Sound Data Formats

Publication date: October 2001

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052, Japan

Sony Computer Entertainment America
919 E. Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

Table of Contents

About This Manual

This is the Runtime Library Release 2.4 version of the *Sound Data Formats* manual.

It describes the common data formats, PlayStation 2 Sequence Data Format (SQ), PlayStation 2 Phoneme Data Format (HD / BD) and Software Synthesizer Waveform Data Format (SS)

Changes Since Last Release

None

Related Documentation

Note: the Developer Support Web site posts current developments regarding the Libraries and also provides notice of future documentation releases and upgrades.

Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
`courier`	Indicates literal program code.
italic	Indicates names of arguments and structure members (in structure/function definitions only).
medium bold	Indicates data types and structure/function names (in structure/function definitions only).
blue	Indicates a hyperlink.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In North America:	*In North America:*
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: PS2_Support@playstation.sony.com Web: http://www.devnet.scea.com/ Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
In Europe:	*In Europe:*
Attn: Production Coordinator Sony Computer Entertainment Europe 30 Golden Square London W1F 9LD, U.K. Tel: +44 (0) 20 7859-5000	E-mail: ps2_support@scee.net Web: https://www.ps2-pro.com/ Developer Support Hotline: +44 (0) 20 7859-5777 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT)

Common Data Formats

Data types

The following data types are defined in order to describe sound data formats.

Table 1

Data type	Meaning
U32	Unsigned 32-bit integer or a code consisting of 4 characters
S32	Signed 32-bit integer
U16	Unsigned 16-bit integer
S16	Signed 16-bit integer
U8	Unsigned 8-bit integer
S8	Signed 8-bit integer
FOFST	Unsigned 32-bit integer that represents an offset (in bytes) from the top of a file.
COFST	Unsigned 32-bit integer that represents an offset (in bytes) from the top of a chunk.
SOFST	Unsigned 32-bit integer that represents an offset (in bytes) from the top of a section.
POFST	Unsigned 32-bit integer that represents an offset (in bytes) from the current location.
CINDEX	Unsigned 16-bit integer that represents the index of a data array.
SINDEX	Unsigned 16-bit integer that represents the index of a data item in a section containing sequential, fixed-length data.
WADDR	Unsigned 32-bit fixed point number that represents a location (a sample) within waveform data. (Fractional part is 8 bits.)
SWADDR	Signed 32-bit fixed point number that represents a location (a sample) within waveform data. (Fractional part is 8 bits.)

WADDR and SWADDR are data types that represent locations within waveform data of individual samples.

Bits 7 to 0 are the fractional part, bits 30 to 8 are the real part, and for WADDR, bit 31 is always 0.

Bits 30 to 28 are set to 0 in the PlayStation 2.

Data consisting of multiple bytes is little-Endian.

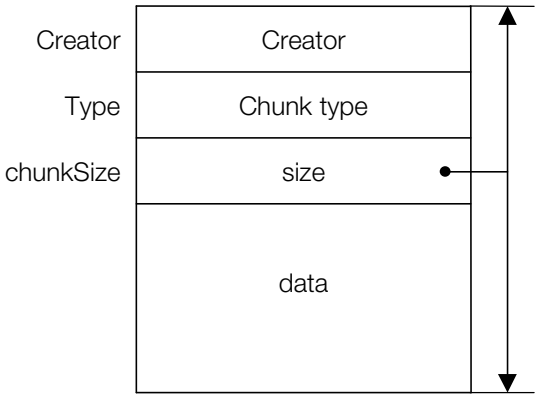
Chunk Structure

Sound data files used in the PlayStation 2 consist of a collection of data blocks called chunks. A chunk has a common data structure as shown below.

Table 2

Field	Type / size	Meaning
Creator	U32	4-character code that represents the data creator.
Type	U32	4-character code that represents the contents of the chunk.
chunkSize	U32	Data size of the chunk (in bytes)
Data	Not specified	Data

Figure 1: Structure of a chunk



PlayStation 2 Sequence Data Format (SQ)

Introduction

SQ is sequence data generated by JAM.

The sequence data of a musical composition is represented as a Midi chunk, the performance order of that sequence data is represented as a Song chunk, the sound effect sequence data for producing sound effects is represented as an SeSeq chunk, and the performance order of that sound effect sequence is represented as an SeSong chunk.

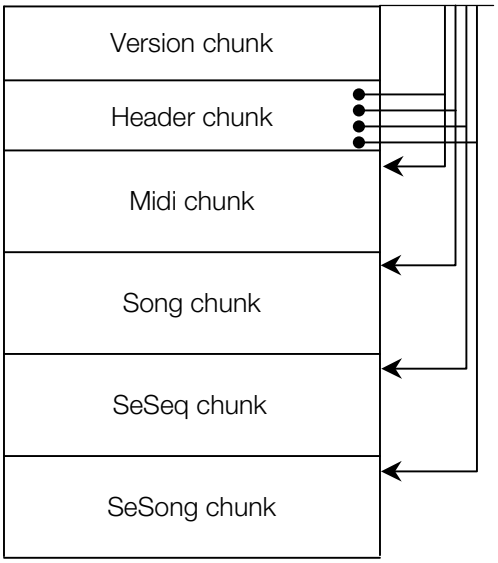
File Structure

An SQ format file consists of six chunks as shown below.

Table 3

Chunk	Size	Contents
Version chunk	16bytes	Version information
Header chunk	32bytes	Information on the entire file, and location of Midi chunks, Song chunks, SeSeq chunks and SeSong chunks.
Midi chunk	Not specified	MIDI sequence data (includes compression table)
Song chunk	Not specified	Data including playback order
SeSeq chunk	Not specified	Sound effect sequence data
SeSong chunk	Not specified	Data including sound effect playback order

Figure 2: Structure of an SQ file



Version Chunk

A version chunk represents the version of a data format. The contents and structure are as follows.

Table 4

Field	Type	Size	Meaning
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Vers
chunkSize	U32	4bytes	Chunk size (=16)
reserved	U16	2bytes	(reserved area)
versionMajor	U8	1byte	Major version
versionMinor	U8	1byte	Minor version

SQ data is currently at versions 1.0 and 2.0.

In version 2.0 an SeSeq chunk and an SeSong chunk were added. Accordingly, the contents of the header chunk have also changed.

Header Chunk

A header chunk represents the structure of the entire file. The contents and structure are as follows.

Table 5

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code **SCEI**
Type	U32	4bytes	Chunk type **Sequ**
chunkSize	U32	4bytes	Chunk size (=**32**)
fileSize	U32	4bytes	Size of entire file (in bytes)
songChunkAddr	FOFST	4bytes	Location of Song chunk (Number of bytes from start of file)
midiChunkAddr	FOFST	4bytes	Location of Midi chunk (Number of bytes from start of file)
seSequenceChunkAddr	FOFST	4bytes	Location of seSeq chunk (Number of bytes from start of file)
seSongChunkAddr	FOFST	4bytes	Location of seSong chunk (Number of bytes from start of file)

Midi Chunk

A Midi chunk keeps MIDI sequence data converted by defined SMF conventions (standard MIDI format).

Structure of Midi Chunk

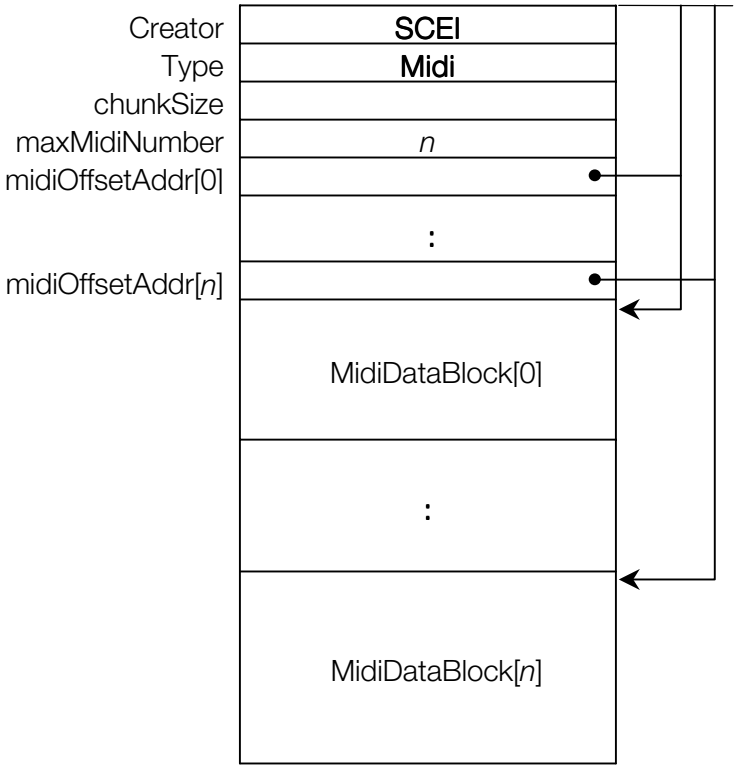
The contents and structure of a Midi chunk are as follows.

