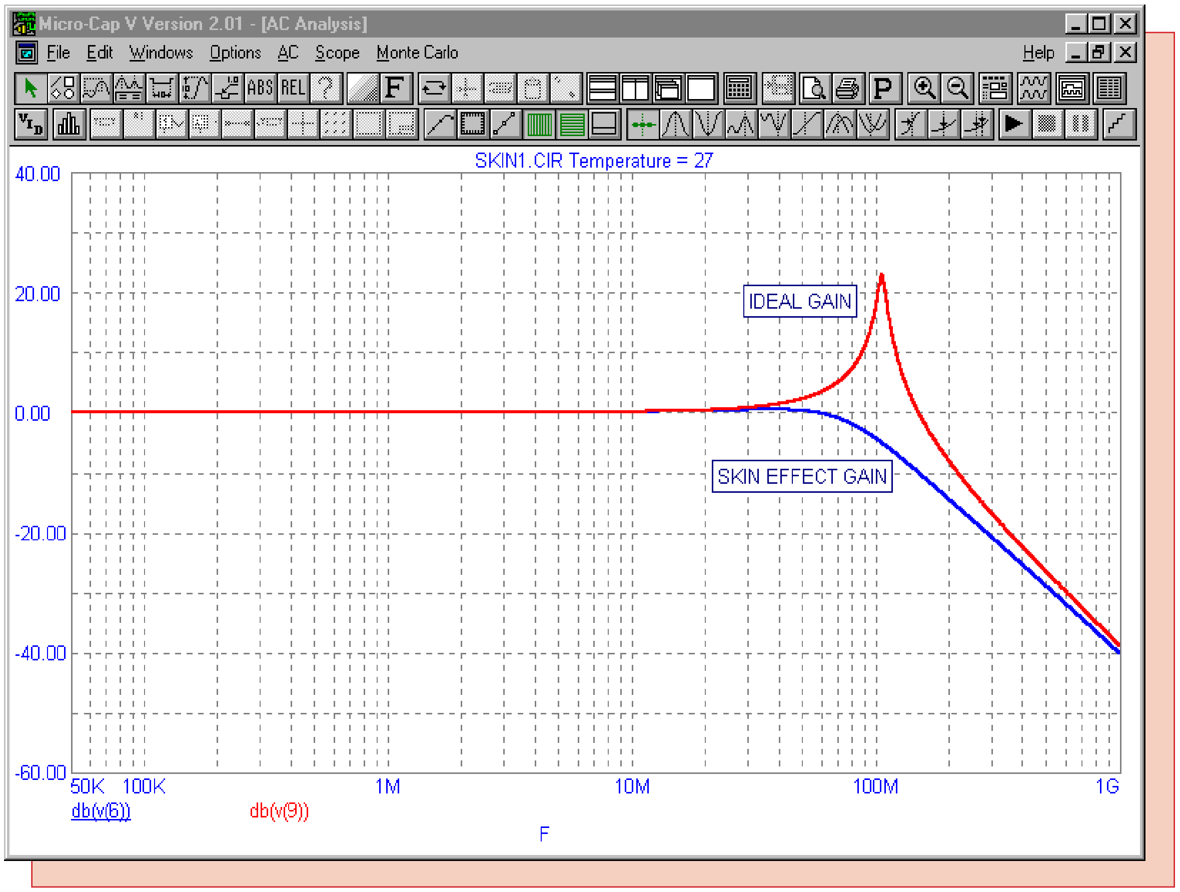
spectrum news

Applications for Micro-Cap™ Users

Fall 1997

Modeling Skin Effect


Micro-Cap V Version 2.01 - [AC Analysis]
File Edit Windows Options AC Scope Monte Carlo Help
SKIN1.CIR Temperature = 27
40.00
20.00
0.00
-20.00
-40.00
-60.00
50K
100K
1M
10M
100M
1G
IDEAL GAIN
SKIN EFFECT GAIN
db(v(6))
db(v(9))
F

Featuring:
- Noise Source Macro
- Modeling Skin Effect
- Common Digital Mistakes
- MC5 File Hierarchy

News In Preview

This issue features an article that describes creating a noise macro source for transient analysis that can easily create random data points for simulation. A second article describes modeling skin effect in resistors through the FREQ attribute and .DEFINE functions. The third article highlights mistakes users make when performing simulations with digital components. This article describes three common mistakes, why they occur, and what to do about them. The final article describes Micro-Cap's file hierarchy. This article is useful when running circuits that are in a directory other than the DATA directory.

Contents

Book Recommendations

- *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
- *Semiconductor Device Modeling with SPICE*, Paolo Antognetti and Giuseppe Massobrio McGraw-Hill, Second Edition, 1993. ISBN# 0-07-002107-4
- *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
- *SMPS Simulation with SPICE 3,* Steven M. Sandler, McGraw Hill, First Edition, 1997. ISBN# 0-07-913227-8
- *MOSFET Modeling with SPICE Principles and Practice,* Daniel Foty, Prentice Hall, First Edition, 1997. ISBN# 0-13-227935-5

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

Micro-Cap V Question and Answer

Question: I've just created a schematic, but whenever I try to enter Probe DC Analysis, I receive an error that states "Source Not Found". How do I get past this error?

Answer: The Probe analysis will always use the analysis limits that are set when you click on DC Analysis underneath the Analysis menu. If you haven't entered DC Analysis yet, the Input 1 text field is set to NONE. Therefore, the Probe analysis is looking for a source with the PART attribute of NONE. Go to the Analysis menu and click on DC Analysis. Specify the Input 1 text field with the PART attribute of a source that exists in your schematic and set the Input 1 Range to the range that you would like the source to be swept over. Hit the F3 key to exit the analysis. Go to the Analysis menu again and click on Probe DC Analysis, and it should run the analysis. This holds true for AC and transient analysis also. If you would like to change the frequency range or the time range of the Probe analysis, you must first enter the standard AC Analysis or Transient Analysis options first.

Question: When I step a parameter, how can I see which waveform goes with which step?

Answer: After a stepping operation, multiple waveforms will be available on the screen. To view which waveform belongs to which step, you need to invoke Cursor mode. This can be done by choosing Cursor, under the Mode selection of the Options menu or by pressing the F8 hotkey. The title of the plot will appear with the stepped parameter value of the waveform that the cursor is currently on. The title, when only stepping one parameter, will look similar to this:

```
PRLC.CIR  Temperature=27  R1.Value=50
```

where the cursor is currently on the waveform produced when R1 was 50 ohms. To shift the cursor to the other steps, press the Up or Down Arrow cursor key.

Question: I have five transistors that reference the same model statement. The BF parameter in the model statement has a LOT tolerance specified. The LOT should provide absolute tracking, but the five transistors are not tracking each other at all during a Monte Carlo analysis. What is happening?

Answer: The LOT tolerance does provide absolute tracking when multiple components reference the same model statement. However, if you go to Global Setting under the Options menu, the PRIVATEANALOG option will be enabled. This option forces all analog component to have their own model statement, thus nullifying LOT's absolute tracking effect. To have absolute tracking, PRIVATEANALOG must be disabled.

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 an icon or a menu item.

Rotating a Component with the Mouse or Spacebar
The easiest way to rotate a component is through the mouse or spacebar instead of the Rotate menu item. To rotate a component upon initial placement in the schematic:

1) Choose the component that is to be placed.
2) Click and hold down the left mouse button to place the component on the schematic.
3) Hit the right mouse button or the spacebar to rotate the component through its eight possible orientations while holding the left mouse button down.

To rotate a component that has already been placed in the schematic:

1) Enable Select mode.
2) Click on the component using the left mouse button and hold the left mouse button down. The component should change to its select color.
3) Hit the right mouse button or the spacebar to rotate the component through its eight possible orientations while holding the left mouse button down.

This feature is described in the Version 2 User's Guide in the "Creating a simple circuit" section of Chapter 3.

Drag Copy
The drag copy feature is a simplified method to performing a copy and paste. It uses the following procedure:

1) Select a circuit object or region to be duplicated.
2) Press CTRL and drag on any item in the selected area. Once the drag operation has begun, the CTRL key may be released.
3) Release the left mouse button to place the copy.

