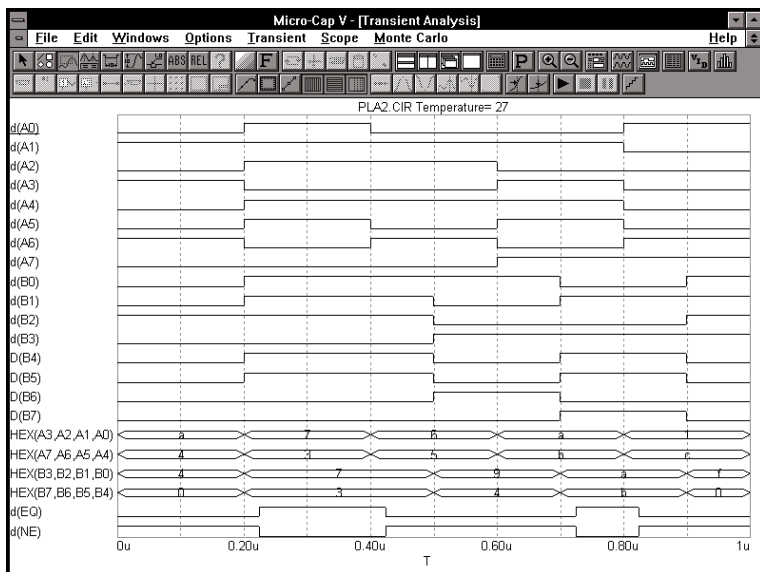


## PLA Modeling



Featuring:

- Accessing Libraries in MC5
- PLA Modeling
- Creating and Using a Macro
- Frequency Dependent Resistors
- MC5 and HP Printer Drivers

---

## News In Preview

In this newsletter, we begin to include articles about our latest generation of simulators, Micro-Cap V. The articles cover some of the most common questions to date about MC5 such as including new libraries for MC5 to access and the creation and use of the macro component. It also contains a warning about some of the new printer drivers from HP.

In the digital modeling area, we go through the process of creating a programmable logic array subcircuit by modeling the PAL16C1 from National Semiconductor. This article steps through the process of creating the subcircuit, giving it a shape and component definition, and using it within a schematic.

We also include a method for creating frequency dependent resistors by using the Laplace source. This article uses skin effect as an example. While this article uses MC4 graphics, it is entirely applicable to MC5.

## Contents

News In Preview .....	2
Book Recommendations .....	3
New Email Number .....	3
Latest Micro-Cap IV Update .....	3
Micro-Cap V Question and Answer .....	4
Micro-Cap III Tech Support .....	4
Accessing New Libraries .....	5
Modeling a Programmable Logic Array .....	8
Creating and Using a Macro .....	15
Frequency Dependent Resistors .....	21
HP Printer Driver Incompatibilities .....	26
Product Sheet .....	28

---

## Book Recommendations

- *Electronic Devices and Circuits Using MICRO-CAP II*, Richard H. Berube, Merrill 1992. ISBN# 0-02-309160-6
- *Electronic Devices and Circuits Using MICRO-CAP III*, Richard H. Berube, Merrill 1993. ISBN# 0-02-309151-7
- *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

## New Email Number

We can be reached through email at 103114.61@compuserve.com. Either questions or circuits may be sent through email. The circuit files are simply text files. To send circuit files, email the file as you would any other text file. One way to ensure that the circuit comes through well is to open the circuit file with any text editor, and then copy and paste the entire file into the email text field. Most circuits should come through fine with either method, but we have noticed that some email providers, such as AOL, substitute characters that make the file unreadable.

## Latest Micro-Cap IV Update

Micro-Cap IV Ver. 3.04 has just been released for the IBM, Mac, and NEC. For those of you who currently have MC4 V3, this is a free update. The main differences are in the Mac where a problem with the Model program on the Power PC was resolved and an algorithm was changed that speeds up the analysis by up to 30%.

---

## Micro-Cap V Question and Answer

Caller: In Micro-Cap V, what happened to the Model Editor?

Tech: The Model Editor still exists in MC5 and is very similar to its incarnation in MC4. It is not accessed in the same way as the Component and Shape Editors anymore. The way to call up the Model Editor is to click on Open under the File menu. In the Open File dialog box, you may either type the library name directly into the text field, or you may specify "Model Library (\*.LBR)" under the List Files of Type and choose one of the listed files. As in MC4, the files that may be opened in the Model Editor are those that have been created by the Model program.

Caller: I have made my own subdirectory to create and store my circuits in, but whenever I try to go into analysis, I get an error that there is no such file or directory as "NOM.LIB" or some other library. How do I get Micro-Cap to find the libraries?

Tech: The problem is that MC5 is looking only in the current directory for the libraries, and if the libraries aren't in this directory, then it will not be able to find them. The way to fix this is to use the environmental variable, MC5DATA. Add this line inside your Autoexec.bat:

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

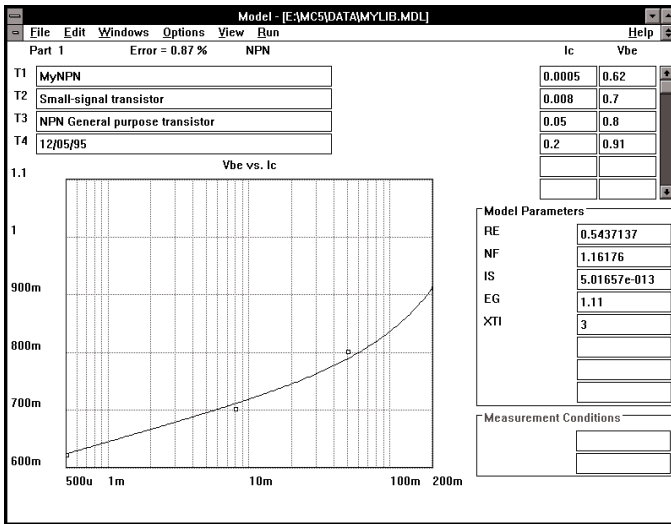
where C:\MC5\DATA is the directory the libraries are stored in. More than one directory may be specified for this variable, with the paths being separated by a semicolon.

