EE122 **Handout # 6**
Spring 97/98
Prof. Greg Kovacs

BASIC GUIDE TO VIRTUAL ELECTRONICS IN SPICE

Much of the following material was adapted from the **SPICE Version 2G User's Guide** (from the Department of Electrical Engineering and Computer Sciences at the University of California at Berkeley). This is not meant as a replacement for a complete set of Spice documentation, just as a quick overview to help you with your prelabs, etc.

VIRTUAL INSTRUMENTS

VIRTUAL POWER SUPPLY

The equivalent to a power supply in Spice is a statement like the following:

```
Vsupply 1 0 12V
```

This statement provides 12 V DC with the positive output connected to node 1 and the negative output connected to node 0. To get bipolar 12 V supplies with the +12V output to node 1 and the -12V output to node 2, you would use:

```
Vsupply 1 0 12V
Vsupply 0 2 12V
```

(note that the first node listed is always the *positive* output)

AC sources can also be defined in a similar way, such as:

```
Valternating 1 0 AC 1V  (this gives 1V AC between nodes 1 and 0)
```

Note that you can sweep the frequency of the AC source for frequency response analysis (see below in the virtual spectrum analyzer section).

VIRTUAL PULSE GENERATOR

The Spice "PULSE" command lets you set up a virtual pulse generator. You must specify the amplitude, delay (from time = 0 to the first pulse), the rise and fall times, the pulse width, and the period. The pulse train generated will repeat indefinitely for the duration of the transient analysis you select.

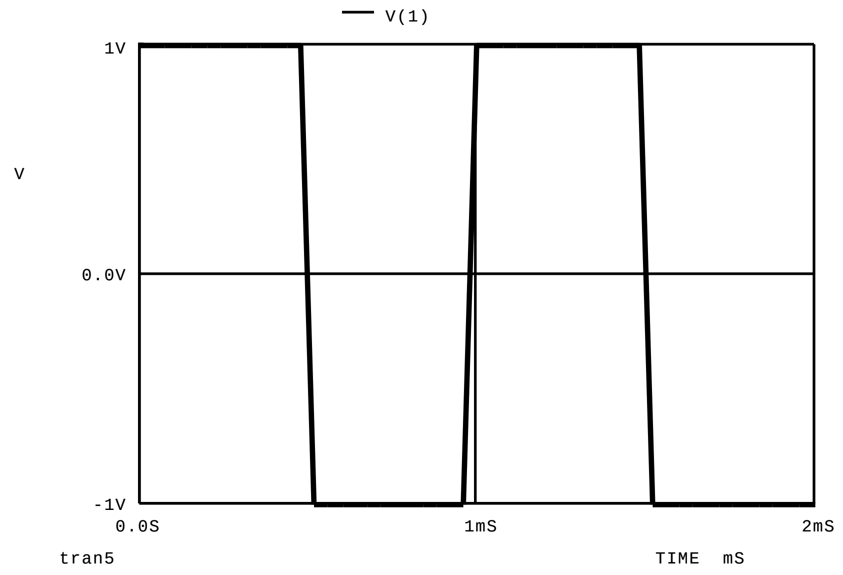
The parameters for the "PULSE" statement used as shown below:

pulse(V1 V2 TD TR TF PW PER)

where

V1 = the initial amplitude
V2 = the pulse amplitude
TD = the delay time in seconds
TR = the rise time in seconds
TF = the fall time in seconds
PW = the pulse width in seconds
PER = the period in seconds
An example of such a pulse generator is shown below. It is set up as a 2 V peak-to-peak, symmetrical squarewave connected to a 100 Ω load resistor to ground. It is set up to have 5 nS rise and fall times, a pulse width of 500 µS, and a period of 1 mS (in other words, it is the same as the output of a signal generator set to 1 KHz).

```
Virtual Pulse Generator Demo
VIN 1 0 pulse(-1V 1V 0nS 5nS 5nS 500uS 1mS)
R1 1 0 100
.TRAN 1uS 2mS 0S 100uS
.END
```
The resulting output from Spice is:



VIRTUAL SINEWAVE GENERATOR

The Spice "SIN" command lets you set up a virtual sinewave generator (used for TRANSIENT analysis only!)[1]. You must specify the offset voltage, peak amplitude, frequency, delay from time zero, damping factor, and phase (the defaults for the last three are zero and you usually do not enter anything for them!).

