1.0) Introduction

This file describes the enhancements and changes included in the 4.04 release of PSpice.

For information regarding installation, operation, and technical details about the package, you may purchase the Circuit Analysis User's Guide for $50.00.

2.0) Updates and Corrections to the User's Guide

All manual modifications for this version have been marked by shading those items. This makes it easy to locate the new MicroSim product features, and calls attention to the items which have been changed.

PSpice (all platforms)

(1) For the BSIM MOS model (level=4): if the model parameter XPART is negative, then the Meyer capacitance model is used. This avoids the potential for negative capacitance values from BSIM's charge-based capacitance model, however care must be exercised regarding charge conservation with the Meyer model.

(2) Circuit files using models with parameters from the PSpice libraries, must define those parameters. If those parameters are not defined within the circuit file, PSpice will use zeros and erroneous results will occur. This inconvenience will be corrected in our next version.

Probe (all platforms)

The X_axis menu now has a new item named "Unrestricted_data", which resets the restriction on the X_axis data set by "Restrict_data".

Probe (VAX only)

The VTxxx keypad <Pg Up> and <Pg Dn> keys are defined incorrectly in the User's Guide. They should be as follows:

```
⁄ƒƒƒƒƒƒƒ¬ƒƒƒƒƒƒƒ¬ƒƒƒƒƒƒƒ¬ƒƒƒƒƒƒƒø
≥PF1    ≥PF2    ≥PF3    ≥PF4    ≥
≥       ≥       ≥       ≥       ≥
≥       ≥       ≥       ≥       ≥
√ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ¥
≥7      ≥8      ≥9      ≥-      ≥
≥       ≥       ≥       ≥       ≥
≥ <Home> ≥  <Up>  ≥ <Pg Up>≥       ≥
√ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ¥
≥4      ≥5      ≥6      ≥,      ≥
≥       ≥       ≥       ≥       ≥
≥ <Left> ≥       ≥ <Right>≥       ≥
√ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ¥
≥1      ≥2      ≥3      ≥ Enter  ≥
≥       ≥       ≥       ≥       ≥
```

≥ <End> ≥ <Down> ≥ <Pg Dn>≥ ≥
√ƒƒƒƒƒƒƒ¡ƒƒƒƒƒƒƒ≈ƒƒƒƒƒƒƒ¥ ≥
≥0 ≥. ≥ ≥
≥ ≥ ≥ ≥
≥ ≥ <Del> ≥ ≥
¿ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ¡ƒƒƒƒƒƒƒ¡ƒƒƒƒƒƒƒŸ

Device Equations (OS/2 only)

The MicroSoft linker v5.03 or later is required. Version 5.01 of the linker, which comes with the MicroSoft C compiler v5.1, has a bug when used with PSpice OS/2 v4.04 Device Equations. You willbe able to link the program but when you go to run it, you will see the error:

SYS1059: The system cannot execute the specified program

If you have v5.01, call MicroSoft at (206) 454-2030 and ask for a later version.

Model Library (all platforms)

The library file "lin_tech.lib" from Linear Technology, Inc. contains 14 models not listed in the July 1990 library listing:

LT118A/LT	LT1002A/LT	LT1014A/LT
LTC1052CS/LT	LTC1053/LT	LTC7652/LT
OP-15E/LT	OP-15F/LT	OP-15G/LT
OP-16E/LT	OP-16F/LT	OP-16G/LT
OP-215E/LT	OP-215G/LT	

Also, note that the SCR and triac models require the Analog Behavioral Modeling option for their use.

DOS/16M Version (ONLY)

The notes in section 2.1.2.5.5 (pages 21-22) about virtual memory are correct, however the meaning of the environment variable "DOS16VM" has changed. The value of this variable now indicates the total amount of virtual memory (that is, the amount of memory you want to have for the application). If the size indicated by DOS16VM is MORE than the available physical memory, a swap file is created to take up the difference. If is size indicated by DOS16VM is LESS than the available physical memory, no swap file is created and the application is limited to physical memory (that is, DOS16VM's value is ignored).

3.0) Program modifications since version 4.03.

PSPICE

1) The Macintosh can now specify search paths for libraries and can save displays in MACPAINT format files.