## Micro-Cap III Tech Support

As of January 1, 1996, technical support for Micro-Cap III has been discontinued. Only emergency support will be granted for this program.

## Accessing New Libraries

One of the most asked question since the release of Micro-Cap V is how to access component models that have been created in the Model program. The following guide will also be helpful in adding new models and subcircuits from vendors, other simulators, or that the user has created.



**Fig. 1 - Mylib.mdl in the Model Program**

The figure above shows an NPN model, MyNPN, that has been created in the Model program and is contained in the file Mylib.mdl. While this file contains only one model, multiple parts and types may be stored in a single model file. This lets the user add to a library without having to manipulate multiple .mdl and .lbr files. Note that in the T3 text field, the text begins with NPN. This lets MC5 know that it is an NPN model and will list it in the model list when an NPN is placed in the schematic. For components with polarity, such as the BJT, JFET, and MosFET, this text field must begin with the corresponding N or P.

Mylib.mdl must be converted into a library that can be read from MC5. There are two options for doing this under the File menu: Create SPICE file and Create Model Library. Create SPICE file creates a text file that contains model statements for all of the parts that are in the .mdl file. Create Model Library creates a binary file that may only be read in the Model Editor. This binary file is only allowed the extension .lbr.

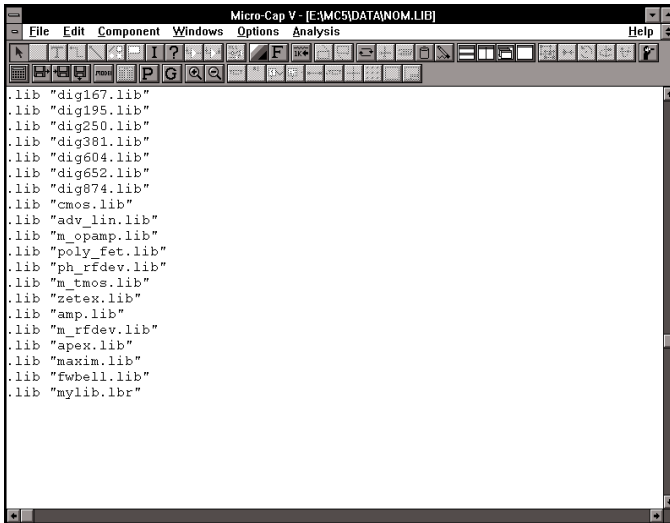
---

The procedure to access these models for simulation is the same for either the SPICE library or the Model library. In this case, we will create a Model library that is given the name Mylib.lbr.

In MC5, click on Open under the File menu, and open the file "Nom.lib". As shown in Figure 2, this is a text file that contains references to all of the libraries that come with MC5. When entering analysis, MC5 will search for models or subcircuits in this order: the schematic, the FILE attribute (for a subcircuit), any .LIB statements, and finally the default .LIB statement. The default .LIB statement references the Nom.lib file, so all libraries contained in this file will be searched for the specified model. The Nom.lib file also controls which models appear in the Model List inside the Attribute dialog box. On a new line, add a .lib statement such as:

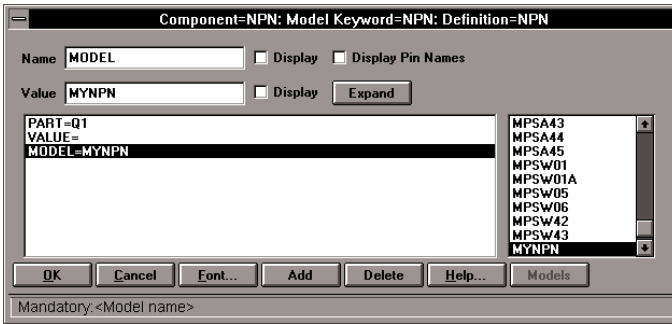
```
.lib "Mylib.lbr"
```

A path may be placed in front of the library name, otherwise the library must be stored in the same directory as the Nom.lib file. Save and close this file.



*Fig. 2 - The Nom.lib file*

The MyNPN model may now be accessed when entering analysis or when placing an NPN transistor into the schematic. To verify this, select the NPN device from the Analog Primitives/Active Devices section of the Component menu. Place the NPN into a schematic, and the Attribute dialog box will be invoked. However, MC5 has to generate a new library index first which will take a minute or two. Once the Attribute dialog box is up, click on the Models command button. This will display a list of all of the NPN models that are contained within the libraries named in the Nom.lib file. This list, shown in Figure 3, may be scrolled until the desired model is showing. Clicking on the model name will assign that model to the MODEL attribute, or the name may just be typed in directly into the MODEL attribute. The new model is now available for simulation.



*Fig. 3 - MyNPN model in the Attribute dialog box*

---

## Modeling a Programmable Logic Array

The programmable logic array is used to model a variety of logic devices. Each output consists of a gate which is chosen by the PLA type. The array is programmed by choosing which inputs are to be connected to each of the output gates. The programmable logic array may be used in conjunction with other logic gates to model commercial programmable logic devices such as the PAL series from National Semiconductor. This article will describe the process of modeling the PAL16C1 device from National's 20-Pin Small PAL Family.

The PAL16C1 device is a 20 pin device which has 16 dedicated inputs and one combinatorial pair of outputs. The other two pins are the power and ground pins. The PAL16C1 is created through National's Schottky TTL process. The internal logic of the device appears in National's Programmable Logic Devices Databook 1989. This device consists of three levels of logic: the logic array, one level of OR gates, and the combinatorial OR/NOR logic. The MC5 subcircuit listing of this device appears in Figure 5. The subcircuit consists of the following sections.