The parameters for the "SIN" statement are shown below:

```
sin (VO VA FREQ TD DF PHASE)
```

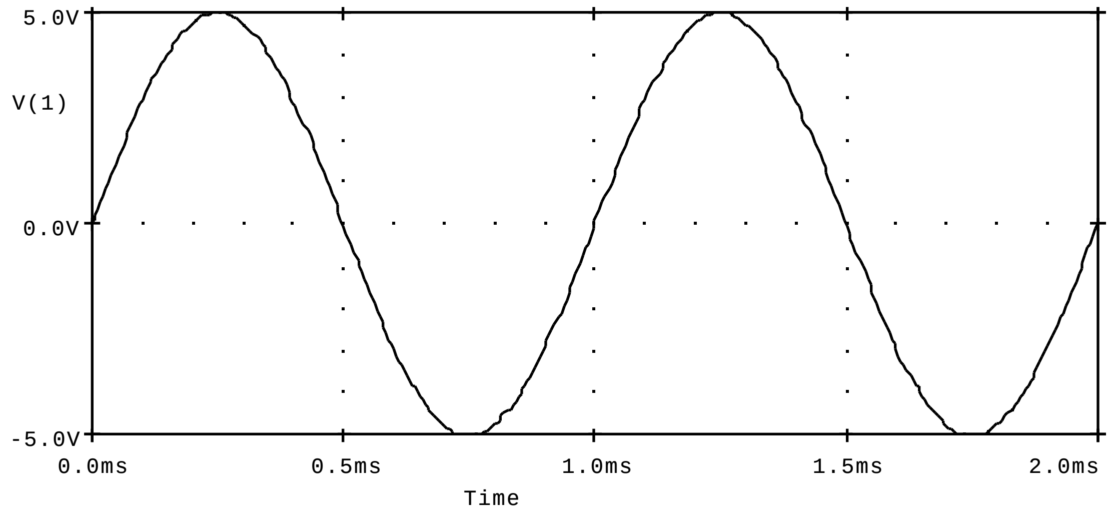
where,

VO = the offset voltage (usually zero)
VA = the peak amplitude
FREQ = the frequency in Hertz
TD = the time delay in seconds (default if no parameter entered = 0)
DF = the damping factor in 1/seconds (default if no parameter entered = 0)
PHASE = the phase in degrees (default if no parameter entered = 0)

An example of such a sinewave generator is shown below. It is set up as a 10 V peak-to-peak, 1000 Hz sinewave connected to a 100 Ω load resistor to ground.

```
Virtual Sinewave Generator Demo
VIN 1 0 sin(0V 5V 1000Hz)
R1 1 0 100
.TRAN 5uS 2mS 0S
.END
```

The resulting output from Spice is:



[1]Note that you have to use an "AC" specification for frequency domain analysis since the "SIN" command will not generate any output in time domain analysis! An example of a proper A.C. input signal for frequency domain work is: Vin 1 0 AC 1V

VIRTUAL OSCILLOSCOPE

Since there are several versions of Spice available, we will start with the most generic method of obtaining a “plot” of the desired voltages or currents from a circuit. The Spice “.PLOT” statement causes a plot of the desired values to be printed using characters to form the plot (not very good to look at, but functional). An example of the use of “.PLOT” to obtain two DC and two AC voltages from a circuit is:

```
.PLOT DC V(1) V(2)
.PLOT TRAN V(3) V(4)
```

A method to obtain excellent graphical plots on the Macintosh™ or IBM™ computers using PSpice is to include the “.PROBE” statement in your Spice deck. This generates a special output file with the “.dat” suffix. This file can be read using the “Probe” program and beautiful graphical displays of selected values can be obtained and printed. The handout on PSpice describes the operation of the “Probe” program.

".TRAN" A FUN STATEMENT FOR TIME DOMAIN STUFF

The .TRAN command (shown above in the SPICE input decks) tells SPICE over what time interval you want to simulate your circuit operation when you have time-varying input waveforms. SPICE uses time zero as the reference starting point. The form of the .TRAN statement is:

```
.TRAN  TimeStep  TimeStop  <TimeStart  <TimeMax>>
```

TimeStart and TimeStop define the time boundaries of the simulation. TimeStart is optional, as SPICE assumes it is zero if it is unspecified. TimeStep is the "sampling interval" within the simulation window. TimeMax is the maximum step size or "sampling interval" that SPICE will use in doing the simulation, even when outside the defined simulation time window(s). TimeMax is a very optional parameter, and can only be defined if TimeStart is also defined, since it must be the fourth argument in the .TRAN statement.

