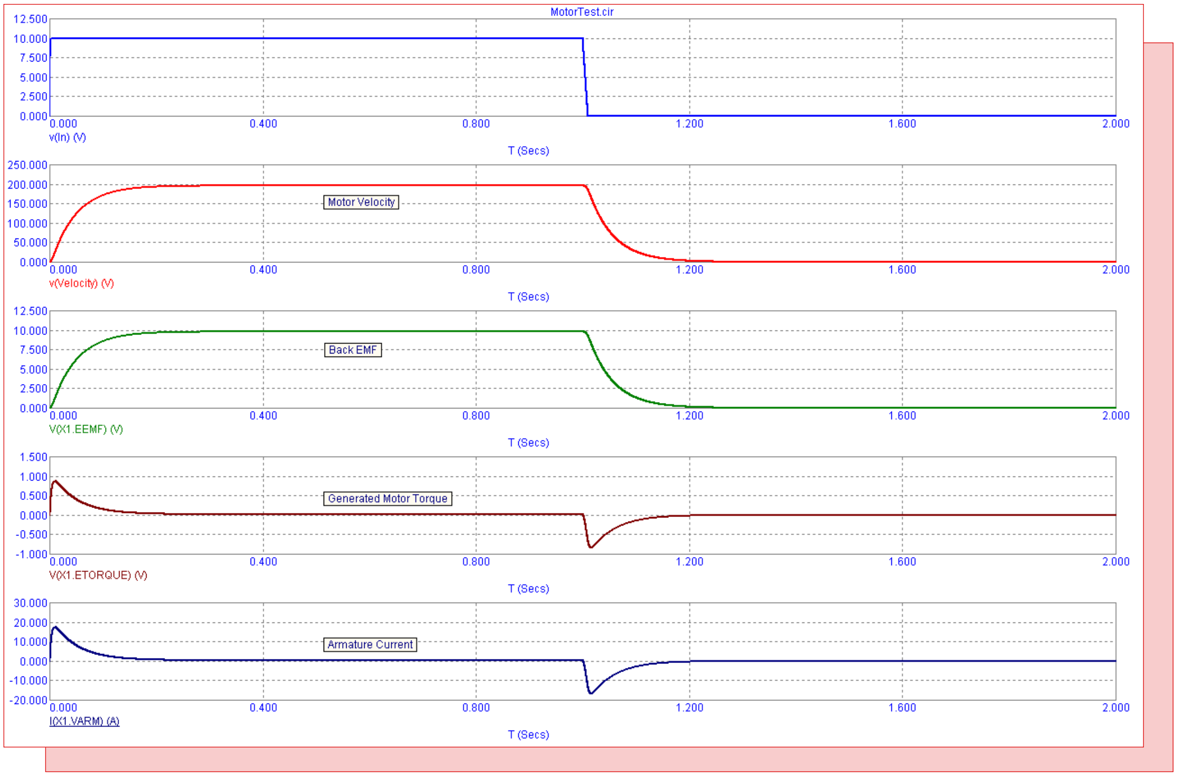
spectrum news

Applications for Micro-Cap™ Users

Fall 2007

News



DC Motor Macro

Featuring:
- Simulating a PID Controller
- DC Motor Macro
- Using the Bus

News In Preview

This newsletter's Q and A section describes the process to take when receiving permission denied errors within Micro-Cap. The Easily Overlooked Feature section describes the capability of the user to assign their own shortcut key combinations to commands in Micro-Cap.

The first article describes how to simulate a PID controller and use it as a control loop feedback mechanism.

The second article describes a macro model of a DC motor.

The third article describes how to use the bus device within a schematic to transport multiple signals and improve schematic readability.

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 having issues with permission denied errors both in the schematic and in the analysis. When I double click on an NPN transistor in the schematic, I receive the following error message:

Permission denied 'C:\MC9\LIBRARY\NOM_LIB.INX'

Error occurred while trying to write index for file 'C:\MC9\LIBRARY\NOM.LIB'.
File: C:\MC9\DATA\circuit1.cir.

Then when I try to run a transient simulation on this circuit, the following error occurs:

Permission denied 'C:\MC9\DATA\circuit1.TNO'

File: C:\MC9\DATA\circuit1.cir

How do I get past these errors?

Answer: The permission denied error occurs when Micro-Cap is unable to perform the necessary read or write to the specified file. In most cases, the inability to perform a write operation is what triggers the error. There are two main causes for this.

1) The folder the file is located in or the specific file itself has been write protected. The Windows operating system controls these permissions. Right click on the folder or file in Windows Explorer and select Properties. The object may be write protected through its Attributes or within the Security page. Make sure that the write operation is allowed. A simple way to check to see if the folder is write protected is to load a program such as Notepad and try to save a small text file to the folder. If you are able to successfully save a file, then the folder is fine, and it may be the specific file that is the issue.

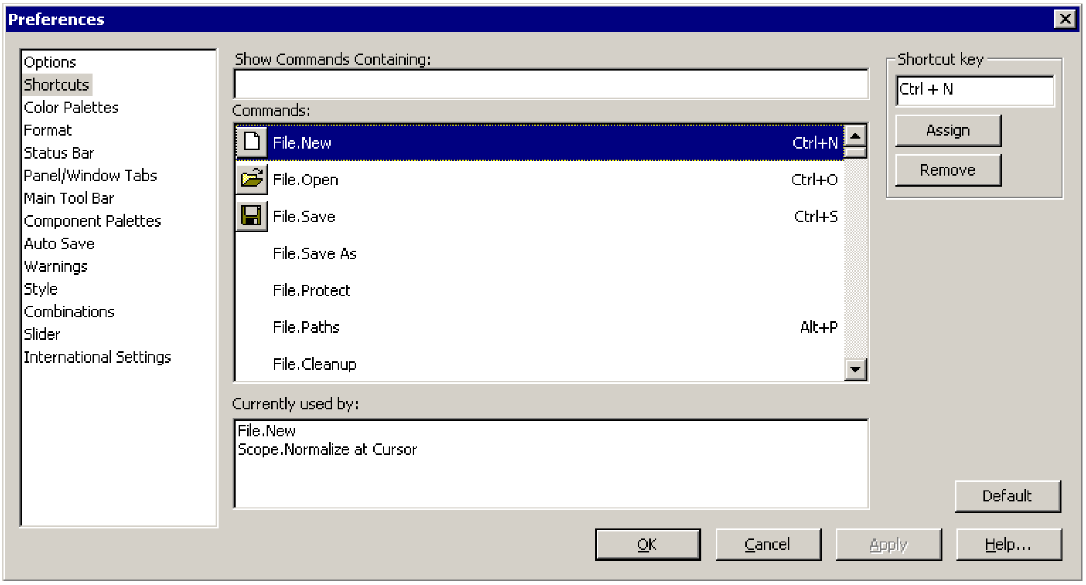
2) The path that Micro-Cap is trying to write the file to does not exist. If any of the folders within the path do not exist, then the write operation will fail. For most cases, the paths that Micro-Cap uses are defined within the Paths dialog box. Go to the File menu and select Paths. In the dialog box that is invoked, check that all of the specified paths are valid. When the OK button is clicked in this dialog box, Micro-Cap will automatically check the validity of all specified paths and will return an error message specifying any invalid paths. An invalid path in this dialog box usually occurs when a Micro-Cap installation is copied directly from one system to another but its location on the hard drive is shifted to a different partition or folder.