2) The following PSpice options are now supported on HP/Apollo systems:

-Wi	start iconic
-Wp <x> <y>	frame position
-WP <x> <y>	icon position

3) Expressions used to define global parameters can now use previously defined global parameters. For example:

```
.PARAM pi = 3.14159265, twopi = {2*pi}
```

4) Library additions: The voltage regulator section of LINEAR.LIB now has 73 models for fixed positive and fixed negative regulators, 3-terminal voltage regulators, and the LM723 precision voltage regulator.

A new library file, HARRIS.LIB, contains seven models from Harris Semiconductor. These include the wide-bandwidth HA-2539/40, the fast-settling HA-5190, and the low noise HA-5102/12 and HA-5104/14.

A new library file, THYRISTR.LIB, contains 100 SCR models. These models use/require Analog Behavioral Modeling.

5) For DOS, DOS/16M, and OS/2: The following options are supported:

-ps = COMn:	Test only the specified com port for the security plug.
-ps = xxxx	Test only the com port at physical address xxxx (hex) for the security plug.

6) For DOS/16M versions of PSpice, Probe, and the Control Shell: These applications have virtual memory capability, which means they can access more memory than is actually installed in your computer. If the entire program, and its data, doesn't fit in RAM, part of it is kept on disk in a swap file. When information is needed from the swap file, it is loaded from the disk into RAM. To create more room for it, another part of the program, or its data, may be swapped from RAM to disk.

These programs will look for two environment variables, TMP and DOS16VM, which will define where the swap file is located and how large the swap file may be in kilobytes. For example, the DOS commands

```
SET TMP=F:\
SET DOS16VM=4000
```

will place the swap file on the F: drive, and allow the swap file to be 4000KB (about 4MB). If TMP is not set, the swap file will be placed in the current directory. If DOS16VM is not set, the swap file will be zero in size. We suggest you set these environment variables in your AUTOEXEC.BAT file.

There must be room for the swap file when the application is started, and the swap file will be deleted when the application finishes. Note: it is counterproductive to set aside extended memory as RAM-disk for the swap file... you will be better off

having the memory directly available to the program.

With virtual memory, only the sections of program code and data that are active need be in RAM. For PSpice, this frees roughly 500KB of RAM for circuit data, since program sections such as read-in and checking are not needed during the main simulation process. For Probe, you can set-up the swap file to view extremely large (megapoint) waveforms. For Probe "snoop" during simulation, Probe may actually swap PSpice out of RAM. When Probe finishes, PSpice will swap itself back into RAM. For all applications, start-up will be a bit faster as only the required code is loaded, instead of the entire program.

Note about large circuit simulations: While circuits that exceed the amount of physical RAM may be attempted, these simulations will be very slow. This is also true for other virtual memory environments, such as OS/2, UNIX, and VAX/VMS. PSpice needs access to nearly all of its data for each iteration, and the constant swap activity will slow the simulation. For large simulations, we suggest 1MB of RAM for every 400 active devices.

PROBE

1) The 8514 display now is available for DOS/16M and OS/2 operating systems.

2) The DEC LN03 and HP PaintJet printers are now supported.

3) A mouse can now be used on all platforms to make menu selections and to perform other miscellaneous functions in addition to using the keyboard.

For example: add trace

To display the list of variable names, press either the <F4> key, or press the right mouse button (placement of cursor is not important).

The list of variable names, voltages and currents, is displayed on the upper portion of the screen. If you press the <F4> key a second time, a menu is displayed allowing you to increase or decrease the number of nodes the user can look at. You may show alias names, show internal subcircuit nodes, remove existing voltages, or remove existing currents. The alias names when listed are indented and represent the identical node as the unindented variable name listed above it.

The mouse can now be used to select the desired variable items. When the variable(s) have been selected, press the left mouse button with the cursor anywhere in the bottom two rows of the screen, and the selected traces will be displayed.

The selected variable items can also be edited, using the keyboard, to generate expressions. If a variable item is selected while the cursor is positioned within an expression, the variable item will be placed at that

position.

The mouse can also be used to select a digital trace. Clicking the left mouse button on the digital trace name moves the high-light to that name. Clicking on the area just above or below the digital trace names will scroll the digital traces (if there are traces "off screen" to scroll. Then, clicking in the bottom two lines of the screen will cause the hightlighted trace name to be the selected trace.

