NGS Modules Overview

PlayStation®Vita Programmer Tool Runtime Library Release 3.50

Table of Contents

1 DSP Effects Module Overview

This document provides an overview for each type of DSP effect Module supplied with NGS, including:

DSP Effects

- ATRAC9™ Player DSP effect Module – Handles streaming of audio data to an NGS Voice and resampling of input ATRAC9™ data to a user defined frequency (Hz).
- Compressor DSP effect Module – Handles both compression and ducking effects.
- Delay DSP effect Module – Handles the emulation of discrete audio reflections.
- Polynomial Distortion DSP effect Module – Handles distortion effects to add harmonics to a sound.
- Amplitude Envelope DSP effect Module – Handles the amplitude alterations over time of an audio signal.
- Filter DSP effect Module – Handles frequency-specific filtering/EQ modifications in the time domain.
- Input Mixer DSP effect Module – Handles the connection between Voices, acting as an input for other voices to write into.
- Mixer DSP effect Module – Handles the amplitude variation of audio signals between two Modules.
- Parametric EQ DSP effect Module – Handles time domain equalization.
- Pauser DSP effect Module – Handles fading out/in of the audio signal if the user pauses or resumes a Voice.
- Player DSP effect Module – Handles streaming of audio data, decoding of HE-ADPCM data to PCM and resampling of input PCM/ADPCM data to a user defined frequency (Hz).
- I3DL2 Reverb DSP effect Module – Handles reverb effects.
- Signal Generator DSP effect Module – Handles continuous tone generation within a Voice.
- Output Module – Handles returning PCM data back to the user.

Each type of DSP effect has an associated structure; each of which is described in the *NGS Modules Reference*.

Accessing and Modifying Parameters Within DSP Effect Modules

There are 2 methods within the API to modify parameters, they are outlined below:

Method 1: Lock / Unlock Parameters

The procedure for using the DSP effects follows a common pattern for each effect. Unless otherwise specified, the functions referred to here are defined in the *NGS Reference*:

(1) Call `sceNgsVoiceLockParams` to access a DSP effect Modules parameters within a Voice. These parameters are stored within a structure type which is specific to the specific DSP Module. During this "lock" period, NGS will continue to process audio data using the parameters stored previous to the lock command being issued.

(2) User can now modify any parameter within the Module parameter structure.

(3) Call `sceNgsVoiceUnlockParams` to allow any changes to the previously locked DSP effect Module to be processed.

Method 2: Block Parameter Setting

This method is supplied as a more efficient way of data-driving parameter settings. For example, from loaded parameter sets that have been developed offline. It also allows the user to setup multiple Modules within a Voice from a single command. Refer to the *NGS Reference* for each Modules `ParamsBlock` structure(s). Effectively, these structures are the same as those used in the lock/unlock method, but also contain an extra header with information on the Module identifier and audio channel.

(1) Create / load your `ParamsBlock` structure(s). To setup multiple Modules each data block should be contiguous within the same memory block.

(2) Call `sceNgsVoiceSetParamsBlock`.

Multichannel DSP Effect Modules

DSP effect Modules can process a single audio channel (mono) or multiple audio channels (2 or more). To detect the number of audio channels that can be processed within a Module, it is required to check the *`pnMemSize`* parameter returned from `sceNgsVoiceLockParams`:

Example (using a Filter DSP Effect Module):

```
SceNgsBufferInfo BuffInfo;
sceNgsVoiceLockParams(VoiceH, index, SCE_NGS_FILTER_PARAMS_STRUCT_ID,
&BuffInfo);
```

Note that parameter data for each audio channel is stored in sequence:

```
SceNgsFilterParams* pParams = (SceNgsFilterParams*)BuffInfo.data;
SceNgsFilterParams* pLeftSettings = pParams;
pParams++;
SceNgsFilterParams* pRightSettings = pParams;
```

Using DSP Effect Module Presets

Presets are supplied with some modules in order to aid quick setup, they can be accessed as follows, note that for modules that require stereo parameters you will also need to initialize the right channel as mentioned previously.

```
SceNgsBufferInfo BuffInfo;
sceNgsVoiceLockParams(VoiceH, index, SCE_NGS_REVERB_PARAMS_STRUCT_ID,
&BuffInfo);
sceNgsModuleGetPreset(SystemH, SCE_NGS_REVERB_ID, SCE_NGS_REVERB_PRESET_CAVE,
&BuffInfo);
sceNgsVoiceUnlockParams(VoiceH, index);
```

2 ATRAC9™ Player DSP Effect Module Overview

The ATRAC9™ Player DSP effect Module performs a number of tasks within NGS:

- Handles streaming of audio data to an NGS Voice
- Resampling of input ATRAC9™ data to a user-defined frequency (Hz)

Audio Data Streaming

The ATRAC9™ Player Module handles the streaming of audio data from memory into a Voice. From here, the NGS system processees the data through all other Modules within the Voice, and then onto subsequent Voices, before finally being ready to be played through the speakers. With this in mind, the ATRAC9™ Player Module can be seen as the place where the audio starts its journey.

The user can specify up to 4 buffers, which contain audio data playback information. Each buffer contains:

Buffer	Memory address pointing to the location of audio data for the Voice to process.
Size	Size of the buffer to play.
Loop Count	Number of times to repeat this buffer.
Next Buffer	Next buffer to process (0–3). Or -1 to end playback once the currently playing buffer has finished.

If you are intending to playback PCM or ADPCM data please use the Player DSP effect Module.

Module Callbacks

It is possible to register a module callback for the ATRAC9™ Player per module instance using `sceNgsVoiceSetModuleCallback()`, this will then trigger in various conditions as shown in the following table:

Condition	nCallbackData	nCallbackData2	pCallbackPtr
End of playback	`SCE_NGS_AT9_END_OF_DATA`	0	0
Swap of buffer	`SCE_NGS_AT9_SWAPPED_BUFFER`	Previous buffer index	Current buffer address
Restart (loop) of buffer	`SCE_NGS_AT9_LOOPED_BUFFER`	Loop count (number of loops performed)	Current buffer address
Header decode error	`SCE_NGS_AT9_HEADER_ERROR`	0	0
Decode error	`SCE_NGS_AT9_DECODE_ERROR`	Byte offset in current buffer	Current buffer address

The callbacks are generated from within the `sceNgsSystemUpdate()` call, post processing.

