

SPICE TUTORIAL

by Chuck Adams, K7QO

Webster's New Collegiate Dictionary defines a tutorial as a technical paper written to give practical information about a specific subject. It is my hope that this writing does in some way meet that requirement.

SPICE, an acronym for Simulation Program with Integrated Circuit Emphasis, is a computer program that was developed at the University of California at Berkeley. Since it was developed with public funding, the program became public domain and although it is copyrighted it is readily available from several sources over the Internet. I will give you pointers to sources that I know of at the end of this tutorial and hopefully you will not procrastinate before they disappear. Keep in mind that SPICE was first developed on mainframe computers in FORTRAN in the early '70s. At that time in the computer revolution there was very little hardware available for doing graphics and SPICE may seem difficult to work with at some times due to graphics features being a later add-on. A number of individuals and companies have modified and extended the user interfaces to include graphics. There are even some frontend schematic programs (schematic capture) that produce data to be input to SPICE, but I have not done much research on these at this time. I have my own schematic routines that I used to produce all the schematic diagrams shown here along with the

plotting software to graph the output. These are programs that I wrote from scratch in C and have been using for a number of years. They work and they do what I want them to do to get work done.

The early versions of SPICE up to the version called SPICE2G6 were done in FORTRAN, and I'll give you a pointer to the source. Starting with SPICE3 the program was rewritten in C, and the two current version numbers are SPICE3F4 and SPICE3F5. SPICE3F5 is just the previous version with bug fixes and some minor optimizations. You can also get the sources to these programs over the Internet. Unless you have a lot of time on your hands, lots of programming experience, know a lot of electrical engineering and solid state physics, and you want to give up a lot of time on the air operating, I do not recommend that you spend much time working on the source files. There are more valuable ways for you to spend your time.

In my 2003 version of this tutorial I recommended you get a program WinSpice3 for Windows. At the time it was free, but in 2007 I find out it is not and requires you to pay a fee after one month of free use. This will not do, so now I will be converting the entire tutorial over to **ngspice**. Ngspice runs under Linux and I will show you how to get it and compile it under Kubuntu 7.04 which comes out in mid-April 2007. It is also the distro that I recommend in my Linux tutorial and a book that I am writing

for first semester high school or first semester college students for their educational purposes.

You are most likely the typical QRPer who experiments and builds things from scratch and you don't have an advanced degree in electrical engineering or do electronic design for a living. You probably research the ARRL Handbook or other literature and study circuits and borrow or use circuit designs for part of the work that you are doing. You probably find one or more circuits that provide the same function, read the specs written up by the designer or author and pick the one that you think best suits your needs. The criteria for selection may be price of components, access or availability of parts, complexity of the circuit, power consumption, etc. After the choice is made, you build the circuit and probably tinker with parts values to see if you can improve on the design in some way and make it better for your use.

Sometimes we succeed in this process and sometimes we fail due to any number of reasons. The problem is that failures, even though we learn more from them than we do our successes, may require a lot of time and sometimes money when we totally destroy expensive components. The use of a computer program like SPICE may help us to eliminate circuits that do not work or optimize those circuits that will work before we sit down with a prototyping board or do ugly construction. It is the purpose of this tutorial to show from the ground up some of the capabili-

ties of the program and where I have found it to be useful.

Because of the wide range of experiences and education of the audience, I will assume no previous experience. For those that are experts already I hope to show you something new or unique along the way, but no guarantees are implied.

And another use for SPICE that I'm interested in is getting new radio amateurs and new individuals interested in electronic circuits, how they operate and learning how to use SPICE to analyze circuits and devices as they encounter them. SPICE makes a great teaching tool and even makes it more fun than sitting down with pen and paper and calculator and just cranking out numbers. A large number of engineering schools use SPICE in a similar manner.

SPICE Installation

I assume that you are running Kubuntu 7.04 Linux or another version of Linux. You have to download the source to `ngspice` and compile it. It requires a number of compilers and other programs to get it to run and you are not going to get these without significant cost if you get them from Microsoft in WA.

Use the **Adept Manager** to make sure you have the following installed. All are supported by Ubuntu and work perfectly for this application.

bison

```
byacc
gawk
gcc
g++
g77
lex
```

Now execute the following commands in the order shown. If the `make` sequence exits with a serious error (there will be some warning messages) then you will need to add the missing software on your particular version of Linux. I may be able to help you, but hopefully you are experienced enough on computers to figure out the problem. I may come back and add a FAQ if there are enough people having problems. I can't teach you everything. You will have to have networking on the system and connected to the Internet to get the source via the method shown below.

NOTE: The lines in red must be one line with no spaces after the `http`. You'll know when you get it wrong.

```
cd
mkdir ngspice_src
cd ngspice_src
```

```
wget -c http://downloads.sourceforge.net/
      ngspice/ng-spice-rework-17.tar.gz
```

```
tar -xvzf ng-spice-rework-17.tar.gz
cd ng-spice-rework-17
```

```
./configure
make
sudo make install
which ngspice
```

The `which` command should return a value of `/usr/local/bin/ngspice` and the `make` command may take up to or more than 5 minutes to complete. It does a lot of stuff. The `sudo` command requires you to enter the system administrator or root password, which you should know since it is your system.

OK. Now we have `ngspice` installed and ready to go. Let's get on with the tutorial and examples will be given on how to execute the program with data files and get meaningful output from the program.

In order to separate all my stuff from yours, let's make a directory on your system, `k7qo`. Do the following.

```
cd
mkdir k7qo
```

This will allow me and you to work together on just the material shown here and then you can work where you want on your new stuff or you can continue to use this directory. I am going to have you execute `ngspice` in this directory with my material and I'll have you get it from my web site for each exercise so that you don't have to type too much into your system. It'll save you a lot of time. I hope.

WHY SPICE?

SPICE was developed during a period of transition in the semiconductor industry. In the mid-1970s the semiconductor complexity of chips and microcomputers was rapidly increasing. Prior to the production of an integrated circuit chip a prototype was made up of discrete components, tested and tuned, and then made into semiconductor material. The production of a single chip for testing and development was too expensive at the time and required long turnaround times for the manufacture of just a few parts to test and evaluate. And the testing and probing of the completed part was almost impossible in many cases due to the lack of test equipment and procedures at the time.

I remember seeing in an early issue of BYTE Magazine (around 1976 – 1978 timeframe) photographs of the development team at Motorola in Austin, TX showing the prototype made up of discrete components for the Motorola 6809 microprocessor. This was done to develop the microprocessor chip, debug the logic, etc. Probing and measuring the logic states and transitions could only be done on large systems at the time. But with the larger 68K series of microprocessors and everything since then, the only way to do microcomputer chip development has been with software such as SPICE and other programs. The scary thing about the current state of computer development is that so much software and automation is needed that much of the

detail is not known to humans but contained and embedded in the software development tools.

The side effect of this is that we as radio amateurs have access to a program that we can use for RF design, and we don't have to write programs from scratch.

