LTSPICE simulating and testing circuits

Part 2: by Nicholas Vinen

This month, we build a flexible and realistic relay simulation in LTspice and then incorporate it into a simulation of the SoftStarter circuit, based on the power supply circuit shown last month.

Last month, we ended our first SPICE tutorial with a working model of the mains power supply from the SoftStarter, a project published in the April 2012 issue. It was designed to reduce the inrush current of mains devices, especially those with capacitor-input power supplies, such as desktop computers.

We commented that LTspice has no built-in ability to simulate the relay in that circuit, so to complete the simulation, we would need to create a relay simulation.

So we're going to show you how to do that this month. We'll start by creating a fairly basic relay simulation and introducing it into our test circuit, to demonstrate that it works.

We will then increase its flexibility and realism. Next time, we'll show you how to set up LTspice to simulate an NTC thermistor, letting us properly simulate the entire SoftStarter circuit.

We'll finish by taking a look at some of the other SPICE tools you'll need to understand in order to simulate even more complex devices.

We won't go over the fine details of using LTspice which have already been described last month, such as how to place components, wire them up and set their values.

If you need a quick refresh, re-read last month's article before diving into this one.

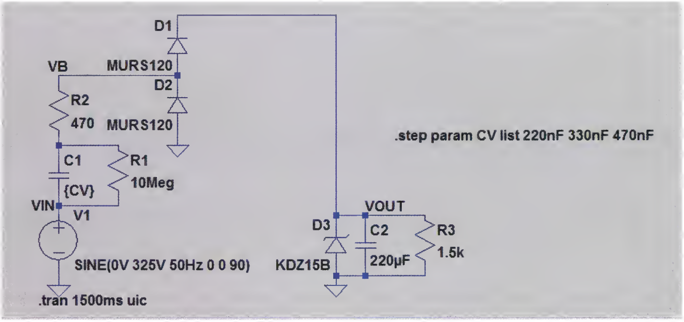


Fig.1: the final circuit from last month's LTspice article, which was very similar to the mains power supply for the SoftStarter from the April 2012 issue of SILICON CHIP.

1. Creating the relay simulation model

As explained last month, SPICE requires models for anything but the most basic components (resistors, capacitors and inductors) in order to properly simulate the properties of devices like diodes, transistors, Mosfets and so on.

But you can also build models for custom devices such as ICs which SPICE may not already have provision for. These are made by creating a "subcircuit" which is hidden inside a component symbol.

Our initial goal is to create a symbol for an SPDT relay with a 12V DC coil and get it to operate as you would expect. That is, initially the COM and NC terminals should be connected by a very low resistance while there should be a very high resistance between the COM and NO terminals.

Once the coil voltage rises sufficiently high (above the "must operate" voltage, about 9V for a 12V relay), those two resistances should be reversed, simulating the relay armature switching.

If the coil voltage then drops below the "must release" voltage (say 3V for a 12V relay), it should go back to its initial state. And the coil should draw a realistic current and should also be inductive, like a real relay coil, to properly test the driving circuitry.

So, launch LTspice and open up the circuit we finished with last month, named "tutorial1.asc". If you didn't go through last month's tutorial and create this file, you can download it from the SILICON CHIP website. The final circuit from last month is shown in Fig.1.

Now create a new, blank circuit for the relay subcircuit by selecting File→New Schematic from the main menu. Save it in the same directory as tutorial1.asc and call it "relay.asc". Start off the relay circuit by placing a resistor in series with an inductor, both arranged vertically. This will form the coil of our relay.

In order to determine their values, we had a look at the data sheet of a typical 2A relay, the Omron G5V-2 (available from element14, Cat 9949496). The data sheet gives the following typical values for a 12V DC coil relay: 41.7mA coil current, 288Ω coil resistance, 0.47H coil inductance (armature off), 0.74H coil inductance (armature on), must operate voltage: 9V and must release voltage: 0.6V.

So we can set our resistor value to 288 (ohms is implied) and for now, let's ignore the effect of the armature switching and just set the inductance value to 0.47 (Henries; you can add an H at the end if you want).