3 Compressor DSP Effect Module Overview

The Compressor effect includes both compression and ducking effects.

Standard Compressor

A compressor is an automatic volume control that reduces the volume when the input becomes louder than a specified threshold.

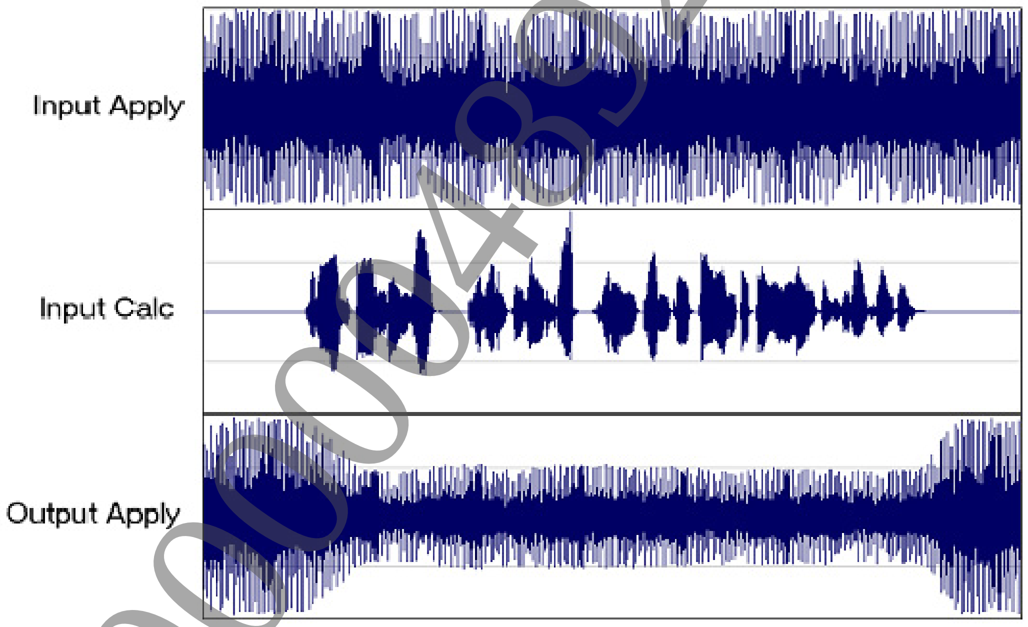
Side Chain Compression (Ducking)

Side chain compression uses a compressor to control the volume of a signal based on the amplitude of a second signal.

This effect is used in real-world situations such as sports or radio commentary, reducing the amplitude of the background signal (i.e. music or the sound of a sports game) when the commentator is speaking, in order to aid the clarity of speech and also to control the overall output volume of both sources.

When the commentator starts speaking, the volume of the other audio signal starts to drop and maintains the set level until the speaker stops, at which point the volume will ramp back up (Figure 1).

Figure 1



Note that it is possible to select the side chain behavior, when the side chain signal is not present, by specifying either `SCE_NGS_COMPRESSOR_PARAMS_STRUCT_ID` or `SCE_NGS_COMPRESSOR_PARAMS_STRUCT_ID_V2` in the parameter structure. For example, when using the side chain compression voice function, `sceNgsVoiceDefGetCompressorSideChainBuss()`, if input 1 is bypassed, this switches off the side chain input. When `SCE_NGS_COMPRESSOR_PARAMS_STRUCT_ID` is specified, the compressor treats the side chain input as having zero volume. When `SCE_NGS_COMPRESSOR_PARAMS_STRUCT_ID_V2` is specified, the compressor reverts to standard mode (i.e. not side-chain), bypassing the side-chain input. This provides the user with extra flexibility: in a rack containing side-chain compressor voices, each voice can be independently set to side-chain or normal mode by bypassing the side chain input mixer function, `sceNgsVoiceBypassModule()`.

Setting up Standard and Side Chain Compression in NGS

The NGS compression effects can act as both a standard compressor or as a ducker (via side chain compression).

For standard compression, the audio which is to be compressed ("Input Apply" in the previous diagram) has to be routed to Input Port 0 of the Compressor DSP effect Module.

To apply a ducking compression, the audio which is to be compressed ("Input Apply" in Figure 1) has to be routed to Input Port 0 of the Module. The ducking signal ("Input Calc" in Figure 1) has to be routed to Input Port 1 (see the *NGS Overview* for details). If no signal is routed to Input Port 1 the compressor performs "standard" compression.

Compressor Controls

The compressor has a number of parameters you can control. For each compressor, you pass:

Ratio	The amount volume is reduced by in relation to how far above the threshold the signal is. A ratio of 1:1 does nothing. A ratio of *n*:1 means if the input rises to *n* dB above the threshold, the compressor allows the output to rise to 1dB over the threshold. The ratio is specified as a multiplier, thus a 5:1 compression ratio = 1 / 5 = 0.2
Threshold	The set point at which the automatic volume reduction kicks in. Below this volume the compressor does nothing. When the input exceeds this level, the compressor reduces the volume automatically, based on the *`fRatio`* and *`fAttack`* parameters.
Attack	The rate at which the volume is reduced once the input exceeds the threshold. If the attack rate is too slow, therefore reducing the volume slowly, distortion can occur as audible bursts of sound are heard breaking the threshold.
Release	The rate at which the volume returns when the input is no longer above the threshold.
Makeup Gain	The output audio control for the compressed audio signal. Since the function of a compressor is only to reduce the volume when the signal is too high, the makeup gain (output volume) control lets you bring the compressed audio back up to an acceptable level. This can have the effect of making the perceived volume of the signal louder than it was originally. The makeup gain is useful if you want to increase the volume of a signal, but reduce peaks that could cause distortion.
Stereo Link	A flag which allows the compression for both left and right audio channels to be calculated from an average or the maximum of each signal.
Peak Mode	A flag which sets the compressor to either "Peak" mode or "RMS" mode. "Peak" mode will calculate the input signal based on the highest audio signal amplitude. "RMS" mode calculates the input signal based on the average (root means squared) audio signal.
Soft Knee	The soft knee value is the dB range over which a "soft knee compression" will operate. Soft knee compression slowly increases the compression ratio as the input level increases, until it finally reaches the ratio specified by the user. Setting this value to 0 will result in "hard knee compression", where the gain is reduced abruptly if the input audio signal exceeds the threshold level.

