Computer Programming
with Python™, Multisim™ & TINA™/ 4E
Laboratory Manual

James M. Fiore

Laboratory Manual

for

Computer Programming

with Python™, Multisim™ & TINA™, Fourth Edition

by

James M. Fiore

Version 4.0.1, 23 July 2020

Published by James M. Fiore via dissidents

ISBN13: 979-8654193452

For more information or feedback, contact:

James Fiore, Professor
Electrical Engineering Technology
Mohawk Valley Community College
1101 Sherman Drive
Utica, NY 13501
jfiore@mvcc.edu

For the latest revisions, related titles, and links to low cost print versions, go to:
www.mvcc.edu/jfiore or my mirror sites www.dissidents.com and www.jimfiore.org

YouTube Channel: Electronics with Professor Fiore

Cover art, *Squarer for Bear*, by the author

Introduction

This laboratory manual is intended for use in an introductory computer programming course for electrical engineering technology students. It begins with a basic explanation of schematic capture and simulation tools and proceeds to the Python programming language. Python (version 3.X) was chosen for several reasons. First, it is a modern, open-source programming environment. Second, it has a relatively shallow learning curve meaning that new programming students can get up and running fairly quickly, yet the language is fairly deep and powerful. It is by no means a "toy" language. Third, it is free and multi-platform, available for Windows, Mac and Linux. This fourth edition is updated to Multisim 14. It also introduces alternate exercises using the TINA simulator from DesignSoft. A free version of TINA, TINA-TI, is available for download on the Texas Instruments web site: https://www.ti.com/tool/TINA-TI.

The programming applications presented tend to be electrical circuit based although some lean closer to quality control issues and a few are intended strictly as a way of stretching out and having some fun. As the language's designer and developers are fans of Monty Python, it is helpful to at least watch a few of their movies in order to appreciate the embedded jokes. Most of the exercises are designed to be completed in a single practicum period of two or three hours, however, a few are a bit more involved and will require more time (such as *Caerbannog* and *Functions and Files*).

Other laboratory manuals in this series include DC and AC Electrical Circuit Analysis, Semiconductor Devices (diodes, bipolar transistors and FETs), Operational Amplifiers and Linear Integrated Circuits, and Embedded Controllers Using C and Arduino. Texts are also available for DC and AC Electrical Circuit Analysis, Embedded Controllers (second edition), Operational Amplifiers (third edition), and Semiconductor Devices.

A Note from the Author

This work was borne out of the need to create a lab manual for the ET154 Computer Programming course at Mohawk Valley Community College in Utica, NY, part of our ABET accredited AAS program in Electrical Engineering Technology. Another important aspect was to come up with an affordable solution for the students. As both the programming language and the manual are free, this much is certainly covered. I am indebted to my students, co-workers and the MVCC family for their support and encouragement of this project. While it would have been possible to seek a traditional publisher for this work, as a long-time supporter and contributor to freeware and shareware computer software, I have decided instead to release this using a Creative Commons non-commercial, share-alike license. I encourage others to make use of this manual for their own work and to build upon it. If you do add to this effort, I would appreciate a notification.

"Without deviation, progress is not possible"

- Frank Zappa

Table of Contents

1A

Introduction to Multisim

Objective

The objective of this exercise is to become familiar with the Multisim™ electrical circuit simulation package by National Instruments in order to create simple schematics and perform basic simulations. The differences between virtual, real, and 3D components will be examined along with the use of virtual instruments to make simulated measurements. Please note that the precise look of the windows, menus and dialog boxes may be slightly different from that pictured depending on the version of Multisim that is being used. Regardless of appearance, the functionality remains. A student edition of Multisim is available at reduced cost.

Procedure

After logging into the computer, open Multisim. If a desktop shortcut is not available Multisim may be accessed via the Programs menu under the NI (National Instruments) menu item. This is a large program and may take a minute or two to load. Eventually, you will be greeted with something similar to the screen shown in Figure 1A-1. As the toolbars are customizable, the precise look of the program may be a little different from that shown. In general, there are a series of toolbars along the top. These are used to select different components and editing or viewing functions. Multisim's schematic capture facility is object based, that is, you "draw" a circuit by selecting predefined objects such as resistors and transistors, and drag them onto the workspace. They are then wired together using the mouse. You can zoom into or out of the workspace using the mouse.

By default, along the left edge is the Design Toolbox browser window. This may be closed to create more working room for the schematic. At the bottom is the tabbed Spreadsheet View that shows simulation data, components, etc. Clicking on entries will highlight the corresponding elements on the schematic worksheet. Along the right edge is a vertical toolbar that contains virtual instruments.

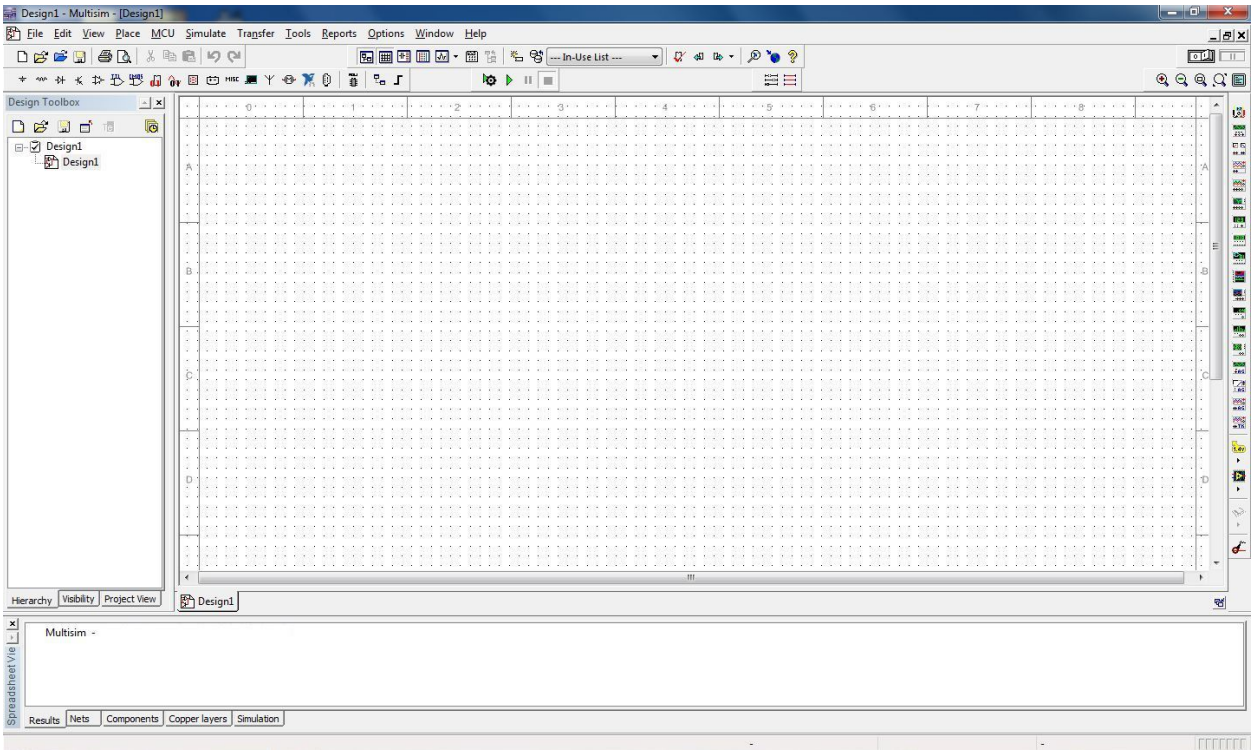

Figure 1A-1

Multisim uses three different kinds of components to create schematics. They are virtual basic and rated components, real (or manufacturer's) components, and 3D virtual components. Virtual and real components use the standard industry schematic symbols. By default, components with physical footprints (most real components) are colored blue and components without a physical footprint (virtual components) are colored black. In contrast, 3D components look more like photographs of the actual component. Examples are shown in Figure 1A-2. Although 3D components add a certain amount of color and realism to a circuit, Multisim contains few of them and thus are not used for simulations generally. We shall not discuss them further.

The difference bewteen virtual and real components is that real components reflect items from a manufacturer's database. The items include physical parameters, such as size and pinouts, which are required for designing proper printed circuit boards. Also, real behavioral models for semiconductor devices such as op amps will be more accurate than the virtual models. Finally, the values of real passive components (resistors, capacitors and inductors) are limited to the nominal values specified by the manufacturer. In contrast, the values of virtual components can be set to almost anything, however, there are no corresponding physical data. As a consequence, if a PCB is needed, virtual components are not the appropriate choice. In practice, if the goal is to create a production circuit, real components will be used. If the goal is to simulate a lab

exercise, virtual components will be used for the passives (rated resistors, capacitors and inductors) and reals will be used for the active components (transistors, diodes, op amps, etc.).

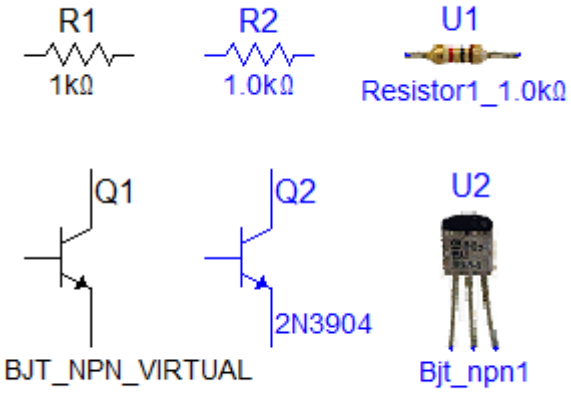

Figure 1A-2

To illustrate the use and editing of components, drag a DC voltage source onto the workspace. It will show up with a default voltage value and label. Double click the symbol. A dialog box will pop up next to it as shown in Figure 1A-3.

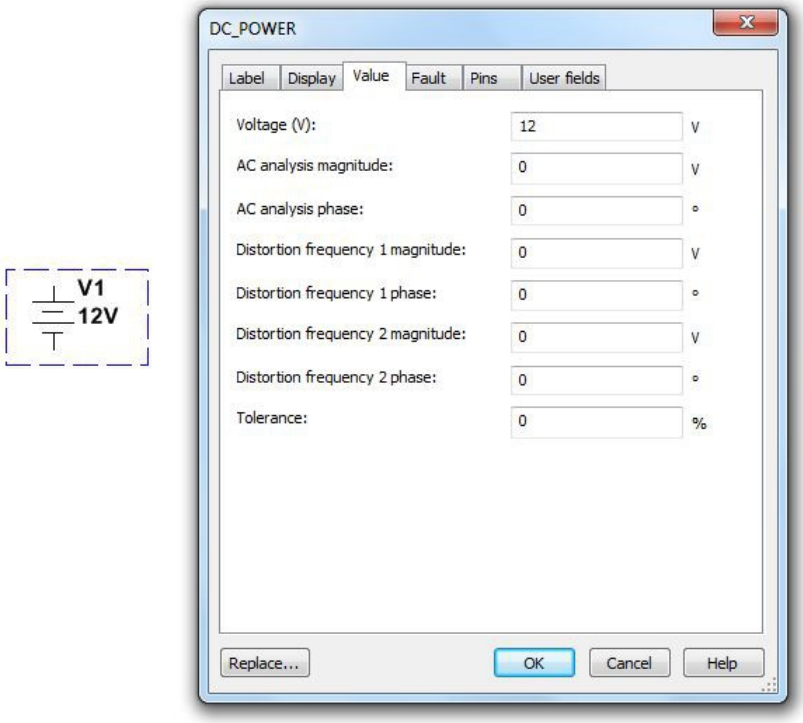

Figure 1A-3

From this dialog you can change a variety of attributes, the most important of which are the voltage value and the name. Double clicking any component will bring up a settings dialog box although the precise items contained within it will vary from component to component. For example, for a resistor there will be a resistance setting instead of a voltage setting. If you need to remove a component, simply select it and hit the Delete key.

Editing the position and orientation of a component is straight forward. Once the item is selected (shown by a surrounding dashed line) it may be moved using either the mouse or the cursor keys. If you need to move a group of components, the mouse may be used to select several items by clicking and then dragging the mouse over them. Every component within the selection box will be highlighted and will move as a group. Also note that it is possible to select the text labels of components and just move them. This can be handy if a label becomes obscured by a wire.

Components may also be rotated and flipped. These commands can be accessed from the main menu, however, it is handy to remember certain keyboard shortcuts (such as Ctrl-R, rotate right). You can also customize the tool bars and add these commands as their own buttons for easy access. The Customize dialog is shown in Figure 1A-4 and is accessed via the Options menu.

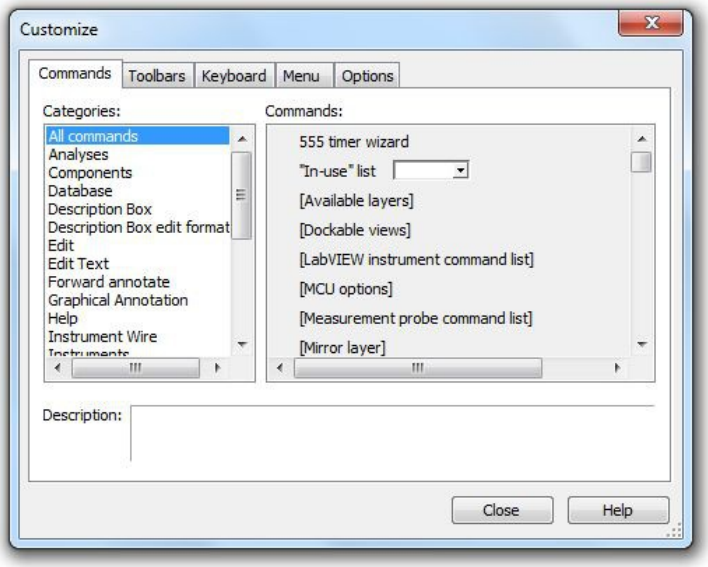

Figure 1A-4

Along with editing the components and customizing the tool bars, you may also customize the look of the work space. Go to the Options menu and select Sheet Properties. From here you can select a variety of color schemes for the components and wiring. You can also select which component items (labels, values, etc.) will be displayed. Fonts may be altered as well. Be fore-warned, it is possible to spend a great deal of time trying to make the work space look pretty

instead of doing truly productive work. Don't fall into this trap. Before we close this dialog, there is one important setting to note and that is the section labeled "Net Names". For now leave it as it is. We shall revisit this in the future.

