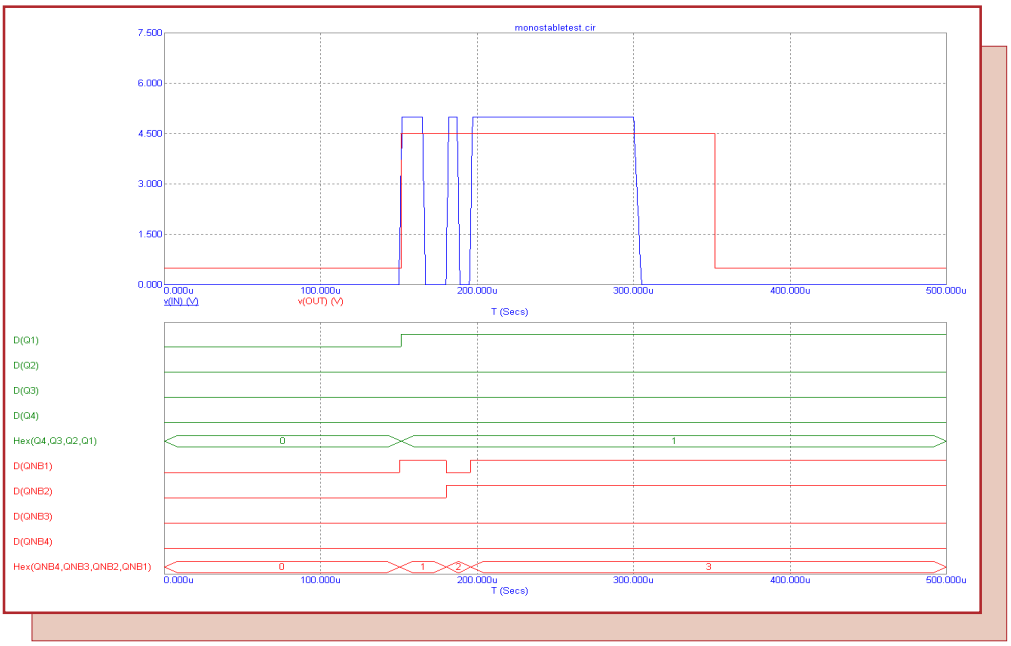
# spectrum news

Applications for Micro-Cap™ Users

# Summer 2006

## News




## Monostable Multivibrator Macro

Featuring:

- Monostable Multivibrator Macro
- Correlating Monte Carlo Parameters
- Using the Sample and Hold Source

## News In Preview

This newsletter's Q and A section describes the differences between the FFT, FFTS, and HARM operators. The Easily Overlooked Feature section describes the Apply Display Properties command which can copy the attribute settings, color, font, and display options of a selected component to all other instances of that component in the schematic.

The first article describes a macro model that can be used to simulate a nonretriggerable monostable multivibrator.

The second article describes how to correlate parameter variations so that they track during a Monte Carlo analysis. The correlation is done by using the lot number value available within the tolerance specification.

The third article describes how to use the Sample and Hold source component that is available within Micro-Cap. The source has the capability to do both sample and hold along with track and hold depending on the attributes defined for the source.

## Contents

## Book Recommendations

### General SPICE

• *Computer-Aided Circuit Analysis Using SPICE*, Walter Banzhaf, Prentice Hall 1989. ISBN# 0-13-162579-9

• *Macromodeling with SPICE*, Connelly and Choi, Prentice Hall 1992. ISBN# 0-13-544941-3

• *Inside SPICE-Overcoming the Obstacles of Circuit Simulation*, Ron Kielkowski, McGraw-Hill, First Edition, 1993. ISBN# 0-07-911525-X

• *The SPICE Book,* Andrei Vladimirescu, John Wiley & Sons, Inc., First Edition, 1994. ISBN# 0-471-60926-9

### MOSFET Modeling

• *MOSFET Models for SPICE Simulation, William Liu, Including BSIM3v3 and BSIM4*, Wiley-Interscience, First Edition, ISBN# 0-471-39697-4

## VLSI Design

• *Introduction to VLSI Circuits and Systems,* John P. Uyemura, John Wiley & Sons Inc, First Edition, 2002 ISBN# 0-471-12704-3

### Micro-Cap - Czech

• *Resime Elektronicke Obvody,* Dalibor Biolek, BEN, First Edition, 2004. ISBN# 80-7300-125-X

### Micro-Cap - German

• *Schaltungen erfolgreich simulieren mit Micro-Cap V,* Walter Gunther, Franzis', First Edition, 1997. ISBN# 3-7723-4662-6

### Micro-Cap - Finnish

• *Elektroniikkasimulaattori,* Timo Haiko, Werner Soderstrom Osakeyhtio, 2002. ISBN# ISBN 951-0-25672-2

### Design

• *Microelectronic Circuits High Performance Audio Power Amplifiers,* Ben Duncan, Newnes, First Edition, 1996. ISBN# 0-7506-2629-1

• *Microelectronic Circuits.,* Adel Sedra, Kenneth Smith, Fourth Edition, Oxford, 1998

### High Power Electronics

• *Power Electronics,* Mohan, Undeland, Robbins, Second Edition, 1995. ISBN# 0-471-58408-8

• *Modern Power Electronics,* Trzynadlowski, 1998. ISBN# 0-471-15303-6

### Switched-Mode Power Supply Simulation

• *SMPS Simulation with SPICE 3,* Steven M. Sandler, McGraw Hill, First Edition, 1997. ISBN# 0-07-913227-8

• *Switch-Mode Power Supply SPICE Simulation Cookbook*, Christophe Basso, McGraw-Hill 2001. This book describes many of the SMPS models supplied with Micro-Cap.

## Micro-Cap Questions and Answers

**Question:** I am interested in performing some Fourier analysis simulations on my circuit. In looking through the manual, I have noticed that there appear to be three operators that seem to perform similar operations: FFT, FFTS, and HARM. What are the differences between each of these operators, and in which situations should I use each operator?