Table 6

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code **SCEI**
Type	U32	4bytes	Chunk type **Midi**
chunkSize	U32	4bytes	Chunk size
maxMidiNumber	U32	4bytes	Maximum value (*n*) of Midi data block
midiOffsetAddr[0]	COFST	4bytes	Location of Midi data block 0 (Number of bytes from start of Midi chunk)
---	---	---	---
midiOffsetAddr[n]	COFST	4bytes	Location of Midi data block *n* (Number of bytes from start of Midi chunk)
MidiDataBlock[0]	-	Not specified	Midi data block 0
---	---	---	---
MidiDataBlock[n]	-	Not specified	Midi data block *n*

A Midi data block number need not refer to all values being used from 0 up to maxMidiNumber. When Midi data block number *i* is not being used (no applicable Midi data block exists), the midiOffsetAddr[i] will have the constant value NO_COFSI (=-1) and the MidiDataBlock[i] will be omitted.

Figure 3: Structure of a Midi chunk



Midi Data Block

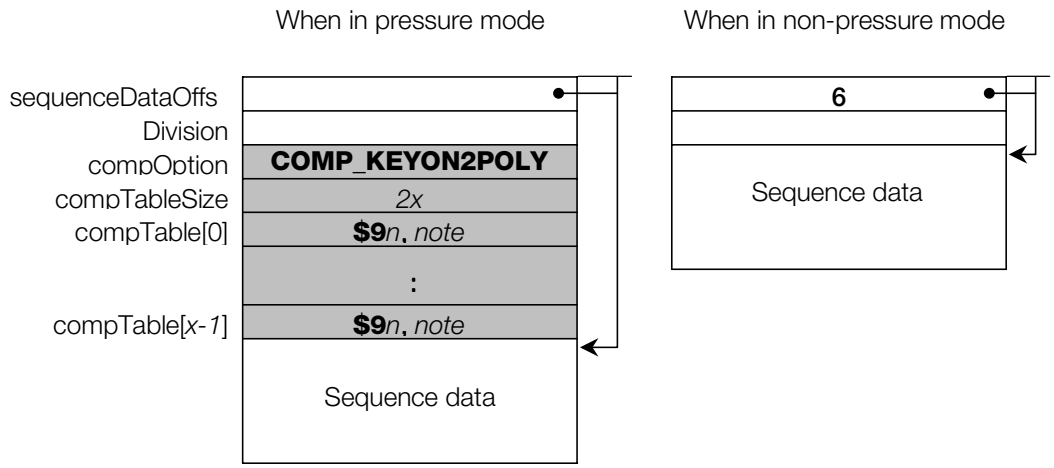
Each Midi data block is provided with sequence data converted from SMF and a compression table when compression mode is applied. The contents and structure are as follows.

Table 7

Field	Type	Size	Contents
SequenceOffset	POFST	4bytes	Location of sequence data (Number of bytes from start of the MidiDataBlock)
Division	U16	2bytes	Division function for 4 musical notes
compOption *	U16	2bytes	Compression modeCOMP_KEYON2POLY(0x0001) : Replaces and compresses Key On with Polyphonic Key Pressure.
compTableSize : *	U16	2bytes	Size of compression table (2x) Maximum 2 x 128 bytes
compTable *	U8 X 2 X x	2x bytes	Compression table (Note ON message that was compressed)
	U8 X not specified	U8 X not specified	Sequence data

compOption / compTableSize / compTable indicated by the * symbol do not exist when no compression mode is applied. This case can be distinguished because the value of sequenceOffset is 6.

Figure 4: Structure of Midi Data Block



Compression Table

Compression mode collects the 3 bytes of a Note On message, which occur often within sequence data, into a table and replaces them with code of a Polyphonic Key Pressure message. In addition, compression mode also reduces velocity to 16 levels, represents this value in 2 bytes, then uses that data to reduce the size of the entire sequence data.

Compression is performed as follows.

1. The frequency at which Note On messages ($9n,note,velocity) occurs is counted.
2. Groups of $9n,note up to the upper 128th group are held in a table.
   In other words, groups of $9n, note at the [ith] place (the occurrence frequency sequence) are placed in compTable[i].
3. Velocity is divided by 8 and becomes v (any remainder is discarded).
4. The Note On messages ($9n,note,velocity) are replaced by two bytes of $As,$tv.
   s, t are values which equal $0t X $10 + $0s = i (above occurrence frequency sequence).

In contrast, compressed sequence data is normal sequence data in which $As,$tv are replaced with compTable[t * 16 + s], v * 8.

Conversion Conventions From SMF To Sequence Data

Sequence data in a Midi data block is converted to SMF (standard MIDI format) according to the following conventions.

- The running status of channel messages is omitted.
- Bit 7 of previous data byte 1 is set to 1 and delta time is omitted when delta time is 0.
- All 3rd bytes of Note Off messages ($8n,$mm,$vv) are omitted.
- The 3rd byte in a Note ON message ($9n,$mm,$vv) is converted to a Note Off message when the 3rd byte is 0. If the options (Omit Polyphonic Key Pressure and Compress the Note-on event using Polyphonic Key Pressure) are set during data creation, compression will be performed using the above-mentioned compression table.

- The Polyphonic Key Pressure message ($An,$mm,$vv) is omitted if the option (Omit Polyphonic Key Pressure) is set during data creation.
- Control change messages ($Bn,cc,$vv) correspond to the following.

Table 8

cc	Function
#00	Bank Select(MSB)
#32	Bank Select(LSB)
#01	Pitch Modulation Depth
#02	Amp Modulation Depth
#07	Channel Volume
#10	Pan
#11	Expression
#64	Damper Pedal
#05	Portament Time
#65	Portament On/Off
#84	Portament Controll
#06	NRPN Data Entry(MSB)
#38	NRPN Data Entry(LSB)
#96	NRPN Increment
#97	NRPN Decrement
#99	NRPN(MSB)
#98	NRPN(LSB)

NRPN can be used with the following.

Table 9

NRPN message	Function
$B0,$63,$00	Loop start setting: Loop start command
$B0,$06,$nn	Loop start number
$B0,$63,$01	Loop end setting: Loop end command
$B0,$06,$nn	Loop end number
$B0,$26,$mm	Repeat frequency (0 = endless loop)
$B0,$63,$02	Reverb type setting: Reverb type command
$B0,$62,$00	Core setting: Core0
$B0,$62,$10	Core setting: Core1
$B0,$06,$00	Reverb Off
$B0,$06,$01	Room
$B0,$06,$02	StudioA
$B0,$06,$03	StudioB
$B0,$06,$04	StudioC
$B0,$06,$05	Hall
$B0,$06,$06	Space
$B0,$06,$07	Echo
$B0,$06,$08	Delay

NRPN message	Function
$B0,$06,$09	Pipe
$B0,$63,$03	Reverb depth setting: Reverb depth command
$B0,$62,$00	Reverb depth positive phase (Core0)
$B0,$62,$01	Reverb depth negative phase (Core0)
$B0,$62,$02	Reverb delay (Core0)
$B0,$62,$03	Reverb feed (Core0)
$B0,$62,$10	Reverb depth positive phase (Core1)
$B0,$62,$11	Reverb depth negative phase (Core1)
$B0,$62,$12	Reverb delay (Core1)
$B0,$62,$13	Reverb feed (Core1)
$B0,$06,$nn	Parameter ($00-$7F)

The NPRN parameter settings must set required control changes from among #99($63), #98($62), #06($06), #38($26) in this order in a different delta time. Loops are set by sequence and cannot be set by tracks. If a loop is set to any track, the loop will play back.

