DETAILED POL SYNTH PROGRAMMING GUIDE

Syd Kreitzman

-------------------------------------- rev history --

Rev Nov 20/2003 ... original spec

Rev Dec 1 /2003 ... added functions idle_freq idle_iq
added detailed procedure to calculate the frequency sweep values

Rev Feb 26/2004 ... added gate control logic into register 2C
and moved old 2C functions to 2D, old 2D function to 2E (see * lines)

Rev May 10/2004 ... added section on how to do the complex modulation; srk

Rev July 19/2004 ... corrected and updated complex modulation section; srk

Rev Sept 10/2004 ... added functionality for iq data recycling by introducing five new
Registers Ncmx, Nc1f, Nc3f, Nc5f, Ncfref into locations 30-31
Also introduced the modulation function parameters A and α to define
the modulation function properties (i.e. linewidth and ending amplitude); srk

Rev Sept 30/2004 ... corrected for the fact that the Niq data gets clocked in at 4*Ncic*25X10^(-9)
seconds (i.e. every 4*Ncic clock periods) ... thereby correcting calculations
for pulse lengths, bandwidths etc; srk

Rev Feb 28/2005 ... added control of output scale factorto give amplitude control to modulated
functions without having to reprogram the entire I/Q array. A new global
control function "scale" has been added. The scaled value is 2^-7*scale,
where scale is an eight bit value from 0 -> 255. The default value of scale
is (256/sqrt(2))=181.

Rev April 25/2005 ... Added Hermite modulation function.

-------------------------------------- end of rev history --

Local Control Register Level Functionality

Function	Value	Register	Access
1f-qm-on	0x	01	r/w
1f-qm-off	1x	01	r/w
1f-cic	2-ffx	02	r/w
1f-scale	0x-ffx	03	r/w
1f-n_iq	0x-7ffx	04-05	r/w
1f-iq_ptr		06-07	r only
1f-on	1,1b	2C bits 0,1	r/w
1f-off	0,0b	2C bits 0,1	r/w
1f-gate-t	1,0b	2C bits 0,1	r/w
1f-gate-f	0,1b	2C bits 0,1	r/w
Nc1f	1x-01x	31 bits 0-5	r/w 0x is not allowed
3f-qm-on	0x	09	r/w
3f-qm-off	1x	09	r/w
3f-cic	2-ffx	0A	r/w
3f-scale	0x-ffx	0B	r/w

```
3f-n_iq            0x-7ffx    0C-0D               r/w
3f-iq_ptr                     0E-0F               r only
3f-on              1,1b       2C bits 2,3         r/w
3f-off             0,0b       2C bits 2,3         r/w
3f-gate-t          1,0b       2C bits 2,3         r/w
3f-gate-f          0,1b       2C bits 2,3         r/w
Nc3f               1x-01x     32 bits 0-5         r/w 0x is not allowed


5f-qm-on           0x         11                  r/w
5f-qm-off          1x         11                  r/w
5f-cic             2-ffx      12                  r/w
5f-scale           0x-ffx     13                  r/w
5f-n_iq            0x-7ffx    14-15               r/w
5f-iq_ptr                     16-17               r only
5f-on              1,1b       2C bits 4,5         r/w
5f-off             0,0b       2C bits 4,5         r/w
5f-gate-t          1,0b       2C bits 4,5         r/w
5f-gate-f          0,1b       2C bits 4,5         r/w
Nc5f               1x-01x     33 bits 0-5         r/w 0x is not allowed

fref-qm-on         0x         19                  r/w
fref-qm-off        1x         19                  r/w
fref-cic           2-ffx      1A                  r/w
fref-scale         0x-ffx     1B                  r/w
fref-n_iq          0x-7ffx    1C-1D               r/w
fref-iq_ptr                   1E-1F               r only
fref-on            1,1b       2C bits 6,7         r/w
fref-off           0,0b       2C bits 6,7         r/w
fref-gate-t        1,0b       2C bits 6,7         r/w
fref-gate-f        0,1b       2C bits 6,7         r/w
Ncfref               1x-01x     34 bits 0-5         r/w 0x is not allowed
fref-freqx         0-ffffffffx 20-23               r/w write order: 20 first 23 last
 fref-freqx = (fref_Hz*2^32/(4x10^7))x=(fref_Hz*(107.3741824))x=(fref_Hz/finc)x
 finc = .009313226Hz
 fmax =  4*10^7 - finc

n_fsweep          0-3ff       24-25                r/w
fsweep_ptr                    26-27                r only

iq-end_idle         ?fx       28 bits 1-4only      r/w
iq-end_niq          ?0x       28 bits 1-4 only     r/w
fsweep-end_idle     1?x       28 bit 5 only        r/w
fsweep-end_nfsweep 0?x        28 bit 5 only        r/w

or reg_28x = (iq-end*0fx)+(sweep-end*10x),
  where iq-end is 1 (iq-end_idle) or 0 (iq-end_niq)
  and sweep-end is 1 (sweep-end_idle) or 0 (sweep-end_nfsweep).
reg_28x is calculated and written wheneverthe values of iq-end or sweep-end are changed.

RF_power_trip_thr 0-ffx      29                   r/w

fsweep_int_strobe 0                          w, r is always 0

fsweep_ptr_reset  0          2b              w, r is always 0

RF_power_trip_stat 0         2d              w cycle will reset, read is 1 or 0

VME_reset          0         2e    w cycle resets all vms and DDS registers to default

Ncmx               1x        30    max number of cycles available to cycle the iq pairs
```