### The Subcircuit Header (.subckt)

The subcircuit header denotes the beginning of the subcircuit definition. It defines the node numbers that are used in calling the subcircuit, along with optional nodes and parameters. Although, the PAL16C1 is a 20 pin device, only 18 pins are listed in the subcircuit call. A check of the logic diagram will reveal that the missing pins, pins 10 and 20, are the ground and power pins. These two pins have been placed as optional pins through the "optional:" keyword. Their default values are the globally defined values of \$G\_DPWR and \$G\_DGND. Having the power and ground pins as optional pins lets you place the PAL16C1 on a schematic without having to externally connect it to power supplies. The two parameters, MNTYMXDLY and IO\_LEVEL, control the digital delay and the digital I/O level, respectively, for the subcircuit. The default value of 0 specifies that these parameters will take the values of DIGMNTYMX and DIGIOLVL, which are set in the Global Settings. The keyword "text:" specifies a text parameter that lets you pass a text file to the subcircuit. The default file that the subcircuit will search for is "PAL16C1.JED". This text file is the one that will be used to program the programmable logic array.



---

\*This subcircuit models the PAL16C1 from National Semiconductor  
\*

```
.subckt PAL16C1 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19
+ optional: 20=$G_DPWR 10=$G_DGND
+ params: MNTYMXDLY=0 IO_LEVEL=0
+ text: JEDEC_FILE="PAL16C1.JED"

U1 PLANDC(16,16) 20 10
+ 2 1 3 19 4 18 5 17 6 14 7 13 8 12 9 11
+ ROW1 ROW2 ROW3 ROW4 ROW5 ROW6 ROW7 ROW8 ROW9 ROW10
+ ROW11 ROW12 ROW13 ROW14 ROW15 ROW16
+ DLY_16C1 IO_S MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}
+ FILE={JEDEC_FILE}

U2 ORA(8,2) 20 10
+ ROW1 ROW2 ROW3 ROW4 ROW5 ROW6 ROW7 ROW8
+ ROW9 ROW10 ROW11 ROW12 ROW13 ROW14 ROW15 ROW16
+ OR1 OR2
+ D0_GATE IO_S MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}

U3 OR(2) 20 10
+ OR1 OR2 16
+ D0_GATE IO_S MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}

U4 NOR(2) 20 10
+ OR1 OR2 15
+ D0_GATE IO_S MNTYMXDLY={MNTYMXDLY}
+ IO_LEVEL={IO_LEVEL}

.model DLY_16C1 upld (tplhTY=25ns tplhMX=35ns tphlTY=25ns tphlMX=35ns)

.ENDS
```

***Fig. 4 - Subcircuit Listing for the PAL16C1***

---

## The Programmable Logic Array (U1)

The programmable logic array is the core of the PAL16C1. As is shown in Figure 4, the array in the PAL16C1 consists of 16 pairs of true and complement inputs that feed 16 output AND gates. The appropriate <pld type> to use in this case is the PLANDC which is an AND array with true and complement inputs. The PLANDC is declared with 16 input pairs and 16 outputs. The optional nodes, 20 and 10, are tied to the power and ground pins of this device. The inputs are then declared, starting with the smallest input line number from Figure 4. The first input pin will be 2 since it is connected to the input line numbers 0 and 1. Then the other inputs are declared going from the next smallest input line number to the largest input line number. Each input will be connected to two consecutive input line numbers because the PLANDC is a complementary input array. After the inputs are defined, the 16 outputs are declared. These outputs can be given any name but will reference each row going from the smallest product line to the largest product line. In Figure 5, they are given the names ROW1 through ROW16. After the outputs appear the model names used by this device. DLY\_16C1 is the <timing model name>. Since there is no feedback within the PAL16C1, all of the timing delay may be placed within this level of logic. The DLY\_16C1 model statement appears at the bottom of the subcircuit in Figure 5. IO\_S is the <I/O model name>. This model was chosen because the PAL16C1 was created through a Schottky TTL process. The model statement for IO\_S is contained in the DIGIO.LIB file. The MNTYMXDLY, IO\_LEVEL, and JEDEC\_FILE parameters are all passed from the parameter declarations in the subcircuit heading.

## The OR array (U2)

The OR array defines two 8 input OR gates. These gates are shown in Figure 4 with each having 8 of the programmable logic array outputs being fed into them (ROW1 through ROW16). The D0\_GATE timing model name refers to a zero delay model that is stored in the DIGIO.LIB file.

## The Combinatorial OR/NOR Logic (U3 and U4)

The U3 and U4 devices define a two input OR gate and a two input NOR gate. These two gates take the outputs of the OR array and produce the outputs at pins 15 and 16 of the PAL16C1 device.

## Adding the PAL16C1 to the Component Menu

To access the PAL16C1 through the Component menu, the device must be linked to a shape and given pins in the Component Editor. Figure 6 shows the settings in the Component Editor that were used for the PAL16C1. To add a device, highlight the group that you would like to place it in, and then click on the Add Component command button. This places a new part into the highlighted group. The name should be the subcircuit name, but may actually be anything. The shape chosen was a 20 lead dip shape, and the device was defined as a subcircuit. The pins from the subcircuit header need to be entered in the Shape/pin display. We will leave out the optional pins because their default values suit our purposes. Click in the Shape/pin display, and type in the first pin "1". Place this at the first lead, and then repeat this process with the other 17 pins. Make sure that the pin names are defined as digital pins. The pin names must be the exact same names that they are defined as in the subcircuit header. Close and save the component settings. The device is now ready for placement in a schematic.