Now, we need to tell SPICE where the external relay connections will be. There will be five: two for the coil plus the COM, NO (normally open) and NC (normally closed) terminals. For the sake of simplicity, let's label the top end of the coil "+" and the bottom end, "-".

To do this, we use the "Label Net" tool in the toolbar, which looks like the letter A in a box. Click this, then type in "+". But before clicking OK, change the "Port Type" option to "Bi-Direct." (which allows signals/current to flow in both directions).

Click OK, then place this port right at the top of your series resistor/inductor combination. Then repeat the same steps to place a port labelled "-" at the other end. The result is shown in Fig.2.

That completes the coil simulation, for the moment, so let's go on to the relay contacts. These are simulated using two "voltage controlled switches".

To place the first one, click on the "Component" button in the toolbar (which looks like a logic gate), then scroll across until you can see the "sw" option. Click on this and you will see the description above says "Voltage controlled switch". Click OK.

Place the first one to the right of the coil components, with its top near the top of the coil, then place a second voltage-controlled switch immediately below it, so that its bottom is near the bottom of the coil. Draw a wire joining the two vertically adjacent switch contacts.

You can now label the top of the top-most switch "NO", the wire joining the two switches "COM" and the bottom of the bottom-most switch "NC", using the same procedure as you did to label the two ends of the coil. Don't forget to set them as bidirectional ports.

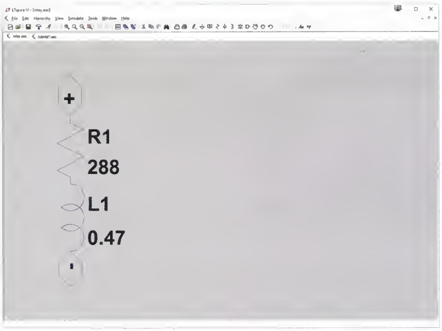


Fig.2: this shows the first part of our 12V DC coil relay with two external connections, modelled after the Omron G5V-2. The "+" and "-" labels are the names of two ports which are used to connect this fragment to the main circuit.

2. Configuring the switches

Besides two contacts, each voltage-controlled switch has terminals labelled + and -, to connect the control voltage.

Wire these up in parallel, ie, + to + and - to -. Then wire the + ends to the top of the coil and the - ends to the bottom of the coil. This is shown in Fig.3.

Now we need to describe how the switches should respond to the control voltages. To do that, we create two switch models and assign one to each switch. This actually turns out to be pretty easy.

The main parameters for a switch model are V_t (threshold voltage), V_h (hysteresis voltage), R_{on} (on-resistance), R_{off} (off-resistance) and I_{limit} (current limit).

You can see the whole set of parameters by accessing LTspice's built-in help (eg, press F1). Just type "sw" in the search box, press enter, then double-click on the "Voltage Controlled Switch" heading which appears below.

Now we create our switch model for S1. Let's call it SWa. Click on the SPICE Directive button in the toolbar (it says "op"), then type:

```
.model SWa SW(Ron=0.01
  Roff=10Gig Vt=6V Vh=3V
  Ilimit=2A)
```

After entering this, click OK and place the directive below the circuit components. This defines the on-resistance as 10mΩ, off-resistance (leakage) as 10GΩ, the switch-on threshold as 9V (V_t+V_h), the switch-off threshold as 3V (V_t-V_h) (in our experience, a realistic value for a 12V relay) and sets the current limit to 2A; LTspice will limit current through the switch to this figure during simulation.

Now right-click on S1 and change its "Value" parameter to "SWa". This tells SPICE to use that model for switch S1.

Using the same procedure, we'll create another switch model called SWb, as follows:

```
.model SWb SW(Ron=10Gig
  Roff=0.01 Vt=6V Vh=3V
  Ilimit=2A)
```

Note that all that's changed is that we've swapped the on-resistance and off-resistance values around, thus reversing the switch logic, ie, it will be off if the control voltage is above 9V and on if it's below 3V (in between, it will retain its previous state).

Having also placed this directive in the circuit, change S2's model to SWb. Your circuit should now look like Fig.4.

That completes our initial circuit defining how the relay works, so save it. Now we create a symbol for it, so we can place it in our main circuit.