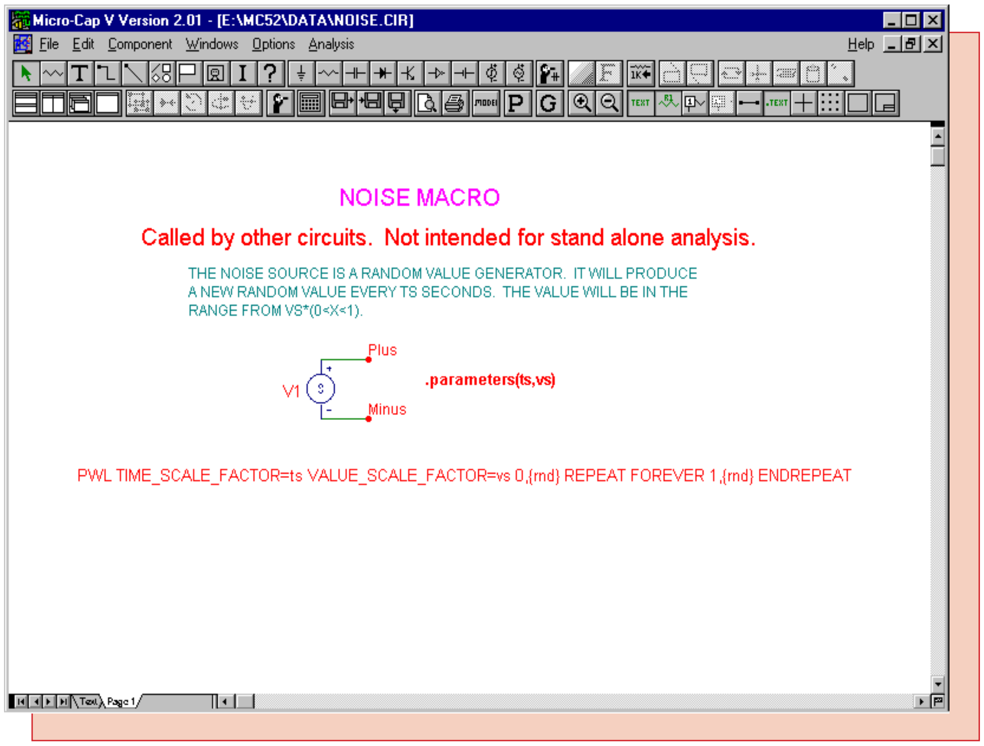
The original objects are left in the same place, but the copy is dragged along with the mouse. The part names are all incremented in this operation, but everything else will remain the same (with the possible exception of grid text depending on whether Text Increment is enabled in the Preferences dialog box). This feature may be used with any element in the schematic such as components, wires, graphic objects, flags, or text. CTRL+Z will undo a drag copy operation.

This feature is described in the Version 2 Reference Manual in the "Drag Copying" section of Chapter 2 and in the Version 2 User's Guide in the "Drag Copying" section of Chapter 3.

Noise Source Macro

The difficulty in creating noise sources for transient analysis has been to produce the large amounts of random data needed for transient analysis. Hundreds, if not thousands, of data points would have to be placed into a user source, table source, or the independent SPICE PWL source for a noise generator to be truly effective. However, the new REPEAT and scale features available for the PWL source, used in conjunction with the RND operator, simplify this process.

A single data point in a PWL source is specified as (<tin>,<in>) where <tin> is a time value and <in> is either a voltage or a current value depending on the source. The REPEAT feature for the PWL source lets you enter one or more data points between REPEAT and ENDREPEAT keywords. These data points will then be repeated for a user specified number of times. The scale feature lets you enter one value that will scale the <tin> value for all data points and another value that will scale the <in> value for all data points. Finally, the RND operator will calculate a random number between 0 and 1 each time it is specified. Figure 1 displays a macro for a noise source.


Micro-Cap V Version 2.01 - [E:\MC52\DATA\NOISE.CIR]
File Edit Component Windows Options Analysis
Help
Text
Page 1
V1
Plus
Minus

NOISE MACRO

Called by other circuits. Not intended for stand alone analysis.

THE NOISE SOURCE IS A RANDOM VALUE GENERATOR. IT WILL PRODUCE
A NEW RANDOM VALUE EVERY TS SECONDS. THE VALUE WILL BE IN THE
RANGE FROM VS*(0<X<1).

.parameters(ts,vs)

PWL TIME_SCALE_FACTOR=ts VALUE_SCALE_FACTOR=vs 0,{rnd} REPEAT FOREVER 1,{rnd} ENDREPEAT

Fig. 1 - Noise Source Macro

The noise source macro is created from a SPICE independent source. This source can be found in the Waveform Sources section with the name 'V'. The macro has two parameters, ts and vs. The ts parameter scales all of the time values in the data points, and the vs parameter scales all of the voltage values in the data points. The VALUE attribute for this source is defined as:

PWL TIME_SCALE_FACTOR=ts VALUE_SCALE_FACTOR=vs 0,{rnd} REPEAT FOREVER 1,{rnd} ENDREPEAT

The PWL keyword defines this source as a piecewise linear source. Only one data point has been defined in the REPEAT-ENDREPEAT loop. This data point, 1,{rnd}, will produce a random data point every ts seconds. The voltage of the data point will be vs*RND, where the value will be in the range of vs*(0 < x < 1). The loop has been specified to run forever, so it will produce random data points for as long as the simulation is specified for. The TIME_SCALE_FACTOR and VALUE_SCALE_FACTOR define the scale values for the PWL.

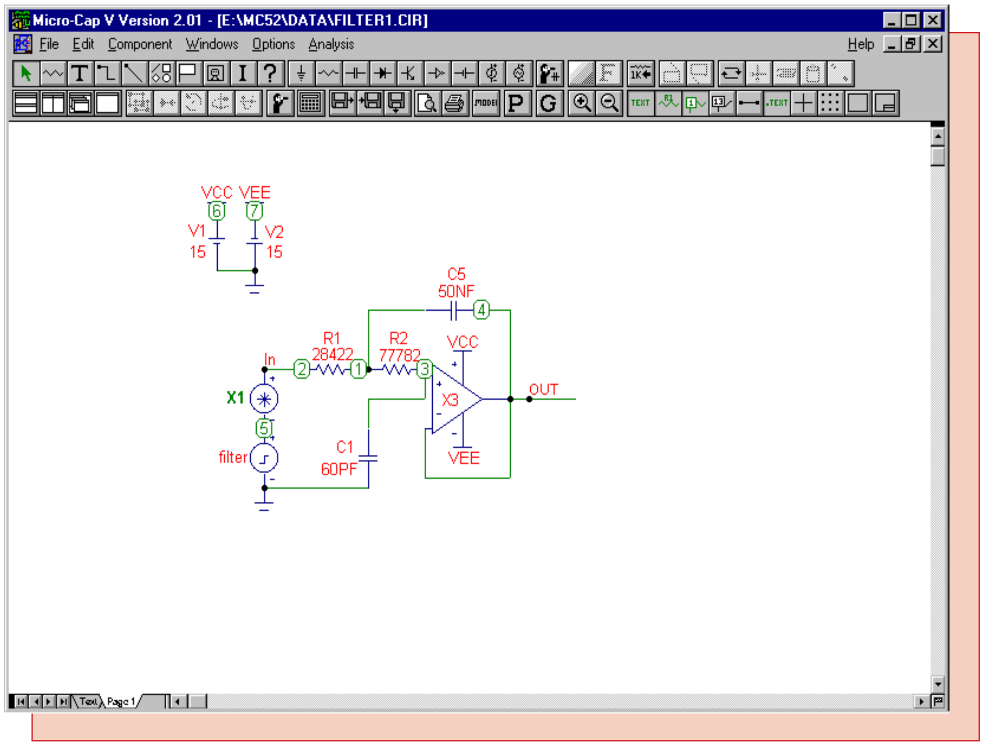

Micro-Cap V Version 2.01 - [E:\MC52\DATA\FILTER1.CIR]
File Edit Component Windows Options Analysis Help
VCC VEE
6 7
V1 15
V2 15
C5 50NF
4
R1 28422
R2 77782
VCC
In
2 1 3
X1
X3
OUT
5
filter
C1 60PF
VEE
Text Page 1

Fig. 2 - Noise Test Circuit

Figure 2 displays the test circuit for the noise source macro. This circuit is a modification of the FILTER.CIR file that comes with MC5. The circuit simulates an active Chebyshev filter that has an injected noise source inserted on top of the input pulse source.