4) A zoom feature as been added to Probe. This provides a quick and simple, visual way to change the x and y ranges (that is to zoom the display in or out).

Previously this was done by using the Set_range or Auto-range functions in two different menus. The Set-range function requires you to know and to enter the new axis values.

The new function provides the ability to zoom into a region you specify "visually", either with a mouse or by moving a cursor with arrow keys.

"Zoom" has been added to the main menu. When selected, the Zoom menu has the options of

Exit Specify_region X_zoom_in Y_zoom_in Zoom_out Auto_range

When Zoom is selected, a small cross-hair will appear in the center of the active plot. It can be moved to any region of any plot, using the arrow keys or by clicking the left mouse button, while not moving the mouse.

Also, while not in a menu selection, pressing the left mouse button and moving the mouse will do a "Specify_region" zoom.

Specify_region
: pressing the spacebar or holding down the left mouse button establishes one corner of the region (to be zoomed to). Using the arrow keys, or moving the mouse, draws a "rubber banding" box. Pressing the spacebar again or releasing the left mouse button, causes the display to be redrawn to new axis coordinates. Both corners of the region must be in the same plot. If the rubber banding box is outside the plot, the "zoom" will be clipped to the edge of the plot.

X_zoom_in
: zooms in by a factor of 2, in the X direction, around the small cross-hair cursor.

Y_zoom_in
zooms in by a factor of 2, in the Y direction, around the small cross-hair cursor.

Zoom_out
zooms out by a factor of 2, in both the X and Y directions, around the small cross-hair cursor. Does not exceed data "auto range" limits. If auto range limits are exceeded in some direction, then only the direction in which they are not exceeded will zoom out by the full factor of 2.

Auto_range
causes both the X and Y axis to auto range to cover all data values of currently displayed traces on the currently selected plot (the plot that the small cross-hair cursor is in).

5) A "Real-Time" waveform viewer (auto_update) is now available under OS/2. This feature allows you to view output waveforms while your simulation is running.

Probe must be run with the /A[sec] (or -A[sec]) flag, where sec is the time interval (in seconds) between auto-updates. If not specified, sec defaults to 30.

In the analog and digital plot menu, there is a new item named "Update", used to activate auto_update. When "Update" is selected, the traces will be updated at the time interval specified on initialization (as long as the displayed traces are of the on-going analysis). The update time interval may be altered by selecting "Change_interval" or Auto_update may be suspended by selecting "Exit" fom the Update menu.

6) In Probe, the expression handling for AC analysis has been changed to do complex arithmetic. In version 4.03 and earlier, output variables always produced real values (e.g., VP(4), VM(16,7), IG(CLOAD)). An output variable without a suffix defaulted to an "M" suffix (magnitude). So, V(4)+V(5) meant the sum of the magnitudes of the voltages at nodes 4 and 5.

In 4.04 this was changed so that output variables default to having complex values for AC analysis. A magnitude function is applied to the result of the entire expression. So, V(4)+V(5) means the magnitude of the sum of the (complex) voltages at nodes 4 and 5. Note that this means that V(4)-V(5) is the same as V(4,5).

"M" (magnitude), "P" (phase), "R" (real part), "IMG" (imaginary part), and "G" (group delay) were added to the function list. So, P(V(4)+V(5)) is the phase of the sum of the (complex) voltages at nodes 4 and 5. Each of these functions produces a real value: its imaginary part is 0.

The handling of expressions for DC and transient analysis is unchanged.

7) For SUN systems: In Probe, Parts, and StmEd the colors used for the foreground, background, and traces on Sun systems can be changed. To do this, bring up the frame menu by clicking the right mouse button on the title bar. Select the "Props" entry. A window will appear with buttons labeled "Save" and "Close" and a total of eight rows of sliders. Each row contains an individual slider for the Red, Green, and Blue component of a color selection.

The first two rows let you set up the foreground (axes, text) and background (empty window area). The other six rows are for traces 1 through 6. (The colors recycle after the available six have been used.) A color component is changed by dragging in the marked area of the slider. The new color will be displayed when you release the mouse button.