INPUT TO SPICE

Remembering that SPICE was first developed in the days of computer input via card decks, you will find that the input is typically one 80 column line per component with the capability to continue long lines onto the next one. Some programs still limit the length of input lines to 80 columns. Also case sensitivity occurs aperiodically, and you will know when this happens to you. I will show all source files for SPICE in upper case and in a typewriter font, and if I show lower case in the source it may be required for that instance. Sometimes the program does not give any warning messages or errors and just fails to output or do what we want. It is not an error on your part or the program. It is just the way it works, and you will learn rather rapidly where these things are critical.

First of all, let me to list the types of components that SPICE can simulate and the letter used for each.

- C Capacitor
- D Diode
- E Voltage-Controlled Voltage Source

F	Current-Controlled Current Source
G	Voltage-Controlled Current Source
H	Current-Controlled Voltage Source
I	Independent Current Source
J	Junction Field-Effect Transistor (JFET)
K	Coefficient of Coupling for Mutual Inductance
L	Inductor
M	Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET)
Q	Bipolar Junction Transistor (BJT)
R	Resistor
T	Transmission Line
V	Independent Voltage Source

I will give an example of most of these, but due to limited time and space will be unable to show you each and every use for typical radio amateur applications.

First we need to define a “node”. A node is a point where two or more components are physically connected together. Think of this as a solder joint or two or more pads on a circuit board that are connected together via the same trace on the board. Each node in a SPICE circuit is given a number and in some programs you can use names, but I will try to restrict myself to numbers in this tutorial to avoid confusion and hopefully avoid making things more difficult. The numbers are positive integers or zero, but zero is reserved for the ground reference point. Be sure to always assign the ground connected points or nodes the value zero. All other node

voltages are referenced with respect to this point in the circuit.

The node numbers are arbitrary. You just take the schematic of the circuit that you wish to simulate and assign unique numbers to each of the nodes in the circuit in much the same way that you assign numbers to the resistors and capacitors and other parts. Remember that the only restriction is that 0 is reserved for ground. The program does not care if you skip numbers, and it will warn you if you have a component with one node that is not connected to another part of the circuit. I would suggest that you incrementally assign node numbers and be extra careful about not duplicating a number between two different nodes. You’ll in effect connect the two points together with the program giving you no warning that an error on your part has occurred.

For each component you assign a name, the node value(s), and the component value. There also may be options that you can add for temperature coefficients or initial conditions for the start of the simulation. I will try to work in an example before I run out of time and space.

Since the component values may have a wide range of magnitudes, SPICE will allow you to use factors for component values from the following choices and the letter does not need to be uppercase, but I tend to use uppercase except for micro,

F = 1E-15	(femto)
P = 1E-12	(pico)
N = 1E-9	(nano)
U = 1E-6	(micro)
M = 1E-3	(milli)
K = 1E3	(kilo)
MEG = 1E6	(mega)
G = 1E9	(giga)
T = 1E12	(tera)

where E denotes the power of 10, so that 1E-15 is 10^{-15} . Please be very careful. The abbreviations for the multipliers or scaling factors will come back to haunt you. Make sure that if you want a one megohm resistor that you use 1MEG or 1E6 and not 1M, as the latter will give you a thousandth of a ohm and you could have just as well put in a short or zero ohm resistor. Also the letter F should not be used in capacitors unless preceded by one of the other multipliers. For example, 0.01UF or 0.01uF would be a correct notation for the value of a 0.01 microFarad capacitor but 0.1E-6F would be a capacitor with a value of 10^{-21} Farad. This one error alone has cost some companies and consultants a lot of time and money. Fortunately I was not the one involved.

If the value consists of a string of numbers and then letters, the first letter is taken as a scaling factor (if it is one of the above) and the rest are ignored except for the case of MEG as a multiplier. This allows one to use uF for microFarad and 1MEGohm to denote the obvious one

meg value for a resistor or 50ohms for a 50 ohm resistor since o is not a valid scale factor or multiplier.

RESISTORS

A resistor is placed in the circuit between nodes N1 and N2 with the line

```
Rname N1 N2 value
```

where name is the name of the resistor, usually in the form of a number, and the resistor has the numerical resistance of value ohms. For example

```
R36 36 37 1.2K
```

represents a 1.2K resistor connected between nodes 36 and 37.

I have shown the lines above and will continue to show the input lines for SPICE in this tutorial indented some to offset them from the text. The lines of input to a SPICE program must begin in column 1; thus the R in the above lines must be in column 1.

There is an optional field on the resistor input line for temperature coefficients if one wants to do temperature analysis of a circuit, but I prefer to leave that for more advanced work in another place and time. Just remember that the fields exist for the input and you will have to research a complete book on SPICE to get the details

if you feel the need to use it. It is critical for semiconductor chip development.

CAPACITORS

A capacitor is placed in the circuit between nodes N1 and N2 with the line

```
Cname N1 N2 value
```

where name is the name of the capacitor, usually in the form of a number, and the capacitor has the numerical capacitance of value Farads. For example

```
C56 103 0 0.047uF
```

represents a $0.047\mu\text{F}$ cap connected between nodes 103 and 0 (ground).

In some books you will see authors use the notation

```
Cname N+ N- value
```

to represent the input line for a capacitor. The notation N+ for the positive node and N- for the negative node for non-electrolytic caps. SPICE couldn't care less about this, and don't let it get to you. A capacitor is a capacitor, and only in the real world do we get into trouble by hooking one up backwards like a tantulum or electrolytic. SPICE will also allow one to specify a non-linear capacitor with polynomial coefficients that determine the value of the capacitor

dependent upon the voltage across the terminals. I assume that we will not be interested in this capability for now. This is useful for working with varactors if you have the curve for the voltage vs. capacitance characteristics. I'll try to come up with a variable bandwidth crystal filter using this as an additional circuit to try, either here or on the web page at a later date.

INDUCTORS

For an inductor the input line consists of

```
Lname N+ N- value
```

for a value Henry inductor connected between the two nodes N+ and N-. The notation is such that a positive increasing current flowing from the positive node to the negative node will induce a positive voltage across the inductor with N+ being the positive or higher potential. So an input line in the form of

```
L3 27 39 15uH
```

would represent L3, a $15\mu\text{H}$ inductor, connected between nodes 27 and 39.

One thing that might help keep you out of trouble is to use the "dot" notation for inductors and transformers and always use the N+ node for the "dotted" end of the inductors in the circuit. I'll illustrate this in the next section on MUTUAL INDUCTORS.

MUTUAL INDUCTORS OR TRANSFORMERS

The line of the form

```
Kname Lname1 Lname2 value
```

allows two inductors, `Lname1` and `Lname2`, described elsewhere in the source file for the circuit to be magnetically linked by a coupling factor `value`. This is nothing more than a transformer. The value must be greater than zero and less than or equal to one.

```
K1 L3 L4 1.0
```

would represent a tightly coupled transformer with L3 and L4 being the two windings and the coupling factor being 1.0. The N+ nodes of the two inductors would be the dotted ends during the SPICE calculations.

There is no limit to the number of windings that can be magnetically coupled, so you can work with trifilar windings if you want. Say we have N windings on the same toroid. We would need $N*(N-1)/2$ coupling factors to describe this transformer. The way you can determine this is that for every winding there are N-1 remaining windings that we can interact with, but the coupling is the same between L_i and L_j and L_j and L_i , thus the division by two.