Example:

```
.TRAN 5uS 2mS 0S
```

This statement (from the virtual sinewave generator example input deck above) instructs SPICE to begin the simulation at time zero (0S), and to "sample" every 5 µsec (5uS) for 2 msec (2mS). Once SPICE has "sampled" these points, then the .PROBE command can be used with PSpice to store the data for later graphical analysis with the "Probe" program, or graphs can be generated immediately within MacSpice™ Professional.

".DC" ANOTHER USEFUL STATEMENT

Another useful statement is the .DC statement. It is used to sweep a DC voltage at a given power supply. (This is useful in computing transfer functions, for instance.) The form of the .DC command is:

```
.DC  Vxxxxxxx  StartVoltage  StopVoltage  VoltageIncrement
```

Example:

```
Vshock  3  7  DC  0.0V
.
.
.DC  Vshock  0.0V  5.0V  0.1V
.
.
.END
```

In the above statements, the DC power supply is connected from node 3 to node 7. The .DC command near the bottom tells SPICE to sweep the voltage at Vshock from 0 volts to 5 volts. The voltage increment is 0.1 volts, so SPICE will "sample" the circuit values every 0.1 volts from 0 up to 5 volts.. You can then use the Probe program (in PSpice) or the graphing commands in MacSpice™ Professional to look at any current or voltage in your virtual circuit as a function of voltage at the power supply Vshock.

VIRTUAL SPECTRUM ANALYZER

You can easily obtain the frequency response of a virtual circuit by sweeping the frequency of the previously defined AC power supplies using either of the following statements:

```
.AC DEC <# points per decade> <start frequency> <stop frequency>
.AC LIN  <total # points> <start frequency> <stop frequency>
```

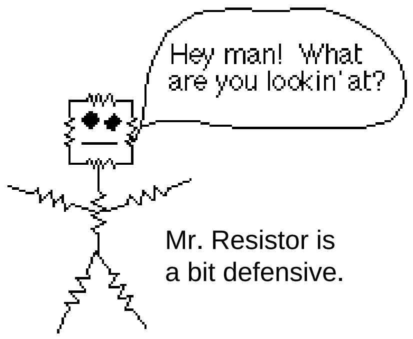
Example:

```
Vshock  3  7  AC  3.0V
.
.
.AC  DEC  50  100Hz  10KHz
.
.
.END
```

In the above statements, the AC power supply is connected from node 3 to node 7, and it has a magnitude of 3.0 volts. The .AC command near the bottom tells SPICE to sweep the frequency in a logarithmic fashion (i.e., decade sweep) with 50 points per decade. The start frequency is 100 Hz, so SPICE will "sample" (i.e., compute) 50 points varying the frequency logarithmically between 100 Hz and 1000 Hz, then 50 points between 1000 Hz and 10,000 Hz. You can then use the Probe program (in PSpice) or the graphing commands in MacSpice™ Professional to look at any current or voltage in your virtual circuit as a function of frequency at the power supply Vshock.

VIRTUAL PASSIVE COMPONENTS

SPICE circuits are described with virtual components. These include not only R's, L's, and C's, but also more fun stuff like transformers, diodes, and transistors. You can use these components along with virtual instruments (discussed later) to simulate such useful items as a **virtual toaster** (since toaster heating elements are just resistors).



Mr. Resistor is a bit defensive.

VIRTUAL RESISTORS

The syntax for telling SPICE where to put a resistor is:

Rxxxxxxx NodeA NodeB Value

This statement connects a resistor (labelled Rxxxxxxx) between NodeA and NodeB.

Examples:

R1 0 1 1000

This connects a 1000 Ω resistor from node 0 to node 1.

RTodd 0 1 1K

This connects another 1000 Ω resistor from node 0 to node 1. You can have as many resistors in parallel as you like.

R19 2 9 1

This connects a 1 Ω resistor from node 2 to node 9.

R19 2 9 1K

This connects a 1 KΩ (10^3 Ω) resistor from node 2 to node 9.

R19 2 9 1MEG

This connects a 1 MΩ (10^6 Ω) resistor from node 2 to node 9. Note here that you cannot simply use M as an abbreviation for "mega" in SPICE. M is used for "milli". [i.e. MVanilli = milli Vanilli] (See the capacitor section below.) MEG is the abbreviation for "mega" used in SPICE.