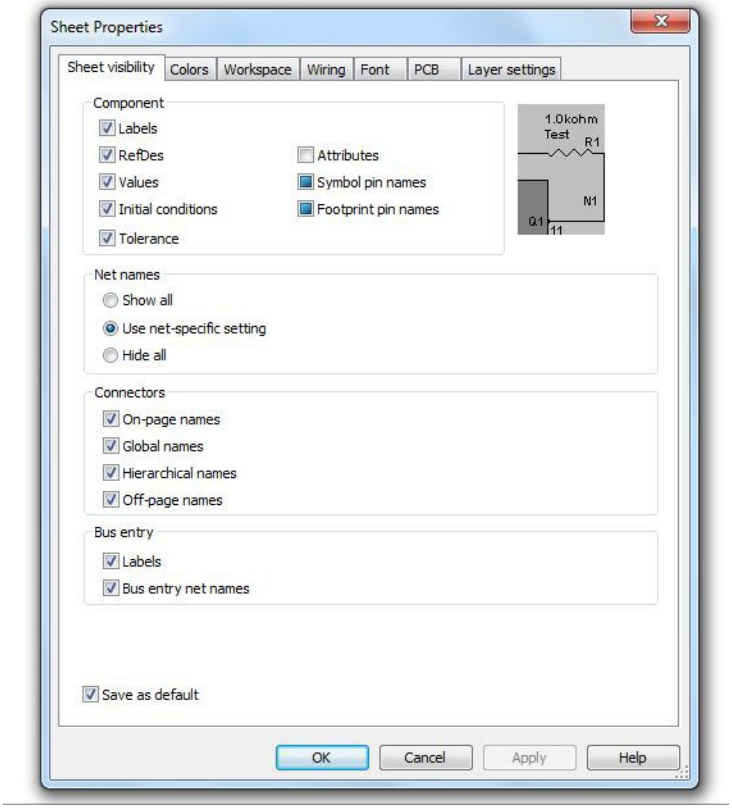

Figure 1A-5

OK, let's create a circuit and perform a simulation. You should already have a DC voltage source on your work space. From the virtual (blue) components tool bars, select two resistors and an earth ground symbol. We shall make a series loop of the three elements with the negative end of the power supply at ground. One resistor will need to be rotated 90 degrees (one horizontal and one vertical). In order to wire the items together, simply click on the free lead of one component and move to the desired lead of another. While moving, Multisim will draw a ghost line. Clicking on the second component will create a proper wire (by default, colored red). Wires are always drawn along the horizontal and vertical with 90 degree bends, not directly from point to point. This is the proper way to draw a schematic in the vast majority of cases.

It is possible to click on the middle of a wire in order to tie multiple items together. A small node circle will be drawn at the connection point. To delete a wire, click on it to select it. You will see a set of small box "handles" around it. To remove the wire, simply hit the Delete key. Note that

you can also move the wire with those handles if desired. In fact, it is possible to align wires at odd angles with these handles (again, this is not typical).

Note that if you move a component, Multisim will automatically move the wires along with it. You do not have to rewire it. Sometimes, especially if components are too close, the wire may be draw in an odd, looping form. The easiest solution is to simply separate the components a little more.

Once the components are in place and wired, double-click each component to set their values as shown in Figure 1A-6. If you're using power rated virtual resistors, also increase their power rating to one watt, each. We shall then add two the two voltmeters. Please note that it is perfectly acceptable to change the component values immediately after dragging them onto the work space; you don't have to wait until they are wired together. It is suggested, though, that you decide upon a particular workflow and stick to it otherwise the chances of forgetting to change components from their default values increases.

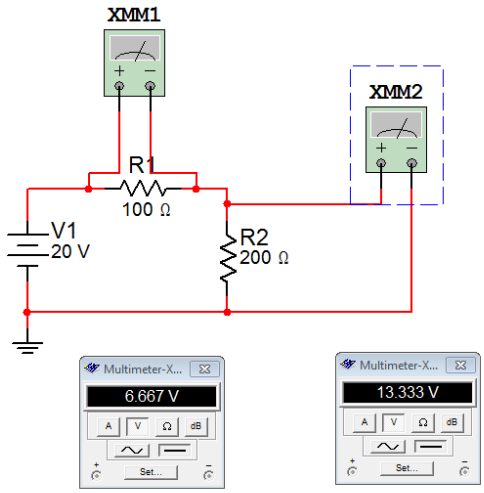

Figure 1A-6

To add the voltmeters (XMM1 and XXM2 in Figure 1A-6), select a virtual DMM from the top of the Instruments toolbar along the right edge of the screen. Drag this onto the work space and wire it across R1 as shown in Figure 1A-6. Grab a second virtual DMM and repeat the process for R2. Now double click on the DMMs. Two small windows will pop up (they might overlap each other, if so, move one over). Select the V (voltage) button and the straight line (DC) button on each of them. The circuit is now ready to perform a simulation.

To start a simulation, you can select the the rocker switch located in the upper right corner of the menu area. This is the virtual on-off switch. Alternately, you can select Run from the Simulate menu or use the green Run button on the simulation toolbar. In a moment, you should see voltages appear on the two virtual DMMs. **In order to edit the circuit, the simulation must be turned off.** So, if we wish to add or delete components, we must remember to "power down" the circuit just as we would in a real lab. To be safe, do this now.

Multisim has a wide variety of virtual instruments. Some of them are fairly simple such as the DMM just used. Other instruments are virtual recreations of real-world test instruments. For example, from the Instruments toolbar select the Tektronix Oscilloscope. You may place this anywhere on your schematic. Now double click on the small icon for this device (XSC1). A rather ornate window opens which appears to be the front panel of a Tektronix TDS 2024 oscilloscope, similar to the models you might see in electronics labs, right down to subtle shadows around the knobs! See Figure 1A-7. This has the advantage of immediacy (assuming you've used this type of oscilloscope before), however, it is not the ultimate way to perform a simulation. We will look at even more powerful and flexible ways of creating simulations in the next exercise. For now, you may wish to delete this new instrument from the work space and then save the existing schematic. Remember to always save to either your student account or to a USB drive. Never save directly to the hard drive in lab.

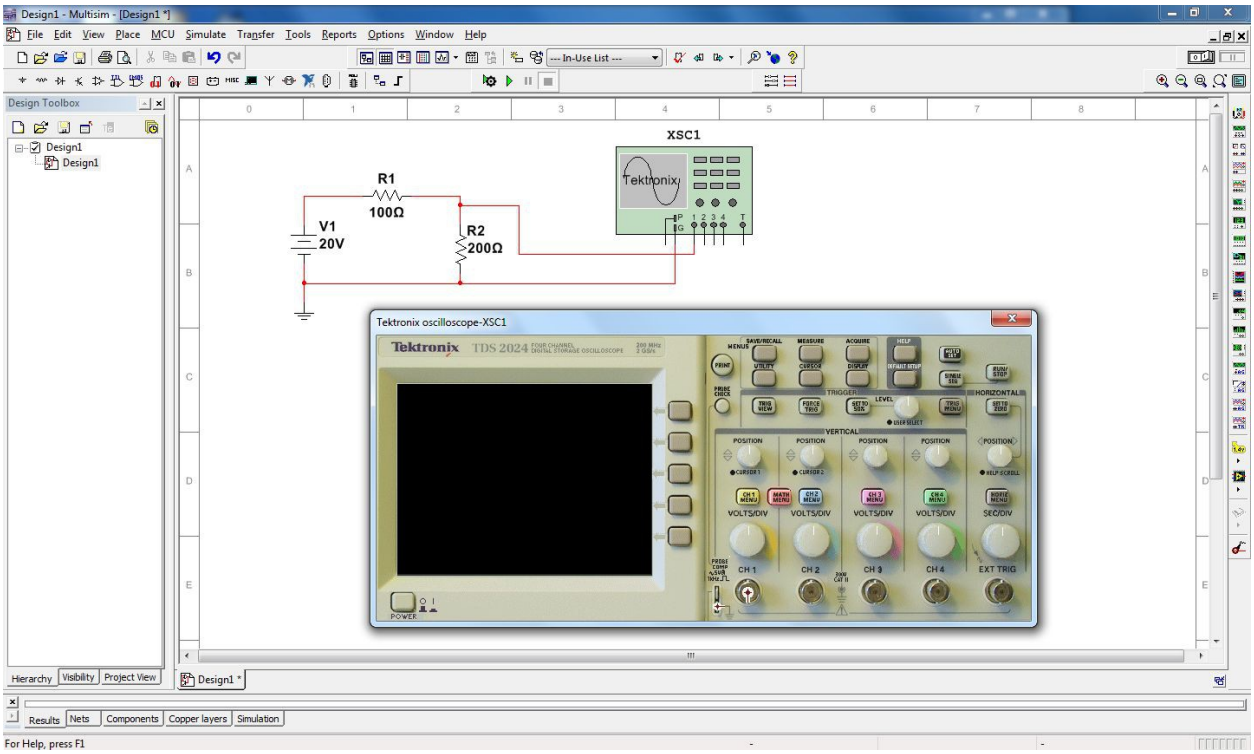

Figure 1A-7

There are three very good “every day” uses for Multisim during your studies: First, it is a very handy tool for verifying lab results. That is, you can recreate a lab circuit, simulate it, and compare the simulation to both your theoretical calculations and lab measurements. Second, it is a handy tool for checking homework if you get stuck on a problem. Third, it is convenient for the creation of schematics, for example, for a lab report. An easy way to do this is to draw the circuit, capture the screen image (Windows key+Print Screen key copies it into the clipboard) and then paste the image into your favorite image manipulation program. From there you can crop it, change contrast, etc. as needed and then paste the modified image into your lab report.

At this point you may wish to experiment a bit by rewiring the circuit to measure current or to try building new circuits. As with any tool, continued practice will hone your skill. Once you are done and have saved the file (the extension should be .msX where X is the Multsim version number), close Multisim and then shut down the computer.

1B

Introduction to TINA

Objective

The objective of this exercise is to become familiar with the TINA™ electrical circuit simulation package in order to create simple schematics and perform basic simulations. TINA stands for Toolkit for Interactive Network Analysis and is produced by DesignSoft. The differences between US (ANSI) and Euro (DIN) schematic symbols, and 3D components will be examined along with the use of virtual instruments to make simulated measurements. Please note that the precise look of the windows, menus and dialog boxes may be slightly different from that pictured depending on the version of TINA that is being used. Regardless of appearance, the functionality remains. A free version of TINA, known as TINA-TI, is available from the Texas Instruments web site at https://www.ti.com/tool/TINA-TI

Procedure

After logging into the computer, open TINA. If a desktop shortcut is not available TINA may be accessed via the Programs menu under the TINA menu item. You will be greeted with something similar to the screen shown in Figure 1B-1. Depending on the version, the precise look of the program may be a little different from that shown (TINA-TI). There are two toolbar strips immediately below the menus. The topmost toolbar contains the usual File Open, Save, and similar commands, along with buttons for text insertion, zoom level and the like. At the extreme right end of this strip is search box for electronic components. The lower strip is made up of a set of tabbed toolbars for electronic components, measurement instruments and other devices. Selecting a given tab (e.g., *Basic*, *Meters*, *Semiconductors*, etc.) shows the toolbar for that set of items.

TINA's schematic capture facility is object based, that is, you "draw" a circuit by selecting predefined objects such as resistors and transistors, and drag them onto the workspace. They are then wired together using the mouse. You can zoom into or out of the workspace using the mouse or the zoom toolbar button.

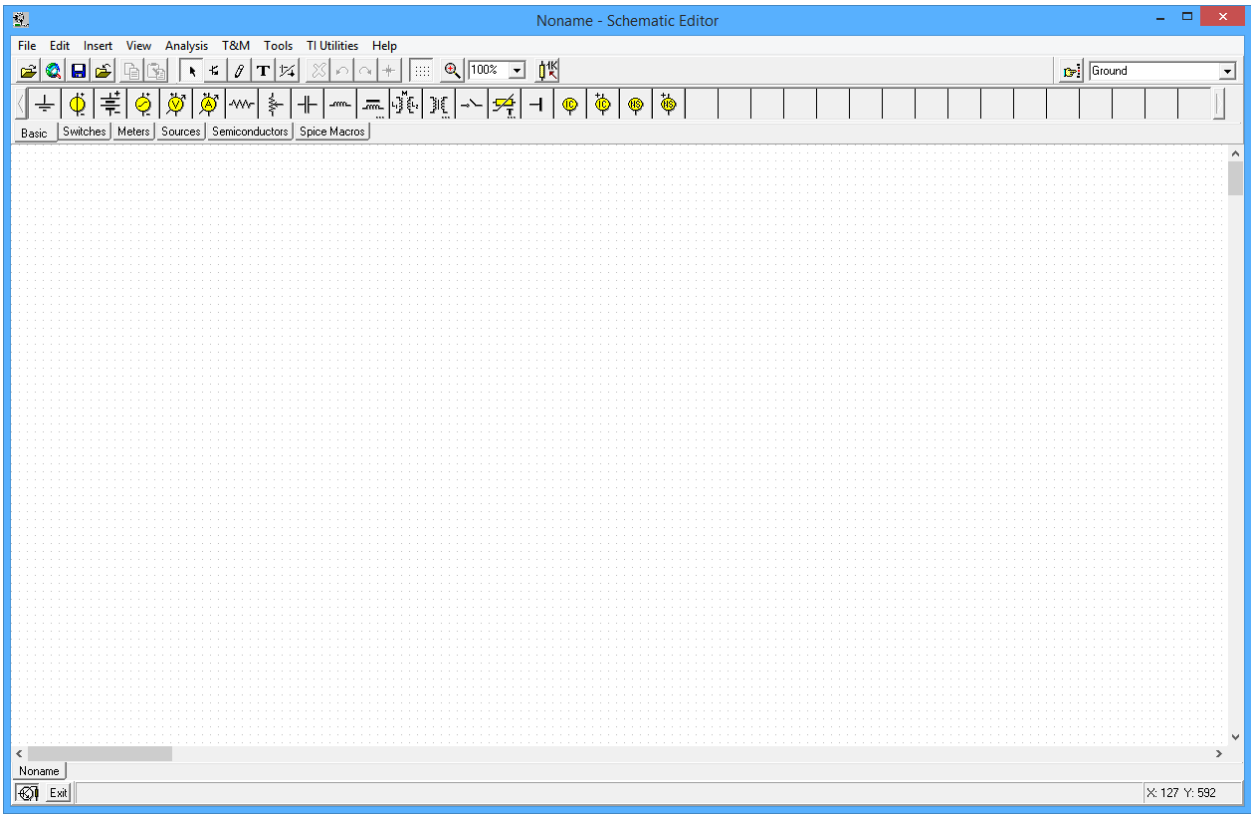

Figure 1B-1

TINA will draw schematics using one of three formats. The first style uses the ANSI (North American) standard for schematic symbols. The second style uses the DIN (Euro) standard. The final style is called "3D" where components look more like photographs of the actual component. Examples are shown in Figure 1B-2 using ANSI, DIN and 3D. You can set the preferred format via the View->Options menu item. More on this in a moment.