Querying Compressor Operation and Volume Levels

It's possible to query the state data from the compressor module in order to gain some metrics on the throughput, see `sceNgsVoiceGetStateData()` and `SceNgsCompressorStates`.

Once the state data has been retrieved it is possible to examine the input and output as follows:

- `fInputLevel` reflects the input volume to the compressor, expressed as a linear value. Note that if the compressor is being used in side-chain mode then this is the input from the controlling side-chain path. Also the formatting of the input value is based on the compressor mode (RMS or peak).
- `fOutputLevel` reflects the output volume of the compressed signal, in standard mode this is the input value multiplied by the gain applied by the compressor, in side-chain mode this reflects the gain applied to the processed signal.

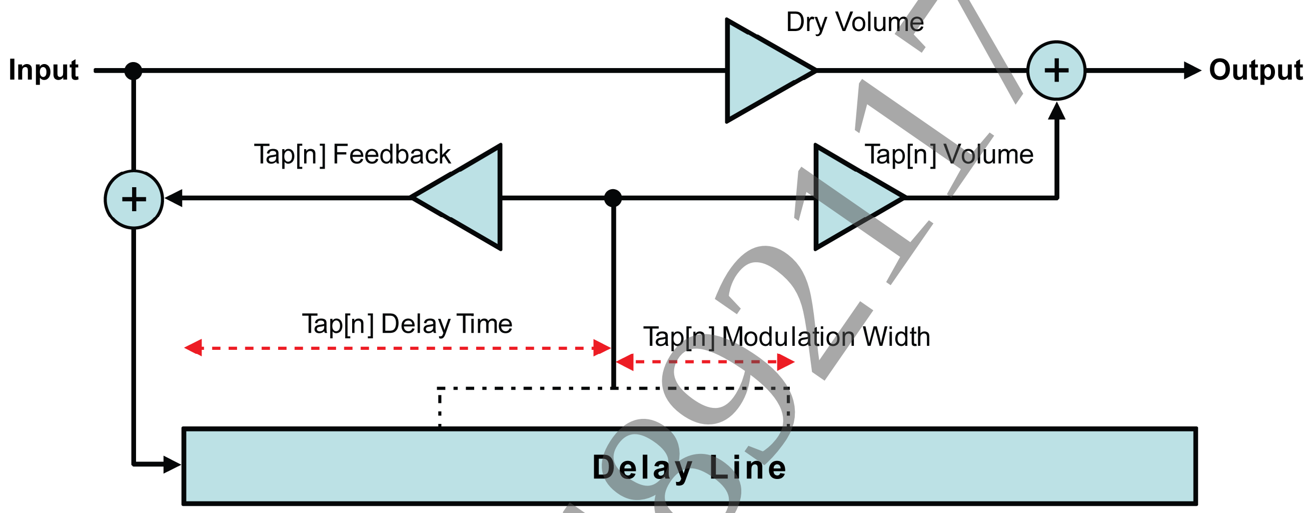
4 Delay DSP Effect Module Overview

The Delay effect emulates discrete audio reflections, such as echoes. This is the type of effect you might get if you went to the Grand Canyon and shouted "Hello!".

Internally there is a single delay line with four separate outputs (also called "taps"). Each of these taps contains an independent volume, feedback and filter as well as modulation controls.

The modulation of the taps is controlled by a single shared LFO (Low Frequency Oscillator), with each tap having an individual phase offset and modulation width.

Figure 2 Delay Module Diagram



5 Polynomial Distortion DSP Effect Module Overview

Distortion effects add harmonics to a sound. Distortion is commonly used on instruments such as electric guitars.

Distortion Algorithm

The Polynomial Distortion algorithm takes one of the two following forms:

- Output = Input*A – ((Input*Input) *B)
- Output = Input*A + ((Input*Input) *B)

The first case is used if the Input signal is a positive value and the second case is used if the Input signal is a negative value. Following this, gate and limiting is performed.

Polynomial Distortion Controls

You can use six floating point parameters to control the Distortion DSP effect. These values are passed into the `SceNgsDistortionParams` data structure.

- A*Input – Controls the A value for the distortion algorithm. Value ranges from 0 to 10.
- B*Input – Controls the B value for the distortion algorithm. Value ranges from 0 to 10.
- Limit – Limiter (range 0-1). Any audio output signal larger than this value is set to this value.
- Gate – Noise gate (range 0-1). Any audio output signal smaller than this value is set to zero (silence).
- Wet Gain – Sets the output amplitude of the distorted signal. Where 0 = silence, 1 = original output, 2 = twice as loud, and so on.
- Dry Gain – Amount of the original (dry) input signal to mix with the distorted (wet) signal. Where 0 = no dry signal, 1 = original dry signal, 2 = twice original dry signal, and so on.

Re-creating the Original Dry Signal if using the Polynomial Distortion Algorithm

To re-create the original input signal, set:

- A = 1
- B = 0
- Limit = 1
- Gate = 0
- Wet Gain = 1
- Dry Gain = 0

6 Amplitude Envelope DSP Effect Module Overview

The Amplitude Envelope DSP Module causes the amplitude of the audio signal to be modified over time. Unlike a standard ADSR envelope, this Module allows for more flexibility. Up to 4 points can be defined by the user. Each point contains:

Time to next point	Time (in milliseconds) it will take to traverse between this point and the next.
Amplitude	Amplitude of this point.
Curve type	Linear or curved line between this point and the next. The curve type is Cubic.

The user can also specify both a loop start and end point within the envelope. This allows for tremolo type effects.

You cannot move to another envelope point when processing is in between points, but you can modify the position of a point mid-processing; thus, creating an effect of moving to another point mid-processing. For example, if you create an envelope of 4 points with the middle 2 looping (tremolo) and you want to exit the looping envelope “mid-play”. You initially specify the destination point of each envelope, effectively limiting the point at which you can exit the envelope. However, you then modify the X and Y position of a point when processing, subsequently moving the start and end points.