**Answer:** Each of these operators do produce similar data. The main difference between them lie in the scale factors used along with whether the operator treats its specified waveform as a magnitude or a complex number.

*FFT operator:* The FFT operator calculates the classical Fourier transform of the specified waveform and returns a complex quantity whose real part contains the scaled Fourier series An coefficients and imaginary part contains the scaled Fourier series Bn coefficients. The Fourier coefficients are scaled by N/2 (DC scaled by N) where N is the number of points calculated during the FFT operation. Since most engineering applications need the unscaled Fourier coefficients, the FFT operator is most useful when calculating the phase values of the Fourier transform such as with the expression PH(FFT(V(Out))).

*HARM operator:* The HARM operator calculates the amplitude of the harmonics for the specified waveform where the amplitude is the multiplier of the sine or cosine function. It returns a magnitude rather than a complex quantity. The HARM operator is only useful for plotting the harmonics of a waveform such as with the expression HARM(V(Out)) or dB(Harm(V(Out))).

*FFTS operator:* The FFTS operator is similar in operation to the HARM operator. It can also be used to plot the harmonics of a waveform such as with the expression FFTS(V(Out)). The difference between the two is that the FFTS operator returns a complex quantity so that it can be used to plot the Fourier series sine and cosine coefficients. The Fourier series cosine coefficients can be calculated through an expression such as Re(FFTS(V(Out))), and the Fourier series sine coefficients can be calculated through an expression such as Im(FFTS(V(Out))).

## Easily Overlooked Features

This section is designed to highlight one or two features per issue that may be overlooked because they are not made visually obvious with a toolbar button.

**Apply Display Properties Command**
When a component is placed in the schematic, it will get its initial display properties from settings in the Schematic Properties dialog box and the Component Editor. The individual display settings of the component can then be edited within its Attribute dialog box. In some cases, it can be useful to apply the display settings of a component to the other instances of the component that exist within the schematic. The Apply Display Properties command provides the ability to do this for the location, visibility, font, and color of the attribute text, the color of the component, and the settings for the display options.

To apply a component's properties to other instances, make sure that the component is selected. Then go to the Edit menu, click on Change, and select Apply Display Properties. The dialog box below will appear:

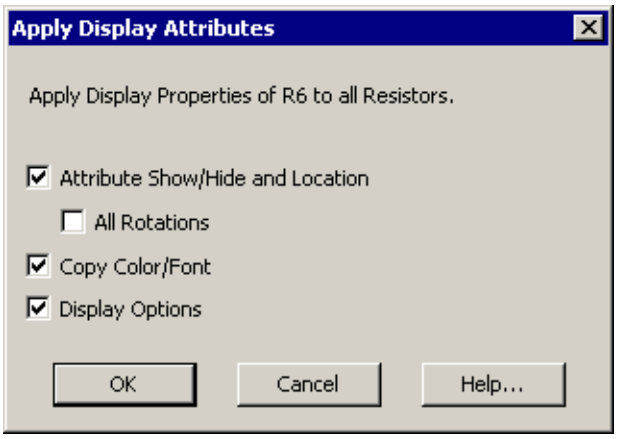



***Fig. 1 - Apply Display Properties dialog box***

The following options are available:

**Attribute Show/Hide and Location:** When enabled, the visibility and location of the attribute text will be copied to other instances of the part. Each component has eight possible rotations. When the All Rotations checkbox is enabled, it will apply to every instance of the part. When the All Rotations option is disabled, only the instances that have the same rotation will be affected.

**Copy Color/Font:** When enabled, the component color and attribute text font and color will be copied to all other instances of the part in the schematic.

**Display Options:** When enabled, the enable/disable settings for the display options (current, power, condition, pin markers, pin names, and pin numbers) will be copied to all other instances of the part in the schematic.

## Monostable Multivibrator Macro

Also known as a one-shot, the monostable multivibrator is a basic two state device where only one of the states is stable. When triggered, the output of the monostable multivibrator will toggle to the unstable state for a specified length of time at which point it will then return to its stable state until triggered again. This device can be used for numerous types of applications such as switch debouncing, siren actuation, machine sequencing, pulse detection or any instance where an output pulse of a fixed time width needs to be produced from an intermittent input pulse. A macro model for a nonretriggerable monostable multivibrator appears in Figure 2. A nonretriggerable type will produce a single timed pulse output response no matter how many input triggers occur during the specified output pulse time.