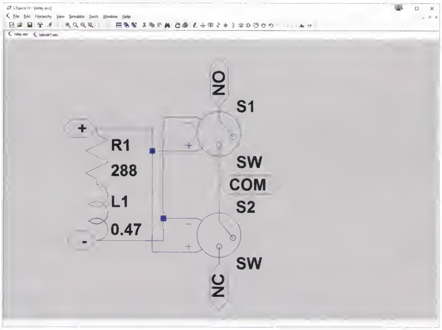
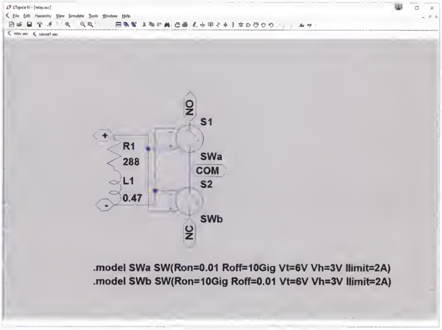


Fig.3: we have now added two voltage controlled switches (for NO and NC) to our simulated relay coil, with the common connection of the two switches connected to the common or COM port on the subcircuit.



```
.model SWa SW(Ron=0.01 Roff=10Gig Vt=6V Vh=3V Ilimit=2A)
.model SWb SW(Ron=10Gig Roff=0.01 Vt=6V Vh=3V Ilimit=2A)
```

Fig.4: these two switch models have been added to tell LTspice how the switches should behave during simulation, in response to their control voltages. These are added by clicking on the SPICE Directive button on the toolbar (at far right).

3. Creating the relay symbol

Select the item in the main menu titled Hierarchy→Open This Sheet's Symbol. When it asks if you want to automatically generate one, say Yes.

The result is shown in Fig.5. It has created a box with five ports, to match the five ports in the circuit, with a label on top. While we could use this in the circuit, it doesn't really look like a relay, so we might as well draw an improved symbol.

Start by deleting everything; select Edit→Delete, then drag a box around the whole lot. This may seem like it makes the exercise pointless but it hasn't, as we now have a symbol file in the right location.

Now choose Edit→Add Pin/Port (or just press "P" on the keyboard). For Label, enter "+" and for Pin Label Justification, choose LEFT. Click OK and place the port near the upper-left corner of the screen.

Add another Pin/Port using the same method, called "-" and place this directly below the "+" port, near the bottom-left corner of the screen.

Now we're going to place another port but just use it as a reference, so don't bother with labelling it. Just press "P", click OK, then place it two grid squares below the "+" port box.

Choose Draw→Line (or press "L" on the keyboard) and draw a vertical line, starting right in the middle of the "+" box and ending right in the middle of the unlabelled box.

Now use the Edit→Delete option to delete the reference port we just placed (drag a box around it). Repeat this procedure to draw a line of the same length up from the "-" port.

Next, use the Draw→Rect option (or press "R" on the keyboard) to draw a box touching the ends of the two lines and centred on them. You should have a result similar to that shown in Fig.6. That represents the coil of our relay.

Next, place three additional ports, to the right of the coil: one labelled "NO", BOTTOM aligned, to the right of the "+" port (in the same vertical position); one labelled "NC", TOP aligned, to the right of the "-" port and immediately below the "NO" port, and one labelled "COM", BOTTOM aligned, halfway between the other two.

You can now proceed to draw the lines shown in Fig.7, representing the relay contacts. Hint: once you've drawn the top half, you can use the Edit→Duplicate command, then rotate and flip it and drop it in place at the bottom to avoid repeating the work.

So that the symbol will appear with a component label next to it later, go to the Edit→Attributes→Attribute Window menu option, then click on InstName and then OK and place the name above the coil, as shown in Fig.7.

The symbol is now complete so save it.

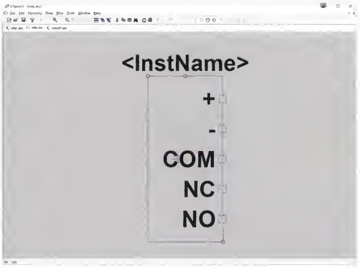


Fig.5: now we move onto creating the relay symbol by selecting Hierachy→Open This Sheet's Symbol on the menu bar in LTspice. This is the default symbol that is created.