To save a color setup, click on the "Save" button. (Changes are not saved automatically.) The color map will be written to the file pspice.col in the current directory. The default setup is read from the file pspice.col in the current directory or anywhere on the PATH when the program starts.

8) The Cursor Menu now contains a "Hard Copy" option which enables you to print the cursors and their values as shown on the display.

9) The default device file is now named PSPICE.DEV, rather than PROBE.DEV. The device file is only shipped with the SUN and HP/Apollo platforms, and will have to be created for the other platforms. A program called SETUPDEV is now included with the PSpice package which allows you to create or update the display and hard copy entries in your device file.

PARTS

1) The mouse function has been added but is not fully tested.

2) The bipolar transistor has been updated to handle power devices.

3) Parts now has an optimizer which gives more accurate model parameters.

4) For SUN systems: In Probe, Parts, and StmEd the colors used for the foreground, background, and the traces on Sun systems can be changed. For details, see the "Probe" section of this document.

DIGITAL SIMULATION

1) Programmable Logic Arrays

The PSpice digital simulator now has primitive logic devices which are programmable logic gate arrays. The logic array is made up of a variable number of inputs, which form

columns, and a variable number of outputs, which form rows. Each output (row) is driven by one logic gate. The "program" for the device determines which of the inputs (columns) are connected to each gate. All of the gates in the array are the same type (AND, OR, NAND, NOR, etc.). Commercially available IC's (PAL's, GAL's, PEEL's, and so on) may have buffers, registers, more than one array of gates, and so on, all on the same part. To model these parts we place one or more Programmable Logic Array devices along with the necessary flipflops, gates and other logic in a .SUBCKT, in a library. (The same way that we model other MSI parts.)

There are two ways to provide the program data for Programmable Logic Arrays. The normal way is to give the name of a JEDEC format file which contains the program data. This file would normally be produced by a PLD design package which translates logic design information into a program for a specific programmable logic part. The other way to program the logic array is by including the program data, in order, on the device line (with the DATA=... construct).

If parts from the MicroSim library are used with a JEDEC file, then the engineer does not need to use the Programmable Logic Array primitive, or any of the model information below, since the library contains all of the appropriate modeling information. Using a PLD from the library is just like using any other logic device from the library, except that you need to tell PSpice the name of the JEDEC file which contains the program for that part. A TEXT parameter named JEDECFILE is used to specify the file name, as shown in the example below:

```
  X1 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12 IN13 IN14
+ OUT1 OUT2 OUT3 OUT4
  + PAL14H4
  + TEXT: JEDECFILE = "myprog.jed"
```

This example creates a 14H4 PAL which is programed by the JEDEC file myprog.jed.

Programmable Logic Array Device Format:

U<name> <pld type>(<#inputs>, <#outputs>) <input_node>* <output_node>* <(timing model) name> <(io_model) name> [FILE=<(file name) text value>] [DATA=<radix flag>$<program data>$] [MNTYMXDLY=<(delay select) value>] [IOLEVEL=<(interface model level) value>]

Where:

pld type is one of:

PLAND	and array
PLOR	or array
PLXOR	exclusive-or array
PLNAND	and-invert array
PLNOR	or-invert array

PLNXOR exclusive-or-invert array
PLANDC and array with true and complement columns for each input
PLORC or array with true and complement columns for each input
PLXORC exclusive-or array with true and complement columns for each input
PLNANDC and-invert array with true and complement columns for each input
PLNORC or-invert array with true and complement columns for each input
PLNXORC exclusive-or-invert array with true and complement columns for each input

file name text value is:

The name of a JEDEC format file which specifies the programming data for the array. The file name may be specified as a text constant (enclosed in double quotes '""'), or as a text expression (enclosed in vertical bars '|'). If a FILE is specified, any programming data specified by a DATA section is ignored. The mapping of addresses in the JEDEC file to locations in the array is controlled by model parameters specified in the timing model.

radix flag is one of:

B binary data follows
O octal data follows (most significant bit has lowest address)
X hexadecimal data follows (most significant bit has lowest address)

program data:

The program data is a string of data values used to program the logic array. The values start at address zero, which programs the array for the connection of the first input pin to the gate which drives the first output. A '0' indicates that the input is not connected to the gate, and a '1' indicates that the input is connected to the gate. (Initially, all inputs are not connected to any gates.) The next value programs the array for the connection of the complement of the first input to the gate which drives the first output (if this is a programmable gate with true and complement inputs) or, the second input connection to the gate which drives the first output. Each additional '1' or '0' programs the connection of the next input or its complement to the gate which drives the first output, until the connection of all inputs (and their complements) to that gate have been programmed. Data values after that program the connection of inputs to the gate driving the second output, and so on.