Global Control Functions

qm-on   > 1f-qm-on,3f-qm-on,5f-qm-on,frfef-qm-on

qm-off > 1f-qm-off,3f-qm-off,5f-qm-off,frfef-qm-off
n_iq 0x-7ffx > 1f-n_iq,3f-n_iq,5f-n_iq,frfef-n_iq all set to same value, default value is 2048 = 8FFx
iq_ptr displays all 1r,3f,5f,fref-iq_ptr
cic_ir 2-ffx > 1f-cic, 3f-cic, 5f-cic, fref-cic all set to same value
idle_freq (specify and load the value of idle_freq into 8FFC-F)
idle_iq (specify and load the i and q modulation default values into 1/3/5/7FFC-F)
Nc 1x-01x > Nc1f, Nc3f, Nc5f, Ncfref all set to same value Nc
scale 0x-ffx (i.e. 8bit 0-255), change 1f-scale, 3f-scale, 5-scale, fref-scale to same scale value

Freq Sweep Loading

fsweep 0-ffffffffx	8000-8FFC	r/w n_fsweep locations to to loaded
	8FFD-8FFF	r/w + the idle frequence

It is probably best to divide the freq steps into multiples of finc, so determine the closest start freq, the closest value of the requested freq step in units of finc (call this delta_fincx) and the number of dealta_fincs required to get to withing 1/2 finc of the requested ending frequency. Then you can program the first location with the closest intitial start frequency, and a delta_fincx to that value successively for each 32bit frequency word. This means that the actual start, stop, and step frequencies are computed as well as the number (i.e. n_fsweep) steps to get there. These number will be slightly different than what the user requested ... but the frequency increments will be absolutely constant. The calculation of the frequency is the same as that indicated for fref_freq. See detailed instructions below.

IQ data	0000-1FFF	}	
	2000-3FFF	}	All sets of registers are loaded with same set of data.
	4000-5FFF	}	The 1/3/5/7FFE - 1/3/5/7FFF locations are always loaded
	6000-7FFF	}	with the idle I/Q data pair.

<u>Deatailed notes to for frequency sweep programing:</u>

i) Determine the actual_start_frequency, it will be the closest frequency to the requested start_freq
ii) Determine the actual_frequency_step, it will the frequency closest to the requested freq_step
iii) Determfine the actual_number_freq_steps to get to the first frequency beyond the requested stop_freq
iv) Compute the actual_stop_frequency

actual_start_frequencyx = (int(start_frequency/finc))x
actual_start_frequency_hz =(int(start_frequency/finc))*finc

actual_frequency_stepx = (int(freq_step)/finc))x
actual_frequency_step_hz = (int(frq_step)/finc))*finc

actual_number_freq_steps = int((stop _freq - start_freq)/actual_frequency_step_hz)
actual_stop_freq_hz = actual_start_freq_hz + actual_number_freq_steps*actual_freq_step

a) into location 8000 put the actual_start_frequencyx according to table 4 on page 6 of the manual (t4p6)
b) into location 8000x+(4*n)x put actual_start_frequencyx +nx*actual_frequency_stepx according to t4p6,
for all n's 1 to actual_number_freq_steps

b) may be accoplished in two ways. You can use the fact that the n=1'th frequency data in location 8000x+ (4*(n+1))x is the data in location 8000x+(4*n)x + actual_frequency_stepx

I suggest that a table of the all the actual_frequency_n be calculated, both in hz and in binary (i.e. hex) which then can be read into the memory according to order specified in t4p6. This give one a chance to look at the table for debugging purposes. Donald thinks that data should just be programmed in on the fly.

End of Frequency Sweep Programing Description.