Note that the amplitude envelope uses the time that the voice has been active to determine the correct position within the envelope, so bypassing the envelope will still result in the correct position being used if it’s not bypassed at a later date.

Key Off / Release Rate

The envelope has a “Key Off” mode. If “Key Off” is specified, the envelope will enter a release phase, which results in the overall amplitude of the envelope fading to silence over a specified time (the release rate). A release rate can be specified by the user in milliseconds.

Key Off / Finalize Voice

When an envelope has been keyed off for the time specified by the release rate (see Key Off / Release Rate), one of two possibilities can be selected to take place:

(a) Voice will automatically stop
(b) Voice will continue to process

In most cases, the user will require the Voice to automatically stop when the release rate phase is complete. However, if DSP effects such as delay or reverb are being processed on the Voice, it may be required that the Voice stay active for a period of time so that these DSP effects can end naturally. Note that the length of time for keeping the Voice active to consider such DSP effects is left to the judgment of the developer.

Loop Control

If *nLoopEnd* is a positive value it specifies the envelope point on which the loop should return to the start.

There are also three special loop control flags which can be used in *nLoopEnd*:

SCE_NGS_ENVELOPE_NO_LOOP

No looping occurs and the envelope remains indefinitely at the final points volume until either it is terminated by `sceNgsVoiceKill()` or switched into the release phase by `sceNgsVoiceKeyOff()`.

SCE_NGS_ENVELOPE_NO_LOOP_TERMINATE

No looping occurs and the envelope remains at the final points volume for a finite period of time, specified by the final points time. The voice then terminates as though `sceNgsVoiceKill()` had been called.

SCE_NGS_ENVELOPE_NO_LOOP_RELEASE

No looping occurs and the envelope remains at the final points volume for a finite period of time, specified by the final points time. The voice then switches to release mode as though `sceNgsVoiceKeyOff()` had been called.

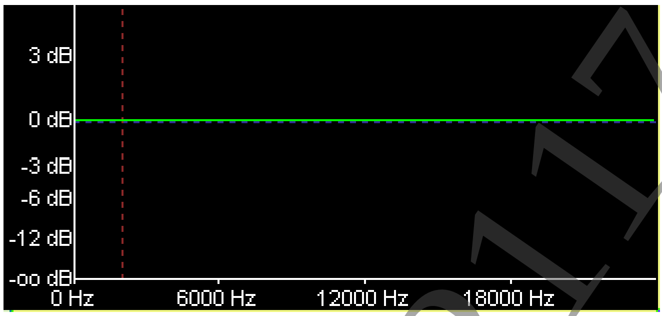
7 Filter DSP Effect Module Overview

The Filter DSP effect Module allows frequency-specific filtering/EQ modifications in the time domain. The following modes are supported (examples output is shown for each mode).

SCE_NGS_FILTER_MODE_OFF

Value = 0. No effect. Figure 3 shows an example.

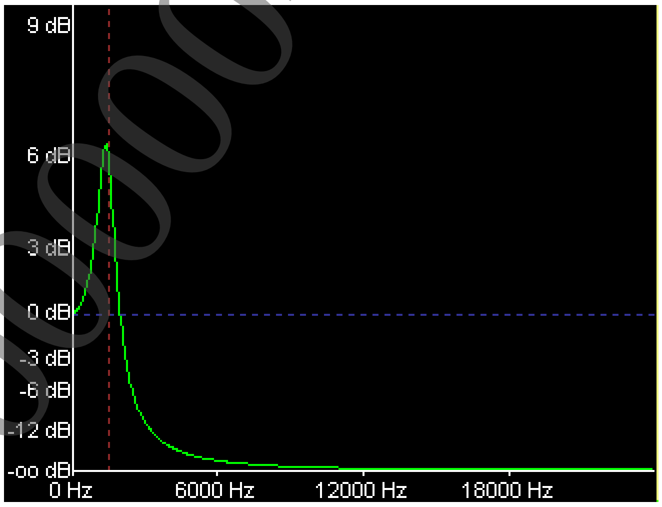
Figure 3 Off



SCE_NGS_FILTER_LOWPASS_RESONANT

A low pass filter removes higher frequencies with a variable volume peak around the cut-off frequency. The roll-off is faster than that of the `SCE_NGS_FILTER_LOWPASS_ONEPOLE` filter mode since this filter is a two-pole design, meaning that the roll-off is -12dB per octave. This filter mode takes both frequency and resonance input values. The *`fFrequency`* parameter defines the frequency at which the roll-off occurs. Figure 4 shows an example.

Figure 4 Cutoff 1500 Hz Resonance = 2



The resonance value controls the resonance of the filter; higher resonance creates a greater accentuation of frequencies around the cut-off frequency.

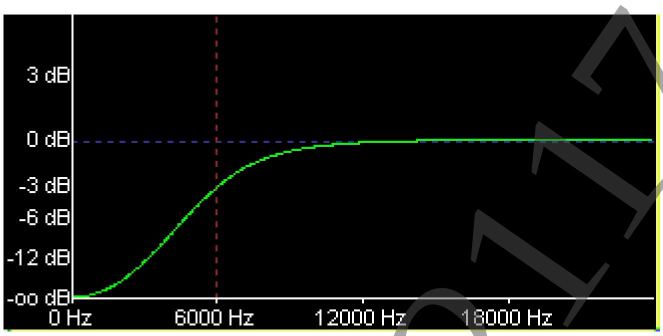
- At resonance = 0.7 the filter has no resonant peak and the cut-off frequency represents the point at which there is a -3dB drop in volume relative to input.

- At resonance = 1.0, the filter has a resonant peak (increase) below the cut-off frequency and a volume output of 0dB difference at the cut-off frequency.
- At resonance = 2.0, the filter has a resonant peak of +6dB at the cut-off frequency.

SCE_NGS_FILTER_HIGHPASS_RESONANT

A high pass filter removes lower frequency with variable resonant peak around the cut-off. This works in a similar manner to the `SCE_NGS_FILTER_LOWPASS_RESONANT` mode, but reduces frequencies below the cut-off frequency. This mode takes both *`fFrequency`* and *`fResonance`* parameters. Figure 5 shows an example.