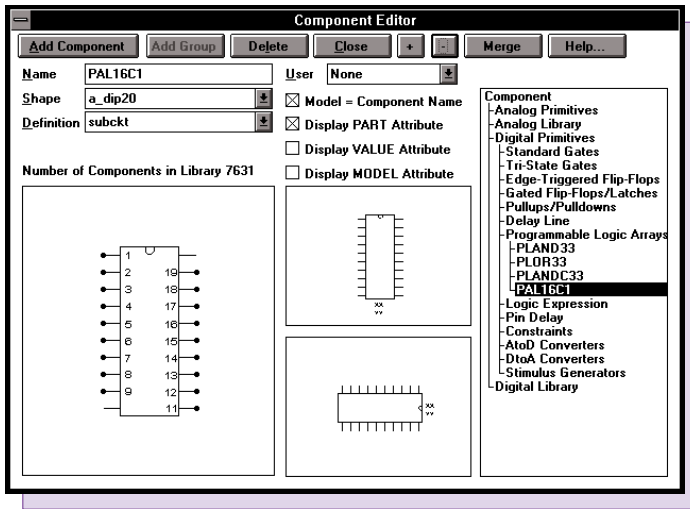
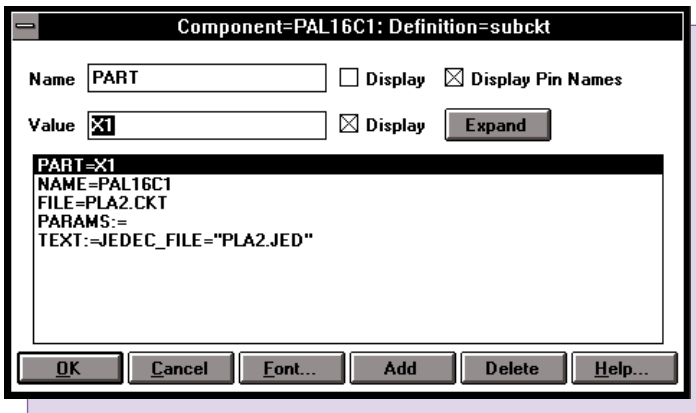


Fig. 5- Component Editor Settings for the PAL16C1

---

## Using the PAL16C1 in a Schematic

The PAL16C1 may now be accessed through the Component menu. Select the device from the group that it was entered in and place it in the schematic. In the Attribute dialog box, a few of the attributes will have to be defined. Figure 7 shows one way in which the PAL16C1 attributes may be defined. The PART attribute may be any name as long as it begins with an underscore or a letter. Having the PART attribute start with an X maintains SPICE compatibility. The NAME attribute must be the subcircuit name that was defined in the subcircuit header. The FILE attribute is the file name that the subcircuit is located in. This only needs to be defined if the file does not exist in the Nom.lib listing. The PARAMS attribute is left blank in order to use the values set in the Global Settings. The TEXT attribute passes a text parameter to the subcircuit. In this case, the text parameter will be the name of a JEDEC file that is used to program the PLA.



*Fig. 6- The PAL16C1 Attribute Dialog Box*

A JEDEC file is an industry wide standard format for programming a PLD. It provides a table that indicates which fuses are to be programmed to implement the desired logic functions. Most, if not all, commercial PLD software should be able to create a JEDEC file. Figure 8 shows the JEDEC file that was used in the PAL16C1 test circuit (shown in Figure 9). This file programs the PAL16C1 as an 8-bit equality comparator.

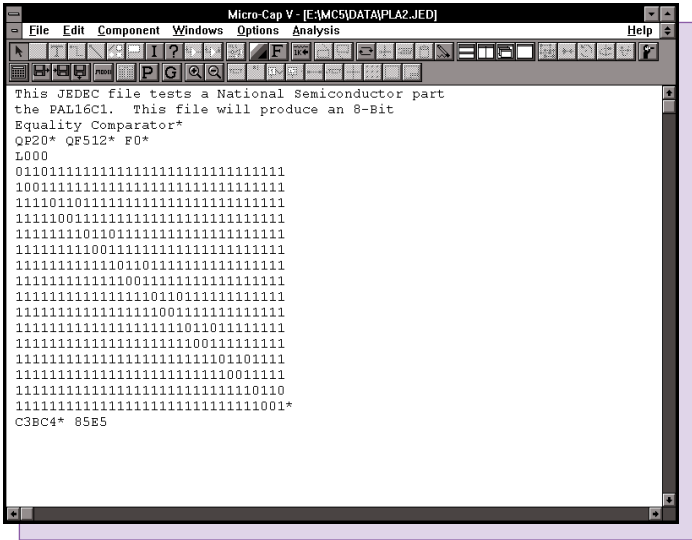


Fig . 7 - The JEDEC file

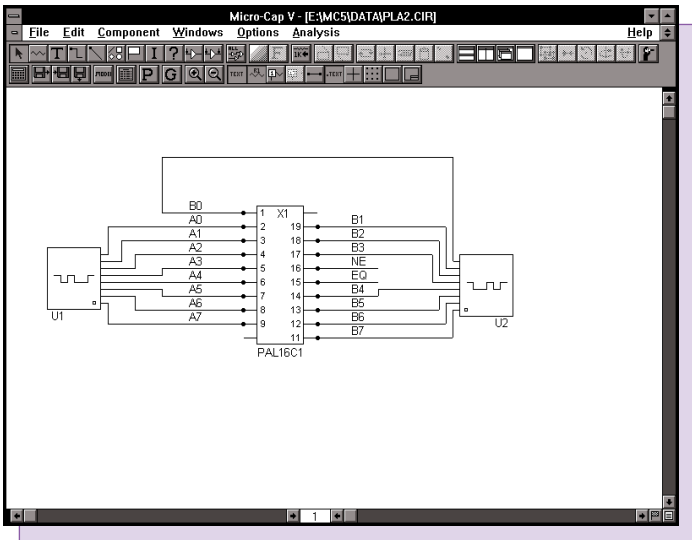
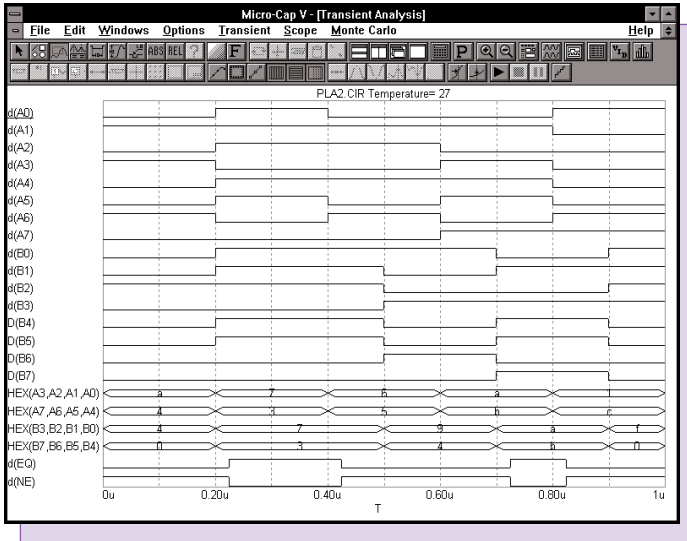


Fig . 8 - The PAL16C1 test circuit



**Fig. 9 - The PAL16C1 Simulation**

The test circuit was set up to compare two 8-bit stimulus generators. Figure 10 shows the results from this simulation. The HEX operator is used to transform the stimulus waveforms into hex values for an easier comparison. At 200ns and 700ns, the stimulus generators are generating the same output, and as a result, the PAL16C1 produces a logic 1 on its equality output, EQ. When the stimulus generators do not match, the NE, not equal, output is at an active high.