The noise source macro has the PART attribute X1, and its VALUE attribute is defined as:

Noise(10u,100m)

The ts parameter is defined as 10u, and the vs parameter is defined as 100m. The noise source will produce random values in the range of 0 to 100mV every 10us.

Figure 3 shows the analysis results for this circuit. The V(In) waveform is a combination of the pulse source and the noise source. Zooming in on this waveform would display a random piece-wise linear point every 10us. The V(Out) waveform is the output of the filter. As can be seen in the analysis, the Chebyshev filter is attenuating the higher frequency contributions of the noise source.

An equivalent current noise source can be generated with the current SPICE independent source. This source is called 'I' and is also available in the Waveform Sources section. It would use the exact same syntax as in this case.

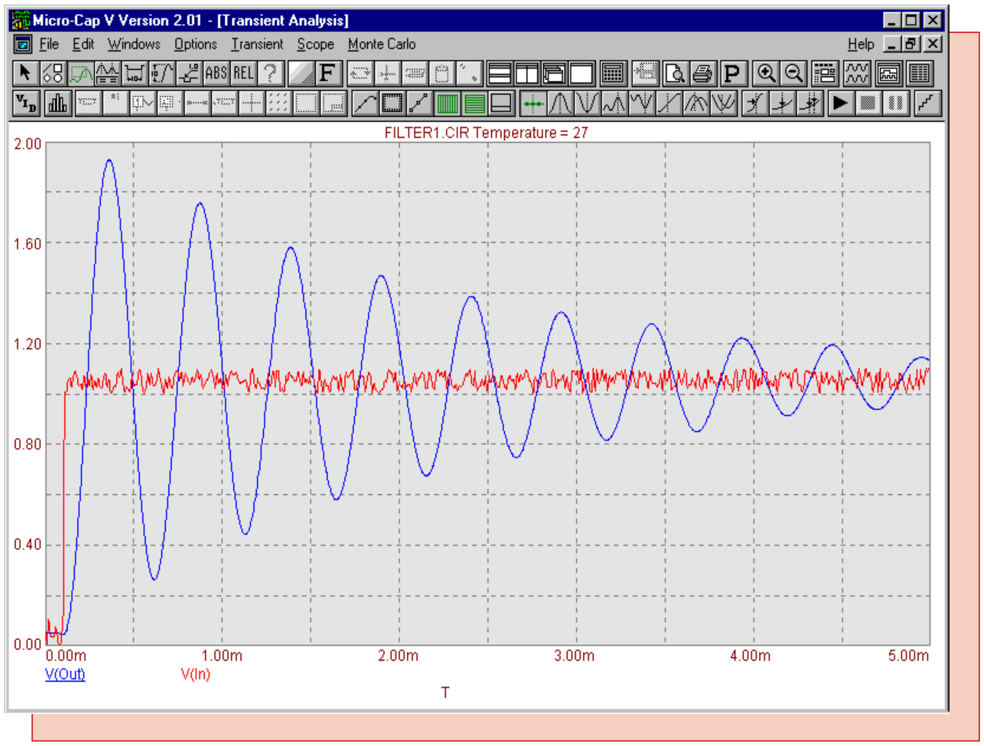


Fig. 3 - Noise Macro Test Analysis

The noise source macro will produce an average value of (vs * .5). Offsets may easily be added into the PWL source to shift the voltages. To shift the random values so that it produces an approximately equal number of positive and negative values, add a -.5 offset to the data points in the PWL source. The VALUE attribute for the PWL source would now look like this:

PWL TIME_SCALE_FACTOR=ts VALUE_SCALE_FACTOR=vs 0,{rnd-.5} REPEAT FOREVER 1,{rnd-.5} ENDREPEAT

The voltages produced with this statement would be in the range of vs*(-.5 < x < .5).

There are a couple of situations to be careful of when using this source. If you run a long transient simulation, but have a very small ts value, then it will slow the simulation down as it will have to produce at least one data point every ts seconds. The second situation would be in running an AC analysis with this source in the schematic. This source has no AC characteristics, so it will act as a short circuit during the small signal analysis. However, to produce the small-signal circuit, AC analysis needs to run a bias point calculation first. The noise source will produce a random value for the bias point operation each time the circuit is simulated. If the circuit is sensitive to the noise source, it may alter the AC analysis results every run. In most cases though, the noise source should only have a negligible effect on the AC analysis results.

Modeling Skin Effect

In order to model skin effect, a resistor must be able to vary with frequency during AC analysis simulations. To simulate this, the FREQ attribute of the resistor must be defined. The VALUE attribute of the resistor is not effective for this type of modeling in AC analysis. The basic premise of AC analysis is that it is a small signal or linear analysis. Micro-Cap calculates the operating point and then linearizes each device about the operating point. Therefore, if the VALUE attribute of a resistor is defined as '10*f + 100', the actual resistance during the AC analysis simulation will be 100 ohms. This occurs because the resistor is linearized at the DC operating point when the frequency is equal to 0. The FREQ attribute, however, will override any specification in the VALUE attribute during a small signal simulation and lets the resistor vary with frequency during an AC run. Capacitors, inductors, and nonlinear function sources also have the ability to vary with frequency in AC analysis.

Skin Effect Definition
Skin effect is the phenomenon where the apparent resistance of a wire increases as the frequency increases. At DC, the charge carriers have an even distribution throughout the area of the wire. However, as the frequency increases, the magnetic field near the center of the wire increases the local reactance. The charge carriers subsequently move towards the edge of the wire, decreasing the effective area and increasing the apparent resistance.

The area through which the charge carriers flow is referred to as the skin depth. The skin depth is frequency dependent, and we will use it to calculate the AC resistance by implementing its transfer function in the FREQ attribute. The first step is to compute the transfer function for the AC resistance.

The AC and DC resistances are related through the effective area of the wire as follows. Refer to Figure 4 for a graphical representation of the variables used.

AC resistance / DC resistance = DC area / AC area

AC resistance = (DC area / AC area) * DC resistance

DC area = PI * R^2

AC area = PI * R^2 - PI * r^2

r = R - e

e = (3.16e3 / (2 * PI)) * (res / (f * k_m))$^{1/2}$

AC resistance = (PI * R^2 / (PI * R^2 - PI * (R - e)^2)) * DC resistance

where e is the skin depth, res is the resistivity, f is the frequency, and k_m is the relative permeability. The AC resistance has now been reduced to depending on only the frequency variable and constants. It may now be implemented in the FREQ attribute of the resistor. The skin depth equation presented here is only valid for higher frequency analysis, so we will analyze it at 50kHz and higher. The next step is to implement these equations for use with a resistor. The best method for doing this is through .DEFINE user functions.

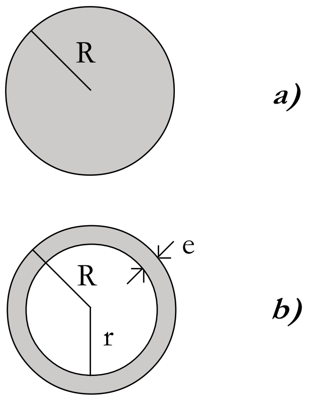


Fig. 4 - a) DC Representation of a Wire
b) AC Representation of a Wire

The .DEFINE user functions are a method for a user to implement his own operators. These functions can be local to the circuit if they are placed in the text area or on the schematic, or they may be global if they are placed in the DEF.MC5 file. The DEF.MC5 file can be accessed through the User Definitions item on the Options menu. The skin effect equations that were just derived can be implemented with the following two .DEFINE functions.

```
.define  skin(DCres,res,km,rad)  ((PI*rad*rad)/((PI*rad*rad)-PI*(rad-em(res,km))**2))*DCres
.define  em(res,km)  503.3*(sqrt(res/(km*f)))
```

The skin function has four parameters. DCres is the DC resistance, res is the resistivity of the material, km is the relative permeability, and rad is the radius of the wire. The skin function calls the em function and passes it two parameters: res and km. The em function calculates the skin depth.