Let's look at a transformer with two windings on a toroid. Let's also for the ease of getting into what is going on say that the core is a T37-6

with 8 turns for the primary L15 and 16 turns for the secondary L16. Knowing the core and the number of turns gives us $0.192\mu\text{H}$ for L15 and $0.768\mu\text{H}$ for L16.

The SPICE source lines for this transformer would consist of the three lines

```
L15 55 56 0.192uH
L16 57 58 0.768uH
K1516 L15 L16 0.995
```

I have not seen in the literature factors less than 0.90 and have I have not had a chance to do the experiments myself to see just what values may be used for the coupling factor to get exact results. I would be glad to hear from those who have done research in this area and it may be an area for someone to do an article in for the QRP ARCI Quarterly or other newsletter(s). Let me know if you have or are planning to do this as it may save me some duplication of effort. If the simulation gives you the same results as you get from the real circuit, then you have succeeded in doing it correctly.

The transformer shown in the above three lines would be represented by the following schematic. If you use a single notation for the transformer, then you can call the transformer T1 and then in the SPICE input file put a LT1P inductor for the primary and a LT1S inductor for secondary to make the notation easy to follow.

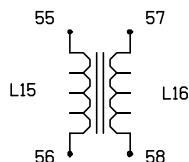


Figure 1. Transformer in SPICE.

If I forget to do it manually, go ahead and place two dots for the two upper terminals in the transformer above as the schematic routine doesn't have it in there yet — work in progress, as I always say. Thanks.

INDEPENDENT VOLTAGE SOURCES

Independent voltage source is a term that refers to voltage sources within a SPICE simulation that are not determined or influenced by outside conditions or other circuit parameters. If you have a 12.68V supply, then you will have 12.68V across the terminals of the supply whether the current through it is $1\mu\text{A}$, 1mA or 1,000A, difficult to do in real life but very easy for the computer to do. We are emulating an ideal battery or other ideal power source for the power sources.

For independent voltage sources the SPICE input line may have several formats, so let me take a few examples to illustrate how we provide voltage sources.

First the generalized format is

```
Vname N+ N- [ [DC DC-VALUE]
```

```
[AC [ACMAG [ACPHASE]]]
[TRANKIND(TPAR1 TPAR2 ...)] ]
```

Now don't panic. I know it looks complicated. The brackets [and] indicate the beginning and ending of optional fields.

Let's first take just a simple DC 12.8V supply. This would be represented by

```
Vcc 31 0 DC 12.8V
```

and please note that here I used letters for the name instead of numbers. Some versions of SPICE may require you to use numbers and not allow alphabetic characters in the field name, so you may have to experiment if you are not using SPICE3F4 or WinSPICE3. One thing that may happen or you will want to know about is what happens if you happen to leave out the DC and voltage value. You will get a voltage source with a DC value of zero in such instances. The voltage acts like a short and provides no energy to the circuit. You can use this voltage source as an ammeter. SPICE does not have a formal definition of an ammeter but with a voltage source of zero you can print or plot the current through the voltage source and we will make use of this later, so don't forget about it.

```
Vammeter 31 0 DC 0V
```

would be a typical line in a SPICE simulation to indicate that the voltage source is going to be

used as an ammeter to measure current through that part of a circuit.

OK, so much for DC, let's look at constant magnitude AC voltage sources. These we can represent with the source line

```
Vac 31 0 AC 1.50V
```

to represent an AC voltage source of 1.5 volts. The voltage will be constant independent of the current through it and the load on it from the rest of the circuit, sometimes impossible to do in real life. Note that nothing is said in this line about the frequency, and we need another control line in the SPICE source file to set the frequency. We'll get to that later, or we can use one of the transient analysis parameters shown in the next few paragraphs.

INDEPENDENT CURRENT SOURCES

Like the independent voltage sources, the generalized input format for independent current sources is

```
Iname N+ N- [ [DC DC-VALUE]
[AC [ACMAG [ACPHASE]]]
[TRANKIND(TPAR1 TPAR2 ...)] ]
```

where a positive current flows from the positive terminal through the source to the negative terminal. Instead of having a voltage source we have a current source and can do the same thing for currents that we can do for voltages. All we

have to do is substitute I for the V and use the same control structures that we will see for the voltages when doing transient analysis.

FIRST CIRCUIT ANALYSIS USING SPICE

Since we are now far along with a description of SPICE and working on what it can do, let's take a little side trip and do an actual simulation using SPICE3 both on a workstation and on a PC. I'll be using SPICE3F4 on the workstation and WinSpice3 on the PC running Windows98 in these examples. Hopefully you have one or the other to work with.

In order to get you on the computer early let's start with a simple passive circuit that only has resistors, capacitors, and inductors and no active devices like transistors or diodes. We'll start with the Five Pole Chebyshev Low Pass Filter that is typically found as the final filter in the PA section of a QRP transmitter. If you have an ARRL Handbook for 1995 or later, go to Chapter 30 page 23. This is 30.23 using the current page numbering scheme.

The schematic for this filter looks like:

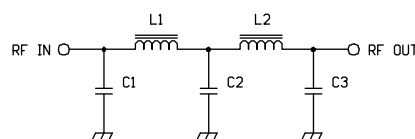


Figure 2. Five Pole Chebyshev Filter.

The output and input design impedances for this

filter are 50 ohms and in order to correctly simulate its behaviour we must make sure to setup 50 ohms for the source impedance and terminate the output to a 50 ohm load. This means that we now have a circuit that looks like:

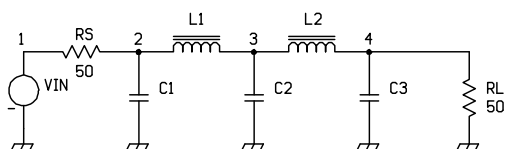


Figure 3. Chebyshev SPICE Simulation Circuit.

We have the transmitter section represented with an internal impedance of 50 ohms and a voltage generator represented by `VIN` and `RS` and the load resistance represented by `RL`, also 50 ohms. I have assigned node numbers to each of the nodes other than ground which will always be node 0, so I won't bother to show it in any diagram that you see from me. Saves me time and energy.

SPICE typically expects to have the input source for simulation to be a file with a name and extension in the form of `FILENAME.CIR`, where `FILENAME` is some name that you chose. So what I'll do for this little exercise is give the file a name of `cheby01.cir`. Every input file for SPICE has the first line treated as a comment or title line, so make sure that the first line is not supposed to be one of the elements in the circuit. I usually put the name of the file that contains the source and some other comments that remind

me just exactly what I was doing. If you need to put comments in the source file, then start each comment line with a `*` or asterisk column 1. Any line that you see with a plus sign (+) in column 1 is a continuation of the previous line. This allows us to have source statements that are more than one line long if we need them and we will.

The source file must end with a line that has `.END` on it. Some of the newer versions of SPICE may allow additional statements after the `.END`, but they are typically some control statements.