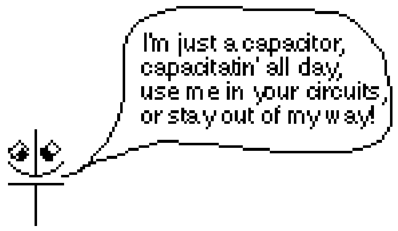
R19 2 9 1G

This connects a 1 GΩ (10^9 Ω) resistor from node 2 to node 9.

R19 2 9 1T

This connects a 1 TΩ (10^{12} Ω) resistor from node 2 to node 9.

VIRTUAL CAPACITORS



The syntax for telling SPICE where to put a capacitor is:

Cxxxxxxx PosNode NegNode Value [IC=<initial condition>]

This statement connects a capacitor (labelled Cxxxxxxx) between PosNode and NegNode. PosNode is the positive node, and NegNode is the negative node. These positive and negative designations generally do not matter for normal capacitors *except when you specify an initial condition!* The initial condition is simply a voltage; that is, <initial condition> is simply a number representing the initial voltage on the capacitor at the start (time-zero) of the simulation. The polarity of this initial voltage is determined with respect to the positive and negative nodes. *The moral here is that you don't need to agonize for hours over which node is positive and which is negative during circuit operation (this can be tricky to figure out for ac circuits); you only need to worry about the polarity with respect to an initial condition.* (The positive and negative designations also matter if you define a *nonlinear* capacitor, but we won't be doing that.)

Examples:

C1 7 5 1E-6

This connects a 1 µF capacitor between nodes 7 and 5.

CTodd 2 4 1.24UF IC=2.5V

This connects a 1.24 µF capacitor between nodes 2 and 4. The capacitor has an initial voltage of 2.5 V, with the positive side connected to node 2 and the negative side connected to node 4.

CTodd 4 2 1.24UF IC=-2.5V

This specifies the exact same thing as the previous statement! This also connects a 1.24 µF capacitor between nodes 2 and 4. The capacitor has an initial voltage of -2.5 V, with the positive side connected to node 4 and the negative side connected to node 2 (which is the same thing as saying that the capacitor has an initial voltage of 2.5 V, with the positive side connected to node 2 and the negative side connected to node 4).

Crap1 3 7 1MF

This connects a 1 mF (10^{-3} F) capacitor between nodes 3 and 7.

Crud1 3 7 1UF

This connects a 1 µF (10^{-6} F) capacitor between nodes 3 and 7.

Crumpet1 3 7 1NF

    This connects a 1 nF (10^{-9} F) capacitor between nodes 3 and 7.

Crust1 3 7 1PF

    This connects a 1 pF (10^{-12} F) capacitor between nodes 3 and 7.

Cripes1 3 7 1FF

    This connects a 1 fF (10^{-15} F) capacitor between nodes 3 and 7.

VIRTUAL INDUCTORS

Inductor campfire stories...

The syntax for telling SPICE where to put an inductor is:

Lxxxxxxx PosNode NegNode Value [IC=<initial condition>]

This statement connects an inductor (labelled Lxxxxxxx) between PosNode and NegNode. As for the capacitor above, PosNode is the positive node, and NegNode is the negative node. Also, as for the capacitor above, these positive and negative designations generally do not matter for normal inductors *except when you specify an initial condition or if you're using the inductor as a sub-element of a transformer (see transformers below)!* The initial condition is simply a current; that is, <initial condition> is simply a number representing the initial current through the inductor at the start (time-zero) of the simulation. The direction of this initial current is determined with respect to the positive and negative nodes. The initial current flows from PosNode, through the inductor, to NegNode. (The positive and negative designations also matter if you define a *nonlinear* capacitor, but we won't be doing that.)

Examples:
(The syntax for inductors and capacitors is very similar, so see the above capacitor examples for a bit more syntactical detail.)