---

## Creating and Using a Macro

One of the most useful components provided by Micro-Cap V is the macro. The macro lets you represent a large number of components on a schematic through a single component. This feature lets you black box a portion of the schematic so that it is represented by a simple shape. It is also useful if you have a schematic that is reused often in other circuits. Creating a macro of this schematic makes it as easy to place into another circuit as any primitive component. The macro is very similar to the subcircuit component except that it invokes a schematic whereas the subcircuit invokes a SPICE netlist.

Upon running an analysis, the macro circuit file is substituted for the macro component, so the first step in creating a macro is to create the schematic.

### Creating the macro circuit

A new schematic file needs to be opened, and then the schematic may be created either from scratch or through a copy and paste operation. The macro circuit is created just like any other circuit. There are two main differences between a macro circuit and a standard circuit: pin names and parameters.

MC5 needs to be able to link the appropriate nodes in the macro with the external connections in the calling circuit. This is done through placing pin names on the nodes where connections are desired. The pin names are placed by a simple grid text operation. Each pin name must be connected directly to the node or there will not be an electrical connection. Any legal node name may be used for a pin name, but it is best to use a name that provides a description of the pin's behavior.

The second difference is that the macro may use a .PARAMETERS statement that lets you pass parameters to the macro through the calling circuit. The format used for the .PARAMETERS statement is:

```
.PARAMETERS(par1[,par2[,par3...]])
```

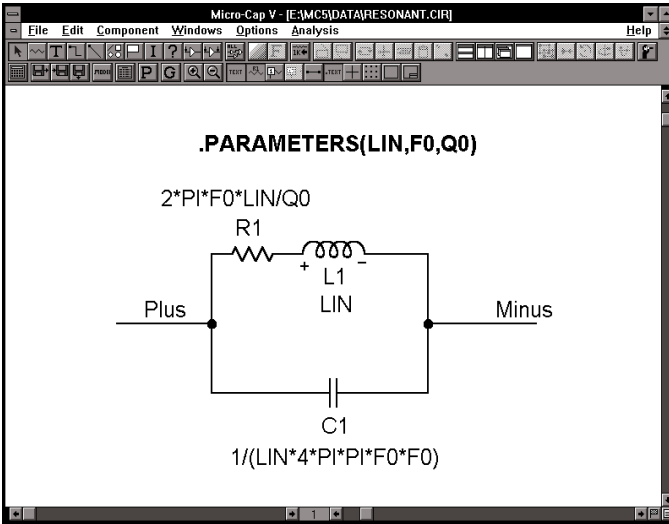
This is useful in creating a generic template of a component. A specific model would only require new parameters in the calling circuit rather than having to edit the macro circuit. A macro circuit does not require a .PARAMETERS statement.

The macro circuit we will use as an example is the two branch parallel resonant network shown in Figure 11. The two pin names of this macro are Plus and Minus, and they appear on the two nodes that are to be externally connected. Three parameters are specified to be passed to the macro in the .PARAMETERS statement. These parameters are the inductance value (LIN), the resonant frequency (F0), and the quality factor at resonance (Q0). These three parameters are used to derive the values for the resistance, capacitance, and inductance. The inductance value is taken directly from the parameter LIN, and the resistance and capacitance values are calculated from the following equations:

$$R = (2 * \pi * F_0 * L) / Q_0$$

$$C = 1 / (L * (2 * \pi * F_0)^2)$$

The right sides of these equations are entered as the VALUE attributes for R1 and C1.

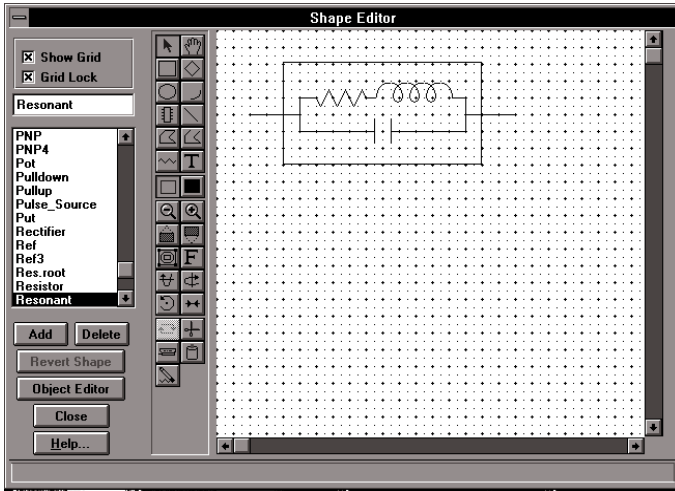


*Fig. 10 - Resonant Macro Circuit*



## Shape and Component Editors

The next steps are to create a shape for the macro and then to set up the macro as a component. Figure 12 shows the shape that was created for the Resonant macro in the Shape Editor. The resistor, inductor, and capacitor were copied and pasted from the existing shapes, and then a rectangle was drawn around them. The appropriate leads were then added. The shape doesn't need to be as complicated as this. It may be as simple as a rectangle with two leads, or even an existing shape.