Also, to make things more complicated we need to tell SPICE to output some information for us either as text or to plot some data that we are interested in. Any commands that are book-keeping or commands to SPICE to do something all start with a period in column 1 and the command name immediately following it. Some of the things that we can do are `PRINT` values or `PLOT` values, and I'll explain these as we go in each of the samples. I will come back and give you the general syntax of the commands after we get some initial runs made and then move on to more complex circuits.

OK, with that all said and done, let's try a simulation of the Five Pole Chebyshev Low Pass Filter. The file will have the name `cheby01.cir` where the 01 in the name I use to indicate that this is file number one of a possible series of runs involving Chebyshev filters. The file for Figure 3 will look like:

```

CHEBY FILTER  cheby01.cir
*
*   FIVE ELEMENT CHEBY FILTER NR. 91
*   ARRL Handbook
*
*   K7QO sample file for SPICE introduction
*
VIN 1 0 AC 1
RS  1 2 50ohms
C1  2 0 390pF
L1  2 3 1.48uH
C2  3 0 750pF
L2  3 4 1.48uH
C3  4 0 390pF
R1  4 0 50ohms
*
*   The following line tells SPICE what range
*   of frequencies to use for the input source
*   VIN
*
.AC LIN 81 6MEG 21MEG
*
.END

```

Now it is time for you to try this circuit with your system. Type in the following lines. The first few will get you to the directory we set up earlier.

```

cd
cd k7qo
wget http://www.k7qo.net/cheby01.cir
ngspice
source cheby01.cir
run

```

The `source` statement reads in the source file

`cheby01.cir` and the `run` statement causes `ngspice` to run the simulation on the circuit.

At this time you should see lines on the screen that look something like:

```
ngspice 1 -> source cheby01.cir
```

```
Circuit: ANALYSIS OF CHEBY FILTER
```

```
ngspice 2 -> run
```

```
Doing analysis at TEMP = 300.150000 and TNOM = 300
```

```
Warning: vin: has no value, DC 0 assumed
```

```
No. of Data Rows : 81
```

```
ngspice 3 ->
```

Now type in

```
plot vdb(1) vdb(2)
```

What do you see? Well, you see a new window pop up with two plots on it, one line in **red** and another curve in **blue**.

TRANSIENT ANALYSIS

For both independent voltage and current sources we have a parameter, `TRANKIND`, that allows us to set the type of transient analysis. There are five available types of transient analysis values allowed in SPICE simulations and they are

- PULSE — for a single pulse or pulse train
 - SIN — for a sine wave that may have a decay factor
 - EXP — for a single pulse with an exponential rise and fall
 - PWL — piece-wise linear function
 - SFFM — single-frequency frequency modulation source
 - DOWN — the fall time for the pulse in going from V2 back to V1
 - WIDTH — the time that the voltage remains at V2 during the pulse
 - PERIOD — the time interval between beginnings of each pulse if more than one
- where all the times are in seconds.

PULSE TIME DEPENDENT VOLTAGE SOURCE

The first of these is the pulse, which may be just a single pulse or a series of pulses. The source line for the voltage (the current time dependent sources are the same as the voltage except with an I) pulse is

```
Vname N+ N- PULSE(V1 V2 DELAY UP DOWN WIDTH PERIOD)
```

where

- Vname — the name for the voltage source
- N+ — the positive terminal of the voltage source
- N- — the negative terminal of the voltage source
- PULSE — the type of transient analysis to be done
- V1 — the starting voltage of the pulse in Volts
- V2 — the pulsed voltage value in Volts
- DELAY — the delay time before the start of the pulse train
- UP — the rise time for the pulse in going from V1 to V2

In order to show you exactly what such a pulse looks like and to give you a circuit to input into SPICE to examine the nature of these parameters here is a test circuit using a single pulse.

TRANSIENT ANALYSIS of a Pulse Train

```
*
* pulse01.cir Chuck Adams, K7QO
*
* Pulse is delayed 10mS
* Starts at voltage level of 1.0V
* Rise time of 1mS
* Pulse Voltage of 2.5V
* Pulse Width of 13mS
* Fall time of 2mS
* Period of 20mS
*
Vinput 1 0 PULSE(1.0V 2.5V 10mS 1mS
+          2mS 13mS 20mS)
Rload 1 0 1ohm
*
* The .TRAN command to SPICE is necessary in order
* to start up the simulation with the pulsed voltage
* source
*
.TRAN 0.1mS 75mS
*
```

```
.PLOT V(1)
.END
```

```
Vname N+ N- SIN(OFFSET AMPLITUDE FREQ
+          DELAY DECAY)
```

The plot for the output looks like:

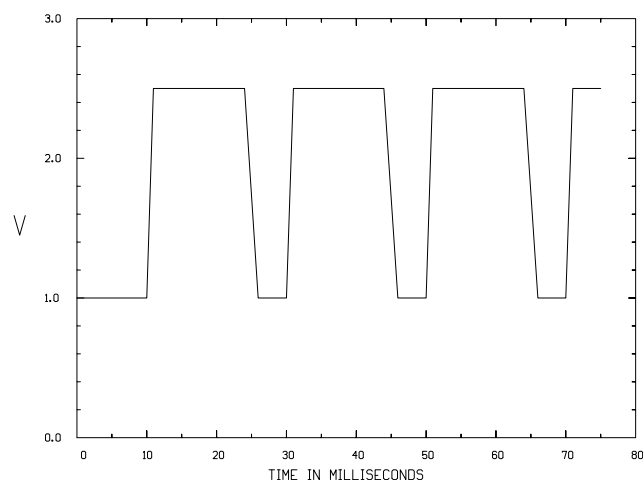


Figure 4. Output from transient analysis.

If you make the period value equal to zero, you will only get a single pulse. This feature is needed and useful to study oscillators as you need a condition to start oscillation. The reason for having the function for time dependent voltage or current sources is to also determine response functions for devices or functions for advanced studies and critical analysis in more demanding areas of circuit simulation.

SINE WAVE VOLTAGE SOURCE

Another wave form for a voltage source is the sine wave. Its input format is:

where

- **Vname** — the name for the voltage source
- **N+** — the positive terminal of the voltage source
- **N-** — the negative terminal of the voltage source
- **SIN** — the voltage source is a sine wave
- **OFFSET** — the offset voltage for the source
- **AMPLITUDE** — the voltage amplitude for the sine component of the source
- **FREQUENCY** — the frequency for the sine component of the source
- **DELAY** — the time delay before the sine wave starts
- **DECAY** — the exponential decay factor for the damped wave

The way this works is that the voltage will start out at **OFFSET** volts until time is equal to **DELAY**. Then the sine wave will start up with an amplitude of **AMPLITUDE** and frequency **FREQUENCY**. If **DECAY** is non-zero, then the amplitude of the sine wave will die off until the simulation stops.