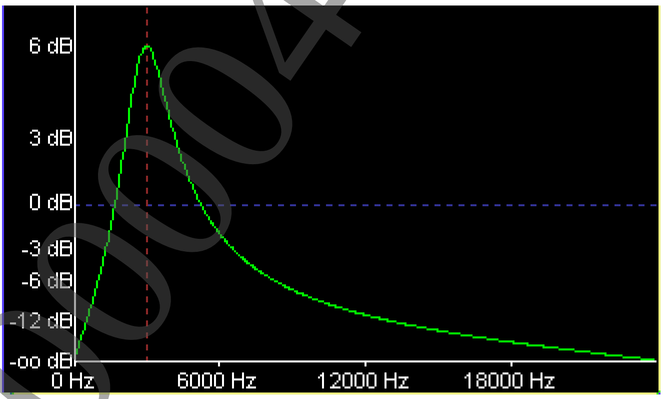
Figure 5 Cutoff 6000 Hz Resonance = 0.7



SCE_NGS_FILTER_BANDPASS_PEAK

This filter mode allows frequencies around the cut-off frequency to pass while reducing frequencies below and above the cut-off frequency. Figure 6 shows an example.

Figure 6 Cutoff 3000 Hz Gain +6dB



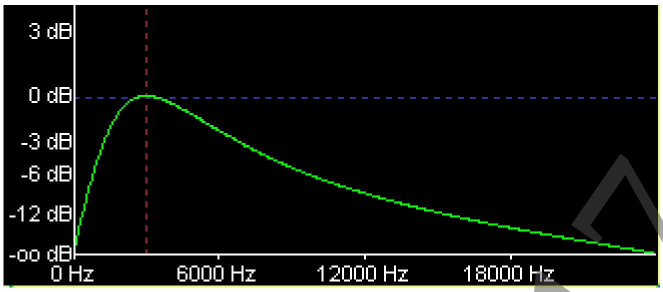
This mode takes both *`fFrequency`* and *`fGain`* parameters.

- Frequency controls the frequency at which the output volume will peak, relative to the input volume.
- Gain controls the volume gain, in decibels at the cut-off frequency.

SCE_NGS_FILTER_BANDPASS_ZERO

This filter mode works in a similar manner to the `SCE_NGS_FILTER_BANDPASS_PEAK` mode, allowing frequencies around the cut-off frequency to pass relatively unaffected. However, frequencies above and below the cut-off get reduced. The maximum output compared to input is 0dB at the frequency supplied. Figure 7 shows an example.

Figure 7 Cutoff 3000 Hz Resonance = 0.5



This mode takes both frequency and resonance input parameters:

- The *`fFrequency`* parameter controls the frequency point at which the output volume is the same as the input volume.
- The *`fResonance`* parameter controls the rate of roll-off from the cut-off frequency. Larger values of resonance give a faster roll-off of volumes per frequency.

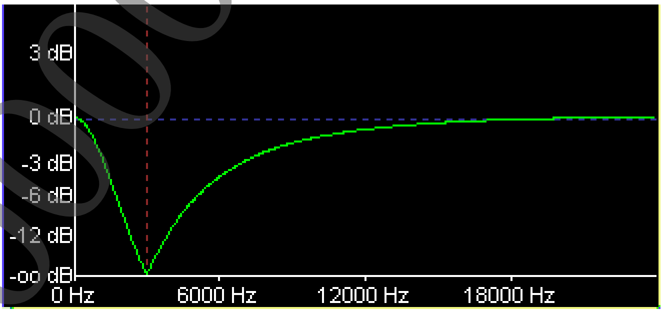
SCE_NGS_FILTER_NOTCH

This filter mode removes frequencies at and around the given frequency value.

- The *`fFrequency`* parameter controls the cut-off frequency at which the greatest volume reduction will occur.
- The *`fResonance`* parameter controls the width of frequencies around the center frequency that will also be reduced. Lower values of resonance represent a greater bandwidth of affected frequencies. Higher resonance values represent a smaller spread of affected frequencies.

Figure 8 shows an example.

Figure 8 Cutoff 3000 Hz Resonance = 0.5



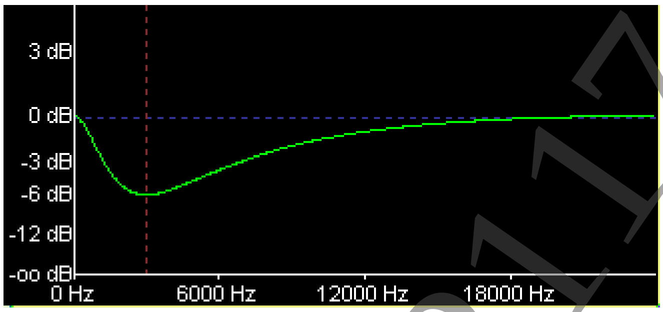
SCE_NGS_FILTER_PEAK

This filter mode provides a high degree of control.

- The *fFrequency* parameter represents the center frequency at which the volumes will be affected.
- The *fGain* parameter controls the amount of volume cut or boost in decibels at the center frequency.
- The *fResonance* parameter controls the bandwidth of affected frequencies. A lower resonance value represents a greater range of affected frequencies around the center frequency. A higher resonance represents a narrower range of affected frequencies.

Figure 9 shows an example.

Figure 9 Cutoff 3000 Hz Gain -6dB Resonance = 0.5



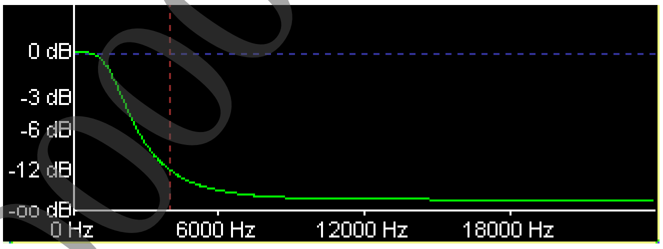
SCE_NGS_FILTER_HIGHSHELF

This filter mode applies a specified volume boost /reduction to frequencies above the transition band.

- The *fFrequency* parameter controls the center frequency of the transition band.
- The *fGain* parameter controls the amount of volume gain or reduction in decibels after the transition band.

