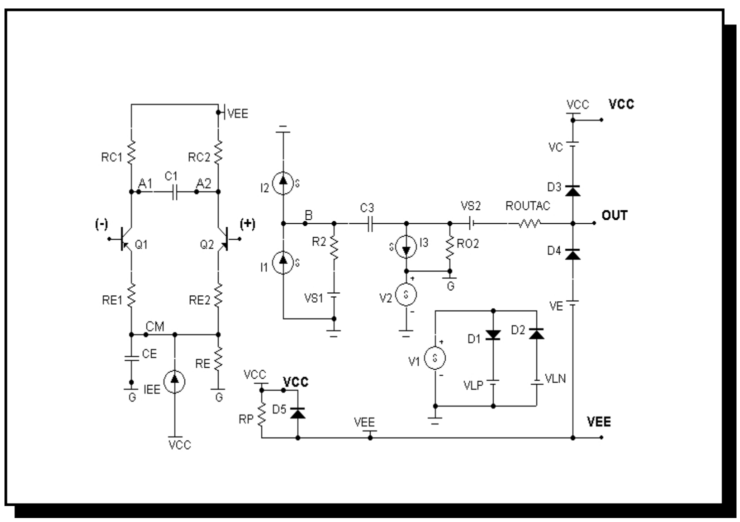


Figure 22-16 The level 3 opamp model with PNP inputs

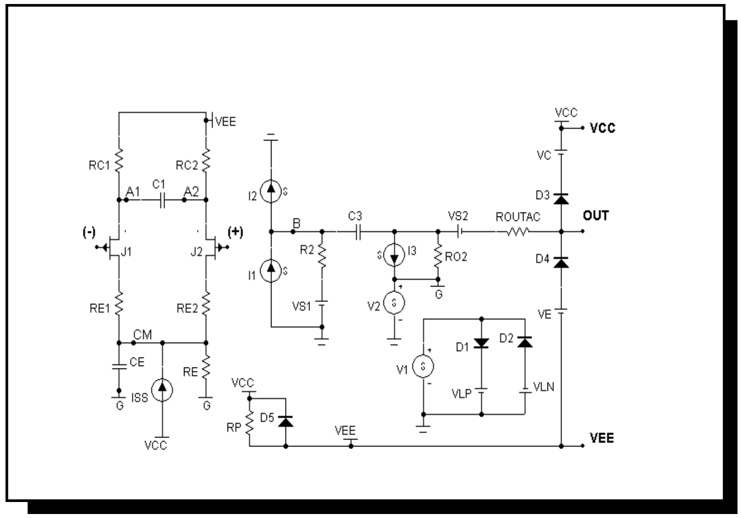


Figure 22-17 The level 3 opamp model with JFET inputs

Definitions

T	Junction temperature in degrees Kelvin
VT	k·T/q
BETA1	Forward beta of Q1
BETA2	Forward beta of Q2
BJT1IS	Saturation current (IS) of Q1
BJT2IS	Saturation current (IS) of Q2
V(A1)	Voltage at node A1
V(A2)	Voltage at node A2
V(CM)	Voltage at node CM
I(VS1)	Current through source VS1
I(VC)	Current through source VC
I(VE)	Current through source VE
I(VLP)	Current through source VLP
I(VLN)	Current through source VLN
V(VCC)	Voltage across source VCC
V(VEE)	Voltage across source VEE
I(VS2)	Current through source VS2
I(GA)	Current of source GA
I(GCM)	Current of source GCM
I(F1)	Current of source F1
V(E1)	Voltage of source E1
V(H1)	Voltage of source H1

Temperature effects

Temperature affects diodes, BJTs, and JFETs in the usual way as described in the model sections for these devices. Default temperature parameters are used.

Level 1 equations

R = ROUTAC+ROUTDC

GM = A/R

Level 2 equations

R = ROUTAC+ROUTDC

GM = $A^{1/3}$/ R

F1 = First pole = GBW/A

U = .5·A·Sin(PM)

V = (1+A·Cos(PM))/(U·U)

F0 = Unity gain frequency = F1·U·(1+sqrt(1-V))

F2 = Second pole = F0/(U·(1-sqrt(1-V)))

C1 = 1/(2·π·F1·R)

C2 = 1/(2·π·F2·R)

Level 3 equations

RC1 = 1/(2·π·GBW·C2)

RC2 = RC1

F1 = First pole = GBW/A

U = .5·A·Sin(PM)

V = (1+A·Cos(PM))/(U·U)

F0 = Unity gain frequency = F1·U·(1+sqrt(1-V))

F2 = Second pole = F0/(U·(1-sqrt(1-V)))

C1 = 0.5/(2·π·F2·RC1)

C2 = C

R2 = 1E5

GA = 1/RC1

VAF = 200

NPN and PNP input stages

NPN input

IC1 = SRP·C2/2

CE = 2·IC1/SRN-C2

PNP input

IC1 = SRN·C2/2

CE = 2·IC1/SRP-C2

BETA1 = IC1/(IBIAS+IOFF/2)

$BETA2 = IC1/(IBIAS-IOFF/2)$
$IEE = (((BETA1+1)/BETA1)+((BETA2+1)/BETA2))\cdot IC1$
$RE1 = ((BETA1+BETA2)/(BETA1+BETA2+2))\cdot(RC1-VT/IC1)$
$RE2 = RE1$
$RP = (|VCC|+|VEE|)^2/(PD-|VCC|\cdot 2\cdot IC1-|VEE|\cdot IEE)$
$RE = VAF/IEE$
$BJT1IS = 1E-16$
$BJT2IS = BJT1IS\cdot(1+VOFF/VT)$

JFET input stage

$IC1 = SRP\cdot C2/2$
$IEE = C2\cdot SRN$
$CE = C2\cdot(SRN/SRP-1)$
$RE = VAF/IEE$
$RE1 = 1$
$RE2 = 1$
$BETA1 = 0.5\cdot GA^2/IEE$
$BETA2 = BETA1$
$RP = (|VCC|+|VEE|)^2/PD$

$RO2 = ROUTDC - ROUTAC$
$GCM = 1/(CMRR\cdot RC1)$
$GB = RC1\cdot A/RO2$
$VLP = IOSC\cdot 1000$
$VLN = VLP$
$VC = VCC - VPS$
$VE = -VEE + VNS$

Controlled source equations

$I(GA) = GA\cdot(V(A1)-V(A2))$
$I(GCM)= GCM\cdot V(CM)$
$I(F1) = GB\cdot I(VS1)-GB\cdot I(VC)+GB\cdot I(VE)+GB\cdot I(VLP)-GB\cdot I(VLN)$
$V(E1) = (V(VCC)+V(VEE))/2$
$V(H1) = 1000\cdot(I(VS2))$
$V(VS1)= 0.0$ (Only used to measure current)
$V(VS2)= 0.0$ (Only used to measure current)

Note that the Level 2 and 3 models use Gain Bandwidth (GBW), not Unity Gain Frequency (F0), as an input parameter. They correctly produce both the specified phase margin and unity gain at the frequency F0. In general, F0 < GBW. This is an enhancement to the standard Boyle model which produces the specified phase margin but not unity gain at the frequency F0.

Pulse source

Schematic format

PART attribute
<*name*>

Examples
P1

MODEL attribute
<*model name*>

Example
RAMP

The PULSE source is similar to the SPICE PULSE independent voltage source, except that it uses a model statement and its timing values are defined with respect to T=0.

Model statement forms

.MODEL <*model name*> PUL ([*model parameters*])

Example
.MODEL STEP PUL (VZERO=.5 VONE=4.5 P1=10n P2=20n P3=100n
+ P4=110n P5=500n)

Model parameters

Name	**Parameter**	**Units**	**Default**
VZERO	Zero level	V	0.0
VONE	One level	V	5.0
P1	Time delay to leading edge	S	1.0E-7
P2	Time delay to one-level	S	1.1E-7
P3	Time delay to trailing edge	S	5.0E-7
P4	Time delay to zero level	S	5.1E-7
P5	Repetition period	S	1.0E-6

Equations

The waveform value is generated as follows:

From	To	Value
0	P1	VZERO
P1	P2	VZERO+((VONE-VZERO)/(P2-P1))·(T-P1)
P2	P3	VONE
P3	P4	VONE+((VZERO-VONE)/(P4-P3))·(T-P3)
P4	P5	VZERO

where From and To are T values, and T=TIME mod P5. The waveform repeats every P5 seconds. Note that P5 ≥ P4 ≥ P3 ≥ P2 ≥ P1.

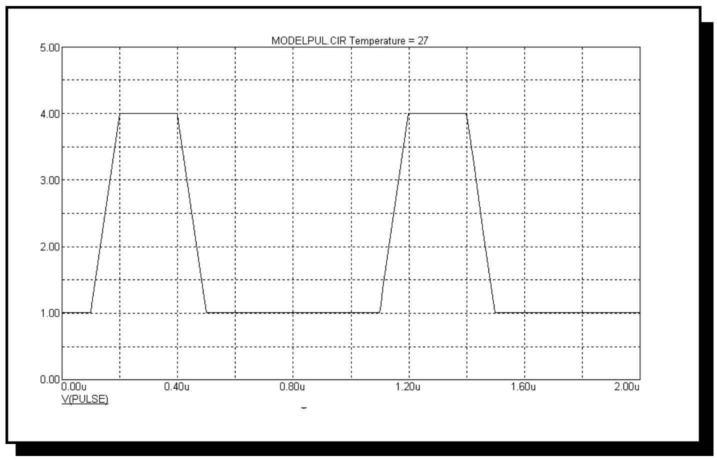


Figure 22-18 Sample waveform for model parameters vzero=1 vone=4 P1=.1u P2=.2u P3=.4u P4=.5u P5=1u

Resistor

SPICE format

Syntax
R<*name*> <*plus*> <*minus*> [*model name*] <*value*> [TC=<*tc*1>[,<*tc*2>]]

Examples
R1 2 3 50
R2 7 8 10K

<*plus*> and <*minus*> are the positive and negative node numbers. The polarity references are used only for plotting or printing the voltage across, V(RX), and the current through, I(RX), the resistor.

Schematic format

PART attribute
<*name*>

Examples
R5
CARBON5

VALUE attribute
<*value*> [TC=<*tc*1>[,<*tc*2>]]

Examples
50
10K
50K*(1+V(6)/100)

FREQ attribute
<*fexpr*>

Examples
2K+10*(1+F/1e9)

MODEL attribute
[*model name*]

Example
RMOD

VALUE attribute

<*value*> may be a simple number or an expression involving time-domain variables. The expression is evaluated in the time domain only. Consider the expression:

100+V(10)*2

V(10) refers to the value of the voltage on node 10 during a transient analysis, a DC operating point calculation prior to an AC analysis, or during a DC analysis. It does not mean the AC small signal voltage on node 10. If the DC operating point value for node 10 was 2, the resistance would be evaluated as 100 + 2*2 = 104. The constant value, 104, is used in AC analysis.

FREQ attribute

If <*fexpr*> is used, it replaces the value determined during the operating point. <*fexpr*> may be a simple number or an expression involving frequency domain variables. The expression is evaluated during AC analysis as the frequency changes. For example, suppose the <*fexpr*> attribute is this:

V(4,5)*(1+F/1e7)

In this expression, F refers to the AC analysis frequency variable and V(4,5) refers to the AC small signal voltage between nodes 4 and 5. Note that there is no time-domain equivalent to <*fexpr*>. Even if <*fexpr*> is present, <*value*> will be used in transient analysis.

Stepping effects

The VALUE attribute and all of the model parameters may be stepped. If VALUE is stepped, it replaces <*value*>, even if <*value*> is an expression. The stepped value may be further modified by the temperature effect.

Temperature effects

There are two different temperature factors, a quadratic factor and an exponential factor. The quadratic factor is characterized by the model parameters TC1 and TC2, or <tc1> and <tc2> from the parameter line. The exponential factor is characterized by the model parameter TCE.

If [TC=<*tc*1>[,<*tc*2>]] is specified on the parameter line, <*value*> is multiplied by a temperature factor, TF.

$$TF = 1+\langle tc1\rangle\cdot(T-Tnom)+\langle tc2\rangle\cdot(T-Tnom)^2$$

If [*model name*] is used and TCE is not specified, <*value*> is multiplied by a temperature factor, TF.

$$TF = 1+TC1\cdot(T\text{-}Tnom)+TC2\cdot(T\text{-}Tnom)^2$$

TC1 is the linear temperature coefficient and is sometimes given in data sheets as parts per million per degree C. To convert ppm specs to TC1 divide by 1E6. For example, a spec of 3000 ppm/degree C would produce a TC1 value of 3E-3.

If [*model name*] is used and TCE is specified, <*value*> is multiplied by a temperature factor, TF.

$$TF = 1.01^{TCE\cdot(T\text{-}Tnom)}$$

If both [*model name*] and [TC=<*tc*1>[,<*tc*2>]] are specified, [TC=<*tc*1>[,<*tc*2>]] takes precedence.

T is the device operating temperature and Tnom is the temperature at which the nominal resistance was measured. T is set to the analysis temperature from the Analysis Limits dialog box. TNOM is determined by the Global Settings TNOM value, which can be overridden with a .OPTIONS statement. T and Tnom may be changed for each model by specifying values for T_MEASURED, T_ABS, T_REL_GLOBAL, and T_REL_LOCAL. See the .MODEL command for more on how device operating temperatures and Tnom temperatures are calculated.

Monte Carlo effects
LOT and DEV Monte Carlo tolerances, available only when [*model name*] is used, are obtained from the model statement. They are expressed as either a percentage or as an absolute value and are available for all of the model parameters except the T_parameters. Both forms are converted to an equivalent *tolerance percentage* and produce their effect by increasing or decreasing the Monte Carlo factor, MF, which ultimately multiplies the final value.

MF = 1 ± *tolerance percentage* /100

If *tolerance percentage* is zero or Monte Carlo is not in use, then the MF factor is set to 1.0 and has no effect on the final value.

The final resistance, *rvalue*, is calculated as follows:

rvalue = <*value*> * R * TF * MF

Model statement form

.MODEL <*model name*> RES ([*model parameters*])

Example

.MODEL RM RES (R=2.0 LOT=10% TC1=.015)

Model parameters

Name	**Parameter**	**Units**	**Default**
R	Resistance multiplier		1.0
TC1	Linear temperature coefficient	$°C^{-1}$	0.0
TC2	Quadratic temperature coefficient	$°C^{-2}$	0.0
TCE	Exponential temperature coefficient	%/°C	0.0
NM	Noise multiplier		1.0
T_MEASURED	Measured temperature	°C	
T_ABS	Absolute temperature	°C	
T_REL_GLOBAL	Relative to current temperature	°C	
T_REL_LOCAL	Relative to AKO temperature	°C	

Noise effects

Noise in resistors is due to the random thermal motion of the carriers and is not affected by the presence of direct DC current as in shot noise. It is modeled with an ideal thermal noise current calculated as follows:

I = NM * sqrt(4*K*T/*rvalue*)...(per unit bandwidth)

NM, multiplies the resistor noise current. A value of zero effectively eliminates the noise contribution from all resistors that use the model.

S_Port (S-Parameter two-port)

Schematic format

PART attribute
<*name*>

Example
SP1

FILE attribute
<*file name*>

The FILE attribute specifies the path and name of the S-parameter file.

Example
E:\mc7\data\Gg10v20m.s2p

The S_PORT source provides a way of modeling devices using S-parameters. Typically these are provided by RF suppliers in a text file as a table of values for S11, S12, S21, and S22. The file typically looks like this:

```
! SIEMENS Small Signal Semiconductors
! BFG194
! Si PNP RF Bipolar Junction Transistor in SOT223
! VCE = -10 V     IC = -20 mA
! Common Emitter S-Parameters:                    August  1996
# GHz  S  MA  R  50
!  f        S11           S21           S12           S22
! GHz     MAG   ANG      MAG   ANG      MAG   ANG      MAG   ANG
 0.010  0.3302  -25.4  35.370  169.9  0.0053   85.3  0.9077  -10.0
 0.020  0.3471  -48.2  33.679  161.6  0.0108   77.5  0.8815  -19.8
 0.050  0.4525  -95.0  27.726  139.2  0.0226   61.4  0.7258  -43.7
 0.100  0.5462 -131.5  19.023  118.7  0.0332   52.2  0.5077  -68.7
 0.150  0.5723 -149.4  13.754  106.4  0.0394   49.1  0.3795  -84.8
 0.200  0.5925 -159.8  10.787   99.1  0.0443   50.1  0.3068  -95.0
 0.250  0.6023 -167.0   8.757   93.4  0.0497   51.2  0.2581 -104.8
 0.300  0.6089 -172.2   7.393   89.0  0.0552   52.4  0.2298 -112.2
 0.400  0.6166  179.7   5.617   82.1  0.0661   54.2  0.1930 -125.5
...
```

MC7 converts the S-parameters to Y-parameters and then implements the S_PORT source as a set of four Laplace table sources. The Y-parameters are calculated directly from the S-parameters using the following standard formulas (See "Microwave Circuit Design", by Vendelin, Pavio, and Rhoda page 16).

D = ((1+S11)*(1+S22)-S12*S21)

y11 = ((1-S11)*(1+S22)+S12*S21) / D

y12 = -2*S12 / D

y21 = -2*S21 / D

y22 = ((1+S11)*(1-S22)+S12*S21) / D

The S_PORT model looks like this:

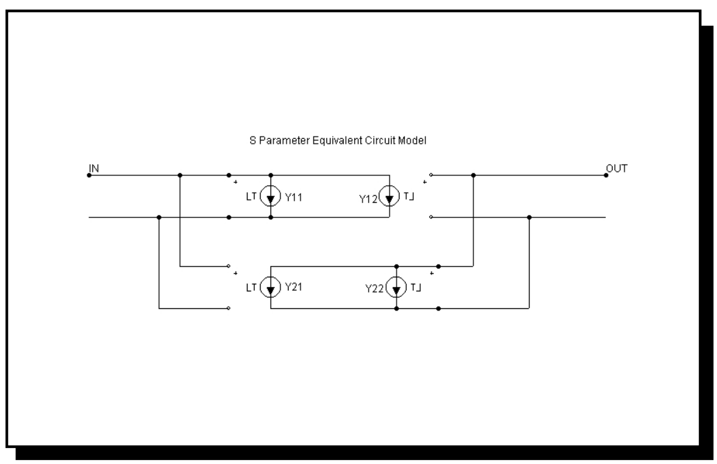


Figure 22-19 S_PORT Equivalent Model

S (Voltage-controlled switch)

SPICE format

S<*name*> <*plus output node*> <*minus output node*>
+<*plus controlling node*> <*minus controlling node*>
+<*model name*>

Example
S1 10 20 30 40 RELAMOD

Schematic format

PART attribute
<*name*>

Example
S1

MODEL attribute
<*model name*>

Example
RELAY

This four-terminal voltage-controlled switch is controlled by the voltage across the two input nodes. The switch impedance is calculated from the input voltage and impressed across the output nodes.

RON and ROFF must be greater than zero and less than 1/Gmin.

A 1/Gmin resistor is placed between the controlling nodes to avoid floating nodes.

Do not make the ratio ROFF/RON larger than about 15 decades. The 15 digits of precision used by the simulator can not make meaningful use of a spread greater than 15 in the ratio.

Do not make the transition region, VON-VOFF, too small as this will cause an excessive number of time points required to cross the region. The smallest allowed values for VON-VOFF is (RELTOL·(max(VON,VOFF))+VNTOL).

Model statement forms

.MODEL <*model name*> VSWITCH ([*model parameters*])

Example
.MODEL S1 VSWITCH (RON=1 ROFF=1K VON=1 VOFF=1.5)

Model parameters

Name	Parameter	Units	Default
RON	On resistance	Ohms	1
ROFF	Off resistance	Ohms	1E6
VON	Control voltage for On state	V	1
VOFF	Control voltage for Off state	V	0

Model Equations

VC = Voltage across the control nodes
LM = Log-mean of resistor values = ln((RON·ROFF)$^{1/2}$)
LR = Log-ratio of resistor values = ln(RON/ROFF)
VM= Mean of control voltages = (VON+VOFF)/2
VD = Difference of control voltages = VON-VOFF
k = Boltzmann's constant
T = Analysis temperature
RS = Switch output resistance

If VON > VOFF
If VC >= VON
RS = RON
If VC <= VOFF
RS = ROFF
If VOFF < VC < VON
$RS = \exp(LM + 3 \cdot LR \cdot (VC-VM)/(2 \cdot VD) - 2 \cdot LR \cdot (VC-VM)^3/VD^3)$

If VON < VOFF
If VC <= VON
RS = RON
If VC >= VOFF
RS = ROFF
If VOFF > VC > VON
$RS = \exp(LM - 3 \cdot LR \cdot (VC-VM)/(2 \cdot VD) + 2 \cdot LR \cdot (VC-VM)^3/VD^3)$

Noise effects

Noise is modeled as a resistor equal to the resistance found during the DC operating point. The thermal noise current is calculated as follows:

$I = \mathrm{sqrt}(4 \cdot k \cdot T/RS)$

Sample and hold source

SPICE format

There is no equivalent Sample and Hold device in SPICE or PSpice.

Schematic format

PART attribute
<*name*>

Examples
S1
S10
SA

INPUT EXPR attribute
<*input expression*>

Examples
V(1,2)
V(10,20)*I(R1)
V(INPUT)

SAMPLE EXPR attribute
<*sample expression*>

Examples
V(1,2)>1.2
V(5)>1.1 AND V(4) >1.2
I(RL)>1e-3

PERIOD attribute
<*sampling_period*>

Examples
100ns
tmax/100
1U

This device is an ideal sample and hold. It samples <*input expression*> when <*sampling_period*> is true *or* every sample period seconds. The behavioral modes are distinguished as follows:

If <*sample expression*> is specified:
<*input expression*> is sampled whenever <*sample expression*> is >= 1.0. <*sample expression*> is normally a Boolean expression. The output voltage of the source is set to the sampled value and remains constant until the next sample.

If <*sample expression*> is unspecified and <*sampling_period*> is specified:
The source samples and stores the numeric value of <*input expression*> once every <*sampling period*> seconds, starting at <*tmin*>. The output voltage of the source is set to the sampled value and remains constant until the next sample.

Neither <*sample expression*> nor <*sampling_period*> is specified:
This will generate an error message. Either <*sample expression*> or <*sampling_period*> must be specified.

Both <*sample expression*> and <*sampling_period*> are specified:
<*input expression*> is sampled whenever <*sample expression*> is >= 1.0.
This is the same as the first case. The <*sampling_period*> is ignored.

Note that while <*input expression*> is usually a node voltage, the expression can involve circuit variables, as in the second example. <*sampling_period*> may include a non-time varying expression, as in the second example, where the <*tmax*> of the analysis run is used to calculate the sampling period.

See the sample file SH2.CIR for an example of how to use the sample and hold component.

Sine source

Schematic format

PART attribute
<*name*>

Examples
S1

MODEL attribute
<*model name*>

Example
Line60

The Sine source is similar to the SPICE SIN independent voltage source. Unlike the SPICE source, it uses a model statement.

Model statement form

.MODEL <*model name*> SIN ([*model parameters*])

Example
.MODEL V1 SIN (F=1Meg A=0.6 DC=1.5)

Model parameters

Name	Parameter	Units	Default
F	Frequency	Hz	1E6
A	Amplitude	V	1.0
DC	DC level	V	0.0
PH	Phase shift	Radians	0.0
RS	Source resistance	Ω	0.001
RP	Repetition period of exponential	S	0.00
TAU	Exponential time constant	S	0.00

Model Equations

If TAU = 0 then

$$V = A\cdot\sin(2\cdot\pi\cdot F\cdot TIME + PH) + DC$$

Else

$$V = A\cdot e^{(-T/TAU)}\cdot\sin(2\cdot\pi\cdot F\cdot TIME + PH) + DC$$

where T = TIME mod RP.

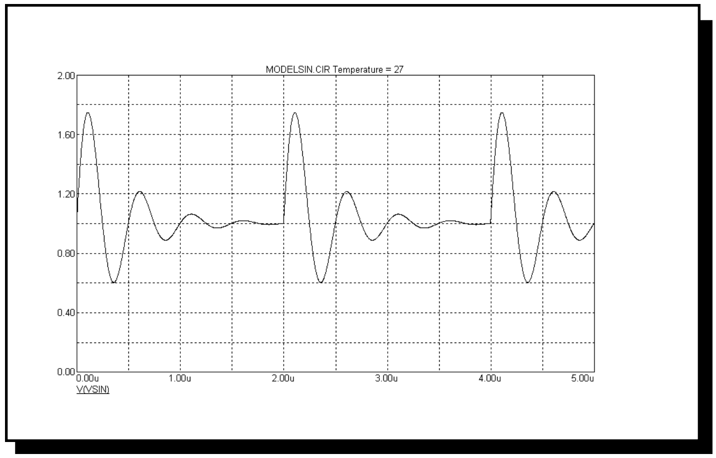


Figure 22-20 Sample waveform for model parameters F=2Meg A=1 DC=1 RP=2U TAU=.4U

Subcircuit call

SPICE format

X<*name*> [*node*]* <*subcircuit name*>
+ [PARAMS: <<*parameter name*>=<*parameter value*>>*]
+ [TEXT: <<*text name*>=<*text value*>>*]

Examples
X1 10 20 AMP
XDIFF 100 200 DIFF PARAMS: GAIN=10

Schematic format

PART attribute
<*name*>

Example
X1

NAME attribute
<*subcircuit name*>

Example
FILTER

FILE attribute
[<*file name*>]

Example
MYFILE.MOD

PARAMS: attribute
[<<*parameter name*>=<*parameter value*>>*]

Example
CENTER=10KHZ BW=1KHZ

TEXT: attribute
[<<*text name*>=<*text value*>>*]

Example
JEDEC="FILENAME"

[*node*]* are the numbers or names of the nodes specified in the .SUBCKT statement. The number of nodes in the subcircuit call must be the same as the number of nodes in the .SUBCKT statement. When the subcircuit is called, the nodes in the call are substituted for the nodes in the body of the subcircuit in the same order as they are listed in the .SUBCKT statement.

Any nodes defined in the .SUBCKT statement with the OPTIONAL: keyword may optionally follow the [*node*]* values. A subset of the optional nodes may be listed, and will be assigned to the internal subckt nodes in the order specified by the OPTIONAL statement. If you skip nodes, they must be skipped from the end of the list. OPTIONAL nodes are available only in a SPICE text circuit subcircuit call. Schematic subcircuits expect the exact number of pins defined for the subcircuit in the Component library. That is why there is no OPTIONAL attribute in the schematic format. This feature is used in the SPICE text file Digital Library to allow specifying optional power supply nodes.

The SPICE <*subcircuit name*> or schematic NAME attribute defines the subcircuit name. It must be the same as the name in the defining .SUBCKT statement.

The schematic FILE attribute defines the name of the file in which the .SUBCKT statement can be found. MC7 looks for the .SUBCKT statement in the following places in the order indicated.

- If the circuit is a schematic:
  - In the Text area.
  - In the file named in the FILE attribute.
  - In one or more files named in a '.LIB' statement.
  - In one or more files named in the default '.LIB NOM.LIB' statement.

- If the circuit is a SPICE text file:
  - In the circuit description text.
  - In one or more files named in a '.LIB' statement.
  - In one or more files named in the default '.LIB NOM.LIB' statement.

Subcircuits are used extensively in the Analog Library and Digital Library sections of the Component library. They are accessed via the default .LIB statement.

The SPICE keyword or schematic PARAMS: attribute lets you pass multiple numeric parameters to the subcircuit. <parameter name> is its name and <parameter value> defines the value it will assume if the parameter is not included in the subcircuit call. For example:

```
.SUBCKT CLIP 1 2
+ PARAMS: LOW=0 HIGH=10
```

Any of these calls are legal:

```
X1 10 20 CLIP                               ;RESULTS IN LOW=0, HIGH=10
X2 10 20 CLIP PARAMS: LOW=1 HIGH=2          ;RESULTS IN LOW=1, HIGH=2
X3 10 20 CLIP PARAMS: HIGH=4                ;RESULTS IN LOW=0, HIGH=4
```

The SPICE keyword or schematic TEXT: attribute lets you pass text parameters to the subcircuit. <*text name*> is the name of the text parameter and <*text value*> defines the value it will assume if the parameter is not included in the subcircuit call. For example:

```
.SUBCKT STIMULUS 1 2 3 4
+ TEXT: FILE="T1.STM"
```

Either of these calls are legal:

```
X1 10 20 30 40 STIMULUS                        ;RESULTS IN FILE="T1.STM"
X2 10 20 30 40 STIMULUS TEXT:FILE="P.STM"      ;RESULTS IN FILE="P.STM"
```

Using the subckt component in an MC7 schematic circuit file is easy. First, you must enter the subcircuit into the Component library using the Component editor. This requires entering:

- **Subckt Name:** Use any unique name. To avoid confusion, it should be the same as the name used in the SUBCKT control statement, although this is not strictly necessary.

- **Shape Name:** Use any suitable shape.

- **Definition:** Use SUBCKT.

Once these items have been entered, the pin assignments must be defined. These define where the node numbers called out in the SUBCKT control statement go on the shape. Pins are assigned by clicking in the Shape drawing area and naming the pin with the subckt node number. The pin is then dragged to the desired position on the shape.

The subckt is then placed in the schematic in the usual way.

To see how subckts are called, see the sample circuit files SUBCKT1 and PLA2.

Switch

Schematic format

PART attribute
<*name*>

Examples
S1

VALUE attribute
<[V | T | I]>,<*n1,n2*> [,<*ron*>[,<*roff*>]]

Examples
V,1,2
I,2ma,3ma
T,1ms,2ms,50,5Meg

This is the oldest of three types of switches. The newer S and W switches exhibit a slower, but smoother transition between the off and on states.

There are three types of dependent switches; current-controlled, voltage-controlled, and time-controlled.

The switch is a four-terminal device. A current sensing inductor must be connected across the two input nodes when the current-controlled switch option is used. Voltage-controlled switches are controlled by the voltage across the two input nodes. Time dependent switches use the transient analysis time variable to control the opening and closing of the switch.

The two controlling nodes for a time-controlled switch are not used and may be shorted to ground or to the two output nodes to reduce the node count by two.

In transient analysis, care must be taken in choosing the time step of the simulation. If the time step is too large, the switch might never turn on. There must be at least one time point within the specified window for the switch to close or open.

The switch parameter syntax provides for a normally-on or a normally-off switch. A normally-on switch is one that is on outside the specified window and off inside the window. A normally-off switch is one that is off outside the window and on inside the window.

The optional <*ron*> impedance defaults to 1E-3 ohm. The optional <*roff*> defaults to 1E9 ohms. When specifying <*roff*>, be careful that off switch resistances are not so low that they load your circuit, nor so high that they generate excessive voltage by blocking the current from an inductor or a current source.

Rules of operation

If n1 < n2 then
    Switch is closed (ON) when
        n1 ≤ X ≤ n2
    Switch is open (OFF) when
        X < n1 or X > n2

If n1 > n2 then
    Switch is open (OFF) when
        n1 ≥ X ≥ n2
    Switch is closed (ON) when
        X > n1 or X < n2

If the first character in the switch parameter is V:
    The switch is a voltage-controlled switch.
    X = voltage across the input nodes.
    n1 and n2 are voltage values.

If the first character in the switch parameter is I:
    The switch is a current-controlled switch.
    X = current through the inductor placed across the input nodes.
    n1 and n2 are current values.

If the first character in the switch parameter is T:
    The switch is a time-controlled switch.
    X = TIME.
    n1 and n2 are values of the TIME variable.

If the switch is closed, the switch has a resistance of <*ron*>. If the switch is open, the switch has a resistance of <*roff*>.

Transformer

Schematic format

PART attribute
<*name*>

Example
T1

VALUE attribute
<*primary inductance*>,<*secondary inductance*>,<*coupling coefficient*>

Example
.01,.0001,.98

The transformer component consists of two inductors with a mutual inductance between them. It is entirely equivalent to two discrete inductors and a linear K device defining a mutual inductance between them.

The coefficient of coupling is related to the mutual inductance by the following equation.

$$k = M / (LP*LS)^{0.5}$$

M is the mutual inductance.
k is the coefficient of coupling and $0 < k < 1$.
LP is the primary inductance.
LS is the secondary inductance.

Transmission line

SPICE format

Ideal Line
T<*name*> <*A port + node*> <*A port - node*>
+<*B port + node*> <*B port - node*>
+[*model name*]
+ Z0=<*value*> [TD=<*value*>] | [F=<*value*> [NL=<*value*>]]

Example
T1 10 20 30 40 Z0=50 TD=3.5ns
T1 10 20 30 40 Z0=150 F=125Meg NL=0.5
T1 20 30 40 50 TLMODEL

Lossy Line
T<*name*> <*A port + node*> <*A port - node*>
+<*B port + node*> <*B port - node*>
+[<*model name*> [*physical length*]]
+ LEN=<*len value*> R=<*rvalue*> L=<*lvalue*> G=<*gvalue*>
+ C=<*cvalue*>

Examples
T1 20 30 40 50 LEN=1 R=.5 L=.8U C=56PF
T3 1 2 3 4 TMODEL 12.0

Schematic format

PART attribute
<*name*>

Example
T1

VALUE attribute for ideal line
Z0=<*value*> [TD=<*value*>] | [F=<*value*> [NL=<*value*>]]

Example for ideal line
Z0=50 TD=3.5ns

VALUE attribute for lossy line
<*physical length*> LEN=<*len value*> R=<*rvalue*> L=<*lvalue*>
G=<*gvalue*> C=<*cvalue*>

Examples for lossy line
LEN=1 R=.5 L=.8U C=56PF
R=.5 L=.8U C=56PF

MODEL attribute
<*model name*>

Example
RELAY

Model statement form

.MODEL <*model name*> TRN ([*model parameters*])

Examples
.MODEL TIDEAL TRN(Z0=50 TD=10ns)
.MODEL TLOSS TRN(C=23pf L=13nh R=.35)

Model parameters for ideal line

Name	Parameter	Units	Default
Z0	Characteristic impedance	Ohms	None
TD	Transmission delay	S	None
F	Frequency for NL	Hz	None
NL	Relative wavelength		0.25

Model parameters for lossy line

Name	Parameter	Units	Default
R	Resistance per unit length	Ohms / unit	None
L	Inductance per unit length	Henries / unit	None
G	Conductance per unit length	Mhos / unit	None
C	Capacitance per unit length	Farads / unit	None
LEN	Physical length	Same as RLGC	None

LEN, R, L, G, and C must use a common unit of length. For example, if C is measured in units of farads/cm, then R must be in ohms/cm, L must be in henrys/cm, G must be in mhos/cm, and LEN must be in centimeters.

Model Equations

The ideal and lossy lines are represented by the model shown in Figure 22-20. The principal difference between the two models is in the implementation of the delay. In the ideal model, the delay is implemented as a linked-list of data pairs (time,value) and breakpoints.

The lossy line implementation of the delay is quite different. It uses the SPICE3 convolution method to represent the line, producing an accurate representation of a lossy line. The method employed is to convolve the input waveform with the

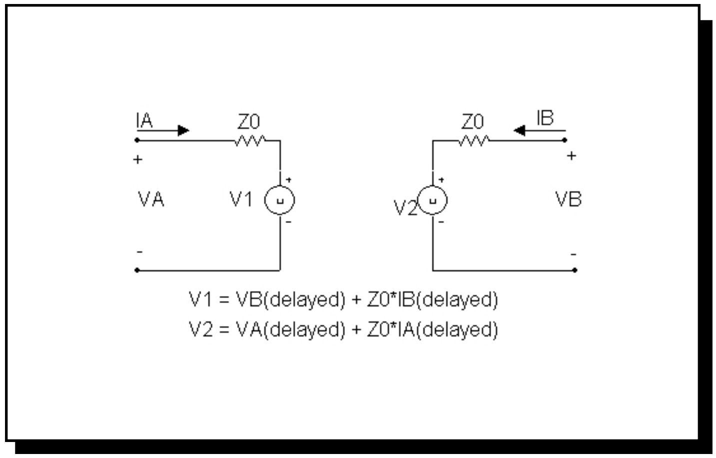


Figure 22-21 Transmission line model

impulse response of the lossy line to produce the output waveform. The impulse response is obtained from a presolved analytical formula. Presolved analytical solutions for the line impulse response are faster and more accurate than evaluating the inverse Fourier transform of the line's S plane representation. Convolution assures that any waveform can be handled, not just steps and linear ramps.

Note that only the RLC, RC, RG, and LC lines are supported. Nonzero values for R, L, C, and G specifying other line types will generate an error.

For both types of transmission lines, both VALUE and MODEL parameter descriptions may be given. VALUE parameters replace MODEL parameters and the final result is checked for adherence to the VALUE attribute syntax rules.

Note that either 'Z0' (Z -zero) or 'ZO' (Z-Oh) may be used in either the VALUE or MODEL parameters.

If the length modifier, <*physical length*>, is given, it overrides the model parameter LEN, if <*model name*> is used.

For examples of lossless transmission line circuits see the TL1, TL2, and TL3 circuits, and for a lossy transmission line circuit see the LTRA3 circuit.

User file source

Schematic format

PART attribute
<*name*>

Example
U1

FILE attribute
<*file name*>

Example
AMP.USR

EXPRESSION attribute
[*expression*]

Example
V(10)+V(11)

The User file source is a voltage source whose waveform or curve comes from an ASCII text file. The files contain a header and N sequential lines, each with a variable number of data values:

Transient analysis:
Two data values per line:Time, Y X set to T
Three data values per line:Time, X, Y

AC:
Three data values per line: Frequency, Real(Y), Imag(Y) X set to Frequency
Five data values per line: Frequency, Real(X), Imag(X), Real(Y), Imag(Y)

DC: Three data values per line:
DCINPUT1, X, Y

where X is the X expression value, Y is the Y expression value, and DCINPUT1 is the value of variable1.

User files may be created using a text editor, by external software, or by saving one or more curves or waveforms after an analysis run. To save a particular waveform, press F10 to invoke the Plot Properties dialog box after the analysis is over and select the waveform to be saved from the Save Curves section of the dialog box.

When you place a user source in a schematic, MC7 reads the current data directory to find any files with the extension *.USR. It then reads the files themselves to see what waveforms are available for use. It presents the files and waveforms as items in drop-down lists in the FILE and EXPRESSION fields so you can easily select the file name and waveform expression.

Curves saved in user files can be displayed directly in an analysis simply by selecting them from the Curves section of the Variables list. To invoke the Variables list, click in the Y Expression field and click the *right* mouse button. The X and Y parts of a waveform are stored as CurveX and CurveY, to let you select the X and Y parts independently.

See the files USER and USER2 for examples of how to use this type of source. Here is the SAMPLE.USR user file as an example of the format used:

```
[Main]
FileType=USR
Version=2.00
Program=Micro-Cap
[Menu]
;Simple    = T,X,Y
;SimpleNoX = T,Y
;Complex  = F,Xr,Xi,Yr,Yi
;ComplexNoX= F,Yr,Yi
;FormatType = Simple | Complex | SimpleNoX | ComplexNoX
;Format=FormatType
WaveformMenu=label vs T
[Waveform]
Label=label vs T
MainX=T
LabelX=T
LabelY=label vs T
Format=Simple
Data Point Count=256
0,0,0
2.352941176E-009,2.352941176E-009,0
4.705882353E-009,4.705882353E-009,0
…
```

W (Current-controlled switch)

SPICE format

W<*name*> <*plus output node*> <*minus output node*>
+<*controlling voltage source name*> <*model name*>

Example
W1 10 20 V1 IREF

Schematic format

PART attribute
<*name*>

Example
W1

REF attribute
<*controlling voltage source name*>

Example
VSENSE

MODEL attribute
<*model name*>

Example
SW

This two terminal device, current-controlled switch is controlled by the current through the source defined by <*controlling voltage source name*>. The model parameters RON and ROFF must be greater than zero and less than 1/Gmin.