Once again here is a sample circuit as above in the case of the pulse but now with a sine wave source superimposed upon a DC offset voltage. I'll go ahead and show the input file and the plot. I do hope that you will take the file and use it to vary the parameters and look at the results.

```

TRANSIENT ANALYSIS of a Sine Transient Function format and it becomes almost instantly recog-
*                               nizable for what it is, i.e. a damped sine wave.
*                               Once again I recommend that you take the above
*                               simple program, run it on your own and exper-
*                               iment with the values. It is the only best way
*                               to help you remember the parameters and their
*                               meanings.
*                               Vinutput 1 0 SIN(1.0V 0.25V 100Hz 10mS 25.0)
*                               Rload 1 0 1ohm
*
* .TRAN Timestep Timestep Tstart TMAX
*
* If you run this simulation, because of the
* 100 sample points per mS and the long plot
* time of 75mS you will note that this takes
* a few seconds to do before SPICE comes back
* But I can tell you this is a lot faster
* than in the old days of the mainframes.
*
* I needed the extra fine resolution to produce
* nice curves in the plot. Play with this.
* You'll learn why I did it this way.
*
* .TRAN 0.01mS 75mS 0 0.01mS
*
* .PLOT TRAN V(1)
* .END

```

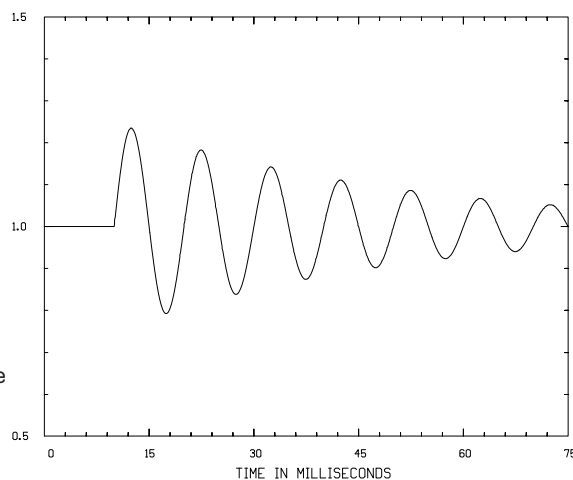


Figure 5. Output from sine wave transient analysis.

Please note the scales for the following plot of the output from the damped sine wave transient analysis. Because I used my own plot routines, I was able to scale to the factors that I wanted. When you run WinSpice3 it will automatically scale for you and you may get a slightly different looking graph. The importance is that the information comes from the program in a graphical

I am going to take the liberty of skipping the other forms of transient analysis for this tutorial due to restricted space. If there is enough demand for the information I would be glad to write it up at a later time. Again, there are books available that contain the details if you feel the need to pursue it further. The remaining forms are rarely used in amateur work.

DIODES

A diode in SPICE is modeled after all the physics of semiconductor materials for a PN junction. There are a number of parameters that go into the behavior of all the kinds of diodes that you can use. The difficulty is finding a model (the values of all the parameters needed) for the part number that you may have but fortunately the more popular diodes such as the 1N914 or 1N4148 silicon diodes are readily available. Models for the 1N34A and some others are more difficult to come by and may only exist in expensive libraries of models only available in the larger more expensive commercial versions of SPICE. I have spent considerable personal time surfing the Internet collecting all the models of diodes, transistors, and anything else that I could find to keep on hand for SPICE simulations.

In order to use a diode in SPICE you need two lines. The first is similar to the input lines for a resistor or other component used to specify the name and nodes but instead of specifying a value you point to a model of the diode. The line has the form

```
Dname N+ N- MODELNAME [AREA] [OFF] [IC=VD]
```

where

- **Dname** — the name for the diode in the circuit
- **N+** — node number for the anode of the diode
- **N-** — node number for the cathode of the diode

- **MODELNAME** — the modelname used to setup all the parameters for the model representing the diode
- **AREA** — an optional parameter of concern to silicon manufacturers on the area for the junctions inside a device
- **OFF** — an initial condition for DC analysis of the circuit
- **IC=VD** — the initial voltage across the diode for any transient analysis

The general format for the model of a junction diode is

```
.MODEL MODELNAME D(PAR1=PVAL1 PAR2=PVAL2 ... )
```

where **MODELNAME** is the name to be referenced in the line defining the diode in the circuit. The parameter names, **PAR1**, etc. that are available and the parameters that they represent are from the following list of 14 possible references. You do not have to give them all when setting parameters for a diode. Some of the parameters have default values and hopefully you won't have to worry about this if you already can find a complete model for the diode that you are using.

- **IS** — saturation current
- **RS** — ohmic resistance
- **N** — emission coefficient
- **TT** — transit time
- **CJO** — zero-bias junction capacitance

- VJ — junction potential
- M — grading coefficient
- EG — activation energy
- XTI — saturation-current temperature exponent
- KF — flicker noise coefficient
- FC — coefficient for forward-bias depletion capacitance formula
- BV — reverse breakdown voltage
- IBV — current at breakdown voltage

```
.DC VIN -1.0V 0.50V 0.01V
*
* Diode Model for 1N34A
*
.MODEL 1N34A D(BV=75 CJO=0.5E-12 EG=0.67
+ IBV=18E-3 IS=2E-7 RS=7 N=1.3 VJ=0.1
+ M=0.17 )
.END
```

So, here are two samples of the use of diode models, one for the 1N34A or 1N270 Ge-diode and one for the 1N4148 or 1N914 Si-diode. These were created to check out the model with a real live diode setup in the following test circuit.

```
Diode Model Test of 1N4148
*
VIN 1 0 DC 1
VTST 1 2 DC 0
D1 2 0 D1N4148
.PRINT DC V(1) V(2) I(VTST)
.DC VIN -1.0V 1.00V 0.02V
.MODEL D1N4148 D(IS=0.1p RS=16 CJO=2p TT=12n
+ BV=100 IBV=0.1p)
.END
```

```
1N34A DIODE MODEL
*
* 1N34A.cir Mar 1999 Chuck Adams, K7QO
*
* Test 1N34A Model Circuit
* 1N34A Ge Diode Model from Roy Lewallen, W7EL
*
VIN 1 0
*
* Voltage source setup to be used as ammeter
*
VTST 1 2 DC 0
D1 2 0 1N34A
.PRINT DC V(2) I(VTST)
.PLOT DC V(2) I(VTST)
*
* Voltage from -1V to +0.5V in 0.01V step
*
```

The following schematic was used to perform the experiment to measure the voltage-current curves for both the 1N34A and 1N4148 diodes and compare with the simulation results for the diode models. Care was used in accounting for current and voltage drops due to instrumentation and time was taken to avoid thermal junction effects where possible.

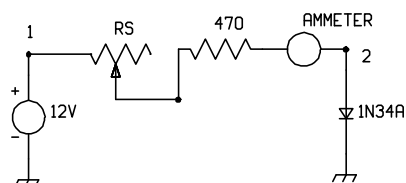


Figure 6. Diode test circuit.

Then the following graph was prepared from the output of the simulation runs and the measured data from the experiment. Excellent agreement was obtained, and variations may be attributed to variations in manufacturing processes. In order to further investigate the models versus actual physical measurements, a more detailed and traceable history of the part (some may have been manufacturing rejects) must be made and a greater number of samples measured. The purpose here is to illustrate the process of modeling and comparison to real physical results. Some parameters in the models may be changed carefully to match the measurements but with great care. The solid curves are from the SPICE model simulations, and the data points are from the physical measurements.