The schematic used to test the skin effect is shown in Figure 5. This test schematic actually consists of three separate circuits. All of them model the resistance of a 1 meter, AWG No. 22 wire. The top circuit models the aluminum material characteristics with skin effect. The middle circuit models the copper material characteristics with skin effect. The bottom circuit models the DC resistance of the copper wire in order to compare an ideal resistor to the skin effect resistor. The inductor and capacitor were given nominal values since we are trying to display the effect that skin effect has in a simulation.

To use the skin function with the resistor, just declare the skin operator with its four parameters in the FREQ attribute of the resistor. For now, place in a nominal value for the VALUE attribute of the resistor. It won't have any affect in AC analysis when the FREQ attribute is defined, but a value must be present. Later in this article, we will cover a method to use skin effect in transient analysis. The FREQ attributes for R1 and R2 are:

```
skin(.0868,2.83e-8,1,.322m)  ;  for  R1
skin(.0537,1.75e-8,1,.322m)  ;  for  R2
```

The FREQ attribute for R3 is left blank as the resistor is modeled as an ideal resistor. The VALUE attribute is defined as .0537 ohms which is the DC resistance for copper.

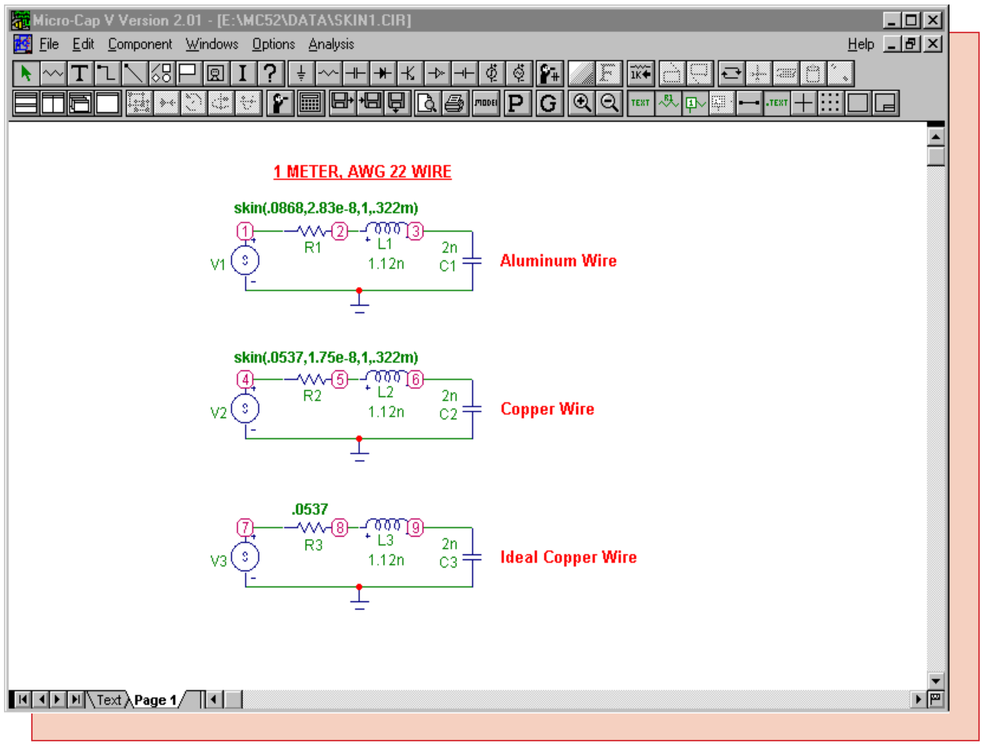

Micro-Cap V Version 2.01 - [E:\MC52\DATA\SKIN1.CIR]
File Edit Component Windows Options Analysis Help
1 METER, AWG 22 WIRE
skin(.0868,2.83e-8,1,.322m)
R1
L1
1.12n
2n
C1
V1
Aluminum Wire
skin(.0537,1.75e-8,1,.322m)
R2
L2
1.12n
2n
C2
V2
Copper Wire
.0537
R3
L3
1.12n
2n
C3
V3
Ideal Copper Wire
Text
Page 1

Fig. 5 - Skin Effect Test Circuit

Two AC analysis simulations were performed on this schematic. The first simulation compares the gain of the copper wire modeled with skin effect resistance versus the gain of the copper wire modeled with ideal resistance. As shown in Figure 6, the higher resistance produced by the skin effect produces a lower gain than the ideal resistor, particularly at higher frequencies. The skin effect resistance also negates the resonance of the reactive components in this simulation.

The second simulation compares the apparent resistance between an equivalent size wire made of aluminum and one made of copper. A curve of resistance vs. frequency can be plotted by specifying R(R1) fro the Y Expression field. As expected, Figure 7 shows that the resistance from the aluminum wire is higher than the copper wire throughout the frequency sweep. This makes sense since both the DC resistance and the resistivity of the aluminum wire is higher.

For AC analysis, the .DEFINE functions will produce precise values in determining the AC resistance. However, the FREQ attribute has no effect in transient analysis. Therefore, the .DEFINE functions need to be modified to run in transient analysis. The way to modify these functions is to add in a fifth parameter that specifies the frequency at which the circuit is running. The two new .DEFINE functions are as follows.

```
.define skint(DCres,res,km,rad,freq) ((PI * rad * rad) / ((PI * rad * rad)-PI *
+ (rad - emt(res,km,freq))**2)) * DCres
.define emt(res,km,freq) 503.3 * (sqrt(res/(km * freq)))
```

The only difference between these functions and the previous functions, besides giving these unique names, is that the f variable has now been changed to a parameter named freq to represent the frequency of operation and that variable has been added to the emt function.

