Apple II
Technical Notes

Developer Technical Support

Apple IIGS
#37: Free-Form Synthesizer Tips

Revised by: Jim Mensch November 1988
Written by: Jim Mensch May 1988

This Technical Note is intended to help a person who is unfamiliar with the Apple IIGS Sound Tool Set use the Free-Form Synthesizer effectively.

The primary function of the Free-Form Synthesizer is to allow an application program to start one or more complex digitized or computed waveforms playing on the Apple IIGS without further intervention from the application. The waveform is a series of bytes, each representing the amplitude of your outgoing sound at a particular moment in time (defined by the sampling frequency you set). After a call to `FFStartSound`, the Sound Tool Set takes care of all chores involved in loading the DOC RAM, setting up registers, and actually playing your sound. Once playing, your sound will continue until either the Sound Tool Set encounters a `NIL` pointer in the waveform list, or until you call `FFStopSound`.

FFStartSound Parameters

`FFStartSound` has only two parameters: the first a `Word` containing channel, generator, and mode information, and the second a `Pointer` to a parameter block.

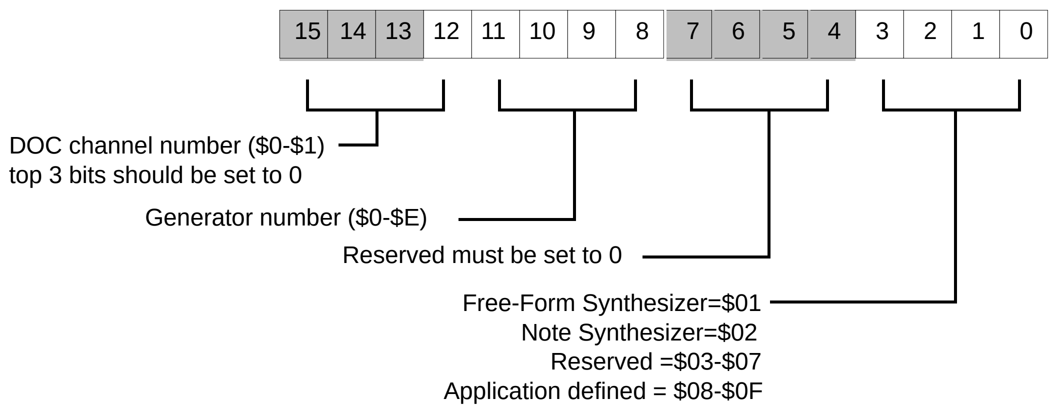
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DOC channel number ($0-$1)
top 3 bits should be set to 0
Generator number ($0-$E)
Reserved must be set to 0
Free-Form Synthesizer=$01
Note Synthesizer=$02
Reserved =$03-$07
Application defined = $08-$0F

Figure 1 – Channel-Generator-Mode Word

The Channel-Generator-Mode `Word` is broken down into 4 nibbles. The low-order nibble specifies the particular synthesizer you are using. (Because this Note is only about the Free-Form Synthesizer, we will be using only a 1 in this nibble.) The adjacent nibble must be set to 0 for now. The next nibble specifies which generator to use. The IIGS has 15 generators from which to choose, and as the application designer, it is up to you to decide which one to use. It might be appropriate, however, to call `FFGeneratorStatus` first to ensure that the generator currently is available. (It could be in use already by a desk accessory or previously started sound.) The high-order nibble specifies which channel to use. The IIGS supports two separate sound channels for output. If you are using a stereo adapter, you could start up many sounds and route them to either channel 0 or channel 1 to get a full stereo effect. (The channel is ignored if you are not using a special piece of multi-channel hardware.)

The parameter block contains parameters describing the sound and how it should be played. Here is a sample Pascal definition of that parameter block:

```
FFParmBlock = record
              waveStart:Ptr;
              waveSize:Integer;
              freqOffset:Integer;
              DOCBuffer:Integer;  { High order byte significant }
              bufferSize:Integer; { Low order byte significant }
              nextWave:^FFParmBlock;
              volSetting:Integer;
             end;
```

The first parameter is a 4-byte address telling the Free-Form Synthesizer where in memory it can locate your sample data. The next parameter is a word specifying the number of 256-byte pages of sound you wish to play. The waveform data should be a series of bytes, each representing one sample. Wave tables must be exact multiples of 256 bytes.