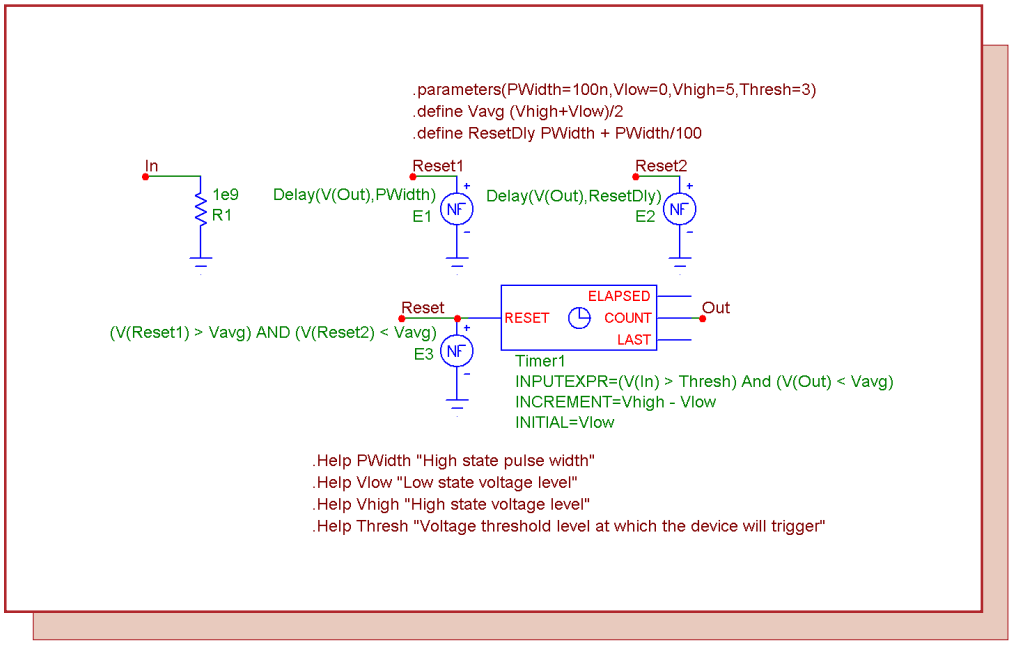



***Fig. 2 - Monostable multivibrator macro circuit***

The monostable macro has four input parameters: PWidth, Vlow, Vhigh, and Thresh. The PWidth parameter defines the length of time in seconds that the output pulse of the multivibrator will maintain its high state level for before reverting back to the stable low state. The Vlow parameter defines the voltage level for the low state output. The Vhigh parameter defines the voltage level for the high state output. The Thresh parameter defines the voltage level that the input signal will need to exceed in order to trigger the device.

The input of the macro is located at node In. A 1Gohm resistor has been placed at this node to provide a DC path to ground. At the heart of the monostable multivibrator macro is a Timer component. The relevant attributes of the Timer component have been defined as follows:

INPUTEXPR = (V(In) > Thresh) And (V(Out) < Vavg)
INCREMENT = Vhigh - Vlow
INITIAL = Vlow

For the Timer component, the voltage at the Count pin will be set to the INITIAL value at the beginning of a simulation and when the Reset pin is set to a non-zero value. When the INPUTEXPR returns a True (non-zero) value, the voltage at the Count pin will be increased by the INCREMENT value. In this case, the input expression will be true when the voltage at node In exceeds the specified Thresh level, and the voltage at node Out is below a value of Vavg which has been defined as follows:

.define Vavg (Vhigh+Vlow)/2

Checking to see if the voltage at Out is less than the average value of the Vhigh and Vlow parameters ensures that the output is at a low state. This models the nonretriggerable feature of the multivibrator by masking any low to high input transitions on the input while the output is in a high state. Since the initial value of the Count pin is set to the Vlow parameter, whenever the input expression produces a true result, the Count pin will be incremented by the difference between the Vhigh and Vlow parameters in order to produce the Vhigh voltage level at the output.

The width of the output pulse is modeled through the use of the three nonlinear function voltage sources, E1, E2, and E3. The E1 and E2 sources both use the Delay operator to produce a delayed version of the Timer's Count pin output. These sources have their VALUE attributes defined respectively as:

Delay(V(Out),PWidth)
Delay(V(Out),ResetDly)

where ResetDly is defined as:

.define ResetDly PWidth + PWidth/100

The output of the E1 source is the Count output waveform but delayed by PWidth seconds. The output of the E2 source produces the same waveform as the E1 source but time shifted by an additional PWidth/100 seconds. The E3 NFV source then uses the output of both of these sources as follows:

(V(Reset1) > Vavg) AND (V(Reset2) < Vavg)

When the output of the E1 source is greater than Vavg and the output of the E2 source is less than Vavg, the E3 source will produce a value of one volt into the Reset pin of the Timer which will reset the Count pin back to its Initial value. The reset pulse is how the output of the multivibrator is set back to the stable low state. The only time the E3 expression will be true is when the output of the timer has been in a high state for PWidth seconds. At that point, the high state at the Count output has finally rippled through the delay time of the E1 source but not through the delay time of the E2 source. The combination of the E1 and E2 sources is used so that the reset pulse will only be active for PWidth/100 seconds, and the monostable multivibrator will then be ready to receive another input trigger after the reset pulse is finished. There are a couple of issues to be aware of with the reset pulse. The first issue is that when the reset pulse to the timer is active, the timer count will not increment so any input trigger that occurs while the reset pulse is high will be ignored. For some simulations, the user may need to edit the ResetDly variable so that the reset pulse has a shorter width so that subsequent input triggers will work. The second issue can occur if the reset pulse has been set to too small of a value. If the simulation does not calculate a data point during the time period that the reset pulse will be high then the entire reset pulse will be skipped and the multivibrator will remain at its high state. Setting the

Maximum Time Step in the Transient Analysis Limits dialog box to a value that would ensure sampling during the reset pulse would solve this issue. The PWidth/100 value was determined to be a good trade off between these two issues while allowing for a wide range of output pulse widths.

A simple example schematic using the monostable multivibrator macro appears in Figure 3. In this example, the monostable multivibrator is being used as a switch debouncer in order to supply a clean clock pulse. Switch bounce can occur because a switch may not switch cleanly and can have multiple transitions during the time required to open or close. A debouncer will suppress the additional transitions from passing through to the rest of the circuit.