Easily Overlooked Features

This section is designed to highlight one or two features per issue that may be overlooked among all the capabilities of Micro-Cap.

User Specified Shortcuts

Many of the commands available within Micro-Cap come assigned with accelerator keys. The user may want to modify the shortcut of a command or add a keyboard shortcut to a command that does not have one. The Shortcuts page within the Preferences dialog box provides this capability. The Shortcuts page lets you customize the accelerator or shortcut keys for all of the menu commands available in Micro-Cap. The page appears as follows:



Shortcut key: Ctrl + N

File.New	Ctrl+N
File.Open	Ctrl+O
File.Save	Ctrl+S
File.Save As	
File.Protect	
File.Paths	Alt+P
File.Cleanup	

Currently used by:
File.New
Scope.Normalize at Cursor

Fig. 1 - Shortcuts page in the Preferences dialog box

Show Commands Containing: This text field narrows down the choices in the Commands list. Typing in a text string will display in the Commands list only those commands that contain the text string. The Commands list is updated dynamically while the string is typed in. For example, type in Clear to show just the commands that contain the word Clear.

Commands: This list shows all of the available commands that can have a shortcut key assigned to them. Narrow down the list by entering a text string in the Show Commands Containing field. Click on a command to add or edit its shortcut key.

Shortcut key: This field defines the shortcut key for the selected command. Use the Ctrl, Shift, or Alt keys along with any standard keys to define an entry. Click on the Assign button to assign the shortcut to the selected command or click on the Remove button to remove the currently assigned shortcut from the selected command.

Currently used by: This field displays all commands that currently use the shortcut key specified in the Shortcut key field.

Simulating a PID Controller

The PID (Proportional-Integral-Derivative) controller is a common feedback mechanism used within closed loop control systems. The controller is used to automatically adjust a variable to keep a specified measurement at a set point. It can be found in applications that need a form of temperature control, flow control, or position control.

The PID controller operates by comparing the set point versus the output measurement and calculating the error between the two. The controller performs three calculations on the error value. The proportional segment calculates the reaction to the current error value. The integral segment calculates the reaction based on the history of the error value. The derivative segment calculates the reaction based on how quickly the error value is changing. The sum of these calculations is used to adjust the control system towards the set point.

Each of the proportional, integral, and derivative calculations has a scale factor that provides a weighting to the importance of that calculation in the controller output. An increase in the proportional scale value will decrease the rise time and reduce the steady state error, but too high of a value will lead to an unstable system. An increase in the integral scale value will eliminate the steady state error but can lead to an overshoot in the transient response. An increase in the derivative scale value reduces overshoot and increases the stability of the system. Potential requirements of the control system such as limiting overshoot also need to be taken into account when determining the scale factors.

Simulating a PID controller is simple in Micro-Cap due to the behavioral macros that come with the program. An example circuit that uses a PID controller to set the position of a mass connected to a spring is displayed in the figure below.

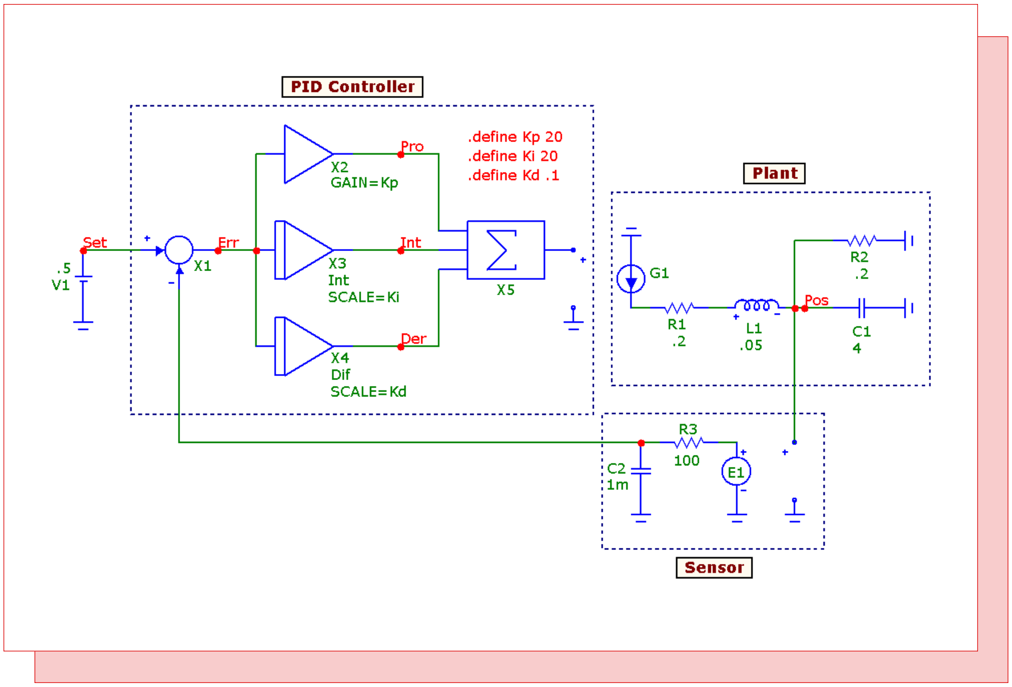


Fig. 2 - PID controller example circuit

The PID controller is modeled using five macros. The Sub macro (X1) calculates the difference between the set point value and the output measurement to produce the error value. This error value is then fed into the proportional, integral, and derivative inputs. The Amp macro (X2) calculates the proportional response. Its Gain parameter has been defined with the Kp symbolic variable. The Int macro (X3) calculates the integral response. Its Scale parameter has been defined with the Ki symbolic variable. The Dif macro (X4) calculates the derivative response. Its Scale parameter has been defined with the Kd symbolic variable. The three symbolic variables have their values set using the following define statements:

```
.define Kp 20
.define Ki 20
.define Kd .1
```

Modifying the values in the define statements will tune the PID controller for the specific application. The outputs of the Amp, Int, and Dif macros are then input into a Sum3 macro (X5) which produces the PID controller output with which the control system is adjusted.