Do not make the ratio ROFF/RON larger than about 15 decades. The 15 digits of precision used by the simulator can not make meaningful use of a spread greater than 15 in the ratio. Do not make the transition region, ION-IOFF, too small as this will cause an excessive number of time points required to cross the region. The smallest value for ION-IOFF is (RELTOL·(max(ION,IOFF))+ABSTOL).

Model statement form

.MODEL <*model name*> ISWITCH ([*model parameters*])

Example
.MODEL W1 ISWITCH (RON=1 ROFF=1K ION=1 IOFF=1.5)

Model parameters

Name	Parameter	Units	Default
RON	On resistance	Ohms	1
ROFF	Off resistance	Ohms	1E6
ION	Control current for On state	A	.001
IOFF	Control current for Off state	A	0

Model Equations

IC = Controlling current
LM = Log-mean of resistor values = $\ln((RON\cdot ROFF)^{1/2})$
LR = Log-ratio of resistor values = ln(RON/ROFF)
IM = Mean of control currents = (ION+IOFF)/2
ID = Difference of control currents = ION-IOFF
k = Boltzmann's constant
T = Analysis temperature
RS = Switch output resistance

If ION > IOFF
  If IC >= ION
    RS = RON
  If IC <= IOFF
    RS = ROFF
  If IOFF < IC < ION
    $RS = \exp(LM + 3\cdot LR\cdot(IC-IM)/(2\cdot ID) - 2\cdot LR\cdot(IC-IM)^3/ID^3)$

If ION < IOFF
  If IC <= ION
    RS = RON
  If IC >= IOFF
    RS = ROFF
  If IOFF > IC > ION
    $RS = \exp(LM - 3\cdot LR\cdot(IC-IM)/(2\cdot ID) + 2\cdot LR\cdot(IC-IM)^3/ID^3)$

Noise effects

Noise is modeled as a resistor equal to the resistance found during the DC operating point. The thermal noise current is calculated as follows:

$I= \mathrm{sqrt}(4\cdot k\cdot T/RS)$

Z transform source

Schematic format

PART attribute
<*name*>

Example
E1

ZEXP attribute
<*transform expression*>

Example
(.10*(Z+1)*(POW(Z,2)-.70*Z+1))/((Z-.56)*(POW(Z,2)-1.16*Z+.765))

CLOCK FREQUENCY attribute
<*clock frequency*>

Example
24Khz

The Z transform source is a voltage source whose waveform is expressed as a complex Z variable expression. It behaves like a Laplace source with Z replaced by EXP(S/<*clock frequency*>), where S = J * 2 * PI * F, and F = frequency.

See the sample file ZDOMAIN.CIR for an example of how to use the source.

Chapter 23 Digital Devices

What's in this chapter

This chapter describes the Micro-Cap digital simulator. It begins with a general description of the digital simulation engine, delay models, digital states and strengths and then describes each of the digital primitives. The primitives are the basic building blocks used to model standard commercial parts. They are used extensively as modeling elements in the Digital Library. They may also be used as primitives in schematics or SPICE circuits.

This chapter discusses:

- The digital simulation engine
- Digital nodes
- Digital states
- Digital strengths
- Timing models
- Propagation delay
- Digital primitives
- The Analog/digital interface
- Digital stimulus devices
- Digital input D/A interface device
- Digital output A/D interface device
- Standard gates
- Tri-state gates
- Flip-flops and latches
- Delay lines
- Pullup and pulldown devices
- Programmable logic arrays
- Multi-bit A/D and D/A converters
- Digital behavioral functions
    - Logic expressions
    - Pin delays
    - Constraint checkers
- Stimulus generators

The digital simulation engine

Micro-Cap includes a general purpose, event-driven, digital logic simulator. It is completely integrated within and time-synchronized with the analog simulator.

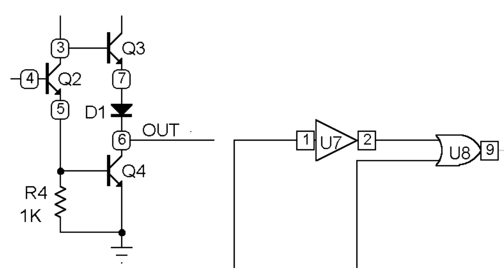
The primitives and behavior of a digital simulator can be defined in many ways. Micro-Cap uses the PSpice™ syntax since it is relatively efficient, widely known, and for many users, it means not having to relearn a new system. Micro-Cap and PSpice™ libraries can be read by either simulator, so portability between the two is improved as well. PSpice™ can't read Micro-Cap schematics, but it can read the Micro-Cap SPICE text circuit files. Micro-Cap can read most PSpice™ text circuit files. Most, but not all, features are supported.

Micro-Cap schematics and SPICE text file circuits may freely mix analog and digital circuits. The system automatically handles the interface between the analog and digital sections. Whenever analog and digital parts share the same node, MC7 automatically breaks the connection and inserts a user specified interface circuit between the two sections. The heart of the interface is an analog to digital interface translator (if the digital node was an input) or a digital to analog translator (if the digital node was an output). This automatic A/D interface is described in more detail later in the "Analog/digital interface" section of this chapter.

Digital nodes

Digital nodes are represented the same way as analog nodes. They may be referred to by node number or node *name.* Nodes are numbered automatically by MC7. Node names are assigned by the user by placing grid text on the node.

Digital node numbers are displayed on schematics inside sharp-cornered rectangles to reflect the angular look of digital waveforms. Analog node numbers use round-cornered rectangles to reflect the softer nature of analog waveforms.



Digital states

The state of a digital node is a combination of a digital level and a digital strength. Together the digital level and digital strength uniquely determine the digital state. The state is represented with one of the following symbols:

State	Description
0	Low
1	High
R	Rising (in transition from 0 to 1)
F	Falling (in transition from 1 to 0)
X	Unknown (level may be 0, 1, or unstable)
Z	High impedance (level may be 0, 1, R, F, X, or unstable)

Table 23-1 Digital states

Logic levels

These six symbols are used to describe what is happening on the digital nodes. The first five symbols, { 0, 1, R, F, X } describe logic levels at any strength greater than the high impedance strength. The 'Z' symbol describes any of the levels at the high impedance strength.

If a node state symbol is 'Z', then the strength of the output node is equal to the high impedance or 'Z' state. If the symbol is a '1', then the state is steady at a high level. If the symbol is a '0', then the state is steady at a low level. If the symbol is an 'R', then the state is rising to a '1'. If the symbol is an 'F', then the state is falling to a '0'. If the symbol is an 'X', then the state may be '0', '1', or unstable.

Logic levels correspond to a particular range of voltage, as defined in the I/O model statement. They do not correspond to a particular value of voltage. For instance, a '1' may be defined in the model statement as the voltage range 1.7 to 7.0 volts. If the digital node makes a transition between a '0' and a '1' then the implied analog voltage may be said to be "at least 1.7 volts", but it may be higher. Digital modeling involves an abstraction that trades information in exchange for simplicity and speed of simulation.

Logic strengths

When two or more digital outputs are connected together at a common node, the simulator determines the state of the common node as follows:

- If the levels from all outputs are the same, then:

  The new level is the common level and the strength is equal to the strength of the strongest node.

- If the levels from all outputs are not the same, then:

  If the strength of the strongest output exceeds the strength of the next strongest output by at least DIGOVRDRV, then the level and strength of the strongest output are assigned to the common node.

  Otherwise, the common node level is set to 'X' and the strength is set to the strength of the strongest output.

The weakest strength is called the high impedance or 'Z' strength. It is defined by the value of DIGDRVZ. Any output, tri-state or otherwise, will be set to the 'Z' state if its impedance is equal to or greater than DIGDRVZ.

The strongest strength, defined by DIGDRVF, is called the forcing impedance.

The strengths of a device's outputs are defined by the device's output impedance. The impedance is either DRVH or DRVL depending upon whether the output is high or low. DRVH and DRVL are obtained from the I/O model statement for the device. DRVH and DRVL are constrained to be in the following range:

$$\text{DIGDRVF} \leq \text{Impedance} \leq \text{DIGDRVZ}$$

This range defines a logarithmic scale. The scale includes 64 strengths that run from an impedance of DIGDRVZ and a strength of 0 to an impedance of DIGDRVF and a strength of 63. Before the simulation run starts, the DRVH and DRVL impedances of each output are assigned a strength from 0 to 63, based upon where their impedance falls on the scale from DIGDRVF to DIGDRVZ. The strengths are calculated as follows:

$$LZ = \ln(\text{DIGDRVZ})$$
$$LF = \ln(\text{DIGDRVF})$$
$$\text{DRVH\_STRENGTH} = 63 \bullet (\ln(\text{DRVH}) - LZ) / (LF - LZ)$$
$$\text{DRVL\_STRENGTH} = 63 \bullet (\ln(\text{DRVL}) - LZ) / (LF - LZ)$$

During the simulation run, each output is assigned the DRVH or the DRVL strength based upon its current state.

DIGOVRDRV, DIGDRVF, and DIGDRVZ are specified in the Global Settings dialog box and can be changed for a particular circuit with the .OPTIONS command.

Tri-state outputs
A common situation where multiple strengths help in modeling is the tri-state bus. In this situation many tri-state devices are connected to a common node. Each has an enable pin. If the enable pin is disabled, the output impedance is DIGDRVZ and the strength is 0. Typically, all outputs but one will be disabled and their strengths will be 0. One output will be enabled and its strength will be greater than 0 and hence the output state will be determined by the one enabled output. It is important to remember that the DRVH or DRVL of the enabled output must still exceed the DIGDRVZ by DIGOVRDRV to control the node state.

Open-collector outputs
Another situation where multiple strengths help is in the modeling of open collector outputs. In this situation, many devices are connected to a common node. Each might typically have impedances of DRVL=100 and DRVH = 20K. A single PULLUP device is connected to the common node. It provides a weak '1' of typically 1K. If any output is a '0', then its strength is sufficient to overcome the weak PULLUP '1' strength (1K) and the even weaker 'Z' strength (20K) of the other output devices. The output becomes a '0'. If all outputs are at a '1' at Z strength (20K), then the PULLUP provides a '1' at higher than Z strength and the output goes to a '1'.

Timing models

All digital primitives except PULLUP, PULLDN, CONSTRAINT, and PINDLY, employ a timing model statement whose parameters specify the various timing characteristics unique to the part. The general types of parameters include propagation delay, pulse width, setup time, hold time, and switching time. The parameter names are constructed from a standard set of abbreviations:

TP	Propagation delay
TW	Pulse width
TSU	Setup time
THD	Hold time
TSW	Switching time

MN	Minimum
TY	Typical
MX	Maximum

LH	Low to high transition
HL	High to low transition
ZL	Z to low transition
ZH	Z to high transition
LZ	Low to Z transition
HZ	High to Z transition

Here are some examples:

TPLHMN: Minimum propagation delay for a low to high transition for standard and tri-state gates.

TWPCLTY: Typical JKFF prebar or clearbar width at the low state. Here the P of the prebar pin name and the C of the clearbar pin name are integrated into the parameter name.

THDCLKMN: Minimum hold time for J and K or D inputs after the active CLK transition.

This is an example of a timing model statement for a standard gate:

```
.MODEL DL_01 UGATE (TPLHMN=8NS TPLHTY=11NS
+ TPLHMX=13NS TPHLMN=6NS TPHLTY=9NS TPHLMX=12NS)
```

Unless otherwise noted, all timing parameters have a default value of zero.

Unspecified propagation delays

The timing model syntax provides minimum, typical, and maximum values for each propagation delay parameter. The names of these parameters always begin with 'TP'. Data books often specify only one or two of these parameters. Since the logic simulator can't just assume a default value of zero for unspecified parameters, it calculates them according to the following rules:

If the typical value is specified:

If the minimum value is unspecified:

TPXXMN = DIGMNTYSCALE · TPXXTY

If the maximum value is unspecified:

TPXXMX = DIGTYMXSCALE · TPXXTY

If the typical value is not specified:

If the minimum and maximum values are both specified:

TPXXTY = (TPXXMN + TPXXMX) / 2

If only the minimum value is specified:

TPXXTY = TPXXMN / DIGMNTYSCALE

If only the maximum value is specified:

TPXXTY = TPXXMX / DIGTYMXSCALE

If none of the values are specified, then obviously:

TPXXMN = TPXXTY = TPXXMX = 0

Default values of the parameters DIGMNTYSCALE and DIGTYMXSCALE are specified from the Global Settings dialog box. The default value is used unless it is changed in a particular circuit with the .OPTIONS command. For example:

.OPTIONS DIGMNTYSCALE=.35

Placing this command in a circuit changes its DIGMNTYSCALE value only.

Note that these rules apply only to propagation delay parameters.

Unspecified timing constraints

Timing constraints include pulse widths and setup and hold times. Typical and maximum values for these timing constraints are frequently missing from data books. Unlike propagation parameters, these missing values cannot be obtained by a simple scaling procedure. Instead, the simulator calculates the missing values according to the following procedure:

If the minimum value is not specified:
minimum = 0

If the maximum value is not specified:
If the typical value is specified:
maximum = typical

If the minimum value is specified:
maximum = minimum

If the typical value is not specified:
typical = (maximum + minimum)/2

Note that a parameter is unspecified if it is missing from the model statement. For example, a model statement with unspecified parameters looks like this:

```
.MODEL TOR UGATE ( )
```

Another approach is to include the unspecified parameters and set them equal to the special value, -1, as a way of signalling to the generation routines that they are to be considered unspecified. This form of model statement looks like this:

```
.MODEL TOR UGATE ( TPLHMN=-1 TPLHTY=-1 TPLHMX=-1
+ TPHLMN=-1 TPHLTY=-1 TPHLMX=-1 )
```

Any parameter set equal to -1 is treated as unspecified and thus calculated from the other parameters according to the above rules. Having the names of unspecified parameters readily available is a convenient memory aid in case they need to be modified. MC7 generates default model statements in this form.

Propagation delays

Loading delays

The propagation delay through a digital device is specified mainly in the timing model through the propagation delay parameters. The I/O model can also affect the propagation delay through the loading delay. The two loading delays (low to high and high to low) are calculated from the capacitive loading on the node prior to the simulation run. The capacitive load is derived from the I/O models for the devices that are connected to the node. The total capacitive load is obtained from the sum of all INLD values from devices whose inputs are connected and OUTLD values from devices whose outputs are connected. The total capacitive load is assumed to be driven by the device's driving impedances, DRVH or DRVL. The two loading delays are calculated for each device as follows:

Loading delay low to high= ln(2)•DRVH•CTOTAL
Loading delay high to low= ln(2)•DRVL•CTOTAL

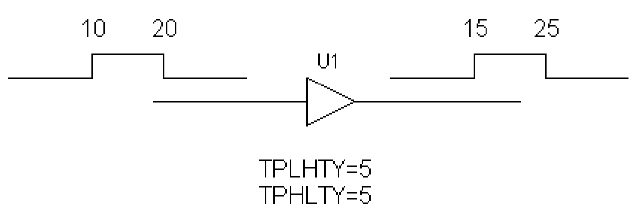
During the simulation, one of these delays, depending upon the transition, is added to the timing model delay when an event is scheduled.

Inertial delays

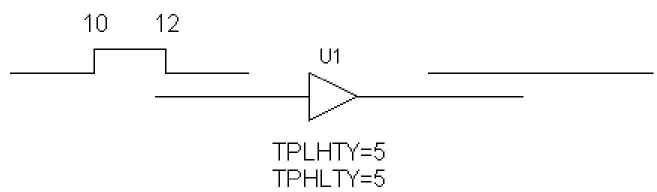
Micro-Cap models all delays except those of the DLYLINE on the inertial model. Inertial delay models work on the physical principle that a signal must be impressed upon a device for a certain minimum time before the device will respond. The principle can be summarized as follows:

If the pulse width is less than the delay the pulse is cancelled.

In this circuit, the pulse width is 10. That is greater than the delay of 5, so the pulse will pass:

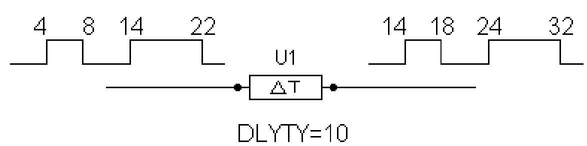


In this example, the pulse width is 2. Since that is less than the specified delay of 5, the pulse is inertially cancelled.



Inertial cancellation can be turned off from the Preferences dialog box.

Transport delays
The alternative to inertial delay is transport delay. In this delay model, all signals



are passed, regardless of the pulse duration. For example, this circuit passes all pulses, even though they are less than the specified delay.

This type of delay is useful when you want to shift a signal by some fixed delay value without removing any of its narrow pulses. Only the DLYLINE device uses transport delay. All delay parameters of all other devices are treated as inertial, except when inertial cancellation is disabled.

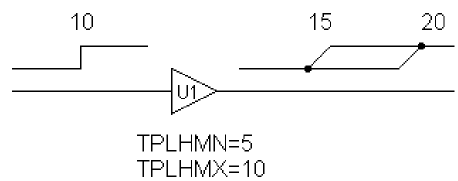
Digital delay ambiguity

Digital devices contain model parameters that specify a minimum, typical, and a maximum value for each timing value. The MNTYMXDLY parameter can be used to specify which of the delays to use:

MNTYMXDLY	**Meaning**
0	Set MNTYMXDLY = DIGMNTYMX
1	Use minimum delays
2	Use typical delays
3	Use maximum delays
4	Use worst-case (both minimum and maximum) delays

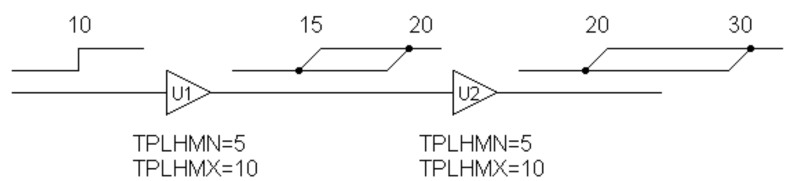
When worst-case delays are used, the simulator produces an ambiguous region between the minimum and maximum delays that represents the period of time when the signal is changing. The signal cannot be said to be in the old state or the new state, it is merely in transition between the two states. This ambiguity region is represented by a rising or a falling state.

For example, in this circuit, the input changes from a '0' to a '1' at 10.

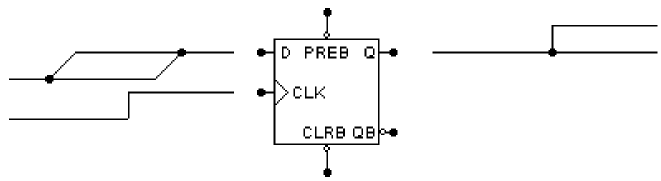


If MNTYMXDLY is 4, the resulting worst-case run produces a rising state at 15 since the minimum low to high delay is 5. The "TPLHMN=5" says that the delay is at least 5. It may be more. The "TPLHMX=10" says that the delay is at most 10. It may be less. The rising state between 15 and 20 reflects the uncertainty.

Ambiguity regions can accumulate as the signal passes through a succession of devices. For example, in this circuit the rising state ambiguity region increases as the signal passes through the two buffers.

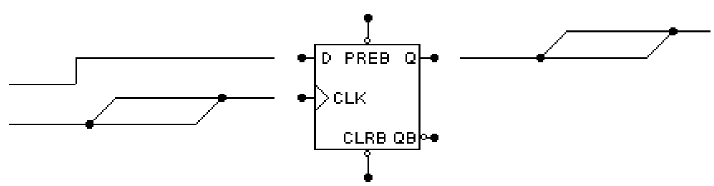


Ambiguity can do more than just fuzz up the signal edges. It can catastrophically alter simulation results. In the circuit below, the ambiguous D input signal makes the output 'X' since there are two possible outcomes. If the clock completes its rise before the D input changes, the Q output remains '0'. If the clock completes its rise after the D input changes, the new '1' state is loaded and the Q output becomes '1'. Since there are two possible opposite states for Q depending upon when the D in-



put changes, the Q output is assigned 'X' to reflect the uncertainty.

Rising inputs do not necessarily generate 'X' results. If the CLK input samples while the D input is stable, then there is no ambiguity and the Q output is known. This is true even if the CLK input is 'R', as in this example.



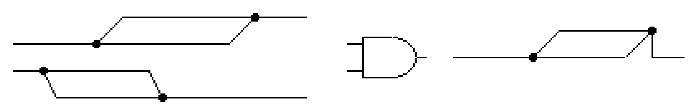
Timing hazards

If the exact arrival times of the inputs to a digital device are unknown, the output may glitch or be set to 'X'. This condition is referred to as a timing hazard.

There are several types of hazards, including:

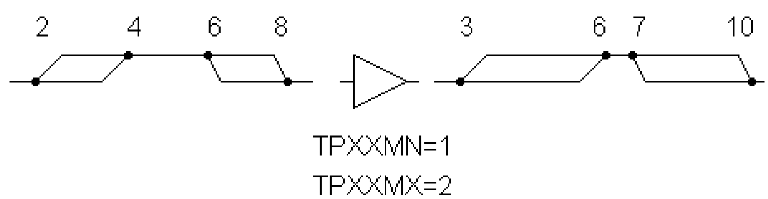
- Convergence hazards
- Cumulative ambiguity hazards
- Critical hazards

Convergence hazards
A convergence hazard occurs when two signals converge to the inputs of a gate and there is overlapping uncertainty in their arrival times, as in this example. This hazard could be represented as a 0-X-0 transition or as a 0-R-0 transition.

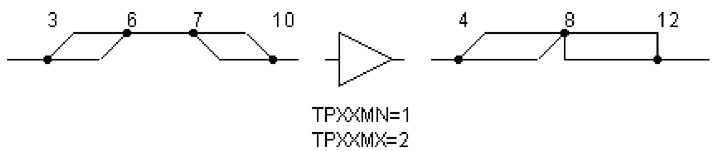


We represented it as a 0-R-0 transition to match the usual representation of the hazard. It should be interpreted as a possible short pulse, extending no longer than the overlap of the uncertainty portions of the two input signals.

Cumulative ambiguity hazards
Digital worst-case timing is activated when a gate's MNTYMXDLY value is set to 4. In this case the simulator adds an 'R' state between the specified minimum delay and maximum delay at each 0-1 and 0-R transition and an 'F' state between the specified minimum delay and maximum delay at each 1-0 and 1-F transition. As a signal passes through a gate with worst-case timing specified, the uncertainties of its transitions increase, as in this example:



As the signal passes through additional stages, the uncertainties can accumulate to the point where parts of the signal are ambiguous, as in this example:



At 7, the simulator attempts to schedule the 'F' state at 8 (7+1). But the '1' state is already scheduled to occur on the output at 8 (6+2). Since this is ambiguous, the simulator signals a hazard with an 'X' at 8.

Cumulative ambiguity is a particular problem with circuits that feed an inverted gate output back to its input to create an oscillator. In such cases, the MNTYMXDLY parameter of the oscillator gate should be set to something other than 4 (worst-case), to stop the signal destruction.

Critical hazards

Convergence and ambiguity hazards indicate potential problems. They may or may not cause a problem in the circuit. If the hazard causes a flip-flop or other storage device to store incorrect data, then they become critical hazards, since they will almost certainly cause a failure in the circuit.

The analog/digital interface

When a digital node and an analog node are connected together in a circuit, the program breaks the connection and inserts between the two parts an interface circuit specified by the I/O model. This interface circuit contains analog devices like resistors, capacitors, diodes, and transistors. It also contains either an analog to digital or digital to analog interface device. These devices provide the fundamental translation between the analog and digital circuits.

In addition, the simulator generates an analog power supply circuit to drive the interface circuits, as specified in the I/O model. The I/O model statement parameter, IO_LEVEL, selects one of four possible interface circuits to use. It is also possible to change the digital power supply used in the interface circuit by modifying the power supply subcircuit.

The simulator also creates additional nodes at the interface between analog and digital circuits. The creation and naming of these nodes is important to understand if you want to plot or print their values.

Nodes

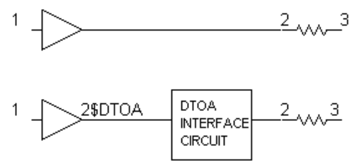
There are only two types of nodes, analog and digital. When an analog node is connected to a digital node, what happens? MC7 breaks the connection and adds an interface circuit between the two parts. The interface circuit is selected from the I/O model based upon the value of the IO_LEVEL attribute. The selection is based upon the following:

Level	Subcircuits	Behavior
1	AtoD1/DtoA1	AtoD creates R, F, and X levels
2	AtoD2/DtoA2	AtoD doesn't create R, F, and X levels
3	AtoD3/DtoA3	Same as Level 1
4	AtoD4/DtoA4	Same as Level 2

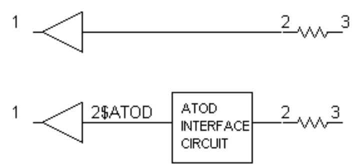
Table 23-2 Digital interface subcircuits

If an analog node is connected to a digital output, then a DtoA subcircuit is used. If an analog node is connected to a digital input, then an AtoD subcircuit is used. If an analog node is connected to both a digital input and a digital output then a

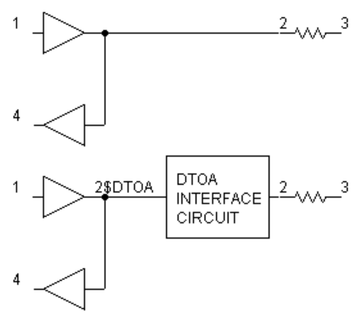
DtoA interface circuit is used. These cases are shown below.



When the digital node is an output node, a new digital node, 2$DTOA, is created and assigned to the digital output, and the DTOA interface circuit, specified by the I/O model statement IO_LEVEL parameter, is inserted between the new digital output node and the analog node.



When the digital node is an input node, a new digital node, 2$ATOD, is created and assigned to the digital input, and the ATOD interface circuit, specified by the I/O model statement IO_LEVEL parameter, is inserted between the new digital input node and the analog node.



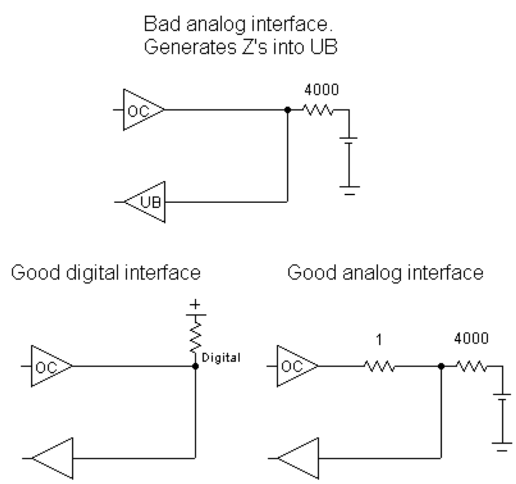
When an analog node is connected to a digital node with both input and output pins connected to it, a new digital node, 2$DTOA, is created and assigned to the digital node. An interface circuit is inserted between the new digital node and the analog node.

Tri-state interface caveat

When you have a digital node with input and output pins connected to it as in the last example, and the digital output comes from a tristate or open collector device, placing an analog resistor at the node will result in Z states entering the digital input. This happens because the output of the resistor does not drive the digital input, so no pullup or pulldown action can occur. To avoid this problem always follow the golden tri-state rule.

Use a digital pullup or pulldown device at tri-state nodes. Do not use an analog resistor. If you must use an analog resistor, separate the digital output from the digital input by a small value resistor.

The situation is summarized in the figure below:



Interface circuits
The I/O model parameter IO_LEVEL specifies which of the four possible interface circuits to use according to the table. If IO_LEVEL is 0, then the interface circuit will be selected by the value of the Global Settings parameter DIGIOLVL.

Level 1 Interface circuits
The level 1 interface circuits generate the intermediate logic levels (R, F, X) between the voltage ranges VILMAX and VIHMIN. A voltage that ramps smoothly from below VILMAX to above VIHMIN and back again, will generate the sequence 0, R, 1, F, 0. An X is generated if the voltage starts in the VILMAX - VIHMIN region, or the voltage reverses in the VILMAX - VIHMIN region and crosses a previously crossed threshold. The level 1 interface circuits are more accurate than level 2 circuits.

Level 2 Interface circuits
The level 2 interface circuits generate only the logic levels (0, 1). An exact switching voltage is assumed. These interface circuits are less accurate but generate no uncertain states (R, F, X) that could potentially cause simulation problems.

Level 3 and Level 4 Interface circuits. These are equivalent to Level 1 and Level 2, respectively, and are not currently used in the standard digital library.

Power supply circuits
When an analog node and a digital node are connected together, and an ATOD or DTOA interface is required, the circuit needs a power supply. The power supply is created automatically using the parameter DIGPOWER from the I/O model statement for a particular digital family, such as 7400 TTL. This parameter specifies the name of the power supply subcircuit to use. The main power supply subcircuit is called "DIGIFPWR". This subcircuit provides the voltage sources and power pins needed by the interface circuits. You can, of course, change the power supply circuit by specifying another subcircuit name, or by altering the "DIGIFPWR" subcircuit itself. We recommend leaving the original "DIGIFPWR" subcircuit intact, and cloning new versions, with related names such as "DIGIFPW1". This lets you easily revert to the old version by simply changing the DIGPOWER="DIGIFPWR" part of the I/O model statement. The I/O model statements are located in the DIGIO.LIB model library file.

The DIGIO.LIB model library file
This file contains the I/O model statements, ATOD and DTOA interface circuits, ATOD (O device) and DTOA (N device) model statements, and power supply circuits for the entire Digital Library.

General digital primitive format

The digital device format is similar to the general SPICE component format. Both analog and digital primitives require node numbers, optional parameters, and model statements. Analog devices use at most a single model statement. Digital devices require a timing model statement and an I/O model statement.

Timing model statements define propagation delays and timing constraints. I/O model statements define the impedances, equivalent circuits, and switching times of the analog/digital interface model. The I/O model is mainly used when a digital node is connected to an analog node, but it also defines the impedances used to resolve states when device outputs of differing strengths are wired together.

Timing model statements embody data unique to each device, so most devices have unique timing model statements. The same is not true of I/O models since the interface is generally standardized for all devices within a particular digital family. For example, all 74LS devices use the same interface specification.

General format
U<*name*> <*primitive type*> [(<*parameter value*>*)]
+<*digital power node*> <*digital ground node*>
+<*node*>*
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subcircuit select value*>]

Examples
U1 JKFF(1) $G_DPWR $G_DGND PRB CLB CKB J K Q QB D0_74 IO_STD
U1 NOR(3) $G_DPWR $G_DGND 10 20 30 40 D0_74 IO_STD

Definition
U<*name*>
This is the part name.

<*primitive type*>
The <*primitive type*> specifies which of the primitive digital types, such as NAND, NOR, JKFF, or PLA, the part is.

[(<*parameter value*>*)]
Depending upon the <*primitive type*>, these are zero or more parameters representing the number of input and/or output nodes.

<*digital power node*> <*digital ground node*>
These nodes provide power to the A/D interface circuits employed when an analog node is connected to a digital node. Usually the global pins $G_DPWR and $G_DGND are used. Refer to the section, "The analog/digital interface" for more information.

<node>*
Depending upon the primitive type, these are the node names of the actual input and/or output node numbers.

<*timing model name*>
This name refers to a timing model statement, which specifies propagation and constraint timing values. Each model has minimum, typical, and maximum values for each timing parameter. The MNTYMXDLY device parameter selects one of the values or it lets the DIGMNTYMX select one globally.

<*I/O model name*>
This name refers to an I/O model statement which specifies impedance and interface circuit information for modeling the analog/digital interface.

[MNTYMXDLY=<*delay select value*>]
This value selects the minimum, typical, or maximum value for each timing parameter as follows:

0=Current value of DIGMNTYMX
1=Minimum
2=Typical
3=Maximum
4=Worst case. Use both minimum and maximum delays.

[IO_LEVEL=<interface subcircuit select value>]
This selects one of four interface circuits named in the I/O model. These interface circuits are used at the digital/analog interface. Refer to the I/O model topic at the end of this chapter for more information.

Timing model form
.MODEL <*model name*> <model type> (<*model parameters*>*)
Each *primitive type* has a unique <model type> and <*model parameters*>.

I/O model form
.MODEL <*model name*> UIO (<*model parameters*>*)
There is one I/O model structure that is used by all primitives.

Parts from the Digital Primitives section need the I/O and Timing model information to be specified when the part is placed in the schematic. Digital Library section parts are modeled as subcircuits and have the requisite I/O and Timing model information predefined in a subckt located in a DIGXXXX.LIB text file library. The models are linked to their I/O and timing models as follows:

7410 schematic part

The 7410 SUBCKT and TIMING models are in the DIG000.LIB referenced by the implicit .LIB NOM.LIB statement.

The I/O model statement, the ATOD and DTOA interface circuits, and the O and N model statements are found in the DIGIO.LIB referenced by the implicit .LIB NOM.LIB statement.

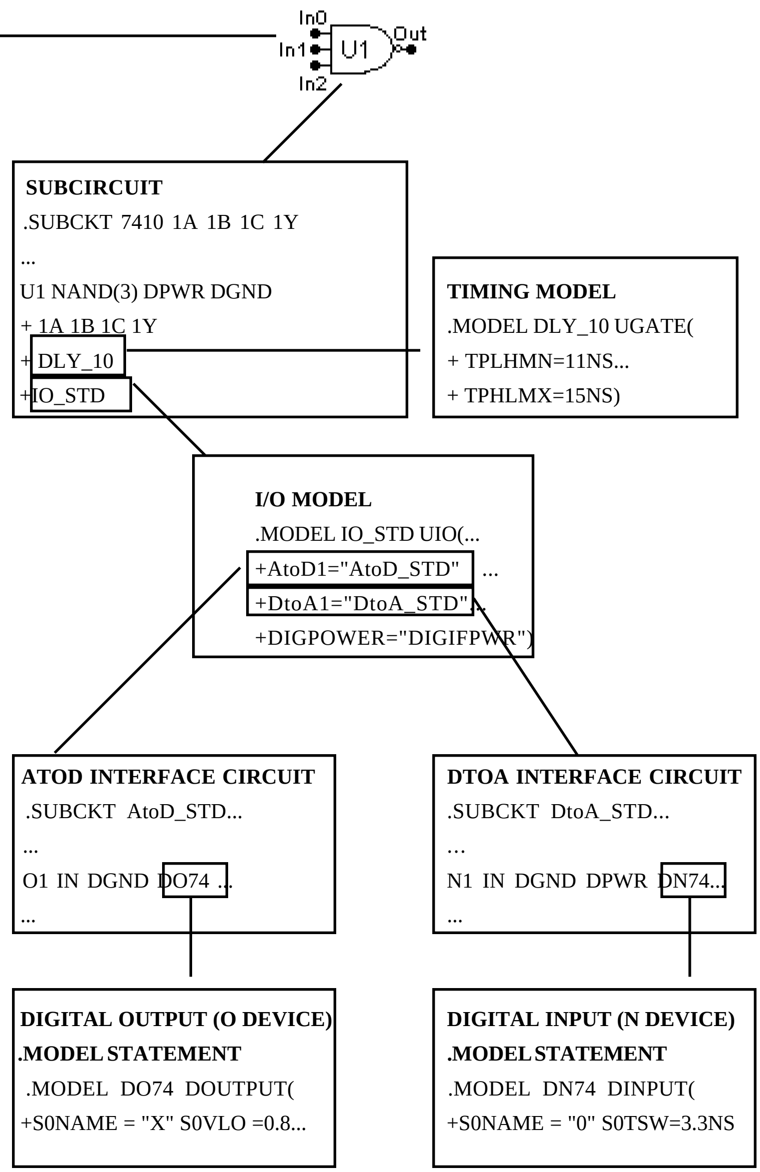


Figure 23-1 Digital Model References

Primitives

Digital primitives are as follows:

Primitive Class	**Type**	**Description**
Standard Gates	BUF	Buffer
	INV	Inverter
	AND	AND gate
	NAND	NAND gate
	OR	OR gate
	NOR	NOR gate
	XOR	Exclusive OR
	NXOR	Exclusive NOR
	BUFA	Buffer array
	INVA	Inverter array
	ANDA	AND gate array
	NANDA	NAND gate array
	ORA	OR gate array
	NORA	NOR gate array
	XORA	Exclusive OR gate array
	NXORA	Exclusive NOR gate array
	AO	AND-OR compound gate
	OA	OR-AND compound gate
	AOI	AND-NOR compound gate
	OAI	OR-NAND compound gate

Table 23-3A Digital primitives

Primitive Class	Type	Description
Tri-state Gates	BUF3	Buffer
	INV3	Inverter
	AND3	AND gate
	NAND3	NAND gate
	OR3	OR gate
	NOR3	NOR gate
	XOR3	Exclusive OR
	NXOR3	Exclusive NOR
	BUF3A	Buffer array
	INV3A	Inverter array
	AND3A	AND gate array
	NAND3A	NAND gate array
	OR3A	OR gate array
	NOR3A	NOR gate array
	XOR3A	Exclusive OR array
	NXOR3A	Exclusive NOR array
Flip-flops / Latches	JKFF	JK negative-edge triggered
	DFF	D positive-edge triggered
	SRFF	SR gated latch
	DLTCH	D gated latch

Table 23-3B Digital primitives (continued)

Primitive Class	**Type**	**Description**
Pullups / Pulldowns	PULLUP	Pullup resistor array
	PULLDN	Pulldown resistor array
Delay Lines	DLYLINE	Non-inertial delay line
Programmable Logic Arrays	PLAND	AND array
	PLOR	OR array
	PLXOR	Exclusive OR array
	PLNXOR	Exclusive NOR array
	PLNAND	NAND array
	PLNOR	NOR array
	PLANDC	AND array plus complement
	PLORC	OR array plus complement
	PLXORC	XOR array plus complement
	PLNANDC	NAND array plus complement
	PLNORC	NOR array plus complement
	PLNXORC	NXOR array plus complement
Multi-bit Converters	ADC	Multi-bit ATOD converter
	DAC	Multi-bit DTOA converter
Behavioral Models	LOGICEXP	Logic expression
	PINDLY	Pin-to-pin delay
	CONSTRAINT	Constraint checker

Table 23-3C Digital primitives (continued)

Gates

Two types of gates are provided:

Standard gates: Their outputs are always enabled. The output impedance is:

Output State	Impedance
0	DRVL (from I/O model)
1	DRVH (from I/O model)

Tri-state gates: Their outputs are enabled by an enable pin. The output impedance depends upon the enable state:

Enable state	Output State	Impedance
1	0	DRVL (from I/O model)
1	1	DRVH (from I/O model)
0	Z	DIGDRVZ

DIGDRVZ is the impedance which corresponds to the Z state. It is specified in the Global Settings dialog box or by a .OPTIONS statement. DRVH and DRVL come from the I/O model.