- Channel Key Pressure messages ($Dn,$mm) are all omitted if the option (Omit Channel Pressure) is set during data creation.
- Meta events are equivalent to the following.

Table 10

Meta event	Function
$FF,$2F,$00	End of Track
$FF,$51,$03,$tt,$tt,$tt	Set Tempo

Song Chunk

A song chunk keeps a song table that specifies settings such as the playback order of sequence data that is included in a MIDI chunk.

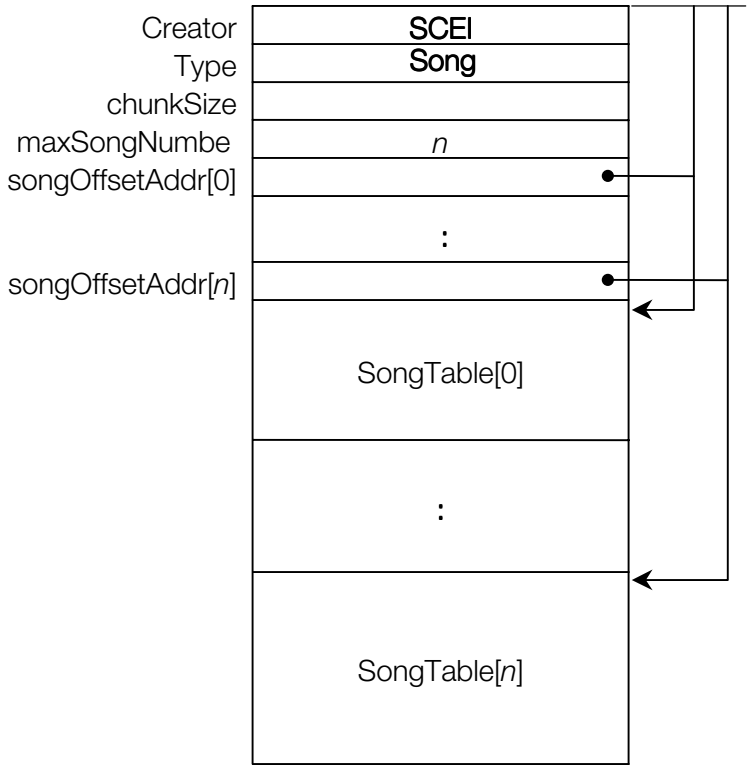
The contents and structure of a song chunk are shown below.

Table 11

Field	Type	Size	Meaning
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type song
chunkSize	U32	4bytes	Chunk size
maxSongNumber	U32	4bytes	Maximum value (*n*) of Song table number
songOffsetAddr[0]	COFST	4bytes	Data block location of song table 0 (Number of bytes from the beginning of the Song chunk)
---	---	---	---
songOffsetAddr[n]	COFST	4bytes	Data block location of song table *n* (Number of bytes from start of the Song chunk)

Field	Type	Size	Meaning
SongTable[0]	-	Not specified	Data block location of Song table 0
---	---	---	---
SongTable[n]	-	Not specified	Data block location of Song table *n*

Figure 5: Structure of Song chunk



A song table is a command string arranged as follows.

Table 12

Command	Function
$A0,$00,$nn	Song Play($nn = Midi data block number in Midi chunk)
$A0,$01,$nn	Song Volume($*nn* = 0~128)
$A0,$02,$nn	Song Volume +
$A0,$03,$nn	Song Volume -
$A0,$04,$nn	Song Pan (0~64~128?left~center~right, minus = negative phase)
$A0,$05,$nn	Song Pan +
$A0,$06,$nn	Song Pan -
$A0,$07,$nn	Set Fade Time (units of 0.1 sec.)(0~255)
$A0,$08,$nn	Song Fade($*nn* = volume after fade)(0~128)
$A0,$11,$nn	Song Repeat($*nn* = return count, 0 = endless loop)
$A1,$nn,$mm	Return destination for Song Repeat
	$*nnmm* = Song table offset (from the start of each song table)
$A0,$21,$nn	Set Tempo(10-255)
$A0,$22,$nn	Tempo +
$A0,$23,$nn	Tempo -
$A0,$7F,$00	Song All Clear (Stops BGM during playback / Stops release also)
$A0,$7F,$01	Song Clear (Stops BGM during playback / Release unchanged)
$A0,$7F,$7F	Song End

SeSeq Chunk

An SeSeq chunk keeps sequence data for sound effects. Commands which represent SeSeq have been provided independently by SCE.

SeSeq Chunk Structure

An SeSeq chunk can have up to 128 SeSeq sets. The contents and structure of an SeSeq chunk are shown below.

Table 13

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code **SCEI**
Type	U32	4bytes	chunk type **Sesq**
chunkSize	U32	4bytes	chunk size
maxSeSequenceSetNumber	U32	4bytes	Maximum value (*n*) of SeSeq set number
tableOffset	POFST	4bytes	Offset to SeSeq set offset table
seSequenceMasterVolume	U8	1byte	SeSeq master volume

Field	Type	Size	Contents
seSequenceMasterPanpot	S8	1byte	SeSeq master panpot (0~64~127 = left~center~right, minus = negative phase)
seSequenceMasterTimeScale	U16	2bytes	SeSeq master time scale (normal speed = 1000, slowest = 1, fastest = 65535)
reserved	U32	4bytes	reserved area
reserved	U32	4bytes	reserved area
seSequenceSetOffsetAddr[0]	COFST	4bytes	Location of SeSeq set 0 (Number of bytes from start of the SeSeq chunk)
...	...	...	...
seSequenceSetOffsetAddr[n]	COFST	4bytes	Location of SeSeq set *n* (Number of bytes from start of the SeSeq chunk)

SeSeq Set Structure

An SeSeq set can have up to 128 SeSeq's. The contents and structure of an SeSeq set are shown below.

Table 14

Field	Type	Size	Contents
maxSeSequenceNumber	U32	4bytes	Maximum value (*n*) of SeSeq number that belongs to the SeSeq set
tableOffset	POFST	4bytes	Offset to SeSeq table
seSequenceSetVolume	U8	1byte	SeSeq set volume
seSequenceSetPanpot	S8	1byte	SeSeq set panpot (0~64~127 = left~center~right, minus = negative phase)
seSequenceSetTimeScale	U16	2bytes	Time scale of this SeSeq (normal speed = 1000, slowest = 1, fastest = 65535)
reserved	U32	4bytes	reserved area
seSequenceOffsetAddr[0]	COFST	4bytes	Location of SeSeq [0] (Number of bytes from start of the SeSeq chunk)
...	...	...	...
seSequenceOffsetAddr[m]	COFST	4bytes	Location of SeSeq [*m*] (Number of bytes from start of the SeSeq chunk)

