Holy City Audio Forum

Former Home of SpinCAD Designer
Skip to content

Search... Search
Advanced search
[Moderator Control Panel]

Reverse engineering my own code

Post a reply

Search this topic... Search

4 posts • Page **1** of **1**

- Edit post (./posting.php?mode=edit&f=42&p=2734)
- Delete post (./posting.php?mode=delete&f=42&p=2734)
- Report this post (./report.php?f=42&p=2734)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2734&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2734)

Reverse engineering my own code (#p2734)

by **Digital Larry** » Sun May 29, 2016 10:09 am

This is probably the third or fourth time I've attempted this. It's a case where being away from developing the code and heavily involved in other things, I find I can no longer quickly put myself back into the mindset when I first invented these things. Yeah, and getting old doesn't help either.

What I'm trying to do is come up with a new SpinCAD Builder definition for a control panel slider that creates a Slider which shows the resulting single pole RDFX based filter rise time in milliseconds. For some reason I'm going somewhat in circles but getting closer. I am writing this down in excruciating detail to attempt to avoid having to do it again from the beginning if I want to create a new type of SliderLabel.

Let's start with the slider which sets the variable for delay length in samples given the delay time in milliseconds.

```
equ delayLength 32767
// variable - Name - low - high - initial - multiplier - precision - option
@sliderLabel delayLength 'Delay Time (ms): ' 32 32767 16384 1 0 lengthToTime
```

OK this defines a JSlider which sets:
- a variable called delayLength
- the corresponding label, "Delay Time (ms)"
- the value of the **variable** when slider is all the way to the left (32)
- the value of the **variable** when slider is all the way to the right (32767)
- the initial **variable** value when a new block/control panel is created (16384) - note, I'm not certain this is ever used
- the "multiplier", which can be used in different ways, depending on the option (1)
- the precision, or how many decimal points to display when displaying the variable (0)
- the option, which controls some things we'll discover a little later- here it is "lengthToTime"

The created Java code, which creates the JSlider, is:

```
delayLengthSlider = new JSlider(JSlider.HORIZONTAL, (int)(32 * 1),(int) (32767 * 1), (int)
(gCB.getdelayLength() * 1));
```

The Xtend code which created this Java is:

```
«IF e.option == "lengthToTime"»
   «e.ename»Slider = new JSlider(JSlider.HORIZONTAL, (int)(«e.minVal» * «e.multiplier»),(int) («e.maxVal» *
«e.multiplier»), (int) (gCB.get«e.ename»() * «e.multiplier»));
«ENDIF»
```

What we see here is that for "lengthToTime" anyway, the JSlider is initialized to the variable value **gCB.getdelayLength()** multiplied by the multiplier, which in this case is 1. For delayLength, the variable value translates directly to the JSlider value. The multiplier was conceived originally (I think) to allow a JSlider, which has to be set with integers, to create double values within the CAD blocks. So, for example, if I wanted to control a variable that went linearly from 0.01 to 1.00, I would set:

minVal = 0.01
maxVal = 1.0
multiplier = 100

A JSlider Change Listener is created. Here is the Java for delayLength:

```
if(ce.getSource() == delayLengthSlider) {
        gCB.setdelayLength((double) (delayLengthSlider.getValue()/1));
   updatedelayLengthLabel();
```

```
}
```

and here is the Xtend that created the central part of at:

```
gCB.set«e.ename»((double) («e.ename»Slider.getValue()/«e.multiplier»));
```

So, when I move the JSlider handle, the change listener catches the value, divides it by the multiplier, and sends that back to the CADBlock to be used directly somewhere in the code.

The second function we call in response to a JSlider change is **updatedelayLengthLabel()**. Let's see what it does.

Java:

```
private void updatedelayLengthLabel() {
   delayLengthLabel.setText("Delay Time (ms):  " + String.format("%4.0f", (1000 *
gCB.getdelayLength())/ElmProgram.getSamplerate()));
}
```

Xtend:

```
«IF e.option == "lengthToTime"»
   «e.ename»Label.setText("«e.controlName» " + String.format("%4.«e.precision»f", (1000 * gCB.get«e.ename»
())/ElmProgram.getSamplerate()));
«ENDIF»
```

So, the label updater gets the value from the CADBlock (which was set in the previous statement), multiplies it by 1000, then divides this by the sample rate variable. This gives the delay time in milliseconds. The conversion from delay samples to milliseconds does NOT use the multiplier. The factor of 1000 is right in the specific code for the **lengthToTime** option.