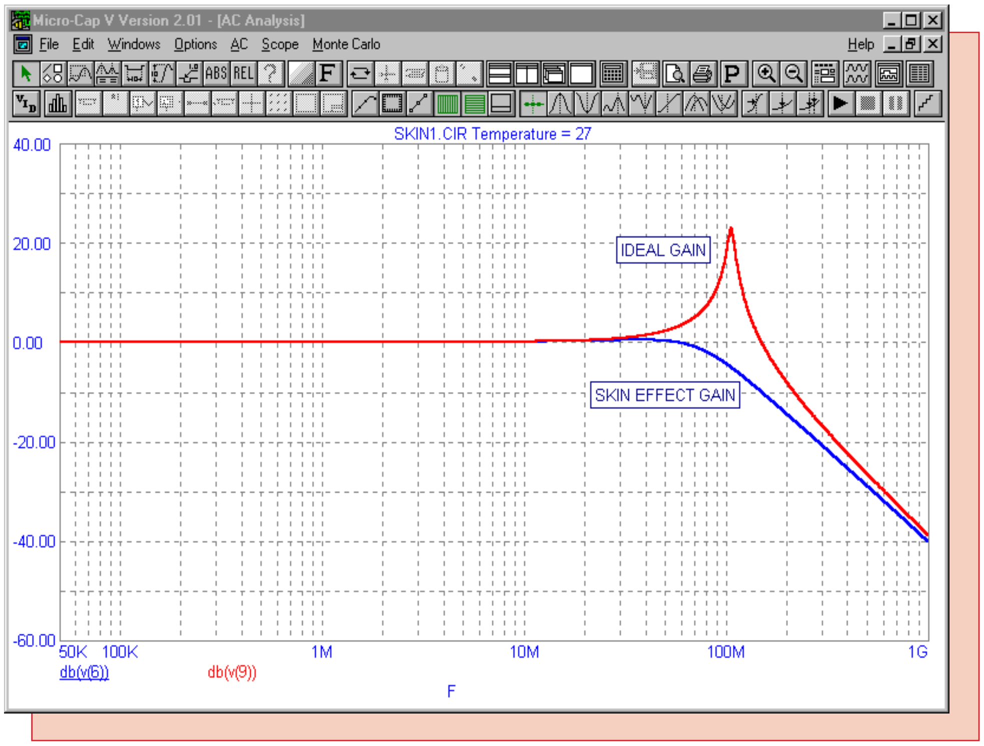


Fig. 6 - Skin Effect Gain vs. Ideal Gain

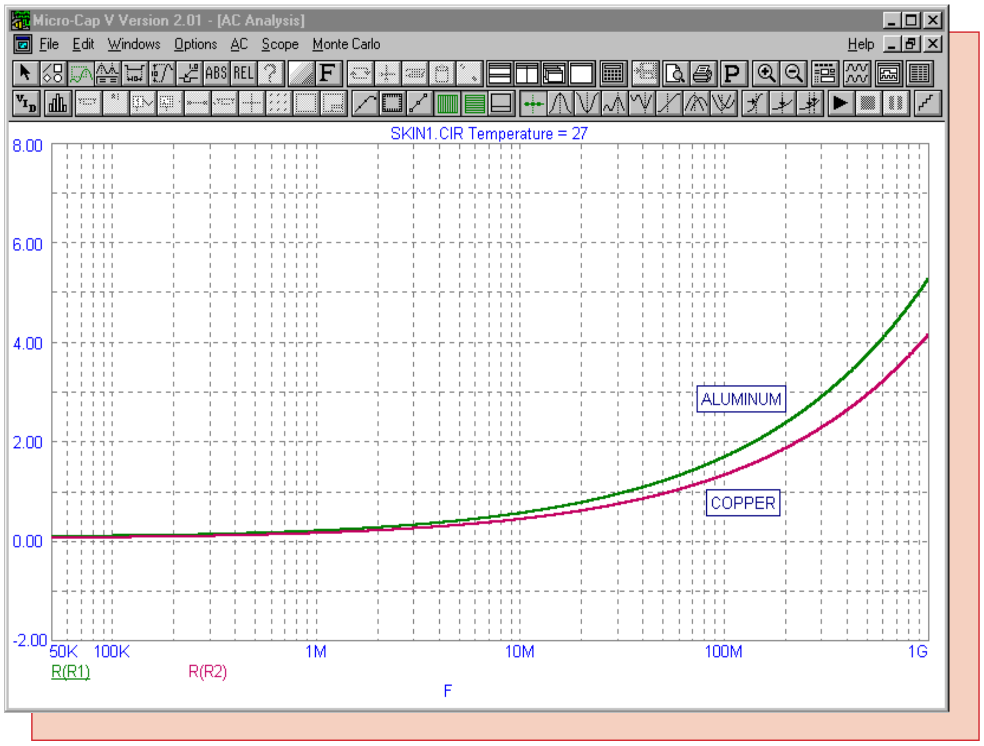


Fig. 7 - Apparent Resistance of Copper vs. Aluminum

For the circuit of Figure 5, the VALUE attributes for R1 and R2 are specified as:

skint(.0868,2.83e-8,1,.322m,fop) ; for R1
skint(.0537,1.75e-8,1,.322m,fop) ; for R2

The VALUE attribute for the V1, V2, and V3 voltage sources are all specified as:

AC 1 sin 0 1 fop

This sets the sources to an AC magnitude of 1V for AC analysis and as a sine wave for transient analysis that has a 0V DC offset, a 1V amplitude, and a frequency defined with a parameter called fop. Notice that the frequency parameter for the skint functions have also been defined with the fop parameter. This parameter represents the frequency of operation. Using a .DEFINE statement in the text area or the schematic area such as:

.DEFINE fop 10Meg

will set the frequency of operation for the voltage sources and the VALUE attributes of the R1 and R2 resistors to 10 MHz. Using a parameter in this manner, makes changing the frequency of the circuit simple since only one edit needs to be made instead of five. Figure 8 displays the transient analysis results of the outputs of the skin effect copper wire and the ideal copper wire. The drastic difference occurs because the frequency of operation chosen, 10 MHz, is in the resonance area for the ideal copper wire.

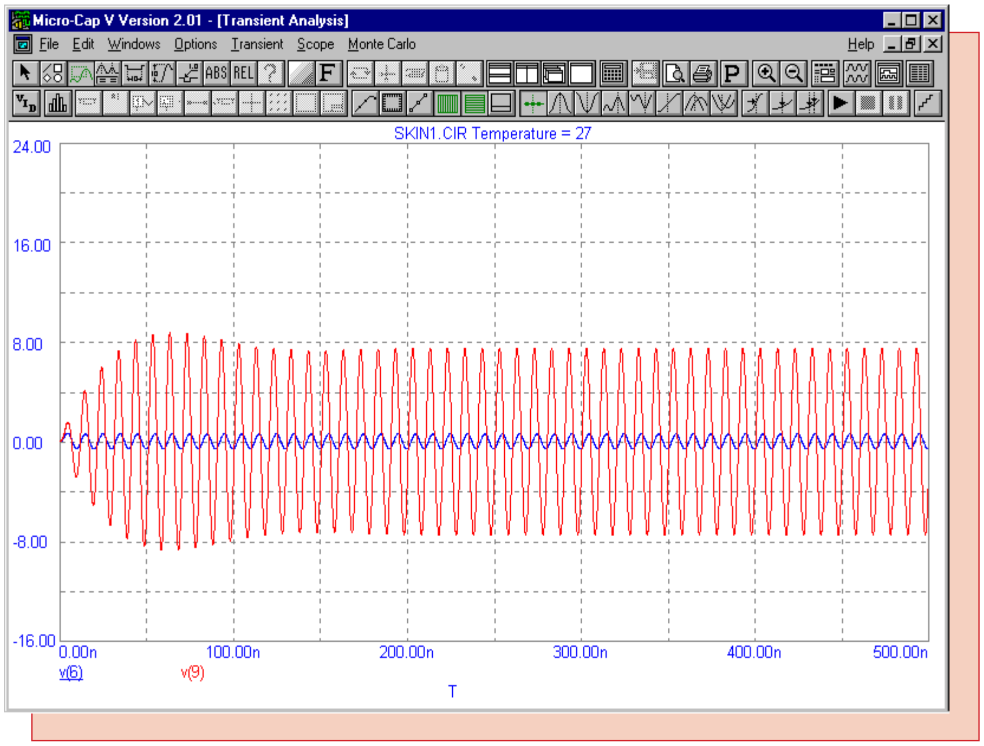


Fig. 8 - Transient Results of Skin Effect vs. Ideal for Copper Wire

Not Initializing Flip-Flops or Latches

One common mistake in using digital flip-flops or latches is the failure to initialize the part. For many configurations, not initializing a flip-flop or latch won't cause a problem, but for some toggle and counter configurations, it will produce undesired results.

The circuit in Figure 9 has two JK flip-flops that are set up to run in toggle mode having both the J and the K inputs set to 1. Both flip flops share the same clock, and both have Fixed Digital components defined as 1 at the Prebar, J, and K inputs. The only difference between the two flip-flops is that the top flip-flop has its Clrbar input fixed to the 1 state, and the bottom flip-flop has a digital stimulus going into its Clrbar input. This digital stimulus has its Command attribute specified as:

0 0 55ns 1

which produces a 0 state for the first 55ns of the simulation, and then is at a 1 state for the rest of the simulation. This will initialize the Q2 output of the flip-flop to the 0 state. Figure 10 displays the transient results of this schematic for a run of 1us. The bottom flip-flop, which has been initialized to 0, is toggling correctly as shown by waveforms D(Q2) and D(Q2B). The outputs of the top flip-flop, which have not been initialized, are at an X, unknown, state for the entire simulation as shown by waveforms D(Q1) and D(Q1B). The reason for this is that the default initialized state of a flip-flop or a latch is the X state. When the negative edge of the clock occurs, the flip-flop is toggled, but toggling an X state can only produce the same X state by definition.

A way to initialize all flip-flops and latches in a schematic without having to place a set or clear pulse on each part is to set the DIGINITSTATE parameter that is available in the Global Settings. A 0 sets all flip-flops and latches to the 0 state, a 1 to the 1 state, and a 2 to the X state.