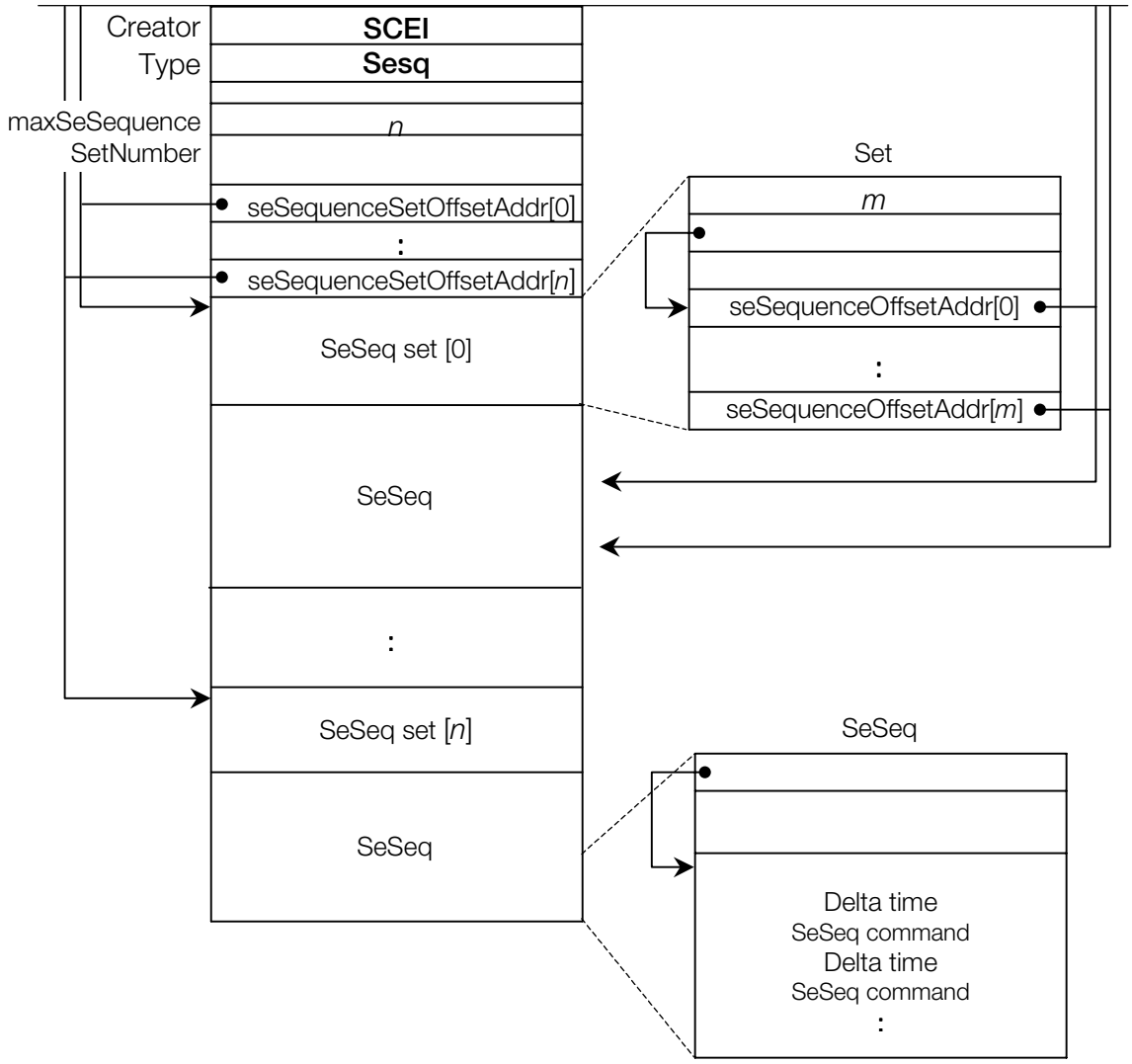
SeSeq Structure

The contents and structure of SeSeq are as follows.

Table 15

Field	Model	Size	Contents
seSequenceDataOffset	POFST	4bytes	Offset to SeSeq data
seSequenceVolume	U8	1byte	SeSeq volume
seSequencePanpot	S8	1byte	SeSeq panpot (0~64~127 = left~center~right, minus = negative phase)
seSequenceTimeScale	U16	2bytes	SeSeq time scale (normal speed = 1000, slowest = 1, fastest = 65535)
seSequenceDataSize	U32	4bytes	Data size of SeSeq (number of bytes)
reserved	U32	4bytes	reserved area
...	...	...	Delta time and SeSeq commands are combined, followed by an arbitrary number. These are terminated with a sequence end command.

Figure 6: Structure of SeSeq chunk



SeSeq Commands

Commands shown in the following table are defined as commands used to represent SeSeq's.

Table 16

Sound effect sequence command	Function
$An,$tt,$nn,$vv	Note On
$An,$tt,$nn,$00	Note Off
$Bn,$tt,$nn,$07,$time,$Volume	Time-Volume: Changes the volume from the present value to $Volume, over the time specified in $time.
$Bn,$tt,$nn,$0C,$time,$Panpot	Time-Panpot-Clockwise (during the specified interval, the current panpot is varied clockwise to the specified panpot) Panpot varies from 0 to 64 to 127, corresponding to a change from left to center to right. A minus sign indicates inverse phase.
$Bn,$tt,$nn,$0D,$time,$Panpot	Time-Panpot-Counterclockwise (during the specified interval, the current panpot is varied counterclockwise to the specified panpot) Panpot varies from 0 to 64 to 127, corresponding to a change from left to center to right. A minus sign indicates inverse phase.
$Bn,$tt,$nn,$0E,$time, $PitchMSB,$PitchLSB	Time-Pitch+: Raises the pitch by the specified value only, over the time specified in $time (units in cents).
$Bn,$tt,$nn,$0F,$time, $PitchMSB,$PitchLSB	Time-Pitch-: Lowers the pitch by the specified value only, over the time specified in $time (units in cents).
$Bn,$tt,$nn,$10, $DepthMSB,$DepthLSB	PitchLFO amplitude+: Raises the amplitude of the pitch LFO by the specified value only (units in 1/128 half steps)
$Bn,$tt,$nn,$11, $DepthMSB,$DepthLSB	PitchLFO amplitude-: Lowers the amplitude of the pitch LFO by the specified value only (units in 1/128 half steps)
$Bn,$tt,$nn,$12,$Cycle	PitchLFO period
$Bn,$tt,$nn,$20,$Depth	AmpLFO amplitude+: Raises the amplitude of the amp LFO by the specified value only.
$Bn,$tt,$nn,$21,$Depth	AmpLFO amplitude-: Lowers the amplitude of the amp LFO by the specified value only.
$Bn,$tt,$nn,$22,$Cycle	AmpLFO period
$FF	Sequence end

Table 17

Parameter	Meaning
n	Sound effect timbre set number ($0~$F) (refer to HD/BD)
tt	Sound effect timbre number ($00~$7F) (refer to HD/BD)
nn	Sound effect ten note number ($00~$7F) (refer to HD/BD)
vv	Velocity ($01~$7F)
time	Variable length delta time similar to SMF (units in msec)
cycle	Period using a variable length declaration similar to SMF (units in msec)

SeSong Chunk

An SeSong chunk keeps playback order data of a sound effect sequence.

The contents and structure have not yet been determined.

PlayStation 2 Phoneme Data Format (HD / BD)

Introduction

HD/BD is a phoneme data (WaveTable) format created with JAM and used in the CSL hardware synthesizer (modhsyn).

This format allows you to save many different parameters such as cross fades smoothly linked between splits, velocity cross fades smoothly linked between velocity ranges, pitch modulation and amp modulation. Parameters for generating sound effect sequence data are also saved in HD / BD format.

File Structure

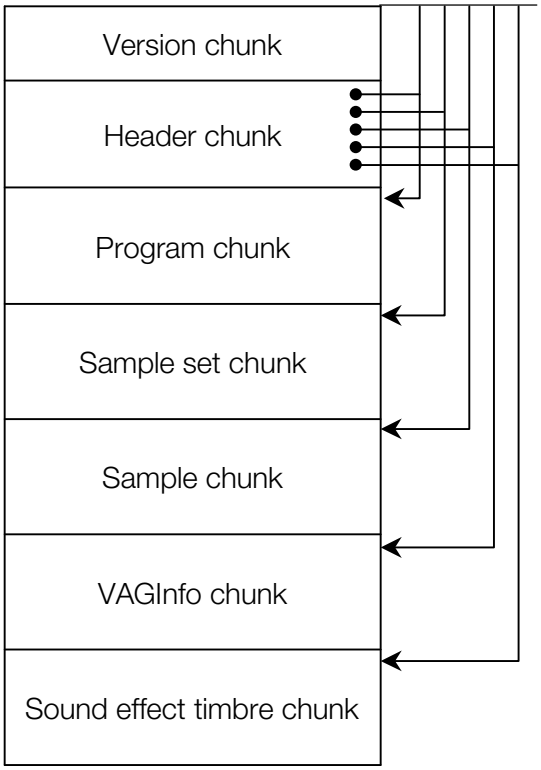
PlayStation 2 phoneme data consists of two files, a header file (hd) and a body file (bd). Both files have identical filenames and are distinguished by the suffixes ".hd" and ".bd" attached to the end of the filename.