The data values must be enclosed in dollar signs ('$'). The data values may have spaces or continuation lines between the data values.

The example below defines a 3-to-8 line decoder. The inputs are IN1 (MSB), IN2, and IN3 (LSB). If the inputs are all low, OUT0 is true. If IN1 and IN2 are low and IN3 is hi, then OUT1 is true, and so on. The programming data has been typed in as an array, so that is easier to read. The comments above the columns identify the true and false (complement) inputs, and the comments at the end of the line identify the output pin which is controlled by that gate. (PSpice does not process any of these comments, they just help make the programming data readable.)

```
UDECODE PLANDC(3, 8)     ; 3 inputs, 8 outputs
+ IN1 IN2 IN     3       ; the inputs
+ OUT0 OUT1 OUT2 OUT3 OUT4 OUT5 OUT6 OUT7    ; the outputs
+ PLD_MDL                ; the timing model name
+ IO_STD                 ; the I/O model name
+ DATA=B$                ; the programming data
* IN1 IN2 IN3
* TF  TF  TF
+ 01  01  01            ; OUT0
+ 01  01  10            ; OUT1
+ 01  10  01            ; OUT2
+ 01  10  10            ; OUT3
+ 10  01  01            ; OUT4
+ 10  01  10            ; OUT5
+ 10  10  01            ; OUT6
+ 10  10  10 $          ; OUT7
```

The timing model for the Programmable Logic Arrays has a model type of UPLD. The model parameters are:

Name	Description	Units	Default
TPLHMN	delay: in to out, low to hi, min	sec	0
TPLHTY	delay: in to out, low to hi, typ	sec	0
TPLHMX	delay: in to out, low to hi, max	sec	0
TPHLMN	delay: in to out, hi to low, min	sec	0
TPHLTY	delay: in to out, hi to low, typ	sec	0
TPHLMX	delay: in to out, hi to low, max	sec	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.	std true/cmp	1 2
OUTSCALE	JEDEC file mapping: amount the JEDEC file address changes for each new output pin (gate).	std true/cmp	# inputs 2*(# inputs)

MNTYMXDLY delay selection: 0=default, 1=min, 2=typ, 3=max 0

2) Read Only Memories

The PSpice digital simulator now has primitive logic devices which are read only memories.

There are two ways to provide the program data for ROMs. The normal way is to give the name of an Intel Hex Format file. This file is read before the simulation starts, and the ROM is "programmed" to match the data in the file. The other way to program the ROM is to include the program data on the device line (with the DATA=... construct).

If parts from the MicroSim library are used with an Intel Hex format program file, then the engineer does not need to use the ROM device primitive, or any of the model information below, since the library contains all of the appropriate modeling information. Using a ROM from the library is just the same as using any other logic device from the library, except that you need to tell PSpice the name of the Intel Hex file which contains the ROM program. A TEXT parameter named HEXFILE is used to specify the file name, as shown in the following example.

```
X1 CEbar OEbar A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
+ O0 O1 O2 O3 O4 O5 O6 O7
+ ROM2732A
+ TEXT: HEXFILE = "mydata.hex"
```

This example creates a 2732A PROM (4K x 8) which is programmed by the Hex file mydata.hex.

NOTE: The 4.04P release (Beta Site 4.04) does not support the reading of Intel Hex Format files.
Read Only Memory Device Format:

U<name> ROM(<#address pins>, <#output pins>) <enable_node> <addr_node>* <output_node>* <(timing model) name> <(io_model) name> [FILE=<(file name) text value>] [DATA=<radix flag>$<program data>$] [MNTYMXDLY=<(delay select) value>] [IOLEVEL=<(interface model level) value>]

Where:

Address nodes are given MSB first.

file name text value is:

The name of an Intel Hex format file which specifies the programming data for the ROM. The file name may be specified as a text constant (enclosed in double quotes '""'), or as a text expression (enclosed in vertical bars '|'). If a FILE is specified, any programming data specified by a DATA section is ignored.

radix flag is one of:

B binary data follows
O octal data follows (most significant bit has lowest address)

X hexadecimal data follows (most significant bit has lowest address)

program data:

The program data is a string of data values used to program the ROM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs ('$'). The data values may have spaces or continuation lines between the data values.