Gates come in two forms, *simple* and *arrays*. *Simple* gates have one or more inputs, but only one output. *Arrays* contain one or more simple gates, with one output per simple gate. *Note that arrays of gates are available for use only in SPICE text circuits or SPICE text subckt libraries. Schematic gates are always simple.*

The usual Boolean rules apply to these gates. If an input state is X, the output is calculated for an input of 1 and an input of 0. If the output state differs for these two calculations, then the output is X. This avoids propagating pessimistic X states. This definition of the X state results in the following identities:

0 AND X = 0	0 NOR X = X
1 AND X = X	1 NOR X = 0
0 NAND X = 1	1 XOR X = X
1 NAND X = X	0 XOR X = X
0 OR X = X	
1 OR X = 1	

Standard gates

SPICE format
U<*name*> <*gate type*>[(<*parameters*>)*]
+<*digital power node*> <*digital ground node*>
+<*input node*>* <*output node*>*
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Examples:
A 5 input NOR gate:
U1 NOR(5) $G_DPWR $G_DGND IN1 IN2 IN3 IN4 IN5 OUT
D0_GATE IO_STD MNTYMXDLY=0 IO_LEVEL=2

A 3 gate NAND array with 2 inputs per gate:
U17 NANDA(2,3) $G_DPWR $G_DGND 1A 1B 2A 2B 3A 3B O1 O2 O3
DLY1 IO_ACT

An AND-OR compound gate having 2 input gates with 3 inputs per gate:
UCMPD AO(3,2) $G_DPWR $G_DGND i1a i1b i1c i2a i2b i2c out
dlymod io_hc_oc MNTYMXDLY=3

Schematic format
PART attribute
<*name*>

Example
U1

TIMING MODEL attribute
<*timing model name*>

Example
74LS

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

MNTYMXDLY attribute
<*delay select value*>

Example
2

IO_LEVEL attribute
<*interface subckt select value*>

Example
1

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Model statement form
.MODEL <*timing model name*> UGATE ([*model parameters*])

Example
.MODEL TOR UGATE (TPLHMN=3ns TPLHTY=5ns TPLHMX=7ns
+ TPHLMN=4ns TPHLTY=6ns TPHLMX=7ns)

The standard gate types and their parameters are shown in Table 23-4. The table uses the standard syntax:

in	one input node
in*	one or more input nodes
out	one output node
out*	one or more output nodes

The phrase <# of inputs> means the number of inputs per gate. The phrase <# of gates> means the number of gates in the array.

Type	Parameters	Nodes	Description
BUF		in, out	Buffer
INV		in, out	Inverter
AND	<no. of inputs>	in*, out	AND gate
NAND	<no. of inputs>	in*, out	NAND gate
OR	<no. of inputs>	in*, out	OR gate
NOR	<no. of inputs>	in*, out	NOR gate
XOR		in1, in2, out	Exclusive OR gate
NXOR		in1, in2, out	Exclusive NOR gate
BUFA	<no. of gates>	in*, out*	Buffer array
INVA	<no. of gates>	in*, out*	Inverter array
ANDA	<no. of inputs>, <no. of gates>	in*, out*	AND array
NANDA	<no. of inputs>, <no. of gates>	in*, out*	NAND array
ORA	<no. of inputs>, <no. of gates>	in*, out*	OR array
NORA	<no. of inputs>, <no. of gates>	in*, out*	NOR array
XORA	<no. of gates>	in*, out*	Exclusive OR array
NXORA	<no. of gates>	in*, out*	Exclusive NOR array
AO	<no. of inputs>, <no. of gates>	in*, out	AND-OR compound gate
OA	<no. of inputs>, <no. of gates>	in*, out	OR-AND compound gate
AOI	<no. of inputs>, <no. of gates>	in*, out	AND-NOR compound gate
OAI	<no. of inputs>, <no. of gates>	in*, out	OR-NAND compound gate

Table 23-4 Standard gate types

Input nodes are listed before output nodes. In a gate array, the node order is: the input nodes for the first gate, the input nodes for the second gate,..., the input nodes for the final gate, the output node for the first gate, the output node for the second gate,..., the output node for the final gate. The number of output nodes is equal to the number of gates.

A compound gate contains two levels of logic. The phrase <*no. of inputs*> is the number of inputs per first-level gate, and <no. of gates> is the number of first-level gates. All of the first-level gate outputs are connected to one second-level gate, so the device only has one output. The node order is: the input nodes for the first first-level gate,..., the input nodes for the final first-level gate, and the output node. For an OA gate, the first-level gates are <*no. of inputs*> input OR gates, and they all feed into one AND gate, which produces the output.

Parameter	**Description**	**Units**
TPLHMN	Delay: min low to high	sec
TPLHTY	Delay: typ low to high	sec
TPLHMX	Delay: max low to high	sec
TPHLMN	Delay: min high to low	sec
TPHLTY	Delay: typ high to low	sec
TPHLMX	Delay: max high to low	sec

Table 23-5 Standard gate timing model parameters

Special Component editor fields
Inputs <*number of inputs*>
The Component editor has a special 'Inputs' field for standard gates. It lets you specify the number of gate inputs. When you enter a value, the system places an output pin and the specified <*number of inputs*> pins on the display and you must drag them into place on the shape.

Tri-state gates

SPICE format
U<*name*> <*tri-state gate type*>[(<*parameters*>*)]
+<*digital power node*> <*digital ground node*>
+<*input node*>* <*enable node*> <*output node*>*
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Examples:
A 3 input tri-state NOR gate:
U20 NOR3(3) $G_DPWR $G_DGND IN1 IN2 IN3 ENABLE OUT
D0_GATE IO_STD MNTYMXDLY=0 IO_LEVEL=2

A 3 gate tri-state AND array with 2 inputs per gate:
UBX AND3A(2,3) $G_DPWR $G_DGND 1A 1B 2A 2B 3A 3B EN O1 O2 O3
DLY1 IO_ACT

Schematic format
PART attribute
<*name*>

Example
U1

TIMING MODEL attribute
<*timing model name*>

Example
74ALS

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
1

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Model statement form
.MODEL <*timing model name*> UTGATE ([*model parameters*])

Example
.MODEL TRIG UTGATE (TPLHMN=2ns TPLHTY=3ns TPLHMX=5ns
+ TPZLMN=4ns TPZLTY=6ns TPZLMX=8ns)

The tri-state gate types and their parameters are shown in Table 23-6. The table uses the standard syntax:

en	one enable node
in	one input node
in*	one or more input nodes
out	one output node
out*	one or more output nodes

Type	Parameters	Nodes	Description
BUF3		in, en, out	Buffer
INV3		in, en, out	Inverter
AND3	<no. of inputs>	in*, en, out	AND gate
NAND3	<no. of inputs>	in*, en, out	NAND gate
OR3	<no. of inputs>	in*, en, out	OR gate
NOR3	<no. of inputs>	in*, en, out	NOR gate
XOR3		in1, in2, en, out	Exclusive OR gate
NXOR3		in1, in2, en, out	Exclusive NOR gate
BUF3A	<no. of gates>	in*, en, out*	Buffer array
INV3A	<no. of gates>	in*, en, out*	Inverter array
AND3A	<no. of inputs>, <no. of gates>	in*, en, out*	AND array
NAND3A	<no. of inputs>, <no. of gates>	in*, en, out*	NAND array
OR3A	<no. of inputs>, <no. of gates>	in*, en, out*	OR array
NOR3A	<no. of inputs>, <no. of gates>	in*, en, out*	NOR array
XOR3A	<no. of gates>	in*, en, out*	Exclusive OR array
NXOR3A	<no. of gates>	in*, en, out*	Exclusive NOR array

Table 23-6 Tri-state gate types

The phrase *<no. of inputs>* is the number of inputs per gate, and the phrase *<no. of gates>* is the number of gates in an array. Input nodes come first, then the enable node, and then the output nodes. In a gate array, the node order is: input nodes for the first gate, input nodes for the second gate,..., input nodes for the final gate, enable node, output node for the first gate, output node for the second gate,..., output node for the final gate. The number of output nodes is equal to the number of gates. All gates in an array of tri-state gates share a common enable input.

Parameter	Description	Units
TPLHMN	Delay: min low to high	sec
TPLHTY	Delay: typ low to high	sec
TPLHMX	Delay: max low to high	sec
TPHLMN	Delay: min high to low	sec
TPHLTY	Delay: typ high to low	sec
TPHLMX	Delay: max high to low	sec
TPLZMN	Delay: min low to Z	sec
TPLZTY	Delay: typ low to Z	sec
TPLZMX	Delay: max low to Z	sec
TPHZMN	Delay: min high to Z	sec
TPHZTY	Delay: typ high to Z	sec
TPHZMX	Delay: max high to Z	sec
TPZLMN	Delay: min Z to low	sec
TPZLTY	Delay: typ Z to low	sec
TPZLMX	Delay: max Z to low	sec
TPZHMN	Delay: min Z to high	sec
TPZHTY	Delay: typ Z to high	sec
TPZHMX	Delay: max Z to high	sec

Table 23-7 Tri-state gate timing model parameters

Special Component editor fields
Inputs <*number of inputs*>
The 'Inputs' field specifies the number of inputs for the gate. When you enter this value, MC7 places an enable pin, an output pin, and the specified <*number of inputs*> pins on the display and you must drag them into place on the shape.

Flip-flops and latches

Both edge-triggered and level gated flip-flops are provided. Edge-triggered flip-flops include the JK flip-flop, JKFF, and the D-type flip-flop, DFF. Both change their state a specified delay time after the active edge of the clock. The active edge of the JKFF is the negative or falling edge. The active edge of the DFF is the positive or rising edge. The gated devices include the set-reset flip-flop, SRFF, and the D-latch, DLTCH. The outputs of these devices follow the input while the gate node is high. The input state sampled during the high state of the gate node is latched and stable during the low state of the gate node. The device is defined as an array of flip-flops, so one or more flip-flops may be instantiated with a single device. The presetbar, clearbar, and clock or gate nodes are common to all flip-flops in the array.

Initialization
Flip-flop devices may be initialized to a particular state by using DIGINITSTATE. The flip-flop true outputs are set according to this table:

DIGINITSTATE	Flip-flop Q output
0	0
1	1
All other values	X

DIGINITSTATE is set in the Global Settings dialog box. It can also be changed for a particular circuit with the .OPTIONS command.

X-levels
X states are not propagated to the output if the logic precludes that from happening. For example, if clearbar = X, and Q = 0, the Q stays at 0, since both clearbar possibilities (clearbar = 0 and clearbar = 1) both produce Q=0. Similarly, if clearbar = X, and Q = 1, Q goes to X since the two clearbar possibilities (clearbar = 0 and clearbar = 1) each produce different Q outputs (Q=0 and Q=1).

Timing violations
Timing constraints, specified in Table 23-8 and 23-11, are checked only if the value is not zero. If a constraint is violated, the simulator places a warning message in the Numeric Output window, and in the text file CIRCUITNAME.TNO.

Arrays of flip-flops
Note that arrays of flip-flops or latches are available only for SPICE text circuits or SPICE text subckt libraries. Schematic flip-flops or latches are always singles.

Edge-triggered flip-flops

Two types of edge-triggered flip-flops are provided, the JKFF and the DFF. Both of these devices change after the active edge of the clock. The active edge of the JKFF is the negative or falling edge. The active edge of the DFF is the positive or rising edge.

SPICE format
U<*name*> JKFF(<*no. of flip-flops*>)
+<*digital power node*> <*digital ground node*>
+<*presetbar node*> <*clearbar node*> <*clockbar node*>
+<*first j node*>...<*last j node*>
+<*first k node*>...<*last k node*>
+<*first q node*>...<*last q node*>
+<*first qbar node*>...<*last qbar node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

U<*name*> DFF(<*no. of flip-flops*>)
+<*digital power node*> <*digital ground node*>
+<*presetbar node*> <*clearbar node*> <*clock node*>
+<*first d node*>...<*last d node*>
+<*first q node*>...<*last q node*>
+<*first qbar node*>...<*last qbar node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Examples
U1 JKFF(2) $G_DPWR $G_DGND
+ PREBAR CLRBAR CLKBAR
+J1 J2 K1 K2 Q1 Q2 Q1BAR Q2BAR
+D0_EFF IO_STD IO_LEVEL=1

U4 DFF(1) $G_DPWR $G_DGND
+PREB CLRB CLK
+DIN Q QBAR DLY_DFF IO_ACT

Schematic format

PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
74XX

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Parameter	Description	Units
TPPCQLHMN	Delay: min preb/clrb to q/qb low to high	sec
TPPCQLHTY	Delay: typ preb/clrb to q/qb low to high	sec
TPPCQLHMX	Delay: max preb/clrb to q/qb low to high	sec
TPPCQHLMN	Delay: min preb/clrb to q/qb high to low	sec
TPPCQHLTY	Delay: typ preb/clrb to q/qb high to low	sec
TPPCQHLMX	Delay: max preb/clrb to q/qb high to low	sec
TWPCLMN	Width: min preb/clrb low	sec
TWPCLTY	Width: typ preb/clrb low	sec
TWPCLMX	Width: max preb/clrb low	sec
TPCLKQLHMN	Delay: min clk/clkb edge to q/qb low to high	sec
TPCLKQLHTY	Delay: typ clk/clkb edge to q/qb low to high	sec
TPCLKQLHMX	Delay: max clk/clkb edge to q/qb low to high	sec
TPCLKQHLMN	Delay: min clk/clkb edge to q/qb high to low	sec
TPCLKQHLTY	Delay: typ clk/clkb edge to q/qb high to low	sec
TPCLKQHLMX	Delay: max clk/clkb edge to q/qb high to low	sec

Table 23-8a Edge-triggered flip-flop timing model parameters

Model statement form
.MODEL <*timing model name*> UEFF ([*model parameters*])

Example
.MODEL JKDLY UEFF (tppcqlhty=10ns tppcqlhmx=25ns tpclkqlhty=12ns
+twpclty=15ns tsudclkty=4ns)

Parameter	Description	Units
TWCLKLMN	Width: min clk/clkb low	sec
TWCLKLTY	Width: typ clk/clkb low	sec
TWCLKLMX	Width: max clk/clkb low	sec
TWCLKHMN	Width: min clk/clkb high	sec
TWCLKHTY	Width: typ clk/clkb high	sec
TWCLKHMX	Width: max clk/clkb high	sec
TSUDCLKMN	Setup: min j/k/d to clk/clkb edge	sec
TSUDCLKTY	Setup: typ j/k/d to clk/clkb edge	sec
TSUDCLKMX	Setup: max j/k/d to clk/clkb edge	sec
TSUPCCLKHMN	Setup: min preb/clrb hi to clk/clkb edge	sec
TSUPCCLKHTY	Setup: typ preb/clrb hi to clk/clkb edge	sec
TSUPCCLKHMX	Setup: max preb/clrb hi to clk/clkb edge	sec
THDCLKMN	Hold: min j/k/d after clk/clkb edge	sec
THDCLKTY	Hold: typ j/k/d after clk/clkb edge	sec
THDCLKMX	Hold: max j/k/d after clk/clkb edge	sec

Table 23-8b Edge-triggered flip-flop timing model parameters

The parameter <*no. of flip-flops*> specifies the number of flip-flops and is available only for SPICE circuits or libraries. Schematic flip-flop components are available only as singles. The three nodes, <*presetbar node*>, <*clearbar node*>, and <*clock node*> are common to all flip-flops in the array.

J	K	CLK	PREB	CLRB	Q	QB
X	X	X	1	0	0	1
X	X	X	0	1	1	0
X	X	X	0	0	Unstable	Unstable
X	X	0	1	1	Q'	QB'
X	X	1	1	1	Q'	QB'
0	0	↓	1	1	Q'	QB'
0	1	↓	1	1	0	1
1	0	↓	1	1	1	0
1	1	↓	1	1	QB'	Q'

Table 23-9 JKFF flip-flop truth table

D	CLK	PREB	CLRB	Q	QB
X	X	1	0	0	1
X	X	0	1	1	0
X	X	0	0	Unstable	Unstable
X	0	1	1	Q'	QB'
X	1	1	1	Q'	QB'
0	↑	1	1	0	1
1	↑	1	1	1	0

Table 23-10 DFF flip-flop truth table

Gated latch

Two types of gated latches are provided, the SRFF and the DLTCH. Both devices change during the high level of the gate.

SPICE format

U<*name*> SRFF(<*no. of latches*>)
+<*digital power node*> <*digital ground node*>
+<*presetbar node*> <*clearbar node*> <*gate node*>
+<*first s node*>...<*last s node*>
+<*first r node*>...<*last r node*>
+<*first q node*>...<*last q node*>
+<*first qbar node*>...<*last qbar node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

U<*name*> DLTCH(<*no. of latches*>)
+<*digital power node*> <*digital ground node*>
+<*presetbar node*> <*clearbar node*> <*gate node*>
+<*first d node*>...<*last d node*>
+<*first q node*>...<*last q node*>
+<*first qbar node*>...<*last qbar node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Examples
U1 SRFF(2) $G_DPWR $G_DGND
+ PREBAR CLRBAR CLK
+S1 S2 R1 R2 Q1 Q2 Q1BAR Q2BAR
+D0_SRFF IO_STD IO_LEVEL=1

U4 DLTCH(1) $G_DPWR $G_DGND
+PREB CLRB GATE
+D1 Q QBAR D_DLTCH IO_ALS

Schematic format
PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
D74

I/O MODEL attribute
<*I/O model name*>

Example
IO_LS

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Parameter	Description	Units
TPPCQLHMN	Delay: min preb/clrb to q/qb low to high	sec
TPPCQLHTY	Delay: typ preb/clrb to q/qb low to high	sec
TPPCQLHMX	Delay: max preb/clrb to q/qb low to high	sec
TPPCQHLMN	Delay: min preb/clrb to q/qb high to low	sec
TPPCQHLTY	Delay: typ preb/clrb to q/qb high to low	sec
TPPCQHLMX	Delay: max preb/clrb to q/qb high to low	sec
TWPCLMN	Width: min preb/clrb low	sec
TWPCLTY	Width: typ preb/clrb low	sec
TWPCLMX	Width: max preb/clrb low	sec
TPGQLHMN	Delay: min gate to q/qb low to high	sec
TPGQLHTY	Delay: typ gate to q/qb low to high	sec
TPGQLHMX	Delay: max gate to q/qb low to high	sec
TPGQHLMN	Delay: min gate to q/qb high to low	sec
TPGQHLTY	Delay: typ gate to q/qb high to low	sec
TPGQHLMX	Delay: max gate to q/qb high to low	sec

Table 23-11A Gated latch timing model parameters

Model statement form
.MODEL <*timing model name*> UGFF ([*model parameters*])

Example
.MODEL SR1 UGFF (tppcqlhty=10ns tppcqlhmx=25ns tpgqlhty=12ns
+twpclty=15ns tsudgty=4ns)

Parameter	Description	Units
TPDQLHMN	Delay: min s/r/d to q/qb low to high	sec
TPDQLHTY	Delay: typ s/r/d to q/qb low to high	sec
TPDQLHMX	Delay: max s/r/d to q/qb low to high	sec
TPDQHLMN	Delay: min s/r/d to q/qb high to low	sec
TPDQHLTY	Delay: typ s/r/d to q/qb high to low	sec
TPDQHLMX	Delay: max s/r/d to q/qb high to low	sec
TWGHMN	Width: min gate high	sec
TWGHTY	Width: typ gate high	sec
TWGHMX	Width: max gate high	sec
TSUDGMN	Setup: min s/r/d to gate edge	sec
TSUDGTY	Setup: typ s/r/d to gate edge	sec
TSUDGMX	Setup: max s/r/d to gate edge	sec
TSUPCGHMN	Setup: min preb/clrb high to gate edge	sec
TSUPCGHTY	Setup: typ preb/clrb high to gate edge	sec
TSUPCGHMX	Setup: max preb/clrb high to gate edge	sec
THDGMN	Hold: min s/r/d after gate edge	sec
THDGTY	Hold: typ s/r/d after gate edge	sec
THDGMX	Hold: max s/r/d after gate edge	sec

Table 23-11B Gated latch timing model parameters (continued)

The parameter <*no. of latches*> specifies the number of latches in the array and is available only for SPICE text circuits or text subckt libraries. Schematic latches are available only as singles. The three nodes, <*presetbar node*>, <*clearbar node*>, and <*gate node*> are common to all latches in the array.

S	R	GATE	PREB	CLRB	Q	QB
X	X	X	1	0	0	1
X	X	X	0	1	1	0
X	X	X	0	0	Unstable	Unstable
X	X	0	1	1	Q'	QB'
0	0	1	1	1	Q'	QB'
0	1	1	1	1	0	1
1	0	1	1	1	1	0
1	1	1	1	1	Unstable	Unstable

Table 23-12 SRFF latch truth table

D	GATE	PREB	CLRB	Q	QB
X	X	1	0	0	1
X	X	0	1	1	0
X	X	0	0	Unstable	Unstable
X	0	1	1	Q'	QB'
0	1	1	1	0	1
1	1	1	1	1	0

Table 23-13 DLTCH latch truth table

Pullup and pulldown

These devices provide a constant output level at a user-specified strength. The outputs are as follows:

Device	Level	Strength
Pullup	1	DRVH (from the I/O model)
Pulldn	0	DRVL (from the I/O model)

These devices provide a strong '1' to pull up or a strong '0' to pull down a group of open-collector devices whose outputs are wired together.

SPICE format

U<*name*> <*resistor type*>(<*number of resistors*>)
+<*digital power node*> <*digital ground node*>
+<*output node*>*
+<*I/O model name*>
+[IO_LEVEL=<*interface subckt select value*>]

<*resistor type*> is one of the following:

PULLUP	array of digital pullup resistors
PULLDN	array of digital pulldown resistors

<*number of resistors*> specifies the number of resistors in the array.

Note that these devices are digital devices, not analog devices. Their sole purpose is to provide a strong constant level to a set of open-collector digital devices whose outputs are wired together. Open-collector devices are simply digital devices with a very large value of DRVH in their I/O model.

PULLUP and PULLDN devices do not use timing models since there is no delay associated with them. They do, however, need I/O models, since they are digital devices.

Example

```
U1 PULLUP(8)
+$G_DPWR $G_DGND
+A1 A2 A3 A4 A5 A6 A7 A8
+ IO_STD
```

Schematic format
PART attribute
<*name*>

Example
U1

I/O MODEL attribute
<*I/O model name*>

Example
IO_ALS

IO_LEVEL attribute
<*interface subckt select value*>

Example
1

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Delay line

This device provides a constant delay according to the timing model parameters. Unlike the other digital devices, there is no inertial cancellation of narrow pulse widths through delay lines.

SPICE format
U<*name*> DLYLINE
+<*digital power node*> <*digital ground node*>
+<*input node*> <*output node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Example
U1 DLYLINE
+$G_DPWR $G_DGND
+IN OUT
+ DMOD IO1

Schematic format
PART attribute
<*name*>

Example
U1

TIMING MODEL attribute
<*timing model name*>

Example
DELAY1

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

MNTYMXDLY attribute
<*delay select value*>

Example
2

IO_LEVEL attribute
<*interface subckt select value*>

Example
1

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Parameter	**Description**	**Units**
DLYMN	Delay: min	sec
DLYTY	Delay: typical	sec
DLYMX	Delay: max	sec

Table 23-14 Delay line timing model parameters

Programmable logic arrays

The programmable logic array is designed to allow modeling of a wide variety of programmable logic devices. The device is constructed of a user-specified number of inputs, visualized as columns, and a user-specified number of outputs, which form the rows. Each output (row) is a gate, whose inputs are selected from the inputs (columns). The device is programmed by choosing the inputs to be a part of the gate. The type of gate is determined by the PLA type. A PLNAND provides NAND gates, a PLOR provides OR gates, and so on. The PLA device provides a programmable core for modeling commercial PLA parts.

There are two ways to program the PLA. The normal way is to provide the data in a JEDEC format file. These files are normally created as the primary output of a PLA design program. The second method is to include the data directly in the SPICE command line or, for schematics, in the DATA attribute.

SPICE format

U<*name*> <*pld type*>(<*no. of inputs*>,<*no. of outputs*>)
+<*digital power node*> <*digital ground node*>
+<*input node*>* <*output node*>*
+<*timing model name*> <*I/O model name*>
+[FILE=<"*file name constant*" | |*file name expression*|>]
+[DATA=<*data constant*> | <*radix flag*>$<*program data*>$]
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Schematic format

PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
PL_04

I/O MODEL attribute
<*I/O model name*>

Example
IO_ACT

FILE attribute
<"*file name constant*" | |*file name expression*|>

Examples
"JED_FILE" ;a file name constant enclosed in "".
|FILEVAR| ;a file name expression enclosed in ||.

DATA attribute
<*data constant*> | <*radix flag*>$<*program data*>$

Examples
data_table ; data_table is defined elsewhere with a .define statement.
b$010101 ; multiline program data
+101011
+011001$

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor fields
Inputs <*number of inputs*>
Outputs <*number of outputs*>
The Component editor has two special fields for PLA devices, 'Inputs' and 'Outputs'. When you enter these values, MC7 places the specified input and output pins on the display and you must drag them into place on the shape.

Definitions
<*pld type*> selects the type of gate and is chosen from the following list:

Type	**Description**
PLAND	AND array
PLOR	OR array
PLNAND	NAND array
PLNOR	NOR array
PLXOR	Exclusive OR array
PLNXOR	Exclusive NOR array
PLANDC	AND array with true and complement inputs
PLORC	OR array with true and complement inputs
PLNANDC	NAND array with true and complement inputs
PLNORC	NOR array with true and complement inputs
PLXORC	Exclusive OR array with true and complement inputs
PLNXORC	Exclusive NOR array with true and complement inputs

Table 23-15 PLA Types

FILE = <"*file name constant*" | |*file name expression*| >
This attribute denotes the name of a JEDEC format file in one of two formats:

File name constant enclosed in double quotes ("")
File name expression enclosed in double bars (||)

SPICE text files and libraries can use either |*file name expression*| or "*file name constant*". |*file name expression*| is limited to a single <*text name*> from a subcircuit TEXT: statement. This is the way a commercial PLD part modeled with a PLA device in a subcircuit is passed a JEDEC file name. For example:

```
.SUBCKT PLD24 I1 I2 O1 O2 O3 O4
+ TEXT: JFILE="JD.STM"
...
U1 PLOR(2,4)
+ $G_DPWR $G_DGND
+ I1 I2
+ O1 O2 O3 O4
+ PLAMODEL
+ IO_STD_PLD
+ FILE=|JFILE|
```

Here, the PLA will use the file name assigned to JFILE. JFILE can be assigned a name when the subckt is used in a schematic or a SPICE file by supplying a FILE attribute. If is not assigned a value, JFILE will use the default value "JD.STM".

Schematic PLA devices that use the FILE attribute must use the "*file name constant*" option as there is no way to define a |*file name expression*| in a schematic.

If the FILE attribute is used, the DATA attribute is ignored. The mapping of the data in the JEDEC file is controlled by timing model parameters.

The data constant lets you create the data table in the text area or as grid text in the schematic where there is more space. It is created with a .define statement in the schematic or in the text area. The contents of the define statement are substituted for the data constant when an analysis is run. For example, consider this define statement:

.DEFINE TAB1 B$01 01 10 11 01 11 01 10 01$

If a PLA uses a data constant of TAB1, the text "B$01 01 10 11 01 11 01 10 01$" will be substituted for TAB1 when an analysis is run. 'Defines' may be used to move long definitions from any attribute VALUE field (except PART) to the text area, or just to create convenient compact constants.

<*radix flag*>$
This is a one character flag that defines the type of data in the DATA attribute. It is one of the following:

B Binary data
O Octal data (most significant bit =lowest address)
X Hex data (most significant bit =lowest address)

<*program data*>$
This text string contains data values that program the PLA. If the data value for a particular input column variable is a '0', the input column is not connected to the gate. If the data value is a '1', the input column is connected to the gate. The phrase "connected" means that a new input to the gate is created and the input (column) line is connected to the new gate input. The data values start at the zero address. For example:

```
U1 PLOR(3,4)           ;3-INPUT,4-OUTPUT OR TYPE PLA
+ $G_DPWR $G_DGND      ;POWER PINS
+ I1 I2 I3             ;THREE INPUT PIN NAMES
+ O1 O2 O3 O4          ;FOUR OUTPUT PIN NAMES
+ PLAMODEL             ;PLA TIMING MODEL NAME
+ IO_STD_PLD           ;PLA I/O MODEL NAME
+ DATA=B$              ;PLA DATA PROGRAM
+ 1 1 1                ;O1 = I1 | I2 | I3
+ 0 1 0                ;O2 = I2
+ 1 0 1                ;O3 = I1 | I3
+ 0 0 1$               ;O4 = I3
...
```

True only(PLAND, PLOR, PLNAND, PLNOR, PLXOR, PLNXOR)
The zero address specifies the first input to the first output gate. The next input specifies the second input to the first output gate, and so on until all possible input connections have been specified. The process repeats for all possible input connections to the second gate until the last gate has been programmed.

True and complement(PLANDC, PLORC, PLNANDC, PLNORC, PLXORC, PLNXORC)
True and complement programming is like true only programming, except that the complement is interleaved and placed after the true. The zero address programs the first true input to the first output gate. The next input programs the connection of the first complement input to the first gate. The next input programs the connection of the second true input to the first output gate. The next input programs the connection of the second complement input to the first gate, and so on until all possible true and complement input connections have been programmed. The process is repeated for all possible true and complement input connections to the second gate until the final gate has been programmed.

The data values must be delimited by dollar signs ($). They may be separated by spaces or placed on continuation lines.

Model statement form
.MODEL <*timing model name*> UPLD ([*model parameters*])

Example
.MODEL PLD1 UPLD (TPLHMN=10ns TPLHTY=25ns TPLHMX=35ns)

Parameter	**Description**	**Default**
TPLHMN	Delay: min low to high	0
TPLHTY	Delay: typ low to high	0
TPLHMX	Delay: max low to high	0
TPHLMN	Delay: min high to low	0
TPHLTY	Delay: typ high to low	0
TPHLMX	Delay: max high to low	0
OFFSET	JEDEC file mapping: Address of first input and first gate program	0
COMPOFFSET	JEDEC file mapping: Address of complement of first input and first gate program	1
INSCALE	JEDEC file mapping: Amount the JEDEC file address changes for each new input pin	true only:1 true & comp:2
OUTSCALE	JEDEC file mapping: Amount the JEDEC file address changes for each new output pin (gate)	true only:<no. of inputs> true & comp:2*<no. of inputs>

Table 23-16 PLA timing model parameters

Multi-bit A/D converter

SPICE format

U<*name*> ADC(<*bits*>)
+<*digital power node*> <*digital ground node*>
+<*in node*><*ref node*> <*gnd node*> <*convert node*>
+<*status node*> <*over-range node*>
+<*output msb node*> <*output lsb node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Example
U10 ADC(8) $G_DPWR $G_DGND
+analog_in reference 0 convert status over B7 B6 B5 B4 B3 B2 B1 B0
+IO_STD_OC_ST

Schematic format

PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
AF_04

I/O MODEL attribute
<*I/O model name*>

Example
IO_AC

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor fields
Bits <*bits*>
The Component editor has a special 'Bits' field for ADC devices. It lets you specify the number of output bits for the ADC. When you enter the number of bits, the system places the following pins on the display and you must drag them into place on the shape.

In
Convert
Ref
Gnd
Status
Over-range
Out0
Out1
.
.
.
OutN-1

If N bits are specified there will be N output pins Out0...OutN-1.

The ADC device converts the analog voltage difference between the <*in node*> pin and the <*gnd node*> pin to an equivalent digital representation. The digital output is the binary equivalent of:

$$\frac{V(\textit{in node, gnd node})^{2^{<bits>}}}{V(\textit{ref node, gnd node})}$$

If the analog input value, V(*in node, gnd node*), is less than zero, then all data bits are set to '0', and over-range is set to '1'. If this value exceeds V(*ref node, gnd node*) then all data bits are set to '1', and the over-range pin is again set to '1'.

The sampling of the input voltage occurs on the rising edge of the CONVERT signal. Only one conversion is performed per rising edge.

The data output pins go to the 'X' state and the STATUS pin goes to the '1' state TPCS seconds after the rising edge of the CONVERT signal. TPSD seconds later, the data output pins change to valid data. TPDS seconds later, the STATUS pin goes to the '0' state. This timing is summarized in the diagram below:

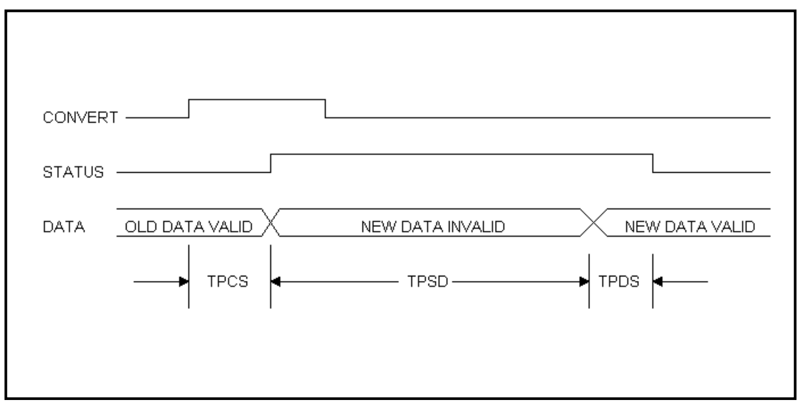


Figure 23-2 ADC timing diagram

Model statement form
.MODEL <*timing model name*> UADC ([*model parameters*])

Example
.MODEL A1 UADC (TPCSMN=5ns TPCSTY=15ns TPCSMX=25ns)

Parameter	**Description**	**Units**
TPCSMN	Delay: min rising edge of Convert to rising edge of Status	sec
TPCSTY	Delay: typ rising edge of Convert to rising edge of Status	sec
TPCSMX	Delay: max rising edge of Convert to rising edge of Status	sec
TPSDMN	Delay: min rising edge of Status to valid data outputs	sec
TPSDTY	Delay: typ rising edge of Status to valid data outputs	sec
TPSDMX	Delay: max rising edge of Status to valid data outputs	sec
TPDSMN	Delay: min data outputs valid to falling edge of Status	sec
TPDSTY	Delay: typ data outputs valid to falling edge of Status	sec
TPDSMX	Delay: max data outputs valid to falling edge of Status	sec

Table 23-17 ADC timing model parameters

See the sample circuit AD16 for an example of the use of this type of device.

Multi-bit D/A converter

SPICE format

U<*name*> DAC(<*no. of bits*>)
+<*digital power node*> <*digital ground node*>
+<*out node*> <*ref node*> <*gnd node*>
+<*input msb node*> <*input lsb node*>
+<*timing model name*> <*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]

Example
U10 DAC(8) $G_DPWR $G_DGND
+analog_out reference 0 B7 B6 B5 B4 B3 B2 B1 B0
+ D0_EFF IO_STD_ST

Schematic format

PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
DACTM

I/O MODEL attribute
<*I/O model name*>

Example
IO_HCT

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor fields
Bits <*bits*>
The Component editor has a special Bits field for DAC devices. It lets you specify the number of input bits. When you enter the number of bits, the system places the following pins on the display and you must drag them into place on the shape.

Out
Ref
Gnd
In0
In1
.
.
.
In<bits-1>

The DAC device converts the decimal value of the binary inputs to an analog voltage and impresses that voltage between the <*out node*> and the <*gnd node*>. The analog output for n input bits, bn-1,...b2, b0 is:

$$V(out) = V(ref,gnd)\ (b_{n-1}^{\ n-1}+...+b_1^{\ 1}+b_0^{\ 0})\ 2^{-n}$$

Binary inputs in the 'X' state translate to analog voltages halfway between the analog voltage for the '0' state and the '1' state.

Following a change on one or more of the binary inputs, the analog output changes linearly from the old state to the new state over the specified TSW time as shown in the timing diagram below.

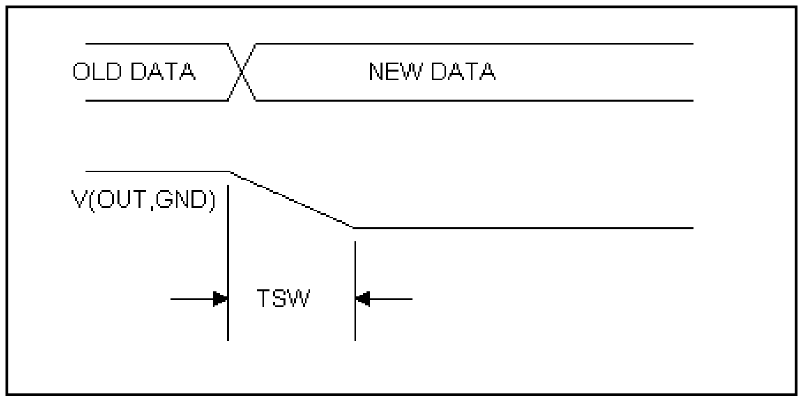


Figure 23-3 DAC timing diagram

See the sample circuit AD16 for an example of the use of this type of device.

Model statement form

.MODEL <*timing model name*> UDAC ([*model parameters*])

Example

.MODEL DAC1 UDAC (TSWMN=5ns TSWTY=15ns TSWMX=25ns)

Parameter	**Description**	**Units**
TSWMN	Switching time: min data change to stable analog out	sec
TSWTY	Switching time: typ data change to stable analog out	sec
TSWMX	Switching time: max data change to stable analog out	sec

Table 23-18 DAC timing model parameters

Behavioral primitives

Behavioral primitives are devices for simplifying the modeling of complex digital parts. They come in three varieties:

Logic expressions
These devices let you describe digital behavior with Boolean logic expressions.

Pin-to-pin delays
These delays let you describe rules governing the propagation delays between pins. The rules are logic expressions based upon the behavior of input pins.

Constraints
These devices let you check for timing constraints and issue warning messages when the constraints are violated. Timing constraints include pulse width, pulse frequency, setup time, hold time, and a general user-defined type.

These devices are used extensively in the Digital library to model commercial logic family parts, so it is not necessary to master their use in order to benefit from them. It is possible to use the digital logic family parts without ever learning about the behavioral primitives. However, if you want to model a part that is not in the library, you will find these primitives to be of great value and well worth the time spent learning them.

Logic expression

The logic expression primitive provides a means for describing complex digital behavior. It lets you define the behavior of outputs with standard Boolean algebra, using as variables, input states, temporary states, and output states.

SPICE format
U<*name*> LOGICEXP(<*no. of inputs*>,<*no. of outputs*>)
+<*digital power node*> <*digital ground node*>
+<*first input node*>...<*last input node*>
+<*first output node*>...<*last output node*>
+<*timing model name*>
+<*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]
+ LOGIC:<*logic assignments*>*

Schematic format
Logic expressions are usually found in the text file libraries. They are not often used directly as schematic components, although they can be. For this reason, only a few sample logic expression components are included in the Component library.

PART attribute
<*name*>

Example
U10

TIMING MODEL attribute
<*timing model name*>

Example
DLOGIC

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

LOGIC attribute
LOGIC:{<*logic expression*>}

This attribute is a full LOGIC statement or, more commonly, the name of a LOGIC statement defined with a .define statement in the text area. Note that the 'LOGIC:' keyword must precede the {<*logic expression*>}.