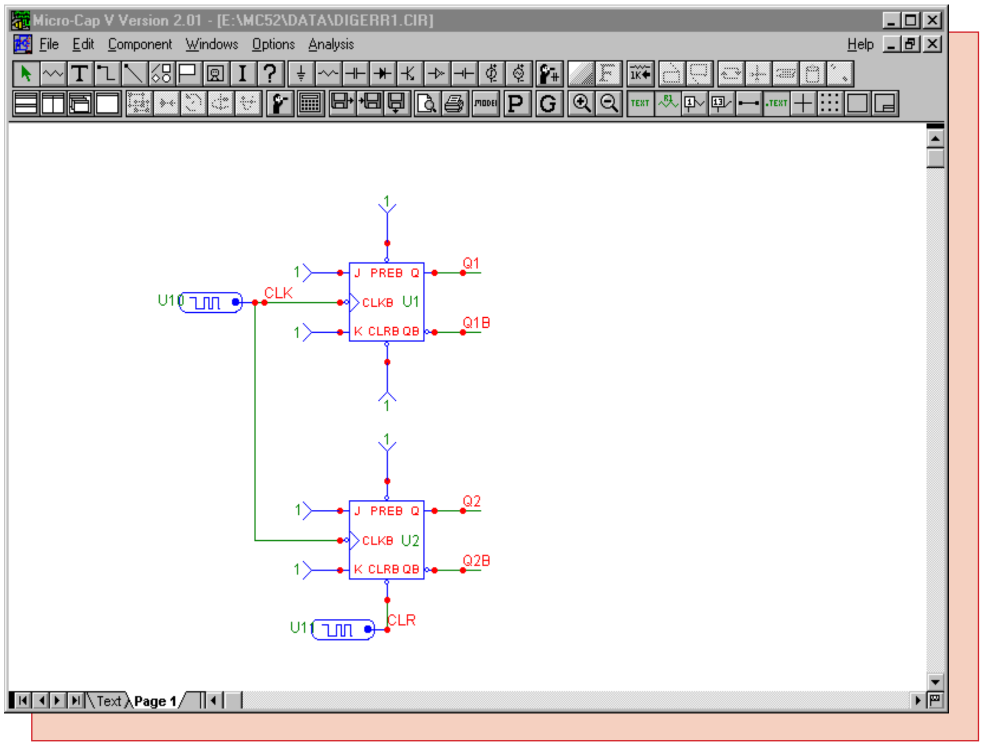

Micro-Cap V Version 2.01 - [E:\MC52\DATA\DIGERR1.CIR]
File Edit Component Windows Options Analysis Help
1
Q1
J PREB Q
CLKB U1
K CLRB QB
Q1B
U1
CLK
1
1
1
1
Q2
J PREB Q
CLKB U2
K CLRB QB
Q2B
U1
CLR
Text Page 1

Fig. 9 - Initialization Circuit

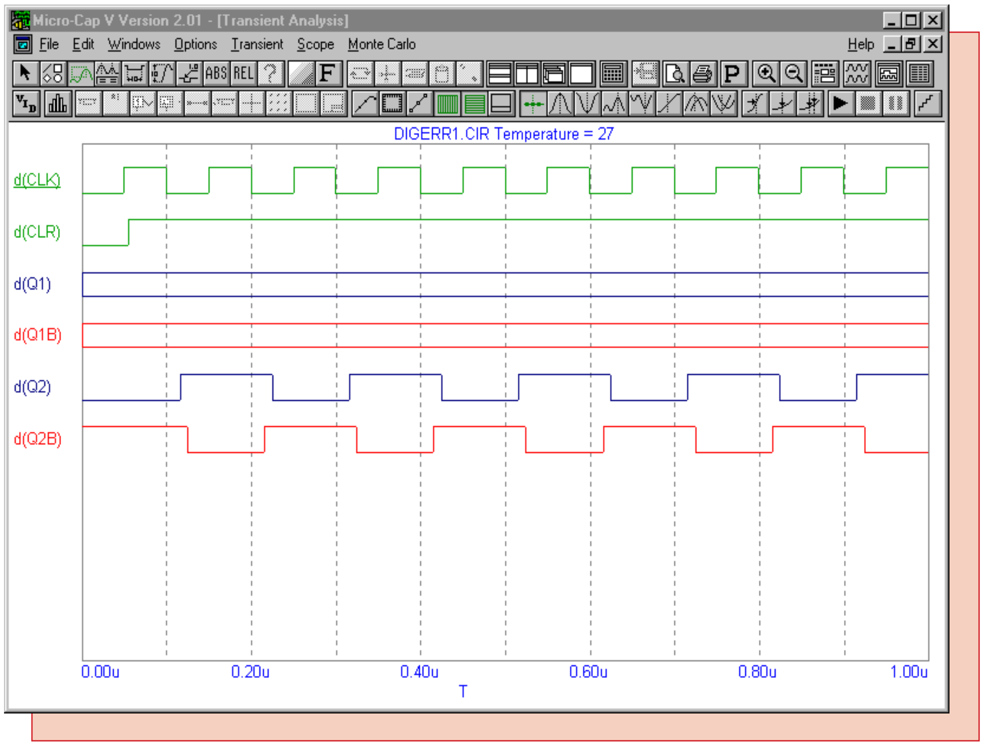

Micro-Cap V Version 2.01 - [Transient Analysis]
File Edit Windows Options Transient Scope Monte Carlo Help
DIGERR1.CIR Temperature = 27
d(CLK)
d(CLR)
d(Q1)
d(Q1B)
d(Q2)
d(Q2B)
0.00u
0.20u
0.40u
0.60u
0.80u
1.00u
T

Fig. 10 - Initialization Analysis

Using Open Collector Parts without Pullup Components
Another common mistake occurs when using an open collector output as the input to another digital device. When used in conjunction with other digital devices, an open collector part needs to have a digital pullup component placed on each of its outputs.

Figure 11 displays a schematic that has a digital stimulus feed two 7405 open collector output inverters. Each of these open collector outputs is then passed on as the input to a 7404 inverter, which has standard outputs. The only difference between the top section and the bottom section is that the bottom section has a digital Pullup component connected to the output of its 7405. The pullup component has its I/O MODEL attribute defined as IO_PULLUP. This I/O model is then defined in the text area as:

.MODEL IO_PULLUP UIO (DRVH=1K DRVL=1Meg)

This model is setup assuming that only digital components will be connected to the node. Otherwise, specific AtoD and DtoA subcircuit models would need to be created for the I/O model. Whenever the node it is connected to is 1, it models a 1 kohm pullup resistor, but when the node is 0, it is essentially an open circuit. Since the output of the inverter is open collector, the pullup is supplying the driving impedance whenever the output is high. Figure 12 displays the results of a transient analysis for this circuit. The waveform D(OUT2) is as expected. The waveform D(OUT1) has an X state whenever a 0 state is expected. This is because there is no digital pullup on the output of the 7405. That 7405 output is unable to drive the 7404 whenever the output is 1 due to the high impedance of the node and this causes the unknown state at OUT1.

In setting DRVH for the I/O model, make sure that the impedance won't dominate any other outputs connected to the node and that it is lower than the value set for DIGDRVZ in the Global Settings.

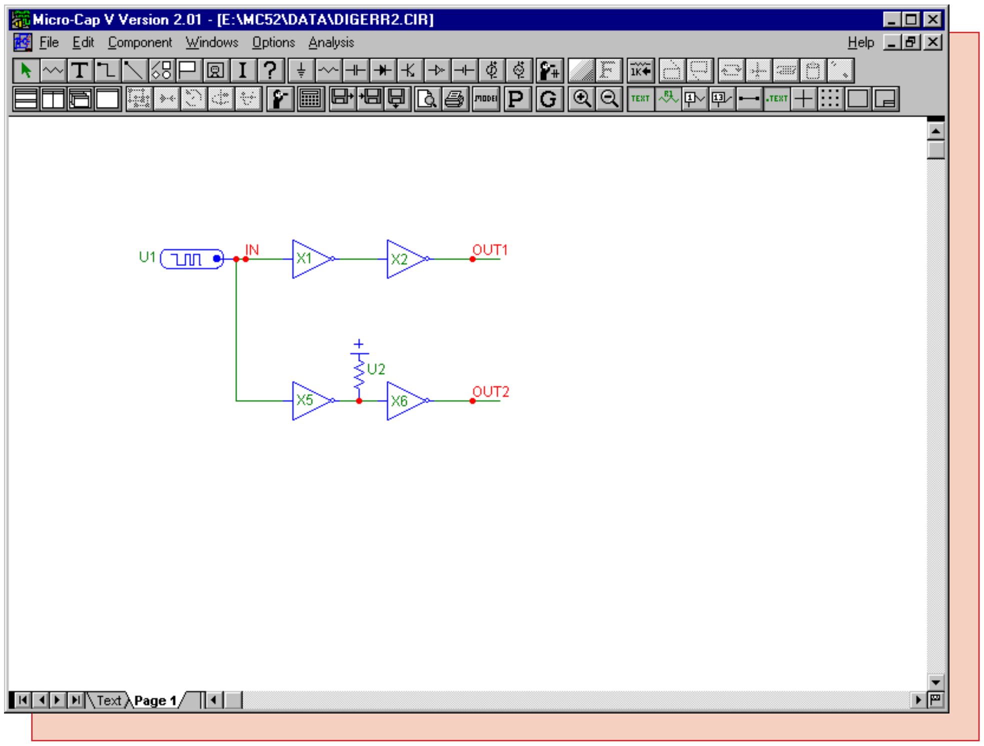

Micro-Cap V Version 2.01 - [E:\MC52\DATA\DIGERR2.CIR]
File Edit Component Windows Options Analysis Help
U1
IN
X1
X2
OUT1
U2
X5
X6
OUT2
Text
Page 1