IQ modulation programing:

Perscription for modulation: Csech (complex sech or ln-sech) and Hermite

0) Select the modulation function and input the parameters A & a.
   Csech: {A , a}={0.1 , 5}
   Hermite: {A , a}={0.39714 , 2.2}

1) Input the requested dnu (in Hz) and define dw=dnu*2*pi
   (dnu is the requested bandwidth (in Hz) that the modulation function will irradiate given a proper level of RF power)

2)Define dw_max = 2x10^(7) a / (A * 512 * 2) , dw_min= 2x10^(7) a / (A * 2048 * 63*32)
   d_nu_max = dw_max/(2*pi) , d_nu_min = dw_min/(2*pi) ,
   Csech: 976.563*10^3 rad/sec , 0.24420*10^3 rad/sec
   155.424*10^3 Hz , 0.03855*10^3 Hz
   Hermite: 108.195*10^3 rad/sec , 0.026844*10^3 rad/sec
   17.220*10^3 Hz , 0.004272*10^3 Hz

3) Check that d_nu_min <= d_nu <= d_nu_max, else return an error stating that the requested d_nu in not with the available limits ... say what these limits are.

4) Compute the preliminary total number of iq points Ntiqtemp
   Ntiqtemp = { a * 2x10^7 / (A * dw) } ; { ... } = nearest smaller integer,
   Csech: Ntiqtemp = {10^9/dw}
   Hermite: Ntiqtemp = {1.18*10^8/dw}
   and confirm that 1024<= Ntiqtemp < 129024*Ncmx=4128768.

5) Assign Nc and Ncic according to the following table:

Ntiqtemp	**Nc**	**Ncic**
1024 - 2047	**1**	**2**
2048 - 4095	**1**	**2**
4096 - 8191	**1**	**4**
8192 - 16383	**1**	**8**
16384 - 32767	**1**	**16**
32768 - 65535	**1**	**32**
65536 - 129023	**1**	**63**
129024 - 258047	**2**	**63**
258048 - 516095	**4**	**63**
516096 - 1032191	**8**	**63**
1032192 - 2064383	**16**	**63**
2064384 - 4128767	**32**	**63**

Confirm that Nc <= Ncmx.

6) Compute Niq and Ntiq
   Niq=[Ntiqtemp/(Nc*Ncic)] , [...] = nearest larger integer ; Ntiq = Niq*Nc*Ncic
   For consistency check, 512<= Niq <= 2048, and Ntiq < 129024*Ncmx;
   if they are not then the explanation/calculation below/above is inconsistent and needs to be corrected/debugged.

7) Program Niq as the argument of the n_iq function.
   Program Nc and Ncic as the arguments of the Nc and icc_ir functions respectively.

8) Calculate tp :
   tp = (10^-7)*Ntiq sec.

The RFon time programmed into the ppg should be = (or >) tp.

9) Compute for n=1 to Niq:
I(n) = <511*Re[func({A*dw*tp}*{n/Niq - 1/2})]>, <...> means closest integer
Q(n) = <511*Im[func({A*dw*tp}*{n/Niq - 1/2})]>

Csech: func(x) = sech(x)^(1+i*5)
a[n] = sech({dw*tp/10}*{n/Niq - 1/2})
phi[n] = 5*ln(a(n))
I[n] = <511*a(n)*cos(phi(n))>
Q[n] = <511*a(n)*sin(phi(n))>

Hermite: func(x) = (1-.957 * x^2) * exp(-x^2)
a[n] = (A*dw*tp*(n/Niq - 1/2)))^2
I[n] = <511*(1-.957*a[n]) * exp(-a[n])>
Q[n] = 0

10) Check that I[<Niq/2>] is either 510 or 511.
Store the I(n),Q(n) data set in decimal and 2's compliment

11) Use the 2's compliment data pairs of (I(n),Q(n))
for n=1 -> Niq for the n_iq function. The same {I,Q}
data goes in all the channels.

<u>End of iq modulation prescription.</u>

<u>Explanation of iq modulation prescription</u>:
(Indented text describes the specific case for ln-sech case.)

i) Chose a functional shape. It can always be expressed as f(dw*A*(t-tp/2)), for 0<=t<=tp, defining the pulse width tp, the band width dw and a scaling factor A. It is assumed that f(0)=1, f(t>0)<1, and f(dw*A*tp/2)<<1 . The reason for the last constraint is so that the RF power turns off properly within the defined pulse shape, otherwise power harmonics at other (non-desirable) frequencies will be introduced into the system at the end of the rf-gate.