Examples
LOFEXP1 ; must be defined by a .define in the text area
LOGIC: C= {A | B & C } ; note the mandatory use of the 'LOGIC:' keyword

LOGIC: TEMP11 = {IN1 ^ IN2 & IN3 ^ IN4 }
LOGIC: TEMP12 = {IN1 ^ IN2 | IN3 ^ IN4 }
LOGIC: OUT1 = { TEMP11 & TEMP12 }

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor fields

Pins <*input pins, output pins*>

The Component editor has a special 'Pins' field for these devices. You don't edit the fields directly. Instead, you click in the Shape/Pin Display and add an input or an output pin. This is how you define the names and the locations of the pins used by the logic expression device.

Note that these devices are used mainly for modeling commercial parts, and are found principally in subcircuits in the Digital library text files. Because they can have a variable number of inputs and outputs, there are an infinite number of pin combinations, and since a component in the Component library requires the specification of pin placements, placing all possible logic expression primitives in the library is clearly impossible. Accordingly, only a few such devices are found in the library, and these are mainly for illustration. These devices are really targeted for use in the subcircuits of the Digital Library section. While you can use them directly in schematics, their real power derives from their use in commercial part models, which can be readily placed and used in schematics.

Format definitions

LOGIC:

Marks the start of a group of one or more <*logic assignments*>. A <*logic assignment*> is one of the following forms:

<*temporary value*> = {<*logic expression*>}
<*output node name*> = {<*logic expression*>}

<*temporary value*>

Any use of a name which is not in the list of the pin names for the device, automatically creates a temporary value which may be used in other <*logic assignments*>. The use of temporary values substantially simplifies and improves the readability of logic expressions, thus reducing errors.

<*output node name*>

This is one of the output node names. Assignment to an <*output node name*> causes the result of the <*logic expression*> evaluation to be scheduled after the appropriate delay from the timing model, to the output node.

<*logic expression*>

This is a C-like, infix-notation expression which generates one of the five valid digital logic levels { 0, 1, R, F, X }. The expression must be enclosed in curly braces {...}. The <*logic expression*> may use the continuation character (+) to span more than one line.

Logic expression operators and their precedence are as follows:

Operator	**Definition**	**Precedence**
~	Unary negation	1
&	AND	2
^	Exclusive OR	3
|	OR	4

Table 23-19 Logic expression operators

Operands are one of the following:
<*input nodes*>
Previously assigned <*temporary values*>
Previously assigned <*output nodes*>
Constants: '0, '1, 'R, 'F, and 'X. Note that the mandatory single quote (') is a part of the constant name.

Parentheses may be freely used to group expressions.

Timing model format
The timing model is identical to the standard gate primitive UGATE format:

.MODEL <*timing model name*> UGATE ([*model parameters*])

Simulation behavior
The LOGIC statement is evaluated when any input or output pin changes state. Each <*logic assignment*> is evaluated in the order listed within the LOGIC statement. Expressions have no delay. Any <*output node name*> that changes is scheduled with the appropriate delay from the timing model. Note that usually the timing model delays are set to zero. Instead, delay is modeled by having the outputs drive the inputs of a pin-to-pin delay device. These devices provide logical control of delay based upon inputs that make implementing data sheet delays easy. Internal feedback is forbidden. This means a <*logic assignment*> should only use previously assigned <*temporary values*> or <*output nodes*>.

Here is an example of a logic expression used in the Digital Library to model the 7483A full adder.

```
* -----7483A ------
* 4-Bit Binary Full Adders With Fast Carry
*The TTL Logic Data Book, 1988, TI Pages 2-257 - 2-261
U1LOG LOGICEXP(9,5) DPWR DGND
+   C0 A1 A2 A3 A4 B1 B2 B3 B4
+   S1_O S2_O S3_O S4_O C4_O
+   D0_GATE IO_STD MNTYMXDLY={MNTYMXDLY}  IO_LEVEL={IO_LEVEL}
+
+ LOGIC:
+   c0bar = {~C0}
+   nor1 = {~(A1 | B1)}
+   nand1 = {~(A1 & B1)}
+   nor2 = {~(A2 | B2)}
+   nand2 = {~(A2 & B2)}
+   nor3 = {~(A3 | B3)}
+   nand3 = {~(A3 & B3)}
+   nor4 = {~(A4 | B4)}
+   nand4 = {~(A4 & B4)}
+   C4_O = {~(nor4 | (nor3 & nand4) | (nor2 & nand4 & nand3) |
+    (nor1 & nand4 & nand3 & nand2) |
+   (nand4 & nand3 & nand2 & nand1 & c0bar))}
+   S4_O = {(nand4 & (~nor4)) ^ (~(nor3 | (nor2 & nand3)
+   | (nor1 & nand3 & nand2) |
+   (nand3 & nand2 & nand1 & c0bar)))}
+   S3_O = {(nand3 & (~nor3)) ^ (~(nor2 | (nor1 &
+     nand2) | (nand2 & nand1 & c0bar)))}
+   S2_O = {(nand2 & (~nor2)) ^ (~(nor1 | (nand1 &
+     c0bar)))}
+   S1_O = {(nand1 & (~nor1)) ^ C0}
```

This shows just the logic expression portion of the 7483A model. Complete models for commercial digital devices like the 7483A may include logic expressions, pin-to-pin delays, constraint devices, and other logic primitives like gates and flip-flops. These are all housed in a subcircuit using the commercial part name, like 7483A.

Pin-to-pin delay

The PINDLY primitive provides a way of modeling complex conditional pin delays. Conceptually, it can be pictured as a set of channels, where the delay through each is controlled by a set of user-defined rules. Rules assign a delay to a particular channel based upon activity at the pins. A typical rule might say:

OUT1 = {CHANGED(REF1,0) & TRN_LH, DELAY(10NS,15NS,18NS)}

This means, "If the REF1 pin changed in the last 0 seconds, and the channel *node* OUT1 is making a low to high transition, then the delay for the OUT1 channel is min=10ns, typ=15ns, max=18ns."

PINDLY primitives are typically used within a subckt between the logic expression outputs and the subckt outputs. That is, the inputs to a PINDLY are usually the outputs of a logic expression. The outputs of a PINDLY are usually the outputs of the subcircuit which embodies the full digital model of the part.

A PINDLY may use multiple conditional rules to determine the delay on each channel. Reference and enable pin states may be used in the rule logic.

SPICE format

U<*name*> PINDLY(<*no. of paths*>,<*no. of enables*>,<*no. of refs*>)
+<*digital power node*> <*digital ground node*>
+<*first input node*>...<*last input node*>
+<*first enable node*>...<*last enable node*>
+<*first reference node*>...<*last reference node*>
+<*first output node*>...<*last output node*>
+<*I/O model name*>
+[MNTYMXDLY=<*delay select value*>]
+[IO_LEVEL=<*interface subckt select value*>]
+[BOOLEAN:<*boolean assignments*>*]
+[PINDLY:<*delay assignments*>*]
+[TRISTATE: ENABLE LO | HI <*enable node*> <*delay assignments*>*]
+[SETUP_HOLD:<*setup_hold specifications*>]
+[WIDTH:<*width specifications*>]
+[FREQ:<*freq specifications*>]
+[GENERAL:<*general specifications*>]

Schematic format
PINDLYs are most commonly found in the text file libraries. They are not often used directly as schematic components, although they can be. For this reason, only a few token PINDLY components are found in the Component library.

PART attribute
<*name*>

Example
P20

PINDELAY attribute
[BOOLEAN:<*boolean assignments*>*]
+[PINDLY:<*delay assignments*>*]
+[TRISTATE: ENABLE LO | HI <*enable node*> <*delay assignments*>*]

This attribute is a full PINDELAY statement or, more commonly, the name of a PINDELAY statement defined with a .define statement in the text area or in a file referenced by a .lib statement in the schematic.

Examples
CARRY_DELAYS ; defined with a .define statement in the text area

PINDLY: OUT1= { (DELAY (10NS,20NS,30NS)) }

PINDLY: BIT2= { CASE (CHANGED(REF1,O),DELAY (10NS,20NS,30NS)) }

BOOLEAN:
+READY={CHANGED_LH(GATE,0)}
+LOW={CHANGED_LH(SEC,0)}
+PINDLY: BIT2 = { CASE (READY & LOW),
+ DELAY (10NS,20NS,30NS)) }

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

MNTYMXDLY attribute
<*delay select value*>

Example
1

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor pin fields
Pins <*paths, enable pins, reference pins*>
The Component editor has a special Pins field for PINDLY devices. You don't edit these fields directly. Instead, you click in the Shape/Pin Display and add a path, enable pin, or a reference pin. Paths have two pins, with an arrow between them. The arrow is drawn from the path input to the path output. This defines the names and locations of the pins that the PINDLY device uses.

PINDLY devices are used mainly for modeling commercial parts, and are found principally in subcircuits in the Digital Library text files. Because they can have an infinite number of pin combinations, and since a component in the Component library requires the specification of pin placements, placing all possible PINDLY devices in the library is clearly impossible. Only a few such devices are found in the library, and these are mainly for illustration. These devices are really targeted for use in text file subcircuits. While you can use them directly in schematics, their real power derives from their use in commercial part models.

Format definitions

BOOLEAN:

Marks the start of a group of one or more <*boolean assignments*> used as temporary variables in subsequent <*delay assignments*>. A <*boolean assignment*> is one of the following forms:

<*boolean variable*> = {<*boolean expression*>}
<*boolean variable*> = any valid variable name

<*boolean expression*>

This is a C-like, infix-notation expression which generates one of the two boolean states, TRUE or FALSE. The expression must be enclosed in curly braces {...}. The <*boolean expression*> may use the continuation character (+) to span more than one line. Boolean expression operators and their precedence are as follows:

Operator	**Definition**	**Precedence**
~	Unary negation	1
= =	Equality	2
!=	Inequality	3
&	AND	4
^	Exclusive OR	5
|	OR	6

Table 23-20 Boolean expression operators

Operands are one of the following:

Previously assigned <*boolean variables*>
Reference functions
Transition functions
<*boolean constants*> (TRUE or FALSE)
Logic levels ,('0, '1, 'R, 'F, 'X) may be used as operands for the '==' and '!=' operators only. This lets the boolean logic examine input states as in, "START={COUNT=='R & RESET=='0}". Note that the mandatory single quote (') is a part of the constant name.

Reference functions

Reference functions provide the means to detect logic level transitions on <*reference nodes*> and <*input nodes*>. They return boolean values and may be used within any <*boolean expressions*>. The functions are:

CHANGED(<*node*>,<*reference time*>)
CHANGED_LH(<*node*>,<*reference time*>)
CHANGED_HL(<*node*>,<*reference time*>)

The CHANGED function returns TRUE if <*node*> has made a transition in the last <*reference time*> seconds, relative to the current time.

The CHANGED_LH function returns TRUE if <*node*> has made a low to high transition in the last <*reference time*> seconds, relative to the current time.

The CHANGED_HL function returns TRUE if <*node*> has made a high to low transition in the last <*reference time*> seconds, relative to the current time.

Reference functions consider only the last or current transition. They do not consider all transitions within the last <*reference time*> seconds.

<*reference time*> may be zero, in which case only current transitions at the evaluation time are considered.

Transition functions

These functions let you detect changes on the <*output nodes*> for which the <*delay expression*> is being evaluated. They return boolean values. They take no arguments, and refer implicitly to the changes on the <*output nodes*> at the current time. The functions are of the format:

TRN_pc...where p refers to the prior state and c refers to the current state.

State values are chosen from the set {L, H, Z, $}. The '$' state refers to the 'don't care' state. Thus TRN_L$ returns TRUE for a transition from the low state to any other state. Note that the states p and c must be different. There is no TRN_HH function, for instance. The complete list of functions is:

TRN_HL	TRN_HZ	TRN_H$
TRN_LH	TRN_LZ	TRN_L$
TRN_ZL	TRN_ZH	TRN_Z$
TRN_$H	TRN_$L	TRN_$Z

Transition functions using the Z state should be used only within TRISTATE sections. Open collector transitions should be modeled with the TRN_LH and TRN_HL functions.

PINDLY:
Marks the start of a group of one or more <*delay assignments*> used to assign path delays to the PINDLY input / output channels. A <*delay assignment*> is of the following form:

<*output node*>* = { <*delay expression*> }

<*output node*>
One or more of the PINDLY <*output node*> names mentioned in the list, <*first digital output node*>...<*last digital output node*>. Several outputs may share the same rules by including them on the left side separated by spaces or commas.

<*delay expression*>
An expression which returns a set of three delay values (min,typ,max) for use on the specified output delay channel. There are two forms:

DELAY(<*min*>,<*typ*>,<*max*>)
This form simply specifies the delays directly. For example:
...
+PINDLY: OUT1 , OUT2 = { DELAY(10ns, -1, 20ns) }
...

Note the use of -1 causes the system to calculate the value using the unspecified propagation delay rules.

The second form uses a more complex CASE statement. Its form is:

CASE(<*boolean expression_1*>,<*delay expression_1*>, ;rule 1
<*boolean expression_2*>,<*delay expression_2*>, ;rule 2
...
<*boolean expression_n*>,<*delay expression_n*>, ;rule n
<*default delay expression*>)

The CASE statement is comprised of a set of one or more rules. Each rule has a <*boolean expression*> and a <*delay expression*> and is evaluated in the order listed in the CASE statement. The first <*boolean expression*> that returns TRUE causes the <*delay expression*> to be assigned to the <*output nodes*>. If no <*delay expression*> returns TRUE, the <*default delay expression*> is assigned to the

<*output nodes*>. Since this situation can easily occur, the <*default delay expression*> is mandatory.

Example

```
+ BOOLEAN:
+   DATA = { CHANGED(DI1,0) | CHANGED(DI2,0) |
+          CHANGED(DI3,0) | CHANGED(DI4,0)}
+   SELECT = {CHANGED(S1BAR,0) | CHANGED(S2,0)}
+   CLEAR = {CHANGED_HL(CLRBAR,0)}
+   STROBE = {CHANGED_HL(STB,0)}
+ PINDLY:
+   B1 B2 B3 B4 = {
+    CASE(
+     DATA & STROBE & TRN_HL, DELAY(-1,16ns,25ns),
+     CLEAR, DELAY(-1,14ns,22ns),
+     SELECT & TRN_LH, DELAY(-1,12ns,20ns),
+     DELAY(-1,17ns,26ns))}
```

This example describes the delays on the four outputs B1...B4. They share four delay rules.

Rule 1: DATA & STROBE & TRN_HL, DELAY(-1,16ns,25ns)
Meaning: If DATA is TRUE (if any of the inputs DI1...DI4 have changed) and STROBE is TRUE (STB made a high to low transition), then the delay is (-1, 16ns, 25ns) for any output (B1...B4) making a high to low transition.

Rule 2: CLEAR, DELAY(-1,14ns,22ns)
Meaning: If CLEAR is TRUE (if CLRBAR has made a high to low transition), then the delay is (-1, 14ns, 25ns) for any of the outputs (B1...B4) making any transition.

Rule 3: SELECT & TRN_LH, DELAY(-1,12ns,20ns)
Meaning: If SELECT is TRUE (if either S1BAR or S2 has changed), then the delay is (-1, 12ns, 20ns) for any output (B1...B4) making a low to high transition.

Rule 4: DELAY(-1,17ns,26ns)
Meaning: If rules 1, 2, and 3 fails, then the delay is (-1, 17ns, 26ns) for any of the outputs (B1...B4) making any transition.

TRISTATE:
Marks the start of a group of one or more tri-state <*delay assignments*> used to assign path delays to tri-state input / output channels. Unlike a PINDLY, a TRISTATE is controlled by an <*enable node*>.

Following the TRISTATE: keyword, an <*enable node*> and its polarity are specified. The polarity is specified by choosing one of two keywords:

ENABLE LO	means low state is the enabled state
ENABLE HI	means high state is the enabled state

The <*enable node*> controls all <*output nodes*> in the current TRISTATE section.

The <*delay expressions*> in a TRISTATE section may employ the Z-state transition functions.

Simulation behavior
The PINDLY statement is evaluated when any input, enable, or output pin changes state. Each <*input node*> is associated with a corresponding <*output node*> in the same order of occurrence on the line. The BOOLEAN sections are evaluated first, then the PINDLY and TRISTATE sections are evaluated and the delays for changing <*output nodes*> calculated. Changing <*output nodes*> are then scheduled for a state change to the new <*input node*> state after the appropriate delay.

Example
This example shows a mixture of BOOLEAN, PINDLY, and TRISTATE sections:

```
U4DLY PINDLY(9,1,13) DPWR DGND
+   Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 INTBAR_O
+   OE
+   STB M S1BAR S2 CLRBAR DI1 DI2 DI3 DI4 DI5 DI6 DI7 DI8
+   DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 INTBAR
+   IO_S MNTYMXDLY={MNTYMXDLY} IO_LEVEL={IO_LEVEL}
+
+ BOOLEAN:
+   DATA = {CHANGED(DI1,0) | CHANGED(DI2,0) |
+         CHANGED(DI3,0) | CHANGED(DI4,0) |
+      CHANGED(DI5,0) | CHANGED(DI6,0) |
```

```
+    CHANGED(DI7,0) | CHANGED(DI8,0)}
+   SELECT = {CHANGED(S1BAR,0) | CHANGED(S2,0)}
+   MODE = {CHANGED(M,0)}
+   CLEAR = {CHANGED_HL(CLRBAR,0)}
+   STROBE = {CHANGED(STB,0)}
+
+ TRISTATE:
+   ENABLE HI=OE
+   DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 = {
+    CASE(
+     CLEAR & TRN_HL, DELAY(-1,18ns,27ns),
+     (STROBE | SELECT) & TRN_LH, DELAY(-1,18ns,27ns),
+     (STROBE | SELECT) & TRN_HL, DELAY(-1,15ns,25ns),
+     DATA & TRN_LH, DELAY(-1,12ns,20ns),
+     DATA & TRN_HL, DELAY(-1,10ns,20ns),
+     TRN_ZH, DELAY(-1,21ns,35ns),
+     TRN_ZL, DELAY(-1,25ns,40ns),
+     TRN_HZ, DELAY(-1,9ns,20ns),
+     TRN_LZ, DELAY(-1,12ns,20ns),
+     DELAY(-1,26ns,41ns))}
+
+ PINDLY:
+   INTBAR = {
+    CASE(
+     STROBE & TRN_HL, DELAY(-1,16ns,25ns),
+     SELECT & TRN_LH, DELAY(-1,12ns,20ns),
+     DELAY(-1,17ns,26ns))}
```

Constraint checker

The CONSTRAINT primitive provides a way of checking complex conditional timing parameters. It lets you check setup and hold time, pulse width and frequency, and includes a general user-defined check to model unique requirements. Constraints usually accept as inputs the part or subcircuit inputs and typically work in concert with logic expressions, pin-to-pin delays, and other digital primitives. Constraints are passive warning devices. They create timing violation messages and place them in the Numeric Output window. They normally do not affect simulation results. The I/O model attribute is required only to handle the case of an accidental connection of a Constraint digital node to an analog node.

SPICE format

U<*name*> CONSTRAINT(<*no. of inputs*>)
+<*digital power node*> <*digital ground node*>
+<*first digital input node*>...<*last digital input node*>
+<*I/O model name*>
+[IO_LEVEL=<*interface subckt select value*>]
+[BOOLEAN:<*boolean assignments*>]
+[SETUP_HOLD:<*setup_hold specifications*>]
+[WIDTH:<*width specifications*>]
+[FREQ:<*freq specifications*>]
+[GENERAL:<*general specifications*>]

Schematic format

Constraints are usually found in the text file libraries. They aren't often used directly as schematic components, although they can be. For this reason, only a few sample constraint components are to be found in the Component library.

PART attribute
<*name*>

Example
C20

CONSTRAINT attribute
[BOOLEAN:<*boolean assignments*>]
+[SETUP_HOLD:<*setup_hold specifications*>]
+[WIDTH:<*width specifications*>]
+[FREQ:<*freq specifications*>]
+[GENERAL:<*general specifications*>]

This attribute is a full CONSTRAINT statement or, more commonly, the name of a CONSTRAINT statement defined with a .define statement in the text area.

Examples
C381_STD ;defined in the text area
WIDTH: NODE = MRBAR MIN_LO = 5n
FREQ: NODE = CP MAXFREQ = 130MEG

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Special Component editor pin fields
Inputs <*inputs*>
The Component editor has a special 'Inputs' field for CONSTRAINTs. You don't edit these fields directly. Instead, you click in the Shape/Pin Display and add an input pin. This lets you define the names and pin locations of the input pins used by the constraint device.

Constraint devices are used mainly for modeling commercial parts, and are found principally in subcircuits in the Digital Library text files. Because they can have a large number of input pins, and since a component in the Component Library requires the input pin placements, including all possible constraint devices in the library is not feasible. Only a few such devices are to be found in the library, and they are mainly for illustration. These devices are really targeted for use in text file subcircuits. While you can use them directly in schematics, their real power issues from their use in the models of commercial digital parts.

Format definitions

BOOLEAN:

Marks the start of a group of one or more <*boolean assignments*> used as temporary variables in subsequent <*specifications*>. A <*boolean assignment*> is of the following form:

<*boolean variable*> = {<*boolean expression*>}

<*boolean expression*>

This is the same as in the PINDLY, with the singular exception that the transition functions (TRN_pc) are not available.

SETUP_HOLD:

Marks the start of a <*setup_hold specification*>. The format is as follows:

+SETUP_HOLD:
+CLOCK <*assertion edge*> = <*input node*>
+DATA(<*no. of data inputs*>) = <*first input node*>...<*last input node*>
+[SETUPTIME=<*time value*>]
+[HOLDTIME=<*time value*>]
+[WHEN=<*boolean expression*>]
+[MESSAGE="<*extra message text*>"]
+[ERRORLIMIT=<*limit value*>]

The CLOCK argument <*input node*> defines the *input node* that serves as a reference for the setup/hold specification. The CLOCK <*assertion edge*> is one of the following:

LH This specifies that the clock low to high edge is the assertion edge.
HL This specifies that the clock high to low edge is the assertion edge.

The DATA arguments <*first input node*>...<*last input node*> specify one or more input nodes whose setup or hold time is to be measured. There must be at least one input node. The arguments must be separated by a space or a comma.

The SETUPTIME argument <*time value*> specifies the minimum time that all

DATA <*input nodes*> must be stable prior to the <*assertion edge*> of the clock. The <*time value*> must be positive or zero. If an <*input node*> has a setup time that depends upon whether the data is LO or HI, then you can use one of these specialized forms:

SETUPTIME_LO=<*time value*>
This is the setup time for a low data state prior to the clock <*assertion edge*>.

SETUPTIME_HI=<*time value*>
This is the setup time for a high data state prior to the clock <*assertion edge*>.

If either of these setup time specifications is zero, MC7 will skip its check.

The HOLDTIME argument <*time value*> specifies the minimum time that all DATA <*input nodes*> must be stable after the <*assertion edge*> of the clock. The <*time value*> must be positive or zero. If an <*input node*> has a hold time that depends upon whether the data is LO or HI, then you can use one of these:

HOLDTIME_LO=<*time value*>
This is the hold time for a low data state after the clock <*assertion edge*>.

HOLDTIME_HI=<*time value*>
This is the hold time for a high data state after the clock <*assertion edge*>.

How it works
The evaluation begins when the specified *Clock node* experiences the specified <*assertion edge*>. The WHEN <*boolean expression*> is evaluated and if TRUE, all SETUPTIME and HOLDTIME blocks with nonzero values are checked during this clock cycle. If the WHEN evaluates to FALSE, then no checks are done for this clock cycle. The WHEN function disables checking when it is inappropriate, as for example, during a RESET or PRESET operation.

Setup time checks occur at the CLOCK <*assertion edge*>. If the specified hold time is zero, simultaneous CLOCK and DATA transitions are allowed, but the previous DATA transition is checked for setup time. If the hold time is not zero, simultaneous CLOCK and DATA transitions are reported as errors. Hold time checks are done on any DATA input that changes after the CLOCK <*assertion edge*>. If the setup time is zero, simultaneous CLOCK and DATA transitions are allowed, but the next DATA transition occurring before the non-asserting edge is checked for hold time. If the setup time is not zero, simultaneous CLOCK and DATA transitions are reported as errors.

If either the CLOCK node or the DATA node are unknown, the check is skipped to avoid the profusion of errors that usually result. If a clock is X when DATA changes or vice-versa, the error has probably already been flagged elsewhere.

The constraint is checked only if <*boolean expression*> is true. If the number of constraint warning messages exceeds <*limit value*>, no further messages are issued. <*extra message text*> if any, is added to the standard error message.

WIDTH:
Marks the start of a group of one or more <*width specifications*> which have the following format:

+WIDTH:
+NODE=<*input node*>
+[MIN_LO=<*time value*>]
+[MIN_HI=<*time value*>]
+[WHEN=<*boolean expression*>]
+[MESSAGE="<*extra message text*>"]
+[ERRORLIMIT=<*limit value*>]

The NODE argument, <*input node*>, defines the node whose input is to be checked for width.

The MIN_LO argument, <*time value*>, defines the minimum time that <*input node*> can be at the 0 level. If <*time value*> is specified as zero, the width check is not performed.

The MIN_HI argument, <*time value*>, defines the minimum time that <*input node*> can be at the 1 level. If <*time value*> is specified as zero, the width check is not performed.

At least one of the two forms, MIN_LO or MIN_HI must be present inside every WIDTH specification.

The constraint is checked only if <*boolean expression*> is true. If the number of constraint warning messages exceeds <*limit value*>, no further messages are issued. <*extra message text*> if any, is added to the standard error message.

FREQ:
Marks the start of a group of one or more <*freq specifications*> which have the following format:

+FREQ:
+NODE = <*input node*>
+[MINFREQ=<*freq value*>]
+[MAXFREQ=<*freq value*>]
+[WHEN=<*boolean expression*>]
+[MESSAGE="<*extra message text*>"]
+[ERRORLIMIT=<*limit value*>]

The NODE argument, <*input node*>, defines the node whose input is to be checked for frequency.

The MINFREQ argument, <*freq value*>, defines the minimum permissible frequency on <*input node*>.

The MAXFREQ argument, <*freq value*>, defines the maximum permissible frequency on <*input node*>.

Either MINFREQ or MAXFREQ must be present in every FREQ specification.

How it works
FREQ checks compare the frequency on <*input node*> with <*freq value*>. If the frequency is greater than MAXFREQ or less than MINFREQ, the appropriate error message is issued.

WIDTH checks compare the low and high pulse widths on <*input node*> with <*time value*>. If the high pulse width is less than MIN_HI or the low pulse width is less than MIN_LO, the appropriate error message is issued.

WIDTH and FREQ checks are identical if the pulse duty cycle is exactly 50%. Otherwise, the WIDTH check is somewhat more flexible, as it allows for the possibility of asymmetric duty cycles.

The constraint is checked only if <*boolean expression*> is true. If the number of constraint warning messages exceeds <*limit value*>, no further messages are issued. <*extra message text*> if any, is added to the standard error message.

GENERAL:
Marks the start of a group of one or more <*general specifications*> which have the following format:

+GENERAL:
+WHEN=<*boolean expression*>

+MESSAGE="<*message text*>"
+ERRORLIMIT=<*limit value*>

How it works
When any of the CONSTRAINT inputs change, the <*boolean expression*> is evaluated. If the result is TRUE, then an error message containing the simulation time and the <*message text*> is generated.

Constraint points to remember:

- Constraints may be one or many.

- Constraints may occur in any order.

- Constraints may be duplicated. That is, you can use more than one WIDTH or SETUP_HOLD specification.

- Constraints may include <*extra message text*> that is appended to the standard internally generated message text.

- Each constraint specification has its own ERRORLIMIT. The default value is copied from the Global Settings value of DIGERRDEFAULT. DIGERRDEFAULT, like all Global Settings values, may be overridden by using the .OPTIONS command. That is, placing the text,

  .OPTIONS DIGERRDEFAULT=12

  in your circuit changes the DIGERRDEFAULT value for the circuit. When more than ERRORLIMIT violations of the particular constraint check have occurred, no further messages are issued. Other constraints continue to issue messages.

- DIGERRLIMIT, from the Global Settings, is used to limit the total number of error messages from all sources.

Example
Here is an example of the CONSTRAINT used in a decade counter. Note the use of multiple WIDTH and SETUP_HOLD specifications.

```
Ucnstr  CONSTRAINT(9)  DPWR  DGND
+      MRBAR PEBAR CP CEP CET D0 D1 D2 D3
```

```
+      IO_STD
+
+   FREQ:
+      NODE  =  CP
+      MAXFREQ  =  130MEG
+      WIDTH:
+      NODE  =  CP
+      MIN_HI  =  4ns
+      WIDTH:
+      NODE  =  MRBAR
+      MIN_LO  =  5n
+   SETUP_HOLD:
+      CLOCK  LH  =  CP
+      DATA(4)  =  D0  D1  D2  D3
+      SETUPTIME  =  5n
+      WHEN = {  (MRBAR  !=  '0)  }
+   SETUP_HOLD:
+      CLOCK  LH  =  CP
+      DATA(2)  =  CET  CEP
+      SETUPTIME_LO  =  6n
+      SETUPTIME_HI  =  11n
+      WHEN  =  {  (MRBAR  !=  '0)  }
+   SETUP_HOLD:
+      CLOCK  LH  =  CP
+      DATA(1)  =  PEBAR
+      SETUPTIME_HI  =  11ns
+      SETUPTIME_LO  =  7ns
+   SETUP_HOLD:
+      CLOCK  LH  =  CP
+      DATA(1)  =  MRBAR
+      SETUPTIME  =  5ns
```

Stimulus devices

Digital networks usually require stimulus devices for testing and simulation. These devices create a temporal sequence of digital states to stimulate the circuit. They are the digital equivalent of the analog sources, SIN, PULSE, USER, V, and I.

There are two types of sources. The stimulus generator (STIM), and the file stimulus (FSTIM). The STIM device uses a language of basic commands to create virtually any waveform. The FSTIM device reads its waveforms from an external file.

There is no timing model, since timing information is an inherent property of the stimulus devices.

Stimulus generator

The stimulus generator primitive provides a flexible language of commands for creating highly complex digital waveforms. Its format is as follows:

SPICE format
U<*name*> STIM(<*width*>,<*format array*>)
+<*digital power node*> <*digital ground node*>
+<*node*>*
+<*I/O model name*>]
+[IO_LEVEL=<*interface subckt select value*>
+[TIMESTEP=<*stepsize*>]
+<commands>*

Schematic format
PART attribute
<*name*>

Example
C20

FORMAT attribute
<*format array*>

Example
1111 ; four binary values

COMMAND attribute
<*command_name*>

Examples
0ns 0 ; 0 at 0ns
10ns 1 ; 1 at 10ns

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

TIMESTEP attribute
[<*stepsize*>]

Example
10ns

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Definitions

<*width*>

In a SPICE file, this is the number of signals or output nodes. For a schematic part, <*width*> is set when the part is enter in the Component library, so simply picking a STIM device from the Component library menu supplies this information.

<*format array*>

<*command*> statements use <*value*> statements to describe the output states. <*format array*> is an array of characters that defines the data format to be used by <*value*>. It is a sequence of characters which specify the number of outputs (signals) that the corresponding character in <*value*> represents. Each character of <*value*> is assumed to be a number in base $2^{<m>}$, where <m> is the corresponding character in <*format array*>. Each <*value*> has the same number of characters as <*format array*>. That is, if <*format array*> is '1111', then every <*value*> used in a <*command*> statement must have four characters also. The total number of characters in <*format array*> must equal <*width*>, and each character is chosen to reflect the data type (1= Binary, 3 = Octal, 4 = Hex).

<*node*>*

For SPICE components, this defines the output node names. For a schematic component, the nodes are acquired automatically and need no specification.

<*stepsize*>

The TIMESTEP argument <*stepsize*> defines the number of seconds per clock cycle. Transition times may be specified in clocks using the 'c' character. Actual *time* is the number of clocks times <*stepsize*>. The default value is zero.

<command_name>

This is a symbolic name for one or more commands that create a digital pattern for the source. It is usually defined in the text area of the circuit (or a library file) in the following form:

.DEFINE <*command_name*> <*command*>*

where <*command*>* is a set of stimulus commands and is defined as follows:

<*command*>*
<<*time*> <*value*>>
<LABEL=<*label name*>>
<<*time*> GOTO <*label name*> <*n*> TIMES>
<<*time*> GOTO <*label name*> UNTIL GT <*value*>>
<<*time*> GOTO <*label name*> UNTIL GE <*value*>>

<<*time*> GOTO <*label name*> UNTIL LT <*value*>>
<<*time*> GOTO <*label name*> UNTIL LE <*value*>>
<<*time*> INCR BY <*value*>>
<<*time*> DECR BY <*value*>>
REPEAT FOREVER
REPEAT <*n*> TIMES
ENDREPEAT

<*time*>
This specifies the time that the new value occurs, or that the INCR, DECR, or GOTO command is executed. <*value*> can be specified in two units (clocks and seconds) and in two forms (absolute and relative). To express the <*time*> in clocks, add the suffix 'C' to the number, as in '5C'. To express the <*time*> in seconds, use the suffix 'S' or none at all.

Relative time is measured from the last time. Absolute time is measured from the start of the simulation. To use relative time, prefix the plus '+' character to <*time*> as in '+10ns, or '+23C'.

<*value*>
This defines the new value for each output node as interpreted by <*format array*>. <*value*> is chosen from the binary set { 0, 1, R, F, X, Z, ? }, the octal set {0-7}, or the hex set {0-F}. The '?' character specifies a single random state of 1 or 0. RND specifies an array of random states, one for each output node.

<*label name*>
This identifies locations for GOTO statements. GOTO <*label name*> causes a jump to the statement following the 'LABEL=<*label name*>' statement.

<n>
This is the number of times to repeat a GOTO loop. A value of -1 creates a loop that repeats forever.

Notes
• Absolute times within loops are converted to relative times based upon the last used time and the last used increment size.

• GOTO labels must have been previously defined in a LABEL statement. No forward references are allowed.

• Absolute time values must be in ascending order, except that the time after a GOTO may be the same as the GOTO.

• When a GOTO command directs execution to the first statement following its specified label, the programs ignores the time value of this first statement.

• UNTIL GT <*value*> means the loop executes until the node(s) value is greater than <*value*>. Similarly, GE means greater than or equal to, LT means less than, and LE means less than or equal to.

Stimulus generator examples

The examples that follow show how to code various waveforms. Example 1, 2, and 3 use a STIM1 (one output). Example 4 uses a STIM2 (two output). Example 5 uses a STIM8 (eight output) device.

For schematics:
For the FORMAT and COMMAND attributes, enter the following:

Example	FORMAT	COMMAND
1	1	IN1
2	1	IN2
3	1	IN3
4	11	IN4
5	44	IN5

In the text area of the schematic, or in the grid text, or in the MCAP.INC, define the commands IN1, IN2, IN3, IN4, and IN5 as follows:

```
.DEFINE IN1
+0NS 1
+10NS 0
+20NS 1

.DEFINE IN2
+  +0NS 1
+  +10NS 0
+  +10NS 1

.DEFINE IN3
+ 0NS 0
+LABEL=BEGIN
+  +5NS 1
```

```
+  +5NS 0
+  +5NS GOTO BEGIN -1 TIMES

.DEFINE IN4
+LABEL=BEGIN
+  +0NS 00
+  +5NS 01
+  +5NS 10
+  +5NS 11
+  +5NS GOTO BEGIN -1 TIMES

.DEFINE IN5
+LABEL=BEGIN
+  +0NS INCR BY 01
+  +10NS GOTO BEGIN UNTIL GE 06
+  +10NS F0
+  +10NS F1
```

The digital outputs generated by these patterns are included with the SPICE examples shown in the following pages.

You can also see the patterns by running the circuit file STIMSAMP.CIR, which contains STIM sources with all of these patterns. Additional examples of stimulus commands can be found in the sample circuit STIM_DEMO.CIR.

For SPICE text files:
The following examples illustrate how to create the same patterns in SPICE files.

Example 1
The first example is a single output device exhibiting a single zero pulse starting at 10ns and lasting to 20ns. There are no loops.

```
U1 STIM(1,1) $G_DPWR $G_DGND
+ 1                ;Output node number
+ IO_STD
+ 0NS 1
+ 10NS 0
+ 20NS 1
```

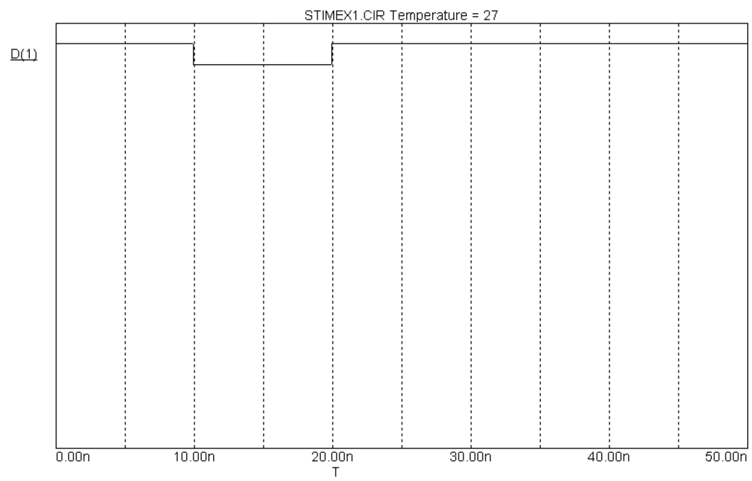


Figure 23-4 Stimulus waveform for examples 1 and 2

Example 2
The next example produces the same output as the first example, but uses relative times instead of absolute times.

```
U1 STIM(1,1) $G_DPWR $G_DGND
+ 1                ;Output node number
+ IO_STD
+ + 0NS 1
+ + 10NS 0
+ + 10NS 1
```

The first transition occurs at 0ns, the next at 0ns+10ns=10ns, and the last occurs at 0ns+10ns+10ns = 20ns, the same as in the first example.