Top

- Edit post (./posting.php?mode=edit&f=42&p=2735)
- Delete post (./posting.php?mode=delete&f=42&p=2735)
- Report this post (./report.php?f=42&p=2735)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2735&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2735)

Re: Reverse engineering my own code (#p2735)

by **Digital Larry** » Sun May 29, 2016 11:57 am

I got a little ahead of myself with the **lengthToTime** option. I should have started with the no-option option, which is something like this, from the aliaser block:

```
equ ripLow 0.002
// variable - Name - low - high - initial - multiplier - precision - option
@sliderLabel ripLow 'Rip Low' 0.001 0.015 0.01 1000.0 3
```

This defines a JSlider which sets:
- a variable called ripLow
- the corresponding label, "Rip Low"
- the value of the variable when all the way to the left (0.001)
- the value of the variable when all the way to the right (0.015)
- the initial variable value when a new block/control panel is created (0.010)
- the "multiplier", which here scales between the desired resolution of the variable and the slider (1000.0)
- the precision, or how many decimal points to display when displaying the variable (3)
- the option, which controls some things we'll discover a little later and is optional. It isn't used in this example!

Since this is the no-option option, let's look at the code:

Java:

```
ripLowSlider = new JSlider(JSlider.HORIZONTAL, (int)(0.001 * 1000.0),(int) (0.015 * 1000.0), (int)
(gCB.getripLow() * 1000.0));
```

Xtend:

```
«e.ename»Slider = new JSlider(JSlider.HORIZONTAL, (int)(«e.minVal» * «e.multiplier»),(int) («e.maxVal» *
«e.multiplier»), (int) (gCB.get«e.ename»() * «e.multiplier»));
```

It's starting to look to me like the "initial" setting is never used, but maybe it is... will try to confirm later.

The Slider Change Listener:

Java:
```
        if(ce.getSource() == ripLowSlider) {
        gCB.setripLow((double) (ripLowSlider.getValue()/1000.0));
           updateripLowLabel();
        }
```

Xtend:

```
gCB.set«e.ename»((double) («e.ename»Slider.getValue()/«e.multiplier»));
```

The multiplier is a factor that keeps the JSlider value an integer. If the smallest value you want to display or resolve to is 0.01, then the multiplier is 100. If the smallest value were to be 0.001, then you should use a multiplier of 1000. This is great when the value you want to control is linear. I want 0 to 100 on the slider to represent a fixed value from 0.0 to 1.00 within the Spin ASM code. Great. Simple. But not everything is so simple. Next, I'll look at some other examples.

Top
- Edit post (./posting.php?mode=edit&f=42&p=2736)
- Delete post (./posting.php?mode=delete&f=42&p=2736)
- Report this post (./report.php?f=42&p=2736)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2736&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2736)

Re: Reverse engineering my own code (#p2736)

by **Digital Larry** » Sun May 29, 2016 2:34 pm

Next up is the log frequency to filter coefficient slider used with 1-pole RDFX filters. This represents equal frequency ratios with equal span of the slider. For example, you could fit 3 octaves on the slider and each one would be represented by 1/3 of the slider travel. I can see now where it started to get confusing, because in this case, the values I used in the sliderLabel declaration ar the frequency values, not the coefficients used in the program. So it is inconsistent with the previous ways of declaring a sliderLabel. At the same time, it makes sense because after all, I want to think of filters in terms of frequency.

```
equ freq 0.15
// variable - Name - low - high - initial - multiplier - precision - option
@sliderLabel freq 'Frequency (Hz)' 80 5000 100.0 100.0 1 LOGFREQ
```

The corresponding slider constructor:

Java:
```
freqSlider = SpinCADBlock.LogFilterSlider(80,5000,gCB.getfreq());
```

Xtend:
```
        «IF e.option == "LOGFREQ"»
           «e.ename»Slider = SpinCADBlock.LogFilterSlider(«e.minVal»,«e.maxVal»,gCB.get«e.ename»());
        «ENDIF»
```

Let's see what a LogFilterSlider looks like:
```
   public static JSlider LogFilterSlider(double fLow, double fHigh, double initVal) {
      int leftLimit = logvalToSlider(fLow, 100.0);
      int rightLimit = logvalToSlider(fHigh, 100.0);
      int initial = logvalToSlider(filtToFreq(initVal), 100.0);
      return new JSlider(JSlider.HORIZONTAL, leftLimit, rightLimit, initial);
   }
```

and what is logvalToSlider (and sliderToLogval):

```
   public static int logvalToSlider(double value, double multiplier) {
      return (int) (multiplier * Math.log10(value));
   }

   public static double sliderToLogval(int pos, double multiplier) {
      return Math.pow(10.0, pos/multiplier);
```

```
   }
```

OK, here's another place where I set up my own downfall. I used the variable called "multiplier", but it is not the multiplier defined earlier. It's actually "points per decade". So I can change that in the code to make it clearer.

Here's the Change Listener for the freq Slider:

Java:

```
if(ce.getSource() == freqSlider) {
        gCB.setfreq((double) SpinCADBlock.freqToFilt(SpinCADBlock.sliderToLogval((int)(freqSlider.getValue()),
100.0)));
   updatefreqLabel();
}
```

Xtend:

```
«IF e.option == "LOGFREQ"»
   gCB.set«e.ename»((double) SpinCADBlock.freqToFilt(SpinCADBlock.sliderToLogval((int)
(«e.ename»Slider.getValue()), «e.multiplier»)));
«ELSE»
...
```

and updatefreqLabel():

Java:

```
      private void updatefreqLabel() {
      freqLabel.setText("Frequency (Hz) " + String.format("%4.1f", SpinCADBlock.filtToFreq(gCB.getfreq())) + "
Hz");
      }
```

Xtend:

```
«IF e.option == "LOGFREQ"»
   «e.ename»Label.setText("«e.controlName» " + String.format("%4.«e.precision»f",
SpinCADBlock.filtToFreq(gCB.get«e.ename»())) + " Hz");
«ENDIF»
```

Reference function in SpinCADBlock.java:

```
   // frequency in Hz gets converted to filter coefficient for first order filter
   public static double freqToFilt(double freq) {
      double omega = 2 * Math.PI * freq/ElmProgram.getSamplerate();
      return 1 - Math.pow(Math.E, -omega);
   }

   // filter coefficient gets converted to frequency in Hz for first order filter
   public static double filtToFreq(double filt) {
      return (-(Math.log(1 - filt)) * ElmProgram.getSamplerate()/(2 * Math.PI));
   }
```

Top

- Edit post (./posting.php?mode=edit&f=42&p=2737)
- Delete post (./posting.php?mode=delete&f=42&p=2737)
- Report this post (./report.php?f=42&p=2737)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2737&sid=d6d1dcc450159655070460679 1e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2737)

Re: Reverse engineering my own code (#p2737)

by **Digital Larry** » Mon May 30, 2016 7:03 am

In another possible twist, I have not yet shown the "setters and getters" where values are passed between the CAD Block (where the code generation is done) and the control panel (where we adjust said values). Almost all of them simply pass the literal value (filter coefficient, gain value, threshold, etc.) back and forth. Of course, there's an exception somewhere. As I'm looking over all of this I realize that all these variations occurred simply because I kept extending what I wanted to do beyond the simple case I started with. I'm hoping that by looking at them all at once, like a happy little family, I might be able to simplify some of it or make them more consistent in general. It hasn't exactly leapt out and grabbed me by the throat though.

Top

Display posts from previous: All posts Sort by Post time Ascending Go

Post a reply

4 posts • Page **1** of **1**

Return to Developer's Corner

Jump to: Developer's Corner Go

Quick-mod tools: Lock topic Go

Who is online

Users browsing this forum: Digital Larry and 0 guests

Administration Control Panel