Fig. 11 - Open Collector Circuit

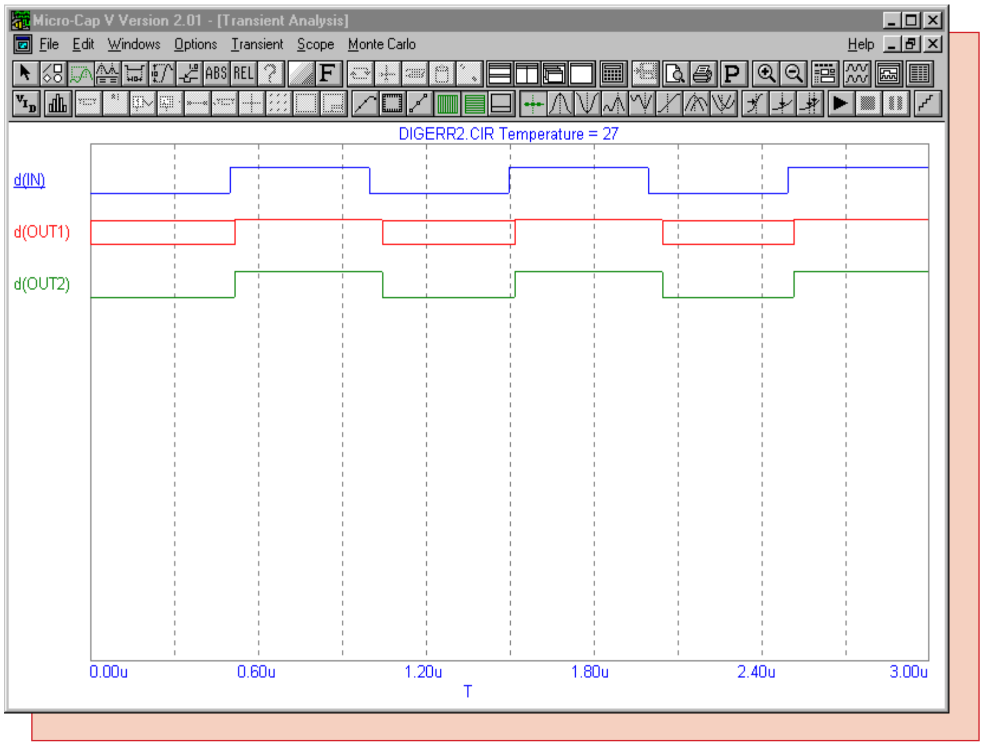

Micro-Cap V Version 2.01 - [Transient Analysis]
File Edit Windows Options Transient Scope Monte Carlo Help
DIGERR2.CIR Temperature = 27
d(IN)
d(OUT1)
d(OUT2)
0.00u
0.60u
1.20u
1.80u
2.40u
3.00u
T

Fig. 12 - Open Collector Simulation

Rise and Fall Conflicts in Mixed Mode Simulation

When running a mixed mode simulation, there are transition states produced between the 0 and 1 states. These states are the R (rise) and F (fall) states. These states signify the uncertainty of the transition. For example, standard TTL components are guaranteed to be at a 0 state below .8V and at a 1 state above 2V. Somewhere in this 1.2V gap is the actual transition between the 0 and 1 states. The R and F states represent the digital state in this voltage gap. They imply that the digital node has transitioned from 0 to 1, for a rise state, or from 1 to 0, for a fall state, at some point during this state, though the exact time is unknown.

This uncertainty can cause conflicts with edge triggered flip-flops. For a positive edge triggered flip-flop, if the D input is at an R or F state when the positive edge of the clock occurs, the outputs of the flip-flop will transition to an X state. The reason for the X state output is that the R and F states represent a state where the node may be either 0 or 1 at any given time so the output must be unknown since the input is not specifically known.

Figure 13 displays a schematic that produces this conflict. The left circuit is identical to the right one except for one attribute. The IO_LEVEL attribute on the right flip-flop has been set to 2 instead of the default 0. This forces the flip-flop on the right to use the level 2 I/O models that only produce 0 and 1 states, so that there will be no R and F conflicts. The left flip-flop uses the default level 1 I/O models.

Figure 14 displays the transient results for this schematic. The D(Q2) and D(QB2) waveforms produce the expected results due to the fact that they are using the level 2 I/O models. The D(Q1) and D(QB1) waveforms use the level 1 I/O models and upon the first positive clock edge, the outputs transition from a 0 to an X. At this edge, the clock's transition is 0-R-1. When the clock turns to an R state, the output is then set to an R state. This R state then goes through the feedback to the D input. When the clock finally hits its 1 state, the D input is now R, and the output is then set to X.

Unknown states flag areas of the simulation where there may be a conflict and that the user may need to look into more. Conditions that may cause these problems are long rise and fall times of analog pulses and undesired propagation delays on a feedback loop. In this circuit, changing the rise time in the V2 pulse source from .1us to .05us will also produce the expected results. However, the conflicts are not always desirable in all simulations. Individual digital primitives may have their I/O model levels changed through their IO_LEVEL attribute. The default 0 in this attribute forces the component to use the setting specified by the Global Setting DIGIOLVL. The DIGIOLVL parameter in the Global Settings can globally control the I/O model level. Setting the parameter to 1 uses the level 1 I/O models and 2 uses the level 2 I/O models.

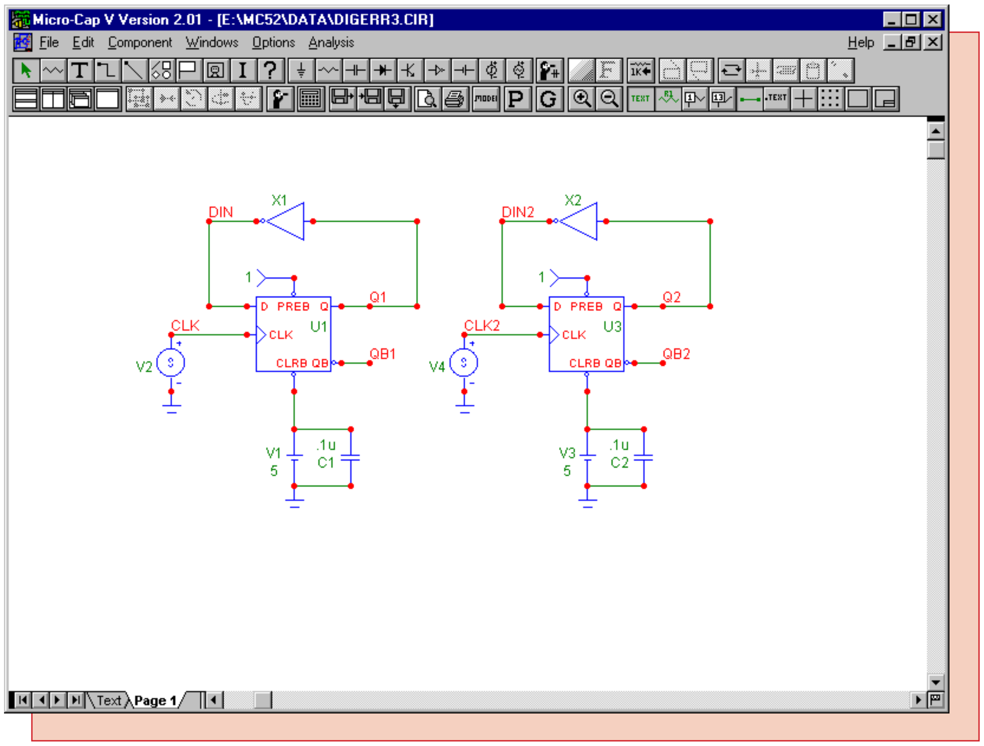

Micro-Cap V Version 2.01 - [E:\MC52\DATA\DIGERR3.CIR]
File Edit Component Windows Options Analysis Help
DIN
X1
1
D PREB Q
Q1
CLK
CLK U1
V2
CLRB QB
QB1
V1
5
.1u
C1
DIN2
X2
1
D PREB Q
Q2
CLK2
CLK U3
V4
CLRB QB
QB2
V3
5
.1u
C2
Text
Page 1