Example 3
This shows how to create a single output alternating 0 - 1 sequence.

```
U1 STIM(1,1) $G_DPWR $G_DGND
+ 1                 ;Output node number
+ IO_STD
+ 0NS 0
+ LABEL=begin
+ + 5NS 1
+ + 5NS 0
+ + 5NS GOTO begin -1 TIMES
```

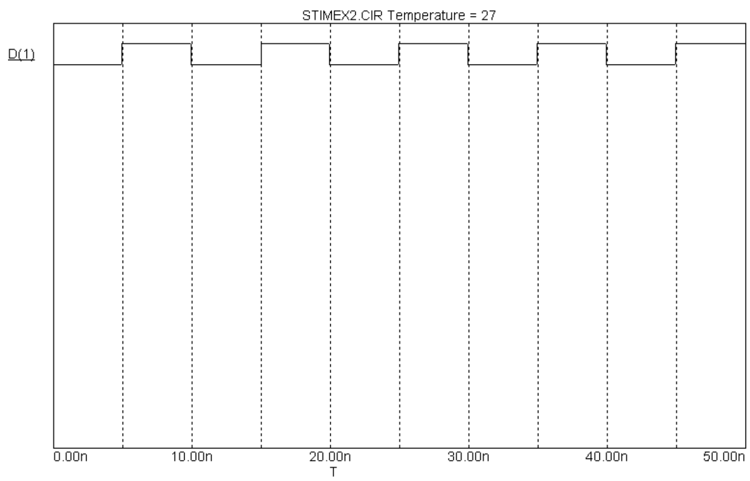


Figure 23-5 Stimulus waveform for example 3

Execution begins at T=0ns with the output set to '0'. At T=5ns (0ns+5ns), the output is set to '1'. At T=10ns (5ns+5ns), the output is set to '0'. At T=15ns (10ns+5ns), the GOTO is executed and control passes to the '+5ns 1' statement following the 'LABEL=begin' statement. The time instruction (+5ns) is ignored and the output is set to '1'. The <*time*> value of the first statement executed as a result of a GOTO is always ignored. The same waveform can be achieved by replacing the portion "+ LABEL=begin...+ 5NS GOTO begin -1 TIMES" with:

```
+ REPEAT FOREVER
+ + 5NS 1
+ + 5NS 0
+ + 5NS ENDREPEAT
```

Example 4

This example shows how to create a two-output alternating 0 - 1 sequence.

```
U1 STIM(2,11) $G_DPWR $G_DGND
+ 1 2              ;Output node numbers
+ IO_STD
+ LABEL=begin
+ + 0NS 00
+ + 5NS 01
+ + 5NS 10
+ + 5NS 11
+ + 5NS GOTO begin -1 TIMES
```

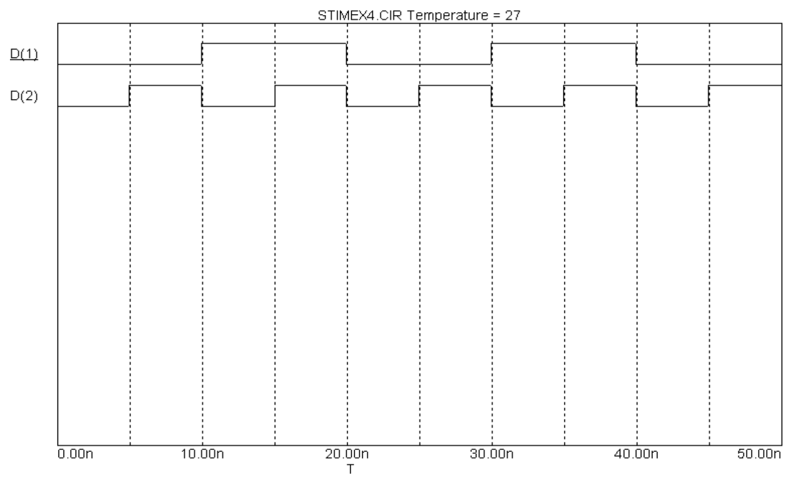


Figure 23-6 Stimulus waveform for example 4

Notice that the STIM(2,11) statement specifies two outputs with a binary format. Execution begins at the '+ 0NS 00' statement, where the outputs are set to '00'. The next command, '+ 5NS 01', occurs at T=5ns (0ns+5ns), where the outputs are set to '01'. The next command, '+ 5NS 10', occurs at T=10ns (5ns+5ns), where the outputs are set to '10'. The next command, '+ 5NS 11', occurs at T=15ns (10ns+5ns), where the outputs are set to '11'. The next command, '+ 5NS GOTO begin -1 TIMES', occurs at T=20ns (15ns+5ns). It causes a jump to the statement '+ 0NS 00' (the one following the 'LABEL=begin command'. From here the loop simply repeats until the end of the simulation run.

Example 5

This next example shows how to use the INCR and UNTIL commands.

```
U1 STIM(8,44) $G_DPWR $G_DGND
+ 1 2 3 4 5 6 7 8
+ IO_STD
+ LABEL=begin
+ + 0NS INCR BY 01                    ;Increment by hex 01
+ + 10NS GOTO begin UNTIL GE 06       ;Count until 06 hex
+ + 10NS F0                           ;After 10NS goto F0 hex
+ + 10NS F1                           ;After 10NS goto F1 hex
```

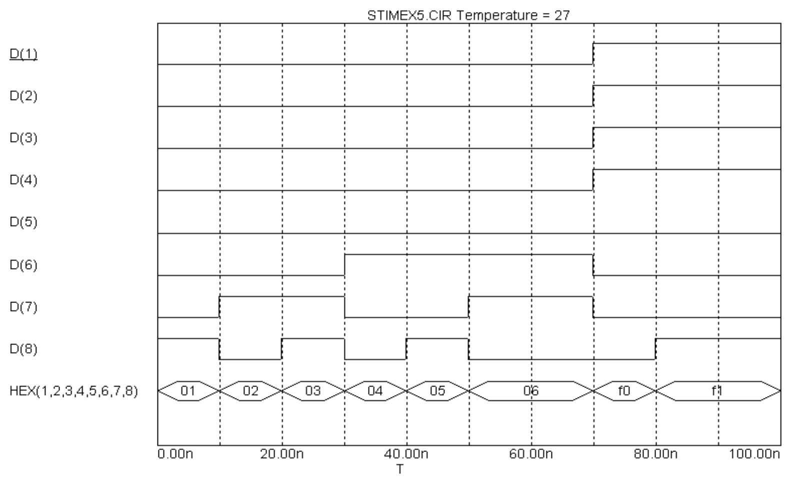


Figure 23-7 Stimulus waveform for example 5

This STIM generates eight outputs. The sequence begins with the eight outputs set to 00 hex during the operating point. At 0ns, the outputs are incremented by 01 from 00 to 01. At 10ns, the GOTO-UNTIL is checked and found to be true. Control is passed to the INCR statement following the LABEL=begin, where the outputs are incremented to 02. At T = 50ns, the outputs have reached 05 and the GOTO-UNTIL test sends execution back to the INCR command. This command at T=50ns, increments the outputs to 06. At T = 60ns, the outputs are at 06 and the GOTO-UNTIL test fails. Execution drops to the '+10NS F0' command and T=70ns. The outputs go to F0 and then execution drops to the '+10NS F1' command and T=80ns. The outputs go to F1 and the STIM program terminates, leaving the outputs at F1 for the remainder of the run.

The File Stimulus device

The File Stimulus device lets you import digital waveforms from a text file. This device lets you import digital waveforms from other simulators or from the MC7 output file (with some editing).

File Stimulus input file format

The File Stimulus device expects to read a file in the following format:

<*Header*>
One or more blank lines
<*Transition tables*>

<*Header*>
This consists of the following:
[TIMESCALE=<*time scale value*>]
[<*first signal name*>...<*last signal name*>]
OCT(<*signal name bit 3*>...<*signal name of lsb*>)...
HEX(<*signal name bit 4*>...<*signal name of lsb*>)...

<*Header*> contains the optional [TIMESCALE=<value>] line and a single line list of signal names. The signal names may be separated by spaces, commas, or tabs. The line may not be continued with the continuation character '+'. The line may contain up to 256 signals. The signal names will be associated with signal values from the <*Transition tables*>, in the same order.

The optional OCT(<*sig1*>, <*sig2*>, <*sig3*>) radix function lets you declare segments of the transition table as octal numbers. It decodes each octal digit from the table into three binary values and assigns these values to the three signal names. The function must have exactly three signal names as arguments and it consumes one octal digit from each row in the transition table.

The optional HEX(<*sig1*>, <*sig2*>, <*sig3*>, <*sig4*>) radix function lets you declare segments of the transition table as hex numbers. It decodes each hex digit into four binary values and assigns the values to the four signal names. The function must have exactly four signal names as arguments and it consumes one hex digit from each row in the transition table.

Signal names not using the HEX or OCT function are assumed to be in binary format. Each signal name in binary format consumes exactly one binary digit from the transition table.

The total number of physical outputs is:

Outputs = binary names + 3•octal names + 4•hex names

The total number of digits in the state portion of the transition table will be:

Digits = binary names + OCT statements + HEX statements

<*Transition tables*>
The transition table format is as follows:

<*time*> <*state value*>*
<*Transition tables*> start on the first non-blank line following the *Header*. Each line must contain a transition time, <*time*>, followed by a space or tab, followed by one or more <*state values*>.

<*time*>

<*time*> is always expressed as an integer. It may be absolute or, if preceded by a '+', relative to the previous time. The <*time*> value is multiplied by the <*time scale value*> to get the actual transition time.

<*value*>

Each digit of <*value*> provides one state for each binary signal name, three states for each OCT set of three signal names, or four states for each HEX group of four signal names. Each digit is stripped from the table and assigned to the signal names in the same left-to-right order. The list of valid digits are:

	Binary	**Octal**	**Hex**
Logic state	0,1	0-7	0-F
Unknown	X	X	X
High impedance	Z	Z	Z
Rising	R	R	None
Falling	F	F	None

Table 23-21 File Stimulus device digit codes

Transition Table example

TIMESTEP=1NS
A B C D HEX(A4,A3,A2,A1) OCT(D3,D2,D1)

0	0000F3	; transition *time* =0
1	000104	; transition *time* =1NS
+2	001015	; transition *time* =3NS (1NS+2NS)
5	001126	; transition *time* =5NS

SPICE format

U<*name*> FSTIM(<*no. of outputs*>)
+<*digital power node*> <*digital ground node*>
+<*node*>*
+<*I/O model name*>
+ FILE=<*stimulus file name*>
+[IO_LEVEL=<*interface subckt select value*>]
+[SIGNAMES=<*signal names from stimulus file*>]

Schematic format

PART attribute
<*name*>

Example
FS1

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

FILE attribute
<*file name*>

Example
MYFILE.STM

SIGNAMES attribute
<*signal names from stimulus file*>

Example
CLEAR PRESET Q QB

IO_LEVEL attribute
<*interface subckt select value*>

Example
0

POWER NODE attribute
<*digital power node*>

Example
$G_DPWR

GROUND NODE attribute
<*digital ground node*>

Example
$G_DGND

Definitions
<*no. of outputs*>
In a SPICE file, this specifies the number of signals or output nodes. For schematic components, this value is defined when the component is first entered into the Component library. Simply picking an FSTIM device from the Component library menu supplies this information.

<*digital power node*> <*digital ground node*>
These nodes are used by the I/O circuits.

<*node*>*
For SPICE circuit components, this list defines the output nodes. For a schematic circuit component, this data is automatically acquired from the output node numbers or node names. Node names are defined by the user by placing grid text on the output nodes. The number of names in this list must equal <*no. of outputs*>.

<*I/O model name*>
This *name* references an I/O model statement which specifies the electrical interface to be used when an analog node and digital node connect. It also specifies the DRVH and DRVL impedances that determine signal strengths when an output is wired to another digital output.

FILE=<*stimulus file name*>
This is the stimulus file *name* specified as a text string enclosed in double quotes.

Example
"MyFile.stm"

[IO_LEVEL=<*interface subcircuit select value*>]
This selects one of four interface circuits named in the I/O model. These circuits are used at the digital/analog interface.

Example
1

[SIGNAMES=<*signal names from stimulus file*>]
This selects one or more of the signal names from the stimulus file <*Header*> for use by one of the output nodes,<node>*. The signal from the stimulus file is assigned to drive the <*node*> positionally associated with it. If no signal names are specified, then the program will expect to find signal names in the stimulus file that match the output node names ,<*node*>*. This command must be the last command in a SPICE file component specification.

Example1
U1 FSTIM(3) $G_DPWR $G_DGND
+ A B C
+IO_FS
+FILE="PATTERN.STM"

In Example 1, there is no SIGNAMES command, so the stimulus file PATTERN.STM must contain the signal names A, B, and C in the <*Header*>. The signals for A, B, and C would be assigned to the FSTIM outputs A, B, and C.

Example 2
U2 FSTIM(3) $G_DPWR $G_DGND
+ A B C
+IO_FS
+FILE="PATTERN.STM"
+ SIGNAMES=X Y Z

In Example 2, there is a SIGNAMES command, so the stimulus file PATTERN.STM must contain the signal names X, Y, and Z in the <*Header*>. The signals for X, Y, and Z would be assigned to the FSTIM outputs A, B, and C.

Example 3
U3 FSTIM(4) $G_DPWR $G_DGND
+ TOM RAY CAR TALK
+IO_FS
+FILE="PATTERN.STM"
+ SIGNAMES=CLIK CLAK

The stimulus file PATTERN.STM must contain the signal names CLIK, CLAK, CAR, and TALK. The signals for CLIK and CLAK would be assigned to the FSTIM outputs TOM and RAY respectively. The signals for CAR and TALK would be assigned to the FSTIM outputs CAR and TALK respectively.

The sample circuit FSTIM8 illustrates the use of the file stimulus device.

I/O model

I/O models provide the information necessary to determine the output strength when devices are wire-ored together, and to create the interface circuits when the digital part is connected to an analog part.

I/O models capture the electrical information common to the IC technology and circuit techniques used to design and build them. Thus, a typical digital family will have only four or five I/O models. The only difference in the I/O models within a digital family is to account for the different circuits employed at the input or output, as, for example, in open-collector outputs and Schmitt-trigger inputs.

I/O model format
.MODEL <*I/O model name*> UIO ([*model parameters*])

Parameter	**Description**	**Units**	**Default**
INLD	Input load capacitance	farad	0
OUTLD	Output load capacitance	farad	0
DRVH	Output high level resistance	ohm	50
DRVL	Output low level resistance	ohm	50
DRVZ	Output Z state resistance	ohm	250K
INR	Input leakage resistance	ohm	30K
TSTOREMN	Min storage time for charge storage node	sec	1E-3
AtoD1	AtoD interface circuit for level 1		AtoDDefault
DtoA1	DtoA interface circuit for level 1		DtoADefault
AtoD2	AtoD interface circuit for level 2		AtoDDefault
DtoA2	DtoA interface circuit for level 2		DtoADefault

Table 23-22 I/O model parameters

Parameter	Description	Units	Default
AtoD3	AtoD interface circuit for level 3		AtoDDefault
DtoA3	DtoA interface circuit for level 3		DtoADefault
AtoD4	AtoD interface circuit for level 4		AtoDDefault
DtoA4	DtoA interface circuit for level 4		DtoADefault
TSWLH1	Low to high switching time for DtoA1	sec	0
TSWLH2	Low to high switching time for DtoA2	sec	0
TSWLH3	Low to high switching time for DtoA3	sec	0
TSWLH4	Low to high switching time for DtoA4	sec	0
TSWHL1	High to low switching time for DtoA1	sec	0
TSWHL2	High to low switching time for DtoA2	sec	0
TSWHL3	High to low switching time for DtoA3	sec	0
TSWHL4	High to low switching time for DtoA4	sec	0
TPWRT	Pulse width rejection threshold	sec	prop delay
DIGPOWER	Power supply subcircuit name		DIGIFPWR

Table 23-22 I/O model parameters (continued)

INLD and OUTLD are used to compute the optional loading delay. This value increases the propagation delay through the device to account for excessive capacitive loading on the node caused by high fan-out. Fan-out is the number of gate inputs connected to a device's output.

DRVH and DRVL are the high state and low state impedances used to determine output strength. Strength is used to resolve the output state when a digital output is connected to other digital outputs.

DRVZ, INR, and TSTOREMN are used to determine which nodes are to be treated as charge storage nets. *Charge storage nets are not available in the current version.*

AtoD1 through AtoD4 and DtoA1 through DtoA4 supply the names of the interface circuits. INLD and the AtoD names do not apply to stimulus sources since they do not have inputs. Refer to the "Analog/digital interface" section for more information.

The switching times TSWLH1... TSWLH4 and TSWHL1...TSWHL4 are subtracted from the digital device's propagation delay on outputs which are connected to analog devices. The purpose is to compensate for the time it takes the DtoA interface circuit to switch. By compensating in this way, the analog signal at the other side of the DtoA interface should reach the switching level just when the digital device does at the stated delay. The values for these switching delays are determined by attaching a nominal load to a digital output, and measuring the switching time. If the switching time is greater than the stated delay, a delay of zero is used. These parameters are used only when analog nodes are connected to digital outputs.

The DIGPOWER parameter specifies the name of the power supply subcircuit to be used when an AtoD or DtoA interface is required. The default value is DIGIFPWR. This power supply subcircuit can be found in the DIGIO.LIB. It is the standard circuit for TTL circuits.

Note that the TPWRT parameter is not included in the current version. It is accepted as a parameter, but not processed.

Digital / analog interface devices

When a digital node and an analog node are connected together in a circuit, the system breaks the connections and inserts between the two parts the interface circuit specified in the I/O model. These interface circuits contain analog devices like resistors, capacitors, diodes, and transistors. They also contain either an analog to digital or digital to analog interface device. These devices provide the fundamental translation between the analog and digital circuits.

Digital input device (N device)

When a digital output node is connected to an analog node, the interface circuit requires an N device. Its function is to translate digital levels to analog voltages and impedances to drive the analog node.

SPICE format
N<*name*> <*interface node*> <*low level node*> <*high level node*>
+<*model name*>
+ DGTLNET=<*digital node name*>
+<*I/O model name*>
+[IS=<*initial state*>]

Schematic format
PART attribute
<*name*>

Example
FS1

MODEL attribute
<*model name*>

Example
D0_AD

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

IS attribute
<*initial state*>

Example
1

Model form
.MODEL <*model name*> DINPUT ([*model parameters*])

Parameter	**Description**	**Units**	**Default**
CLO	Capacitance to low level node	farad	0
CHI	Capacitance to high level node	farad	0
S0NAME	State '0' character abbreviation		
S0TSW	State '0' switching time	sec	
S0RLO	State '0' resistance to low level node	ohm	
S0RHI	State '0' resistance to high level node	ohm	
S1NAME	State '1' character abbreviation		
S1TSW	State '1' switching time	sec	
S1RLO	State '1' resistance to low level node	ohm	
S1RHI	State '1' resistance to high level node	ohm	
. . .	. . .		
S19NAME	State '19' resistance to high level node	ohm	
S19TSW	State '19' switching time	sec	
S19RLO	State '19' resistance to low level node	ohm	
S19RHI	State '19' resistance to high level node	ohm	

Table 23-23 The N device model parameters

When a digital device output is connected to an analog node, MC7 automatically breaks the connection and inserts the DtoA circuit specified in the I/O model. That circuit always employs an N device, whose function is to translate the digital states to impedance changes on the analog side. The process is described in more detail in the "The analog / digital interface" section in this chapter.

The equivalent circuit of the N device is as follows:

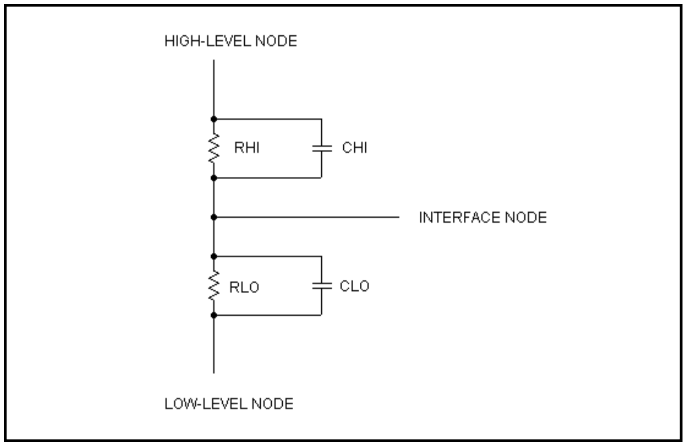


Figure 23-8 N device equivalent circuit

The N device contains two resistors and two optional capacitors. The resistance of the resistors changes in response to changes in the digital input. In a SPICE file, the digital input node is specified by <*digital node name*>. In a schematic, the digital input node is simply the node associated with the 'Digital' pin of the N device. When this digital input node changes from a '0' to a '1', the value of RHI changes linearly versus time from the '0' state high resistance to the '1' state high resistance. Similarly, the value of RLO changes linearly versus time from the '0' state low resistance to the '1' state low resistance.

The transition from the old resistance to the new resistance is accomplished in a linear fashion over the switching time specified in the DINPUT model for the new state. The output voltage changes from the old level to the new level during the switching time. The output curve looks somewhat like an exponential due to the simultaneous change of the two resistor values. The transition values of resistance for each state are obtained from the DINPUT model. Normally the <*high level node*> and <*low level node*> are connected to the voltage sources that correspond to the highest and lowest logic levels. The connection is usually made within the particular DtoA interface circuit called for in the I/O model.

Digital inputs may be any of the following states {0, 1, R, F, X, Z}. The N device will generate an error message if any state other than these are presented.

The initial condition of the digital input to the N device is determined during the initial operating point calculation. To override this value, you can use the IS command:

IS=<*initial state*>

The digital input is set to <*initial state*> at T = tmin and remains there until one of the devices driving the node changes the node's state.

Digital output device (O device)

When a digital input node is connected to an analog node, the A/D interface circuit requires an O device. Its function is to translate analog voltages into digital levels on the digital node.

SPICE format
O<*name*> <*interface node*> <*reference node*>
+<*model name*>
+ DGTLNET=<*digital node name*>
+<*I/O model name*>

Schematic format
PART attribute
<*name*>

Example
FS1

MODEL attribute
<*model name*>

Example
D_AD

I/O MODEL attribute
<*I/O model name*>

Example
IO_STD

When a digital device input is connected to an analog node, the system automatically breaks the connection and inserts the AtoD circuit specified in the I/O model. That circuit always employs an O device. The function of the O device is to translate the analog voltages on the analog side to digital states on the digital side. The exact process is described in more detail in the "The analog/digital Interface" section of this chapter.

Model form
.MODEL <*model name*> DOUTPUT ([*model parameters*])

Parameter	Description	Units	Default
RLOAD	Output resistance	ohm	1/Gmin
CLOAD	Capacitance to high level node	farad	0
S0NAME	State '0' character abbreviation		
S0VLO	State '0' low level voltage	volt	
S0VHI	State '0' high level voltage	volt	
S1NAME	State '1' character abbreviation		
S1VLO	State '1' low level voltage	volt	
S1VHI	State '1' high level voltage	volt	
. . .	. . .		
S19NAME	State '19' character abbreviation		
S19VLO	State '19' low level voltage	volt	
S19VHI	State '19' high level voltage	volt	
SXNAME	State to use when the voltage is outside all state ranges		

Table 23-24 The O device model parameters

The equivalent circuit of the O device is as follows:

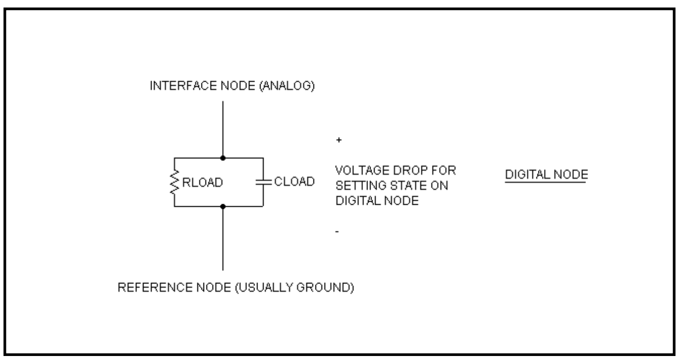


Figure 23-9 O device equivalent circuit

The O device contains a resistor, RLOAD, in parallel with a capacitor, CLOAD. They are connected between the analog *<interface node>* and the *<reference node>*, which is usually analog ground. The analog voltage across the parallel combination is monitored by the O device. The digital output of the O device is the state whose 'State N low level voltage' and 'State N high level voltage' bracket the actual voltage across the parallel RC network. To determine which bracket contains the voltage level, a progressive search is employed. The search starts at the current state bracket. If the voltage is outside the state range, it tries the next highest bracket. If the search fails at state 19, it checks state 0. If this fails, it tries the next highest state. If the entire model is unsuccessfully searched, the system uses the SXNAME if it has been defined. Otherwise it uses the state with the nearest voltage match.

This searching algorithm allows for the easy creation of hysteresis loops. Consider the following model statement:

```
S0NAME='0' S0VLO=-1.5 S0VHI=1.7
S1NAME='1' S1VLO=0.9 S1VHI=7.0
```

Suppose the voltage starts at 0.0. The digital level is '0'. As the voltage rises it must exceed the 'S0VHI=1.7' value to exit the '0' state. When it does, it drops into the '1' state. When the voltage starts to decline, it must drop below the 'S1VLO = 0.9' level to drop into the '0' state. This provides a hysteresis value of S0VHI - S1VLO = 1.7 - 0.9 = 0.8 volts. Similar hysteresis values can be designed into the other states.

The state characters used in the model statement must be chosen from the set { 0, 1, R, F, X, Z }. The 'Z' state is usually not used since it conveys no level information and is really just a statement about the impedance level. Other characters will generate error messages and stop the simulation.

Chapter 24 Fourier Analysis and Digital Signal Processing

What's in this chapter

This chapter describes how to use the Digital Signal Processing operators to do Fourier analysis on circuit waveforms. It includes these topics:

- How digital signal processing functions work
- Digital Signal Processing functions
- A DSP example
- The DSP dialog box

How digital signal processing functions work

The Digital Signal Processing (DSP) functions are a group of math functions for extracting frequency and time information from transient analysis waveforms and AC analysis curves. All of the functions employ the same internal Fast Fourier Transform (FFT) routine. That routine requires two basic parameters to do its work; N, the number of data points, and DF, the sampling frequency.

N: Number of data points
N is determined in one of two ways:

- N = The Number of Points field in the DSP dialog box if its Status is On.
- N = The binary number nearest the actual number of data points computed during the analysis run if Status is Off.

N must always be a power of two and is also constrained to the following range:

- Minimum = 64 or 2^6
- Maximum = 262144 or 2^{18}

DF: Sampling frequency
The sampling frequency, DF, is calculated as follows:

Transient analysis:

If the DSP dialog box Status flag is Off:

DF = sampling frequency = 1'st harmonic = 1 / <*tmax*>
<*tmax*> is from the Time Range field of the analysis dialog box.

If the DSP dialog box Status flag is On:

DF = 1'st harmonic = 1 / (<*Upper Time Limit*> **-** <*Lower Time Limit*>)
where <*Upper Time Limit*> and <*Lower Time Limit*> are from the DSP dialog box.

AC analysis:

If the DSP dialog box Status flag is Off:

1'st harmonic = DF = sampling frequency = <*fmax*> / N

<*fmax*> is from the Frequency Range field of the analysis dialog box.

If the DSP dialog box Status flag is On:

DT = Time Step = 1 / DF = N / <*Upper Freq Limit*>
where <*Upper Freq Limit*> is from the DSP dialog box.

DF is the interval between sample points in the output FFT and is also referred to as the first harmonic. The frequency range sampled is:

Data Point	Contents
1	0 Hz or DC value
2	H1 = 1'st harmonic at F = DF
3	H2 = 2'nd harmonic at F = 2*DF
...	...
N	HN = N-1'th harmonic at F = (N-1)*DF

The DSP functions can be used in both AC and transient analysis.

In transient analysis, time domain waveforms (curves of real voltage vs. time) are sampled and fed to the FFT routines which employ N samples of DF frequency steps to calculate, typically, an FFT or a HARM function.

In AC analysis, frequency spectra (curves of complex voltage vs. frequency) are sampled and fed to the FFT routines which employ N samples of the time step, DT, to calculate, typically, an IFT function.

Resolution, range, and accuracy are controlled by the FFT parameters N and DF.

In transient analysis:
If the DSP dialog box Status flag is Off, increasing <*tmax*> in transient analysis decreases DF, resulting in finer frequency resolution and a lower maximum frequency, since fmax = (N-1)*DF.

Increasing N increases the upper frequency range.

In AC analysis:
If the DSP dialog box Status flag is Off, increasing <*fmax*> in AC analysis decreases the time step, DT, resulting in a finer time step resolution and a lower tmax, since tmax = (N-1)*DT. Increasing N increases the maximum time range.

Digital Signal Processing functions

The Digital Signal Processing (DSP) functions include the following:

- **HARM(u)**

This function returns the amplitude of the harmonics of waveform u. Amplitude means the multiplier of the sine or cosine function. For example, for this function,

V(T) = 3.0*SIN(2*PI*1E6*T)

HARM would return 3.0 for the harmonic associated with 1E6.

HARM is the closest thing to the original SPICE .FOUR command. It returns the following values:

Data Point	Contents
1	H0 = DC value
2	H1 = Amplitude of 1'st harmonic
3	H2 = Amplitude of 2'nd harmonic
…	…
N	HN = Amplitude of N-1'th harmonic

Examples:

HARM(V(OUT))	Harmonic amplitude of the voltage waveform V(OUT).
HARM(IC(Q1))	Harmonic amplitude of the current waveform IC(Q1).

Note that, in transient analysis, the first harmonic occurs at DF = 1 / <tmax>.

- **THD(S[,FR])**

This function returns a running sum of the total harmonic distortion of the spectrum S as a percent of the harmonic magnitude at the reference frequency FR. It returns the following values:

Data Point	Contents
1	0
2	0
3	Distortion of 2'nd harmonic as % of first harmonic
4	Distortion of 2'nd and 3'rd harmonic as % of first
…	…
N	Distortion of all harmonics as % of first harmonic

In general, the m'th value of the THD is given by:

$$THD_m = 100*((H_2^2 + H_3^2 + H_4^2 +... H_m^2) / H_1^2)^{0.5}$$

where $H_m = (REAL(H_m)^2 + IMAG(H_m)^2)^{0.5}$

If FR is unspecified, or if the FR value specified does not match one of the harmonic frequencies, n*DF, to within 1%, FR is set to the first harmonic.

Examples:

THD(HARM(V(1)))	THD of the harmonics of the waveform V(1).
THD(HARM(I(D1)))	THD of the harmonics of the waveform I(D1).

- **IHD(S[,FR])**

This function behaves the same as the THD function but returns individual harmonic distortion as a percent of the harmonic magnitude at the reference frequency FR. It returns the following values:

Data Point	Contents
1	0
2	0
3	Distortion of 2'nd harmonic as % of first harmonic
4	Individual distortion through 3'nd harmonic
...	...
N	Individual distortion through N-1'th harmonic

In general, the m'th value of the IHD is given by:

$$IHD_m = 100*(mag(H_m) / mag(H_1)$$

If FR is unspecified, or if the FR value specified does not match one of the harmonic frequencies, n*DF, to within 1%, FR is set to the first harmonic.

Examples:

IHD(HARM(V(A)))	IHD of the harmonics of the waveform V(A).
IHD(HARM(IB(Q1)))	IHD of the harmonics of the waveform IB(Q1).

- **FFT(u)**

This function returns a classical Fourier transform of waveform u. It does not return the harmonics as HARM does. The FFT calculates a set of Fourier coefficients scaled by N/2 (DC scaled by N). Recall that the Fourier series approximates a waveform x(t) as follows:

$x(t) = a_0/2 + \Sigma\ (a_N \cdot \cos(2 \cdot \pi \cdot N \cdot f1 \cdot t) + b_N \cdot \sin(2 \cdot \pi \cdot N \cdot f1 \cdot t))$

where f1 is the fundamental frequency.

The FFT function returns a complex quantity whose real part contains the scaled Fourier series a_N coefficients and the imaginary part contains the scaled Fourier series b_N coefficients:

Data Point	REAL(FFT())	IMAG(FFT())
1	$a_0 \cdot N$	0
2	$a_1 \cdot N/2$	$b_1 \cdot N/2$
3	$a_2 \cdot N/2$	$b_2 \cdot N/2$
...	...	...
N	$a_{N-1} \cdot N/2$	$b_{N-1} \cdot N/2$

The FFT and its inverse are defined such that IFT(FFT(x(t))) = x(t). That is, the inverse of the transform of a waveform is equal to the original waveform.

Examples:

FFT(V(A))	FFT of the voltage waveform V(A).
FFT(IB(Q1))	FFT of the collector current waveform IB(Q1).

*The HARM, THD, and FFT functions transform a time-domain waveform into a frequency-domain spectrum, so when you use one of these in a Y expression, the X expression should be F for frequency. The X range should be initially set to between 5*DF and 10*DF.*

- **IFT(S)**

This function returns a classical inverse Fourier transform of a spectrum S. A spectrum is a list of the value of some complex expression vs. frequency. In AC analysis all expressions are of this type, so IFT(V(1)) would produce meaningful results. In transient analysis, IFT(V(1)) would make no sense, since V(1) would be a time domain waveform. However, an expression like IFT(FFT(V(1))) would produce meaningful results in transient analysis.

Example:

IFT(V(5)*I(R10))	IFT of the spectrum V(5)*I(R10).

*An IFT transforms a frequency-domain spectrum into a time-domain waveform, so when you use an IFT in a Y expression, the X expression should be T for time. The initial time range should be set to between 10*DT and 100*DT.*

- **CONJ(S)**

This function returns the conjugate of a spectrum S. The conjugate of the complex number a + b•j is a - b•j . This function simply negates the imaginary part of the spectrum.

Example:
CONJ(FFT(V(1))) Conjugate of the spectrum FFT(V(1)).

- **CS(u1,u2)**

This function returns the cross spectrum of two waveforms, u1 and u2. The cross spectrum is defined as CONJ(FFT(u2))*FFT(u1)*DT*DT.

Example:
CS(V(1)),V(2)) Cross spectrum of V(1) and V(2)

- **AS(u)**

This function returns the auto spectrum of a waveform u. The auto spectrum is defined as AS(u) = CONJ(FFT(u))*FFT(u)*DT*DT.

Example:
AS(I(RL)) Auto spectrum of waveform I(RL)

- **AC(u)**

This is the auto correlation of waveform u. The definition is:

AC(u) = IFT(CONJ(FFT(u))*FFT(u))*DT

This function is useful for finding periodic signals buried in noisy waveforms.

Example:
RE(AC(V(10))) Auto correlation of waveform V(10)

See the sample circuit FFT5.CIR for an example of this type of function.

- **CC(u,v)**

This is the cross correlation of waveforms u and v. The definition is:

CC(u,v) = IFT(CONJ(FFT(v))*FFT(u))*DT

This function is useful for finding the time delay betwen two periodic signals.

Example:
CC(V(1),V(2))   Cross correlation of waveform V(1) with V(2).

See the sample circuit FFT3.CIR for an example of this type of function.

- **COH(u,v)**

This is the coherence function of waveforms u and v. The definition is:

COH(u,v) = CC(u,v)/sqr(AC(u(0))*AC(v(0)))

Example:
COH(V(1),V(2))   Coherence of waveform V(1) with V(2).

DSP operators

The math functions available to manipulate the output of DSP functions are:

Short form	Long form	Function
RE(S)	REAL(S)	Real part of spectrum S
IM(S)	IMAG(S)	Imaginary part of spectrum S
MAG(S)	MAG(S)	Magnitude of spectrum S
PH(S)	PHASE(S)	Phase of spectrum S

Either the long form or the short form may be used. Note that the MAG function is redundant. Plotting V(1) is the same as plotting MAG(V(1)), since the plotting routines always plot the magnitude of the final result.

DSP Accuracy

Accuracy of the lower harmonics is affected mainly by the size of the time or frequency step in the analysis run. If there are many high frequency components, then accuracy will be affected by N, since N determines fmax.

The table below summarizes how accuracy is affected by the time step in the circuit FFT1.CIR. This circuit contains a single source generating a pure mixture of DC, 1Mhz, 2Mhz, and 3Mhz sine waves. The result below is for the HARM function applied to the soure node, HARM(V(1)). Ideally, the HARM function should return 1.5, 1.0, 2.0, and 3.0 exactly. The extent to which the results differ from these exact values determines the accuracy.

The error in parts per billion (1E9) with N=1024 is as follows:

Max Time Step and % of 1'st harmonic period	1'st harmonic	2'nd harmonic	3'rd harmonic
1ns (.1%)	3300	13000	30000
.1ns (0.01%)	33	130	300
.01ns (0.001%)	0.3	1.3	3.0

The error in decibels DB(ERROR) with N=1024 is as follows:

Max Time Step and % of 1'st harmonic period	1'st harmonic	2'nd harmonic	3'rd harmonic
.001U (.1%)	-109	-97	-91
.0001U (0.01%)	-149	-137	-131
.00001U (0.001%)	-191	-177	-170

N has virtually no effect on the accuracy of the lower harmonics.

DSP Speed

The speed of the FFT routine is approximately a linear function of N. Always use a value of N that is as small as possible. A value of 1024 is nearly always a good choice. Remember, N affects only fmax, because fmax = N*DF. Sometimes you may want to increase the tmax to decrease DF to get finer frequency resolution. This has the side effect of lowering fmax, so you may need to increase N to maintain the desired fmax.

A DSP example

To illustrate how DSP functions are used load the file FFT1. It looks like this:

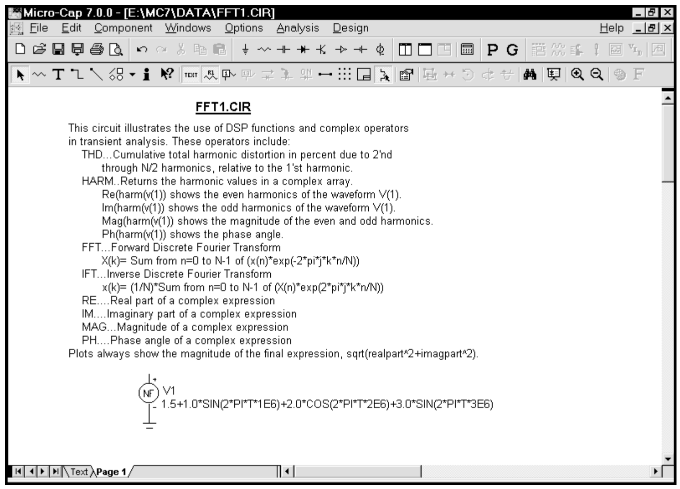


Figure 24-1 The FFT1 circuit

This circuit contains a node to which is attached a function source with an output expression of:

1.5+1.0*SIN(2*PI*T*1E6)+2.0*COS(2*PI*T*2E6)+3.0*SIN(2*PI*T*3E6)

This source generates a waveform that is a mix of the following pure sinusoids:

F	Amplitude
0	1.5
1E6	1.0
2E6	2.0
3E6	3.0

There are no capacitors or inductors in the circuit, so there is no initial transient to obscure the steady state output we're interested in.

Select **Transient** from the **Analysis** menu. This dialog box illustrates several important steps in getting good results with DSP functions.

The Analysis limits dialog box looks like this:

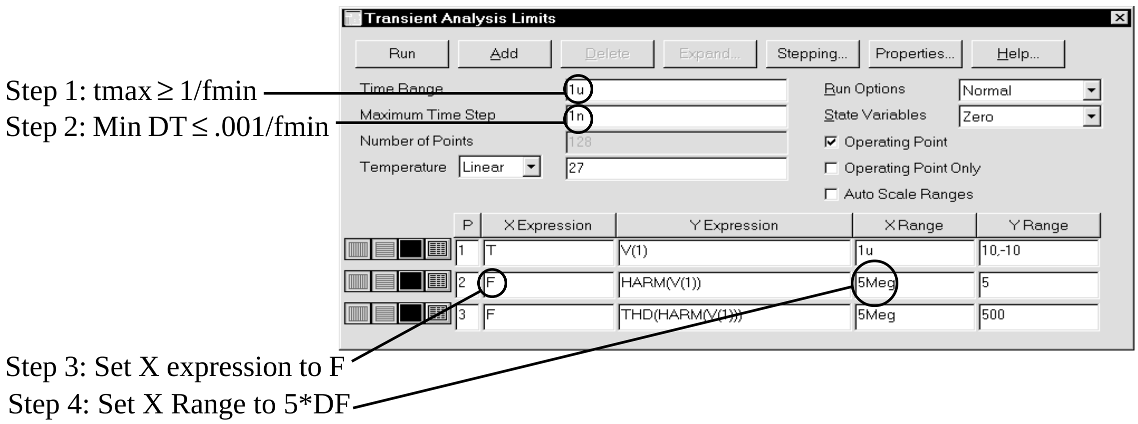


Figure 24-2 The FFT1 Analysis Limits dialog box

Step 1:

The DSP functions depend upon a good choice of the sampling frequency, DF, and a sufficient number of data points, N. The value of DF is

DF = F1 = 1 / *tmax*

The format of the Time Range field is:

<*tmax*>[,<*tmin*>]

If *tmin* is zero, as it usually is, then <*tmax*> is just the value of the Time Range field. Then,

DF = F1 = 1 / Time Range value.

