Holy City Audio Forum

Former Home of SpinCAD Designer

The servo delay challenge, or teach a man to fish...

The servo delay challenge, or teach a man to fish... (#p2223)

by **Digital Larry** » Fri Mar 27, 2015 8:49 am

Some people (you know who you are) have been requesting a REAL servo delay. The delay block currently called "Servo Delay" in fact is NOT a servo delay. I would probably call it a "RMPA delay" in that it uses the RMPA instruction to read the value from a point in the delay buffer. That point can be fixed or it can be adjusted by a control, either a pot, or LFO, or any signal going from 0 to 1. That's all dandy but it makes a lousy chorus or flanger block because it can only resolve to actual samples in RAM.

The CHO RDA instruction of the FV-1 makes this easier - it can interpolate between actual samples to give a smoother sound when sweeping the delay time. The chorus and flanger blocks that recently came out use CHO RDA with the built-in Sin/Cos LFOs. That restricts you to using Sin waves for LFOs, so you can't use a triangle or envelope signal etc. for delay time modulation.

The Spin knowledge base includes some instructions on how to create a real Servo Delay using CHO RDA with the Ramp LFO and then modulating the Ramp width on the fly to make it home in onto the target location in delay RAM, smoothly sweeping through on the way.

I'd like to put out a challenge to anyone who wants to help create this block. Your submission should be in the form of an annotated Spin ASM file.

By annotated, this is what I mean.

Suppose there's some value that you'd like to control with a control panel slider. The easiest example would be the gain value in an RDAX instruction such as the input gain.

```
equ inputGain 0.5
@sliderLabel inputGain Input_Gain  -24 0 -6 1.0 1 DBLEVEL
```

This defines a double Java variable named inputGain, and creates a Control Panel slider with a label "Input_Gain", that goes between -24 and 0 dB, default setting (when block is created) of -6 dB. The 1.0 that follows is the factor that the converted value gets divided by before actually being used, and the 1 means show 1 digit to the right of the decimal point. Finally, DBLEVEL means to show the value in dB (20 * log10 (inputGain)).

I have to admit when looking at this, that the implementation of various features has been incredibly inconsistent! 🤨

Further down in the code, I just use:

```
RDAX input1, inputGain
```

and voila, the value calculated from the slider gets put into the code when the code is generated for a patch.

If you wanted a parameter, let's just use inputGain again, that went from 0.001 to 1.0, default value 0.5, and showing 2 decimal places, then it would be something like this:

```
equ inputGain 0.5
@sliderLabel inputGain Input_Gain  0.001 1.0 0.5 1.0 2
```

There are other options but you get the point.

So, with this in mind, can you create a servo delay with:
1) Adjustable input gain
2) Adjustable fixed tap output location (could be % of the whole buffer)
3) Adjustable ramp servo gain (controls how fast the servo reacts to the delay time control input)

This is a small example of the SpinCAD Builder language I wrote to simplify creating new blocks. There are other directives beyond @sliderLabel.