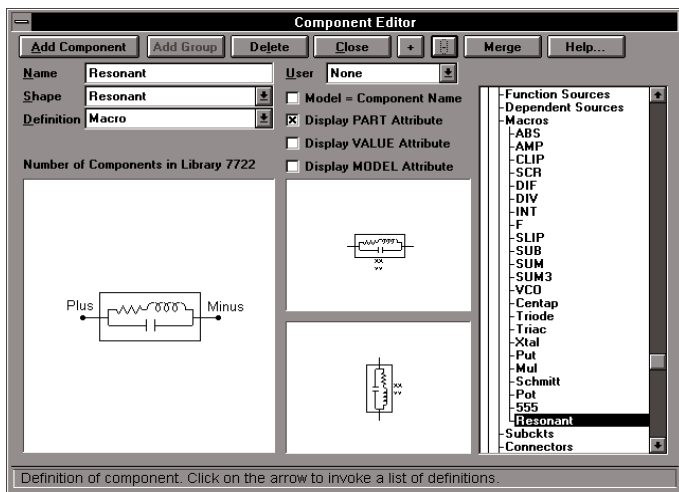


*Fig. 11 - The Resonant Shape in the Shape Editor*

Enter the Component Editor to set up the macro as a component. The Component Editor settings for the Resonant component are shown in Figure 13. To add a component, highlight the group that the component will be placed in and click on the Add Component command button. In this case, the component was placed in the Macros group of the Analog Primitives.

The Name of the component must be the exact name of the macro circuit. In this case, the name would be Resonant as the macro circuit was called Resonant.cir. The Shape is chosen from the list of shapes available in the Shape Editor, so we chose the shape created earlier, Resonant. The Definition field specifies the electrical definition of the component which in this case will be macro.

The last important step in entering the macro is defining the pins. The pins must match the pin names that were created in the macro circuit. The pins Plus and Minus were added by clicking in the Shape/pin display window and specifying their names and the fact that they are analog pins. The pin marker dot and the pin name may both be dragged to the desired location such as the end of the lead. The other options in the Component Editor, such as positioning the attribute text, mainly define how the component is shown when initially placed in the schematic.



*Fig. 12 - The Resonant Component Editor Settings*

## Testing the Macro

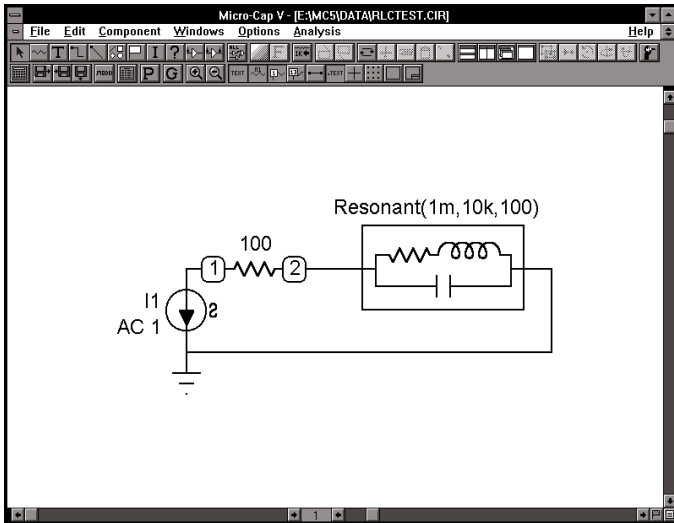
The test circuit we used for the Resonant macro is shown in Figure 14. This is a simple circuit consisting of the macro, a 100 ohm series resistor, and a 1 amp AC independent current source. The VALUE attribute of the macro has been defined as:

```
Resonant(1m,10k,100)
```

The name in the VALUE attribute must match the name of the macro circuit, and the number of parameters specified must equal the number of parameters specified in the macro's .PARAMETER statement.

The parameters are passed to the macro in the order that they appear in the .PARAMETERS statement. In this circuit, LIN is defined as 1m, F0 is defined as 10k, and Q0 is defined as 100.

The circuit is set up for an AC analysis run to measure the impedance of the circuit. To do this in MC5, the AC source must be the I independent source that has its value attribute defined as 'AC 1'. This component is found under the Waveform Sources group of the Analog Primitives under the name 'I'. The source must then be placed across the nodes where the impedance is to be measured. The waveform measured is the voltage across the I independent source. In actuality, this gives us the equivalent input impedance of the entire circuit since the current is 1 amp and therefore  $V$  is equal to  $Z$ .

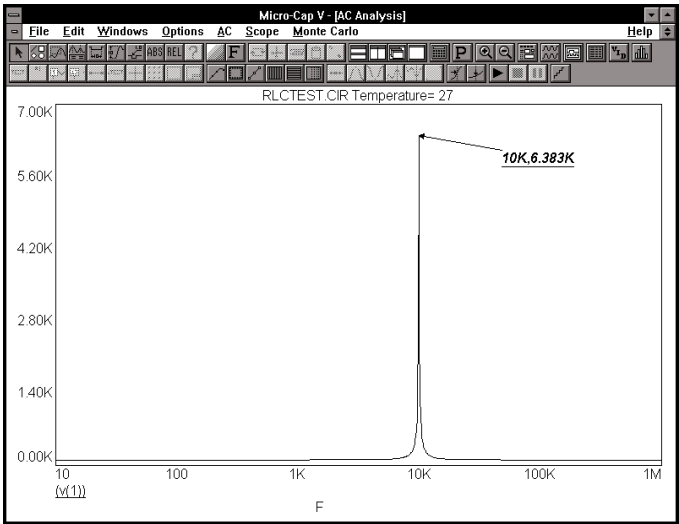


*Fig. 13 - The Resonant Macro Test Circuit*

The AC analysis is swept over the frequency range of 10Hz to 1MHz. The Frequency Step is set to Fixed Log, and the Number of Points is defined as 251. These two steps need to be done to insure that a data point is calculated at the resonance frequency. If the analysis is set to Auto Frequency Step, the impedance peak we are trying to analyze may not appear if there isn't a data point calculated at that frequency. Figure 15 shows the results of the AC analysis. At the resonance frequency, the impedance of the macro should be:

$$Z = R * Q_0^2 - j * R * Q_0$$

This equation calculates an impedance value of 6.283k which when added to the 100 ohm series resistor produces the 6.383k value that is displayed at 10kHz in the analysis.