In this example, the PID controller is used to position a mass that is connected to a spring by adjusting the force that is applied to the system. There are also a couple of friction elements present in the system. The mechanical diagram of this control system is displayed in the figure below.

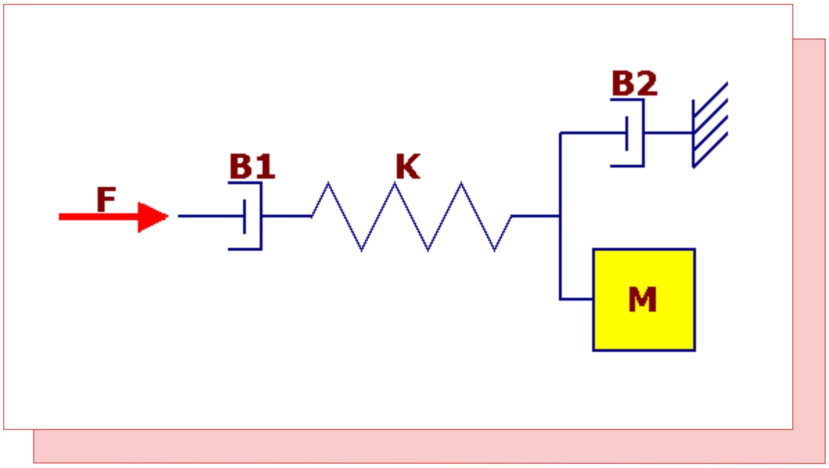


Fig. 3 - Mechanical equivalent of the plant circuitry

In order to simulate this mechanical system in Micro-Cap, the elements need to be converted into their electrical analogs. The force applied to the system is represented by a current. Since this current needs to be adjusted by the PID controller, an IofV dependent source (G1) is used where the voltage input to the source is the output of the controller. The two friction elements, B1 and B2, are represented by resistors R1 and R2. These resistors actually model inverse friction so the resistance values need to be defined as 1/friction. The analog of the spring, K, is the inductor L1. The inductor models the compliance of the spring so the inductance needs to be defined as 1/spring constant. The analog of the mass, M, is the capacitor C1 connected to ground. The capacitance is defined with the mass value. The voltage of the node between the capacitor and the inductor is equivalent to the position of the mass in the system.

The sensor that performs the measurement of the mass position in this case is just a simple VofV dependent source. For many control systems, the output measurement would need to be converted into an appropriate unit to compare versus the set point. However, since the output measurement and the set point value are on a direct one to one ratio, the gain has been set to one for the dependent source. An RC time delay was added to the output of the source to model a transport delay from the sensor output to the feedback input of the PID controller.

This schematic is then simulated in transient analysis. The V1 battery which defines the set point value has been stepped from .25V to 1V in .25V increments, and the simulation time is 5s. The resultant transient simulation is displayed below.

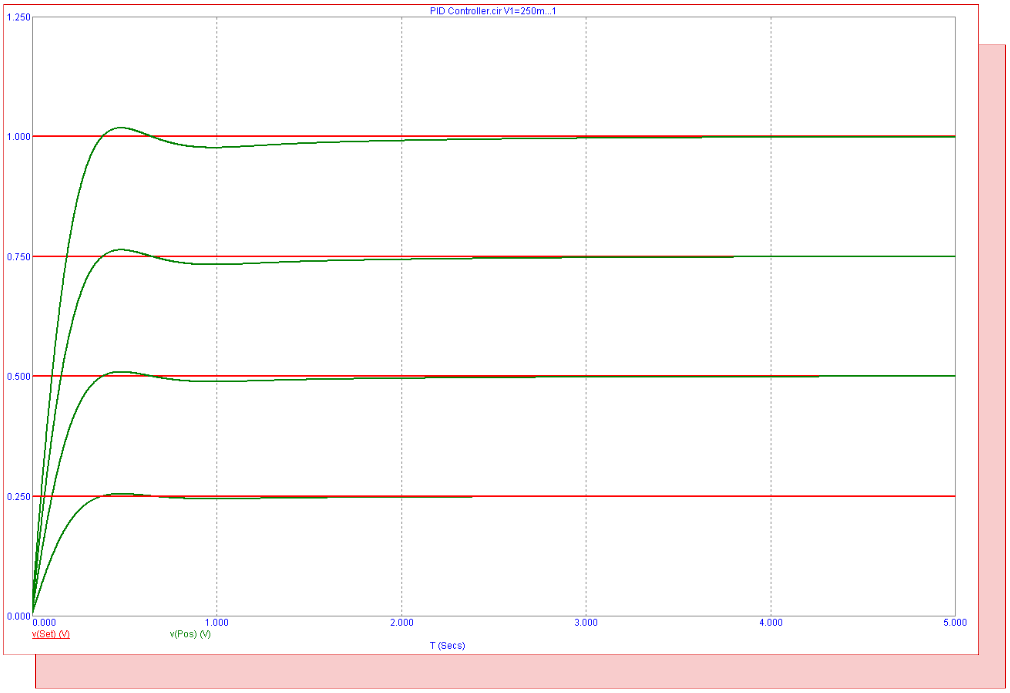
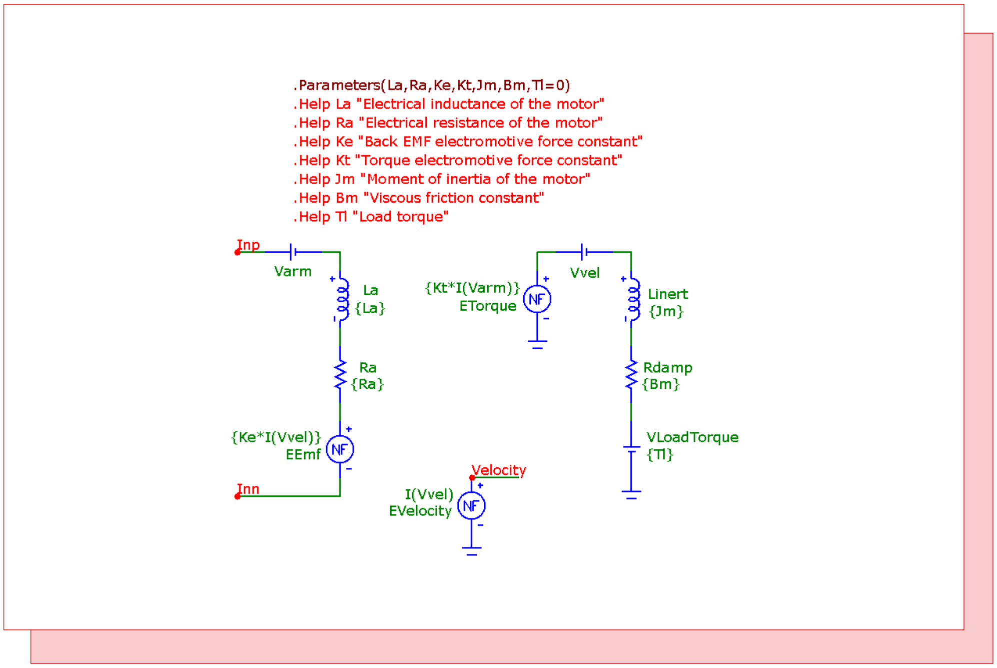