Maybe you can figure some of them out. Here's an example of the SpinCAD Builder code for the existing flanger block. Let me know if there are any questions!

```
@name Flanger
@audioInput input Input
@audioOutput output1 Output
@audioOutput center Center_Tap
@controlInput rateIn LFO_Rate
@controlInput widthIn LFO_Width

equ rateMax 511
equ number6554000 6554000.0
equ output1 reg0

// total allocated memory buffer for this delay
// create a Control Panel with a Slider Label
equ delayLength 64
@sliderLabel delayLength Delay_Time 16 512 64 1 2 lengthToTime

equ tap1Center 0.5
; @sliderLabel tap1Center Tap_1_Center 0.25 0.75 0.5 1000.0 2

equ rate 20
@sliderLabel rate LFO_Rate 0.0 511.0 20.0 100.0 2 SINLFOFREQ

// max width (in samples) of LFO = delayLength/2.
// this allows the center to be placed from 25% to 75%
// From AN-001:
// The equation to calculate the coefficient for the amplitude for a given delay length for use in a WLDS
instruction is:
// Ka = (N * 32767)/16385

equ width 30   // double value of width %
@sliderLabel width LFO_Width 5.0 100.0 30.0 100.0 1

// ComboBox needs to show LFO 0 or LFO 1
// in block code, need to select between SIN0 or SIN1 parameter
// setter/getter passes index of combobox item, for now
// let's see if we can make it work.

equ lfoSel 0
@comboBox lfoSel 'LFO 0' 'LFO 1'

@isPinConnected Input
;
; Memory declarations
equ delayOffset -1
@getBaseAddress
mem delayl delayLength ; delay should actually be delay * 2 as panel setting is sweep center

@multiplyDouble x1 delayLength width
@divideDouble x3 x1 number6554000

@isEqualTo lfoSel 0
skp run,START
wlds SIN0, 50, 64
START:
@else
// starting speed and width need to be figured from control panel
skp run,START1
wlds SIN1, 50, 64
START1:
@endif

// want to try to get to this:
// From AN-001:
// The equation to calculate the coefficient for the amplitude for a given delay length for use in a WLDS
```

```
instruction is:
// Ka = (N * 32767)/16385
// Where:
// N: Delay length in samples, 16385 max( or length of delay buffer).
// back to SpinCAD ============================
// Then there is the width factor, 5% to 100%
// Valid values are in the range of 0 to 32767 for Ka.
// Ka = (scale * ( N * 32767))/16385
// Finally if needed it is all scaled by the control input
// double x1 = delayLength * 32767
// double x2 = x1 * scale
// double x3 = x2 / 16385

@isPinConnected LFO_Width

rdax   widthIn, x3
@isEqualTo lfoSel 0
wrax SIN0_RANGE, 0
@else
wrax SIN1_RANGE, 0
@endif
@endif

// Here's the expression for scaling the LFO rate by the control input.
// we read the control and scale it by the slider setting/511 which is max.
// sfxb.readRegister(speedIn, lfoRate/511.0);

@isPinConnected LFO_Rate
@divideDouble temp1 rate rateMax
rdax   rateIn, temp1
@isEqualTo lfoSel 0
wrax SIN0_RATE, 0
@else
wrax SIN1_RATE, 0
@endif
@endif

ldax input
; Write it to delay and clear ACC
wra delayl,0

@isEqualTo lfoSel 0
@readChorusTap 0 0 tap1Center delayLength delayOffset
@else
@readChorusTap 1 0 tap1Center delayLength delayOffset
@endif

; @readChorusTap 0 1 tap2Center delayLength delayOffset

; Interpolated sample in ACC, write it to DACL and clear ACC
wrax output1,0
; That's it!
@isPinConnected Center_Tap
equ center reg1
rda delayl^, 1
wrax center, 0
@setOutputPin Center_Tap center
@endif

@setOutputPin Output output1
@endif
```

Top

- Edit post (./posting.php?mode=edit&f=42&p=2224)
- Delete post (./posting.php?mode=delete&f=42&p=2224)
- Report this post (./report.php?f=42&p=2224)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2224&sid=d6d1dcc4501596550704606791e62b4c)

- Reply with quote (./posting.php?mode=quote&f=42&p=2224)

Re: The servo delay challenge, or teach a man to fish... (#p2224)

by **Digital Larry** » Fri Mar 27, 2015 10:27 am

@audioInput defines an integer value (that winds up being a register reference) and what the name of that pin should be. It is an audio input and shows up on the left side of the block.

Audio and control inputs and outputs are defined by similar statements.

@isPinConnected lets you generate code differently (or maybe not at all) depending on whether a pin is connected to something else in the model. Normally I substitute a default value, but you can do anything.

@isPinConnected

has to be bracketed by

@endif

you can put an

@else

in the middle.

For most blocks, I don't generate anything at all unless the input pin is connected.