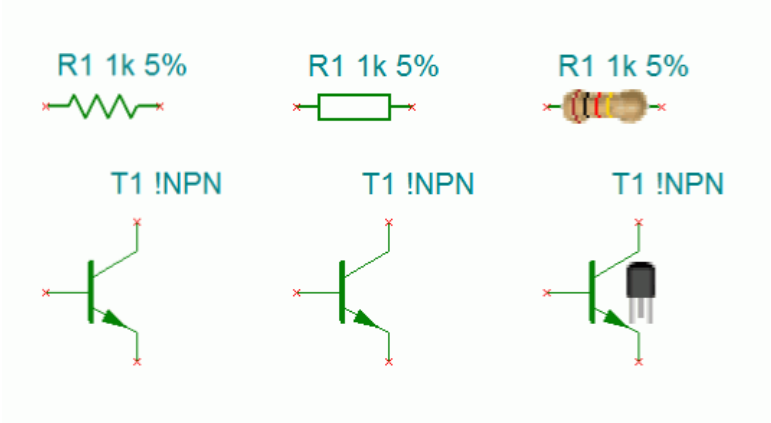

Figure 1B-2

Generally, we do not use the 3D style for most work. It is, however, useful for newcomers as a way of reinforcing and practicing use of the resistor color code, and for general package identification (such as the TO-92 transistor case pictured in Figure 1B-2). TINA will show the proper color sequence for resistors using either two or three significant digits along with a proper tolerance band (i.e., four or five bands). A comparison is shown in Figure 1B-3.

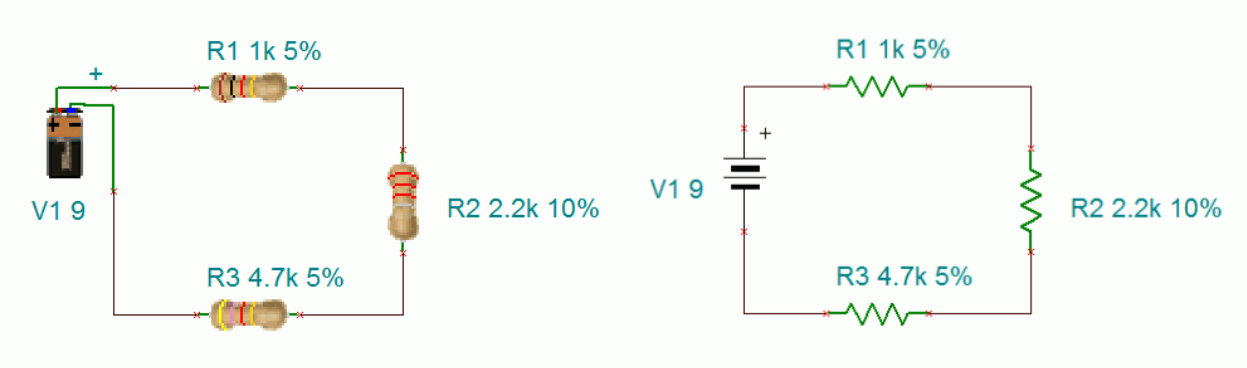

Figure 1B-3

Along with the appropriate electronic parameters of a device, physical data regarding shape and pinout are required if a printed circuit board (PCB) will be created. For example, a particular device such as an operational amplifier may be available in several different package styles including through-hole or surface mount variants. Our goal here is to become familiar with capturing the schematic and performing simulations, such as verifying a laboratory exercise. Consequently, we shall not concern ourselves with the needs of PCB layout at this point.

To illustrate the use and editing of components, select the *Basic* tab and drag a DC voltage source (third item) onto the workspace. It will show up with a default voltage value and label. Double click the symbol. A dialog box will pop up next to it as shown in Figure 1B-4.

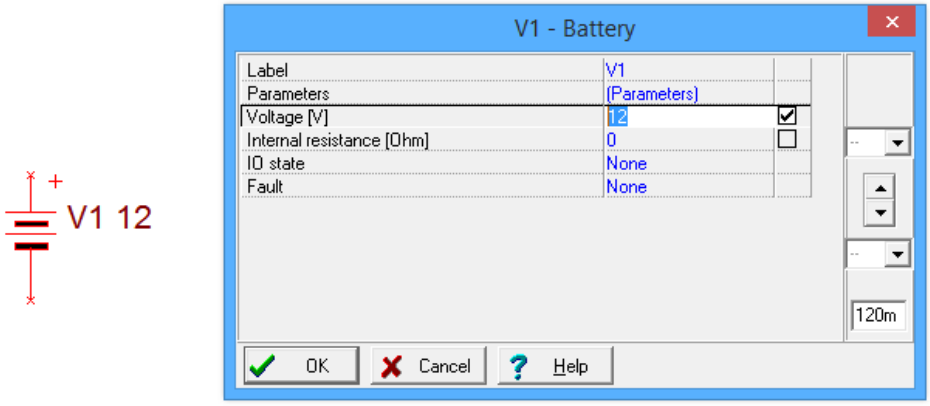

Figure 1B-4

From this dialog you can change a variety of attributes, the most important of which are the voltage value and the label. Double clicking any component will bring up a settings dialog box although the precise items contained within it will vary from component to component. For example, for a resistor there will be a resistance setting instead of a voltage setting. If you need to remove a component, simply select it and hit the Delete key.

Electronic components can be placed into two broad categories; generic parts and more accurate "real world" models typically generated by the manufacturer of that part. This is particularly true with complex components or devices. For example, Figure 1B-5 shows a pair of NPN transistors. The one labeled "!NPN" is a simple generic model. The lower, highlighted transistor references a specific part number, in this case the 2N3904, a popular general purpose device.

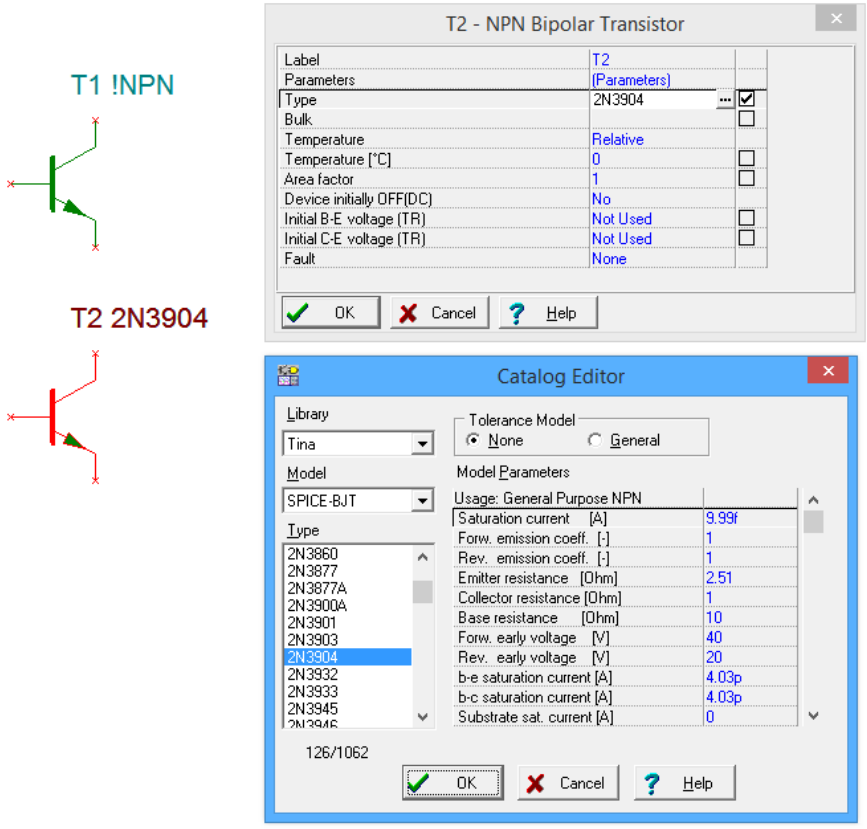

Figure 1B-5

Double clicking the generic transistor opens up its settings dialog box. Initially, under Type, it will say "NPN", referring to the generic. Next to this is a small box labeled with an ellipsis (...). Clicking on the ellipsis will open a second settings box, the Catalog Editor. From here, a specific manufacturer's part can be selected from the scrolling list on the left. If need be, specific parameters can be altered in the list on the right.

Remember, whenever you see the small ellipsis box, clicking on it will open another settings box so that you can specify further details.

Editing the position and orientation of a component is straightforward. Once the item is selected (highlighted in red by default) it may be moved using either the mouse or the cursor keys. If you need to move a group of components, the mouse may be used to select several items by clicking and then dragging the mouse over them. Every component within the selection box will be highlighted and will move as a group. Components may also be rotated and flipped. These commands can be accessed from the main menu, however, it is handy to remember certain keyboard shortcuts (such as Ctrl-R for rotate right, or clockwise). Also note that it is possible to select the text labels of components and move or rotate them independently of the component. This can be handy if a label becomes obscured by a wire. To do so, make sure the component is not selected (unhighlighted) and then click directly on the label. It can now be moved or rotated.

Along with editing the components, you may also customize the look of the work space. From the file menu, select Page Setup. This opens the dialog box shown in Figure 1B-6 and allows you to specify the sheet size for the schematic and related properties.

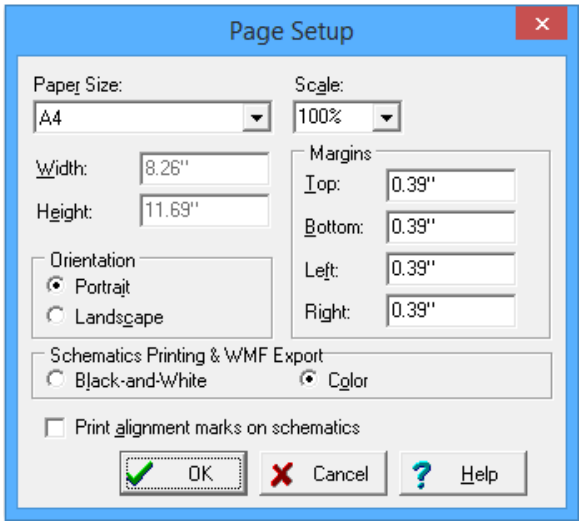

Figure 1B-6

Other attributes, such as whether or not to show the grid, component values, labels, and the like can be selected from the View menu. This is shown in Figure 1B-7. Of particular importance is the final item in this list, Options, which was mentioned earlier. Selecting Options brings up the Editor settings box shown in Figure 1B-8.

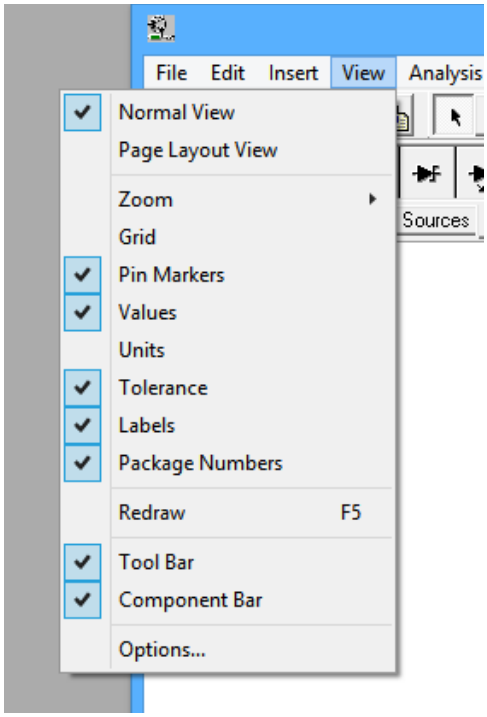

Figure 1B-7

The Editor options include the aforementioned ANSI, DIN and 3D styles. It also includes the base function for AC (it is recommended this be set to sine). Further, you can adjust the color scheme for your workspace and a few other global parameters.

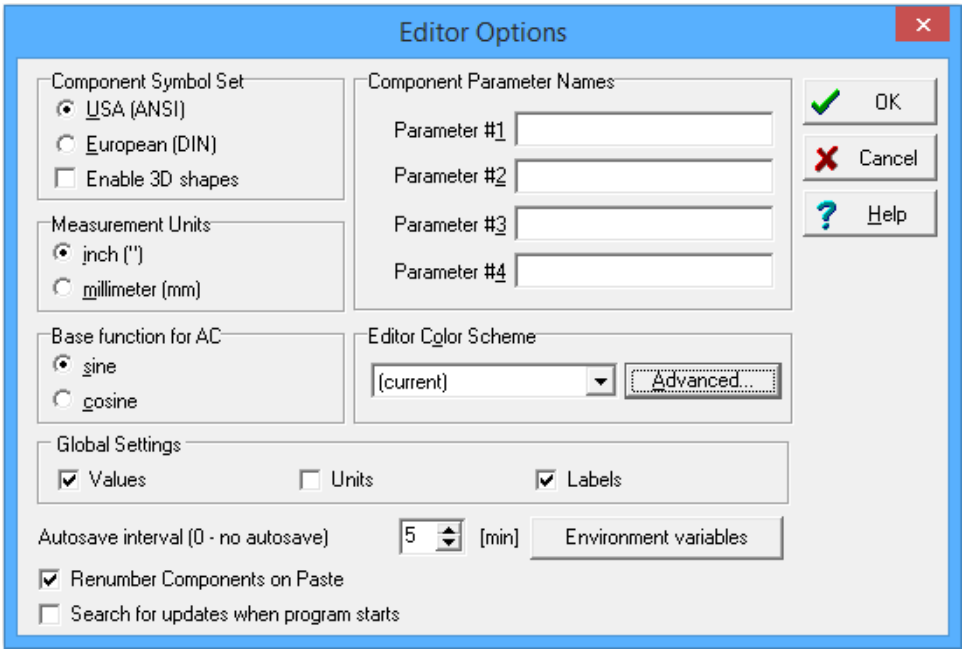

Figure 1B-8

OK, let's create a circuit and perform a simulation. If you have been experimenting, remove any components on your workspace (select with the mouse and then press the Delete key). From the *Basic* components tab, select a DC voltage source, two resistors and an earth ground symbol. We shall make a series loop of the three elements with the negative end of the power supply at ground. One resistor will need to be rotated 90 degrees (one horizontal and one vertical). In order to wire the items together, move the mouse to the end of one component you wish to wire. The connection ends are denoted with a small red x. When the mouse cursor is close, it will turn into a pen shape. Simply click on the lead of the component and move the mouse to the desired lead of another component. While moving, TINA will draw a red line. Clicking on the second component will create a proper wire (by default, colored black). Wires are always drawn along the horizontal and vertical with 90 degree bends, not directly from point to point. This is the proper way to draw a schematic in the vast majority of cases.