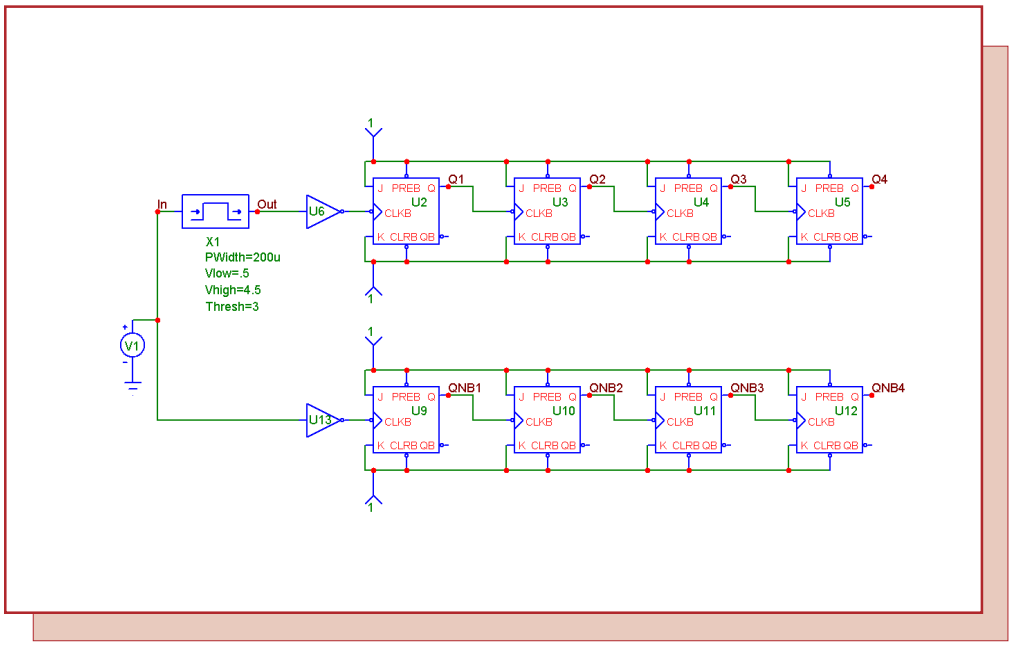



***Fig. 3 - Monostable multivibrator example circuit***

The switch bounce waveform is modeled through the use of the PWL capability within the Voltage Source component. The PWL capability provides the means to define the X,Y data points of a waveform and can be very useful when creating unique pulse waveforms. This source is then fed into the monostable multivibrator macro. The parameters for the macro have been defined as:

PWidth = 200u
Vlow = .5
Vhigh = 4.5
Thresh = 3

The monostable multivibrator will produce a 4.5 volt, 200us pulse at the output when the input signal to the macro exceeds 3 volts on a positive going transition. The output of the multivibrator provides a clean clock pulse to a four bit counter that consist of JK flip-flops that are setup to count upon each valid transition of the clock input. An inverter is placed between the macro and the counter so that the macro output will be converted into the negative going clock transition that the JK flip-flops are looking for.

A duplicate counter circuit is also present in the schematic. This circuit does not have the monostable multivibrator macro between the switch bounce waveform and the counter. This circuit will be used to show the deleterious effect that the switch bounce can have on the counter circuit if a debouncer is not present. The resultant transient analysis of this schematic is displayed in Figure 4.

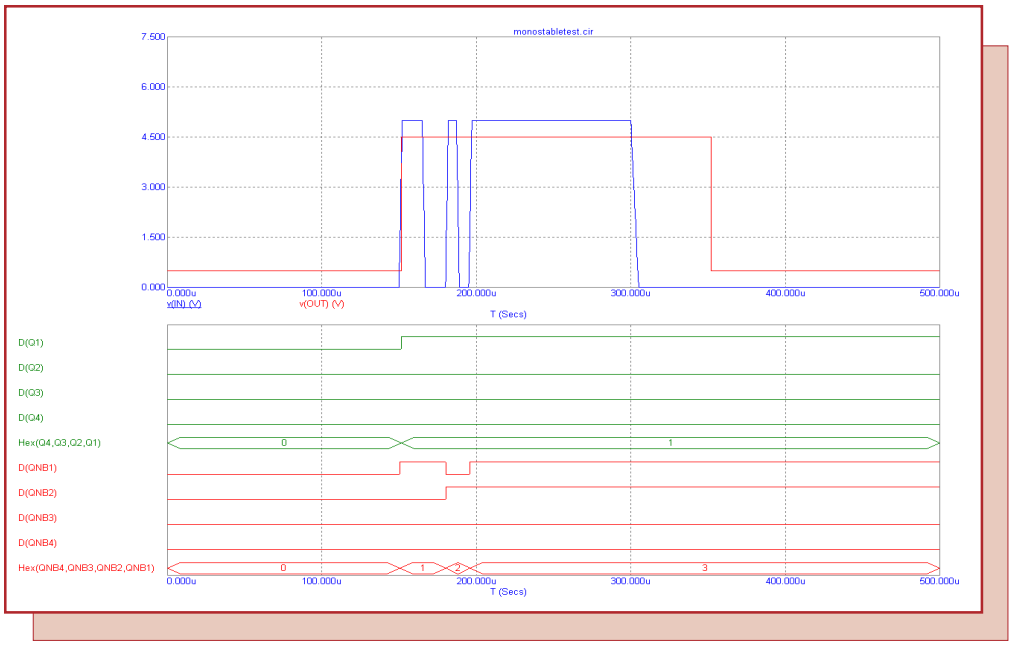



***Fig. 4 - Transient simulation results***

The transient simulation has been run for 500us. The top plot displays the input and output waveforms for the multivibrator macro. Note how the multiple transitions in the switch bounce input waveform have been turned into a single pulse at the output of the multivibrator.

The bottom plot shows the digital output of both of the counter circuits. The D(Q1) through D(Q4) waveforms display the output of the counter that has had its clock signal debounced. Only a single count has occurred as the Hex output of this counter has gone from zero to one. The D(QNB1) through D(QNB4) waveforms display the output of the counter that has the switch bounce waveform as the clock signal. In this instance, each of the transitions that occurred in the switch waveform end up causing the counter to increment, and the Hex output of this counter has gone from zero to three.

## Correlating Monte Carlo Parameters

A Monte Carlo simulation provides a method to view how the circuit performance can be affected by varying specified circuit parameters. The Monte Carlo simulation performs a number of runs varying the circuit parameters randomly within the tolerances defined in the schematic for each run. This is to ensure that the circuit will behave within acceptable limits while accounting for any possible variation in component values within the tolerance band.