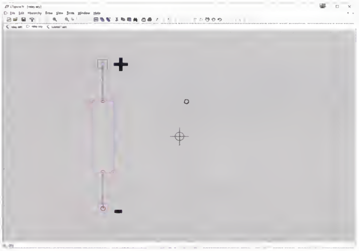


Fig.6: we could have used the default symbol but decided to instead build one that looks more like a relay symbol, starting with the coil, which is drawn with lines and boxes.

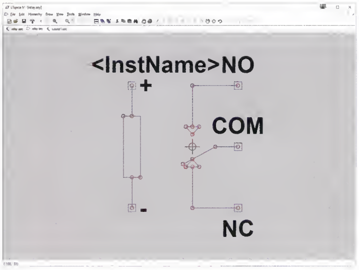


Fig.7: now we've added lines depicting the armature and normally open/normally closed switch contacts and placed the appropriate ports at the end of each line.

4. Using the relay model

Now that the relay model is ready to test, switch back to the "tutorial1.asc" tab, which will reveal our earlier circuit.

We had placed a 1.8kΩ resistor across diode D3 to provide a simulated load to the circuit. Since the relay coil will be a real load, we no longer need this resistor, so delete it, then use the File→Save As menu option to save the modified circuit as "tutorial2.asc".

Now to place the relay in the circuit. Click on the "Component" option in the toolbar (which looks like a logic gate), then at the top of the dialog, where it says "Top Directory", click on the directory name and select your User directory instead. Your new symbol should appear (see Fig.8). Click OK and place this so that you can wire it up across D3, then do so.

So that we can see when the relay switches in the simulation, wire the NO terminal to the coil +, the NC terminal to ground and connect a resistor between COM and GND and set its value to 1kΩ.

Right-click on the ".tran" directive and change the Stop Time to 500ms and Time to Start Saving Data to 0. Change C1 to 1μF, to ensure the power supply will be able to handle the relay load, then save the result. Your circuit should look similar to ours (Fig.9).

We can now run the simulation and if you plot the voltages at VOUT and the COM terminal of X1 (our relay), you should see something similar to Fig.10. The green trace shows the voltage at VOUT. Note how, as soon as it surpasses 9V, the relay switches on. VOUT then drops slightly due to the extra loading from R3 (1kΩ) but since it does not drop below 3V, the relay remains switched on.

You can now experiment by changing the value of R3 to determine what sort of load the circuit can handle before the relay will start to drop out and oscillate. We found the threshold to be just below 220 ohms (see Fig.11).

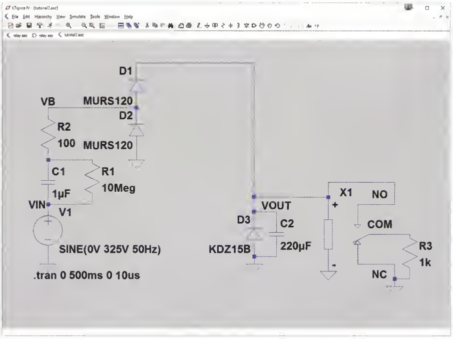
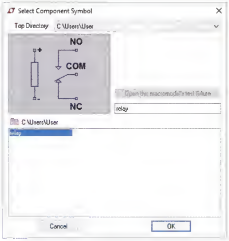


Fig.9: C1 must be changed to 1μF to ensure that the power supply can handle the relay's load, for a simulated 12V DC coil. This causes the circuit to draw more current from the mains on each cycle, keeping C2's voltage up.

Fig.8 (below): placing your new symbol in the LTspice circuit.


Select Component Symbol
Top Directory: C:\Users\User
NO
COM
NC
Open this macromodel's test fixture
relay
C:\Users\User
relay
Cancel
OK

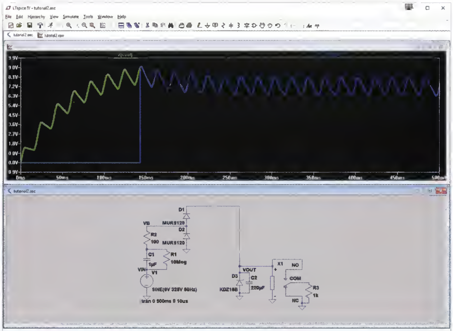


Fig.10: a plot showing the voltage between VOUT and the COM terminal of X1. Once the coil voltage is high enough, the simulated relay switches on and the supply voltage drops slightly, due to the current then flowing through R3.