To delete a wire, click on it to select it. You will see a set of small "handles" on it. To remove the wire, simply hit the Delete key. Note that you can also move the wire with those handles if desired.

Note that if you move a component, TINA will automatically move the wires along with it. You do not have to rewire it. If a component is later deleted, this will result in "dangling wires" which also should be deleted.

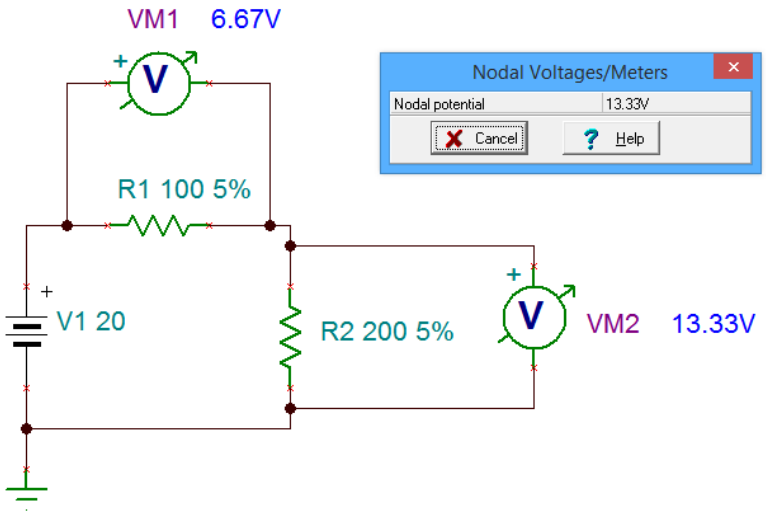

Figure 1B-9

Once the components are in place and wired, double-click each component to set their values as shown in Figure 1B-9. We shall then add two the two voltmeters (fifth item in the *Basic* tab). Please note that it is perfectly acceptable to change the component values immediately after

dragging them onto the work space; you don't have to wait until they are wired together. It is suggested, though, that you decide upon a particular workflow and stick to it otherwise the chances of forgetting to change components from their default values increases.

The circuit is now ready to perform a simulation.

From the Analysis menu, select DC Analysis->Calculate nodal voltages. In a moment, the values of the voltages will be printed next to the voltmeters. Also, a small window will pop up. With this window, you can investigate particular node voltages or component voltages and currents. You will notice that the mouse cursor will have changed shape into a measurement probe. You can click on a node to see a voltage with respect to ground (a node is denoted with a small, solid black circle on the wire where items connect, such as where a meter connects to a wire). Further, if you click on a component, the voltage across it and the current through it will be displayed in the little pop up window. This is a nice interactive feature.

TINA also has a wide variety of virtual instruments. These can be accessed via the T&M (Test and Measurement) menu. For example, Figure 1B-10 illustrates a virtual oscilloscope. This has the advantage of immediacy (assuming you've used an oscilloscope before), however, it is not the ultimate way to perform a simulation. We will look at even more powerful and flexible ways of creating simulations in the next exercise.

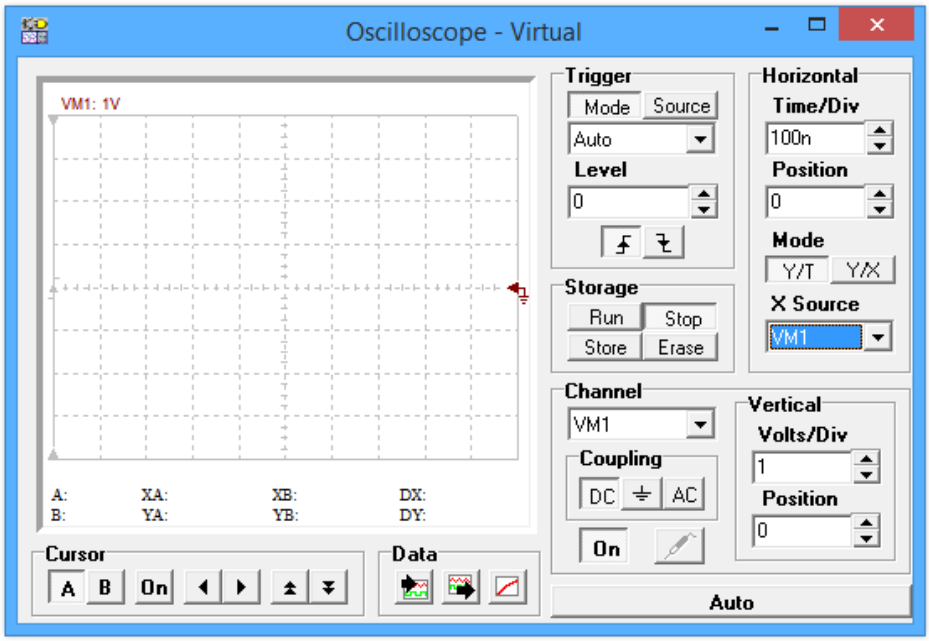

Figure 1B-10

For now, you may wish to delete this new instrument from the work space and then save the existing schematic. Remember to always save to either your student account or to a USB drive. Never save directly to the hard drive in lab.

There are three very good "every day" uses for TINA during your studies: First, it is a very handy tool for verifying lab results. That is, you can recreate a lab circuit, simulate it, and compare the simulation to both your theoretical calculations and lab measurements. Second, it is a handy tool for checking homework if you get stuck on a problem. Third, it is convenient for the creation of schematics, for example, for a lab report. An easy way to do this is to draw the circuit, capture the screen image (Windows key+Print Screen key copies it into the clipboard) and then paste the image into your favorite image manipulation program. From there you can crop it, change contrast, etc. as needed and then paste the modified image into your lab report.

At this point you may wish to experiment a bit by rewiring the circuit or to try building new circuits. As with any tool, continued practice will hone your skill. Once you are done and have saved the file (the extension should be .TSC), close TINA and then shut down the computer.

2B

Multisim Extensions

Objective

The objective of this exercise is to become more familiar with the Multisim electrical circuit simulation package in order to use more generalized simulations via the Grapher Window.

Procedure

In previous work we have examined the basic functionality of Multisim, namely basic schematic capture functions such as component selection, placement, and parameter editing, along with simple simulations using virtual instruments such as a Digital Multimeter (DMM) to measure DC voltage. While virtual instruments are quick and easy to use, and offer some amount of familiarity, they are necessarily limited in other aspects. Some of the issues with virtual instruments include:

1. Limited measurements per unit, for example a single measurement for a DMM, requiring multiple units for multiple measurements.

2. The need to rewire the instruments (and hence the schematic) in order to take different readings.

3. Excessive amount of work space area obscured by the instrument(s) with accompanying clutter.

4. No convenient way of storing and recalling prior simulations.

5. No convenient means of exporting simulated measurement data to other programs.

To address these issues Multisim allows non-instrument simulations through the use of the Grapher window. The Grapher is a single, general purpose window that presents simulation data

in both text and graphical form, as appropriate. Large amounts of data may be displayed simultaneously. The display itself is highly customizable (titles, axis ranges, colors, fonts, etc.). Each simulation is given its own sheet or tab and these may be saved for future reference. To top it off, nothing needs to be wired into the existing schematic and the Grapher may be minimized for greatest access to the schematic. The Grapher may be used for both DC and AC simulations, and includes interactive measurement tools for certain simulation types.

Because nothing is wired to the schematic, some means of identifying the measurement points is required. Multisim, like virtually all other simulation programs, does this through the use of numbered *nodes*. A node is simply a connection point in the circuit. Ground is always node number 0. The numbers increase as connections are made. Consequently, if the same circuit is entered by two different people who wire the components in a different order, the node numbers will not be the same between the two circuits. This is generally not a problem. As a side note, Multisim will not back-fill nodes that have been deleted. For example, if a circuit has been wired up to node ten and then node five is deleted, the next node to be wired will be numbered eleven, not five. Node number five will simply remain unused. Again, this is not a problem. It is important to note that, internally, Multisim always refers to connections via their node numbers, whether you use them or not.

When using virtual instruments, node numbers can add a certain amount of clutter to the schematic, consequently they are not shown by default. To show node numbers, select Sheet Properties from the Options menu. The dialog box of Figure 2A-1 opens. This dialog box should be familiar as it was used in the prior exercise to set schematic colors and the like.

If the dialog box looks a little different from Figure 2A-1, make sure that the Sheet Visibility tab is selected (leftmost tab). The center area is labeled Net Names. This controls whether or not node numbers will be visible on the schematic. To make node numbers visible, select Show All. Select OK to close the dialog box. We will now proceed to create a simple schematic.

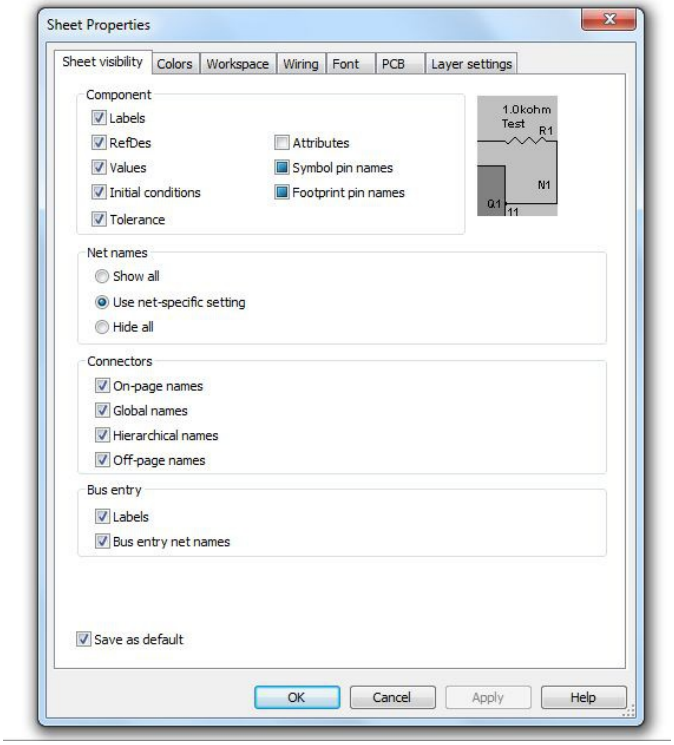

Figure 2A-1

Using virtual components, drag a DC voltage source, an earth ground and two resistors onto the work space. Connect them in a series loop and edit the component values to 20 volts for the source and 4000 and 6000 ohms for the two resistors. Once completed, the circuit should look something like Figure 2A-2.

Note that ground is node zero. This is always the case. If the components were wired from left to right, the other two nodes will be one and two as shown. If the wiring steps were reversed, the node numbers will be reversed.

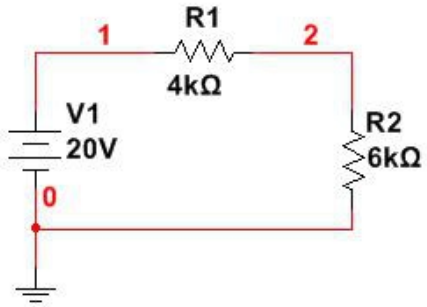

Figure 2A-2

We shall perform a simulation to inspect a few DC voltages. Instead of wiring in virtual DMMs, we shall select the Analyses and Simulation item under the Simulate menu (see Figure 2A-3).

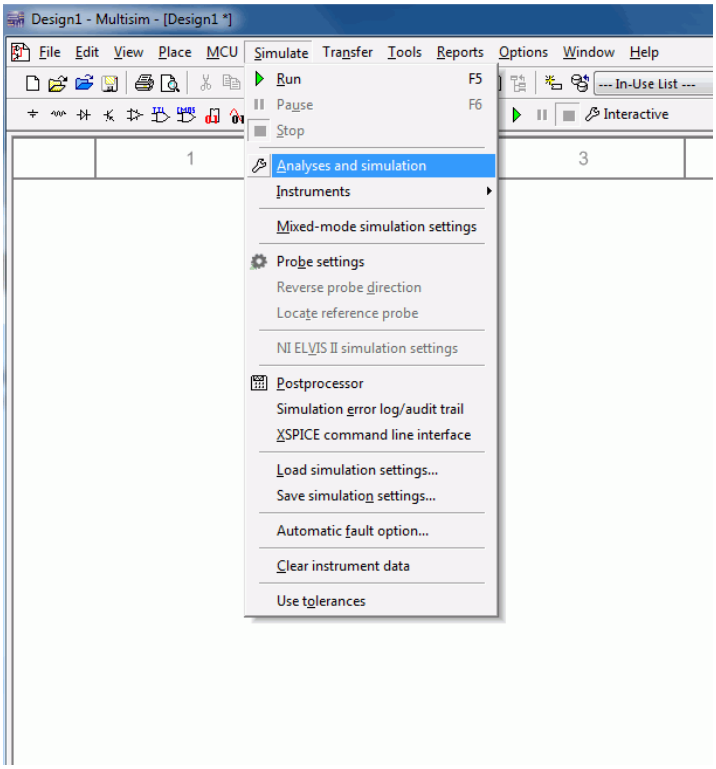

Figure 2A-3

A dialog box will open (see Figure 2A-4). There are well over a dozen analyses available. To find DC voltages or currents in our circuit, select DC Operating Point from the Analysis list. The dialog box will update and now contains two list areas. On the left will be a listing of available node voltages along with branch currents. The right side contains those items currently chosen for analysis. Node voltages will be shown with a V prefixed to the node number as in V(1). Branch currents will use an I prefix. In version 13 and earlier of Multisim the Analyses menu has a sub-menu of analysis choices. Simply select the analysis type from the sub-menu and then continue with the settings as described.

Select both nodes one and two on the input list with the mouse. To transfer them to the output list, select the Add button between the two lists. For now, this is all we need to concern ourselves with but it is worth mentioning that it is possible to create mathematical expressions of variables as well. For example, one node voltage could be subtracted from another in order to find the

voltage across a single component or a group of components. Further, a node voltage could be squared and divided by a circuit resistance to determine the power dissipation.

Select the Run button at the bottom of the dialog to start the simulation.