Top
- Edit post (./posting.php?mode=edit&f=42&p=2225)
- Delete post (./posting.php?mode=delete&f=42&p=2225)
- Report this post (./report.php?f=42&p=2225)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2225&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2225)

Re: The servo delay challenge, or teach a man to fish... (#p2225)

by **Digital Larry** » Sat Mar 28, 2015 6:55 am

There's an alternate approach we can take here.

The "Mod Delay" block is in fact an attempt at the real servo delay from the knowledge base. Here's the code that is generated:

```
;------ Mod Delay
SKP RUN ,1
WLDR 0, 0, 4096
RDAX ADCL,1.0000000000
WRA 0,0.0
CHO RDAL,2
RDAX REG1,-0.2500000000
WRAX RMP0_RATE,0.0000000000
CHO RDA,2,REG | COMPC,0
CHO RDA,2,0,1
WRAX REG2,0.0000000000
```

And here's the Java code that created this (it was done before I came up with the SpinCAD Builder language). What's wrong with this? There's definitely something wrong with it.

```
   private void modDelay(SpinFXBlock sfxb, int chorusLength) {
      int input = -1;
      SpinCADPin p = this.getPin("Audio Input 1").getPinConnection();
      if (p != null) {
         input = p.getRegister();
         int chorout = sfxb.allocateReg();

         int Control1 = -1;

         p = this.getPin("Modulation In").getPinConnection();
         if (p != null) {
            Control1 = p.getRegister();
         }

//         ;modulated delay for flanger etc
//         equ control pot0
```

```
//          mem moddel 8192
          sfxb.FXallocDelayMem("moddel", delayLength);
          sfxb.comment("Mod Delay");

//          ;set up LFOs, only at start up
//          skp run, START
          sfxb.skip(RUN, 1);
//          wldr 0,0,4096      ;ramp0 used for modulation operations
          sfxb.loadRampLFO(0, 0, 4096);
//          START:

//          rdax adcl,1
          sfxb.readRegister(input, 1.0);
//          wra moddel,0       ;write to head of delay
          sfxb.FXwriteDelay("moddel", 0, 0);
//          cho rdal,rmp0      ;servo ramp0 to correct position using value in control
          sfxb.chorusReadValue(RMP0);
//          rdax control,-0.25    ;scaled so that 0 - 1 sweeps full range
          sfxb.readRegister(Control1,  -0.25);
//          wrax rmp0_rate,0
          sfxb.writeRegister(RMP0_RATE, 0);
//          cho rda,rmp0,reg|compc,moddel   ;read from delay
          sfxb.FXchorusReadDelay(RMP0, REG | COMPC, "moddel", 0);
//          cho rda,rmp0,0,moddel+1
          sfxb.FXchorusReadDelay(RMP0, 0, "moddel+", 1);
//          wrax dacl,0
          sfxb.writeRegister(chorout, 0);
          this.getPin("Audio Output 1").setRegister(chorout);
          System.out.println("Mod Delay code gen!");
      }
   }
```

Top
- Edit post (./posting.php?mode=edit&f=42&p=2226)
- Delete post (./posting.php?mode=delete&f=42&p=2226)
- Report this post (./report.php?f=42&p=2226)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2226&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2226)

Re: The servo delay challenge, or teach a man to fish... (#p2226)

by **Digital Larry** » Sat Mar 28, 2015 7:03 am

http://www.spinsemi.com/knowledge_base/ ... _technique (http://www.spinsemi.com/knowledge_base/coding_examples.html#Servo_technique)