Fig. 4 - Transient response to the stepped set point

This simulation has the scale factors defined as Kp=20, Ki=20, and Kd=.1. Note that the position of the mass will be adjusted so that it is equivalent to the set point value in each branch of the simulation. There is a slight overshoot due to the integral which can be eliminated by setting the Kd parameter to 1. Modifying these scale factors will have a large effect on the transient response of the system so the tuning of these values is very dependent on the control system that is being adjusted.

DC Motor Macro

A common actuator used within control systems is the DC motor. The DC motor is designed to generate mechanical energy in the form of torque from an electrical energy input. Micro-Cap can model both the electrical portion of the motor as well as the mechanical portion. To model the mechanical system of the motor, an analogous electrical system is developed from the system equations. The macro circuit of the DC motor appears in the figure below.



```
.Parameters(La,Ra,Ke,Kt,Jm,Bm,Tl=0)
.Help La "Electrical inductance of the motor"
.Help Ra "Electrical resistance of the motor"
.Help Ke "Back EMF electromotive force constant"
.Help Kt "Torque electromotive force constant"
.Help Jm "Moment of inertia of the motor"
.Help Bm "Viscous friction constant"
.Help Tl "Load torque"
```

Fig. 5 - DC Motor macro

The macro circuit has seven parameters that are passed through to it: La, Ra, Ke, Kt, Jm, Bm, and Tl. The La and Ra parameters define the electrical inductance and resistance of the motor within its armature windings. The Ke parameter defines the back-emf electromotive force constant. The Kt parameter defines the torque electromotive force constant. The Jm parameter defines the inertia of the motor. The Bm parameter defines the viscous friction constant. The Tl parameter defines the torque of the load.

The electrical system of the DC motor is represented by the following equation:

Vin = Ra * Ia + La *dIa/dt + Vemf

where Ia is the armature current that will determine the torque generated by the motor, Ra is the electrical resistance of the motor, La is the electrical inductance of the motor, and Vemf is the back-emf voltage. The back-emf voltage is the voltage generated as the rotor of the motor moves through a magnetic field. As the velocity of the motor increases, the back-emf voltage increases. This voltage counteracts the input voltage of the motor. The back-emf voltage is calculated with the following equation:

Vemf = Ke * ω

where ω is the speed of the motor. The electrical system is modeled by placing an inductor (La), a resistor (Ra), and a nonlinear function voltage source (EEmf) in series. The EEmf function source has its VALUE attribute defined as:

Ke*I(Vvel)

where I(Vvel) is the electrical analog to the speed of the motor in the mechanical system. The Inp and Inn nodes in the macro circuit are the input nodes to the voltage differential that will drive the motor.

The mechanical system is defined by the following expression.

$Tm = Jm * d\omega/dt + Bm * \omega + Tl$

Tm is the torque generated by the motor. It is calculated with the following expression.

$Tm = Kt * Ia$

The torque generated is the armature current multiplied by the torque electromotive force constant. In the macro circuit, the torque is calculated through the NFV source called ETorque whose VALUE attribute is defined as:

Kt*I(Varm)

where I(Varm) measures the current through a zero volt battery in series with the armature circuitry.

The right hand side of the mechanical system equation has the same format as the electrical expression of a series RL circuit. Jm is equivalent to the inductance. Bm is equivalent to the resistance, and ω is equivalent to the current through the circuit. The Linert inductor and the Rdamp resistor in the macro circuit model this part of the equation. The VLoadTorque battery is defined with the Tl parameter and simulates a constant load torque placed on the motor shaft. The Vvel component is a zero volt battery that measures the current, as an equivalent of the motor velocity, for use in the back-emf source.

Finally, the NFV source named EVelocity is used to create a voltage equivalent to the motor velocity. This voltage will be used as an output of the macro so that the motor velocity is easily accessible in the main schematic. This output is useful if the motor needs to be geared down, or the position of the motor needs to be determined by integrating the velocity.

A simple circuit is created to test the motor macro. The circuit consists of a voltage source that is used as the input to the DC motor macro. The voltage source has its VALUE attribute defined as follows:

Pulse 0 10 0 1m 10m 999m 10

This source drives the motor with a 10V step voltage for one second. The values for the DC motor macro parameters were taken from an article (Reference 1) that describes a similar SPICE model for the motor. The macro parameters were set to the following values.

La = 1.5mH
Ra = .5ohm
Ke = .05 V*sec/rad
Kt = .05 N*m/A
Jm = 250u N*m/rad/s^2
Bm = .1m N*m/rad/s
Tl = 0 N*m

This test circuit is run in transient analysis with a simulation time of two seconds. The resultant transient plots are displayed below.

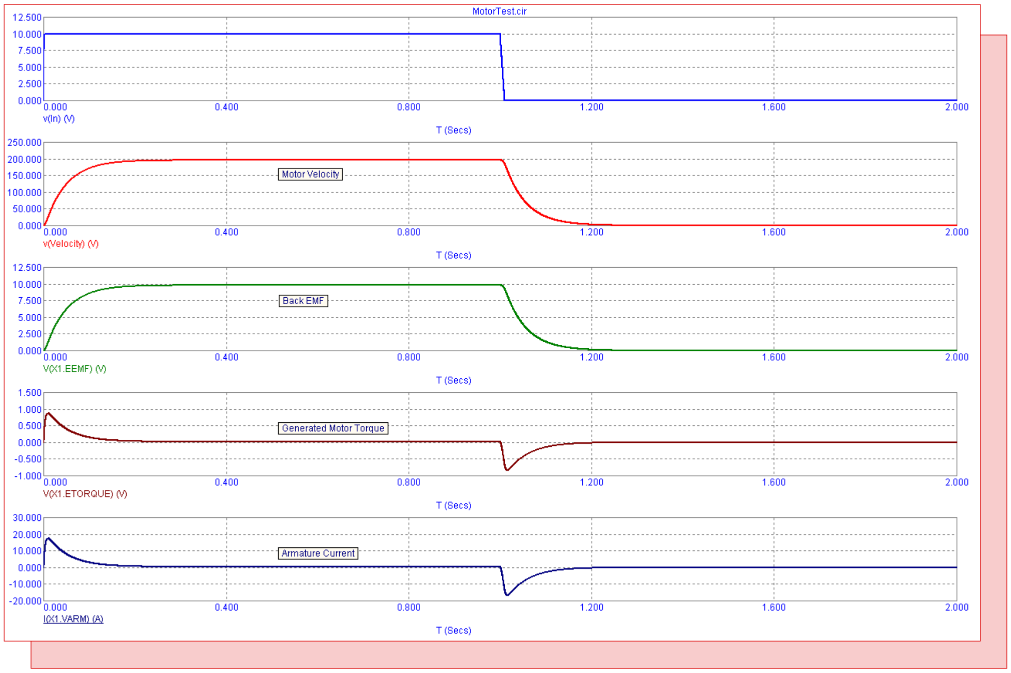


Fig. 6 - DC Motor transient analysis

The top waveform is a plot of the input voltage.