5. Improving the relay model

We're now going to improve the relay model in two ways. Firstly, we're going to allow you to set the relay voltage when you place the symbol, allowing you to have multiple relays with different nominal coil voltages in the same circuit, if necessary.

Secondly, we're going to make it more realistic, by adding a switch-on delay, a break-before-make characteristic and varying the coil inductance when the relay switches.

Varying the nominal coil voltage requires us to vary the coil resistance, inductance and switch thresholds and hysteresis.

To do this, first switch back to (or re-open) "relay.asc" and then add a new directive (using the "op") button which reads:

.param Vcoil 12V

Place this in the circuit. This sets the default coil voltage to 12V but allows it to be overridden.

If we examine the G5V-2 data sheet, we can see that we can compute the coil resistance for a given voltage as 2 × V_{coil}^2.

The coil inductance (with armature off) can be approximated as V_{coil}^2 ÷ 300. The "must operate" voltage is 0.75 × V_{coil} while a typical drop-out voltage will be around 0.25 × V_{coil}.

Have a look at Fig.12. We have moved the switches over to the right to make more room (using the Drag tool) and then changed the values of R1 and L1 and the models for the two switches to contain expressions which calculate their new parameters based on the value of V_{coil}.

Note how the expressions used in component values are surrounded by braces "{}", which tells LTspice that it needs to evaluate these expressions at simulation time, to determine the values.

The model parameters are already subject to evaluation at simulation time, so no braces are added there; we simply substituted mathematical formulae based on V_{coil} for V_t and V_h.

Save the new model, then go back to the main circuit and right-click on the relay, X1. You can now check the box next to the "PARAMS:" label, then just to the right, type in "Vcoil=9V".

If you re-run the simulation, you will now find that the relay does not switch on, because VOUT does not exceed 6V, due to the lower coil resistance of the relay (162Ω).

You can now change C1 to 1.5µF and re-run the simulation. The relay will now switch on due to the increased coil voltage, at around 6.5V, and remains on since the minimum supply of around 3V is enough to keep the lower-voltage relay latched.

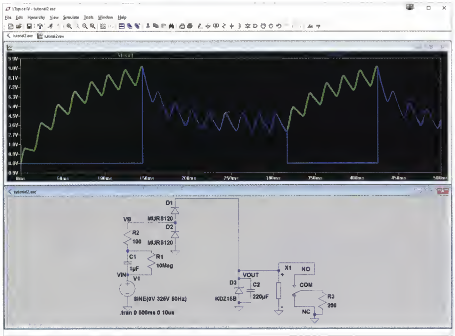
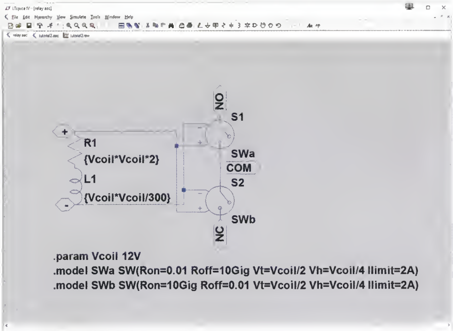


Fig.11: reducing the value of R3 causes the supply voltage to drop once the relay switches on, causing it to drop out and "chatter". This will allow us to determine the maximum load the circuit can handle before the relay drops out.



```
.param Vcoil 12V
.model SWa SW(Ron=0.01 Roff=10Gig Vt=Vcoil/2 Vh=Vcoil/4 Ilimit=2A)
.model SWb SW(Ron=10Gig Roff=0.01 Vt=Vcoil/2 Vh=Vcoil/4 Ilimit=2A)
```

Fig.12: we now add a parameter called "Vcoil" and change the switch models so they use this to calculate the switching thresholds. This will allow us to change the relay coil operating voltage when placing this subcircuit in another circuit.