> Although a memory pointer can be loaded (ADDR_PTR), and used to address memory, if it is to be clear of audible artifacts, an interpolation must be performed at each sample, as the delay is varied. This normally would mean tearing the address pointer value into an integer and a fractional part, and using the fractional value to interpolate between adjacent samples. This can be automatically performed with the LFOs however, if they can be controlled to 'point' to the proper position in memory. Often, the delay-varying signal is a POT input or perhaps a sine LFO, or even a generated triangle wave.
>
> Let's say the value we want to access is in a register we'll call MPOS, and we'll use the RMP0 LFO to 'servo' to that location. We'll assume the accumulator is cleared going into the routine:
>
> skp run,1 ;only establish the LFO on the first sample pass
> wldr rmp0,0,4096 ;set rmp0 to its widest range
> cho rdal,rmp0 ;load in the current RMP0 pointer
> rdax mpos,-1 ;subtract the desired position
> wrax rmp0_rate,0 ;write the position error to the ramp rate register

For some reason I'm scaling the input by -0.25 instead of -1.0 as in the example. Well, I'll work on this a bit today. Challenge is still open though!

Top
- Edit post (./posting.php?mode=edit&f=42&p=2231)
- Delete post (./posting.php?mode=delete&f=42&p=2231)
- Report this post (./report.php?f=42&p=2231)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2231&sid=d6d1dcc4501596550704606791e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2231)

Re: The servo delay challenge, or teach a man to fish... (#p2231)

by **Digital Larry** » Sun Mar 29, 2015 5:58 am

In case you're wondering, this is partially a stalling tactic. Yesterday I got the ga_demo_flanger patch from the Spin web site and stripped out the reverb to make a flanger block using the servo technique and a built-in triangle LFO. Works fine on the FV-1 and sounds like you are on the Tilt-a-Whirl about ready to let go of your hot dogs and popcorn 🤒😷😵 in the SpinCAD simulator. You've all heard my whining about the SpinCAD simulator and how it's really not my fault, blah blah blah. I've dug into this deeply before and learned a lot and forgot most of it.

My conclusion as of yesterday is that the Ramp generation itself is OK or pretty close to OK. The Sample/Hold block uses it (via CHO RDAL) and it seems to work as expected there. What's really off is the CHO RDA instruction when using a Ramp LFO. So I'm going back to basics with my pencil and notepad imagining how such a thing should work.

I'm going to need to branch the code so that if other things come up, they won't be interrupted by me ripping into this.

Top

- Edit post (./posting.php?mode=edit&f=42&p=2233)
- Delete post (./posting.php?mode=delete&f=42&p=2233)
- Report this post (./report.php?f=42&p=2233)
- Information (./mcp.php?i=main&mode=post_details&f=42&p=2233&sid=d6d1dcc450159655070460679 1e62b4c)
- Reply with quote (./posting.php?mode=quote&f=42&p=2233)

Re: The servo delay challenge, or teach a man to fish... (#p2233)

by **Digital Larry** » Mon Mar 30, 2015 6:06 am

I may be on to something here. I brought the servo signal out so I could visualize it, and whenever I increased the pot value controlling the desired servo point, the delay pointer would go down, wrap around up top, and then settle down to the target value (which was above where I started). If I decreased the pot value, then it would just track down to that target. Turns out that the sign of the Ramp LFO rate was getting calculated but never used for anything! Now either that was there all along or I messed it up because I've been tromping all over in that RampLFO.java trying to figure this out. Now I am getting nice servo-ing whether up or down in the simulator.

Another note. I had to back away from the code and figure in my mind, how is it really working? In the Sin LFO, there are 8 fractional bits used for interpolation. However, the Ramp LFO has 16384 = 14 fractional bits. I must say this is just an assumption because it's not documented anywhere publicly. So, I had to figure out how to use the top 8 of the 14 fractional bits for interpolation.

The other thing I notice is that servoing puts a bit of slewrate limiting (or something like that) in the tracking of the LFO signal, so if you're using a triangle wave input for example and increase the LFO rate, the servo'ed amplitude goes down. Lots of times that's what you want, and maybe sometimes it isn't.

Now, for some reason the ga_flanger_demo is not simulating correctly. But this morning started off with a pretty big push in the right direction. I hate to get my hopes up, but can pitch shift simulation be far behind? 😵

Top

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

Post a reply

6 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