Only model parameters and symbolic variables (created through define statements) are available for tolerancing. Both absolute (LOT) and relative (DEV) tolerances are available. Specifying a tolerance for a model parameter is done by including the keywords LOT and/or DEV after the model parameter value. An example of this for the BF parameter in an NPN model would be:

.Model N1 NPN (IS=1f BF=300 Lot=30% Dev=5% BR=50)

With the standard use of the tolerance specifications, the only correlation between different devices will be between those that reference the same model. The parameter values between devices using the same model will track exactly when just the LOT keyword is specified and the appropriate flag of either the PrivateAnalog or the PrivateDigital option in the Global Settings is disabled. When enabled, the PrivateAnalog and PrivateDigital options force each device to use their own private instance of the model so that no correlation is possible. If the DEV keyword is specified or the corresponding Private option is enabled, each device that references the model will have a unique tolerance value.

For the symbolic variables, only the absolute tolerance is available because only one instance of the variable is present. The LOT tolerance in this case is specified before the variable name. An example of tolerancing a variable called RTest would be:

.Define {Lot=10%} RTest 50

Since only one instance of the variable is present, all devices that reference the symbolic variable will be correlated.

The LOT and DEV keywords have optional syntax that provides an additional means of correlating the tolerance values of devices within a schematic. The full syntax for the LOT and DEV keywords are:

LOT[/<lot#>][/<distribution name>] = <value>[%]
DEV[/<lot#>][/<distribution name>] = <value>[%]

The <lot#> specifies which of ten random number generators, numbered 0 through 9, are used to calculate parameter values. This lets you correlate parameters within an individual model statement as well as parameters between models or symbolic variables. Using the <lot#> lets any toleranced parameter be correlated to any other toleranced parameter simply by sharing the same <lot#> value. The DEV random number generators are distinct from the LOT random number generators. Tolerances without this specification get unique random numbers.

The <distribution name> specifies the distribution that will be assigned to the toleranced parameter. It can be defined as Uniform, Gauss, or WCase to define the distribution as equal probability, Gaussian, or worst case. This setting will override the global distribution setting defined within the Monte Carlo Options dialog box for that parameter. A resistor model whose R multiplier is being toleranced can be specified as:

.MODEL RMOD RES (R=1 Lot/1/Gauss=10%)

This model statement would correlate the R parameter with any other parameter whose <lot#> specification has been assigned the value of 1 and sets the distribution method for this parameter to Gaussian.

An example schematic showing the use of the <lot#> in correlating parameters between model statements is shown in Figure 5. The schematic models a bass boost circuit where the R4 and R5 resistors simulate a 10kOhm potentiometer. There are numerous ways to model a potentiometer, but in order to display the use of the correlation feature, it will be modelled using just two 5kOhm resistors. The R4 resistor has its Model attribute defined as RMOD1, and the R5 resistor has its Model attribute defined as RMOD2. The two models are stored in the Models text page and are defined as:

.MODEL RMOD1 RES (R=1 Lot/1=50%)
.MODEL RMOD2 RES (R=1 Lot/1=-50%)

In both models, the R parameter, which is a resistance multiplier, is the parameter that is being toleranced. Since the R parameter is equal to 1 in the models, this technique will tolerance the actual resistance value by the specified percentage for any resistor that references the model. In the LOT keyword specification, the <lot#> entry in both models has been defined as 1. During a Monte Carlo simulation, the R parameter from each model will be correlated with each other. Note that the tolerance percentage for RMOD2 has actually been defined as -50% whereas the tolerance percentage for RMOD1 is 50%. Defining the percentages in this manner will actually set an inverse correlation between the two model parameters. For example, if the R4 resistor is toleranced to produce a resistance value of 5.3kOhm, then the R5 resistor will be toleranced to

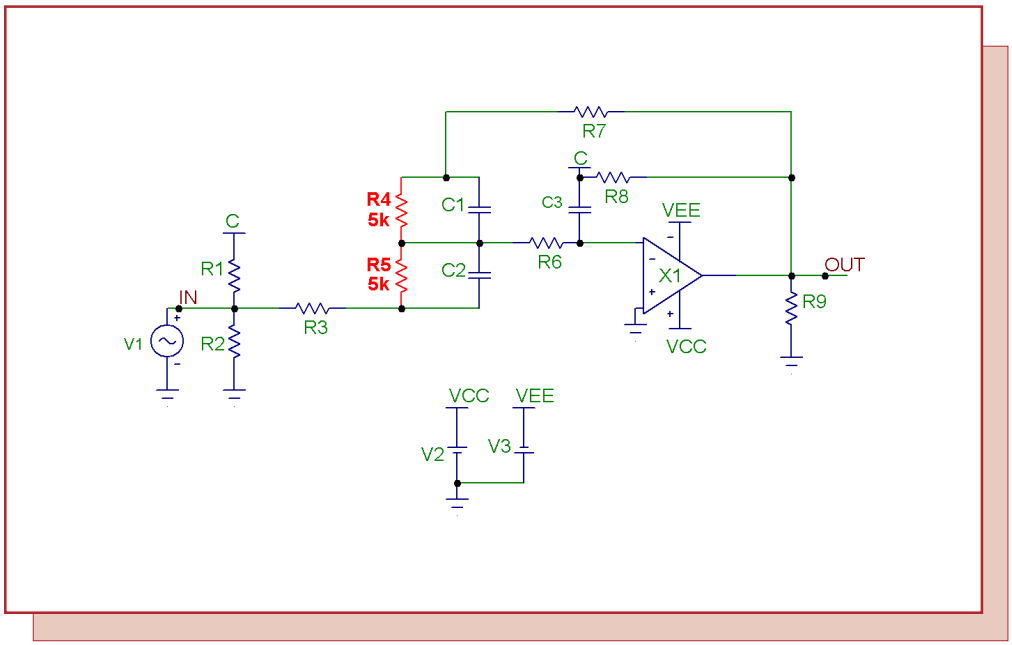



*Fig. 5 - Correlating tolerances circuit example*

produce the value of 4.7kOhm. If both tolerances had been set to 50%, then both resistances would have been set at 5.3kOhm. For the R4 and R5 resistors, the inverse correlation means that the series combination resistance of the two will always be equivalent to 10kOhm during any Monte Carlo run. Setting the resistors up in this manner is equivalent to tolerancing the position of the wiper in a 10kOhm potentiometer.