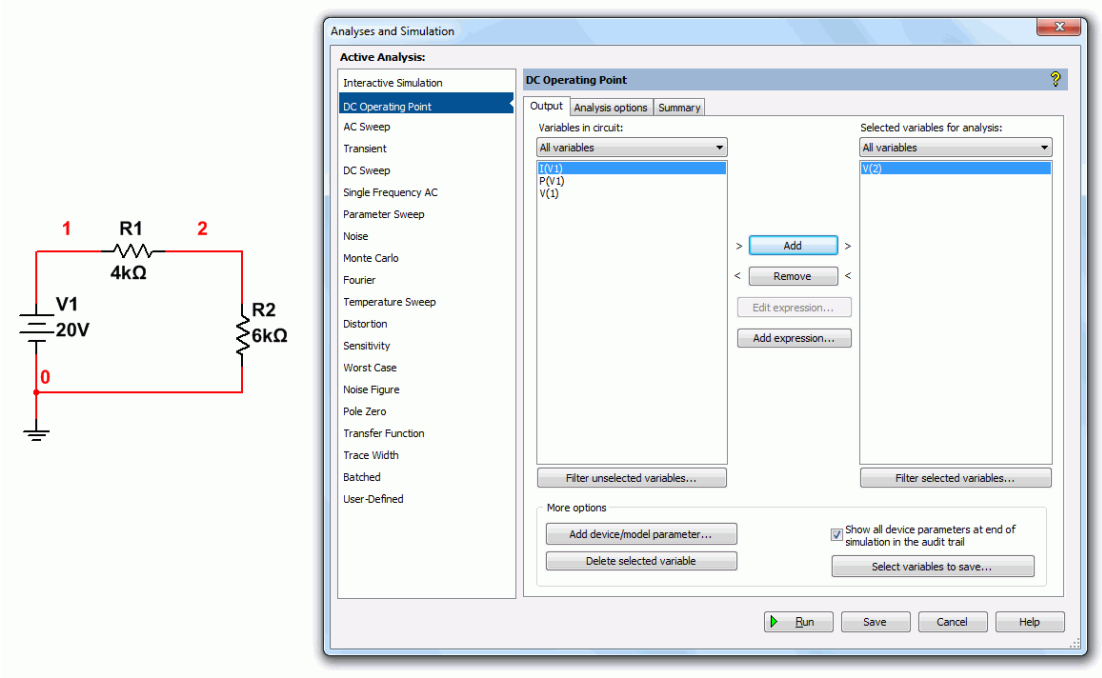

Figure 2A-4

After a moment, the Grapher window will appear as shown in Figure 2A-5. For this simulation, the Grapher simply lists the node number and the corresponding voltage in two adjacent columns. This is a nice, compact display, much nicer than the individual DMM windows, especially if a large number of voltages are needed.

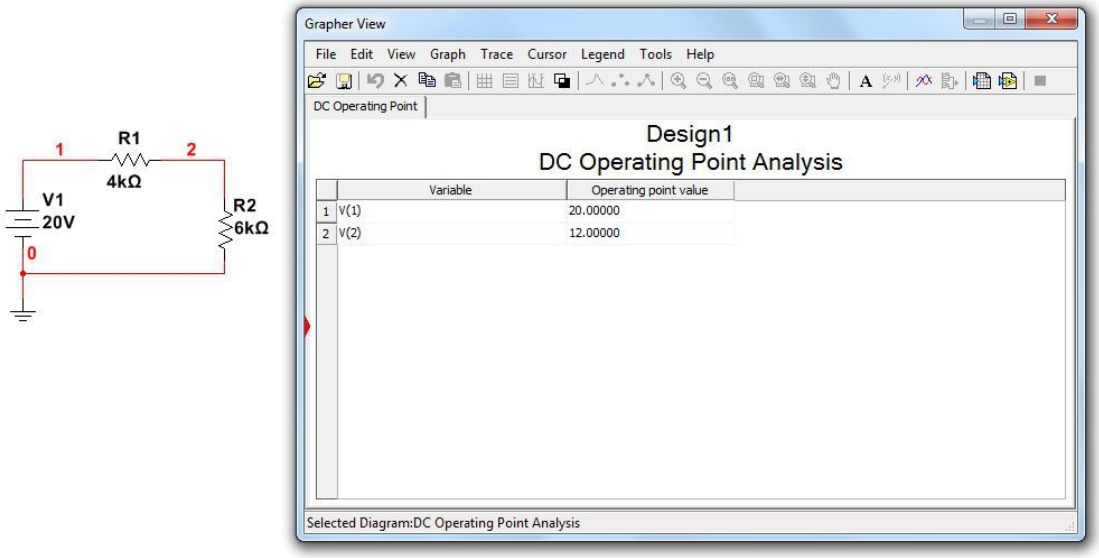

Figure 2A-5

Along the top of the Grapher window is a toolbar area for saving, cutting, customizing, and so forth. As useful as this is, the Grapher comes into its own when used with AC simulations.

Close the Grapher and modify the circuit to replace the DC source with an AC signal source as shown in Figure 2A-6. Make sure that the new source is set to 20 volts with a frequency of 1000 Hz (1 kHz).

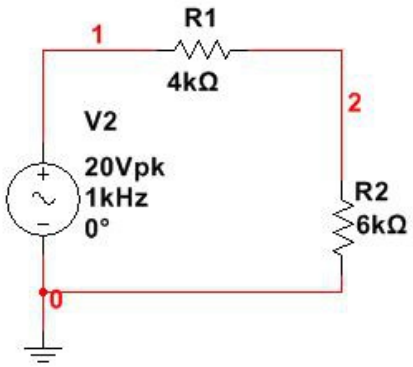

Figure 2A-6

We are going to examine the node voltages once again, however, this time the voltages will vary over time. If we were to use virtual instruments for this we'd choose once of the virtual oscilloscopes. In this case, however, we shall choose Transient Analysis from the Analyses and Simulation menu. A dialog box will open like that of Figure 2A-7.

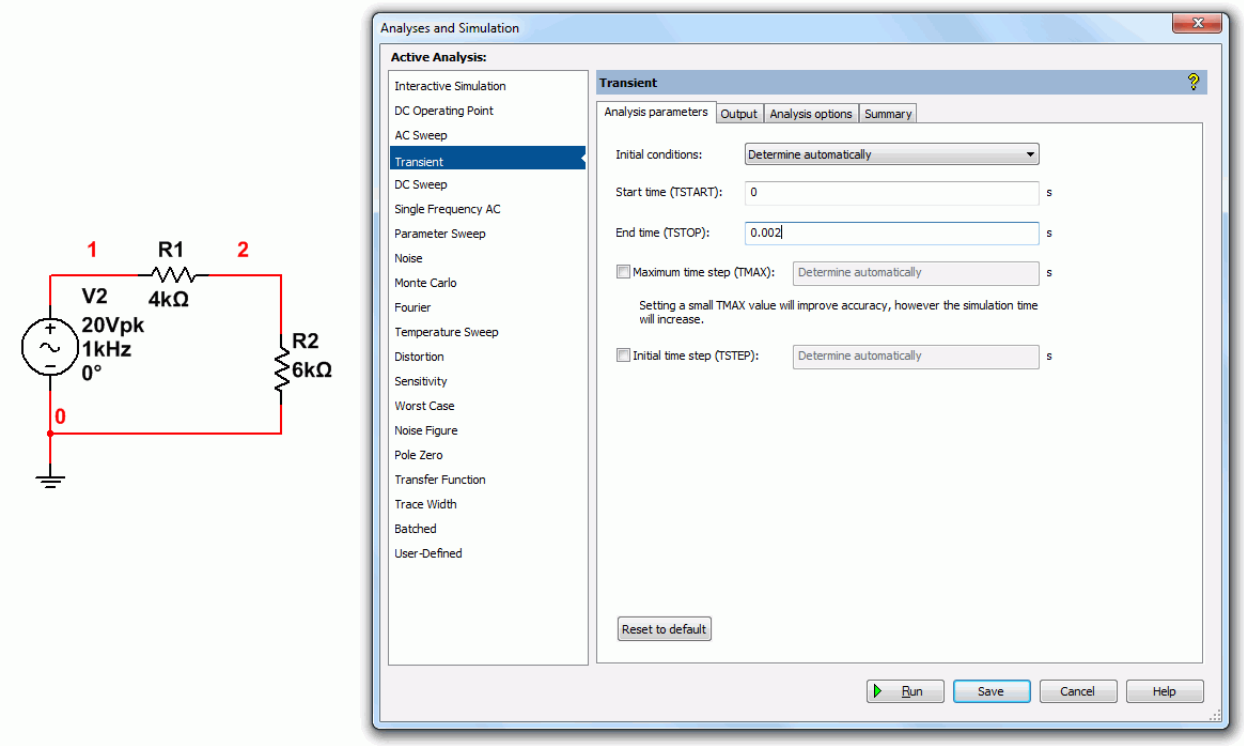

Figure 2A-7

The first thing we need to do is select the range of times we wish to see. Set the Start time to 0 and the End time to two milliseconds (0.002 seconds). Now select the Output tab. This is the same tab you saw back in Figure 2A-4. Make sure that voltage nodes one and two are in the output analysis (right side) list. Now select Simulate. The Grapher will reopen with a display similar to that of Figure 2A-8.

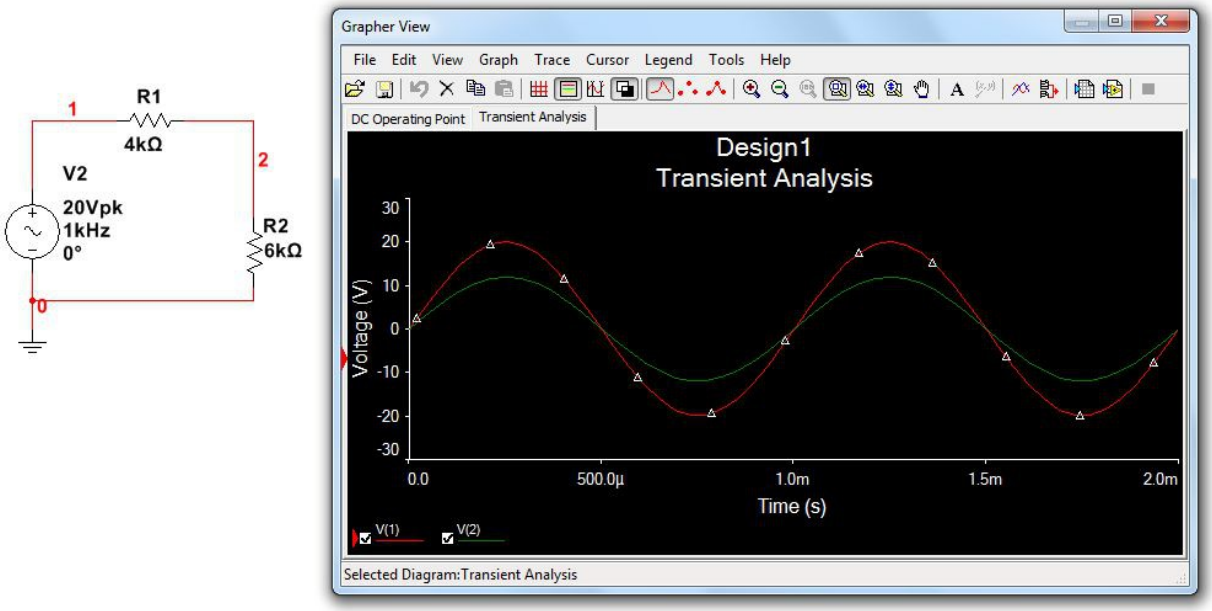

Figure 2A-8

The red and green traces graph the voltages at nodes one and two as they change through time. Note that there are many ways to customize this display. For example, selecting the white/black double square toward the middle of the toolbar will toggle the background color and labels between black and white. The grid pattern will overlay a measurement grid and the set of horizontal lines will bring up a color-node number legend. You can also click on the labels and axes to change them.

One of the more useful items is the pair of interactive cursors. The button for this is between the Legend and the Magnify buttons. Selecting this will bring up a pair of vertical cursors which may be grabbed and moved with the mouse. A separate cursor window will open. This will list the coordinates of the cursors on the plot lines along with the differences between the points (that is, how far apart the points are in time and in voltage). It will also list maxima and minima along with other useful data. An example of the Grapher with many of these settings in place is shown in Figure 2A-9.

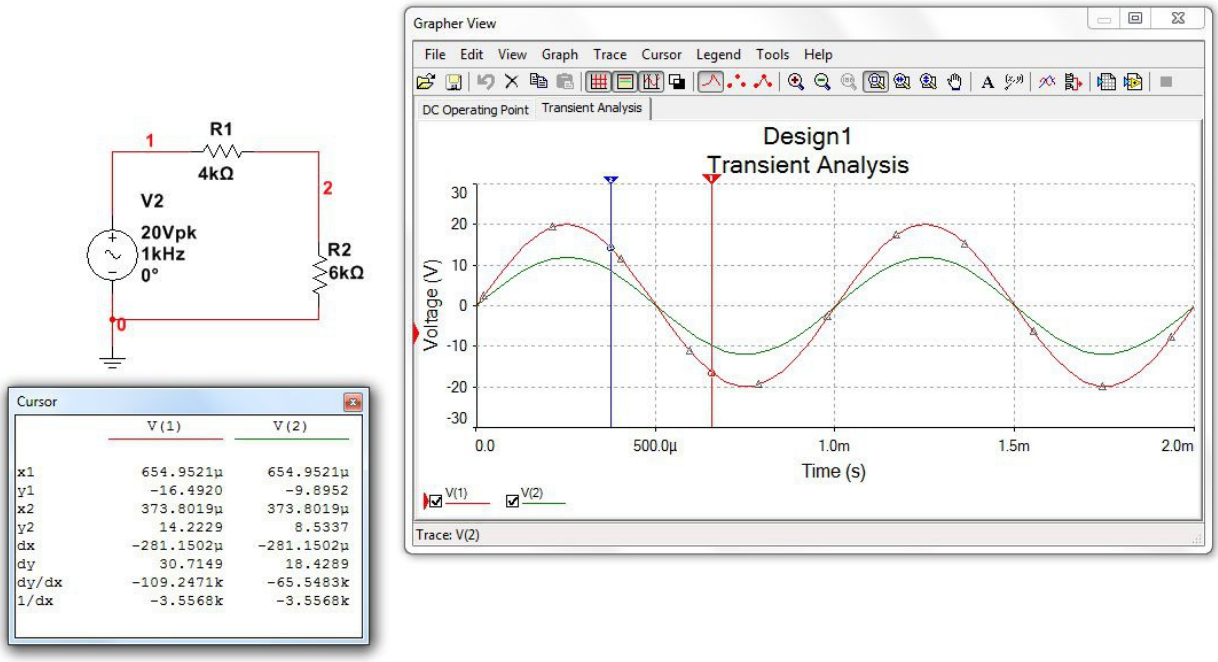

Figure 2A-9

Finally, note that the original DC Operating Point tab is still there. That is, you can quickly recall the original DC analysis by just selecting this tab. It should be clear by now that while the Grapher takes a little more to set up, it is far more flexible and useful than simple virtual instruments.

Assignment

Recreate the transistor amplifier schematic shown in Figure 2A-10. Try to make it as close to this drawing as possible. All parameters must be the same. Device designators must also be the same (after all, these will be cross referenced to the PCB layout and bill of materials). The only items to change are the date (use today's date) and the "Designed by" entry (insert your name).