This requires that:

tmax ≥ 1/FMIN   where FMIN is the smallest expected harmonic.

This necessity is embodied in Step 1:

Step 1: Set tmax ≥ 1/FMIN

If you're unsure of what FMIN is, assume it is equal to the lowest frequency of any input signal sources present in the network. If there are no sources present, as for example, in an oscillator, use the expected oscillator frequency.

Step 2:
The accuracy of the DSP will be controlled by the minimum time step. The smaller the better, but the smaller the DT the longer the analysis will take. As a good rule of thumb make DT 0.1% of the period of the FMIN. This leads to Step 2.

Step 2: Set the Minimum Time Step to 0.001 / FMIN.

Step 3:
For any DSP with a time domain argument, such as FFT and HARM, you must have F (frequency) for the X expression: This leads to Step 3.

Step 3: Set the X expression to F.

Step 4:
If N is large, then the horizontal axis automatically scales to:

N*DF

This is usually too compressed to see the frequencies of interest. This leads to Step 4.

*Step 4: Set the X Range to a small multiple of DF, say 5*DF to 20*DF.*

Step 5:
DSP functions assume that the waveform is periodic. If the circuit has non-periodic initial transients, they will generate errors in the DSP calculations. Remove these by using the DSP dialog box. Since this circuit has no capacitors or inductors, there are no initial transients, so the precaution is not necessary here.

Step 5: Set the Lower Time Limit in the DSP Parameters dialog box to exclude initial transients.

Press F2 to run the simulation. The results look like this:

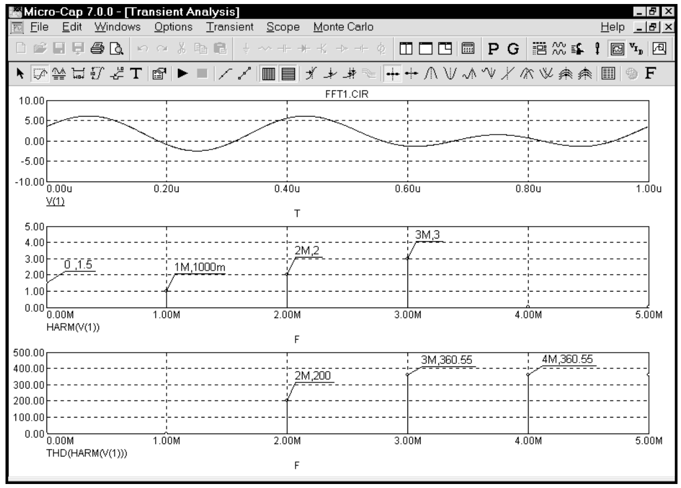


Figure 24-3 The graph with tmax = 1u

The second graph, showing the HARM(V(1)), produces the expected results:

DC	0	1.5
1	1Mhz	1.0
2	2Mhz	2.0
3	3Mhz	3.0

To illustrate the impact of changing tmax, change Time Range to 10u and change the X Range of the first waveform, V(1) to 10u. Press F2 to run the analysis:

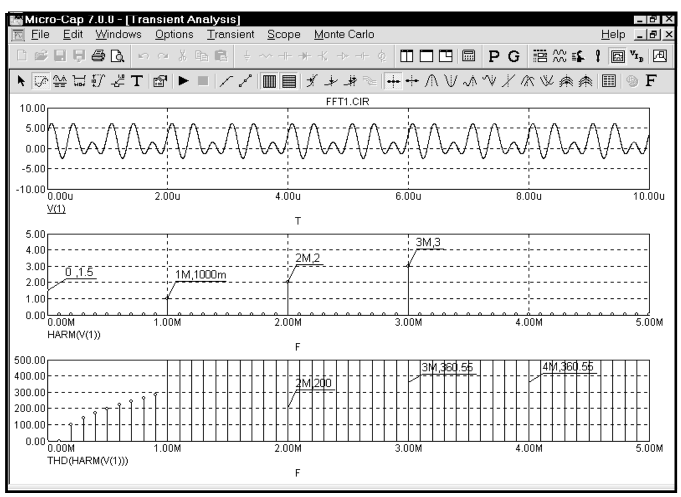


Figure 24-4 The graph with tmax = 10u

Notice that the HARM() function still correctly plots the 0Mhz, 1MHz, 2MHz, and 3Mhz values, but the THD function has changed. As specified the THD function uses the first harmonic or reference frequency. Because we changed tmax from 1u to 10u, the first harmonic frequency changed:

Old first harmonic = 1Meg = 1/1u.

New first harmonic = 100K = 1/10u.

The reference frequency shifted from 1Mhz to 100Khz.

To avoid this problem, you can specify the optional second parameter, FR. For example, in this case we would use:

THD(HARM(V(1)),1MEG)

The program scans the N*DF values and finds a match at N=10 because

10*DF = 10*100Khz = 1Mhz.

It then uses the 10th harmonic (1Mhz) as the THD reference frequency. The plot then looks like this:

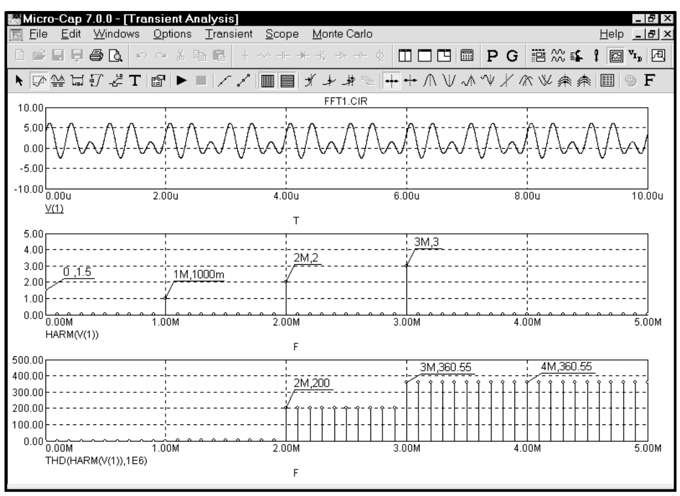


Figure 24-5 The graph with tmax = 10u and FR = 1Mhz

Notice that the THD now shows the expected values of 100% at 1Mhz, 200% at 2Mhz, and so on.

The DSP Parameters dialog box

Normally the FFT parameters N and DF are obtained implicitly from the analysis dialog box. That is, DF = 1/ <*tmax*> and N = the binary number closest to the actual number of data points used in the analysis.

To obtain more control, use the DSP dialog box. This dialog box is accessed from the Transient or AC menu. The transient analysis version looks like this:

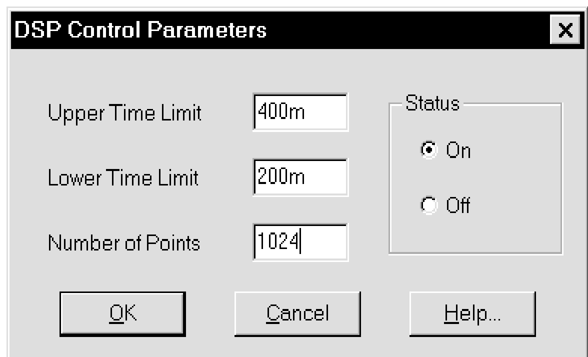


Figure 24-6 FFT7 transient analysis DSP dialog box

The AC analysis version looks like this:

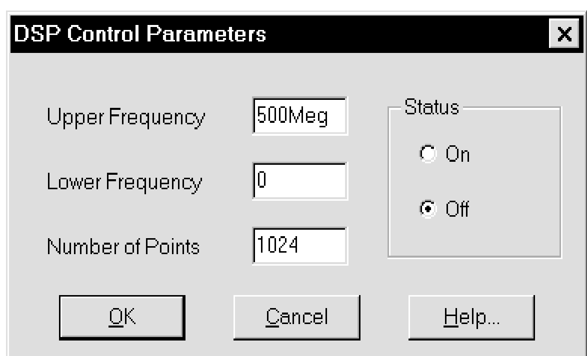


Figure 24-7 FFT7 AC analysis DSP dialog box

Each of these dialog boxes lets you specify the time or frequency range to apply the DSP function to and the number of data points to use in the FFT routine. The ability to limit the time range is useful in transient analysis, where it is used to eliminate initial transients in the waveform being analyzed.

To illustrate how this is done, load the file FFT7. Run transient analysis. The screen looks like this:

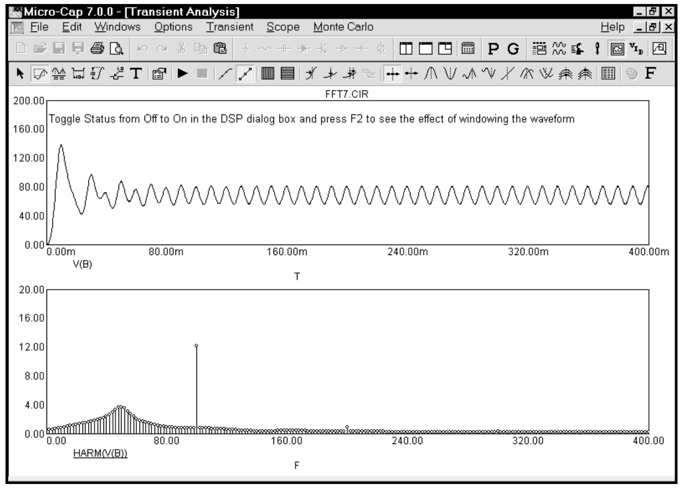


Figure 24-8 The graph with the initial transients

The plot of V(B) includes initial, *non-periodic* transients. The FFT is meant to be applied to *periodic* waveforms and any lack of periodicity shows up as spurious low frequency harmonics. To eliminate the transients and the attendant harmonics, open the DSP dialog box. Select the **DSP Parameters** item from the **Transient** menu and click the On button in the dialog box. Click on the OK button. This enables the settings already stored there, shown in Figure 24-6. These values specify that all DSP functions are to be windowed from 200ms to 400ms. This means that the portion of the waveform prior to 200ms is to be discarded. The portion from 200ms to 400ms is to be retained. The DSP dialog box also specifies the value of N, the number of data points to be used in the FFT routines.

Press F2 to rerun the analysis. The results look like Figure 24-9.

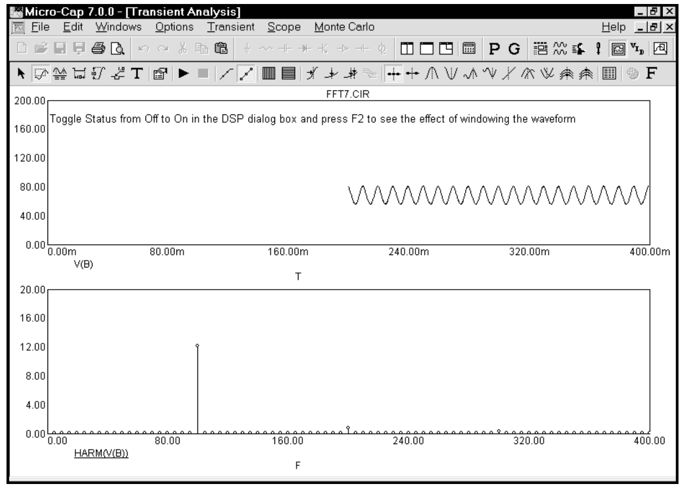


Figure 24-9 The graph without the initial transients

With the DSP dialog box enabled, the procedure used is this:

1. Remove the 0-200ms portion of the V(B) waveform.

2. Create a new waveform, by interpolating 1024 equally spaced points from the old waveform.

3. Apply the HARM function to this new waveform.

The top part of the graph shows the truncated, interpolated V(B) waveform with its 0-200ms portion removed. The bottom part shows the HARM(V(B)). With the removal of the waveform transient and its spurious harmonics, the true harmonic content can now be clearly seen.

Chapter 25 Libraries

What's in this chapter

This chapter describes the various libraries found in Micro-Cap. It explains how they are created, edited, and used:

This chapter describes:

- The Shape library
- The Component library
- The Model library
- The Package library

Features new in Micro-Cap 7

- The use of internal paths (folders) for data and libraries

The Shape library

The Shape library contains the graphical shapes used in schematics to represent components. Each shape is comprised of one or more graphical primitives. Shapes are created, edited, and maintained by the Shape editor. The name of the file that holds the standard Shape library is called STANDARD.SHP. This file is found on the same directory as the MC7.EXE program.

It is possible to have more than one shape library. Multiple libraries can be maintained and viewed with the Shape editor, but it is recommended that the original shape library, STANDARD.SHP, be left unmodified in its original state.

The Package library

The Package library contains the pin and package information needed to create netlist files for interfacing with external PCB packages. For each component that is included in a PCB netlist file, there must be an entry in the Package library. These entries are created, edited, and maintained by the Package editor. The name of the file that holds the original Package library supplied with MC7 is called STANDARD.PKG.

The Component library

Component libraries provide the circuit parts used in MC7 schematics. A component library stores the name of each component, the shape it uses, the electrical definition, component text placement, and pin information. All components, from resistors to macros and SPICE subcircuits are selected from a component library. There can be more than one component library file. The name of the original component library file supplied with MC7 is STANDARD.CMP. It is found on the same directory as the MC7.EXE program.

The library itself is organized hierarchically into these groups:

Analog Primitives
Analog Library
Digital Primitives
Digital Library

Animation
Filters (Present when at least one filter macro has been created by the user)
Macros (Present when at least one macro circuit has been created by the Make Macro command)

Components in the Analog Primitives and Digital Primitives parts of the library require the user to select a model name, or for simple components like a resistor, a value.

Components in the Analog Library and Digital Library sections of the library do not require the user to select a model name. The user need only select the desired part name. With these parts, the model or subcircuit name is equated automatically to the part name. Selecting a part name also selects the model, subcircuit, or macro name used to access the electrical modeling information.

Model statements and subcircuits for the parts in the Analog Library and Digital Library are already available in the Model library and are accessed with the default '.LIB NOM.LIB' statement implicit in every circuit.

It is possible to use or have more than one component library. Multiple libraries can be maintained, merged, and viewed with the Component editor, but it is recommended that the original library, STANDARD.CMP, should be left unmodified in its original state.

The Model library

The Model library provides the electrical modeling information required for any type of analysis or simulation. The MC7 model library is contained in the set of files listed in the general file NOM.LIB. If you examine the files referenced in this file, you will find that the models are provided in four forms:

Model parameters lists: These are lists of part names, types, and model parameters. The lists are stored in binary files using the extension LBR. These files can be viewed and edited only with the Model editor. Much of the Analog Library is stored in this form. The major exception is the Vendor section, which is mostly subcircuit models contained in text files using the extension LIB.

Model statements: These are conventional SPICE .MODEL statements. They hold the part names, types, and model parameters. They are stored in text files using the LIB extension. These are primarily used as a part of the subcircuit form of modeling.

Subcircuits: These are conventional SPICE subcircuits that describe the equivalent circuit for the part. The subcircuits are stored in text files using the LIB extension. All of the Digital Library is implemented with subcircuits.

Macros: These are conventional MC7 macros that describe the equivalent circuit for the part. The general macros are stored in schematic files using the CIR or MAC extension. The specific macro call that implements a particular part model is stored as a macro call in a text file using the LIB extension. Some parts of the Analog Library are implemented in this form.

Of these four basic forms, the most common are the model parameter lists and subcircuits.

Adding new parts to the Model library

To add new parts to the Model library you must do two things:

• Add the new part to the Component library (see Chapter 4).
• Add a .MODEL, .SUBCKT, or .MACRO statement that electrically models the part to a file and add the file name to the NOM.LIB file.

After these tasks are done, the new model/part name should show up on the Component menus.

How models are accessed

When you select an analysis, MC7 accesses electrical modeling information for any parts in the circuit that require models. How does it do this? It searches for the modeling information (model parameter lists, model statements, subcircuits, or macro statements) in one of the following places:

Local Search within the circuit:

- If the circuit is a schematic:
    - In the grid text or text area
    - In the file named in the File attribute (if the device has one).
    - In one or more files named in a .LIB *filename* statement.
    - In one or more files named in the default .LIB NOM.LIB statement.

- If the circuit is a SPICE text file:
    - In the circuit description text.
    - In one or more files named in a .LIB *filename* statement.
    - In one or more files named in the default .LIB NOM.LIB statement.

Global Search using the Library path(s):

MC7 always begins its search for the model information by looking *locally* in the circuit itself. If it doesn't find it there it looks *globally* in the LIBRARY folder specified at **File menu / Paths / Model Library and Include Files.** If more than one path is specified, it searches them in left to right order. For example, if the Library Path is:

C:\MC7\LIBRARY ; D:\OTHER ; E:\ELSEWHERE

MC7 will first search in C:\MC7\LIBRARY. If it fails to find the models it needs there it will search for them in D:\OTHER, and then lastly in E:\ELSEWHERE.

Chapter 26 Performance Functions

What's in this chapter

Performance functions are mathematical procedures designed to extract circuit performance measurements from curves generated during an analysis. This chapter describes their capabilities. It includes:

- What are performance functions?
- Performance functions defined
- The Performance Function dialog box

What are performance functions?

MC7 provides a group of functions for measuring performance-related curve characteristics. These functions let you measure performance related values such as rise time, fall time, pulse width, frequency, period, and many others. These functions may be used to analyze any curve generated during the course of an analysis. There are several ways in which performance functions may be used:

• Immediate mode: In this mode, you click on the Go To Performance  button and select a function from the list. The function is then applied to the curve specified in the Expression list box curve and the numeric result printed in the dialog box.

• Performance plots: In this mode, you do multiple runs by stepping numeric parameters and then create a plot showing how the performance function varies with the stepped variables. You can create two and three dimensional performance plots.

• Monte Carlo plots: In this mode, you run multiple Monte Carlo runs and then create a histogram showing how the performance function varies statistically.

Performance functions defined

MC7 provides a group of functions for measuring performance-related curve characteristics. These functions include:

Rise_Time	This function marks the N'th time the Y expression rises through the specified Low and High values. It places the cursors at the two data points, and returns the difference between the X expression values at these two points. This function is useful for measuring the rise time of time-domain curves.
Fall_Time	This function marks the N'th time the Y expression falls through the specified Low and High values. It places the cursors at the two data points, and returns the difference between the X expression values at these two points. This function is useful for measuring the fall time of time-domain curves.
Peak_X	This function marks the N'th local peak of the selected Y expression. A peak is any data point algebraically larger than the neighboring data points on either side. It places the left or right cursor at the data point and returns its X expression value.
Peak_Y	This function is identical to the Peak_X function but returns the Y expression value. This function is useful for measuring overshoot in time-domain curves and the peak gain ripple of filters in AC analysis.
Valley_X	This function marks the N'th local valley of the selected Y expression. A valley is any data point algebraically smaller than the neighboring data points on either side. It places the left or right cursor at the data point and returns its X expression value.
Valley_Y	This function is identical to the Valley_X function but returns the Y expression value. It is useful for measuring undershoot in time-domain curves and the peak attenuation of filters in AC analysis.

Peak_Valley This function marks the N'th peak and N'th valley of the selected Y expression. It places the cursors at the two data points, and returns the difference between the Y expression values at these two points. This function is useful for measuring ripple, overshoot, and amplitude.

Period The period function accurately measures the time period of curves by measuring the X differences between successive instances of the average Y value. It does this by first finding the average of the Y expression over the simulation interval where the Boolean expression is true. Then it searches for the N'th and N+1'th rising instance of the average value. The difference in the X expression values produces the period value. Typically a Boolean expression like "T>500ns" is used to exclude the errors introduced by the non-periodic initial transients. This function is useful for measuring the period of oscillators and voltage to frequency converters, where a curve's period usually needs to be measured to high precision. The function works best on curves that pass through their average value once per fundamental period. It will not work well on curves that contain significant harmonics of the fundamental. The function places the cursors at the two data points, and returns the difference between the X expression values at these two points.

Frequency This is the numerical complement of the Period function. It behaves like the Period function, but returns 1/Period. The function places cursors at the two data points.

Width This function measures the width of the Y expression curve by finding the N'th and N+1'th instances of the specified Level value. It then places cursors at the two data points, and returns the difference between the X expression values at these two points.

High_X This function finds the global maximum of the selected branch of the selected Y expression, places either the left or the right cursor at the data point, and returns its X expression value.

High_Y This function finds the global maximum of the selected branch of the selected Y expression, places either the left or the right cursor at the data point, and returns its Y expression value.

Low_X This function finds the global minimum of the selected branch of the selected Y expression, places either the left or the right cursor at the data point, and returns its X expression value.

Low_Y This function finds the global minimum of the selected branch of the selected Y expression, places either the left or the right cursor at the data point, and returns its Y expression value.

X_Level This function finds the N'th instance of the specified Y Level value, places a left or right cursor there, and returns the X expression value.

Y_Level This function finds the N'th instance of the specified X Level value, places a left or right cursor there, and returns the Y expression value.

X_Delta This function finds the N'th instance of the specified Y expression range, places cursors at the two data points, and returns the difference between the X expression values at these two points.

Y_Delta This function finds the N'th instance of the specified X expression range, places cursors at the two data points, and returns the difference between the Y expression values at these two points.

X_Range This function finds the X range (max - min) for the N'th instance of the specified Y range. First it searches for the specified Y Low and Y High expression values. It then searches all data points between these two for the highest and lowest X values, places cursors at these two data points, and returns the difference between the X expression values at these two points. It differs from the X_Delta function in that it returns the difference in the maximum and minimum X values in the specified Y range, rather than the difference in the X values at the specified Y endpoints.

Y_Range This function finds the Y range (max - min) for the N'th instance of the specified X range. First it finds the specified X Low and X High expression values. It then searches all data points between these two for the highest and lowest Y values, places cursors at these two data points, and returns the difference between the Y expression values at these two points. It differs from the Y_Delta function in that it returns the difference in the maximum and minimum Y values in the specified X range, rather than the difference in the Y values at the specified X endpoints. This function is useful for measuring filter ripple.

Slope This function places cursors at the two data points that straddle the data point nearest the specified X value, and returns the slope between the two cursors.

Phase Margin This function finds the phase margin of a plot. A dB(expr) plot and a PHASE(expr) plot must be present for it to work properly. This function is only available in AC analysis.

The Performance Function dialog box

These functions may be used on a single analysis plot, in Monte Carlo analysis, and in 3D plotting. When a run is complete, click on the Go To Performance Function button. The dialog box looks like this:

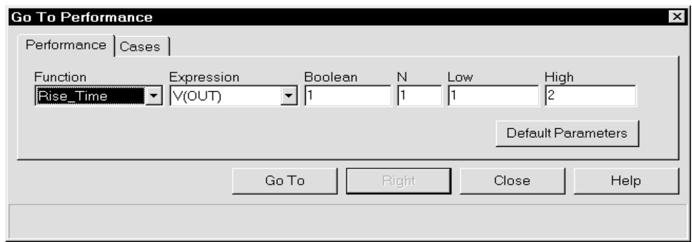


Figure 26-1 Performance Function dialog box

There are two panels: Performance and Cases. The Cases panel lets you select a branch if more than one is available due to stepping. The Performance panel lets you select a performance function to apply to the selected curve branch chosen from the Cases panel.

The Performance panel provides the following fields:

Function: This selects one of the Performance functions.

Expression: This selects the expression for the function to work on. Only expressions that were plotted during the run are available.

Boolean: This Boolean expression must be true for the performance functions to consider a data point for inclusion in the search. Typically this function is used to exclude some unwanted part of the curve from the function search. A typical expression here would be "T>100ns". This would instruct the program to exclude any data points for which T<=100ns.

N: This integer specifies which of the instances you want to find and measure. For example, there might be many pulse widths to measure. Number one is the first one on the left starting at the first time point. The value of N is incremented each time you click on the Go To button, measuring each succeeding instance.

Low: This field specifies the low value to be used by the search routines. For example, in the Rise_Time function, this specifies the low value at which the rising edge is measured.

High: This field specifies the high value to be used by the search routines. For example, in the Rise_Time function, this specifies the high value at which the rising edge is measured.

Level: This field specifies the level value to be used by the search routines. For example, in the Width function, this specifies the expression value at which the width is measured.

The buttons at the bottom provide these functions:

Go To (Left): This button is called Go To when the performance function naturally positions both cursors (as the Rise_Time and Fall_Time functions do). It is named Left when either the left or right cursor, but not both, would be positioned by the function. When named Left, this button places the left numeric cursor at the position dictated by the performance function.

Right: This button places the right numeric cursor at the position dictated by the performance function. For example, the Peak function can position either the left or right cursor.

Close: This button closes the dialog box.

Default Parameters: This button calculates default parameters for functions which have them. It guesses at a suitable value by calculating the average value or the 20% and 80% points of a range.

Performance functions all share certain basic characteristics:

A performance function at its core is a search: The set of data points of the target expression is searched for the criteria specified by the performance function and its parameters.

Performance functions find the N'th instance: Every performance function has a parameter N which specifies which instance of the function to return. The only exceptions are the High and Low functions for which, by definition, there is a maximum of one instance.

Each performance function call increments the N parameter: Every time a performance function is called, N is incremented after the function is called. The next call automatically finds the next instance, and moves the cursor(s) to subsequent instances. For example, each call to the Peak function locates a new peak to the right of the old peak. When the end of the curve is reached, the function rolls over to the beginning of the curve.

Only curves plotted during the run can be used: Performance functions operate on curve data sets after the run, so only curves saved (plotted) during the run are available for analysis.

Boolean expression must be TRUE: Candidate data points are included only if the Boolean expression is true for the data point. The Boolean expression is provided to let the user include only the desired parts of the curve in the performance function search.

At least PERFORM_M neighboring points must qualify: If a data point is selected, the neighboring data points must be consistent with the search criteria or the point is rejected. For example, in the peak function, a data point is considered a peak if at least PERFORM_M data points on each side have a lower Y expression value. PERFORM_M is a Global Settings value and defaults to 1. Use 2 or even 3 if the curve is particularly choppy or has any trapezoidal ringing in it.

Performance function plots

Performance Function plots may be created when multiple analysis runs have been done with temperature or parameter stepping. To illustrate, load the file PERF1. Run transient analysis. The plot should look like this:

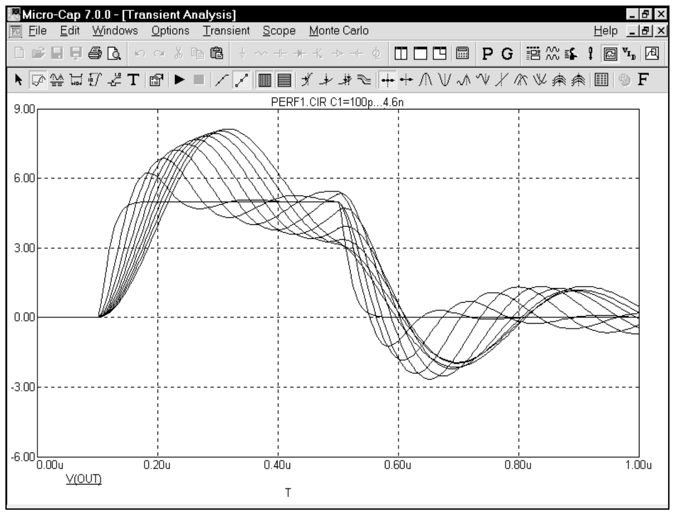


Figure 26-2 The transient analysis run

From the **Transient** menu, select **Performance Windows - Add Performance Window**. This presents the Properties dialog box for performance plots.

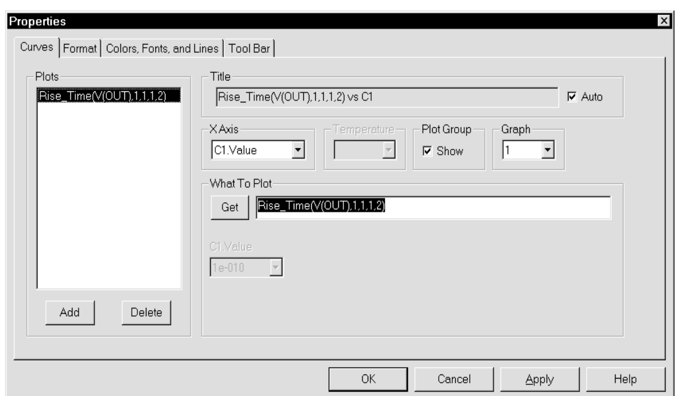


Figure 26-3 The Properties dialog box for performance plots

The Plot Properties dialog box lets you control the performance plot window. It can be used for controlling the performance plot display after or even before the plot. The dialog box provides the following choices:

- **Plot**

  *Curves***:** This list box lets you select the curve that the performance functions will operate on. The Add and Delete buttons at the bottom let you add or delete a particular performance plot.

  Title: This field lets you specify what the plot title is to be. If the Auto button is checked, the title is automatically created from the circuit name and analysis run details.

  X Axis: This lets you select the stepped variable to use for the X axis.

  Temperature: If temperature was also stepped, each value creates a separate curve. This lets you select the temperature whose curve the performance functions will operate on.

  Curve: This check box controls whether the selected performance curve will be plotted or not. If you want to hide the curve from the plot group, remove the check mark by clicking in the box.

  Plot Group: This group's list box controls the plot group number.

  What To Plot: This group lets you select the performance function and its parameters.

  Stepped Variables List: If more than one variable has been stepped, there will be one or more List boxes. These list boxes let you select which instance of the stepped variable(s) the performance functions will process. For instance, if you stepped R1, L1, and C1 through 5 values each and chose R1 for the x axis variable, there would be a list box for L1 and one for C1. From these two lists, you could select any of the 5 X 5 = 25 possible curves and plot a performance function versus the R1 value.

- **Format**

  Curves: This list box lets you select the plot that the other fields apply to.

X: This group includes:

Range Low: This is the low value of the X range used to plot the selected curve.

Range High: This is the high value of the X range used to plot the selected curve.

Grid Spacing: This is the distance between X grids.

Bold Grid Spacing: This is the distance between bold X grids.

Scale Format: This accesses a dialog box where you select the numeric format used to print the X axis scale.

Value Format: This accesses a dialog box where you select the numeric format used to print the curve's X value in the table below the plot in Cursor mode and in the tracker boxes.

Auto Scale: This command scales the X Range and places the numbers into the Low, High, and Grid Spacing fields. The effect on the plot can be seen by clicking the Apply button.

Log: If checked this makes the scale log.

Y: This group provides a similar set of commands for the Y axis group.

Same Y Scales: Enabling this check box forces the Auto Scale command to use a single common scale for all plots within a plot group. If the box is unchecked, the Auto Scale function will generate multiple scales for each plot in a plot group.

Use Common Formats: Clicking this button copies the X and Y formats of the selected curve to format fields of all plots.

- **Colors, Fonts, and Lines**
  Objects: This list box lets you select the object that the other commands (color, font, lines) apply to. These include:

  General Text: This is text used for axis scales, titles, cursor tables, and curve name.

Grid: This is the analysis plot grid.

Graph Background: This is the analysis plot background.

Window Background: This is the window background.

Select: This is the Select mode.

Select Box: This is the Select mode box.

Initial Object: This governs the initial properties of analysis text, graphical objects, and numeric tags added to the plot. Object properties can be changed after they are added to a plot by double-clicking on them.

Tracker: These are the trackers.

Plot Names: These are the curve names.

Varied (Color): This group whose name changes to reflect the selected object, lets you change its color.

Curve Line: This group lets you change the color, width, and pattern of the plot object. The Rainbow option is not available. Windows 95 and 98 have pattern control only if the width is equal to 1. NT systems provide pattern control for any width.

Font: This group lets you change the font of the selected object.

Size: This group lets you change the text size of the selected object.

Style: This group lets you change the text style of the selected object.

Effects: This group lets you change text effects of the selected object.

Sample: This group shows a sample object using the currently selected font, color, width, and pattern properties appropriate to the object.

- **Tool Bar:**

This page lets you select the buttons that will appear in the local tool bar area below the Main tool bar.

Tool Bar: This list box lets you select the different local tool bars.

Buttons: This box lets you select the buttons that are to appear in the selected tool bar.

Show Button: If checked, the selected button is shown in the selected tool bar.

Top: If enabled, the tool bar is placed at the top part of the window.

Left: If enabled, the tool bar is placed at the left part of the window.

All On: This command places all buttons in the tool bar.

All Off: This command places no buttons in the tool bar.

Default: This command places the default set of buttons in the tool bar.

The four buttons at the bottom have the following functions:

OK: This button accepts all changes, exits the dialog box, and redraws the performance plot. Subsequent runs will use the changed properties and they will be retained in the circuit file, if the file itself is later saved.

Cancel: This button rejects all changes, exits the dialog box, and redraws the analysis plot using the original properties.

Apply: This button displays the analysis plot using the current settings in the dialog box to show how the display would be affected by the changes. The changes are still tentative, until the OK button is clicked.

Help: This button accesses local help information.

Click on the OK button. This selects the Rise_Time function with default parameters and creates the following performance plot.

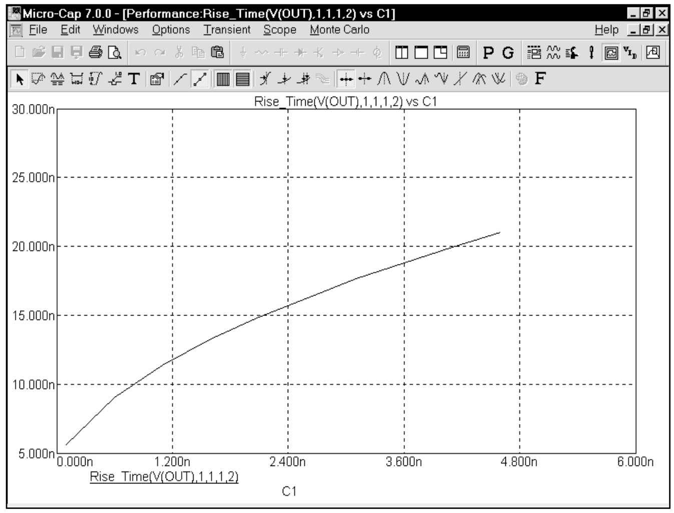


Figure 26-4 Rise time performance plot

This plots shows how the rise time of V(1) (as measured between 1 and 2 volts) varies with the value of C1, the only stepped variable. Double-click on the graph or press F10. This invokes the Properties dialog box. Click on the Add button. Click on the Get button and select Fall_Time from the Function list box. This adds the Fall_Time function plot to the performance window. It looks like this:

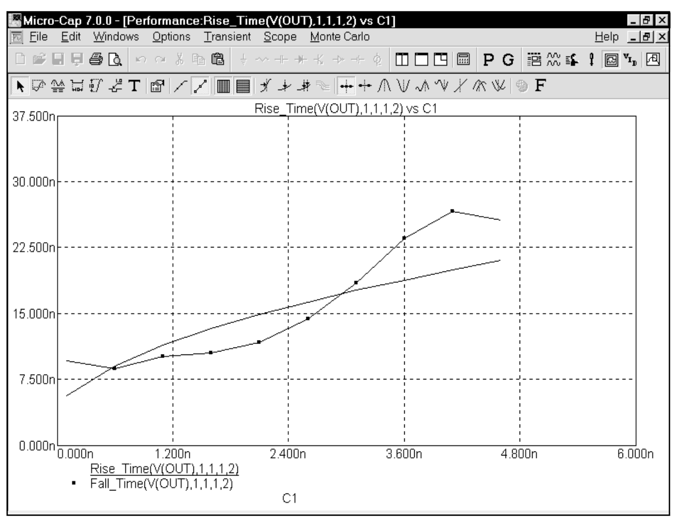


Figure 26-5 Rise and fall time performance plots

The window shows plots of both the Rise_Time and Fall_Time functions. Press F11 and click the Yes item in the Step It group of the Parameter 2 panel. Hit OK and press F2. The runs look like this:

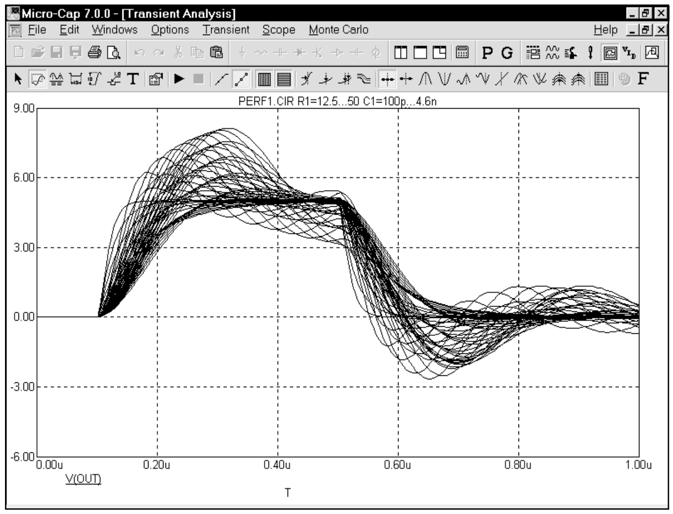


Figure 26-6 Stepping both C1 and R1

Press CTRL + F6 until you see the performance plot. It should look like this:

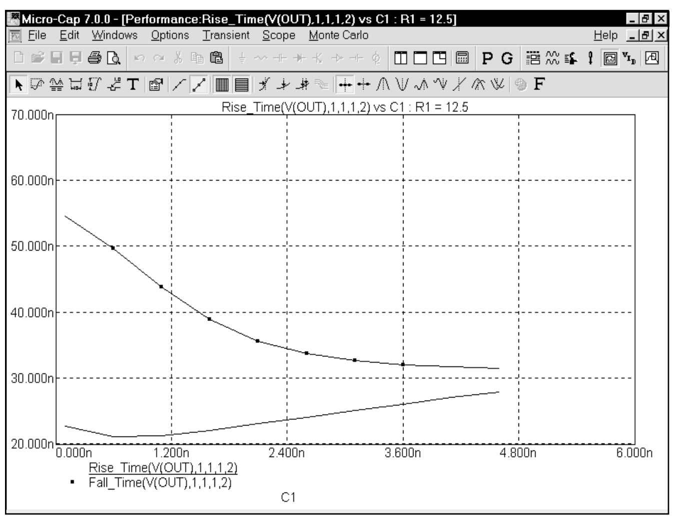


Figure 26-7 Performance plots for R=12.5

In the new runs, C1 varies from .1n to 4.6n and R1 varies from 12.5 to 50. The plot shows the functions vs. the C1 variable only. What about R1? Since a 2D plot can only show functions versus one variable, the plot itself is prepared for a single value of the other stepped variable. In this case the plot is for R1=12.5 only. To see other values, press F10, and select a new value from the R1.Value list box for each of the two plots. Here is the plot for R=25.

