Tap Tempo to LFO Frequency

The FXCore tap tempo function is very easy to use in delay type effects as the value returned is the time between taps in samples. However it can be useful to tap in a value that represents a desired LFO frequency for effects like a phaser or flanger.

This app note presents a simple bit of code that will do just that, it takes the tap tempo value, manipulates it and updates the LFO frequency coefficient so that the tap time represents one cycle of a sine wave.

The most important part of this code is that division in the linear domain is subtraction in the log domain:

$$a/b = EXP2(LOG2(a) - LOG2(b))$$

Note that “a” must always be smaller than “b” in the FXCore because when we do the EXP2 function it must be able to return a value greater than 0 and less than 1.0 as such you may need to scale values then later multiply them back up for the final result.

The equation to go from a tap value to an LFO coefficient is:

LFO coefficient = 13,493,037,698/tap tempo value

This equation was derived by creating a table of tap values and the LFO coefficients for these tap values, plotting them and doing a curve fit to the values.

Now we can see there are a few issues to deal with:

1. 13,493,037,698 is larger than can be expressed in an S31 format number
2. The numerator must be smaller than the denominator

First we make a few conditions:

1. Sample rate 48KHz
2. Fastest a person could be expected to tap in a value is 4Hz
3. Maximum time of 10 seconds (0.1Hz) at 48KHz

From this we can determine that the tap tempo would return a value of 12,000 for 4Hz at 48KHz (simply 48KHz/4Hz) so the numerator must be less than 12,000.

If we divide the numerator by 2^{22} we get:

$$13{,}493{,}037{,}698/2^{22} = 3{,}217 \text{ or } 0x0C91$$

This number is clearly less than 12,000 and fits within a S31 format number, so the equation is now:

$$\text{LFO coefficient} = (3{,}217/\text{tap tempo value}) * 2^{22}$$

This equation will work for any sample rate where the minimum tap count is greater than 3,217


```
// an-6.fxc
// Example taking tap tempo value and calculating the coefficient
// for an LFO
//
// User0 will flash at the LFO rate
// User1 will turn on after the first tap and turn off after the second tap
or when the
// tap_limit value times out. This allows a user to see if it is waiting for
a second
// tap or is ready for a new tap time.

.equ div 0x0c91                  ; 3217 in hex
.equ multt 0x0040                ; 2^22 is 0x00400000 so only need upper 16-
bits as lower are all 0
.equ tap_limit 480000            ; max tap time of 10 seconds at 48K
.rn bright r3
.rn temp r4
.rn timer r5

.sreg maxtempo tap_limit         ; set the maxtempo SFR

// Read in tap tempo and convert to LOG domain
cpy_cs r0, taptempo
log2 r0
cpy_cc r0, acc32

// This is a constant we divide by the tap tempo also convert to LOG domain
wrdld r1, div
sr r1, 16                        ; shift right so it aligns with the tap
tempo value
log2 acc32
cpy_cc r1, acc32

// Subtraction in LOG domain is same as division in linear domain
subs r1, r0

// Convert back to linear
exp2 acc32

// Multiply to scale back up
wrdld r0, multt                  ; loads the value into upper 16-bits, lower
16 set to 0
multrr acc32, r0

// And write to LFO0 frequency control register
cpy_sc lfo0_f, acc32


; flash led
cpy_cs     acc32, samplecnt      ; Get the sample counter
andi       acc32, 0xFF           ; Mask b[7:0]
jnz        acc32, doPWM          ;

; Reload new PWM value from LFO0_s into "bright"
```

```
cpy_cs     temp, lfo0_s            ; read in sin wave ranges -1.0 to +1.0
(well, almost)
sra        temp, 1                 ; /2 to +/- 1/2
addsi      acc32, 0.5              ; ranges 0 to 1
sra        acc32, 23               ; shift the PWM value in place
cpy_cc     bright, acc32           ; save it

doPWM:
; Performing the decrement prior to driving the LED makes sure
; that the LED can go completly off.
addi       bright, -1              ; subtract 1 from on count
cpy_cc     bright, acc32           ; Save updated "bright"
xor        acc32, acc32            ; Clear acc32 for the LED off case
jneg       bright, doLED           ;
ori        acc32, 1                ; Set acc32[0] for the LED on case

doLED:
set        user0|0, acc32          ; set the usr1 output per the acc32 LSB


; As we are dealing with a long time between taps we use the User1 LED to
indicate if we are
; waiting for the second tap or not

andi       flags, TAPPE            ; is this a button push event?
jz         acc32, no_push          ; if not then jump away
andi       flags, TB2NTB1          ; is it a tap 1 event?
jnz        acc32, tap2ev           ; no so jump to tap2 routine
// if here then it is a tap 1 event, load the timer
wrdld      timer, tap_limit.u
ori        timer, tap_limit.l

no_push:
jz         timer, nothing          ; if 0 then nothing to do
addi       timer, -1               ; subtract 1 from timer count
cpy_cc     timer, acc32
jz         timer, tap2ev           ; if we hit 0 then turn off USER1
ori        acc32, 1                ; still greater than 0 so keep User1 LED on
set        user1|0, acc32
jmp        nothing

tap2ev:
wrdld      timer, 0
set        user1|0, timer

nothing:
or         acc32, acc32            ; jumps must always target a valid
instruction
```

Experimental Noize Inc. reserves the right to make changes to, or to discontinue availability of, any product or service without notice.

Experimental Noize Inc. assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using any Experimental Noize Inc. product or service. To minimize the risks associated with customer products or applications, customers should provide adequate design and operating safeguards.

Experimental Noize Inc. make no warranty, expressed or implied, of the fitness of any product or service for any particular application.

In no even shall Experimental Noize Inc. be liable for any direct, indirect, consequential, punitive, special or incidental damages including, without limitation, damages for loss and profits, business interruption, or loss of information arising out of the use or inability to use any product or document, even if Experimental Noize Inc. has been advised of the possibility of such damage.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Experimental Noize Inc. products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications"). Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Experimental Noize Inc. products are not designed nor intended for use in military or aerospace applications or environments. Experimental Noize Inc. products are not designed nor intended for use in automotive applications.

Experimental Noize Inc.

Scottsdale, AZ USA

www.xnoize.com

sales@xnoize.com