The AC analysis results of a Monte Carlo simulation are shown in Figure 6. The Monte Carlo options have been defined to perform a 100 run simulation using a Gaussian distribution.

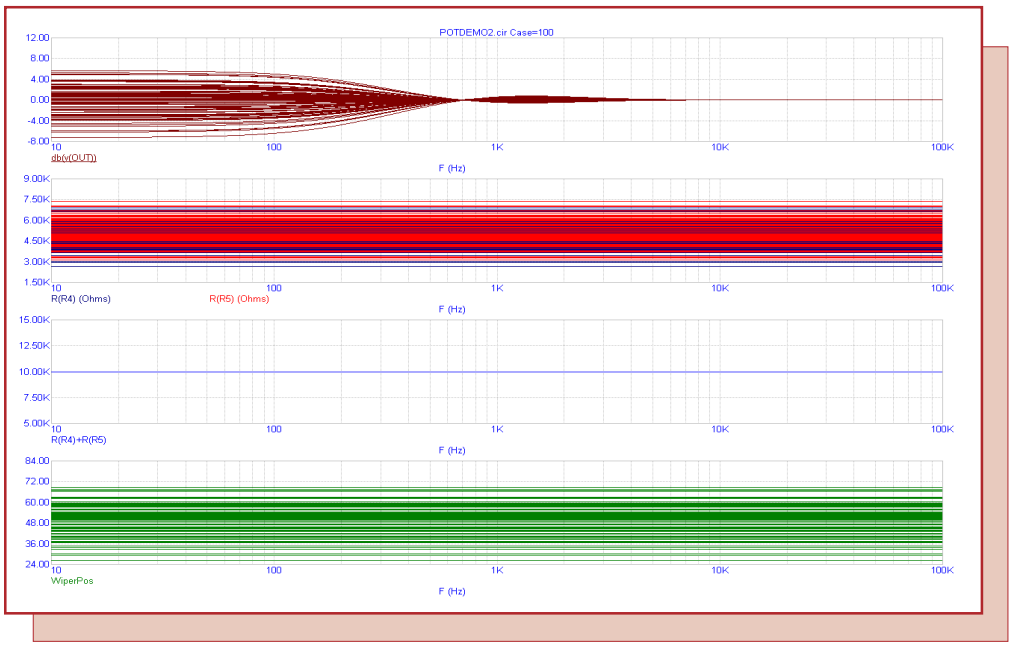



***Fig. 6 - Monte Carlo simulation output***

The first plot displays the output voltage of the bass boost circuit in decibels by plotting the expression dB(V(Out)).

The second plot displays the resistance values for both the R4 and the R5 resistor for each of the Monte Carlo runs.

The third plot displays the sum of the R4 and R5 resistances during the run. Due to the inverse correlation, the sum of the two resistors always comes out to 10kOhm. The outcome of the simulation is similar to randomly moving the wiper of the potentiometer to a new position and then viewing the resulting frequency response.

The fourth plot displays the output of a variable called WiperPos. The WiperPos variable was created with a .Define statement in the schematic whose syntax is:

.define WiperPos ((10k-R(R5))/10k)*100

The WiperPos equation will calculate the value of the wiper position in terms of the percentage that the wiper has been moved. At the 0% position, all 10kOhms would be represented by the R5 resistor, and at the 100% position, all 10kOhms would be represented by the R4 resistor.

A nice way to view the inverse correlation of the R4 and R5 resistances is to create a performance plot from their associated resistance waveforms. The performance plot uses performance functions to harvest a single data point from each branch of the simulation and then creates a plot from the corresponding set of data points. The performance plot created from one Monte Carlo simulation is shown in Figure 7.

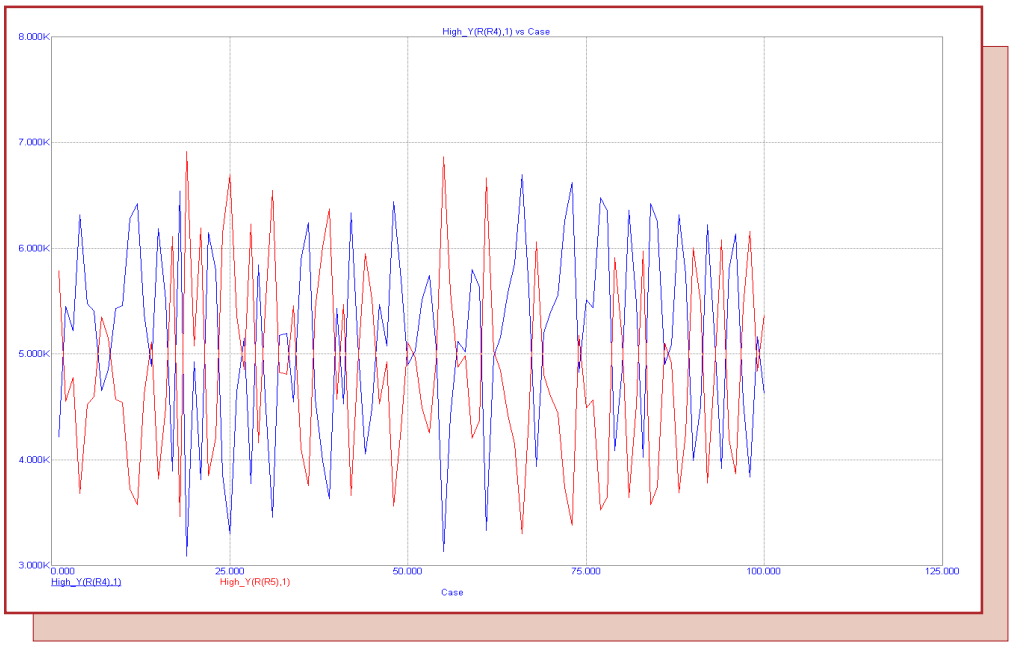



***Fig. 7 - Correlated R4 and R5 resistance values***

There are two waveforms in the performance plot. The performance functions used to create these waveforms are:

High_Y(R(R4),1)
High_Y(R(R5),1)

The High_Y performance function returns the maximum Y value for the specified waveform for each branch of the Monte Carlo simulation. Since the resistances are constant during each branch, the High_Y function just returns the resistance value for that branch. These waveforms are plotted versus the Monte Carlo branch that each value was calculated from. The inverse correlation between the two resistors can be easily viewed with this plot.

