DAJ - Here is JOS's translation into english of the generator processing.
See also SAMSIM.SAI[SIM,DAJ] for a SAIL implementation (with all the right
word sizes, etc.).

GENERATORS

Parameters

Associated with each generator are the following quantities:

GD FrqSwp20 (20 bits) alpha -- oscillator frequency sweep rate
GS OscFrq28 (28 bits) omega -- oscillator frequency
GK OscAng20 (20 bits) theta -- oscillator angle
GN NumCos11 (11 bits) number of cosines to be summed
GM CosScl4 (4 bits) binary scale of cosine or sum of cosines
GP AmpSwp20 (20 bits) delta -- decay rate
GQ CurAmp24 (24 bits) phi -- decay exponent
GL AmpOff12 (12 bits) asymptote
GSUM OutSum6 (6 bits) sum memory address into which output is added
GFM FmSum7 (7 bits) sum memory address from which frequency modulation
data is taken
FmSum7 = QAAAAAA
Q: 0 generator-last-pass quadrant
1 modifier-last-pass quadrant
AAAAAA: sum address within quadrant
Gmode 3 (10 bits) generator mode
Gmode10 = RRRREESSSS

Processing

Calculations performed for a generator, governed by its mode, proceed as detailed below.

1) The word in sum memory addressed by FmSum7 is read (20 bits); the sum is formed of it and the high-order 20 bits of OscFrq28 (call the result FmPhase20).

2) If the oscillator side is running, FrqSwp20, right-adjusted with sign extended, is added into OscFrq28.

3) If the oscillator mode is SIN(J+Fm), FmPhase20 is taken; otherwise OscAng20. Call the 20-bit result Phase20, and its high-order 13 bits Phase13.

4) If the oscillator side is running, FmPhase20 is added into OscAng20.

5) If the run mode is WRITEDATA, the word in sum memory addressed by FmSum7 is sent to the CPU as the next write-data item; if the run mode is DACOUT it is sent to the DAC addressed by the low-order 4 bits of FrqSwp20.

6) In oscillator modes other than SIN(K) and SIN(J+Fm), Phase13 is multiplied by NumCos11. Call the low-order 12 bits of the product, with two bits equal to 01 appended to the right, the 14-bit result SinAdr. In oscillator modes SIN(K) and SIN(J+Fm), SinAdr is the high-order 13 bits of Phase20, with a bit equal to 1 appended to the right.

7) If the oscillator mode is SIN(K) or SIN(J+Fm), pi/2 is taken (the binary number 010...0); otherwise Phase13. Call the result CscAdr.

8) In floating point, the product csc (CscAdr) * sin (SinAdr) is formed; then converted to fixed point with a scale factor of 2**(-CosScl4). Call the result (13 bits) TblOut13.

9) The result of the oscillator side (13 bits, call it OscOut13) is
then determined according to the oscillator mode.

```
SSSS: SUMCOS     TblOut13
      SAWTOOTH   Phase13 (but 0 when Phase13 is 1000000000000)
      SQUARE     -1/2 (on a scale from -1 to +1) if Phase13 is negative,
                   else +1/2
      PULSE      +1/2 if overflow occured in step 1) or 4) above;
                   else 0.
      SIN(K)     TblOut13
      SIN(J+Fm)  TblOut13
```

10) The high-order 12 bits of CurAmp24 are taken (call the result CurAmp12).

11) If the envelope side is running, AmpSwp20 right-adjusted, sign
extended, is added into CurAmp24 (overflow dealt with according
to the run mode). (The overflow condition is CurAmp24 changing
sign such that the high-order bit of the resultant CurAmp24 equals
the sign bit of AmpSwp20.)

12) If the envelope mode is 10 or 11, 2**(-CurAmp12) is looked up;
otherwise CurAmp12 is taken. Call the resulting 12 bits NewAmp12.
Scaling is such that if CurAmp12 is 0 then 2**(-CurAmp12) is
111 111 111 101 binary; if CurAmp12 is 000 100 000 000 binary,
then 2**(-CurAmp12) is 011 111 111 110.

13) If the envelope mode is 01 or 11, NewAmp12 is added to AmpOff12; else
it is subtracted from AmpOff12. This creates Env12, the result
of the envelope side.

14) OscOut13 is multiplied by Env12. If the run mode specifies adding
into sum memory, the high-order 19 bits of the rounded product,
right-adjusted with sign extended, are added into the sum
memory location designated by OutSum6; except that in run mode
READDATA, the product is added to the next read-data item from the
CPU and the sum replaces the contents of the sum memory
location addressed.

OK (type a command or type opt for Options):

CCRMA NOTES

Write data format: 4 bits of 0, then 20 bits of data, then 12 bits of 0. (DAJ)

Use of special no-ops:

1. The timer no-op with all x bits=0 is being used for encoding Slog and EdSam

   information (like breakpoints). (DAJ April 26, 1983)

2. The timer no-op with the x bits 29 and/or 30 set is being used for encoding

   comments as follows: (DAJ/AWN June 12, 1983)

   bit 29 bit 30

0	1	The data field of this instruction has 2 text characters. (stored as 7 bit ascii, right justified)
1	0	start of comment
1	1	end of comment

   These commands have bits 33-35 = 0.

3. Analog output lowpass 3dB points at 4.5 kHz, 9, 13.5 and 18 kHz
   (i.e., A=4.5, B=4.5, C=4.5) [old DACs]

4. It's not clear from the documentation, so to clarify: On the # TICKS
   command, the number to be supplied for Q=1 is the total number of ticks
   per pass minus 2. (TVR - 7 August 1984)

5. BIL has discovered empirically that the modifier latch mode operation
   (page 15, line 28), which reads,

   00100: latch (sample and hold). S := L1; If B*M1 is not 0, L1 := A

   should actually read

   00100: latch (sample and hold). S := L1; If B*M1 is not 0, L1 := A*M0

6. DAJ - If the #ptix field has a value > 255, a single write data generator
   will write double samples! (#TICKS command, Q=0)

7. The "don't care" high order bits in the high-ticks command are ignored
   (the #TICKS command, Q=1). This means that the box cannot run slower

   than 5003 Hz.

8. In generator processing description, step 12, it says:

   Scaling is such that if temp6 is 0 then 2**(-temp6) is
   111 111 111 101 binary; if temp6 is 000 100 000 000 binary,
   then 2**(-temp6) is 011 111 111 110.

   The scaling involved is a left shift of temp6 by 4 bits.

9. 23-Jun-86 0026 DAJ

   What IS the rationale between '10 - '17 for dacs?

   There are two sets of slots for DAC cards (upper and lower). The upper
   slots are for 14-bit DACs, the lower for 16-bit. There is room for 8
   14-bit DACs, so they are addresses 0 to 7. Pete put the new DAC card
   into the slot corresponding to '13 (who knows why) in the 8 16 bit DAC
   slots '10 to '17. Actually there are 4 slots for each, but each slot
   has two channels (potentially) with the upper being first in addressing.
   Slots are counted from right to left, so '13 is the lower connector of
   the second slot from the right of the lower set of slots. Actually...

10. DAJ - It seems that (from pg. 16, line 3) the modifier mode one pole
    which is given as

```
    10001:  one pole.  S := L1*M1 + B*M0; L1 := S
```

is really

```
    10001:  one pole.  S := L1*M1 + B*L0; L1 := S
```