A header file is a data file that defines the relationship between program numbers, note numbers, velocities and waveform data and consists of the following six chunks.

Table 18

Chunk	Size	Contents
Version chunk	16bytes	Header version information
Header chunk	64bytes	Information on the entire file, and location of each chunk under program chunk
Program chunk	Not specified	Information defining the relationship between tone / sound range and a sample set.
Sample set chunk	Not specified	Information defining the relationship between velocity range and sample groups.
Sample chunk	Not specified	Information defining the relationship between sample number and VAGinfo / envelope.
VAGInfo chunk	Not specified	Pointer to waveform data (body file)
Sound effect timbre chunk	Not specified	Sound effect data

Figure 7: Structure of header file (hd)



The structure of a body file will be described later.

Version Chunk

A version chunk represents the version of a data format. The contents and structure are as follows.

Table 19

Field	Type	Size	Meaning
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Vers
chunkSize	U32	4bytes	Chunk size(=16)
reserved	U16	2bytes	(reserved area)
versionMajor	U8	1byte	Major version
versionMinor	U8	1byte	Minor version

The HD / BD format is currently at versions 1.1 and 2.0.

In version 2.0 a sound effect timbre chunk was added. Accordingly, the contents of the header chunk have also changed.

Header Chunk

A header chunk represents the structure of the entire file. The contents and structure are as follows.

Table 20

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Head
chunkSize	U32	4bytes	Chunk size(=64)
headerSize	U32	4bytes	Size of entire header file(in bytes)
bodySize	U32	4bytes	Size of entire body file(in bytes)
programChunkAddr	FOFST	4bytes	Location of program chunk (Number of bytes from start of file)
samplesetChunkAddr	FOFST	4bytes	Location of sample set chunk (Number of bytes from start of file)
sampleChunkAddr	FOFST	4bytes	Location of sample chunk (Number of bytes from start of file)
vagInfoChunkAddr	FOFST	4bytes	Location of VAGinfo chunk (Number of bytes from start of file)
seTimbreChunkAddr	FOFST	4bytes	Location of sound effect timbre chunk (Number of bytes from start of file)
reserved	-	24bytes	(reserved area)

Program Chunk

A program chunk is a data chunk that indicates which sample set to use for a program number and a split (register: range of a note number). A program chunk also keeps parameters such as volume, panpot and LFO that apply to each program.

Complete Structure of Program Chunk

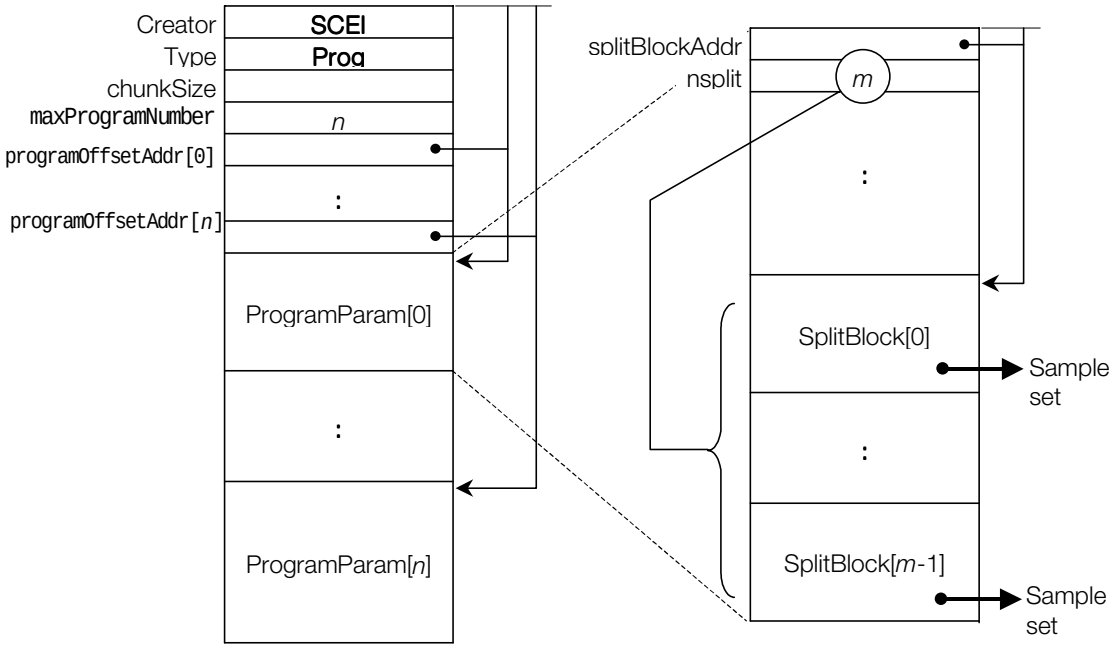
The contents and structure of a program chunk are shown below.

Table 21

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Prog
chunkSize	U32	4bytes	Chunk size
maxProgramNumber	U32	4bytes	Maximum value (*n*) of program number
programOffsetAddr[0]	COFST	4bytes	Location of parameter block of program 0 (Number of bytes from start of file)
...	...	...	...
programOffsetAddr[*n*]	COFST	4bytes	Location of parameter block of program n (Number of bytes from start of file)
ProgramParamBlock[0]	-	Not specified	Parameter block of program 0
	...	...	...
ProgramParamBlock[*n*]	-	Not specified	Parameter block of program *n*

A program number need not use all values from 0 up to maxProgramNumber. When program number i is not being used, the programOffsetAddr[i] will be the constant NO_COFST (=-1) and the ProgramParamBlock [i] will be omitted.

Figure 8: Structure of program chunk



Program Parameter Block

The contents and structure of a ProgramParamBlock are as follows.

Table 22

Field	Type	Size	Contents
splitBlockAddr	POFST	4bytes	Location of SplitBlock[0] (Number of offset bytes from this location)
nSplit	U8	1byte	Number (*m*) of SplitBlocks belonging to this program
sizeSplitBlock	U8	1byte	Size of one SplitBlock(=20)
progVolume	U8	1byte	Program volume(0~128)
progPanpot	S8	1byte	Program panpot (0~64~127~=left~center~right, minus = negative phase)
progTranspose	S8	1byte	Transpose (units in notes)
progDetune	S8	1byte	Detune (note 1)
keyFollowPan	S8	1byte	Change width of panpot that changes for each octave
keyFollowPanCenter	U8	1byte	keyFollowPan yields a result of 0 Note number (0~127)
progAttr	U8	1byte	Program attribute ROUND_PAN(0x01) : Allows panpot negative phase
reserved	U8	1byte	(reserved area)
progLfoWave	U8	1byte	LFO waveform(pitch) (note 2)
progLfoWave2	U8	1byte	LFO waveform(amp) (note2)
progLfoStartPhase	U8	1byte	LFO start phase (pitch) (note 3)
progLfoStartPhase2	U8	1byte	LFO start phase (amp) (note 3)
progLfoPhaseRandom	U8	1byte	LFO start phase random (pitch)
progLfoPhaseRandom2	U8	1byte	LFO start phase random (amp)
progLfoCycle	U16	2bytes	LFO period (pitch) units in msec
progLfoCycle2	U16	2bytes	LFO period (amp) units in msec
progLfoPitchDepth	S16	2bytes	Pitch modulation amplitude+ (note 1)
progLfoPitchDepth2	S16	2bytes	Pitch modulation amplitude- (note 1)
progLfoMidiPitchDepth	S16	2bytes	MIDI pitch modulation maximum amplitude+ (note 1)
progLfoMidiPitchDepth2	S16	2bytes	MIDI pitch modulation maximum amplitude- (note 1)
progLfoAmpDepth	S8	1byte	Amp modulation amplitude+
progLfoAmpDepth2	S8	1byte	Amp modulation amplitude-
progLfoMidiAmpDepth	S8	1byte	MIDI modulation maximum amplitude+
progLfoMidiAmpDepth2	S8	1byte	MIDI modulation maximum amplitude-

Field	Type	Size	Contents
SplitBlock[0]	(note4)	20bytes	Split parameter of split 0 Block
...	...	...	...
SplitBlock[*m*]	(note4)	20bytes	Split parameter block of split *m*

Note 1: Detune and pitch modulation use units of 128 equal half steps.