**Fig. 14 - The AC Analysis Results**

---

## Frequency Dependent Resistors

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 a resistor is defined as '10\*f+100', the actual resistance during the AC analysis simulation will be 100. This occurs because the resistor is linearized at the DC operating point when the frequency is equal to 0. Trying to implement a frequency dependency in a resistor will not work for AC analysis. However, a resistor may be represented by a Laplace source which is the only type of component that can accept frequency dependent equations.

The Laplace sources are dependent sources whose transfer function is a function of the complex frequency variable,  $S$ . In AC analysis, the transfer function is computed from the expression  $S=2*\pi*frequency*j$ . To generate a function with a zero degree phase angle, which is necessary for the resistor, multiply  $S$  by  $(-S)$ . This will cancel out the phase and leave the value  $(2*\pi*frequency)^2$ . To get frequency, simply divide by  $(2*\pi)^2$  and then use the square root operator. The equation appears as follows:

$$F = \sqrt{(-s*s)/(4*\pi*\pi)}$$

### A Skin Effect Example

One common usage for a frequency dependent resistor would be when simulating skin effect. 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 the use of a Laplace source. The first step is to compute the transfer function needed to represent the AC resistance.

The AC and DC resistances are related through the effective area of the wire as follows. Refer to Figure 16 for a graphical representation of the variables used.

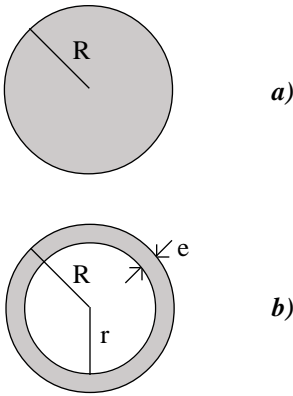
$$\text{AC resistance/DC resistance} = \text{DC area/AC area}$$

$$\text{AC resistance} = (\text{DC area/AC area}) * \text{DC resistance}$$

$$\text{DC area} = \text{PI} * \text{R}^2$$

$$\text{AC area} = \text{PI} * \text{R}^2 - \text{PI} * \text{r}^2$$

$$\text{r} = \text{R} - \text{e}$$



**Fig. 15 - a) DC representation of a wire  
b) AC representation of a wire**

$$\text{AC resistance} = (\text{PI} * \text{R}^2 / (\text{PI} * \text{R}^2 - \text{PI} * (\text{R} - \text{e})^2)) * \text{DC resistance}$$

$$\text{e} = (3.16\text{e}3 / (2 * \text{PI})) * (\text{res} / (\text{f} * \text{k}_m))^{1/2}$$

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 a constant. It may now be implemented in a S variable transfer function. 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 create the macro circuit that will implement this transfer function.

The skin effect macro appears in Figure 17 below. It consists of a LFVofI Laplace source, an IofI dependent source, and a resistor. The resistor is included only to create a legal circuit loop with the IofI dependent source. The actual resistance is represented by the Laplace voltage source.

The IofI dependent source senses the current through the Laplace voltage source. It then produces that same current at its output, due to the gain of 1 and feeds that current into the input of the Laplace source. The transfer function has been defined with the impedance equations shown previously. In this case, the transfer function has been divided into two .define text statements. This was done to provide easier readability. Since the input to the Laplace source is the current through its output and the transfer function is the resistance, the output will be the voltage drop specified by the equation  $V=R*I$ . Essentially, this macro is a voltage source that represents the voltage drop that would occur had a resistor been present.

The macro has four parameters defined: 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.

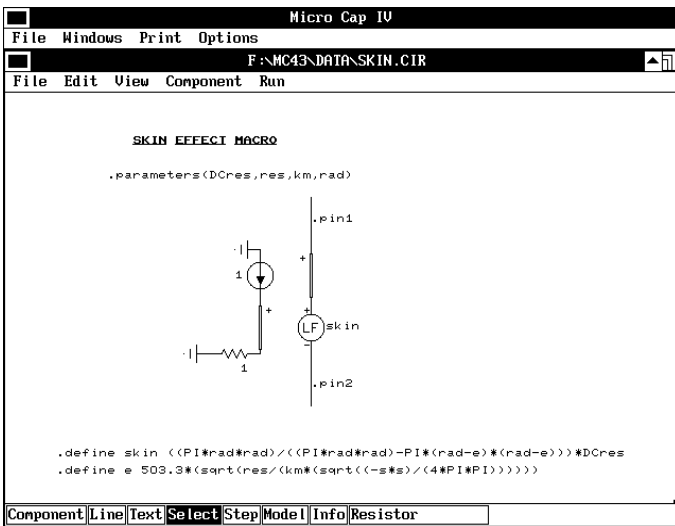
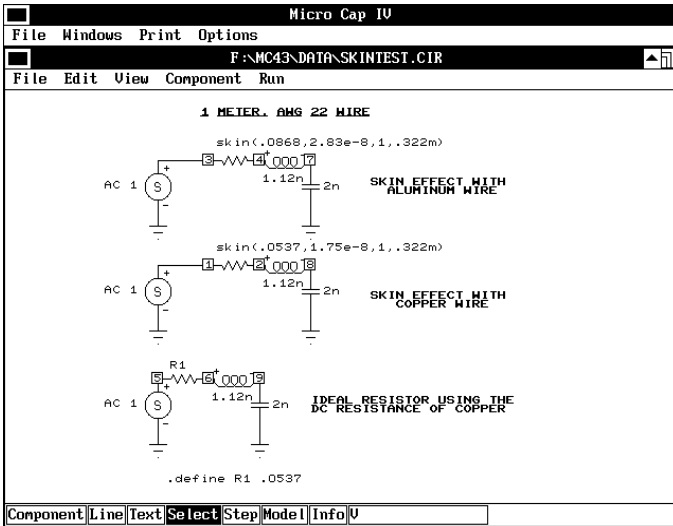