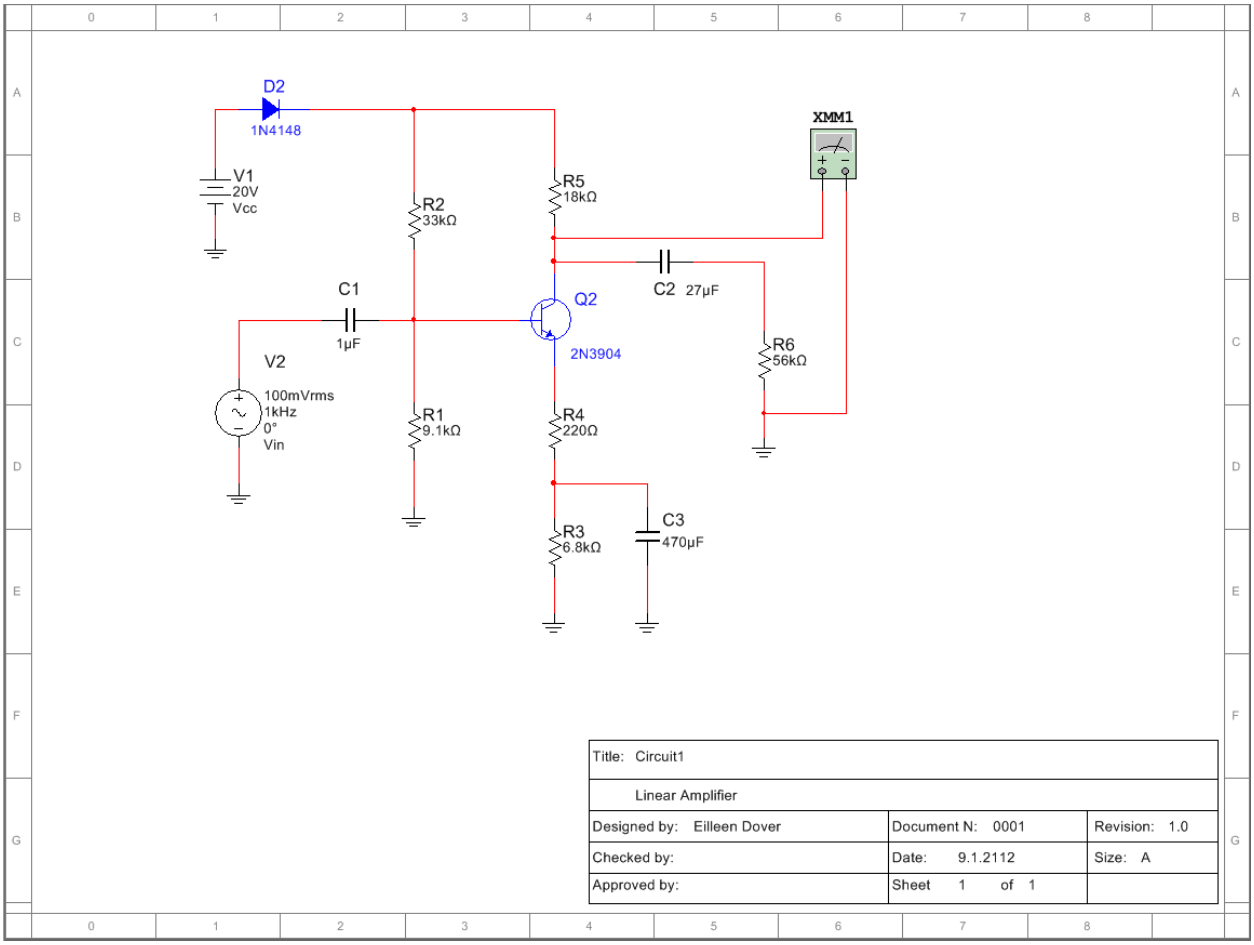
D2
1N4148
V1
20V
Vcc
R2
33kΩ
R5
18kΩ
XMM1
C1
1µF
Q2
2N3904
C2 27µF
R6
56kΩ
V2
100mVrms
1kHz
0°
Vin
R1
9.1kΩ
R4
220Ω
R3
6.8kΩ
C3
470µF
Title: Circuit1
Linear Amplifier
Designed by: Eilleen Dover
Document N: 0001
Revision: 1.0
Checked by:
Date: 9.1.2112
Size: A
Approved by:
Sheet 1 of 1

Figure 2A-10

2B

TINA Extensions

Objective

The objective of this exercise is to become more familiar with the TINA electrical circuit simulation package in order to use more generalized simulations via the Output Windows.

Procedure

In previous work we have examined the basic functionality of TINA, namely basic schematic capture functions such as component selection, placement, and parameter editing, along with simple simulations using voltmeters and the nodal voltages probe to measure DC voltage. While this method is relatively quick and easy to use, it is limited in other aspects. Some of the issues include:

1. Limited measurements per unit, for example a single measurement for a voltmeter, requiring multiple units for multiple measurements.

2. The need to rewire the instruments (and hence the schematic) in order to take different readings.

3. Excessive amount of work space area obscured by the instrument(s) with accompanying clutter.

4. No convenient way of storing and recalling prior simulations.

5. No convenient means of exporting simulated measurement data to other programs.

To address these issues TINA allows for an all-in-one analysis using a Table of Results and also a graphing window. Large amounts of data may be displayed simultaneously. These data may also be saved as a text file for later examination or for input into other programs.

Nothing “extra” needs to be added to the circuit. Instead of choosing the DC Analysis->Calculate nodal voltages menu item, instead, select the DC Analysis->Table of DC results menu item. To illustrate, begin by building the circuit shown in Figure 2B-1. Drag a DC voltage source, an earth ground and two resistors onto the work space. Connect them in a series loop and edit the component values to 20 volts for the source, and 4000 and 6000 ohms for the two resistors.

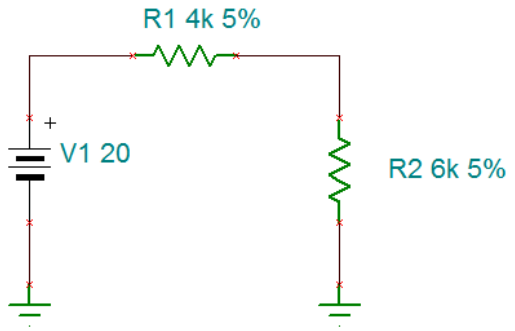

Figure 2B-1

Notice that no voltmeters are required. From the Analysis menu, select DC Analysis->Table of DC results, as shown in Figure 2B-2.

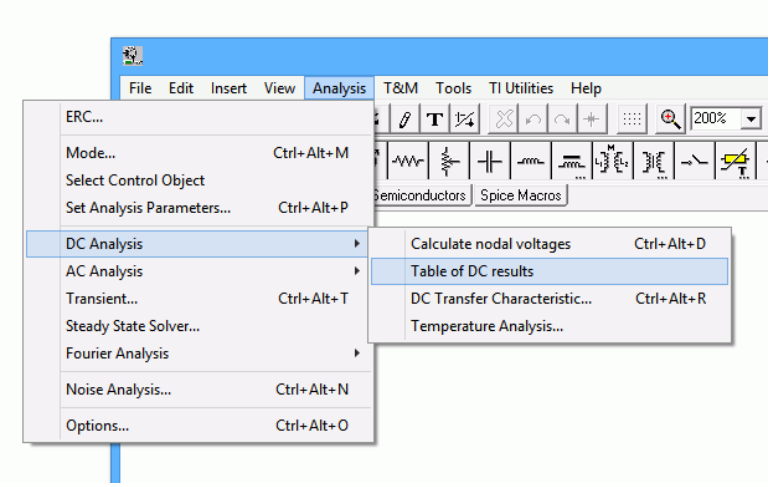

Figure 2B-2

In a moment, the Table of Results window will open as shown in Figure 2B-3. You may need to enlarge the window in order to see all of the values.

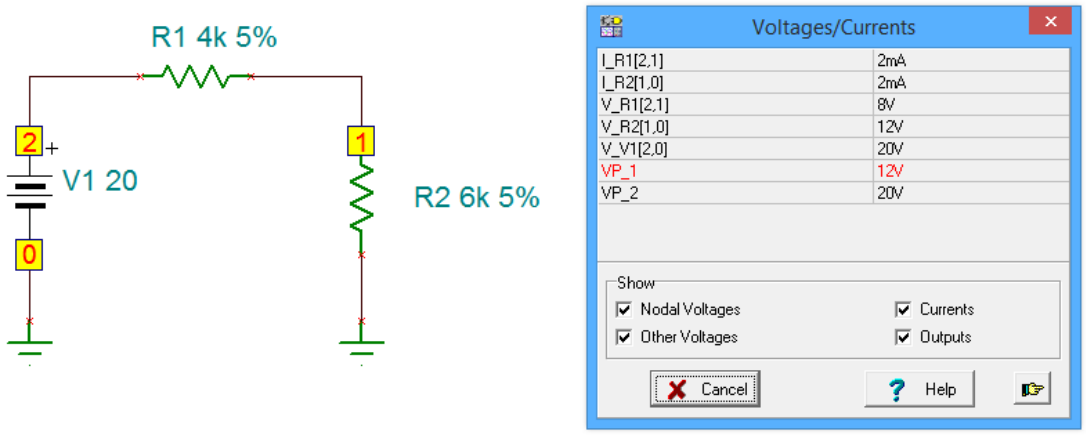

Figure 2B-3

This window lists all of the component voltages and currents in the circuit. Essentially, it's like using the probe on everything and then collecting all of the data in one place. This list can be saved as a text file by selecting the small "finger point" button in the lower right. All values are identified by the component's name. You will also notice that every connection point in the circuit has been given a number. Ground is always 0 and the remaining points are sequentially numbered as the circuit is wired together. Note that voltage points 1 and 2 (VP_1 and VP_2) are included in the list and that each component's current or voltage is also identified by these numbers. For example, the voltage across resistor R1 is identified as V_R1[2,1]. The [2,1] indicates that the component is connected from node 2 to node 1. The first number is the connection for the positive lead of the meter while the second is the connection for the negative lead. Thus, this indicates the polarity and direction of the voltage and current.

It is worth noting that although TINA also has an ammeter similar to its voltmeter, this method tends to be much more efficient for current measurement because ammeters must be inserted in series with the component of interest. That is, the circuit must be rewired to accommodate the ammeter, unlike a voltmeter which is simply placed across the component(s) of interest.

At this point, you might be wondering why you would ever *not* use this method, abandoning the original method and voltmeters entirely. There are two cases where using the original method and/or voltmeters may be preferred. First, in very large circuits the table of values can be overwhelming. Second, as is, there is no obvious way of determining a voltage that appears

across several components, short of adding up the individual voltages manually. In this case, a voltmeter can still be used, its value will show up in the table alongside the other component voltages.

Finally, you may prefer to give the connection points names, such as *Vin* or *node A*, instead of the default numbering scheme. This can be achieved by adding *Voltage Pins* to the schematic. This is the first item in the *Meters* tab and looks like the letter C with a tail. These items can be placed anywhere on a schematic, and once added, the names can be changed. This is illustrated in Figure 2B-4. Two voltage pins have been added to the locations formerly known as nodes 1 and 2. The names have been changed to “b” and “a”, respectively. Note that the table of voltages and currents now uses these letters instead of the numbers.

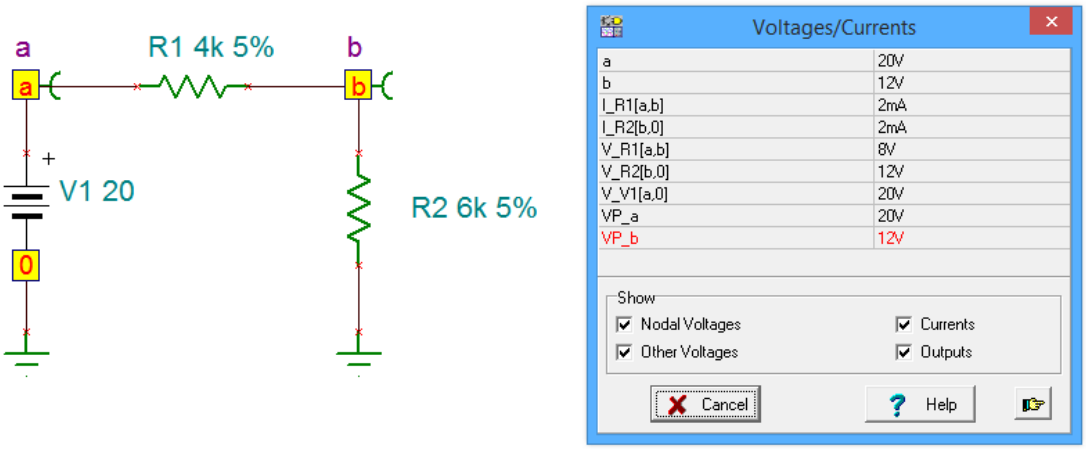

Figure 2B-4

Finally, the table still makes use of the mouse probe. If you click on a Voltage Pin, its entry will be highlighted in the table. If you click on a component, the entries for both its current and voltage will be highlighted.

There are times when graphical data is preferred over simple numeric data. To illustrate this, we will shift our attention to time domain waveform analysis using an AC source. To do this, first delete the DC source in your existing circuit. Replace it with an AC voltage generator. This is the fourth item in the *Sources* tab. Your circuit should like similar to the one shown in Figure 2B-5.