Figure 10 shows an example.

Figure 10 Cutoff 4000 Hz Gain -24dB

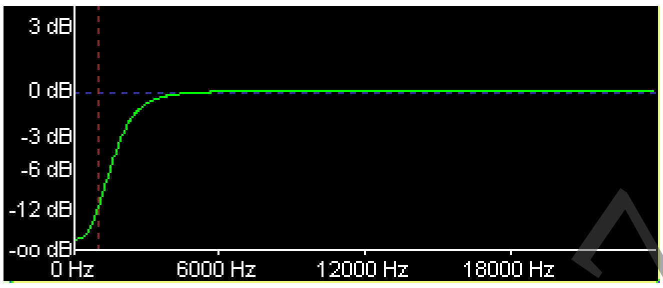


Frequencies below the transition band remain unaffected; frequencies above the transition band are altered by gain in decibels. Frequencies within the transition band are reduced up to a maximum of gain in decibels; with frequencies at the cut-off frequency supplied being modified by half the gain.

SCE_NGS_FILTER_LOWSHELF

This filter mode works in a similar manner to the `SCE_NGS_FILTER_HIGHSHELF` mode, with the exception that frequencies below the transition band will be boosted / reduced by the specified gain in decibels. Figure 11 shows an example.

Figure 11 Cutoff 1000 Hz Gain -24dB

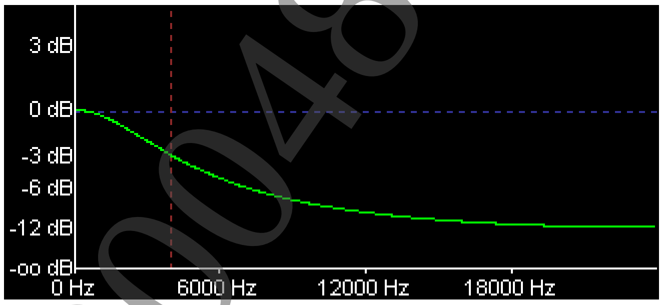


SCE_NGS_FILTER_LOWPASS_ONEPOLE

This filter mode applies a low pass filter which removes higher frequencies.

The *`fFrequency`* parameter corresponds to the cut-off frequency at which there is a -3dB drop in output volume relative to input. This filter has a -6dB per octave roll-off. This means that at double the cut-off frequency, the output will be -6dB and for every doubling of frequency thereafter, the output volume, relative to the input volume, will be reduced by a further -6dB. This filter therefore provides a very gentle/subtle roll-off. Figure 12 shows an example.

Figure 12 Cutoff 4000Hz



SCE_NGS_FILTER_HIGHPASS_ONEPOLE

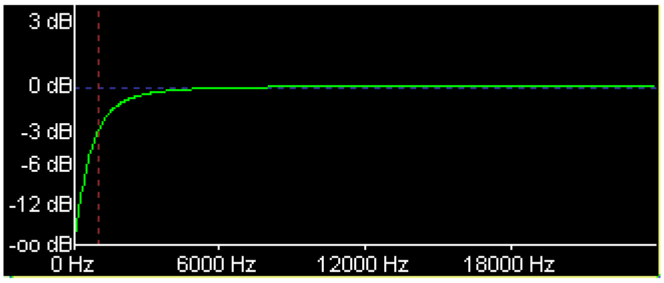
This filter mode applies a high pass filter, which removes lower frequencies.

The *`fFrequency`* parameter corresponds to the cut-off frequency at which there is a -3dB drop in output volume relative to input. This is a -6dB per octave cut-off. This means:

- At frequency/2, the output volume relative to input is -6dB.
- At frequency/4, the output volume relative to input is -12dB.
- At frequency/8, the output volume relative to input is -18dB.
- And so on.

Figure 13 shows an example.

Figure 13 Cutoff 1000 Hz



SCE_NGS_FILTER_ALLPASS

This filter mode produces very little volume difference between input and output frequencies; instead, it modifies the phase of the output frequencies relative to the input frequencies.

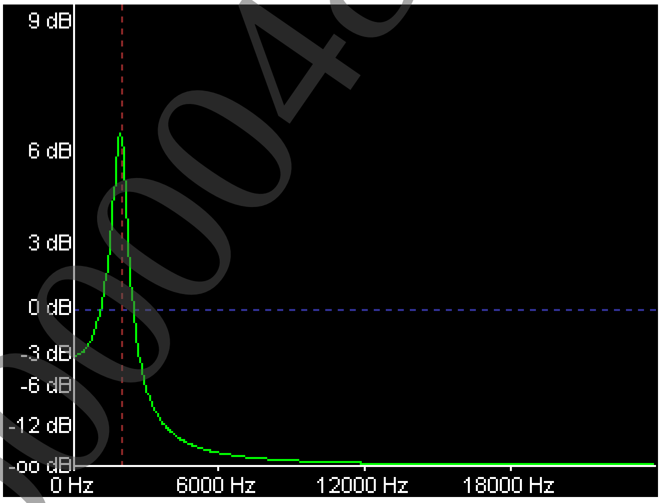
- The `fFrequency` parameter controls the centre frequency for modification.
- The `fResonance` parameter controls the roll-off of phase.

SCE_NGS_FILTER_LOWPASS_RESONANT_NORMALIZED

This filter mode is a special form of the low pass resonant filter. Frequencies below the cut-off frequency are reduced (although in a flat manner across all frequencies) in order to reduce overloading of the output volume. The amount of volume reduction below the cut-off is relative to the resonance. Therefore, a larger resonance value (and larger peak) creates a greater reduction of frequencies below the cut-off.

Figure 14 shows an example.

Figure 14 Cutoff 2000 Hz Resonance = 3.0



8 Input Mixer DSP Effect Module Overview

The Input Mixer Module allows voices to be connected together by acting as an input for other voices to write into. This mixing is controlled via the patch commands in the NGS System Functions.

This should not be confused with the Mixer DSP Effect Module; see Chapter 15, Mixer DSP Effect Module Overview.

For more information on patch commands, see the *NGS Reference*.