## Using the Sample and Hold Source

Sample and hold circuits are typically used in signal processing, event analysis, and analog to digital conversions among other applications. A sample and hold can be used as an interface for any device that requires a steady input signal for a specified length of time. Micro-Cap contains a component called Sample and Hold which is a dependent source that provides the basic functionality of an ideal sample and hold.

Located in the Analog Primitives / Special Purpose group under the Component menu, the Sample and Hold can operate in either a time periodic sample and hold mode or a track and hold mode. There are three attributes that control the operation of the Sample and Hold source: INPUT EXPR, SAMPLE EXPR, and PERIOD. The INPUT EXPR attribute is the input expression that the source will be sampling. While this attribute would typically be defined with a node voltage such as V(Out), the expression may also contain any valid circuit variable. The SAMPLE EXPR attribute defines a Boolean expression that determines when the input expression should be sampled. The PERIOD attribute defines a time period at which the input expression should be sampled. The source works as follows:

If the SAMPLE EXPR attribute is defined, whenever this Boolean expression evaluates to true (non-zero), the output of the source will be set to the value of the input expression. When the SAMPLE EXPR attribute evaluates to false, the output of the source will remain constant at the last value successfully sampled. In this case, the source will operate in track and hold mode.

If the PERIOD attribute is defined, the source will sample and store the value of the input expression once every PERIOD seconds. The output of the source will then remain constant until the next sample. In this case, the source will operate in sample and hold mode.

If both the SAMPLE EXPR attribute and the PERIOD attribute are defined, the SAMPLE EXPR will have priority, and the source will operate in track and hold mode.

A simple example of the use of the Sample and Hold Source is shown in Figure 8. The analog circuitry models a TTL inverter. The input to the inverter is at node In and is defined by a pulse source that produces a 3.5V, 40ns pulse every 100ns. The output of the inverter is at node A. The Sample and Hold source, S1, in the schematic produces an output at node B. For the initial simulation, the attributes of the Sample and Hold source are defined as:

INPUT EXPR = V(A)
SAMPLE EXPR =
PERIOD = 5n

The source will operate in sample and hold mode. It will sample the voltage at node A every 5ns. The resultant transient analysis output is shown in Figure 9. The transient simulation has been run for 200ns. The initial sample of the V(A) expression by the source occurs at T=0. Every 5ns after that, the Sample and Hold source will resample the V(A) expression and proceed to hold that value for 5ns until the next sampling.

Using the same circuit, the Sample and Hold source will next be defined to work in track and hold mode by simply defining the SAMPLE EXPR attribute. The attributes for the source are set to:

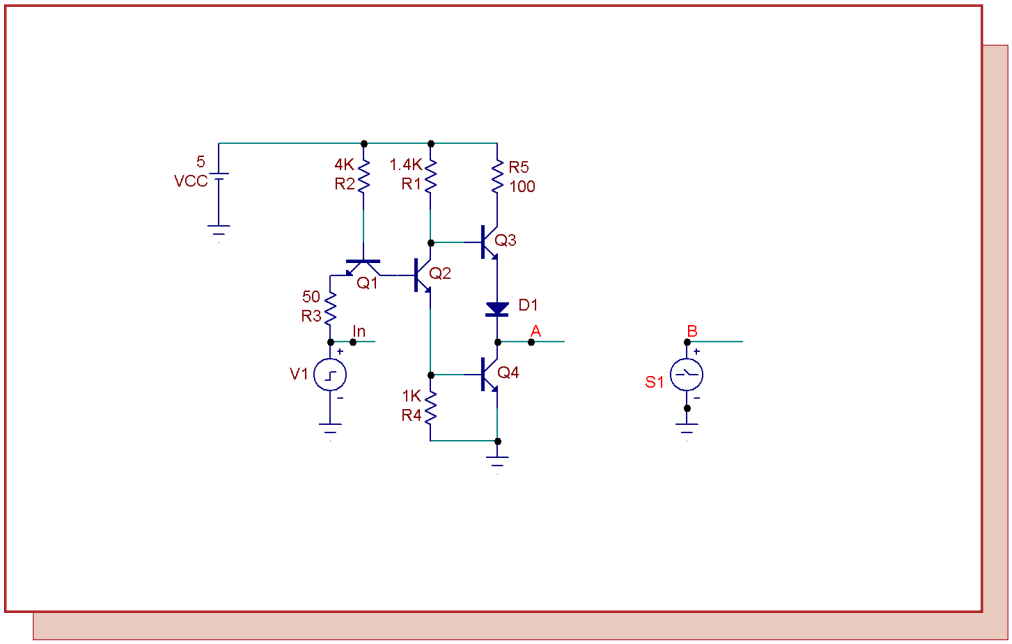



*Fig. 8 - Sample and Hold example circuit*

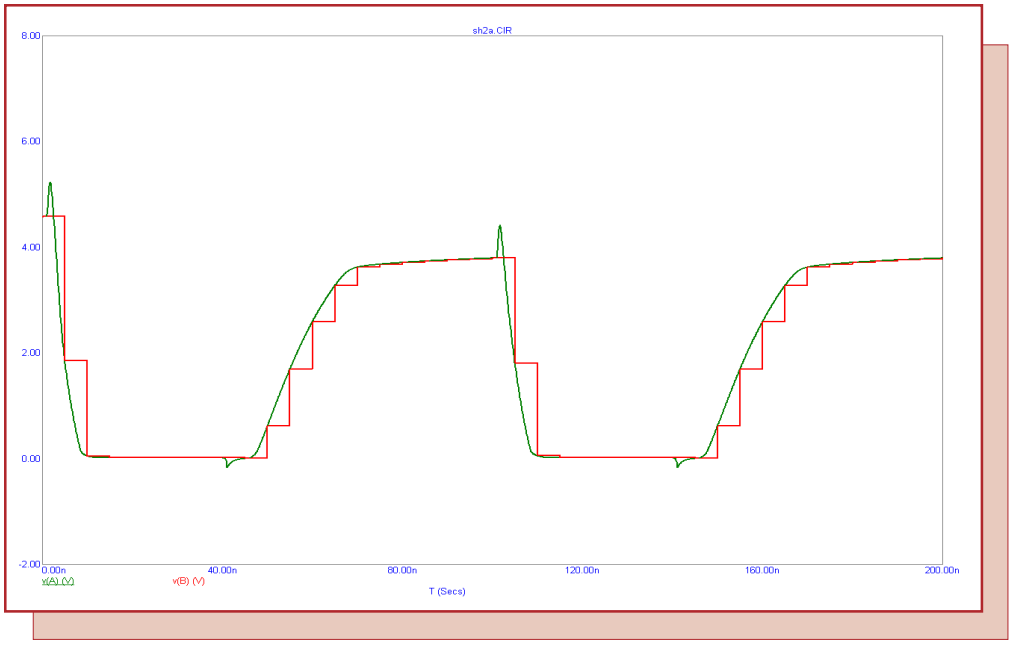



*Fig. 9 - Sample and hold mode analysis plot*

INPUT EXPR = V(A)
SAMPLE EXPR = (T Mod 200n > 40n) And (T Mod 200n < 120n)
PERIOD = 5n

Since the SAMPLE EXPR attribute has priority over the PERIOD attribute, the PERIOD attribute value can remain defined or the user can choose to delete it. The source will operate the same either way. The Boolean expression that has been defined for the SAMPLE EXPR attribute in this case will evaluate to true for an 80ns width during each 200ns of simulation time. The Mod function is a remainder after integer division operator that provides the periodic functionality for the expression. During each 200ns simulation time window, the expression will be true when the time value is greater than 40ns and less than 120ns in that 200ns window. For example, the expression will be true between 40ns and 120ns, 240ns and 320ns, 440ns and 520ns, etc.

The transient simulation of the source in track and hold mode is shown in Figure 10. While the SAMPLE EXPR is true (between 40ns and 120ns), the output of the Sample and Hold source tracks exactly with the V(A) expression that it is sampling. At 120ns, the SAMPLE EXPR becomes false, and the source will maintain the last valid sample for the rest of the simulation.

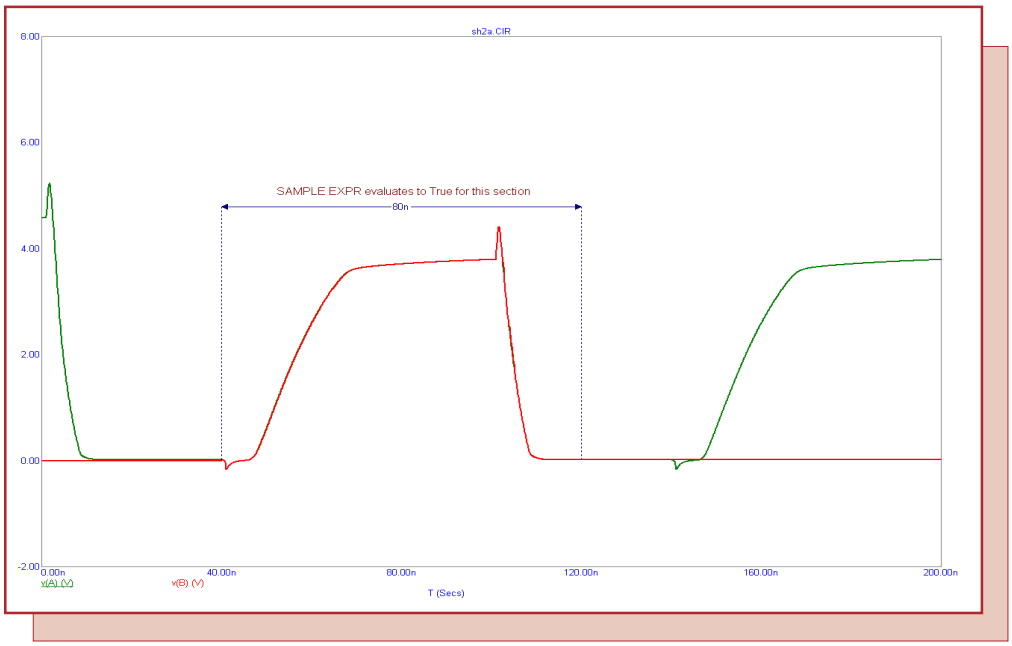



***Fig. 10 - Track and hold mode analysis plot***

## Product Sheet

### Latest Version numbers

Micro-Cap 8 ..... Version 8.1.2
Micro-Cap 7 ..... Version 7.2.4
Micro-Cap 6 ..... Version 6.3.3
Micro-Cap V ..... Version 2.1.2

### Spectrum's numbers

Sales ..... (408) 738-4387
Technical Support ..... (408) 738-4389
FAX ..... (408) 738-4702
Email sales ..... sales@spectrum-soft.com
Email support ..... support@spectrum-soft.com
Web Site ..... http://www.spectrum-soft.com
User Group ..... micro-cap-subscribe@yahoogroups.com