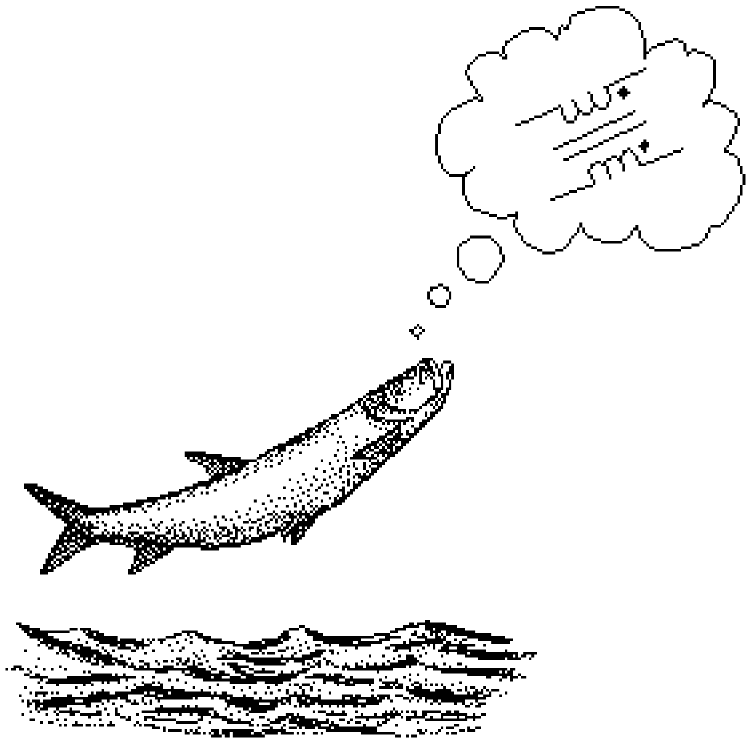
L1 7 5 1E-3

    This connects a 1 mH inductor between nodes 7 and 5.

LTodd 2 4 1.24MH IC=2.5MA

This connects a 1.24 mH inductor between nodes 2 and 4. The inductor has an initial current of 2.5 mA, with the initial current flowing from node 2, through the inductor, to node 4.

VIRTUAL TRANSFORMERS



Since transformers are not funny at all, we substituted a picture of Wally the Trout thinking about a transformer.

The syntax for telling SPICE where to put a transformer is not as straightforward as above. Instead of defining a transformer as a stand-alone circuit element, SPICE allows you to "wire up" all of the individual inductors you want, and then allows you to specify coefficients of coupling between all of the inductors. This makes sense if you recall that a transformer is simply two coupled inductors. Thus, to "wire up" a single transformer you "wire up" two individual inductors (paying careful attention to the polarity of the nodes) and then define the coefficient of coupling between them in a separate statement. (This is more powerful than having a simple transformer stand-alone component, because you can have coupling between all of the inductors in a circuit. The coefficient of coupling between two inductors in a circuit defaults to zero.) *To determine the correct polarities for the inductors, you can use the "dot" convention and place a "dot" on the positive node of each inductor.*

The syntax for telling SPICE the coefficient of coupling between two inductors in a circuit is:

Kxxxxxxx Lyyyyyyy Lzzzzzzz CouplingValue

Examples:

K1 L19 LTodd 0.99
The coefficient of coupling between inductors L19 and LTodd is 0.99.

KXFRMR LA LV 0.022
The coefficient of coupling between inductors LA and LV is 0.022.

SWEEPING COMPONENT VALUES

Sometimes, you want to see what the effects of varying a component value would be on the performance of a circuit. For DC analysis (such as determining the bias point of a circuit), this can be equivalent to adjusting a potentiometer or a variable capacitor. In SPICE, sweeping a component value is analogous to sweeping a DC voltage using the .DC statement. In fact, it uses the .DC statement! An example of the "conventional" use of the .DC statement is:

```
.DC Vbias 0 6 0.1
```

This would sweep the voltage "Vbias" from 0 to 6 volts in 0.1 V increments.

To sweep a resistor, you would first set up a .model statement for the swept resistor and then sweep its resistance multiplier parameter "r." An example is shown below where the resistor Rp1 is swept from 100 Ω to 1 KΩ in 10 Ω steps:

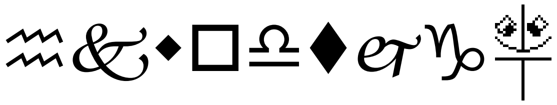
```
Rp1  3 4 rmod 1
.model rmod res(r=100)
.DC res rmod(r) 100 1K 10
```

(In case you are wondering, the "1" after rmod is its value, 1 Ω, which is multiplied by the resistance multiplier "r" which is set to 100 initially in the .model statement.)