9 Player DSP Effect Module Overview

The Player DSP effect Module performs a number of tasks within NGS:

- Handles streaming of audio data to an NGS Voice
- (Optional) Decoding of HE-ADPCM data to PCM
- Allows for sample accurate playback

Resampling of Input Data (PCM or HE-ADPCM) to a User-defined Frequency (Hz) Audio Data Streaming

The Player Module handles the streaming of audio data from memory into a Voice. From here, the NGS system will process the data through all other Modules within the Voice, and then onto subsequent Voices, before finally being ready to be played through the speakers. With this in mind, the Player Module could be seen as the place where the audio starts its journey.

The user can specify up to 4 buffers, which contain audio data playback information. Each buffer contains:

Buffer	Memory address pointing to the location of audio data for the Voice to process.
Size	Size of the buffer to play.
Loop Count	Number of times to repeat this buffer.
Next Buffer	Next buffer to process (0–3). Or -1 to end playback once the currently playing buffer has finished.

If you are intending to playback ATRAC9™ data please use the ATRAC9™ Player DSP effect Module.

HE-ADPCM Decoding

If required, this Module will decode HE-ADPCM data into PCM.

Resampling (Setting the Playback Frequency)

The Player Module allows you to specify the frequency at which the audio data is to be processed. For example, if the original input data (PCM or HE-ADPCM) is required to be played at 48Khz, specifying a playback rate of 24Khz would result in the audio being heard one octave lower, and taking twice the duration as the original.

It is also possible for the user to supply a frequency scalar value. This value is used to scale the specified playback frequency, allowing for techniques such as doppler to be easily implemented.

Sample Accurate Playback

Each NGS update will generate a packet of PCM data containing a fixed number of samples per audio channel. By specifying an offset value, the generated Voice PCM data will be mixed into this packet from the offset.

Note however that other module changes, such as the setting of a playback frequency, will be processed from the beginning of the PCM packet. As such, these DSP effects are not capable of processing data from a specific sample offset.

Module Callbacks

It is possible to register a module callback for the Player per module instance using `sceNgsVoiceSetModuleCallback()`, this will then trigger in various conditions giving the following values:

Condition	nCallbackData	nCallbackData2	pCallbackPtr
End of playback	`SCE_NGS_PLAYER_END_OF_DATA`	0	0
Swap of buffer	`SCE_NGS_PLAYER_SWAPPED_BUFFER`	Previous buffer index	Current buffer address
Restart (loop) of buffer	`SCE_NGS_PLAYER_LOOPED_BUFFER`	Loop count (number of loops performed)	Current buffer address

The callbacks are generated from within the `sceNgsSystemUpdate()` call, post processing.

10 Parametric EQ DSP Effect Module Overview

The Parametric EQ DSP Module is a powerful time domain equalization tool. It consists of four filters arranged in a series. For example:

```
IN -> [FILT 0] -> [FILT 1] -> [FILT 2] -> [FILT 3] -> OUT
```

An example arrangement for the filters would be:

Filter Index	Mode	Frequency (Hz)	Resonance	Gain Range (dB)	Use
0	Low Shelf	80	N/A	+/- 18	Bass boost/cut
1	Peak	400	1	+/- 18	Low mid boost/cut
2	Peak	1500	1	+/- 18	High mid boost/cut
3	High Shelf	6500	N/A	+/- 18	Treble boost/cut

Please see Chapter 7, Filter DSP Effect Module Overview for more information on filter modes.

11 Pauser DSP Effect Module Overview

The Pauser DSP effect Module handles fading out/in the audio signal if the user pauses or resumes a Voice. Fading the signal out (if the user pauses a Voice) and back in again (if the user resumes a Voice) removes any unwanted audiable pops or clicks that may otherwise be heard if audio amplitude is suddenly silenced. The user can specify the number of samples that will be affected during both the fade out and fade in.

12 Pitch Shift DSP Effect Module Overview

The pitch shift DSP modifies the pitch of the input sound whilst maintaining its duration.

The pitch offset can be edited by setting the `fPitchOffsetInCents` parameter. This parameter represents the change in pitch defined in cents, where 1 semitone = 100 cents.

The range of modification is -1200 to +1200 cents, though at extreme values audible artefacts maybe more noticeable.

13 I3DL2 Reverb DSP Effect Module Overview

For an overview of the I3DL2 specification, please see the following document:

- *Interactive 3D Audio Rendering Guidelines*, Level 2.0

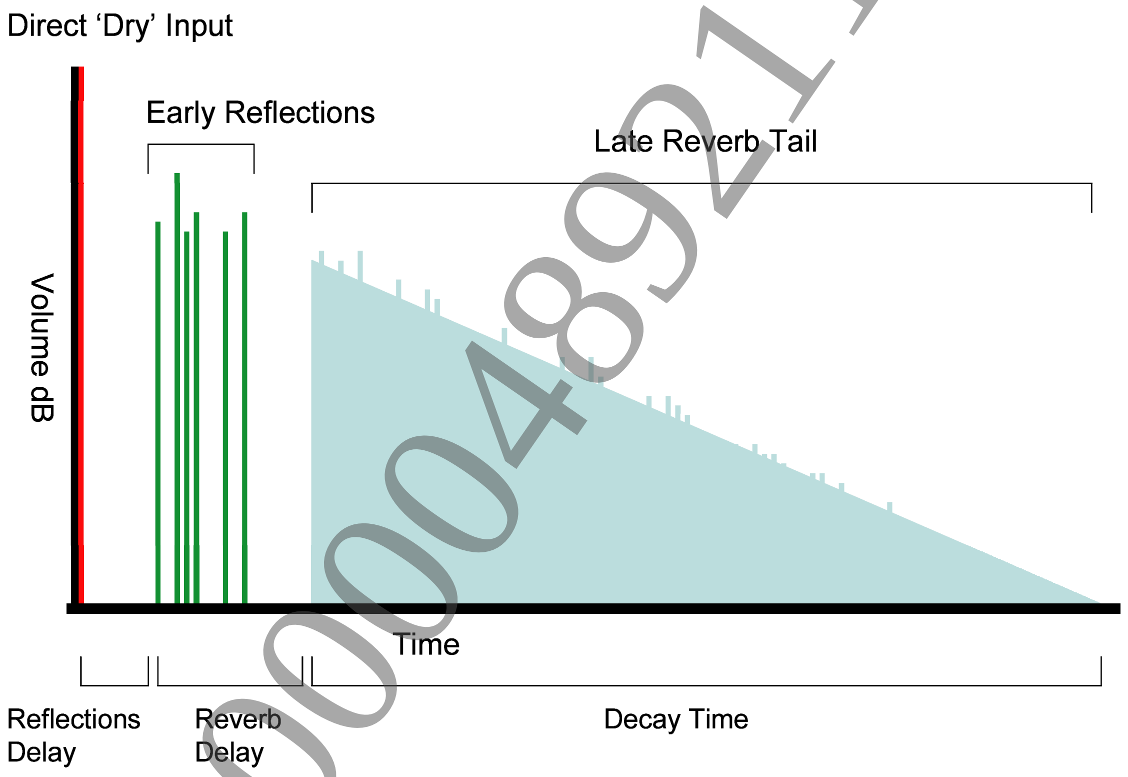
  (MIDI Manufacturers Association, last updated: September 20, 1999)
  http://www.iasig.org/pubs/3dl2v1a.pdf

