**SPICE User Variables**

**and If Command**

6-14-16

This document describes SPICE user variables and the SPICE If command.

***SPICE User Variables***

It is now possible to define user variables within SPICE at all HFIR Instruments that run SPICE. Once a user variable is defined, its value can be used in other SPICE commands in much the same way that the position of a motor can be used in a SPICE command.

Consider the following contrived macro:

# SPICE User Variables Example

# Scan over a spiral in s1 and stl.

# (10 revolutions, 4 mm radius, start at 30 deg)

loop i,0,100

let spiral\_fraction=%i/100

let spiral\_r=4\*(1-@(spiral\_fraction))

let spiral\_theta=30.0+360\*10\*@(spiral\_fraction)

drive stl @(spiral\_r)

drive s1 @(spiral\_theta)

scan n 0 preset time 2

endloop

Although the macro is rather silly, it does illustrate the use of SPICE user variables. Variables are defined using the ‘let’ command, as in:

let spiral\_fraction=%i/100

let spiral\_r=4\*(1-@(spiral\_fraction))

let spiral\_theta=30.0+360\*10\*@(spiral\_fraction)

Note that SPICE user variables can set according to formulas (as in 4\*(1-@(spiral\_fraction)). These formulas are Python expressions, so they are quite versatile. Also note that the value of a SPICE user variable can be substituted using the ‘@(…)’ syntax, much like substituting the position of a SPICE motor. The values can be substituted in other SPICE commands as well:

drive stl @(spiral\_r)

drive s1 @(spiral\_theta)

SPICE user variables can be deleted using the delvar command, as in:

delvar spiral\_r

This command deletes the SPICE user variable ‘spiral\_r’. The variable can no longer be used in SPICE unless it is redefined.

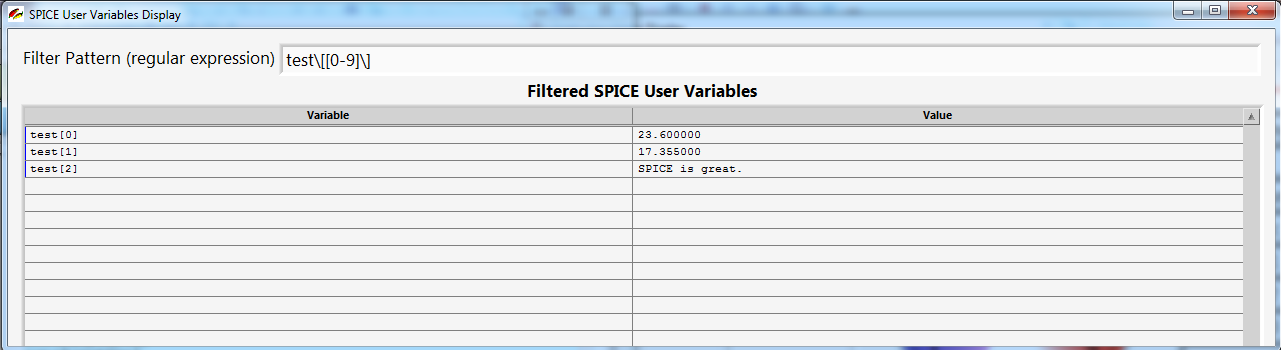
The ‘showvars’ command will display a window showing all defined SPICE user variables. So entering command

showvars

on the SPICE command line brings up this:



Note that at the top of the showvars window you can define a ‘Filter Pattern’. This is a regular expression string that can be used to filter the display to show only the variables you are interested in. For example, if we only want to look at the ‘test[]’ variables with single digit indexes inside the square braces we can use the filter pattern ‘test\[[0-9]\]’:



In addition, there is another way to define a SPICE user variable using the ‘let\_persisted’ command, for example:

let\_persisted vanadium\_2theta\_pos=22.3

As we might expect, this command sets the SPICE user variable vanadium\_2theta\_pos to 22.3 degrees, but in addition it sets the variable in a persistent way, so that SPICE will remember the value even if SPICE is restarted after a crash or shutdown.

It must be admitted that the spiral example macro presented above is not very useful, and its functionality can be replicated without SPICE user variables, although a little less conveniently. However, there are many cases where SPICE user variables are quite useful. The SANS SPICE Table Scans system uses them extensively.

Let’s look at a somewhat more useful example for sample changer automation. We will use the following four SPICE macros:

**run\_samp\_chg\_banjo.macro**

# Initialize banjo sample changer variables.

call init\_samp\_chg\_banjo

# Scan over the entire sample changer.

call scan\_samp\_chg\_banjo

**init\_samp\_chg\_banjo.macro**

let samp\_chg[banjo].offset=1.7

let samp\_chg[banjo].increment=20.0

let samp\_chg[banjo].positions=6

call init\_samp\_chg\_banjo\_pos

**init\_samp\_chg\_banjo\_pos.macro**

let samp\_chg[banjo][0]=@(samp\_chg[banjo].offset)

loop i,1,@(samp\_chg[banjo].positions)-1

let samp\_chg[banjo][%i]=@(samp\_chg[banjo][%i-1])+@(samp\_chg[banjo].increment)

endloop

**scan\_samp\_chg\_banjo.macro**

loop i,0,@(samp\_chg[banjo].positions)-1

drive vm1c @(samp\_chg[banjo][%i])

scan n 0 preset time 2

endloop

Notice the ‘call’ commands in the first two macros. The ‘call’ command is just like the SPICE ‘run’ command (to run a macro), except the loops in the macro being called are not unrolled immediately. This allows us to use values of SPICE user variables within ‘loop’ statements.

Now consider what happens when we execute the following SPICE command:

run\_samp\_chg\_banjo

First, the sample changer is initialized for the ‘banjo’ changer, which, in our example, has 6 positions separated by 20 mm. After initialization is complete, the sample in each position is scanned, assuming the sample changer axis is ‘vm1c’.

***SPICE If Command***

SPICE now has an ‘if’ command that allows conditional execution.

Consider the following macro:

# Email notification example of ‘if’ statement.

loop temp=100,300,50

set\_tempa %temp

waitpost vti %temp 5 600 1800

if "abs(@(vti)-%temp)>5" email "gregoryrd@ornl.gov" "VTI temperature out of range"

scan n 0 preset time 600

endloop

This macro scans at temperatures from 100 to 300 in steps of 50 degrees. At each temperature, if the temperature is not within range, an email notification is sent. Note that the if statement can contain Python expressions (as in abs(@(vti)-%temp)>5). Any SPICE command can be executed conditionally using an if statement, including calling another SPICE macro.

In addition, the if statement also supports an ‘else’ clause so that an alternative action can be taken, for example:

if "abs(@(vti)-10)>5" email "abc@d.com" "bad temp" else email "abc@d.com" "good temp”

***Conclusion***

I hope Instrument Scientists and other users will be able to make use of SPICE user variables and if commands in creative and productive ways. Of course, using the existing Python scripting in SPICE is much more versatile, but there is something to be said for being able to quickly automate experiments using macros and the SPICE command line.