Fig. 13 - Rise and Fall Conflict Circuit

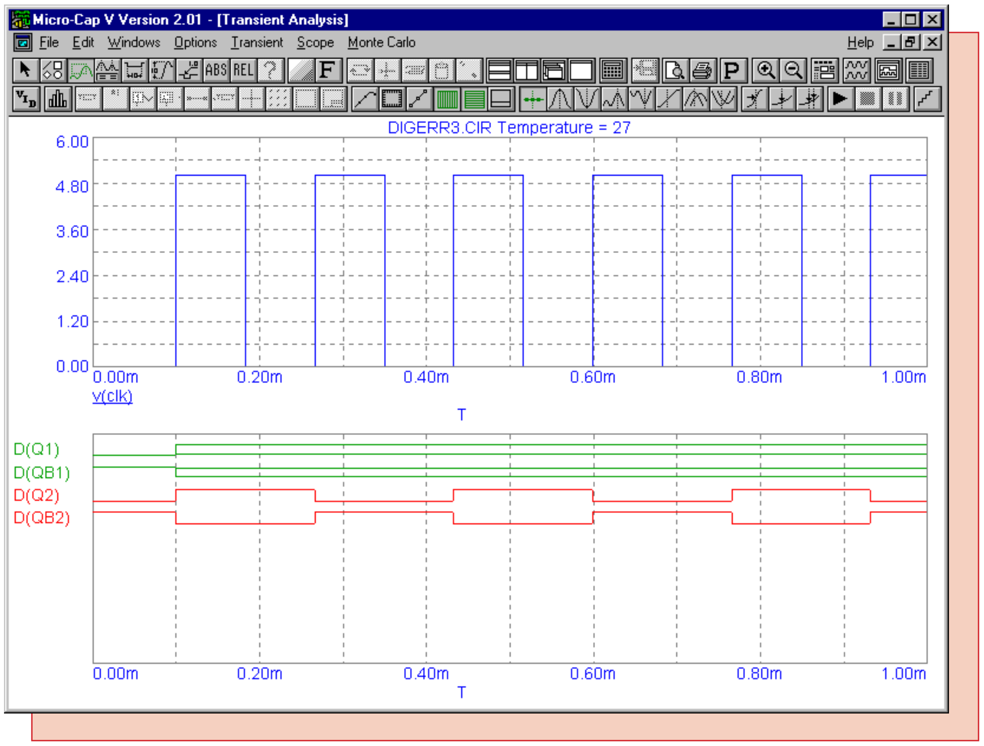


Fig. 14 - Rise and Fall Conflict Simulation

MC5 File Hierarchy

The file hierarchy is important to know when running circuits that are not in the DATA directory. The order and location that MC5 searches for files can have an impact on the circuit simulation because it can change the models used.

Environmental Variable

The environmental variable, MC5DATA, sets the default path that MC5 uses to find libraries and macros. This variable should be set to the path where the DATA directory resides. Multiple paths may be specified by placing a semicolon between paths. In Windows 3.X or Windows 95, this variable is set in the Autoexec.bat file by typing on a new line a statement such as:

```
SET MC5DATA=C:\MC5\DATA
```

In Windows NT 4.0, go to the Control Panel and double click on System. Click on the Environment tab. The MC5DATA variable would be set in the User Variables for Administrator area. The Variable and Value text fields would need to be filled in, and then click on the Set command button. The text fields would look similar to:

Variable	MC5DATA
Value	C:\MC5\DATA;C:\CIRCUITS

Library Searching Hierarchy

With a schematic, models are searched for in a specific order. The order is as follows:

1) Searches in the schematic or text area.
2) Searches for the file specified in the FILE attribute (if the device has one).
3) Searches any libraries specified by a .LIB statement.
4) Searches the libraries specified in the Nom.lib file.

The first instance of the model found will be the one used.

Adding a Model Globally

The method for adding a model globally is to simply place the library file name that the model is contained in into the Nom.lib file. The Nom.lib file is a text file that contains multiple .LIB statements. Any library listed in this file is automatically accessed whenever an analysis is run from any directory as long as the environmental variable is pointing to the location of the Nom.lib file. The Nom.lib file is found in the DATA directory, and may be opened through the standard File-Open menu command. After opening this file, add in a new .LIB statement on a new line. For example, if the library file Myparts.lib contains the parts to be added and is present in the DATA directory with the Nom.lib file, a new line would be added as follows:

```
.lib  "myparts.lib"
```

If the same library file is in a different directory, a path would also need to be included, such as:

```
.lib  "C:\Library\myparts.lib"
```

The basic nomenclature for libraries is that binary libraries created from the Model program must have the extension .LBR, and text files containing .Model and .Subckt SPICE models should have any extension but .LBR and .CIR. The .LIB extension is preferable for text files.

Adding a Macro Globally
There is only one rule for adding a macro globally. The macro circuit must be present in a directory that is referred to by the environmental variable. It is easiest to keep the macro circuit in the DATA directory, but macros may also be stored in a separate directory if the environmental variable declares it. For example, if the macro circuits are to be stored in a subdirectory called MACROS, the environmental variable should be as follows:

```
SET MC5DATA = C:\MC5\DATA;C:\MC5\MACROS
```

It may not be desired for all models and macros to be defined as global. A user may want to simulate each project in its own subdirectory and have models and macros that are only used for that project. While they can edit the environmental variable and Nom.lib each time they start a new project, that can be tiresome and difficult to go back to a previous project.

Adding a Model Locally
The best way to add a model locally is to place a .LIB statement into the text or schematic area of the circuit being simulated that specifies the local library file name. If the library is stored in the same directory as the circuit file, the .LIB statement would be as follows:

```
.lib  myparts.lib
```

otherwise a path would have to be specified also. Models may also be copied directly into the circuit itself. However, if you are simulating multiple circuits in a project, the .LIB method is better since editing the model would only require editing one file. Remember that a model in the text or schematic area takes precedence over one from a .LIB statement.

Adding a Macro Locally
There is no command needed for adding a macro locally. It only needs to be in the same directory as the circuit being simulated. This directory does need to be the active directory though. The way to view which directory is active is to invoke the Open File dialog box by clicking on Open under the File menu. The directory specified in this dialog box is the active directory. To change directories, simply open up a file from the desired directory. Opening up a file through the Recent Files list under the File menu does not change the active directory.

A path may be specified in the VALUE attribute of a macro. For example, when using the Amp macro, the VALUE attribute would typically appear similar to:

```
Amp(10)
```

However, to make sure that the circuit can always grab the correct macro no matter the active directory, the VALUE attribute may also be defined as:

```
C:\MC5\DATA\Amp(10)
```

MC5 would then search the DATA directory for the Amp macro circuit.

Product Sheet

Latest Version numbers

Micro-Cap V IBM/NEC	Version 2.01
Micro-Cap IV IBM/NEC/MAC	Version 3.04

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

Spectrum's Products

• Micro-Cap V IBM	$3495.00
• Micro-Cap V NEC	$3495.00
• Micro-Cap IV IBM	$2495.00
• Micro-Cap IV MAC	$2495.00
• Micro-Cap IV NEC	$2495.00

You may order by phone or mail using VISA or MASTERCARD. Purchase orders accepted from recognized companies in the U.S. and Canada. California residents please add sales tax.