Fig. 16 - The Skin Effect Macro

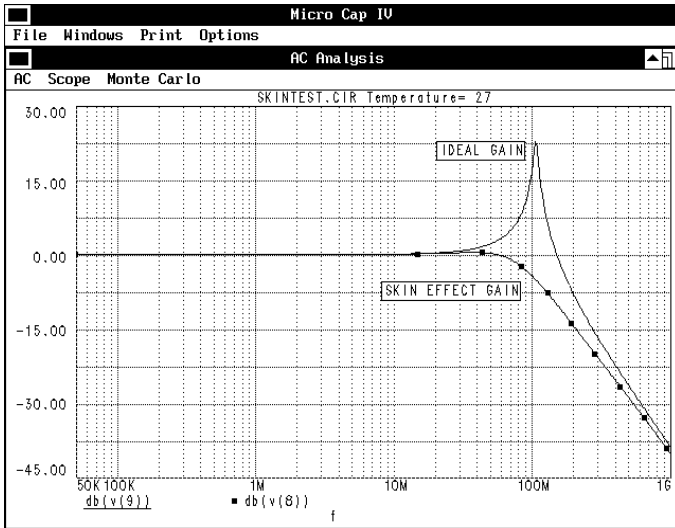
In the Component Editor, this macro was linked to a resistor shape and given the definition Macro. The schematic used to test the skin effect macro is shown in Figure 18. This test schematic actually consists of three separate circuits. All of them model the resistance of a 1 meter, AWG No. 22 wire. 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 macro. The inductor and capacitor were given nominal values since we are trying to display the effect that skin effect has in a simulation.



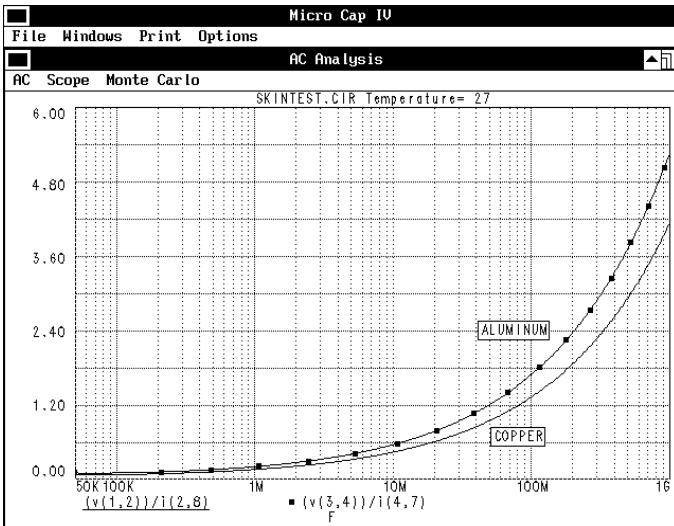
**Fig. 17 - The Skin Effect Test Circuit**

Two simulations were performed on this schematic. The first simulation compares the gain of the copper skin effect macro versus the gain of the ideal resistor. As shown in Figure 19, the higher resistance produced by the skin effect minimizes the resonance of the reactive components. The second simulation compares the apparent resistance between an equivalent size wire made of aluminum and one made of copper. As expected, Figure 20 shows that the resistance from the aluminum is higher throughout the frequency sweep. The skin effect equations were manually calculated at 10MHz to confirm the macro results for both materials, and the larger error percentage of the two amounted to only .03%.





*Fig. 18 - Skin effect vs. Ideal Gain Comparison*



*Fig. 19 - Apparent Resistance of Copper vs. Aluminum*

---

## HP Printer Driver Incompatibilities

An incompatibility brought to our attention recently by one user occurs between the new HP printer drivers and the Sentinel security key. These drivers use nonstandard bidirectional communication which can cause trouble to other devices that are present on the parallel port. Rainbow, the manufacturer of the security keys, provided us with some solutions for this occurrence in one of their technical notes.

If you have multiple parallel ports:

The easy solution to this problem is to place the security key on a different parallel port that the HP driver is not accessing.

If you have a single parallel port:

The most obvious solution is to change the printer driver. An older HP compatible version of the driver should work with the key. If you are using the HP5L printer, there is a PCL version of the driver available on the install disk. Custom installation must be selected during the install process to access this driver. Other HP printers may have similar solutions.

For HP Deskjet Printers:

The following steps may be used to disable the bidirectional communication mode in the HP Deskjet 855c, 850, 680c, 660, and 540 drivers.

- 1) Exit from all Windows applications.
- 2) Choose Run from the File menu in the Program Manager.
- 3) In the Run dialog box type the following appropriate command line:

HPVCNFIG ; for HP 850, 855c, 660c printers  
HPRCFG01 ; for new HP 850 and 855c printers  
HPFCFG02 ; for new HP 660c printers  
HPFCFG01 ; for HP 660c-SE printers  
HPVCNFIG ; for HP 540 printers

4) Click OK, and then click Continue when a message appears that says the program will modify the System.ini file.

5) The following selections should be changed in the dialog box that appears:

---

For HP 850 and 660 printers:

- A) Disable the "print cartridge query".
- B) Disable the "bidirectional communication".

For HP 540 printers:

- A) Enable the "reverse channel".
- B) Enable the "simple IOD".

6) Restart Windows.

7) From the Main group, choose Control Panel and then Printers. Click on the Connect command button. Highlight the LPT1.DOS port and then disable the "fast printing direct to port" in the lower left hand corner. Close the dialog box.

If your printer is printing garbage with the key attached, disabling the "fast printing direct to port" off in Windows will sometimes help.

In the extreme case, where none of this helps, and for some reason, you must use the newest printer driver, the only alternative is to install a new parallel port.

For additional information, you should contact HP support.

---

## PRODUCT SHEET

### Latest Version numbers

Micro-Cap V IBM/NEC ..... Version 1.1  
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 ..... 103114.61@compuserve.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, MASTERCARD. Purchase orders accepted from recognized companies in the U.S. and Canada. California residents please add sales tax.