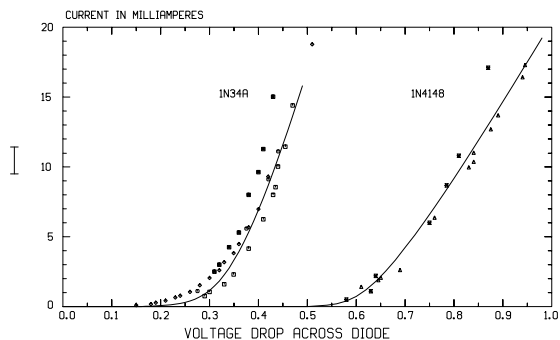


Figure 7. 1N34A and 1N4148 diodes.

You are now at the point whereby you can model simple circuits that include the diode. You can now model an RF probe, a half-wave rectifier, a full-wave rectifier, a voltage doubler, and various other circuits that involve the use of diodes and passive devices in their circuits. So take some time and look in the ARRL Handbook and see if there are some circuits that you can now simulate. It will be fun. Only with experimentation with the program can you really grow to appreciate its capabilities.

Bipolar Junction TRANSISTOR — BJT

The next device to look at is the transistor in SPICE. The first one to consider is the BJT. The format for the device that we will use is

```
Qname NC NB NE MODELNAME
```

where the files are

- NC — node number to which the collector is connected
- NB — node number to which the base is connected
- NE — node number to which the emitter is connected
- MODELNAME — model name for the transistor parameter definitions

The BJT model line has the format similar to the diode model line, but has one for NPN transistors and one for PNP.

```
.MODEL MODELNAME NPN(PAR1=PVAL1 PAR2=PVAL2 ... )
.MODEL MODELNAME PNP(PAR1=PVAL1 PAR2=PVAL2 ... )
```

but there are 40 model parameters that may be used to completely describe the transistor. I'll pass on listing them as I did for the diode. Let me just show you a sample of three commonly used transistors.

```
.model PN2222A NPN(Is=14.34f Xti=3 Eg=1.11
+ Vaf=74.03 Bf=255.9 Ne=1.307 Ise=14.34f
+ Ikf=.2847 Xtb=1.5 Br=6.092 Nc=2 Isc=0
+ Ikr=0 Rc=1 Cjc=7.306p Mjc=.3416 Vjc=.75
+ Fc=.5 Cje=22.01p Mje=.377 Vje=.75
+ Tr=46.91n Tf=411.1p Itf=.6 Vtf=1.7 Xtf=3
+ Rb=10)
* Fairchild case=T092
```

```
.model 2N3904 NPN(Is=6.734f Xti=3 Eg=1.11
+ Vaf=74.03 Bf=416.4 Ne=1.259 Ise=6.734f
+ Ikf=66.78m Xtb=1.5 Br=.7371 Nc=2 Isc=0
```

```

+ Ikr=0 Rc=1 Cjc=3.638p Mjc=.3085 Vjc=.75 *
+ Fc=.5 Cje=4.493p Mje=.2593 Vje=.75 IB 0 1 DC 1MA
+ Tr=239.5n Tf=301.2p Itf=.4 Vtf=4 Xtf=2 VCE 2 0 DC 12V
+ Rb=10) Q1 2 1 4 2N2222A
* Fairchild case=T092 VT 4 0 DC 0

.model 2N3906 PNP(Is=1.41f Xti=3 Eg=1.11 .PRINT DC I(VT)
+ Vaf=18.7 Bf=180.7 Ne=1.5 Ise=0 Ikf=80m .DC VCE 0 10V 0.2V IB 0 1MA 200UA
+ Xtb=1.5 Br=4.977 Nc=2 Isc=0 Ikr=0 Rc=2.5 * model parameters for a 2N2222A transistor
+ Cjc=9.728p Mjc=.5776 Vjc=.75 Fc=.5 .model 2N2222A NPN(Is=14.34f Xti=3 Eg=1.11
+ Cje=8.063p Mje=.3677 Vje=.75 Tr=33.42n + Vaf=74.03 Bf=255.9 Ne=1.307 Ise=14.34f
+ Tf=179.3p Itf=.4 Vtf=4 Xtf=6 Rb=10) + Ikf=.2847 Xtb=1.5 Br=6.092 Nc=2 Isc=0
* Fairchild case=T092 + Ikr=0 Rc=1 Cjc=7.306p Mjc=.3416 Vjc=.75
+ Fc=.5 Cje=22.01p Mje=.377 Vje=.75
+ Tr=46.91n Tf=411.1p Itf=.6 Vtf=1.7
+ Xtf=3 Rb=10)
.END

```

Now once again don't be overwhelmed by the magnitude of data involved. Just stick with the standard model libraries or get models from the web. I'll have some pointers at the end of this tutorial, and later I'll have the data on my web page that you can download if you can surf. I'll show all the models and the associated files for this tutorial in the simulations here.

OK, the first thing that we need to do is test the models and look at the characteristic curves for say the PN2222A. What I'll do is show you the file that generates the data and a plot of the data, then I'll explain what is going on.

Here is the SPICE input file that generates the data for the curves for the PN2222A model.

```

GENERATE 2N2222A CE Curve
*
* 2N2222A.cir Chuck Adams, K7QO
*

```

Taking the output from the above program and plotting, we get the following graph:

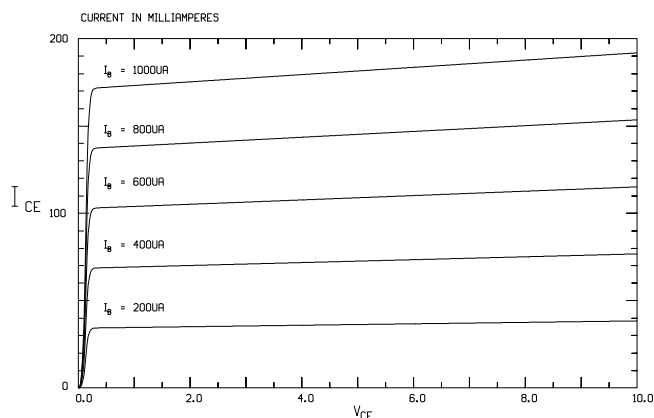


Figure 8. 2N2222A Curves.

What we have done in the SPICE source program is set up a current source I_B that will vary from 0 to 1mA in steps of $200\mu A$ while we vary V_{CE} from 0 to 10 volts. The good news is that we get the expected curves. The bad news is that SPICE does not tell us that if we do this experiment with a real 2N2222A, it will not survive the high power levels and will almost instantly self-destruct when it reaches a critical power level. So even though SPICE is a great program, it does have its limitations.

MODELING AN IF AMPLIFIER

OK, now we can get down to some actual real modeling of a circuit that we can use in a receiver as an IF amplifier, since we now have everything that we need in SPICE to do it. And since you are relatively new at this game, I would not expect you to develop a circuit from scratch. So, let's go to the ARRL "Solid State Design for the Radio Amateur" written by Wes Hayward, W7ZOI, and Doug Demaw, W1FB, and look at a simple Class A RF amplifier found in Figure 7 on page 21. I am going to modify the circuit slightly to reduce the overall power consumption. Here is the circuit with modification.