The second waveform plots the voltage at the Velocity pin of the macro which displays the motor velocity.

The third waveform plots the back-emf voltage inside the macro by referencing the voltage of the EEmf source. Note that X1 is the part name of the macro in the test circuit.

The fourth waveform plots the generated motor torque by referencing the voltage of the ETorque source in the macro.

The bottom waveform plots the armature current in the motor.

References

1) http://www.ecircuitcenter.com/Circuits/dc_motor_model/DCmotor_model.htm

Using the Bus

The bus capability within Micro-Cap provides a convenient method for transporting multiple signals while limiting the clutter of having multiple wires in the schematic. The bus is most useful for digital circuitry where related sets of signals are often routed through the same areas of the schematic such as with a memory or data set.

The bus is easy to implement in the schematic. It requires the use of two modes: Bus mode and Wire mode. The Bus mode is used to create the connectors that retrieve the individual signals from the bus wire. The Wire mode creates the bus wire that links the bus connectors. This article will describe the process of using a bus in a schematic.

The circuit used in this example will be a simple digital circuit. The circuit takes a pair of four-bit digital signals and inputs them to a 7483A adder. Both of the four-bit signals along with their sum from the output of the adder are input into 7448 BCD to seven segment decoders. This is an ideal situation to use the bus as the four-bit data sets are routed together throughout the circuit. Although this example consists of digital nodes, analog nodes can also be linked through the bus in the same manner. The digital circuit using standard wires to transport the signals is shown below.

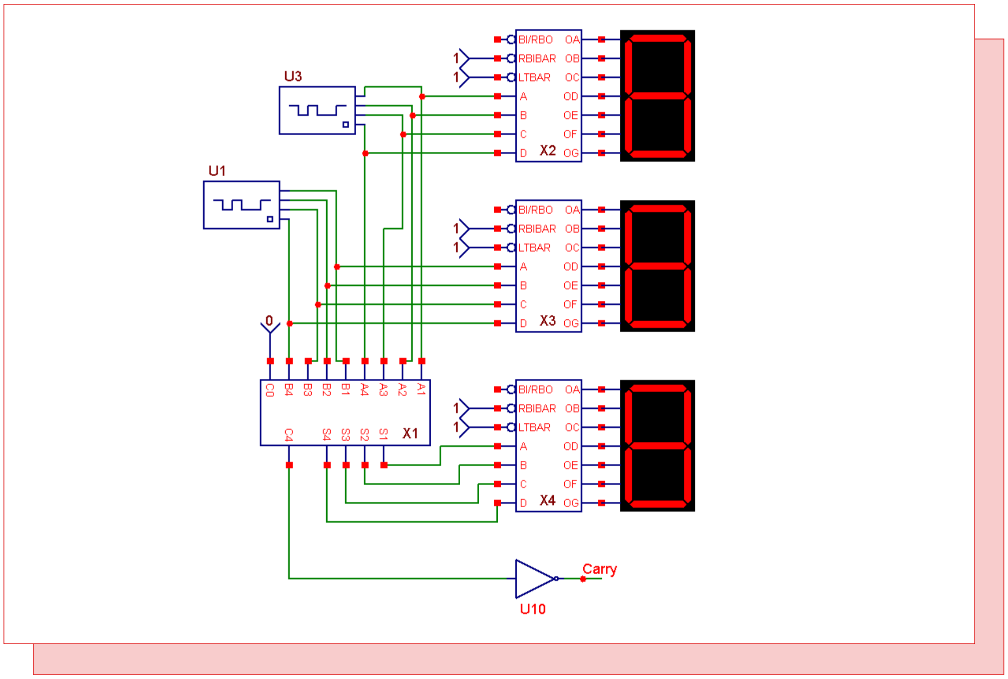


Fig. 7 - Bus example circuit using standard wiring

Even on a simple circuit such as this, it takes a little bit of effort to track the paths of the signals. Using the bus will greatly enhance the readability of the schematic. The first step is to delete all of the existing wires in the schematic as they will all be replaced with buses. The circuit without any wiring appears in Figure 8.

The first step in creating a bus is to place a bus connector. A bus connector is created by first enabling Bus mode, . Clicking in the schematic when in Bus mode invokes the Bus dialog box shown in Figure 9.