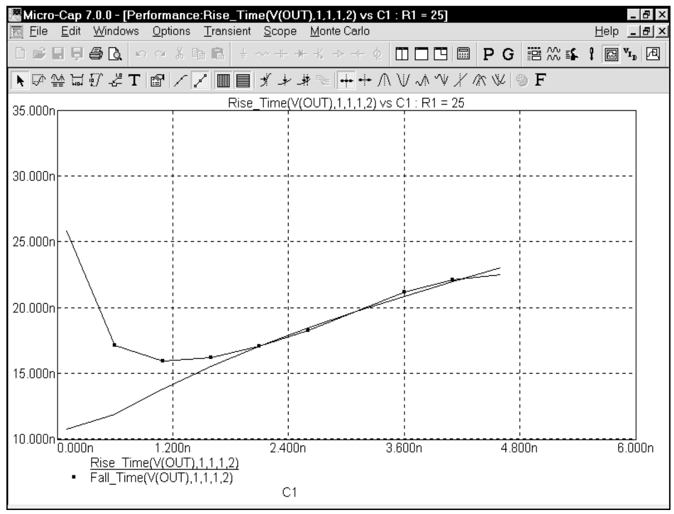


Figure 26-8 The Fall_Time performance plot for R=25

When two or more variables are stepped, there is another way to view the results. You can do a 3D performance plot. To illustrate, select **Add 3D Window** from the **3D Windows** item on the **Transient** menu. From the 3D Properties dialog box select the Performance item from the Y Axis Type group, and select the Fall_Time function from the Function list box in the What To Plot group. Click OK. The result looks like Figure 26-9.

This 3D plot shows the Fall_Time function plotted along the (vertical) Y axis vs. C1 along the (horizontal) X axis and R1 along the (normal to paper) Z axis. It is the same data but presented a different way. In fact if you examine the 3D plot, you can see the 2D fall time plot within it.

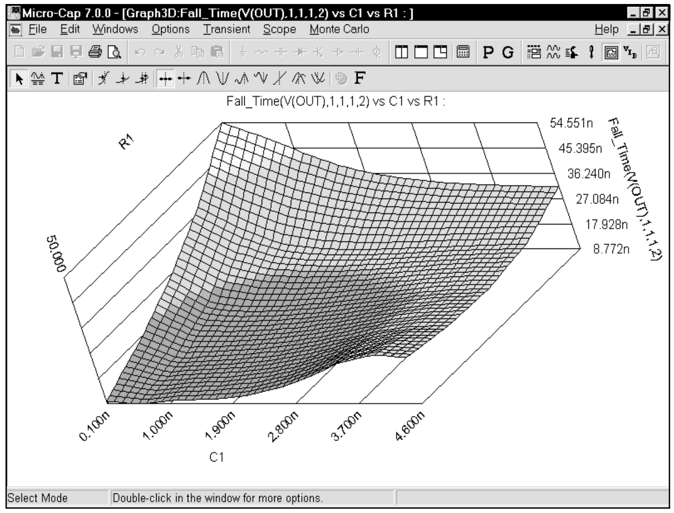


Figure 26-9 The 3D Fall_Time performance plot

Chapter 27 3D Graphs

What's in this chapter

Micro-Cap provides 3D graphs for plotting and visualizing simulation results. This chapter shows you how to use them.

The chapter is organized as follows:

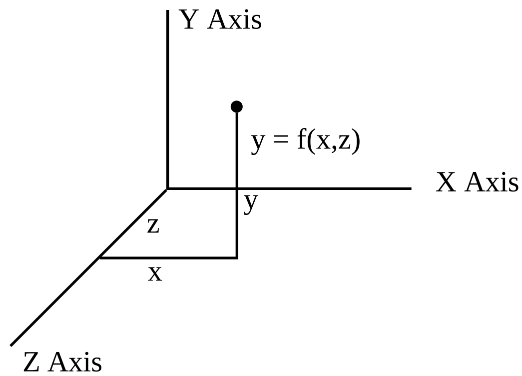
- How 3D plotting works
- 3D example
- The 3D dialog box
- Cursor mode in 3D
- 3D performance functions
- Changing the plot orientation
- Scaling in 3D

Features new in Micro-Cap 7

- Ability to use performance function expressions in 3D performance plots instead of single performance functions.

How 3D plotting works

In a 3D plot there are three variables. Each is associated with one of three mutually orthogonal axes, X, Y, and Z. The Z axis may be thought of as pointing out from the paper toward the user. The X and Y axes are coplanar with the paper.



The Y axis can plot one of two things:

- Any *expression* (curve) plotted during the run.
- *Performance function expressions* using curves plotted during the run.

Expressions can be plotted if either temperature stepping or parameter stepping has been employed during the run.

Performance functions expressions can be plotted if at least two variables have been stepped. One of the two can be temperature.

The first stepped variable is plotted along the Z axis. For example, if you stepped temperature and chose to plot V(1) on the Y axis, then T (Time) would normally be plotted on the X axis, and the stepped variable, temperature, would normally be plotted on the Z axis. If you stepped R(R1) and you chose to plot V(1) on the Y axis, then T (Time) would normally be plotted on the X axis, and the stepped variable, R(R1), would normally be plotted on the Z axis.

T or the second stepped variable is plotted along the X axis. For example, if you stepped R(R1) and C(C1), then you could plot an expression like V(1) vs. T on the X axis and either R(R1) or C(C1) on the Z axis, or a performance function like Rise_time vs. R(R1) on the X axis and C(C1) on the Z axis.

3D example

To illustrate how 3D plotting works, load the file 3D1. It looks like this:

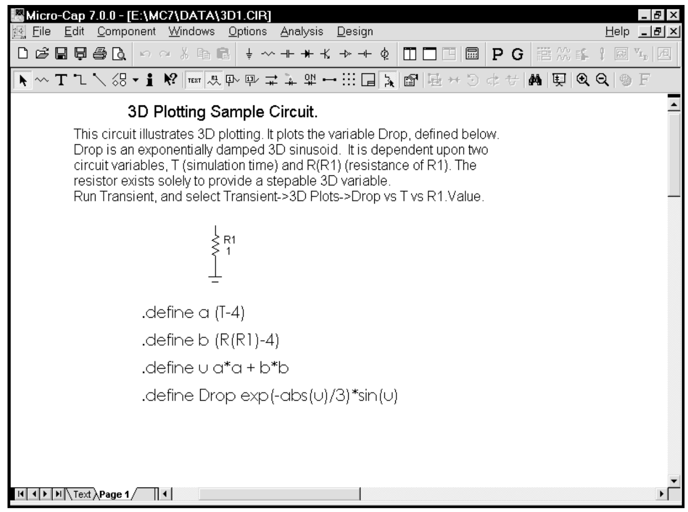


Figure 27-1 The 3D circuit

The circuit consists of a resistor, R1, and a symbolic variable, Drop, which is dependent upon the resistance of R1 and T. Select **Transient** from the **Analysis** menu. The analysis limits look like this:

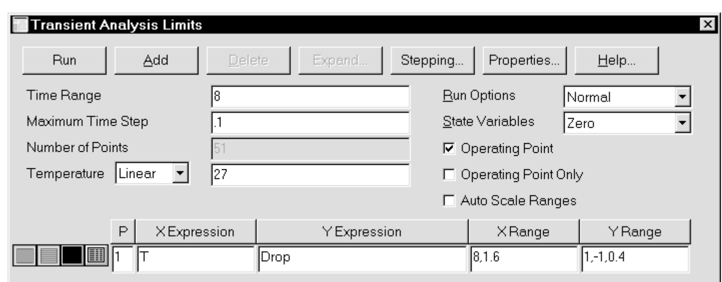


Figure 27-2 The analysis limits of 3D1

According to these limits T will vary from 0 to 8 with maximum steps of .1. We are plotting the value of the Drop variable. Since there are no capacitors or inductors in this circuit, the actual time step size starts small and quickly ramps up to the maximum of .1.

The Stepping dialog box looks like this:

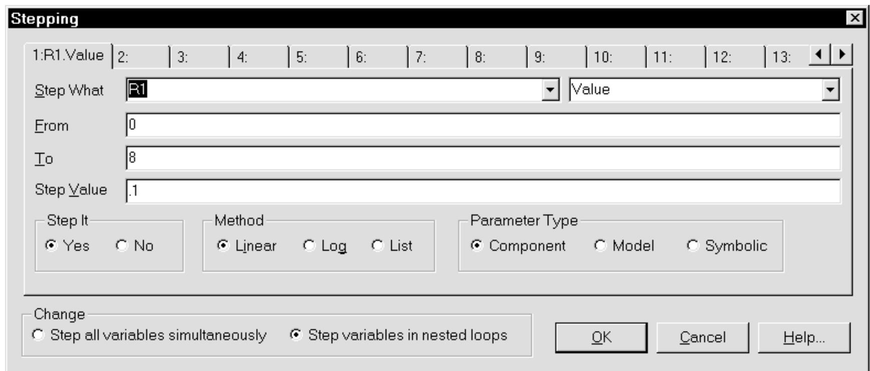


Figure 27-3 R(R1) is stepped from 0 to 8

According to this dialog box, the value of R1 will be linearly stepped from 0 to 8 in steps of 0.1. Click OK. Press F2 to start the run. The results look like this:

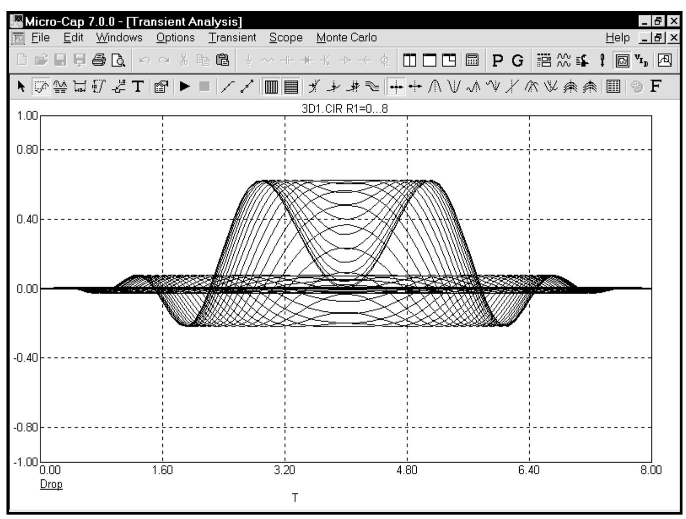


Figure 27-4 The 2D analysis plot of Drop

The plot shows the value of the variable Drop, with T varying from 0 to 8 and R(R1) varying from 0 to 8. There are 81 runs (1+8.0/0.1).

Select the **Drop vs. T vs. R1.Value** item from the **3D Windows** item from the **Transient** menu. This plots the value of the symbolic variable, Drop, along the Y axis, vs. T along the X axis, vs. R(R1) along the Z axis like this:

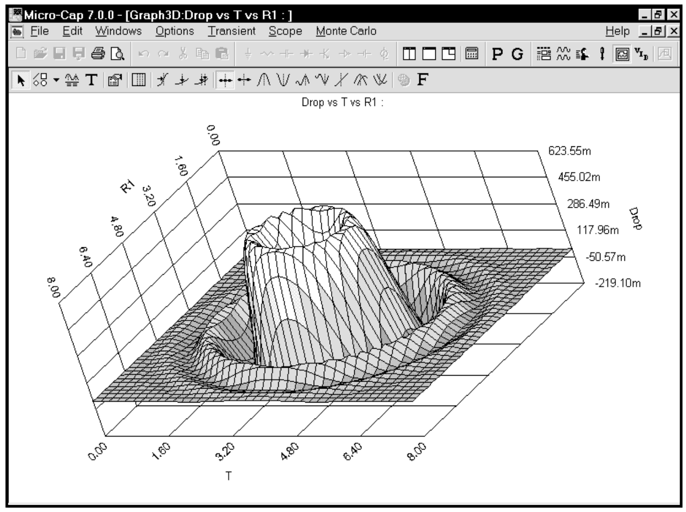


Figure 27-5 The 3D plot of Drop

In this case, we stepped only one variable R(R1). During the run, MC7 calculated 81 values of R(R1), the Z axis variable, and a little more than 81 values of T, the X axis variable, for each of the 81 values of R(R1). For each of these approximately 6400 values, it calculated the value of the Drop function.

When a 3D plot is requested, MC7 *interpolates* the specified number of data points from the simulation run at equally spaced values. It then produces a 3D plot where the intersection of the X and Z isolines mark the interpolated data points. The patches between the grid lines are colored according to the value at the adjacent isoline. The color used (color spectrum, gray, clear, etc.) is specified in the 3D Plot Properties dialog box.

The Properties dialog box for 3D plots

When you add or change a 3D graph you employ its properties dialog box. To see what it looks like press F10. The Plot panel of the dialog box looks like this:

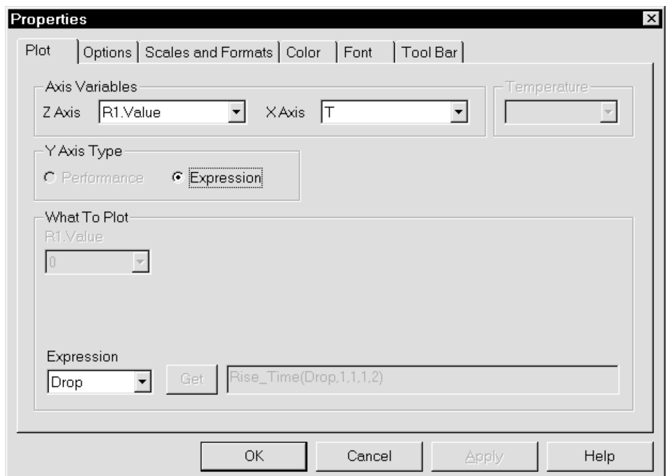


Figure 27-6 The Plot panel of the 3D dialog box

The dialog box controls all aspects of the 3D graph. The command buttons at the bottom access the principal features of the graph:

OK: This accepts any changes made and exits the dialog box.

Cancel: This ignores any changes made and exits the dialog box.

Apply: This redraws the plot behind the dialog box using the current settings so you can see what the settings look like before committing to them.

Help: This accesses the Help system.

There are six panels:

Plot: This button accesses the main plot features of the 3D graph. In particular it includes these items:

Axis Variables:

Z Axis: This field lists the variables available for plotting along the Z axis. These include any stepped variable.

X Axis: This field lists the variables available for plotting along the X axis. If the Y Axis Type Performance item is enabled, this list will include all stepped variables except the one already selected for the Z axis. If the Performance item is not selected this list will include only the unique X expressions from the Analysis Limits dialog box. Usually this means only T (Time) or F (Frequency).

Temperature:

If more than one temperature has been run, and temperature is not being used as an axis variable, this field lets you select which run to plot.

Y Axis Type:

Performance: This option selects performance functions for the Y axis variable. If this option is enabled, any of the performance functions operating on any expression printed or plotted during the run may be used for the Y axis variable.

Expression: This option selects any of the expressions printed or plotted during the run.

What to Plot:

Top Row: This row lists excess stepped variables not already selected as axis variables. If a variable is not being used as an axis variable, you can select which of its stepped values you want to plot. Each value produces a different 3D surface plot.

Bottom Row: This row lists one of two things:

If the Performance item has not been selected:
The row lets you select an expression from the Analysis Limits Y Expression fields for plotting on the Y axis.

If the Performance item has been selected:
The row lets you select a performance function and its related data fields, Expression, Boolean, N, etc.

Options: This button accesses items which control the existence, quantity, or content of certain graph features. It looks like this:

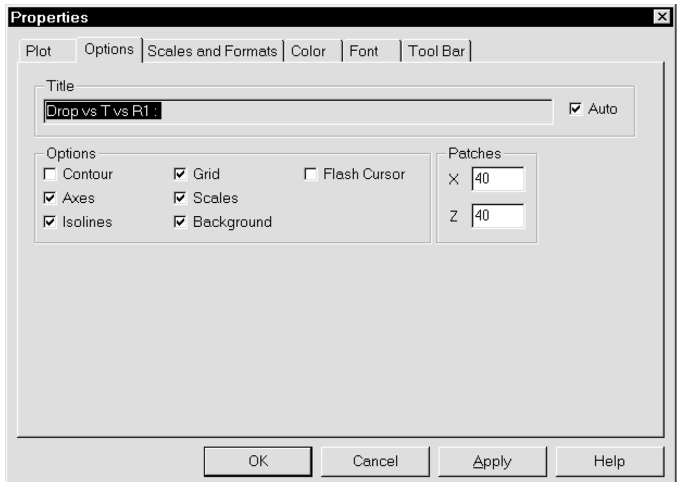


Figure 27-7 The Options panel of the 3D dialog box

Title: This is the title of the 3D plot. The initial title is always of a form that reflects the names of the axis variables. Its format is "Y axis variable vs. X axis variable vs. Z axis variable". You can edit this field to change the title text if Auto is disabled. However, if the Auto field is enabled, the title will automatically revert to the axis name form as changes are made to axis variable names.

Options: These items control the existence of certain graph features:

Contour: This option plots a 2D contour of the Y axis variable.

Axes: If enabled, this option adds the three axes, X, Y, and Z. Note that the axes are colinear with the grid, so to see the effect of this feature, the Grid feature must be disabled.

Isolines: This option draws isolines on the plot along which X and Z are constant. The intersection of these lines comprise the boundaries of the colored surface patches of the 3D plot.

Grid: This option draws parallel grid lines in each coordinate plane to delineate equal fractional parts of the scale.

Scales: This option draws numeric scales along each axis.

Background: This option paints the three coordinate planes clear when disabled, regardless of their color settings.

Flash Cursor: This option flashes the two numeric cursors for easier visibility.

Patches:

X: This controls the number of patches along the X axis. This number defaults to 40. It may be as high as you wish, but remember, the number of total patches is equal to the product of the number of X and Z patches. Picking 40 for each axis requires 40*40 = 1600 patches. Picking 1000 for each axis requires 1000*1000 = 1,000,000 patches. Each patch must be imaged through a set of 3D trigonometric transforms. These take time to calculate and slow down the drawing. Also, with more than 200 patches the surface is covered with isolines lines spaced so close it's hard to see the surface, so limit this value to 20 to 200.

Z: This controls the number of patches along the Z axis. The same considerations as for the X axis apply here as well.

Scales and Formats: This panel accesses the numeric format features of the 3D graph. Its options are shown below:

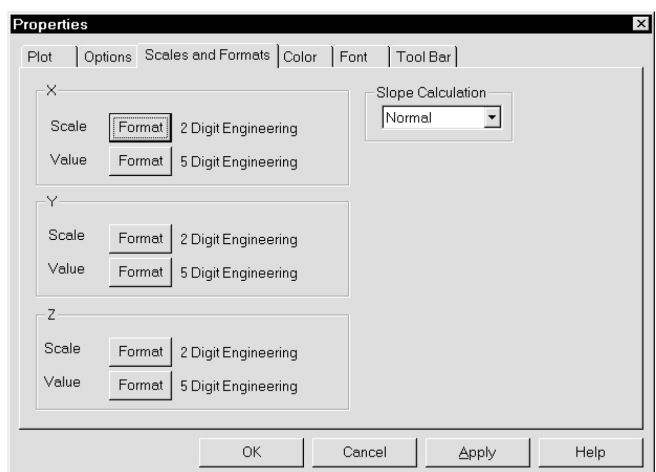


Figure 27-8 The Format panel of the 3D dialog box

For each of the three axes, X, Y, and Z, you can control the Scale and Value numeric format.

Scale Format: This is the numeric format used to print the X axis scale. There are two basic formats:

Value Format: This numeric format is used to print the curve's X, Y, and Z values in the table below the plot in Cursor mode. Its format is the same as the scale format described above.

• **Slope Calculation:** This lets you select among three forms of slope calculation: Normal, dB/Octave, and dB/Decade.

Color: This panel accesses the color features. The panel looks like this:

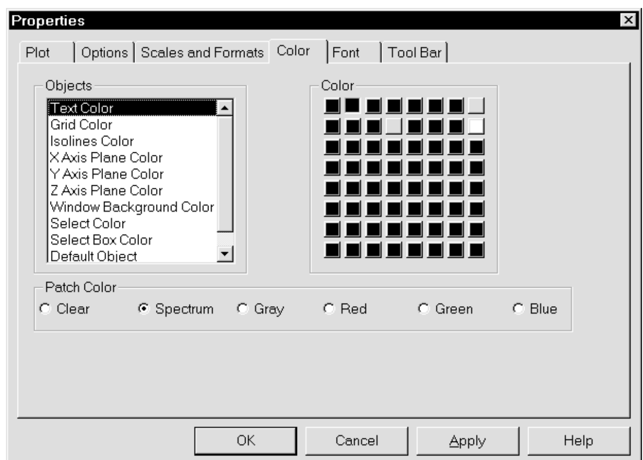


Figure 27-9 The Color panel of the 3D dialog box

This panel lets you control the color of the following features:

Objects: This group includes the color of the grid, text, isolines, X, Y, and Z axis planes, and the window background. To change an object's color, first select it, then select a new color from the palette.

Patch Color: This panel lets you control the color of the patches that comprise the 3D surface. You can choose from:

- **Clear:** This option leaves the patches transparent, producing what looks like a wire mesh if the Isolines option is enabled, or no plot at all if it isn't enabled.

- **Spectrum:** This option colors the patches with the colors of the spectrum ranging from blue for the lowest values to red for the largest values.

- **Gray:** This option colors the patches with shades of gray that range from black for the low values to white for the high values.

- **Red:** This option colors the patches with shades of red that range from dark red for the low values to light red for the high values.

- **Green:** This option colors the patches with shades of green that range from dark green for the low values to light green for the high values.

- **Blue:** This option colors the patches with shades of blue that range from dark blue for the low values to light blue for the high values.

- **Font**
This standard panel lets you select font, style, size, and effects for all of the text used in the 3D plot.

- **Tool Bar**
This standard panel lets you select the buttons that will appear in the local tool bar of the 3D window.

To add a 3D plot window, select **Transient / 3D Windows / Add 3D Window**. This displays a new 3D Properties dialog box and lets you select the desired plot features.

Cursor mode in 3D

As with normal 2D plots, a Cursor mode is available to numerically examine the 3D plot. Press F8 or click on the Cursor mode  button. The 3D plot will now look like this:

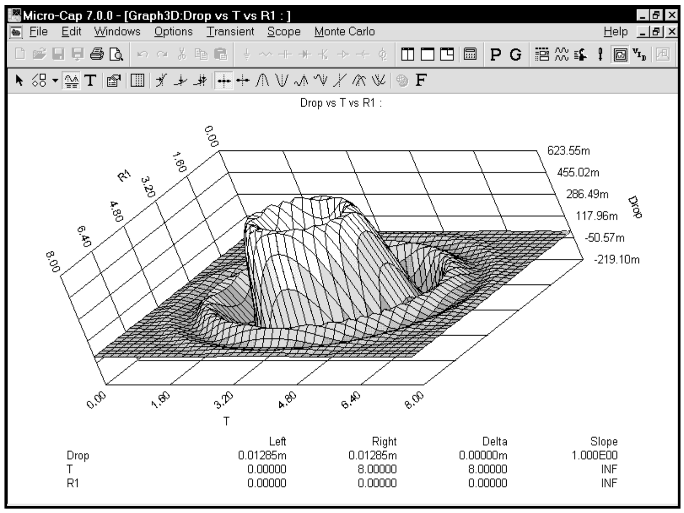


Figure 27-11 The 3D plot in Cursor mode

Click the mouse somewhere near the middle of the 3D surface plot. A cursor should appear. Press the LEFT ARROW key several times. Press the RIGHT ARROW key several times. Notice that the cursor selects a changing set of X axis, or T, values, but moves along a grid of constant Z axis, or R(R1), value.

Press the DOWN ARROW key several times. Press the UP ARROW key several times. Notice that the cursor selects a changing set of Z axis, or R(R1), values, but moves along a grid of constant X axis, or T, value.

All of the usual Cursor function modes are available. In particular the Next, Peak, Valley, High, Low, Inflection, Global High, Global Low, Go To X, Go To Y, and Go To Performance modes are available. For example, click on the Peak button. Press the RIGHT ARROW cursor key several times. The left numeric cursor is placed on each successive peak of the constant Z, or R(R1), curve.

3D Performance functions

To illustrate how to select performance functions for 3D plotting, load the file 3D2. Select **AC** from the **Analysis** menu. Press F2 to start the runs. The runs look like this:

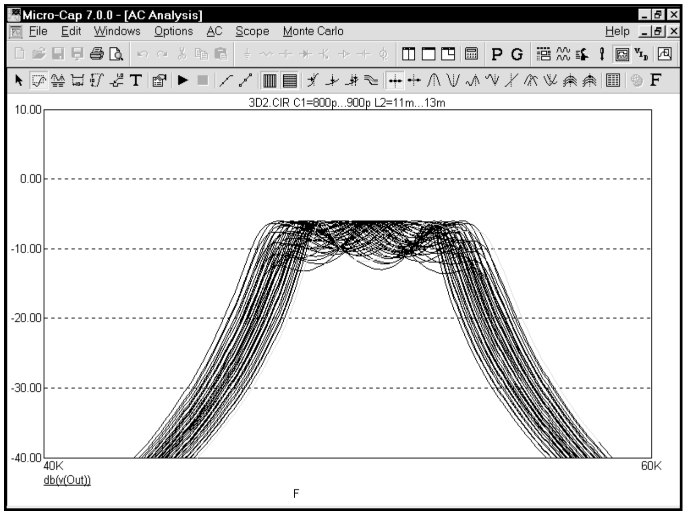


Figure 27-12 The 3D2 runs

Select **AC / 3D Windows / Width...** This produces the following plot:

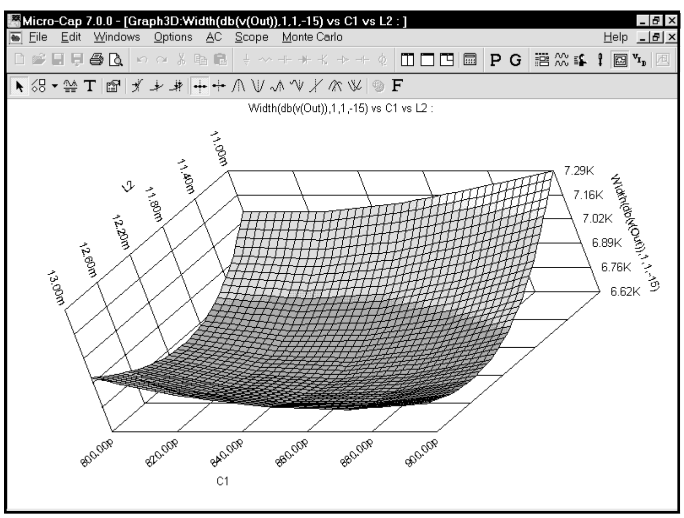


Figure 27-13 The plot of Width vs. L2 and C1

Here we have plotted the Width performance function:

Width(db(v(Out)),1,1,-15)

This performance function measures the width of the db(V(Out)) function at a Boolean of 1 (always true), first instance, at the Y value of -15. This effectively measures the -15 db bandwidth of the bandpass filter.

The value of L2 is stepped from 11mh to 13mh, in steps of .25mh, while stepping C1 from 800pf to 900pf in steps of 25pf. This produces 9*5, or 45 runs. The width is measured for each of these runs and the 3D graph plots the value of the Width function vs. the C1 value along the X axis and the L2 value along the Z axis. The plot shows how the filter bandwidth varies with C1 and L2.

Select **AC / 3D Windows / Y_Range()**. This produces the following plot:

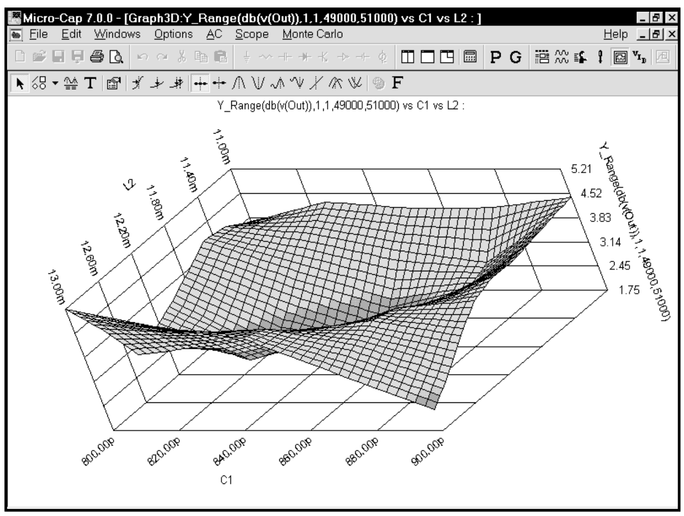


Figure 27-14 The plot of Y_Range vs. L1 and C2

Here we've plotted the Y_Range performance function:

Y_Range(db(v(Out)),1,1,49000,51000)

This performance function measures the variation of the db(V(Out)) function at a Boolean of 1 (always true), first instance, over the X range of 49Khz to 51Khz. This effectively measures the passband ripple of the filter.

Changing the plot orientation

The orientation of the 3D plot can be changed in several ways:

- By dragging the axis endpoints with the mouse.
- By using the keyboard
  - **Q:** Rotates the plot clockwise about the X axis.
  - **A:** Rotates the plot counter clockwise about the X axis.
  - **W:** Rotates the plot clockwise about the Y axis.
  - **S:** Rotates the plot counter clockwise about the Y axis.
  - **E:** Rotates the plot clockwise about the Z axis.
  - **D:** Rotates the plot counter clockwise about the Z axis.
  - **X:** View from perpendicular to the X=0 plane.
  - **Y:** View from perpendicular to the Y=0 plane.
  - **Z:** View from perpendicular to the Z=0 plane.
  - **CTRL + HOME:** Standard view orientation.
  - **C:** Toggles between contour and 3D view.

Scaling in 3D

The choice of scaling is determined entirely by the range of axis variables. There is no user control of scaling.

Chapter 28 Optimizer

What's in this chapter

The optimizer included with MC7 uses a variation of the Powell direction-set method for optimization. It is a robust technique that works well for the kinds of optimization problems found in circuit design.

The chapter is organized as follows:

- How the optimizer works
- The Optimize dialog box
- Optimizing low frequency gain
- Optimizing matching networks
- Curve fitting with the optimizer

How the optimizer works

The optimizer systematically changes user-specified parameters to maximize, minimize, or equate a chosen performance function, while keeping the parameters within prescribed limits and conforming to any specified constraints.

It works in any analysis mode, letting you optimize transient, AC small-signal or DC characteristics. Anything that can be measured in any analysis mode by any performance function can be maximized, minimized, or equated. Equating is the process of matching specified points on a curve. A typical application would be trying to match points on a Bode plot gain curve.

To see how optimization works, load the file circuit OPT1.CIR. It looks like this:

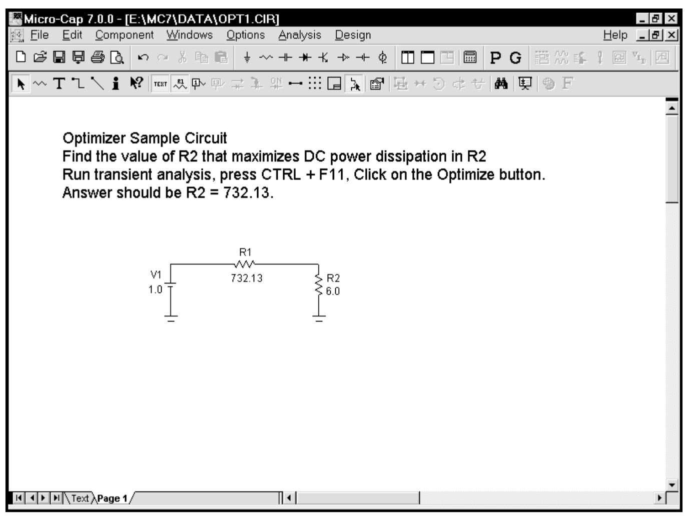


Figure 28-1 The OPT1 circuit

This circuit delivers power to a 6.0 ohm load from a 1.0 volt battery through a 732.13 ohm source resistance. We'll use the optimizer to find the value of R2 that maximizes its power.

The Optimize dialog box

Run transient analysis and select **Transient / Optimize** or press CTRL + F11. This loads the Optimize dialog box, which looks like this.

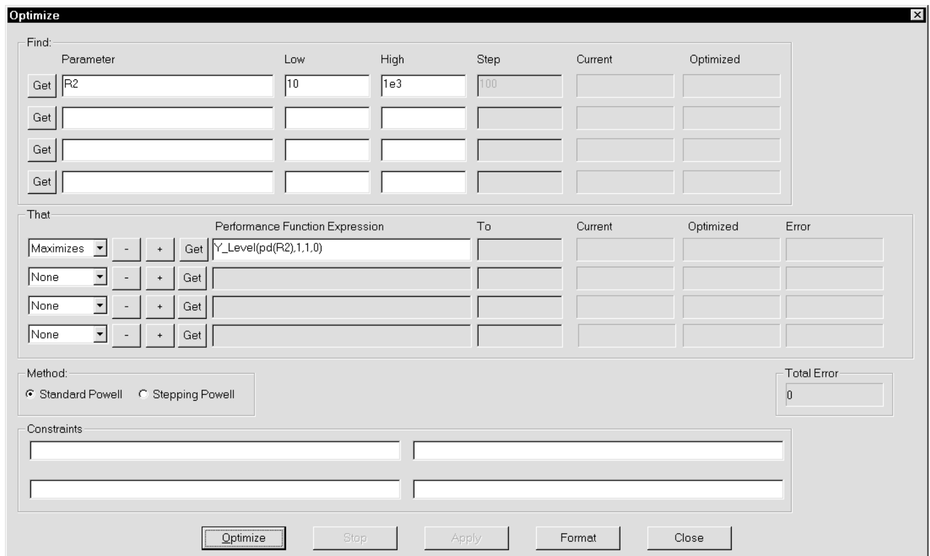


Figure 28-2 The Optimizer dialog box

The syntax of the optimization process is:

Find {Parameter value}
That {Maximizes, minimizes, equates} {Performance function} To {Value}
Using {Standard Powell or Stepping Powell Method}
While {Boolean Constraints}

The user supplies the items in braces {}.

The dialog box offers these options:

Find:

Parameter: This is where you select the parameter to be optimized. Click on the Get button to select a parameter. The choice of parameters is the same as in the Stepping dialog box.

Low: This is the lower limit on the parameter value.

High: This is the upper limit on the parameter value.

Step: This is the amount by which the Stepping Powell method will step the parameter value between local optimizations.

Current: The current value of the parameter is displayed here during the optimization process.

Optimized: The most optimal value of the parameter found so far is displayed here during the optimization process.

That:

Maximize, Minimize, Equate list box: This is where you select one or more optimizing criteria. You can maximize or minimize the chosen performance function. You can also match a curve by using the equate option together with the Y-Level performance function.

-*:* This removes the optimizing criterion for the current row.

+*:* This adds a new optimizing criterion at the end of the list. You can have multiple maximize or minimize criteria, or multiple equate criteria, but you cannot mix maximize / minimize criteria with equate criteria. Each criterion carries equal weight.

Get: This selects the performance function and its parameters.

To: The optimizer will try to match the selected performance function to this value if equate is selected. If Y_Level is used, the optimizer tries to match the Y expression value to the value in this field.

Current: The current value of the performance function is displayed here during the optimization process.

Optimized: The most optimal value of the performance function found so far is displayed here during the optimization process.

Error: This shows a measure of the error for the row criteria.

Method:

Standard Powell: This is the standard Powell direction-set optimizer.

Stepping Powell: This method steps the parameters from *Low* to *High* using steps of *Step*. At each step, the standard Powell method is used to find the local optimum. If the local optimum is better than the current best value, it is retained, and the next step is taken. This is the method to use where you want to get the best of many local optima. It is also the slowest since the optimizer does N standard Powell optimizations, where

N =number of steps for parameter 1 *
number of steps for parameter 2 *
…
number of steps for last parameter

N can be very large so use this method judiciously.

Total Error:
This shows the sum of all criteria errors. In equate optimization, it shows the square root of the sum of the squares of the differences between the target and actual values.

Constraints:
There are four fields for constraints. Each constraint is entered as a Boolean statement. Here are some examples.

PD(R1)<=100m
V(OUT)>=1.2
VCE(Q1)*IC(Q1)<=200m

The buttons at the bottom of the dialog box give these options:

Optimize:
Starts the optimizer.

Stop:
Stops the optimizer.

Apply:
Modifies the circuit by changing its parameters to the optimized values.

Format:
Lets you choose the numeric format of the displayed values.

Close:

Quits the optimizer.

The settings for the OPT1 circuit determine the value of the resistor R2 that maximizes the power dissipated in R2. It finds the value of R2 that maximizes

Y_Level(PD(R2),1,1,0)

PD(R2) is the power dissipated in resistor R2.

Y_Level(PD(R2),1,1,0) is the Y expression value of the curve PD(R2) at the X expression (T) value of 0, which is the DC operating point value.

Click on the Optimize button. After a few seconds the optimizer finishes and presents the optimal value of R2 = 732.13 ohms.

We could have guessed this value from the simplicity of the circuit. Maximum real power is delivered when the load impedance equals the conjugate of the source impedance. In this case maximum power is delivered when R1 and R2 are the same value, 732.13 ohms.

Optimizing low frequency gain

Load the circuit file OPT2. It looks like this:

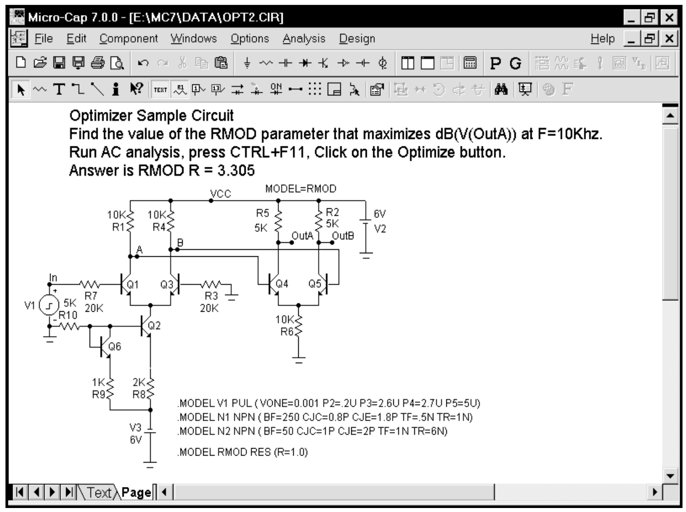


Figure 28-3 The OPT2 circuit

Select AC analysis and press F2. Press F8. Note that the gain at F=10KHz is about 55.6 dB. The standard run looks like this:

Initial gain at 10KHz is about 55.6 dB.