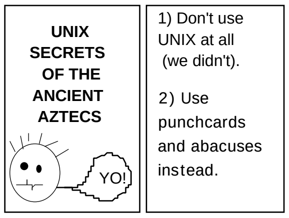
Similarly, other model parameters can be swept, such as those of capacitors and inductors. Who knows, in some versions of SPICE, even semiconductor parameters might be "sweepable."

VIRTUAL ACTIVE COMPONENTS

The previous section documented the "passive" virtual components. If these were the only components SPICE could deal with, then we would not be the high-tech society that we are today. For example, the ancient Babylonians were simulating RLC circuits using abacuses around 700 B.C. The following hieroglyphics transcribed from an ancient Mayan tablet demonstrates the state of the art in 400 A.D. (this is an early SPICE deck to simulate a Mayan Space Shuttle, which is how they explored the universe):



Fortunately, these days SPICE can simulate "active" components, i.e. solid state components such as diodes, bipolar junction transistors (BJTs), junction field-effect transistors (JFETs), and metal oxide semiconductor field-effect transistors (MOSFETs). Without these components we could not have such wonderful high-tech advances as Mr. Coffee™ and televangelism. So now we present...


UNIX SECRETS OF THE ANCIENT AZTECS
YO!
1) Don't use UNIX at all (we didn't).
2) Use punchcards and abacuses instead.

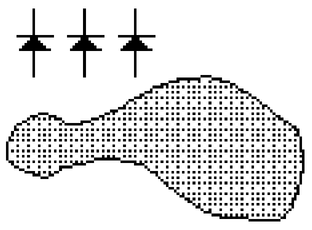
Virtual active components are really great, but they are significantly more complicated than virtual passive components. For instance, the most you have to worry about with inductors is specifying the initial currents and getting the polarities correct. However, for active components you need to specify *a lot* of different options. This can really be a pain in the ass if you have a number of diodes or transistors with basically the same characteristics, because you have to keep typing the same option values over and over. Fortunately, SPICE provides a way around this. You can define a basic diode or transistor model, and then when you are wiring up these active components in your circuit you can simply reference the model instead of re-typing the option values all of the time. In other words, all of the parameters that will be the same for a given set of active components can be defined together (as a basic model), and the parameters that will be different, such as transistor size, can be defined separately as you wire up each transistor.

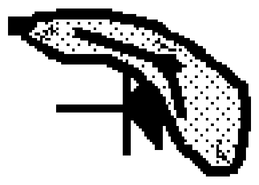
In EE 122 we will generally give you all of the transistor parameters that you need. We are showing you here not only how to simply wire up these virtual active devices, but also giving you some information on the various parameters that characterize a diode or BJT. Analog designers in the Real World[2] spend a great deal of time trying to optimize these parameters for a given chip. Any variations in the integrated circuit fabrication process can alter one or many of these parameters. This is one of the reasons why IC

[2]Since you're Stanford students, you don't have to worry about that for a while.... (JUST KIDDING!)

fabrication is done in a clean room: to minimize the potential variations on the IC fabrication process caused by dust, dirt and body oils.

VIRTUAL DIODE


Three Diodes and a Turkey Leg
(Salvador Dali's First Painting)


Big Diode and a Melted Watch
(Salvador Dali's Second Painting)

The basic syntax for wiring up a virtual diode is:

Dxxxxxxx PosNode NegNode ModelName <Area>

PosNode and NegNode define the positive and negative polarity of the diode, as well as which nodes it is connected between. The ModelName refers to a diode model (defined immediately below) that defines a wide range of device parameters for a diode. <Area> is an optional area factor, included here just to demonstrate that you can define diodes of different sizes that still have the same model parameters.

You define the diode model as:

.MODEL ModelName D <Option1> <Option2> ...

ModelName is the name of this particular model (i.e., set of diode parameters). The D tells SPICE that this is a model for a diode, so it will correctly interpret all of the options. The options allow you to set values for a large number of diode device parameters. Some of these diode parameters are given here:

Name	*Parameter*	*Units*	*Default*	*Example*
IS	Saturation Current	A	1.0E-14	1.0E-14
RS	Ohmic Resistance	Ω	0	10
TT	Transit Time	sec	0	100E-12
CJO	Zero-bias Junction Capacitance	F	0	2E-12
VJ	Junction Potential	V	1	0.6
BV	Reverse Breakdown Voltage	V	∞	40.0

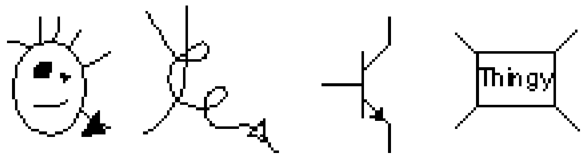
As pointed out earlier, you will generally be given these parameters and others whenever you need to wire up a virtual diode, but they are presented here so that the parameters do not seem so mysterious. Note that some of these parameters will be more fully defined as we proceed in EE 122.