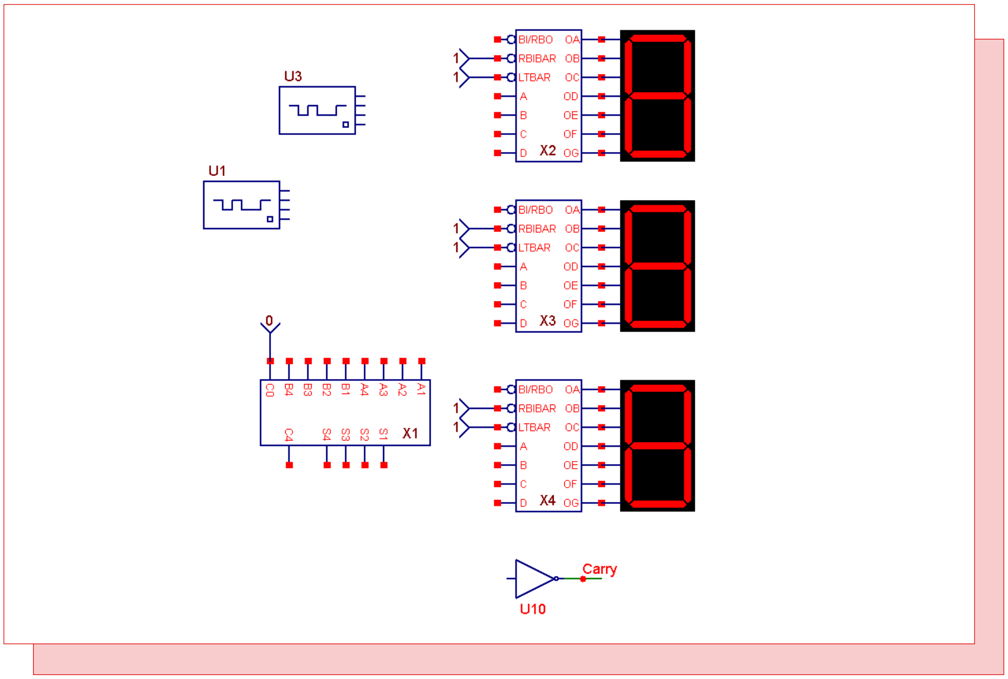
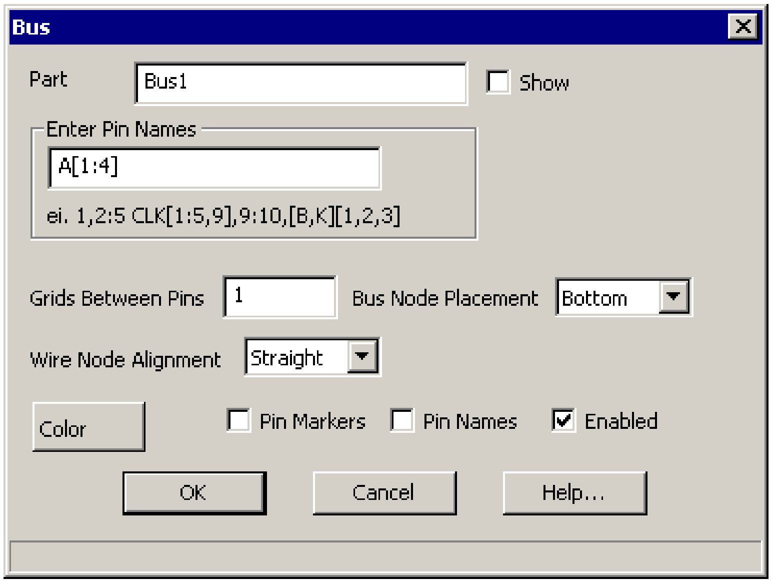


Fig. 8 - Bus example circuit with no wiring

The Bus dialog box lets you define the individual pin names of the connector along with the location and spacing of the bus node and wire nodes. The main entries in the dialog box are as follows:

Part: This field contains the bus connector name. The name has no effect on the simulation.

Enter Pin Names: This field sets the pin names, implicitly defining the number of separate wire nodes for the bus connector. The pin names determine the connectivity of the bus. For example, a pin named A1 will be connected to any other pins named A1 that are on other connectors that share the same bus wire. The names can be listed separately or using the following forms.



Bus

Part: Bus1 □ Show

Enter Pin Names: A[1:4]

ei. 1,2:5 CLK[1:5,9],9:10,[B,K][1,2,3]

Grids Between Pins: 1

Bus Node Placement: Bottom

Wire Node Alignment: Straight

□ Pin Markers □ Pin Names ☑ Enabled

Fig. 9 - Bus dialog box

A,B,C,D - A four pin bus with pins labeled A, B, C, and D
A[1:4] - A four pin bus with pins labeled A1, A2, A3, and A4
C[1:4,8,9] - A six pin bus with pins labeled C1, C2, C3, C4, C8, and C9
0:7 - An eight pin bus with pins labeled 0, 1, 2, 3, 4, 5, 6, and 7
[A,B][1,2] - A four pin bus with pins labeled A1, A2, B1, and B2

Grids Between Pins: This specifies the number of schematic grids between pins.

Bus Node Placement: This specifies where to place the bus connector. There are three options; top, middle, and bottom.

Wire Node Alignment: This specifies how the wires will emerge from the connector. The wires may be set to straight, slanted up, or slanted down.

Creating the Buses
The first bus added to the circuit will connect pins from the U3 stimulus source, the X1 7483A adder, and the X2 7448 decoder.

First, enable Bus mode . Then click in the schematic by the U3 component. The Bus dialog box will appear. Define the pin names as A[1:4] which will create a four pin connector with pins named A1, A2, A3, and A4. Set the Grid Between Pins as 1, Bus Node Placement as Bottom, and Wire Node Alignment as Straight. Hit OK. The bus connector will appear in the schematic. The connector can be moved or rotated just like any other component. Move and rotate the connector so the four wire nodes connect directly to the pins of the stimulus source. The bus node should be at the bottom of the device. The circuit should look like Figure 10.

Now we will add the bus connector for the X2 device. In Bus mode, click once again in the schematic to invoke the Bus dialog box. Define the pin names as A[1:4] so it shares the same signals as the first

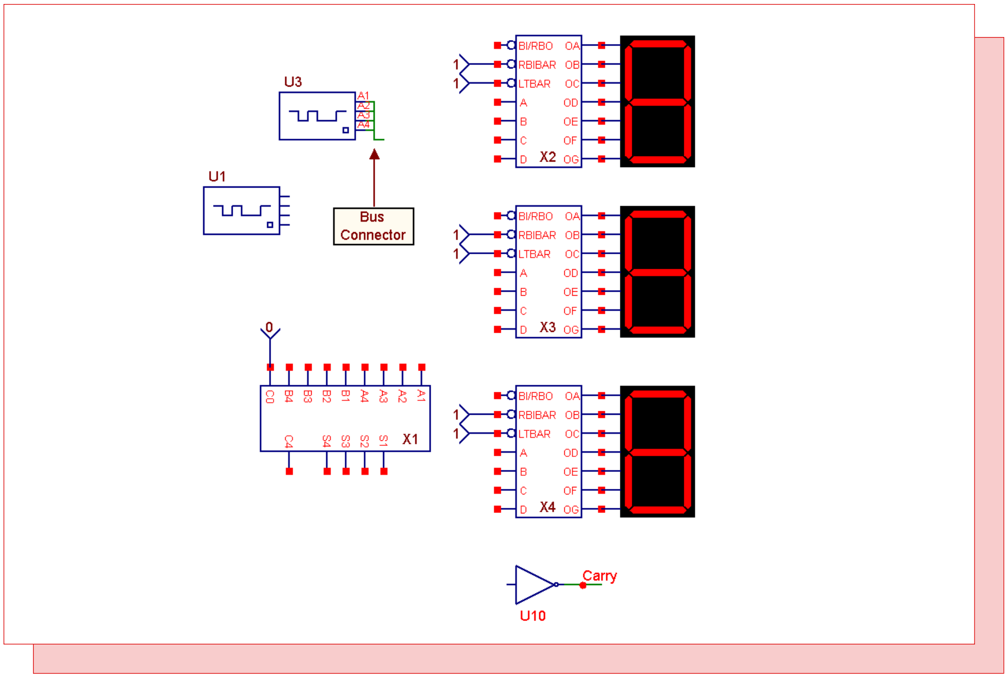


Fig. 10 - Circuit with the first bus connector added

connector. Set the Grid Between Pins as 2, Bus Node Placement as Middle, and Wire Node Alignment as Up. Hit OK. Rotate and move the component so that the wire nodes of the bus connector connect to the A, B, C, and D inputs of the decoder. The A1 pin of the connector should connect to the A input of the decoder.