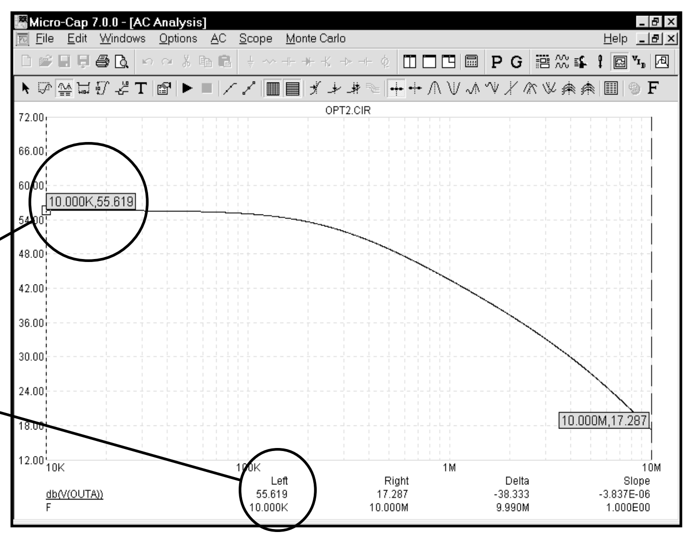


Figure 28-4 AC analysis before optimization

Select **AC/ Optimize** or press CTRL + F11. This loads the Optimize dialog box for AC analysis. It looks like this:

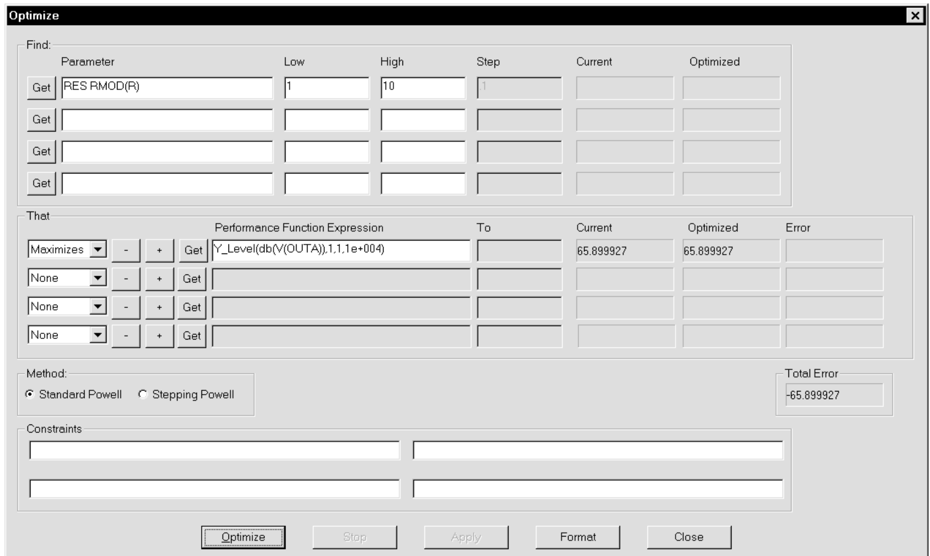


Figure 28-5 The Optimizer dialog box for OPT2

These settings for the OPT2 circuit determine the value of the model parameter R in the resistor model RMOD that maximizes the 10KHz gain of the circuit. RMOD is the model used by the two 5K load resistors R2 and R5. The optimization choices find the following:

The value of RES RMOD(R) that maximizes

Y_Level(db(V(OUTA)),1,1,1e+004)

db(V(OUTA)) is the dB value of the output voltage of the diffamp circuit.

Y_Level(db(V(OUTA)),1,1,1e+004) is the value of the curve db(V(OUTA)) at the X expression (F) value of 1E4, which is the lowest frequency in the AC analysis run.

Click on the Optimize button. After a few seconds the optimizer finds the optimal value of R = 3.305. Since R multiplies the nominal resistance, this means that the value of the load resistors R2 and R5 that maximizes db(V(OUTA)) is:

3.305*5k = 17K

We used Standard Powell here. Most circuit optimizations problems have simple local minima like this and are best served with this method.

Click on the Apply and Close buttons, then press F2. When the run is over, press F8. The Apply and Close buttons copy the optimized values to the circuit and F2 produces a new run with the optimized R model parameter. F8 puts us in Cursor mode so we can readily read the new values. The display should look like this:

Optimized 10kHz gain is about 65.9 dB.

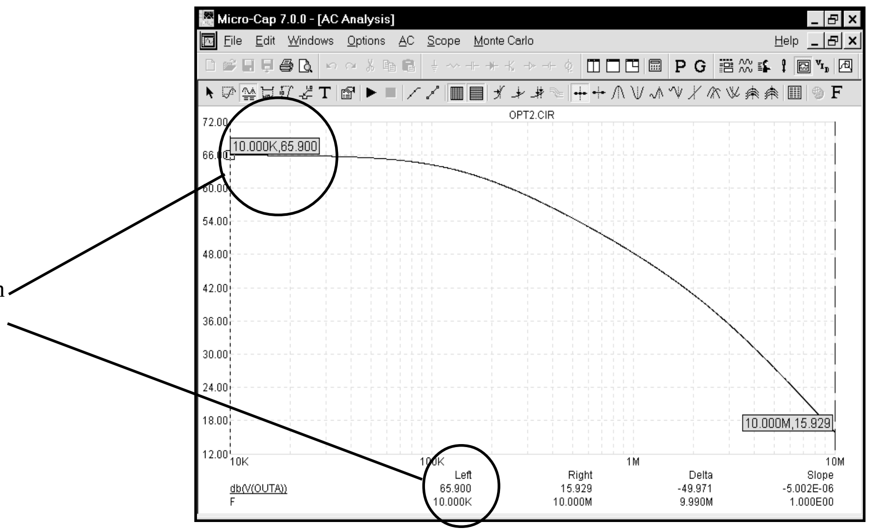


Figure 28-6 AC analysis before optimization

The gain at F=1E4 is now about 65.9 dB.

When the Apply button is clicked the old model name is replaced by a new one with the newly optimized parameter value. The old model statement with the old name is left unchanged in the text area of the circuit. All parts that used the old model name are changed to use the new model name and thus the new optimized parameter value.

Optimizing matching networks

Load the circuit file OPT3. It looks like this:

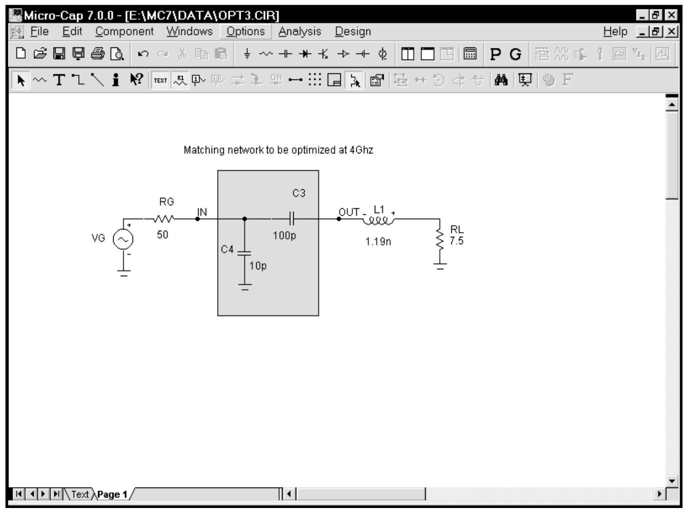


Figure 28-7 The OPT3 circuit

Select AC analysis and press F2. Press F8. Note that the RL power, V(RL)*I(RL) peaks at about F=1.3GHz. The standard run looks like this:

RL power peaks at about 1.3GHz.

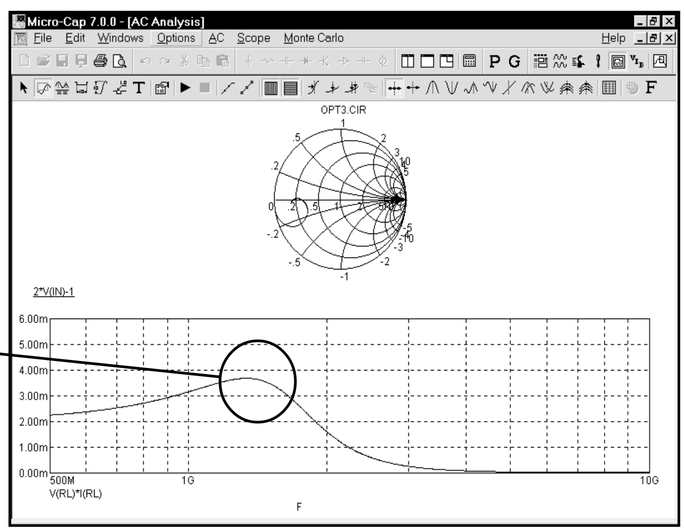


Figure 28-8 AC analysis before optimization

Select **AC/ Optimize** or press CTRL + F11. This loads the Optimize dialog box for AC analysis. It looks like this:

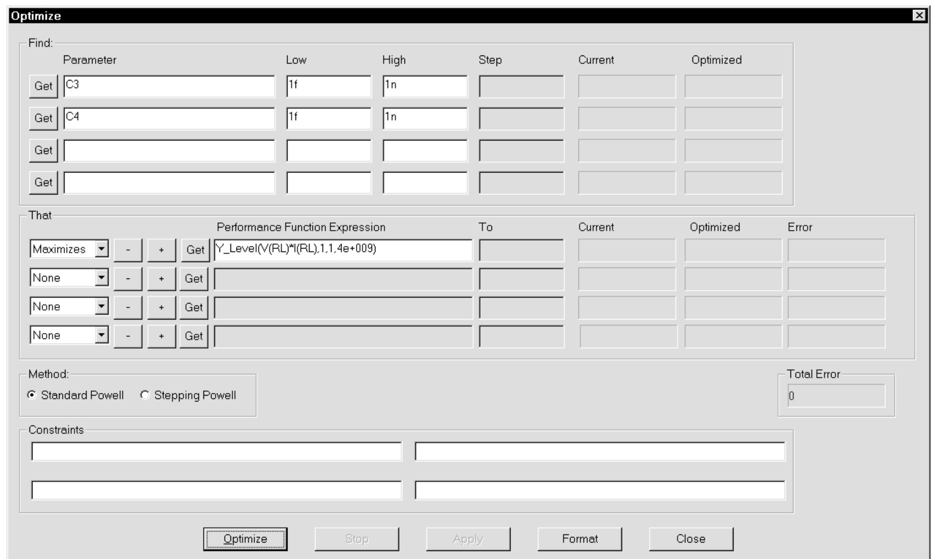


Figure 28-9 The Optimizer dialog box for OPT3

These settings find the values of the C3 and C4 network-matching capacitors that maximize the AC power, V(RL)*I(RL), delivered to the load, RL at 4GHz. To summarize, the optimizer finds the following:

The value of C3 and C4 that maximizes

Y_Level(V(RL)*I(RL),1,1,4e+009)

V(RL)*I(RL) is the AC power delivered to the load RL.

Y_Level(V(RL)*I(RL),1,1,4e+009) is the value of the curve V(RL)*I(RL) at the X expression (F) value of 4GHxz, which is the frequency at which we want the matching network to deliver peak power.

Click on the Optimize button. After a few seconds the optimizer finds the optimal value of C3 = 3.3pF and C4 = 1.894pF. These values deliver about 5mW to the load at 4GHz.

Click on the Apply and Close buttons, then press F2. When the run is over, press F8. The Apply and Close buttons copy the optimized values of C3 and C4 to the circuit and F2 produces a new run with the optimized values. The display should look like this, after locating and tagging the peak.

RL power peaks at 5mw at 4GHz.

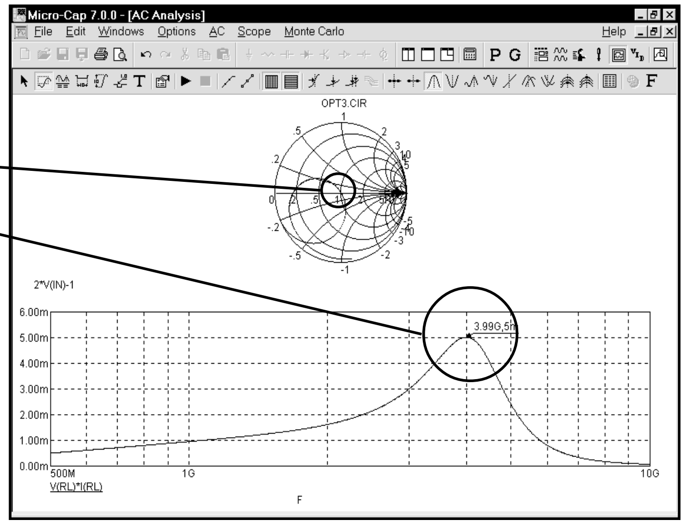


Figure 28-10 AC analysis after optimization

Note that the Smith chart is plotting 2*VIN-1 which, for this circuit, is equivalent to plotting the scattering parameter, S11. Note also that S11 goes through the Smith chart origin (1,0) at 4 GHz, another indication that the matching capacitor network has been optimized to deliver maximum AC power to the load.

Curve fitting with the optimizer

Load the circuit file OPT4. It looks like this:

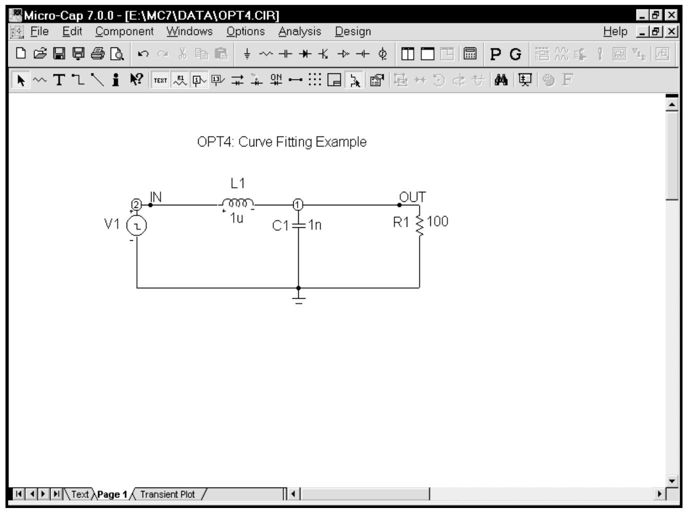


Figure 28-11 The OPT4 circuit

Select AC analysis and press F2. Press F8. The standard run looks like this, after placing the cursors at 2MHz and 10MHz. Note the value of db(V(OUT)) is 1.397 at 2MHz and -9.583 at 10MHz.

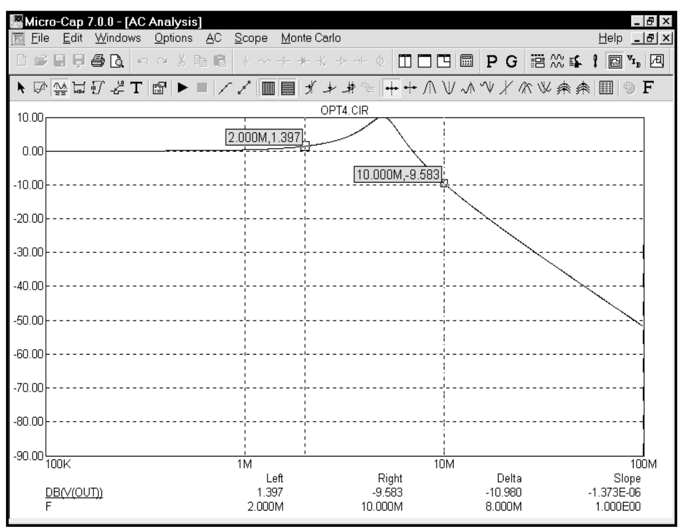


Figure 28-12 AC analysis before optimization

Select **AC/ Optimize** or press CTRL + F11. This loads the Optimize dialog box for AC analysis. It looks like this:

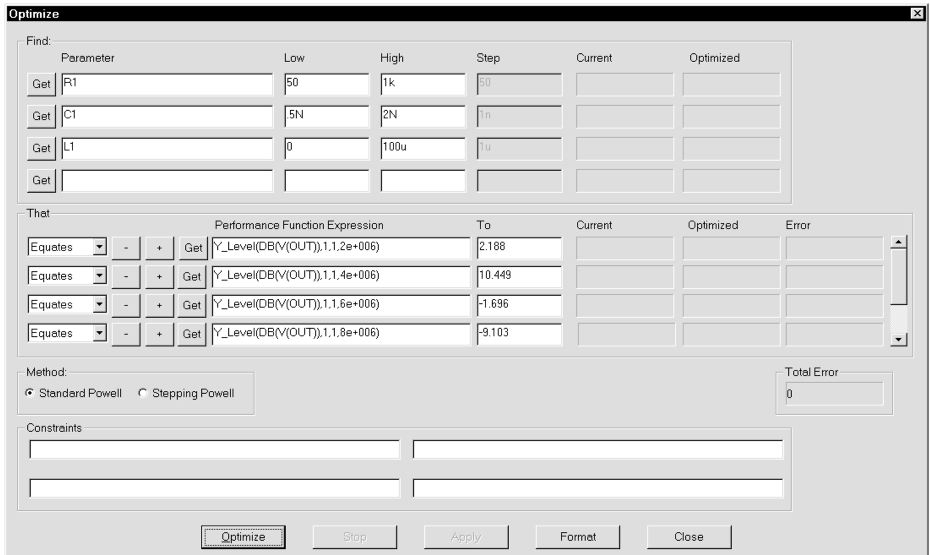


Figure 28-13 The Optimizer dialog box for OPT4

These settings for the OPT4 circuit determine the value of R1, C1, and L1 that equate the six values of DB(V(OUT)) to specified values at different frequencies. To summarize, it finds the following:

The value of R1, C1, and L1 that:

Equates Y_Level(DB(V(OUT)),1,1,2e+006) to 2.188
Equates Y_Level(DB(V(OUT)),1,1,4e+006) to 10.449
Equates Y_Level(DB(V(OUT)),1,1,6e+006) to -1.696
Equates Y_Level(DB(V(OUT)),1,1,8e+006) to -9.103
Equates Y_Level(DB(V(OUT)),1,1,10e+006) to -13.939
Equates Y_Level(DB(V(OUT)),1,1,20e+006) to -27.134

In short, we are trying to match six data points, at frequencies ranging from 2 MHz to 20 MHz on the plot of db(v(out)).

In equate optimization, the optimizer actually minimizes the square root of the sum of the squares of the differences between the target and actual values. The Total Error field always shows this root-mean-square error measure.

Click on the Optimize button. After a few seconds the optimizer finds the optimal values; R1= 255, L1=3u, and C1 = 500pF.

Click on the Apply and Close buttons, then press F2. When the run is over, press F8. The Apply and Close buttons copy the optimized values to the circuit and F2 produces a new run with the optimized values. The display should look like this, after locating and tagging the 2MHz and 10MHz values.

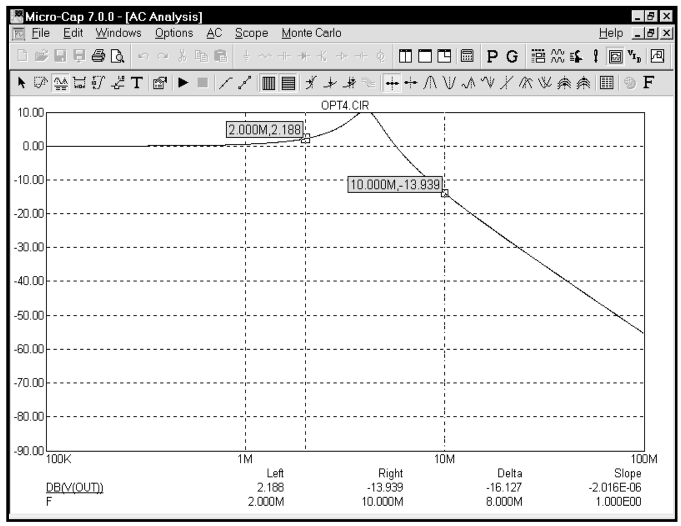


Figure 28-14 The optimized plot

Checking two of the equate values, at 2Mhz and 10MHz, we find they equal the specified target values 2.188 and -13.939 respectively.

The optimizer was able to match these values exactly in this example. That is because the target values were taken originally from a circuit that was identical to the optimized circuit. We know that the target values are realizable because they came from an actual circuit. In cases like this it is sometimes possible for the optimizer to find *scaled* versions of the same circuit, since these have exactly the same circuit response. A scaled circuit is derived from a circuit by multiplying every resistance and inductance value by a scale factor and by dividing every capacitor value by the same factor. There are an infinite number of such scaled circuits. This is not usually a problem, only something to be aware of. You may get many optimal solutions to this kind of problem.

Appendix A File types

Bold files are essential to the proper operation of the program and should not be removed.

Italicized files save user preferences and should not be removed, although the program will work without them.

Underlined files are user-generated and, while optional, may contain important user-created data.

The remaining files are optional and usually can be deleted to conserve space.

*.1, *.2...*.n - Monte Carlo worst case circuit files
*.AMC - AC Monte Carlo statistics
*.ANO - AC numeric output
*.ASA - AC save analysis / probe simulation
***.BIN - Filter designer circuit template file**
*.BMP - Graphics file in BMP format
*.BOM - Bill of Materials
***.CAP - Listing of capacitor values used in the filter designer**
*.CIR - Micro-Cap schematic file
*.CKT - SPICE circuit file
***.CNT - Help contents file**
***.CMP - Component Library file**
**.DAT - User preferences file.*
*.DMC - DC Monte Carlo statistics
*.DNO - DC numeric output
*.DOC - Results file created when MC7 is run under batch mode
*.DSA - DC save analysis / probe simulation
*.EMF - Windows extended metafile (picture file)
*.ERR - PCB error files
***.EXE - MC7 and MODEL program files**
**.FLT - Filter preferences file*
*.GIF - Graphics file in GIF format. MC7 can read but not write GIF files
***.HLP - Help file**
**.INC - User definitions file MCAP.INC.*
***.IND - Listing of inductor values used in the filter designer**
*.INX - Index file
*.JED - JEDEC file

*.JPG - Graphics file in JPG format. MC7 can read but not write JPG files
***.LBR - Micro-Cap binary library file**
***.LIB - SPICE library file**
***.MAC- Micro-Cap macro file**
***.MC7 - Micro-Cap usage statistics file**
***.MC7 - Micro-Cap program file used for the demos or error messages**
***.MDL- Model program data file**
*.OPT- Results file created when MODEL is run under batch mode
*.NET - Orcad, Protel, PADS, or Accel netlist file
***.PKG - Package Library file**
***.RES - Listing of resistor values used in the filter designer**
***.SAN - Minimum, maximum, and default limits for model parameters**
*.SEN - Sensitivity analysis output file
***.SHP - Shape Library file**
*.STM - Waveform file for digital file stimulus component
*.SVV - State Variables editor text output file
*.TMC - Transient Monte Carlo statistics
*.TNO - Transient numeric output
*.TOP - Binary output file used in State Variables read operation
*.TSA - Transient save analysis / probe simulation
*.TXT - Text file created by the Component editor Parts List command
*.USR - User waveform file for plotting or the User Source component
*.WMF - Windows meta file (picture file)

Appendix B

Model library files

AD.LIB	Demo file for part import illustration
ADV_LIN.LIB	Advanced Linear ICs
AMP.LIB	Amp connectors
ANALOG.LIB	Analog Devices ICs
APEX.LIB	Apex Microtechnology ICs
APEXPWM.LIB	Apex PWM Models
BURRBN.LIB	Burr Brown ICs
CMOS.LIB	CD4000 series digital ICs
COMLIN.LIB	Comlinear ICs
CUR_REGS.LIB	Current regulator diodes
DIG000.LIB	74xx digital ICs
DIG150.LIB	74xx digital ICs
DIG167.LIB	74xx digital ICs
DIG195.LIB	74xx digital ICs
DIG250.LIB	74xx digital ICs
DIG381.LIB	74xx digital ICs
DIG604.LIB	74xx digital ICs
DIG652.LIB	74xx digital ICs
DIG874.LIB	74xx digital ICs
DIGIO.LIB	Digital I/O interfaces
DIGPLD.LIB	Programmable logic devices
DIODE.LBR	Diodes
ECL.LIB	ECL digital ICs
EDIODE.LBR	European diodes
ELANTEC.LIB	Elantec ICs
EUROPE.LBR	European transistors
FAIRCH.LIB	Fairchild discrete components
FWBELL.LIB	F.W. Bell hall generators
GENSEMI.LIB	General Semiconductor diodes
HARHIP.LIB	Harris Semiconductor bridge drivers
HARPRMOS.LIB	Harris Semiconductor power MOSFETs
HARRHMOS.LIB	Harris radiation hardened power MOSFETs
HARRIS.LIB	Harris Semiconductor ICs
HPDIODE.LIB	Hewlett Packard diodes
HPMOS.LBR	Hitachi power MOSFETs
INFINEON.LIB	Infineon discrete components
INTERSIL.LIB	Intersil MOSFETs
IRF.LIB	International Rectifier transistors
IRPMOS.LBR	International Rectifier MOSFETs

JAPAN.LBR	Japanese transistors
JDIODE.LBR	Japanese diodes
JFET.LBR	JFETs
LASERIES.LIB	Harris LA series varistors
LINEAR.LIB	Linear Technology ICs
LTOPAMP.LBR	Linear Technology opamps
M_IC.LIB	Motorola integrated circuits
M_IGBT.LIB	Motorola IGBT devices
M_OPAMP.LIB	Motorola OPAMPs
M_OPTO.LIB	Motorola optoelectronics
M_POWBJT.LIB	Motorola bipolar power devices
M_RECT.LIB	Motorola rectifiers
M_RFDEV.LIB	Motorola RF transistors
M_SMALL.LIB	Motorola small signal devices
M_TMOS.LIB	Motorola TMOS
M_ZENER.LIB	Motorola zeners
MAXIM.LIB	Maxim ICs
MICROSEM.LBR	Microsemi diodes
MPBJT.LBR	Motorola power BJTs
MPMOS.LBR	Motorola power MOSFETs
MSBJT.LBR	Motorola small-signal BJTs
MSENSOR.LIB	Motorola sensors
MZENER.LBR	Motorola zener diodes
NATION.LIB	National Semiconductor ICs
NOM.LIB	Master list of all library files
NP_DCDC.LIB	Newport Components DC-DC Converters
NSOPAMP.LBR	National Semiconductor opamps
NTC.LIB	Siemens thermistors
ONSEMI.LIB	ON Semiconductor Discrete Components
OP27.LIB	Used in one of the auto demos.
PASSIVE.LIB	Resistors
PCORE.LBR	Philips ferroxcube magnetic cores
PH_BJT.LIB	Philips Transistors
PH_DIODE.LIB	Philips Diodes
PH_FET.LIB	Philips JFETS
PH_MOS.LIB	Philips MOSFETS
PH_RFDEV.LIB	Philips RF transistors
PH_SHOT.LIB	Philips schottky diodes
PMOPAMP.LBR	Precision Monolithics opamps
POLYFET.LIB	Polyfet RF MOSFETs
PROFMOS.LBR	Profusion MOSFETs

RECTIFIE.LIB	Motorola rectifiers
SHINDEN.LIB	Shendengen diodes
SIMID.LIB	Siemens inductors
SIOV.LIB	Siemens varistors
SMPS_CB.LIB	Switched-mode power supply models by C. Basso
STMICRO.LIB	Standard Microsystems linear opamps
TCORE.LBR	Tokin magnetic cores
THY_LIB.LIB	SCRs and Triacs
TI.LIB	Texas Instruments ICs
TIOPAMP.LBR	Texas Instruments opamps
TRANS.LIB	Transmission lines
TUBE.LIB	Vacuum tube models by Duncan Munro
UTILITY.LBR	Generic models, pulse and sine sources
VARACTOR.LBR	Varactor diodes
VISHAY.LIB	Siliconix MOSFETS
XFMR.LIB	Transformers
XTAL.LIB	KDS crystals
ZASERIES.LIB	Harris ZA series varistors
ZETEX.LIB	Zetex diodes and transistors

Appendix C Sample circuit files

Schematics

283	Use of digital primitives to model a 283 logic unit
381	Use of digital primitives to model a 381 logic unit
3D1	Use of 3D plots
3D2	Use of 3D plots
555ASTAB	Use of the 555 macro in an astable application
555MONO	Use of the 555 macro in a monostable application
A_BOOST_CM_OL	Boost current mode averaged model open loop plot
A_BOOST_CM_ZOUT	Boost current mode averaged model Zout plot
A_BOOST_VM	Boost voltage mode averaged model open loop plot
A_BUCK_CM	Buck current mode averaged model open loop plot
A_BUCK_VM	Buck voltage mode averaged model open loop plot
A_BUCKBOOST	Buckboost current mode averaged model open loop plot
A_FLYBACK	Flyback voltage mode averaged model open loop plot
A_FORWARD	Forward voltage mode averaged model open loop plot
A_NCP	NCP1200 Converter
A_RESO_DC	Resonant converter DC analysis
A_RESO_OL	Resonant converter averaged model open loop plot
A_SEPIC	Single Ended Primary Inductance Converter
AD16	Use of the AtoD and DtoA elements
ANIM	Use of the animation components
ANIM3	Use of the animation components
BAX	Steps a resistor to model a pot element
BPFILT	Analysis of a bandpass filter
BUTTERN	Use of a Laplace source to represent a Butterworth filter
CARLO	Monte Carlo routines in transient and AC analysis
CARLO2	Monte Carlo routines in DC analysis
CARLO4	Monte Carlo routines in transient and AC analysis
CHOKE	Analysis of a diode choke circuit
CMOS	MOSFETs in an inverter configuration
COLPITTS	Analysis of a colpitts oscillator
CONVERTER3	Three-phase converter with zero-crossing detectors
CORE	Use of the core model and plotting a BH curve
CORE3	Use of the nonlinear core model with multiple inductors
COUNTER	Analysis of a binary counter
COUNTER2	Analysis of a BCD counter
CROSSOVR	Analysis of a passive 1kHz cross-over network
CURVES	BJT IV curves

DECODER	Use of a digital subcircuit as a decoder
DIFFAMP	Analysis of a differential amplifier
DIG_POWER	How to change the digital power supplies
DIRA	Use of the operators d, avg, sum, and rms
ECLGATE	Analysis of an analog equivalent ECL gate
F1	Use of the VCO macro
F2	Use of a nonlinear function source
F3	Use of a nonlinear function source
F4	Use of the Triode macro
FFT1	Use of DSP and complex operators
FFT3	Use of cross-correlation and auto-correlation operators
FFT4	Use of the IFT operator
FFT5	Use of the auto-correlation operator
FFT7	Use of the DSP dialog box to eliminate startup transients
FILTER	Analysis of a Chebyshev filter and use of the Noise macro
FSK2	Use of the FSK modulator macro
FSTIM8	Use of the file stimulus component
GASFET	Use of the GaAsFET component
GILBERT	Analysis of a Gilbert multiplier
GUMMEL	Use of the Gummel-Poon SPICE BJT model
GYRTEST	Use of the gyrator macro
IVBJT	Use of DC analysis to plot the IV curves of a BJT
L1	Use of a Laplace source to model a passive network
L2	Use of Laplace sources to model transmission lines
L3	Use of a Laplace source to model a Butterworth filter
LM117REG	Using the LM117 model
LTRA3	Use of the lossy transmission line
MIXED	Analysis of a mixed-mode circuit
MIXED1	Analysis of a mixed-mode circuit
MIXED4	Analysis of a mixed-mode circuit
MODELRLC	Use of temperature stepping
MOSCAPS	Plotting of MOSFET capacitance curves
MOSDIFF	Analysis of a MOSFET differential amplifier
NOISEBJT	Plotting of input and output noise
NYQUIST	Plotting of a Nyquist graph
O7	Analysis of a mixed-mode circuit
OPAMP1	Use of the three levels of opamps
OPT1	Using the Optimizer to maximize power transfer
OPT2	Using the Optimizer to maximize low frequency gain
OPT3	Using the Optimizer to design matching networks
OPT4	Using the Optimizer in curve fitting

OSC1	Use of the Schmitt macro in an oscillator
P1	Use of the Laplace table source for a RC network
PERF1	Demonstrates the use of performance plots
PERF2	Demonstrates the use of performance plots
PLA2	Use of a PLA subcircuit as an equality comparator
PLA3	Use of the PLA digital primitive
POTDEMO	Use of the pot macro
PRINT	Use of the print preview for the schematic
PRLC	Analysis of a simple passive network
PSK2	Use of the PSK modulator macro
RCA3040	Analysis of a RCA3040 component
RELAY	Using the relay models
RISE	Use of Monte Carlo routines for rise times
S_2FLY_CM	Two-Switch Flyback Converter
S_2FOR_CM	Two-Switch Forward Converter
S_BOOST_CM	Boost Current Mode Converter
S_BOOST_VM	Boost Voltage Mode Converter
S_BUCK_CM	Buck Current Mode Converter
S_BUCK_SYN	Synchronous Buck Voltage Mode Converter
S_BUCK_SYN2	Synchronous Buck Current Mode Converter
S_BUCK_VM	Buck Voltage Mode Converter Converter
S_BUCKBOOST_CM	Buck-Boost Current Mode Converter
S_BUCKBOOST_VM	Buck-Boost Voltage Mode Converter
S_FLYBACK_CM	Flyback Current Mode Converter
S_FLYBACK_VM	Flyback Voltage Mode Converter
S_FORWARD_CM	Forward Current Mode Converter
S_FORWARD_VM	Forward Voltage Mode Converter
S_FULL_CM	Full Bridge Current Mode Converter
S_FULL_VM	Full Bridge Voltage Mode Converter
S_FULL_XFMR	Full Bridge with XFMR Current Mode Converter
S_HALF_CM	Half Bridge Current Mode Converter
S_HALF_VM	Half Bridge Voltage Mode Converter
S_HALF_XFMR	Half Bridge with XFMR Current Mode Converter
S_NCP	NCP1200 Converter
S_PUSH_CM	Push-Pull Current Mode Converter
S_PUSH_VM	Push-Pull Voltage Mode Converter
SH2	Use of the sample and hold component.
SMITH	Use of the Smith chart
SPARK	Use of the spark-gap macro
STIM_DEMO	Use of the digital stimulus generators
STIMSAMP	Use of the digital stimulus generators

STIMTST2	Use of a Stim generator in counting from 0 to F
STIMTST3	Use of the INCR command in a Stim generator
STIMTST4	Use of the random characters in a Stim generator
SUBCKT	Use of an analog subcircuit
SUBCKT1	Adding subcircuits to the library
SWITCH	Use of the three types of the Switch component
SYSTEM1	Analysis of a mechanical system
SYSTEM2	Use of behavioral modeling components
T1	Use of nonlinear table sources
THY1	Use of the Put, Triac, and SCR macros
THY2	Analysis of a SCR phase control
TL1	Use of transmission line and plotting line variables
TL2	AC simulation of a transmission line
TL3	Plotting the input small signal impedance
TRANS	Use of the three methods of implementing a transformer
TTLINV	Use of mixed mode analysis
TUBE_AMP	Vacuum tube amplifier
TUBE6L6	Vacuum tube circuit
UA709	Analysis of a UA709 opamp
UA723_REG	Using the UA723 model
UA741	Analysis of a UA741 opamp
USER	Use of the User source
USER2	Use of multiple User sources
XTAL1	Use of the crystal macro
ZDOMAIN	Use of the Z transform source

SPICE files

ASTABLE.CKT	Analysis of a SPICE circuit
CHOKE.CKT	SPICE equivalent of CHOKE
ECLGATE.CKT	SPICE equivalent of ECLGATE
PLA1.CKT	Use of a PLA subcircuit in a SPICE file
PLA2.CKT	The PLA subcircuit that is used in PLA2
RCA3040.CKT	SPICE equivalent of RCA3040
RTLINV.CKT	Analysis of a SPICE equivalent inverter
SCHMITT.CKT	Analysis of a SPICE Schmitt trigger
TTLINV.CKT	SPICE analysis of a TTL inverter
UA709.CKT	SPICE equivalent of UA709
UA741.CKT	SPICE equivalent of UA741

Appendix D Accelerator keys

General

Alt+Backspace	Undo
Alt+F1	SPICE Help
Alt+F4	Exit Program
Alt+Z	Statistics
Ctrl+Num+	Zoom-In
Ctrl+Num-	Zoom-Out
Ctrl+0	Main Tool Bar
Ctrl+1-9	Component Palette 1-9
Ctrl+A	Select All
Ctrl+C	Copy
Ctrl+Delete	Clear Cut Wire
Ctrl+F	Find
Ctrl+F4	Close Open Window
Ctrl+F6	Next Window
Ctrl+Insert	Copy
Ctrl+N	New File
Ctrl+O	Open File
Ctrl+P	Print
Ctrl+S	Save File
Ctrl+V	Paste
Shift+Insert	Paste
Ctrl+X	Cut
Ctrl+Y	Redo
Ctrl+Z	Undo
Ctrl+Shift+F	Find Component
Ctrl+Shift+G	Global Settings
Ctrl+Shift+P	Preferences
Ctrl+Shift+W	Print Active Window
Del	Clear
F1	Help
F10	Properties
Shift+F4	Tile Vertical
Shift+F5	Cascade

Schematic / Text Area

Delete	Delete selected objects
Alt+1	Transient Analysis
Alt+2	AC Analysis
Alt+5	Transfer Function
Alt+6	Sensitivity
Ctrl+Alt+1	Probe Transient
Ctrl+Alt+2	Probe AC
Ctrl+Alt+3	Probe DC
Ctrl+D	Component Mode
Ctrl+E	Select Mode
Ctrl+G	Toggle Drawing/Text
Ctrl+H	Help Mode
Ctrl+Home	Upper Left Schematic
Ctrl+I	Info Mode
Ctrl+M	Make Macro
Ctrl+R	Rotate Box
Ctrl+W	Wire Mode
Ctrl+Shift+R	Toggle Rubberbanding
Ctrl+Left	Pan Left
Ctrl+Right	Pan Right
Ctrl+Up	Pan Up
Ctrl+Down	Pan Down
PgUp	Pan Up One Page
PgDn	Pan Down One Page
Ctrl+PgUp	Pan Left One Page
Ctrl+PgDn	Pan Right One Page
Ctrl + }	Match Parenthesis
F3	Repeat Last Find

Text Area / Text File

Alt+F1	SPICE Help
Ctrl+D	Delete Line
Ctrl+L	Set to Lowercase
Ctrl+U	Set to Uppercase

Schematic / Analysis

Ctrl+T	Text Mode
Ctrl+W	Wire Mode
Spacebar	Toggle Select / Mode

Analysis

Ctrl+N	Normalize

F2	Run Analysis
F3	Exit Analysis
F4	Analysis Window
F5	Numeric Output
F6	Auto Scale
F7	Scale Mode
F8	Cursor Mode
F9	Limits
F11	Stepping
F12	State Variables Editor
Down Arrow	Left Cursor Down a Branch
Left Arrow	Left Cursor to Left
Right Arrow	Left Cursor to Right
Up Arrow	Left Cursor Up a Branch
Shift+Down Arrow	Right Cursor Down a Branch
Shift+Left Arrow	Right Cursor to Left
Shift+Right Arrow	Right Cursor to Right
Shift+Up Arrow	Right Cursor Up a Branch
Ctrl+Down Arrow	Pan Plot Down
Shift+Ctrl+X	Go To X
Shift+Ctrl+Y	Go To Y
Ctrl+L	Tag Left Cursor
Ctrl+R	Tag Right Cursor
Ctrl+Left Arrow	Pan Plot Left
Ctrl+Right Arrow	Pan Plot Right
Ctrl+Up Arrow	Pan Plot Up
Ctrl+W	Watch Mode
Ctrl+Shift+C	Toggle Cursor Tracker
Alt+Home	Place Cursor on Top Branch
Alt+End	Place Cursor on Bottom Branch

3D Plot

A	Counter-Clockwise X
C	Toggle Contour / 3D
D	Counter-Clockwise Z
E	Clockwise Z
Q	Clockwise X
S	Counter-Clockwise Y
W	Clockwise Y
X	Perpendicular to X=0
Y	Perpendicular to Y=0
Z	Perpendicular to Z=0

Index

Symbols

A

B

C

D

E

F

N

O

P

Q

R

S

T

U

V

W

X

Y

Z