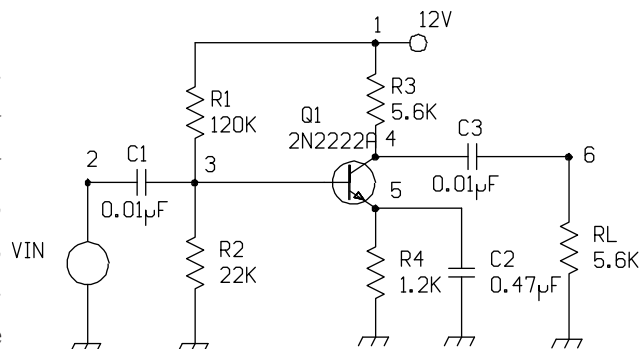


Figure 9. Simple Class A RF amplifier.

Here is the SPICE input file representing the above circuit.

```
Single Stage Amplifier using 2N2222A
*
* Modeled after Wes Hayward's RF amplifier
*
* amp2.cir   Mar 1999   Chuck Adams, K7Q0
```

```

*
Vcc 1 0 12

VIN 5 0 AC 1

R1 1 3 120K
R2 3 0 22K
R3 1 2 5.6K
R4 4 0 1.2K
RL 6 0 5.6K

C1 5 3 0.01uF
C2 4 0 0.47uF
C3 2 6 0.01uF

Q1 2 3 4 Q2N2222A

.AC DEC 10 1E3 1E9

.PRINT AC VDB(6)
.OPTION LIMPTS=5000

.MODEL Q2N2222A NPN(Is=14.34f Xti=3 Eg=1.11
+ Vaf=74.03 Bf=255.9 Ne=1.307 Ise=14.34f
+ Ikf=.2847 Xtb=1.5 Br=6.092 Nc=2 Isc=0
+ Ikr=0 Rc=1 Cjc=7.306p Mjc=.3416 Vjc=.75
+ Fc=.5 Cje=22.01p Mje=.377 Vje=.75
+ Tr=46.91n Tf=411.1p Itf=.6 Vtf=1.7
+ Xtf=3 Rb=10)

.END

```

Running the simulation in SPICE yields a gain vs frequency plot of:

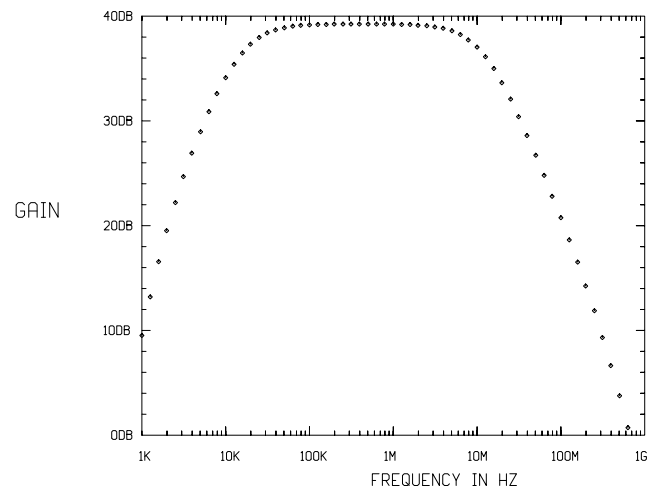


Figure 10. Gain vs Frequency for the amplifier.

In the process of making modifications to the circuit I managed to reduce the power consumption down from 86mW to 12mW with a change in the resistances and by increasing the value of C2 also increased the bandwidth. C2 will be difficult to get to $0.47\mu\text{F}$ without an electrolytic but the point is to show that by simple modifications to existing circuits one can experiment and find behavior that would require more time to find on the workbench.

If we were to use the above circuit as an IF amplifier in a receiver we would not require the increased bandwidth and would be happy with the original $0.01\mu\text{F}$ value. Just to show you how this works, here is the another graph with only the

value of C2 changed to a lower value of $0.022\mu\text{F}$. At first it does not look encouraging, but remember an IF frequency between 4.0MHz and 10.0MHz the amplifier would do nicely with the gain at least 36dB.

I could spend another three or four pages showing studies that I have made with cascaded amplifiers and a configuration called cascode, but I'd like for you to further your advances by doing these yourself. This purpose of this tutorial is to introduce you to some of the things that can be done and hopefully you will be motivated to study more and use the program yourself.

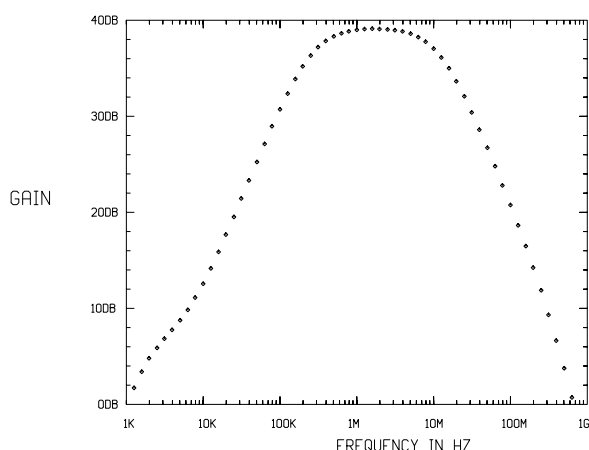


Figure 11. Gain vs Frequency.

OSCILLATORS

We have looked at amplifiers, now let's look at an oscillator and we get SPICE to do a simulation. I am going to take just a Colpitts Oscillator and show the schematic, the input file to SPICE, and the output of the oscillator.

First the schematic. This is one that I had shown on my web page for QRP-L on May 1, 1998 during the Elmer101 series. The Elmer101 series was a series of postings by a large number of individuals involved in building the Small Wonder Labs SW-40+ transceiver. In fact there were over 1,000 postings on the topic in 1998 just to point out how popular the series was. The purpose to get a large group of new builders started with a lot of mentoring from others via the Internet. My personal thanks to Dave Benson, NN1G, President and founder of Small Wonder Labs, in making it all possible with his effort to rapidly come out with a new kit that helped the project.

So, I took the VFO section of the SW-40+ as shown here and modeled it with some modifications so as not to have to do all the varactor issues. It is presented here as a starting point for experimenters and SPICE modelers to use in modeling other VFOs in the literature or development of a new one. This schematic appeared in the Autumn 1998 issue of QRPp on page 10. Actually I had to go back and redo it as I lost the original file somewhere in juggling between systems at work at the time and when I left.

So don't hold me down to the micronic detail (I made that up obviously). This is not rocket science.