Note 2: LFO waveforms are specified with the following constants.

Table 23

Macro constant	Value	Waveform
LFO_WAVEFORM_NON	0	No LFO
LFO_WAVEFORM_SAWUP	1	Rising sawtooth wave
LFO_WAVEFORM_SAWDOWN	2	Falling sawtooth wave
LFO_WAVEFORM_TRIANGLE	3	Triangle wave
LFO_WAVEFORM_SQUEARE	4	Rectangular wave
LFO_WAVEFORM_NOISE	5	Noise
LFO_WAVEFORM_SIN	6	Sine wave
LFO_WAVEFORM_USER	0x80	User defined (not supported)

Note 3: Values that can be specified are from 0 ~ 255 corresponding to 0 ~ 359 degrees.

Note 4: sceHardSynthSplitBlock

Split Parameter Block

A split parameter block (sceHardSynthSplitBlock) is a data structure that expresses the difference in waveform between high and low registers, even with the same musical instrument. In addition, a split parameter block is provided with data that represents the register (note number range) and the sample set number used for that register as well as other sound properties. The contents and structure are as follows.

Table 24

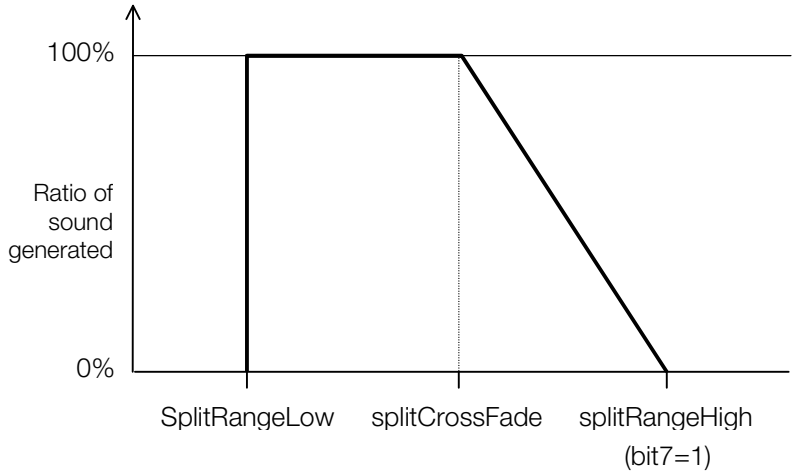
Field	Type	Size	Meaning
sampleSetIndex	CINDEX	2bytes	Sample set number (information for linking to a sample set chunk)
splitRangeLow	U8	1byte	Lower limit note number of this split (0-127)
splitCrossFade	U8	1byte	Cross fade start note number (0-127)
splitRangeHigh	U8	1byte	Upper limit note number of this split (0-127)
splitNumber	U8	1byte	Split number (0-127)
splitBendRangeLow	U16	2bytes	Bend range- (note1)
SplitBendRangeHigh	U16	2bytes	Bend range+ (note1)
keyFollowPitch	S8	1byte	Change width of pitch LFO that changes for each octave (note 1)
keyFollowPitchCenter	U8	1byte	keyFollowPitch yields a result of 0 Note number (0-127)
keyFollowAmp	S8	1byte	Change width of amp LFO that changes for each octave
keyFollowAmpCenter	U8	1byte	keyFollowAmp yields a result of 0 Note number (0-127)
keyFollowPan	S8	1byte	Change width of panpot that changes for each octave
keyFollowPanCenter	U8	1byte	keyFollowPan yields a result of 0 Note number (0-127)
splitVolume	U8	1byte	Split volume (0-128)
splitPanpot	S8	1byte	Split panpot (0~64-127=left-center-right, minus = negative phase)
splitTranspose	S8	1byte	Transpose (units in notes)
splitDetune	S8	1byte	Detune (units = 128 equal half steps) (note 1) Bend range and keyFollowPitch use units of 128 equal half steps.

(Note 1) Bend range and keyFollowPitch use units of 128 equal half steps.

Cross Fade

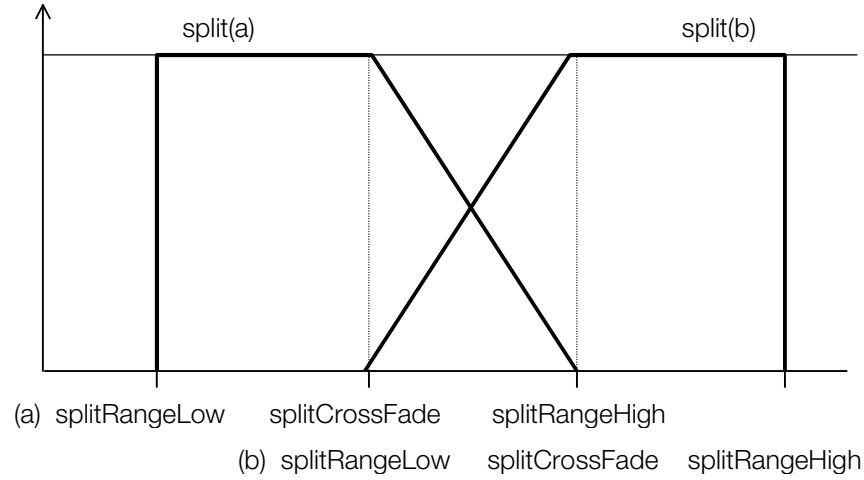
Specifying a cross fade means to smooth the boundaries of a register when bit 7 of splitRangeLow or splitRangeHigh is set to 1. For example, as shown in the figure below, when bit 7 of splitRangeHigh is set to 1, the sound generated by the split range from splitCrossFade up to splitRangeHigh will gradually decrease.

Figure 9: Specifying a cross fade



Normally, as shown in the figure, two splits are set such that the cross fade portion overlaps.

Figure 10



Sample Set Chunk

A sample set chunk is data that receives a link from a program chunk and indicates which sample will be sounded that corresponds to a velocity range. Sample set chunks are used to represent different waveforms that are generated according to the strength of a key touch, even with the same musical instrument in the same register.

Complete Structure of Sample Set Chunk

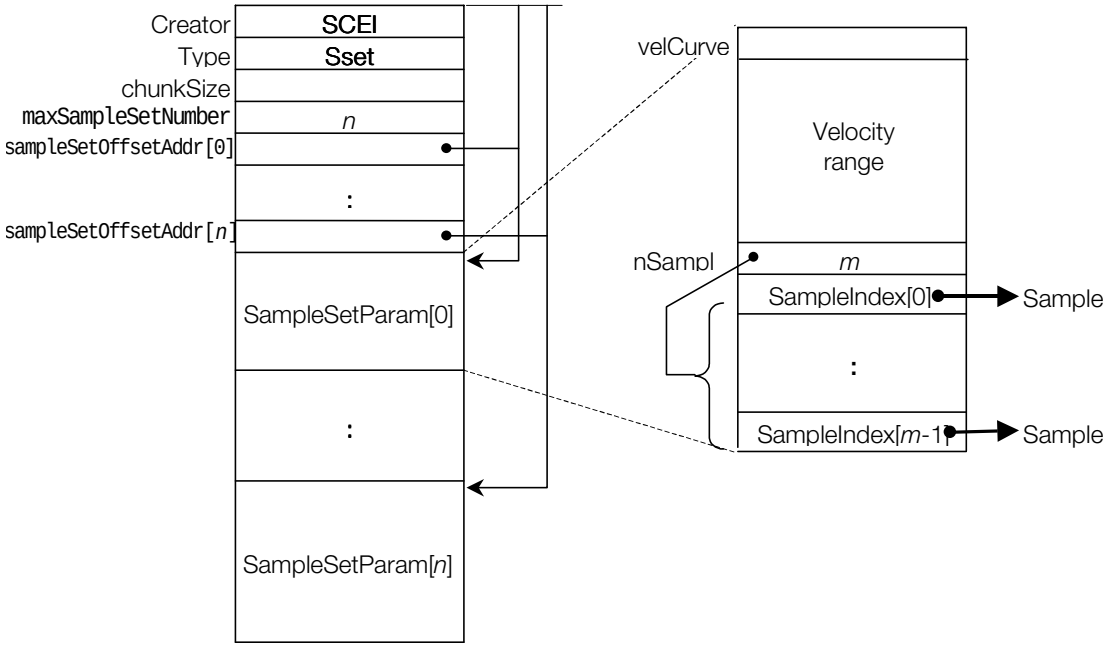
The contents and structure of a sample set chunk are shown below.