The third connector for this bus will go to the X1 adder. In Bus mode, click in the schematic. Set the Grid Between Pins as 2, Bus Node Placement as Middle, and Wire Node Alignment as Straight. Hit OK. Rotate and move the component so that the wire nodes of the connector connect to the A1, A2, A3, and A4 inputs of the adder. The A1 pin of the connector should connect to the A1 input of the decoder.

Now it is time to wire the connectors together. Enable Wire mode . Drag a wire from the bus node of the first connector to the bus node of the second connector. The bus wire is displayed on the schematic with a thicker width than the standard wire to visually denote that it is a bus wire. Now drag another wire from the bus node of the third connector up until it connects to the existing bus wire. The bus wires connect in the same manner as standard wires do. Note that bus wires cannot connect to anything but the bus nodes of the bus connectors or to other bus wires. The results should appear similar to the circuit in Figure 11. Bus connector pins that share both the same pin name and the same bus wiring will be connected together. For example, the output of the U3 stimulus source that goes into the A3 pin is connected to the C input of the X2 decoder and the A3 input of the X1 adder.

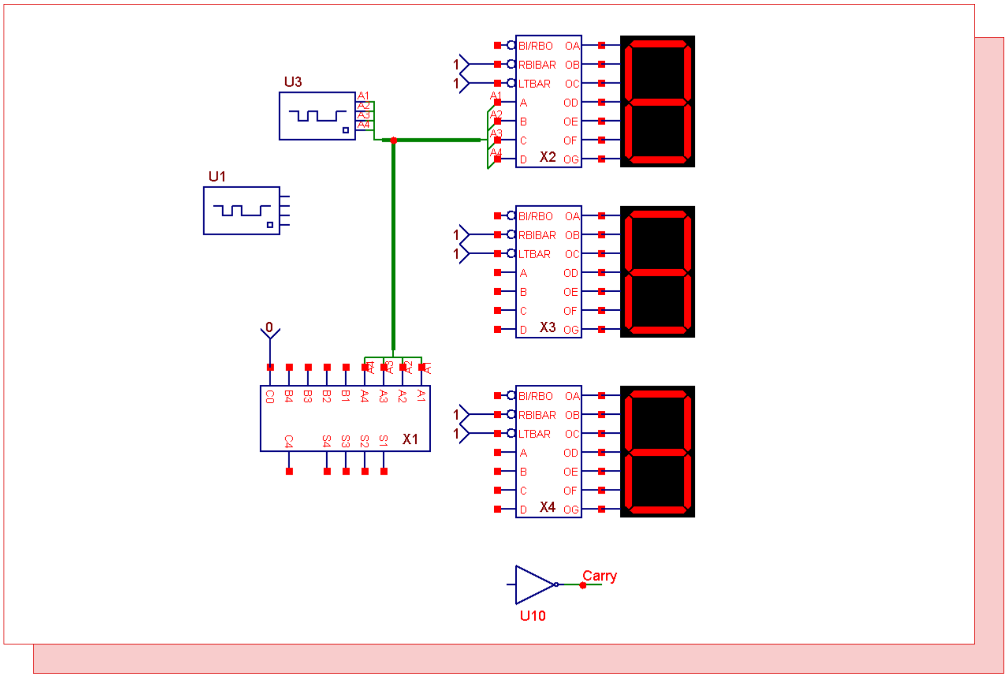


Fig. 11 - Circuit with the first bus added

The second bus will connect pins from the U1 stimulus source, the X1 7483A adder, and the X3 7448 decoder. The pin names for this set of connectors will be defined as B[1:4] which will create four pin connectors with pins named B1, B2, B3, and B4. Since these connectors will be on a different bus wire than the first set of connectors, we could have used A[1:4] again, but in practice, it is better to have a different set of pin names for each bus. Using the same technique and other settings used to create the first bus, the three bus connectors are then added to the schematic and wired together. The results appear in Figure 12.

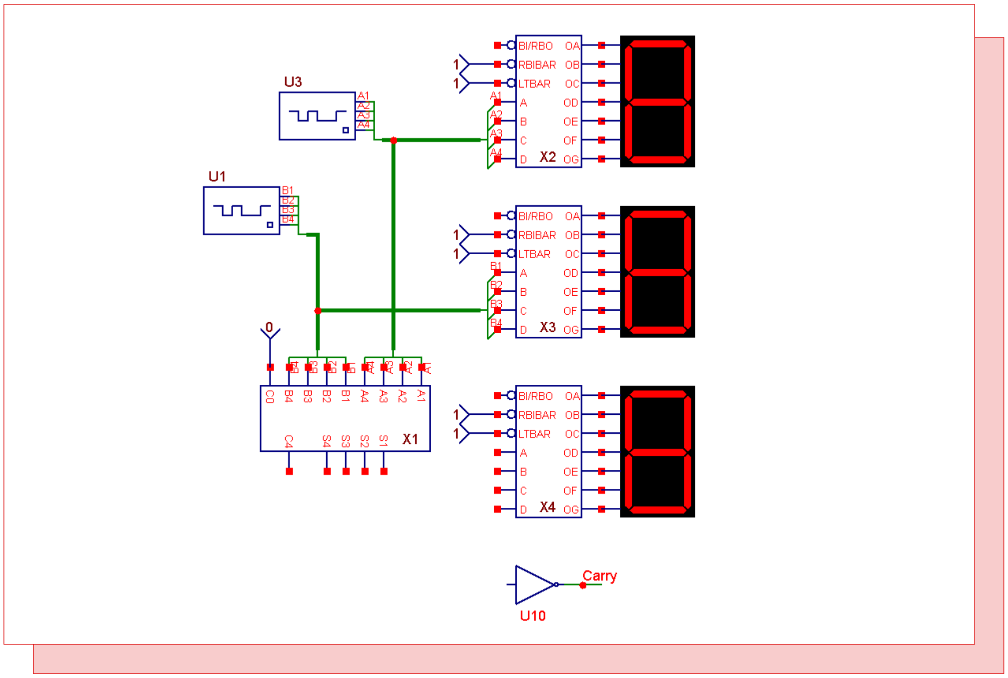


Fig. 12 - Circuit with the second bus added

The third bus will connect the outputs of the adder to the inputs of the X4 decoder and U10 inverter. In Bus mode, click below the adder. Define the pin names as O[1:5] which will create a five pin connector with pins named O1, O2, O3, O4, and O5. Set the Grid Between Pins as 2, Bus Node Placement as Middle, and Wire Node Alignment as Straight. Hit OK. In this case, we will move the connector so it is near the pins of the adder but not directly connected to it. Then we will use standard wires to connect the bus connector pins to the adder pins. The O1 pin should connect to the S1 adder output, and the O5 pin should connect to the C4 adder output. The circuit should look similar to Figure 13.