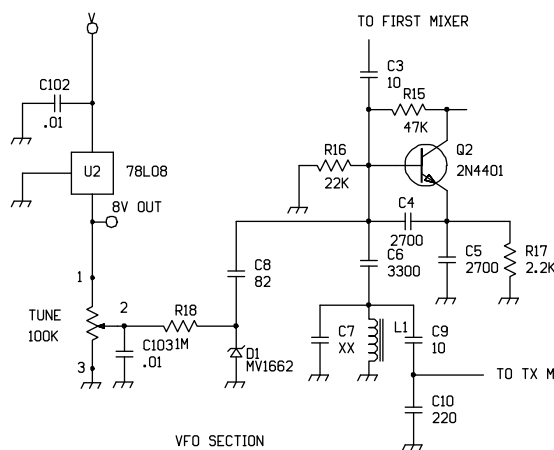


Figure 12. Colpitts Oscillator Schematic.

```

*
* The following diode represents the varactor
*
CD1 5 0 20p

R60 5 0 100MEG
C6 1 4 3300p
L1 4 0 3.36u
Q1 2 1 3 Q2n2222a
R15 2 1 47k
R16 1 0 22k
R17 3 0 2.2k
C4 1 3 1300p
C5 3 0 3300p

.TRAN 10n 18u 0 1n
.PRINT TRAN V(4)
.OPTIONS ITL5=50000,LIMPTS=50000
*
.END

```

And from this I created a file for input to SPICE that looks like:

NN1G Colpitts Oscillator Circuit

```

* NN1G Colpitts Oscillator in SW-40+ Transceiver

.model Q2N2222A NPN(Is=14.34f Xti=3 Eg=1.11
+ Vaf=74.03 Bf=255.9 Ne=1.307 Ise=14.34f
+ Ikf=.2847 Xtb=1.5 Br=6.092 Nc=2 Isc=0
+ Ikr=0 Rc=1 Cjc=7.306p Mjc=.3416 Vjc=.75
+ Fc=.5 Cje=22.01p Mje=.377 Vje=.75
+ Tr=46.91n Tf=411.1p Itf=.6 Vtf=1.7 Xtf=3
+ Rb=10)

Vcc 2 0 7.5V

C8 1 5 120p

```

In the simulation I arbitrarily picked the voltage to display in the following graph and in the .PRINT command in the source file as the top of L1. The purpose here is to show the startup characteristics of the oscillator and the fact that it does resonate quite nicely. Also if you run this simulation on WinSpice3 and expand the display you can determine that the resonant frequency is slightly above 3.0MHz. I like to call it 3.040MHz, since the IF frequency is 4.000MHz and that would put the rig on 7.040MHz. Just for fun.

So here is the graph that I get from the output of the voltage at node 4 at the top of L1 in the

schematic.

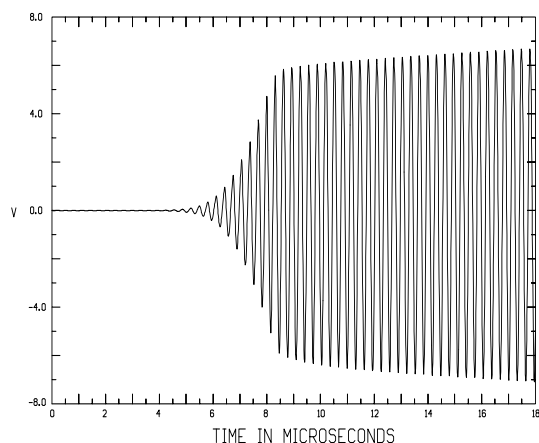


Figure 13. Voltage at L1 in the oscillator.

One of the details that I can not show here is that if you have SPICE running on a workstation and maybe even on a PC with LINUX all running with the X-window system SPICE3F4 has an interface whereby you can point and click on anypoint of the graph and get the exact data at that point. But even if you don't have such a system you can use the PRINT command to get all the data and then search for the point that you need. It's all there you just may have to put in a little extra effort to get it.

Because of the nature of this tutorial and a time and space constraint I am limited in just how much material to cover. I have chosen some ar-

reas that I hope are of interest and will get you started in the right direction. We are looking at the tip of the iceberg on all the functionality of SPICE. It obviously has a lot to offer but like any large software project you must invest a lot of time and research to become both functional and literate about its capabilities. I read somewhere that it takes a minimum of 40 hours to learn a new software package just to the point of getting comfortable in using it. So it is with SPICE.

I conclude with a list of books that I have in my library that I refuse to part with. Most of the books are out of print but maybe you will run across one at a swap meet or a book store in your travels and will latch onto it. Check with you local community and university libraries as they just might have a copy on the shelves. I don't have a favorite as they each have some new view on a topic not found in any of the other books. Start with the one(s) that you can find on the web bookstores or you can afford. You have my email address. Let me hear from you.

dit dit de K7QO

Books for Reference

This list is not and as I update will never be complete. So many books and so little time.

Banzhaf, Walter., "Computer-Aided Circuit Analysis using SPICE". New Jersey, Prentice Hall. 1989. pp 307 in paperback. Emphasis on SPICE on mainframe computers with ASCII graphics. This of interest to those with old version of SPICE in FORTRAN.

Chattergy, Rahul, "Spicy Circuits - Elements of Computer-Aided Circuit Analysis". Florida, CRC Press. 1992. pp 241 in hardcover. Jury is still out on this book. I need time to work through it. I know the author and the series editor. Small world.

Connelly, J. Alvin, and Pyung Choi, "Macro-modeling with SPICE". New Jersey, Prentice Hall. 1992. pp 273 in hardcover. Has floppy with models but 5.25" floppy was unreadable as most likely damaged by magnetic fields with checkout system in the store.

Fenical, L.H., "P Spice - A Tutorial". New Jersey, Prentice Hall. 1992. pp 344 in paperback. A good book for the beginner with plenty of examples with schematics, SPICE output, and graphical output from PSpice.

Foty, Daniel, "MOSFET Modeling with SPICE". New Jersey, Prentice Hall. 1997. pp 653 in hardcover. With a MSRP of \$104, this is by far the most expensive of the lot. Same class

as the Massobrio book below in that hams are not the intended audience for this book.

Keown, John, "MicroSim PSpice and Circuit Analysis". New Jersey, Prentice Hall. 1998 in paperback. With CD evaluation copy of PSpice and Schematic.

Kielkowski, Ron, "Inside SPICE". New York, McGraw-Hill, Inc. 1998 hardcover. With CD with software that is freely distributable and is most likely the same as the one at ftp.lehigh.edu. See notes below on web sources for SPICE and PSpice.

Lamey, Robert, "The Illustrated Guide to PSpice". New York, Delmar Publishers Inc. pp 219 in paperback. With floppy with all examples.

Massobrio, Giuseppe and Paolo Antognetti, "Semiconductor Device Modeling with SPICE". New York, McGraw-Hill, Inc. 1993. pp 479 in hardcover. Written for people in semiconductor manufacturing and their modeling needs.

Monssen, Franz, "MicroSim PSpice with Circuit Analysis". New Jersey, Prentice Hall. pp 548 in paperback. Excellent examples and graphics.

Rashid, Muhammad H., "SPICE for Circuits and Electronics using PSpice". New Jersey, Prentice Hall. 1995. pp 364 in paperback.

Roberts, Gordon W. and Adel S. Sedra, "SPICE Second Edition". New York, Oxford University Press. pp 447 in paperback. I like this book

because of the ASCII output from SPICE and graphics and samples with PSpice also.

Tuinenga, Paul W., "SPICE: A Guide to Circuit Simulation And Analysis Using PSpice". New Jersey, Prentice Hall. 1995. pp 288 in paperback. Another good beginner book although weighted heavy to PSpice since author is shown as an employee of MicroSim Corporation which was bought by another company just recently.