On a practical note, most of these parameters are determined for actual diodes by fabricating a given diode (in a repeatable process) and using some of the resulting diodes to determine the above parameters. Then the circuit designer can use this information (in SPICE) to design circuits using these diodes, and the best circuit designers can suggest changes to the fabrication process that will optimize the resulting devices for a given chip.

Example:

```
DCLAMP  2  5  DMODEL5
DTodd  3  6  MyDiode
.
.
<Other components defined>
.
.
.MODEL  MyDiode  D  IS=1.0E-13  VJ=0.7
.MODEL  DMODEL5  D  BV=30.0  VJ=0.5
.
.
.END
```

VIRTUAL BIPOLAR JUNCTION TRANSISTOR (BJT)



The Four Finalists in Shockley, Bardeen, and Brattain's "Best Symbol for the Transistor Contest," 1952....

The basic syntax for wiring up a virtual BJT is:

```
Qxxxxxxx  CollectorNode  BaseNode  EmitterNode  ModelName  <Area>
```

CollectorNode, BaseNode, and EmitterNode define the node connections for the collector, base, and emitter of the BJT. The ModelName refers to a BJT model (defined immediately below) that defines a wide range of device parameters for a BJT. <Area> is an optional area factor, included here just to demonstrate that you can define BJTs of different sizes that still have the same model parameters. (Not shown here are options for setting initial conditions and defining a substrate node.)

You define the BJT model as:

```
.MODEL  ModelName  NPN  <Option1>  <Option2> ...
or
.MODEL  ModelName  PNP  <Option1>  <Option2> ...
```

ModelName is the name of this particular model (i.e., set of BJT parameters). The NPN or PNP tells SPICE that this is a model for an NPN or PNP BJT respectively, so it will correctly interpret all of the options. The options allow you to set values for a large number of BJT device parameters. Some of the more important BJT parameters are given here:

Name	*Parameter*	*Units*	*Default*	*Example*
----	---------	-----	-------	-------
IS	Transport Saturation Current	A	1.0E-16	1.0E-15
BF	Ideal Maximum Forward Beta	None	100	100
VAF	Forward Early Voltage	V	∞	200
ISE	B-E Leakage Saturation Current	A	0	1.0E-13
ISC	B-C Leakage Saturation Current	A	0	1.0E-13
RB	Zero-bias Base Resistance	Ω	0	100
RE	Emitter Resistance	Ω	0	1
RC	Collector Resistance	Ω	0	10
CJE	B-E Zero-bias Depletion Cap.	F	0	2E-12
VJE	B-E Built-in Potential	V	0.75	0.6
TF	Ideal Forward Transit Time	sec	0	100E-12
CJC	B-C Zero-bias Depletion Cap.	F	0	2E-12
VJC	B-C Built-in Potential	V	0.75	0.5
TR	Ideal Reverse Transit Time	sec	0	10E-9
CJS	Zero-bias Coll.-Substrate Cap.	F	0	2E-12

As pointed out earlier, you will generally be given these parameters and others whenever you need to wire up a virtual BJT, bu they are presented here so that the parameters do not seem so mysterious. Note that some of these parameters will be more fully defined as we proceed in EE 122.

Example:

```
QDETONATE  2  4  5  NPNMODEL1
QTodd  3  6  6  MyPNP
.
.
<Other components defined>
.
.
.MODEL  NPNMODEL1  NPN  VJE=0.7  BF=150  CJE=1PF
.MODEL  MyPNP  PNP  IS=1.0E-15  BF=50  VJE=0.6
.
.
.END
```

OTHER STUFF LIKE MOSFETS, JFETS, AND LUNCHEON MEATS

The virtual JFET and the virtual MOSFET will not be discussed here because they won't be used in EE 122. SPICE uses these in a fashion very similar to that of the virtual BJT.