6. Increasing realism

While it will have a negligible impact on this simulation, in some cases, attention to detail in the operation of the simulated component may be the difference between the simulation giving results that are true to life or not.

Since it isn't too difficult, let's incorporate the relay latching delay, break-before-make characteristics and coil inductance changes in case those are important later.

Our updated model is shown in Fig.13. We have disconnected the NO, COM and NC terminals from S1 and S2 and connected a 1V voltage source (V1) across the switches instead so that the junction of the two switches changes from 0V and 1V immediately when the relay should switch on.

This then passes through an RC filter comprising a 1GΩ resistor and 1pF capacitor. The very high resistor value and very low capacitance were chosen so that the capacitor charging current is insignificant compared to the coil current.

We need to connect the negative end of the capacitor to the coil negative end to keep the simulator happy (it doesn't like floating sections of the circuit; another option would be to make this connection with a high-value resistor).

The RC filter provides both a short delay and also allows the following switches, S3 and S4, to have different thresholds so that one will switch off before the other switches on.

The NO, COM and NC terminals are connected to S3 and S4 as they were connected to S1 and S2 before. But S3 and S4 use different, fixed control voltage switching levels.

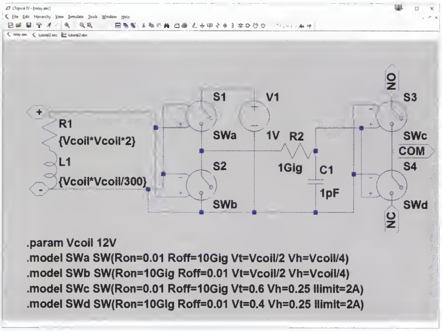
When the relay turns on, S4 switches off as C1 exceeds 0.15V and S3 switches on once it goes above 0.35V, giving the break-before-make action, simulating the motion of the armature through the space between the two contacts.

Similarly, at switch-off, the threshold for S3 is 0.85V while C1 must discharge further, to below 0.65V, before S4 switches back on.

The effect of these changes can be seen in the simulation shown in Fig.14, where the main circuit has been changed so that both the openings and closings of both contacts can be observed.

Note how the voltage at the NC terminal (green) drops to 0V (due to the 200Ω pull-down resistor) about 1ms (the transfer time) before the voltage at the NO terminal (blue) shoots up due to that contact closing.

The final relay.asc and relay.asy files can be downloaded from the Silicon Chip website, along with the tutorial2.asc circuit, as shown in Fig.10.



```
.param Vcoil 12V
.model SWa SW(Ron=0.01 Roff=10Gig Vt=Vcoil/2 Vh=Vcoil/4)
.model SWb SW(Ron=10Gig Roff=0.01 Vt=Vcoil/2 Vh=Vcoil/4)
.model SWc SW(Ron=0.01 Roff=10Gig Vt=0.6 Vh=0.25 Ilimit=2A)
.model SWd SW(Ron=10Gig Roff=0.01 Vt=0.4 Vh=0.25 Ilimit=2A)
```

Fig.13: the updated relay model shown above incorporates a switching delay and hysteresis. S1 & S2 produce a control voltage which passes through an RC filter. The resulting voltage then controls the simulated armature of S3 and S4.

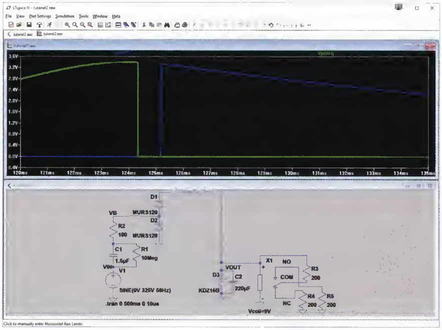


Fig.14: the green and blue lines above show the effect of the supply voltage at the COM terminal being switched to the NC and NO terminals. As you can see, the updated relay model now has a "break-before-make" characteristic.

7. Varying the coil inductance

This is a pretty small detail but in some cases, it might be important. As we mentioned earlier, a relay's coil inductance changes as it switches since the magnetic circuit is also changing.

However, this is pretty tricky to simulate in a generic way, since in some cases (such as the G5V-2), coil inductance increases with the armature on while in other cases, like the smaller G5V-1 version, it decreases. This depends on the relay's construction.