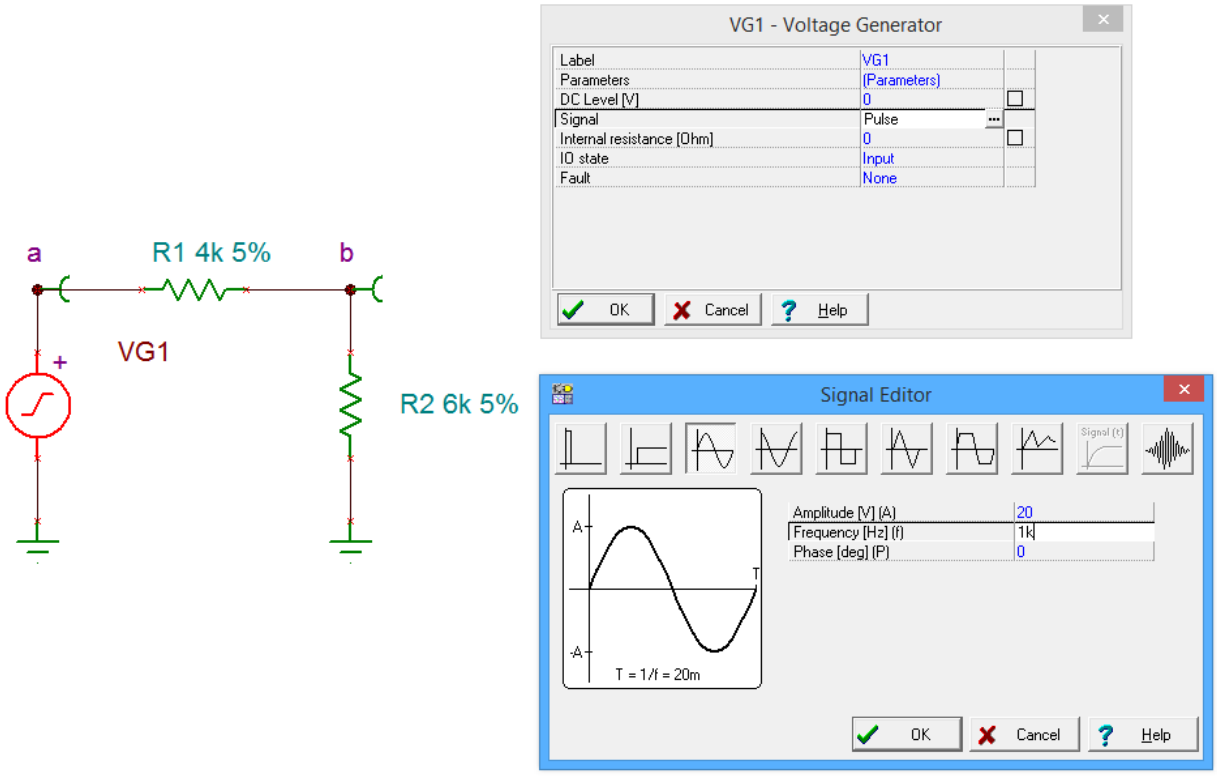

Figure 2B-5

The AC generator is very flexible. We would like to create a 20 volt peak sine wave at a frequency of 1 kHz. To do this, double click on the generator to open its settings dialog box. Click on the Signal row to reveal a small ellipsis button (…). Clicking on the ellipsis brings up the Signal Editor. Select the sine shape (third item) and set the Amplitude to 20 and the frequency to 1000 Hz (1k, for short). Select OK for both of these windows.

From the Analysis menu, select Transient. The Transient Analysis dialog opens, as shown in Figure 2B-6.

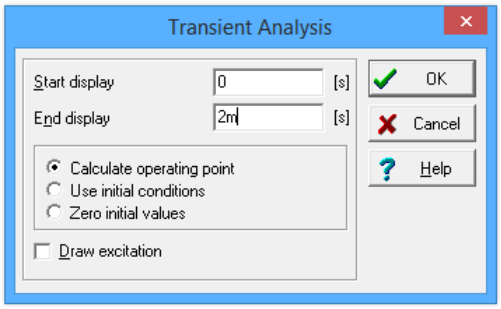

Figure 2B-6

We would like to see the first two milliseconds of our waveform. At 1 kHz, we should see two full cycles. Enter a start time of 0 and and end time of 2m (0.002 seconds). After selecting OK, a graphing window opens as shown in Figure 2B-7. Notice that the time scale runs from 0 to 2 milliseconds and the amplitude spans from -20 to +20 volts. Two waveforms are drawn, node voltages *a* and *b*, as requested by our Voltage Pins.

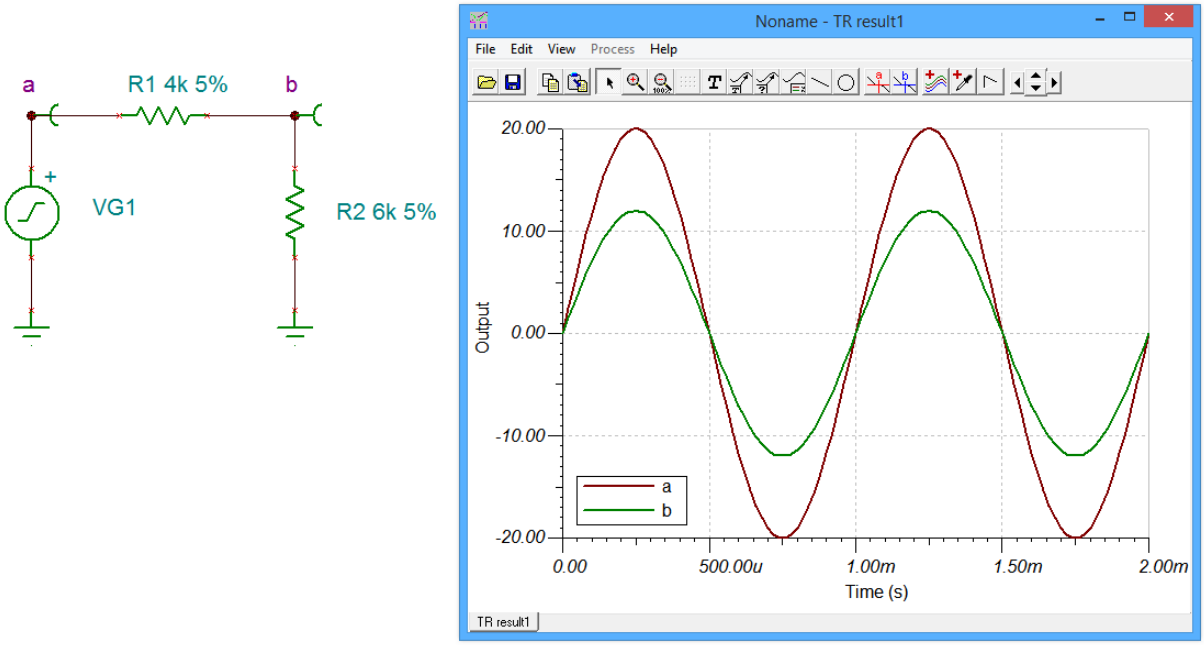

Figure 2B-7

There are several useful tools and variations you can utilize with this graphing window. They are accessed either through the menus or the toolbar. For example, you can zoom into any part of the wave for closer inspection by using the zoom tool (magnifying glass). For precise measurements of the waveforms, you can make use of the cursors. There are two cursors labeled *a* and *b* (red and blue crosses in the toolbar). To use a cursor, select the cursor button and then hover the mouse over the waveform of interest. The mouse pointer will change shape into a cross. Click on the waveform. This will produce a pair of horizontal and vertical lines that track the waveform. You can move the cursor by grabbing the associated letter tab at the top of the graph with the mouse. The values at the intersection are displayed in a small output window. If both cursors are active, the output window will also include the differences between the two sets of cursor values. This is illustrated in Figure 2B-8 where the the bottom line displays the difference in the horizontal X values (here, that's time), and the vertical Y values (in this case, voltage).

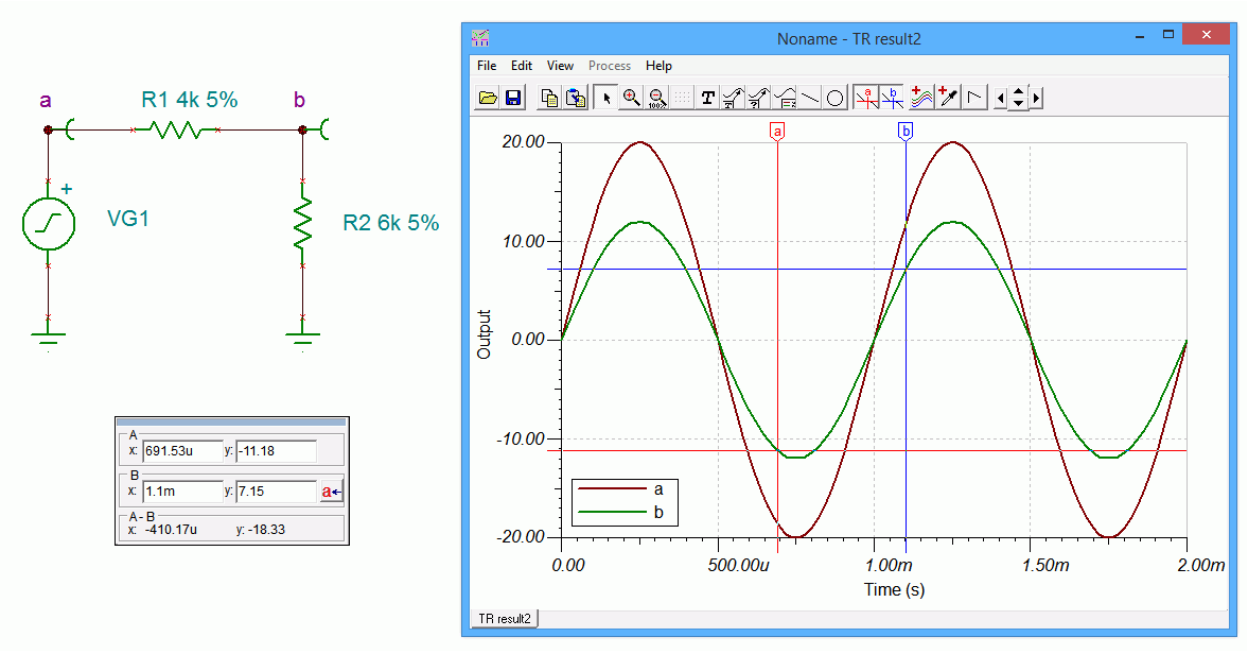

Figure 2B-8

You may wish to separate the curves instead of displaying them together. This can be achieved through the View menu options Separate/Collect curves, as shown in Figure 2B-9.

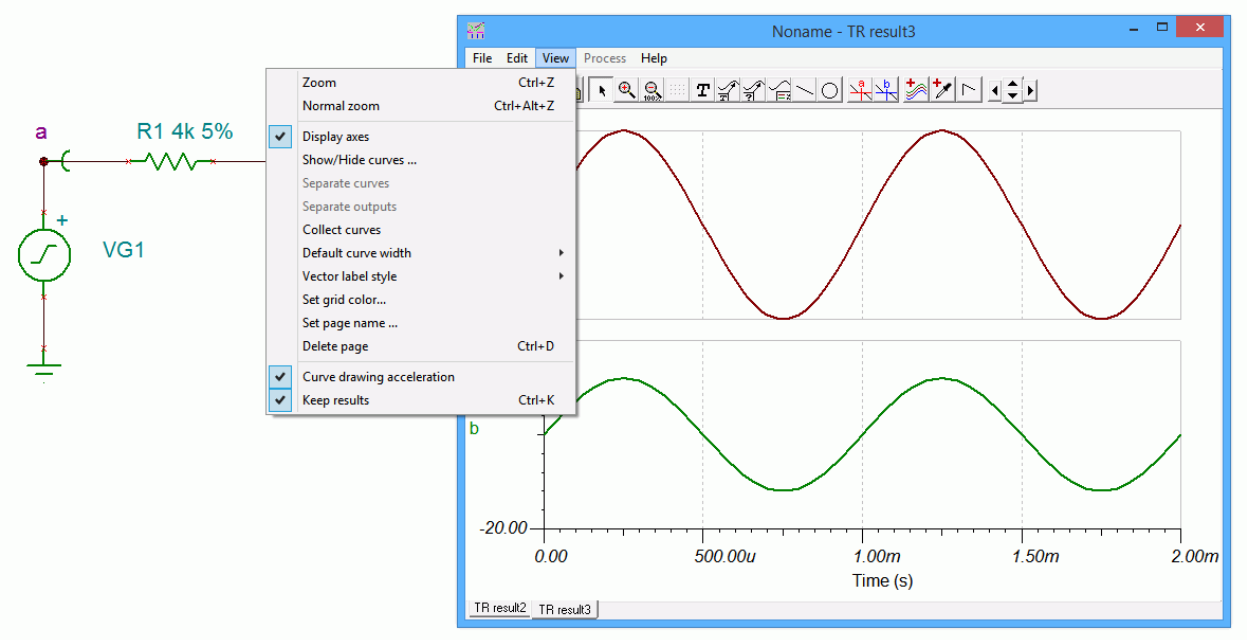

Figure 2B-9

Some other graph attributes can be modified through the View menu such as the color of the grid and the visibility of the axes. Double clicking on a curve will allow you to set the color of the curve, its thickness and other characteristics. Unique labels can be applied using the Text tool (capital T on the toolbar and essentially the same as the one found in the main program toolbar). A pointer can be added to the text with the use of the Text Pointer button (a small T with an arrow). The horizontal and vertical axes can be formatted by double clicking on them. This will bring up the Set Axis dialog box. Here you can change the label, set the numeric format and precision, and also set the scale range and type. This is illustrated in Figure 2B-10.

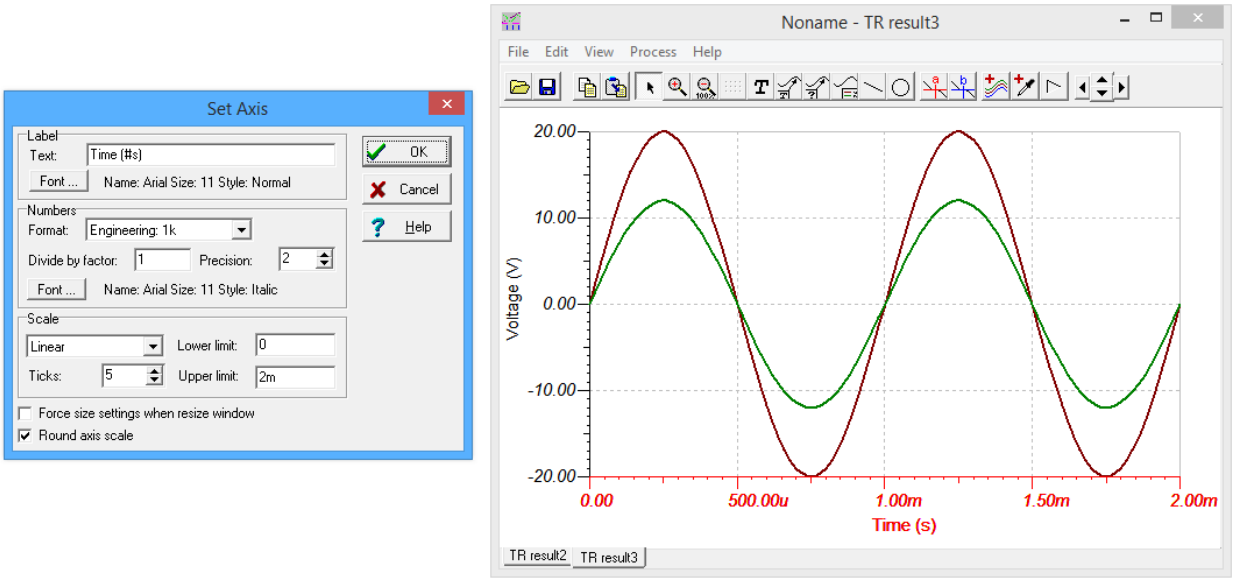

Figure 2B-10

One particularly interesting toolbar button is the one immediately to the right of the cursors. This is the Post Processor. Selecting this will allow you to create complex expressions that can then be plotted. For example, you might wish to find the difference between two voltages or the current through a section of a circuit by dividing its voltage curve by the associated resistance.