Table 25

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Sset
chunkSize	U32	4bytes	Chunk size
maxSampleSetNumber	U32	4bytes	Maximum value (*n*) of sample set number
sampleSetOffsetAddr[0]	COFST	4bytes	Parameter of sample set 0 Location of block (Number of bytes from start of chunk)
...	...	...	...
sampleSetOffsetAddr[*n*]	COFST	4bytes	Location of parameter block of sample set *n* (Number of bytes from start of chunk)
sampleSetParam[0]	*	Not specified	Parameter block of sample set 0
	...	...	...
sampleSetParam[*n*]	*	Not specified	Parameter block of sample set *n*

* sceHardSynthSampleSetParam

Figure 11: Structure of sample set chunk



Sample Set Parameter Block

The contents and structure of a sample set parameter block (sceHardSynthSampleSetParam) are as follows.

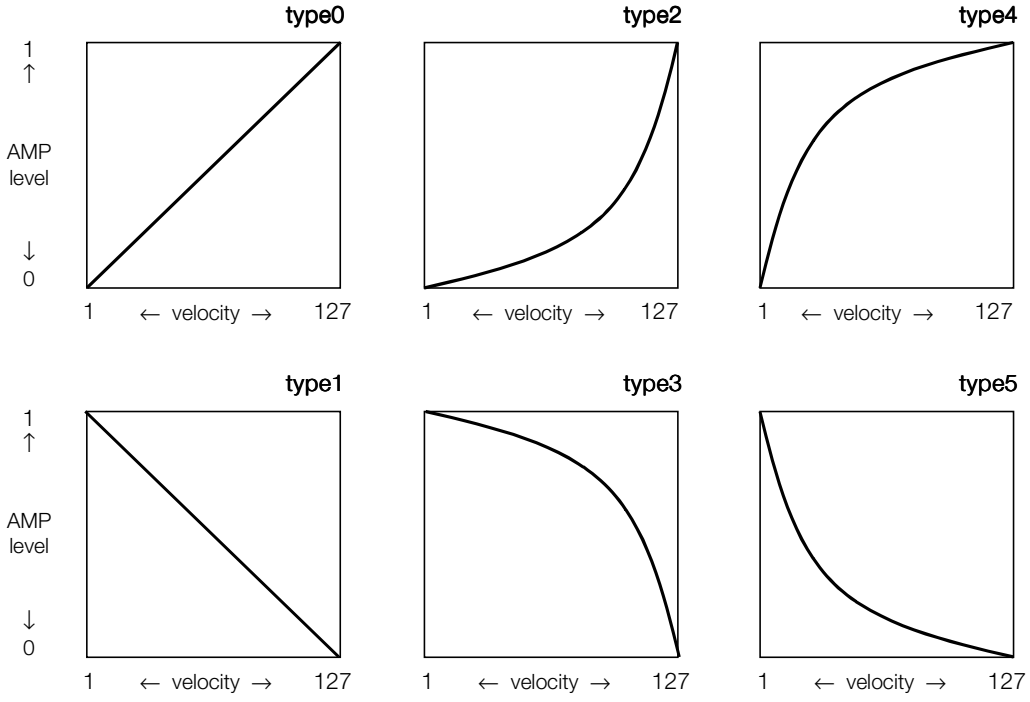
Table 26

Field	Type	Size	Contents
velCurve	U8	1byte	Velocity curve type
velLimitLow	U8	1byte	Generate sound with this sample set. Lower limit value of velocity
velLimitHigh	U8	1byte	Generate sound with this sample set. Upper limit value of velocity
nSample	U8	1byte	Number (*m*) of samples belonging to this program
sampleIndex[0]	CINDEX	2bytes	Sample number (information for linking to a sample chunk)
...	...	...	...
sampleIndex[*m*-1]	CINDEX	2bytes	Sample number (information for linking to a sample chunk)

Velocity Curve Type

The velCurve field of the sample set parameter block is a velocity curve used in the sample set. It represents the method of converting from a velocity value to an AMP level.

Figure 12: Velocity curve type



Sample Chunk

A sample chunk is data that receives a link from a sample set chunk and indicates which waveform data will be used that corresponds to a velocity range. A sample chunk also contains information such as envelope information.

Complete Structure of Sample Set Chunk

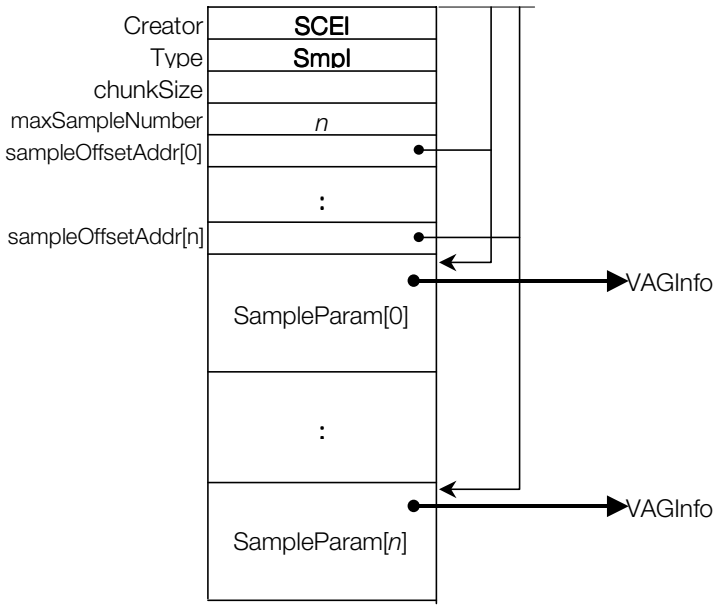
The complete contents and structure of a sample chunk are shown below.

Table 27

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Smpl
chunkSize	U32	4bytes	Chunk size
maxSampleNumber	U32	4bytes	Maximum value (n) of sample number
sampleOffsetAddr[0]	COFST	4bytes	Location of parameter block of sample 0 (Number of bytes from start of chunk)
...	...	...	...
sampleOffsetAddr[n]	COFST	4bytes	Location of parameter block of sample n (Number of bytes from start of chunk)
SampleParam[0]	*	42bytes	Parameter block of sample 0
	...	...	...
SampleParam[n]	*	42bytes	Parameter block of sample n

* sceHardSynthSampleParam

Figure 13: Structure of sample chunk



Sample Parameter Block

The contents and structure of a sample parameter block (sceHardSynthSampleParam) are shown below.