Note that the link to this site was available at the given URL on October 16, 2012. It is possible, however, for the applicable pages to be moved or modified.

This section contains a brief overview and description of the modifications and additions to the standards regarding the NGS implementation.

Figure 15 shows an overview of the nature of the reverb and the relative timings, including Reflections Delay, Reverb Delay, and Decay Time parameters.

Figure 15 Overview of the Reverb



Within the I3DL2 specifications, volumes are expressed throughout in millibels (mB) – a millibel is equivalent to 1/100th of a decibel (dB). The volume controls for the early reflections (*`fReflections`*) and late reverb tail (*`fReverb`*) represent the attenuations of total output energy of the stage; relative to input energy, not including the effects of filtering. These volumes can be boosted above 0mB for either or both of the sections. The volume parameter *`fRoom`* acts as a main volume control, scaling the other output volumes.

There are two reverb parameter ID values available which can be used interchangeably. It is recommended you specify `SCE_NGS_REVERB_PARAMS_STRUCT_ID_V2` within the `desc.id` value for the parameter structure, instead of `SCE_NGS_REVERB_PARAMS_STRUCT_ID`, as this fixes some known issues. The older identifier (`SCE_NGS_REVERB_PARAMS_STRUCT_ID`) remains available for backward compatibility. For more information, see "(I3DL2) Reverb Structures" in the *NGS Modules Reference*.

Basic Structure of Reverb

The basic structure of the reverb is as follows (in order of processing):

- **High Pass Filter** – This is a two pole high pass filter controlled via the parameters `fHFReference` and `fRoomHF`. The `fHFReference` parameter sets the high pass reference frequency and `fRoomHF` sets the attenuation in millibels at the given reference frequency.
- **Low Pass Filter** – This is controlled via the parameters `fLFReference` and `fRoomLF`.
- **Pre-Delay** – This is the delay between the filtered input and the first of the early reflections, controlled via `fReflectionsDelay`.
- **Reflections** – The pattern, spacing, volume, and total time from the first reflection until the late reverberation can be controlled:
    - `eEarlyReflectionPattern` controls which reflections pattern is used. There are left and right variants for each pattern, designed to be used in conjunction with stereo/multichannel setups.
    - `fReverbDelay` controls the total delay time from the first reflections until the start of the late reverberation. This also controls the scale of the reflections as all the early reflections occur within this time. Setting a time of 0 seconds forces the reflections to superimpose to a single reflection.
    - `fEarlyReflectionScalar` scales the total time of the reflections pattern, thus a lower value 'bunches' the reflections closer together in time, forcing them towards the time of the first reflection. This parameter was added as an extension to the standard as the output of the early reflections stage feeds the late reverberation stage; therefore, modifying the early reflections will colourise the late reverb section.
    - `fReflections` controls the attenuation in millibels of the output energy of the early reflection. This volume is combined with the main volume parameter `fRoom`.
- **Late Reverb Stage A** – `fDecayTime` controls the length of time the late reverb decays in seconds, while the `fReverb` parameter controls the overall volume of the late reverb section. `fDecayHFRatio` controls the ratio of high to low frequency decay in the late reverb. `fDensity` controls the modal density of reflections in the late reverb.
- **Late Reverb Stage B** – The `fDiffusion` parameter controls the echo density in the late reverb. A low `fDiffusion` value 0% gives fewer and more discreet reflections in the late reverb. A large value 100% gives lots of dense reflections in the late reverb.

14 Signal Generator DSP Effect Module Overview

The Signal Generator Module allows for a continuous tone to be generated within a Voice. Using this Module allows such tones to be played without the need for the user to generate PCM data to play from the Player Module.

The Signal Generator Module generates a single tone. A tone contains the following:

Sine	Generate a sine wave signal.
Triangle	Generate a triangle wave signal.
Saw	Generate a saw tooth signal.
Noise	Generate a white noise signal.
Noise [PSP™ (PlayStation®Portable)]	Generate a white noise signal using the algorithm as used in the PSP™ (PlayStation®Portable) audio libraries.
Pulse	Generate a pulse waveform. A pulse with can also be set.

A phase value can also be specified per tone. The phase allows the setting of the pulse width.

15 Mixer DSP Effect Module Overview

The Mixer Module simply allows the signals from two modules to be mixed, and the resulting audio signal amplitude to be modified.

This should not be confused with the Input Mixer DSP Effect Module, see Chapter 8, Input Mixer DSP Effect Module Overview.

16 Output Module DSP Effect Overview

The output module allows the user to retrieve PCM data from NGS. This can then be passed to an output library (see the `sceAudioOut` library and `sceAudioOutOutput()` function).

The data is retrieved via the `sceNgsVoiceGetStateData()` function.

The size of the data in bytes can be calculated as follows:

number of channels * *granularity* * `sizeof(short)`

Where:

- *number of channels* represents the number of channels in the Voice containing the module.
- *granularity* is the NGS system granularity that the system was initialized with (see `SceNgsSystemInitParams` structure in the *NGS Reference*).

Each sample is a 16-bit (`short`) sample, and stereo data is interleaved per sample (left, right, left, right, etc).