Fig.15 shows a modified version of the relay model which varies the inductance as it switches. A parameter called "Ldelta" controls the change in inductance; if it's positive, the inductance increases when the relay switches on by the proportional amount (ie, 0.5 = 50%) and if it's negative, it decreases the inductance by a similar amount.

To achieve this, we slowly switch a second inductor in parallel with the main inductor using a P-channel Mosfet. Unfortunately, SPICE lacks the concept of a voltage (or current) controlled resistance, so a Mosfet is the closest thing we have.

Voltage source V2 is used to provide the fixed gate bias to bring it on the edge of conduction while voltage-controlled voltage source E1 amplifies the relay control voltage to switch the Mosfet either on or off as the relay switches.

Formulas built into the various parameters shown below the circuit calculate the required secondary inductor value and Mosfet gate scaling coefficients to provide a smooth transition in inductance as the simulated relay switches. The changes in coil current profile over time for three different values of L_{delta} (0.5, -0.5 and 0.01) are shown above the circuit.

Note that you can not set L_{delta} = 0 as the formulas would break down. If you don't need this detail in your simulation, you're probably better off sticking with the simpler relay model which will be faster to simulate.

Building a complete SoftStarter circuit

The next tutorial will provide the information needed to finish and simulate the complete SoftStarter circuit.

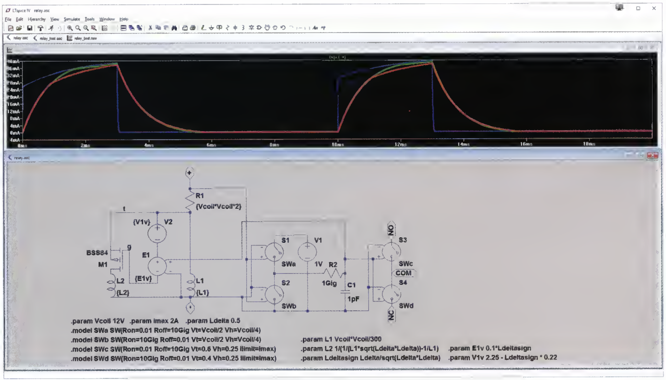
The critical piece we're still missing is the NTC thermistor. Simulating this is quite complex because it involves calculating the instantaneous dissipation, modelling the resulting heating, tracking the temperature and then reducing its resistance as the temperature builds.

This will involve designing several other very useful subcircuit building blocks which will no doubt come in useful for many other purposes.

These include an analog multiplier (to multiply the voltage and current to calculate power), precision rectifier, absolute voltage generator and finally the NTC thermistor itself.

In the process of designing these blocks, we will explain how to use voltage-controlled voltage sources, current-controlled voltage sources, constant current sources/sinks, voltage-controlled current sources/sinks, current mirrors (built using current-controlled current sources/sinks) and provide some other handy hints for building SPICE models such as the best way to buffer and invert signals, and apply gain or attenuation.

For now, feel free to experiment with the models and circuits we've covered in this instalment. **SC**



```
.param Vcoil 12V  .param imax 2A  .param Ldelta 0.5
.model SWa SW(Ron=0.01 Roff=10Gig Vt=Vcoil/2 Vh=Vcoil/4)
.model SWb SW(Ron=10Gig Roff=0.01 Vt=Vcoil/2 Vh=Vcoil/4)      .param L1 Vcoil*Vcoil/300
.model SWc SW(Ron=0.01 Roff=10Gig Vt=0.6 Vh=0.25 ilimit=imax)   .param L2 1/(1/(L1*sqrt(Ldelta*Ldelta))-1/L1)   .param E1v 0.1*Ldeltasign
.model SWd SW(Ron=10Gig Roff=0.01 Vt=0.4 Vh=0.25 ilimit=imax)   .param Ldeltasign Ldelta/sqrt(Ldelta*Ldelta)   .param V1v 2.25 - Ldeltasign * 0.22
```

Fig.15: a modified version of our relay model which varies the coil inductance (by L_{delta}) as it switches on and off. Depending on the type of relay being simulated, coil inductance can increase or decrease when the coil is energised.