The following example defines a 4-bit by 4-bit to 8-bit multiplier ROM.

```
UMULTIPLY ROM(8, 8)      ; 8 address bits, 8 outputs
+ AIN0 AIN1 AIN2 AIN3    ; the first 4 bits of address
+ BIN0 BIN1 BIN2 BIN3    ; the second 4 bits of address
+ OUT0 OUT1 OUT2 OUT3 OUT4 OUT5 OUT6 OUT7 ; the outputs
+ ROM_MDL                ; the timing model name
+ IO_STD                 ; the I/O model name
+ DATA=X$                ; the programming data
*  B input value:
*  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
+ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; A =0
+ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F ; A =1
+ 00 02 04 06 08 0A 0C 0E 10 12 14 16 18 1A 1C 1E ; A =2
+ 00 03 06 09 0C 0F 12 15 18 1B 1E 21 24 27 2A 2D ; A =3
+ 00 04 08 0C 10 14 18 1C 20 24 28 2C 30 34 38 3C ; A =4
+ 00 05 0A 0F 14 19 1E 23 28 2D 32 37 3C 41 46 4B ; A =5
+ 00 06 0C 12 18 1E 24 2A 30 36 3C 42 48 4E 54 5A ; A =6
+ 00 07 0E 15 1C 23 2A 31 38 3F 46 4D 54 5B 62 69 ; A =7
+ 00 08 10 18 20 28 30 38 40 48 50 58 60 68 70 78 ; A =8
+ 00 08 12 1B 24 2D 36 3F 48 51 5A 63 6C 75 7E 87 ; A =9
+ 00 0A 14 1E 28 32 3C 46 50 5A 64 6E 78 82 8C 96 ; A =A
+ 00 0B 16 21 2C 37 42 4D 58 63 6E 79 84 8F 9A A5 ; A =B
+ 00 0C 18 24 30 3C 48 54 60 6C 78 84 90 9C A8 B4 ; A =C
+ 00 0D 1A 27 34 41 4E 5B 68 75 82 8F 9C A9 B6 C3 ; A =D
+ 00 0E 1C 2A 38 46 54 62 70 7E 8C 9A A8 B6 C4 D2 ; A =E
+ 00 0F 1E 2D 3C 4B 5A 69 78 87 96 A5 B4 C3 D1 E1$; A =F
```

The timing model for the ROM has a model type of UROM. The model parameters are:

Name	Description	Units	Default
TPADHMN	delay: address to data, low to hi, min	sec	0
TPADHTY	delay: address to data, low to hi, typ	sec	0
TPADHMX	delay: address to data, low to hi, max	sec	0
TPADLMN	delay: address to data, hi to low, min	sec	0
TPADLTY	delay: address to data, hi to low, typ	sec	0
TPADLMX	delay: address to data, hi to low, max	sec	0
TPEDHMN	delay: enable to data, hiZ to hi, min	sec	0
TPEDHTY	delay: enable to data, hiZ to hi, typ	sec	0
TPEDHMX	delay: enable to data, hiZ to hi, max	sec	0
TPEDLMN	delay: enable to data, hiZ to low, min	sec	0
TPEDLTY	delay: enable to data, hiZ to low, typ	sec	0
TPEDLMX	delay: enable to data, hiZ to low, max	sec	0
TPEDHZMN	delay: enable to data, hi to hiZ, min	sec	0
TPEDHZTY	delay: enable to data, hi to hiZ, typ	sec	0
TPEDHZMX	delay: enable to data, hi to hiZ, max	sec	0
TPEDLZMN	delay: enable to data, low to hiZ, min	sec	0
TPEDLZTY	delay: enable to data, low to hiZ, typ	sec	0
TPEDLZMX	delay: enable to data, low to hiZ, max	sec	0
MNTYMXDLY	delay selection: 0=default, 1=min, 2=typ, 3=max		0

3) Random Access Read-Write Memories

The PSpice digital simulator now has primitive logic devices which are random access read-write memories.