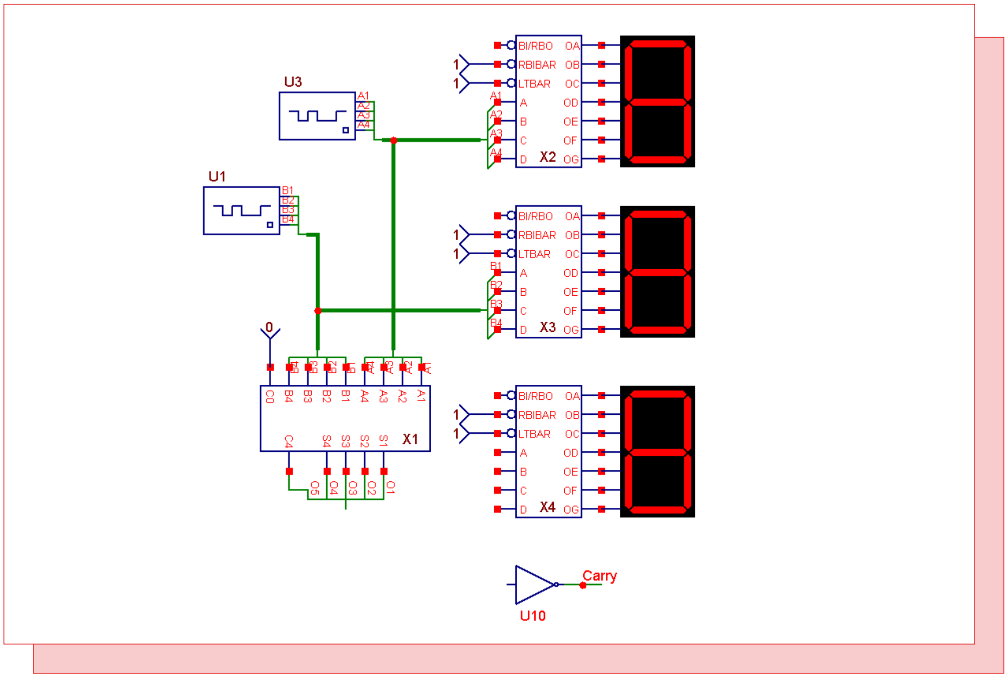


Fig. 13 - Circuit with the third bus started

Now we will add the bus connector for the X4 device. In Bus mode, click in the schematic to invoke the Bus dialog box. Define the pin names as O[1:4]. Since only the S1 to S4 outputs of the adder are needed as inputs for this decoder, the bus connector only uses the pin names that were connected to those signals with the other connector. Set the Grid Between Pins as 2, Bus Node Placement as Bottom, and Wire Node Alignment as Up. Hit OK. Rotate and move the component so that the wire nodes of the bus connector connect to the A, B, C, and D inputs of the decoder. The O1 pin of the connector should connect to the A input of the decoder.

The inverter just needs a single signal off of the bus. In Bus mode, click in the schematic. Define the pin names field as O5. This will input the C4 signal from the adder into the inverter once the bus is wired. The Grid Between Pins is not relevant with just a single pin so leave it at 2. Set the Bus Node Placement as Middle and Wire Node Alignment as Up. Hit OK. Rotate and move the component so that the O5 pin is connected to the inverter input.

Finally, wire the bus nodes of these three connectors together. The final schematic appears as below. Note that to further clean up the schematic, the pin names were hidden on the connectors for the first two buses. The pin name display is controlled by double clicking on a connector when in Select mode and enabling or disabling the Pin Names checkbox. Comparing the final schematic to the original schematic with the individual wires gives a clear demonstration on how using the bus can improve the readability of a schematic.

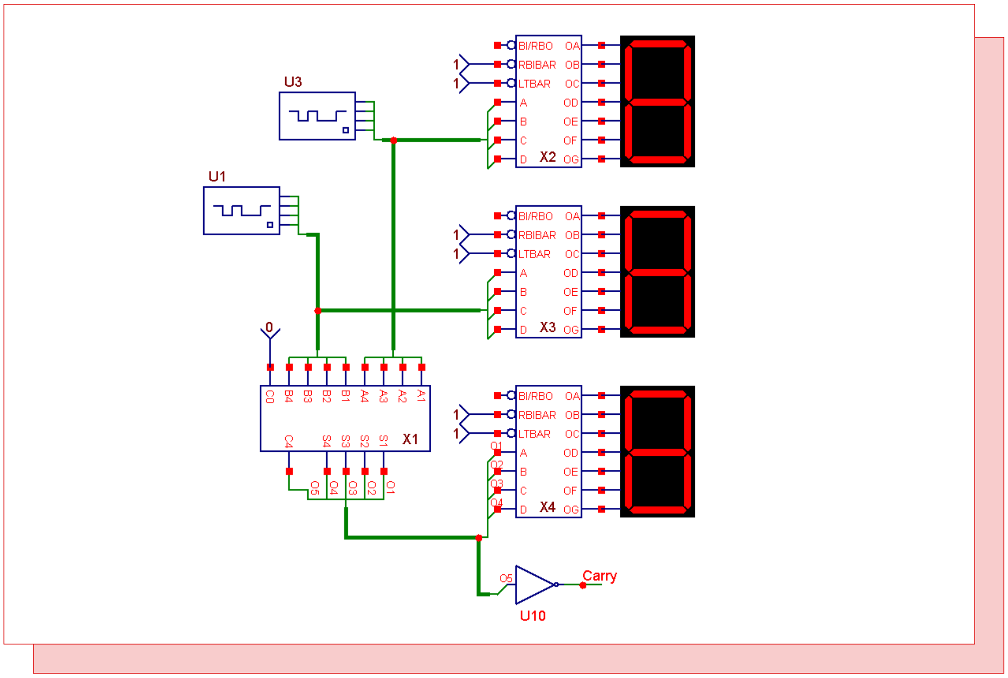


Fig. 14 - Circuit with all the buses added

Product Sheet

Latest Version numbers

Micro-Cap 9	Version 9.0.3
Micro-Cap 8	Version 8.1.3
Micro-Cap 7	Version 7.2.4

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