Note: A zero value in the waveform can cause a sound to stop, so be sure to check your data to ensure that this does not happen.

The frequency offset parameter specifies the sampling frequency that the Free-Form Synthesizer should use during playback. This number can be computed by the following formula:

`freqOffset` = ((32*Sample rate in Hertz)/1645)

The frequency offset parameter is the most often misunderstood parameter, so I will explain a little about sampling rates. The sampling rate is how many samples (bytes) per second to play. If you have a digitized wave that represents 2 seconds of sound, and it takes up 44K of memory, then it was sampled at 22 kHz (which, by the way, is good for full sound reproduction). The sampling rate must be at least twice that of the maximum fundamental frequency you want to sample. However, for good sound reproduction, you may want to sample at least eight times the fundamental frequency in order to capture the higher harmonics of musical instruments and the human voice.

The DOC starting address and buffer size tell the Free-Form Synthesizer which portion of the 64K sound RAM to use as a buffer during playback. The wave is taken from your waveform in chunks and placed in sound RAM for playback. Each time the buffer nears empty, it will need to be reloaded with more sound. The size of the buffer specified determines how often the Free-Form Synthesizer must interrupt the 65816 to reload the buffer. The buffer size must be a power of two because of the way the sound General Logic Unit (GLU) specifies addresses. (The value for this parameter must also be a power of two.) A good length to use would be at least 1/10 second of sound. For example, if you were using a sampling rate of 16 kHz (16,000 samples per second), you would want a buffer at least 2,048 bytes long, or about 8 pages. It does not hurt to round this number up. You manage the DOC RAM, so you should decide what memory to use. It is usually a good idea to have multiple buffers if you have a chain of waves. (I like leaving page zero free, as the Note Synthesizer uses the data in the first 256 bytes, and accidentally placing a zero in that page could cause it to fail.)

The next wave pointer is a 4-byte pointer to the next parameter block. With this parameter you can string together many waveforms for more continuous sound, or you can make your sounds infinitely recursive by pointing back to the original wave form.

The volume setting is a word which represents the relative playback volume. It can range from 0 to 255.

Other Tips

When you shut down the Sound Tool Set, it will stop all pending sounds, so be sure to leave ample time between starting and ending a sound. If you have a series of wave forms strung together, you can change their parameters on the fly. Changes take effect as soon as the waveform is started. (You could use this to find the correct sampling frequency of a wave, by having the next wave pointer point back to the start of your parameter block. This would cause the sound to play indefinitely. You then could change the `freqOffset` value, and the sound would change each time it is restarted.)

Here is a sample code segment (in APW Assembler format) that creates a 1-kHz wave in memory sampled at 16 kHz and plays it:

```
FFSound       DATA

theSound      ds     $2000              ; FFSound wave...
MyFFRecord    dc     A4'theSound'       ; address of wave
              dc     i'$20'             ; size of wave in pages..
Rate          dc     i'311'             ; 16-kHz sample rate
              dc     i'1'               ; DOC starting address
              dc     i'$0800'           ; DOC buffer size
              dc     a4'0'              ; no next wave
Vol1          dc     i'$007F'           ; kinda medium..

; 1-kHz triangle wave sampled at 16 kHz one full segment
oneAngle      dc     i1'$40,$50,$60,$70,$80,$90,$A0,$B0'
              dc     i1'$C0,$B0,$A0,$90,$80,$70,$60,$50'
              End

TestFF        Start
              Using  FFSound
MakeWave      ANop
              ldx    #$0000
MW0010        txa                       ; get index
              and    #$000F             ; use just low nibble as index
              tay                       ; into triangle wave table
              lda    oneAngle,y         ;
              sta    theSound,X         ; and store it into sound buf
              inx
              inx
              cpx    #$2000             ; we Done?
              blt    MW0010             ; nope better finish
              PushWord     #$0001
              PushLong     #MyFFRecord
              _FFStartSound
              rts
              end
```

Further Reference

- *Apple IIGS Toolbox Reference*, Volume 2