The RAM is normally initialized with unknown data at all addresses. There are two ways to provide other initialization data for RAMs. The normal way is to give the name of an Intel Hex Format file. This file is read before the simulation starts, and the RAM is initialized to match the data in the file. The other way to initialize the RAM is to include the initialization data on the device line (with the DATA=... construct).

If parts from the MicroSim library are used with an Intel Hex format initialization file, then the engineer does not need to use the RAM device primitive, or any of the model information below, since the library contains all of the appropriate modeling information. Using a RAM from the library is just the same as using any other logic device from the library, except that you may want to tell PSpice the name of an Intel Hex file which contains the RAM initialization. A TEXT parameter named HEXFILE is used to specify the file name, as shown in the example below. If no initialization is needed, then you can omit the HEXFILE parameter, or set it to an empty name ("").

```
X1 CSbar WEbar A0 A1 A2 A3 A4 A5 A6 A7 A8 A9
+ O0 O1 O2 O3
+ RAM2114A
+ TEXT: HEXFILE = "mydata.hex"
```

This example creates a 2114A RAM (1K x 4) which is programmed by the Hex file mydata.hex.

NOTE: The 4.04P release (Beta Site 4.04) does not support the reading of Intel Hex Format files.
Random Access Read-Write Memory Device Format:

```
U<name> RAM(<#address bits>, <#output bits>)
<read_enable_node> <write_enable_node> <addr_node>*
<write_data_node>* <read_data_node>* <(timing model)
name> <(io_model) name> [FILE=<(file name) text
value>] [DATA=<radix flag>$<initialization data>$]
[MNTYMXDLY=<(delay select) value>] [IOLEVEL=<(interface
model level) value>]
```

Where:

Address nodes are given MSB first.

file name text value is:

The name of an Intel Hex format file which specifies the initialization data for the RAM. The file name may be specified as a text constant (enclosed in double quotes '""'), or as a text expression (enclosed in vertical bars '|'). If a FILE is specified, any initialization data specified by a DATA section is ignored.

radix flag is one of:

B binary data follows
O octal data follows (most significant bit has lowest address)
X hexadecimal data follows (most significant bit has lowest address)

initialization data:

The initialization data is a string of data values used to initialize the RAM. The values start at address zero, first output bit. The next bit specifies the next output bit, and so on until all of the output bits for that address have been specified. Then the output values for the next address are given, and so on.

The data values must be enclosed in dollar signs ('$'). The data values may have spaces or continuation lines between the data values.

The initialization of a RAM using the DATA=... construct is the same as the programming of a ROM. See the ROM for an example.

The timing model for the RAM has a model type of URAM. The model parameters are:

Name	Description	Units	Default
TPADHMN	delay: address to read data, low to hi,min	sec	0
TPADHTY	delay: address to read data, low to hi,typ	sec	0
TPADHMX	delay: address to read data, low to hi,max	sec	0
TPADLMN	delay: address to read data, hi to low,min	sec	0
TPADLTY	delay: address to read data, hi to low,typ	sec	0
TPADLMX	delay: address to read data, hi to low,max	sec	0
TPERDHMN	delay: read enable to read data, hiZ to hi, min	sec	0
TPERDHTY	delay: read enable to read data, hiZ to hi, typ	sec	0
TPERDHMX	delay: read enable to read data, hiZ to hi, max	sec	0
TPERDLMN	delay: read enable to read data, hiZ to low, min	sec	0
TPERDLTY	delay: read enable to read data, hiZ to low, typ	sec	0
TPERDLMX	delay: read enable to read data, hiZ to low, max	sec	0
TPERDHZMN	delay: read enable to read data, hi to hiZ, min	sec	0
TPERDHZTY	delay: read enable to read data, hi to hiZ, typ	sec	0
TPERDHZMX	delay: read enable to read data, hi to hiZ, max	sec	0
TPERDLZMN	delay: read enable to read data, low to hiZ,min	sec	0
TPERDLZTY	delay: read enable to read data, low to hiZ,typ	sec	0
TPERDLZMX	delay: read enable to read data, low to hiZ,max	sec	0
TSUDEWMN	min setup time: data to write enable rise, min	sec	0
TSUDEWTY	min setup time: data to write enable rise, typ	sec	0
TSUDEWMX	min setup time: data to write enable rise, max	sec	0
TSUAEWMN	min setup time: address to write enable rise, min	sec	0
TSUAEWTY	min setup time: address to write enable rise, typ	sec	0
TSUAEWMX	min setup time: address to write enable rise, max	sec	0
TWEWHMN	min width: enable write hi, min	sec	0
TWEWHTY	min width: enable write hi, typ	sec	0
TWEWHMX	min width: enable write hi, max	sec	0
TWEWLMN	min width: enable write low, min	sec	0
TWEWLTY	min width: enable write low, typ	sec	0
TWEWLMX	min width: enable write low, max	sec	0