For the ln-sech mod function the complex modulation function is
w1(t) = w1_max* (sech(b*t)^(1+i*u) , i^2=-1
or
w1(t) = w1_max*(sech(b*t))*exp^(i*phi(b*t)) & phi(b*t) = u*ln(sech(b*t))
The irradiated line width is dw=2*u*b (i.e. between +- u*b) and a value of u=5 is a good value which delivers a fairly nice selective rectangular frequency selection slice. Therefore the pulse shape is
f=sech(dw*.1*(t-tp/2))^(1+iu), for 0<=t<=tp . i.e. A=.1

Let Niq be the number of digitized iq pairs. The maximum Niq is 2048, and we impose a minimum Niq of 512 to get decent modulation shape resolution/faithfulness. Each Niq pair is read (and interpolated) into the dsp in 4*25*Ncic*Nc nanoseconds. Thus the entire modulation pulse width is tp=10^(-7)*Niq*Ncic*Nc. Where Nc is the number of times (cycles) each iq pair is repeated as it is fed from the iq memory into the dsp modulation digitizers. The total number of 100ns points is Ntiq= Niq*Ncic*Nc and the form of the function in iq memory is iq(n) = <511*f(dw*A*(n - Niq/2))>, The constant 511 reflects a 10 bit bipolar amplitude programmable in binary 2's compliment format.

For the ln-sech function the data in iq memory looks like
iq(n) = <511{sech((dw*tp/10)*(n/Niq - .5))}^{1+i5}>,
= <511{sech((dw*Ncic*Nc*Niq*10^(-7)/10)*(n/Niq - .5))}^{1+i5}>

To chose Ncic, Nc, and Niq first requires relating the band width/shape of the modulation function to the pulse length. Then an approximate Ntiqtemp is determined so that at n=Niq (t=tp) the function is small.

From the value of Ntiqtemp, Ncic and Nc can be determined from a table (in the previous section) and then the value of Niq (and therefore Ntiq and tp) can be determined.

ii) Ntiqtempt calculation: First define a parameter α as follows:

a) For functions that attenuate to zero in time define α so that the value of $f(\alpha)<=.015$.
b) For functions that do not attenuate to zero define α as Ndnu*A*pi where Ndnu is the number of inverse bandwidths you require in the pulse. Ndnu will usually be of the order of unity in these cases.

Then Ntiqtemp is defined so that dw*A*Ntiqtemp*10^(-7)/2=α. i.e.
Ntiqtemp ={ α * 2 *10^(7) / (A * dw) } = {10^9/dw},
{ ... } = nearest smaller integer,

(For the ln-sech function we use α=5, i.e. sech(5) $\approx$.013, which is fine.)

iii) Pick Ncic and Nc from the table. This table was produced to chose the best combination of Ncic*Nc*Niq that will deliver a faithful modulation pulse. As a guideline we tried to keep Niq reasonably high to yield good resolution in the modulation line shape. However, if dw is sufficiently high, then one requires smaller Ncic*Nc*Niq to do the job. Using the guideline that the minimum acceptable Niq is 512 then the product of Niq*Nc*Niq can be categorized as in the table to cover the entire dynamic range

Ntiqtemp	Nc	Ncic
1024 - 2047	1	2
2048 - 4095	1	2
4096 - 8191	1	4
8192 - 16383	1	8
16384 - 32767	1	16
32768 - 65535	1	32
65536 - 129023	1	63
129024 - 258047	2	63
258048 - 516095	4	63
516096 - 1032191	8	63
1032192 - 2064383	16	63
2064384 - 4128767	32	63

iv) Calculate the Niq = [Ntiqtemp/(Nc*Ncic)] that is required. […] = next largest integer.

v) Also one must ensure the frequency band requested is within the physical limits available. These limits depend on the modulation function chosen and the requirement of the smallness of f at tp/2.
Min Ntiq=Ntiqmn=2*512, Max Ntiq=Ntiqmx=63*2048*Ncmx. Ncmx is currently 32. The relationship between Ntiq and dw is dw*A*Ntiq*5*10^(-8)=α, therefore

dw_max = 2*10^(7) α / (A * 512 * 2) (in radians/sec),
dw_min=dw_max / (128 Ncmx)

dw must be constrained to be within this range.

for the ln-sech, A=.1 and α = 5 giving
dw_max=10^9/(512*2)rad/sec = 976.56Krad/sec = 155.42KHz/sec
dw_min =10^9/(2048*63*Ncmx)rad/sec = 7.7505/NcmxKrad/sec = 1.2335/Ncmx KHz/sec

The order of the programming will not follow the order of the explanation .., but should follow the order of the example implementation for the ln-sec function in the previous section.

End of iq modulation description.