The process is fairly straightforward. The Post Processor dialog is illustrated in Figure 2B-11. On the left is a list of available signals. In this example we have the voltages from node a to ground and from node b to ground. Suppose we would like to plot the voltage seen across R1. This is voltage *a* minus voltage *b*. Select a from the list and copy it to the Line Editor by clicking on the large downward arrow. Now, select the minus sign from the drop down list of built-in functions and copy this down to the Line Editor. Finally, select *b* from the curves list and copy it down.

We now have an expression to plot. It can be previewed with the Preview button before selecting OK. This curve is shown in bright blue in the graphing window.

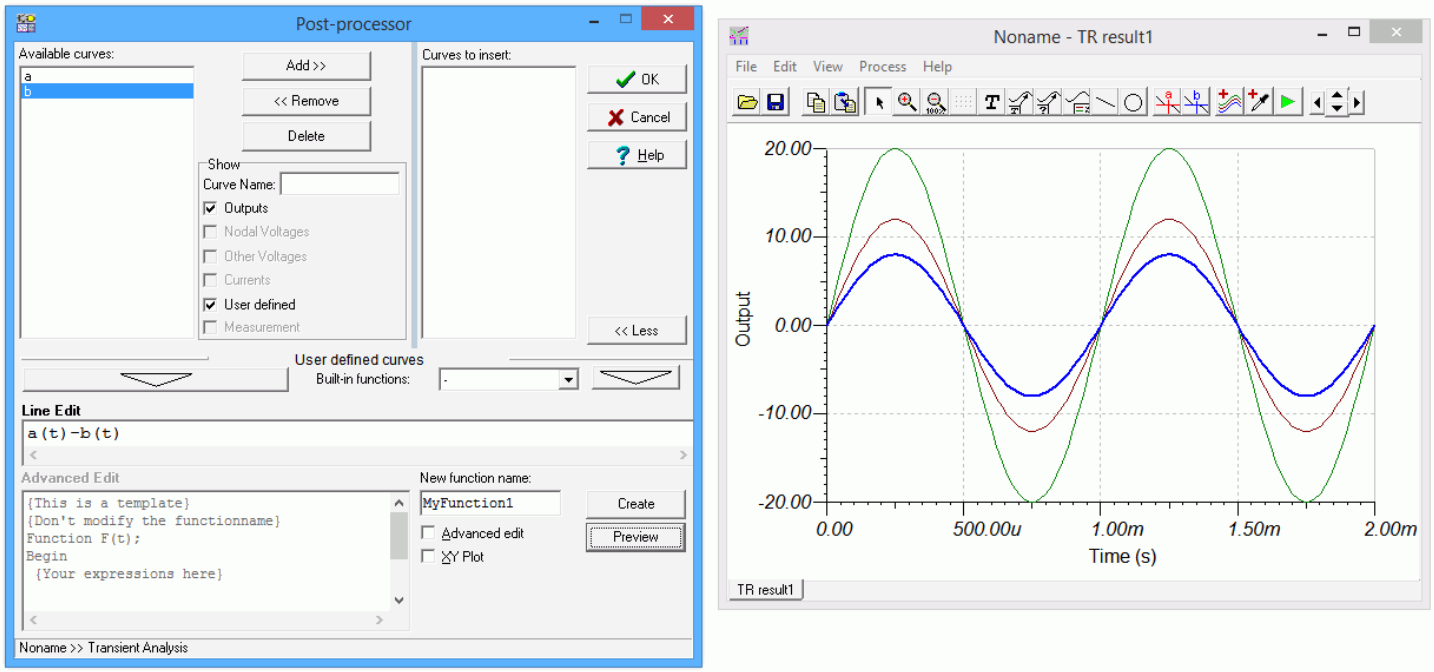

Figure 2B-11

Assignment

Recreate the transistor amplifier schematic shown in Figure 2B-12. Try to make it as close to this drawing as possible. All parameters must be the same. Device designators must also be the same (after all, these will be cross referenced to the PCB layout and bill of materials). The only items to change are the date (use today's date) and the designer's name entry (insert your name).

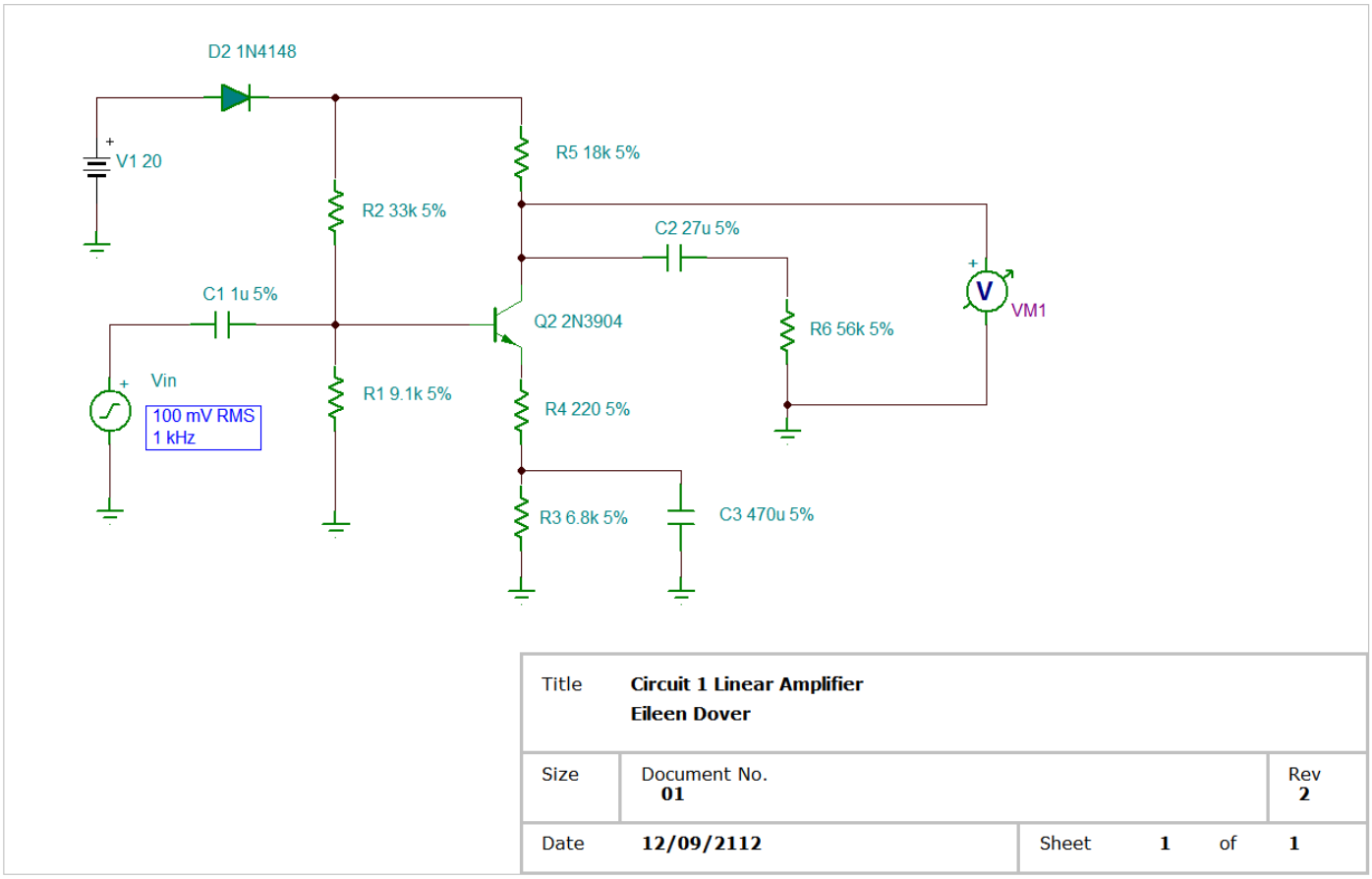
D2 1N4148
V1 20
R2 33k 5%
R5 18k 5%
C2 27u 5%
C1 1u 5%
Q2 2N3904
R6 56k 5%
VM1
Vin
100 mV RMS
1 kHz
R1 9.1k 5%
R4 220 5%
R3 6.8k 5%
C3 470u 5%
Title
Circuit 1 Linear Amplifier
Eileen Dover
Size
Document No.
01
Rev
2
Date
12/09/2112
Sheet
1
of
1

Figure 2B-12

3

Introduction to Python

Objective

The objective of this exercise is to become familiar with the Python IDE for version 3.X while introducing basic mathematical operations, variable types, and printing options.

Background

Virtually all modern programming languages make us of an IDE, or Integrated Development Environment, which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle).

Before opening IDLE, it is worth recalling that there are three basic types of simple variables in Python: integers (whole numbers, commonly used as an index), floats (that is, numbers with a decimal point, AKA real numbers), and strings (collections of alphanumeric characters such as names, sentences, or numbers that are not manipulated mathematically such as a part number or zip code). A legal variable name must start with a letter. It is then optionally followed by some collection of letters, numerals and the underscore. It cannot contain any other characters or spaces, and cannot be a *reserved word* (i.e., a word with a special meaning in the language such as a command or operator). In Python, variables may be created by simply declaring them and assigning a value to them. Examples include:

```
a=2.3
name="Joe"
```

It is best to think of the equal sign as "gets". That is, think of the first example as "the variable `a` gets the floating point value 2.3" and the second as "the variable `name` gets the string *Joe*". An assignment command such as these literally reserves space in the computer's memory for the variable and tags it with the variable name. Then, it stores the appropriate value at that location for future use.

Procedure – Output Window

Open IDLE by selecting Python from the Start menu and then choosing the option to open *IDLE (Python GUI)*. Do **not** open the command line. Alternately, open the *IDLE (Python GUI)* icon on the desktop. A simple text window known as the Python shell will open. It should have a white background with a text message at the top and immediately below that, a cursor prompt

```
>>>
```

The shell serves two functions. First, it can serve as a sort of scratch pad to try snippets of code (shown in the steps below). Second, it can serve as the text output window for larger programs. **Do not try to use this window for the creation of complete programs that you wish to save.** We shall use this window to create a few variables and perform some basic manipulations on them. Type the following and then hit the Enter key:

```
a=5
```

The >>> should reappear. This command defines a variable called `a` and copies the integer value `5` into it. In similar manner, type in the following commands:

```
b=13
x=5.0
y=13.0
m="Mary"
n="Nancy"
```

It is very important that the "`.0`" portions be included. This is how integers and floats are distinguished: floats always have a decimal point, integers don't. Also, it is possible to define the strings using the apostrophe ' versus the quote ". This can be handy if you need to have a string that includes a quote or apostrophe within it; merely define the string with the other character. In any case, the computer's memory now looks something like this:

name	value
a	5
b	13
x	5.0
y	13.0
m	Mary
n	Nancy

The trick now, of course, is to access these values, manipulate them, and see the results. An important command for this process is the `print` command. `print` will print what follows it, either variables or expressions, on to the output window. Note that like all built-in commands and functions in Python, this command is all lower case. Capitalizing it will generate an error. Also, note that commands will be color coded orange-red.

At the prompt, type the following:

```
print( a )
```

The output should be the integer `5`
Now type:

```
print( a, x, m, n )
```

In this case, the following sequence should result:

```
5 5.0 Mary Nancy
```

Continue with the following expression:

```
print( a + b )
```

This results in the value `18`. This line retrieves the values of `a` and `b` from memory, adds them together, and prints the result on the output window. Neither `a` nor `b` are altered in the process. Alternately, we could have created a brand new variable and printed it. The result will be the same. Enter the following two lines to verify this:

```
c = a + b
print( c )
```

The only difference is that this version adds a new "slot" called `c` to the memory map above. It is worth noting that once a variable is created, its value may be recomputed over and over if desired. For example, type the following:

```
c = 20 + a
print( c )
```

The result should be `25`. The first line computes a new value which then overwrites the prior value of `18`.

Besides addition, the other main math operators are –, `*` (multiplication), `/` (division), `**` (exponents, which can also be performed using the function `pow(x,y)` for x^y), `%` (modulo) and `//` (floor divide). Parentheses `()` may be used to force the execution of some operations before others. Parentheses have the highest precedence and are followed by multiplication, division, addition and subtraction. That is, the expression `a=b+c*d` will multiply `c` by `d` before `b` is added. To force the addition first, use parentheses: `a=(b+c)*d` Remember, think of the equal sign as "gets" as in "`a` gets the value computed by…". It is an assignment, not a true mathematical relation. That is, if at some point in the future the value of `b` was to change, `a` will not automatically be altered to reflect that change. This allows you to do the following:

```
c = c + 1
```

Type this in. What do you think the result will be?

The line above may appear a little odd. After all, how can something equal itself plus one? Remember, this is an assignment, not a mathematical relation. What it says is, "Retrieve the current value of `c`, add one to it, and store the result back in `c` (overwriting the original value). Print out the value of `c`. You should be get 26 (the prior value of 25 plus one).

Continuing with the other math operators, type:

```
print( y/x )
```

The result should be `2.6`. Now try the following:

```
print( b/a )
```

The result is also `2.6` even though both variables are integers (an integer, of course, can't contain a fractional portion). In essence, Python *promotes* the variables to floats in order to maintain precision, producing a floating point answer. Now try:

```
print( b/x )
```

In this case the answer is again `2.6`. This is because in a mixed calculation between a float and an integer, the integer is again promoted to a float in the calculation in order to maintain the precision of the floating point variable. You can force a variable to be promoted (or demoted) by using the `float()` and `int()` functions.

Now try:

```
print( b%a )
```

The result should be `3`. The modulo operator produces the remainder of the divide, that is, `a` goes into `b` two whole times with `3` left over. Finally, we have floor divide:

```
print( 18.2//4.1 )
```

The result should be `4.0`. You can think of floor divide as like integer divide but for floats. That is, `4.1` goes into `18.2` `4.0` times. Further, the "left over" is `1.8`, which you can verify with the command `print( 18.2%4.1 )`.

What do you expect the results to be from the following?

```
print( y//x )
print( y%x )
```

Type in the above lines and see if you were correct.

At times it is useful to limit the number of digits that are printed. By default, Python uses up to 18 digits if required. Try this:

```
print( x/y )
```

The result is `0.38461538461538464`. To limit this to fewer digits, the `round()` function may be used. The first argument is the value of expression to be rounded and the second is the number of digits after the decimal point. Now enter this:

```
print( round(x/y,3) )
```

The result should be `0.385` (rounding to the third digit).