THDEWMN min hold time: write enable fall to data change, min sec 0
THDEWTY min hold time: write enable fall to data change, typ sec 0
THDEWMX min hold time: write enable fall to data change, max sec 0
THAEWMN min hold time: write enable fall to addr change, min sec 0
THAEWTY min hold time: write enable fall to addr change, typ sec 0
THAEWMX min hold time: write enable fall to addr change, max sec 0
MNTYMXDLY delay selection: 0=default, 1=min, 2=typ, 3=max 0

The RAM has separate read and write sections, with separate data and enable pins, and shared address pins. To write to the RAM the address and write data signals must be stable for the appropriate setup times, then write enable is raised. It must stay high for at least the minimum time, then fall. Address and data must remain stable while write enable is high, and for the hold time after it falls. Write enable must remain low for at least the minimum time, before changing.

To read from the RAM, raise read enable, and the outputs will change from hiZ to the appropriate value after a delay. The address may change while read enable is hi, and if it does, the new data will be available at the outputs after the delay.

Nothing prevents both read and write enable from being true at the same time, although most real devices would not allow this. The new value from the write is sent to the read data outputs on the falling edge of write enable.

4) New TEXT parameter

.TEXT Text parameter definition

General Form:

.TEXT < <name> = " <text value> " >*

.TEXT < <name> = | <text expression> | >*

Examples

```
.TEXT MYFILE = "FILENAME.EXT"
.TEXT FILE = "ROM.DAT", FILE2 = "ROM2.DAT"
.TEXT PROGDAT = |"ROM"+TEXTINT(RUN_NO)+".DAT"| ;
.TEXT DATA1 = "PLD.JED", PROGDAT =
|"\PROG\DAT\"+FILENAME|
```

The keyword .TEXT is followed by a list of names with text values. The values may be text constants (enclosed in "), or text expressions (enclosed in |). Text expressions may contain only text constants or previously defined parameters.

<name> can not be a .PARAM name, or any of the reserved .PARAM names.

Once defined a text parameter can be used in only these places: to specify a JEDEC file to program a Programmable

Logic Array, to specify an Intel Hex file to program a Read Only Memory, to specify an Intel Hex file to initialize a Random Access Read-Write Memory, to specify a text parameter value for a subcircuit, as part of a text expression used in one of these places or in a .TEXT statement. Note that text parameters and expressions are not useful unless you have the Digital Simulation Option.

Text expressions may contain: text constants (enclosed in "), text parameters, the '+' operator which concatenates two text values, or the TEXTINT(<value or expression>) function which returns a text string which is the integer value closest to the value of the <value or expression> (<value or expression> is a floating point value).

STIMULUS EDITOR

1) A mouse can now be used on all platforms to make menu selections and to perform other miscellaneous functions easier than can be done using the keyboard.

2) For SUN systems: In Probe, Parts, and StmEd the colors used for the foreground, background, and the traces on Sun systems can be changed. For details see the "Probe" section in this document.

CONTROL SHELL

1) "Set-up" has been moved from the Probe Menu to the bottom of the Files Menu. It has been renamed "Display/Prt Setup...".

2) The default device file is now PSPICE.DEV. The Control Shell will not let you run StmEd or Probe if a device file is not found.

3) The Probe/Auto-Run menu item now defaults to yes.

4) The editor and browser fall back to allocating 16K of memory if they can't find 32K.