Table 28

Field	Type	Size	Contents
VagIndex	CINDEX	2bytes	VAG Info number (information for linking to a VAGInfo chunk)(NO_CINDEX = noise)
velRangeLow	U8	1byte	Sound generation lower limit velocity(0~127)
velCrossFade	U8	1byte	Cross fade start velocity(0~127)
velRangeHigh	U8	1byte	Sound generation upper limit velocity(0~127)
velFollowPitch	S8	1byte	Change width of pitch LFO using velocity (0~127) (note 1)
velFollowPitchCenter	U8	1byte	Velocity value for which velFollowPitch yields a result of (0~127)
velFollowPitchVelCurve	U8	1byte	Velocity curve type for pitch LFO
velFollowAmp	S8	1byte	Change width of amp LFO using velocity
velFollowAmpCenter	U8	1byte	Velocity value for which velFollowAmp yields a result of (0~127)
velFollowAmpVelCurve	U8	1byte	Velocity curve type for amp LFO
sampleBaseNote	U8	1byte	Note number that forms a base (0~127)
sampleDetune	S8	1byte	Detune (0~127) (note 1)
samplePanpot	S8	1byte	Panpot (0~64127= left~center~right, minus = negative phase)
sampleGroup	U8	1byte	Group (0~127), 0 = no group
samplePriority	U8	1byte	Priority (priority max: 127)
sampleVolume	U8	1byte	Volume (0 ~ 128)
reserved	U8	1byte	(reserved region)
sampleAdsr1	U16	2bytes	Envelope ADSR1 (note 2)
sampleAdsr2	U16	2bytes	Envelope ADSR2 (note 2)
keyFollowAr	S8	1byte	Change width of AR for each octave
keyFollowArCenter	U8	1byte	Note number for which keyFollowAr yields a result of 0(0~127)
keyFollowDr	S8	1byte	Change width of DR for each octave

Field	Type	Size	Contents
keyFollowDrCenter	U8	1byte	Note number for which keyFollowDr yields a result of 0 (0~127)
keyFollowSr	S8	1byte	Change width of SR for each octave
keyFollowSrCenter	U8	1byte	Note number for which keyFollowSr yields a result of 0 (0~127)
keyFollowRr	S8	1byte	Change width of RR for each octave
keyFollowRrCenter	U8	1byte	Note number for which keyFollowRr yields a result of 0 (0~127)
keyFollowSl	S8	1byte	Change width of SL for each octave
keyFollowSlCenter	U8	1byte	Note number for which keyFollowSl yields a result of 0 (0~127)
samplePitchLfoDelay	U16	2bytes	Pitch LFO delay time (units in msec)
samplePitchLfoFade	U16	2bytes	Pitch LFO fade time (units in msec)
sampleAmpLfoDelay	U16	2bytes	Amp LFO delay time (units in msec)
sampleAmpLfoFade	U16	2bytes	Amp LFO fade time (units in msec)
sampleLfoAttr	U8	1byte	Attributes for LFO
sampleSpuAttr	U8	1byte	Attributes for SPU

Note 1: Units are 128 equal half steps.

Note 2: Refer to the System Manual "SPU2 Overview" for details on the ADSR1 / ADSR2 data format used to represent envelopes.

The value NO_CINDEX (=-1) in the VagIndex field means that SPU2 generated noise will be used in place of waveform data.

Specifying a cross fade means to smooth the boundary of a velocity range when bit 7 of velRangeLow / velRangeHigh is set to 1. Because the structure of a cross fade is similar to the structure in a split, refer to page 17 for details on the structure.

sampleLfoAttr represents attributes of LFO using the following bit patterns.

Table 29

Macro constant	Value	Meaning
LFO_KEY_ON_PITCH	0x01	Key On when Pitch LFO trigger
LFO_KEY_OFF_PITCH	0x02	Key Off when Pitch LFO trigger
LFO_BOTH_PITCH	0x04	Key On and Key Off when Pitch LFO trigger
LFO_KEY_ON_AMP	0x10	Key On when Amp LFO trigger
LFO_KEY_OFF_AMP	0x20	Key Off when Amp LFO trigger
LFO_BOTH_AMP	0x40	Key On and Key Off when Amp LFO trigger

sampleSpuAttr represents attributes which provide SPU2 channel control using the following bit patterns.

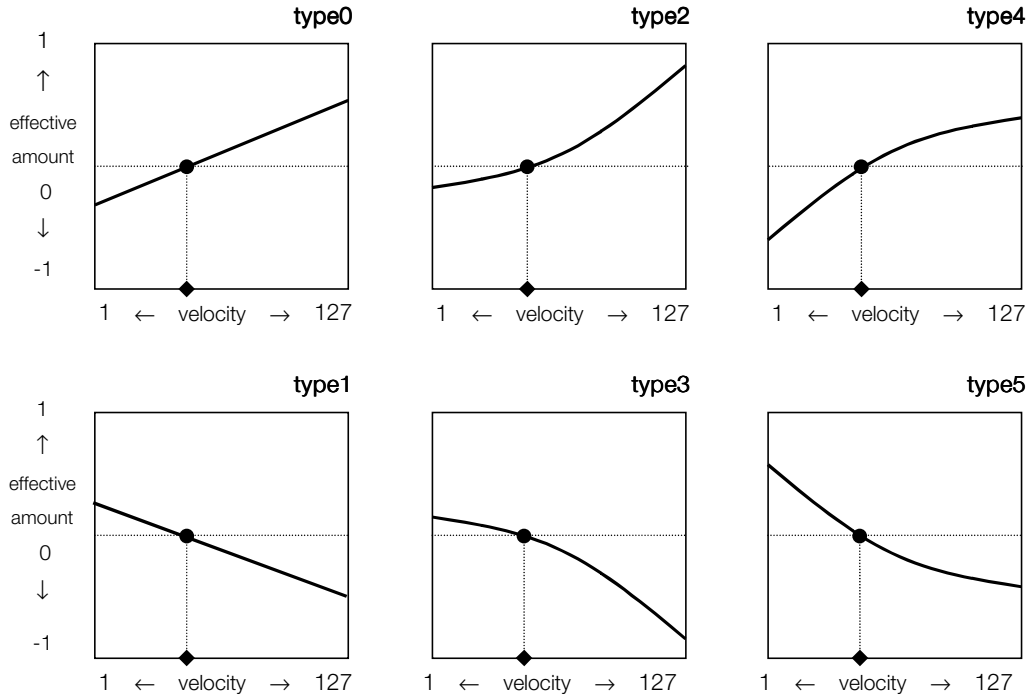
Table 30

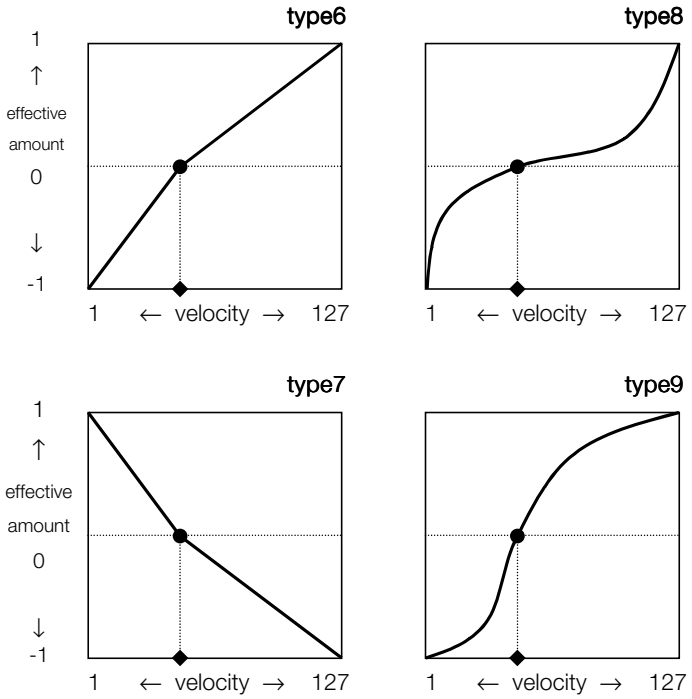
Macro constant	Value	Meaning
SPU_DIRECTSEND_L	0x01	SPU2 [VMIXL0/1] register
SPU_DIRECTSEND_R	0x02	SPU2 [VMIXR0/1] register
SPU_EFFECTSEND_L	0x04	SPU2 [VMIXEL0/1] register
SPU_EFFECTSEND_R	0x08	SPU2 [VMIXER0/1] register
SPU_CORE_0	0x10	CORE0 select
SPU_CORE_1	0x20	CORE1 select

Velocity Curve Type

velFollowPitchVelCurve and velFollowAmpVelCurve represent a velocity curve type that allows the effective amount of LFO to change as a function of velocity value.

Figure 14: LFO velocity curve type





Regardless of the type, the point at which the effective amount is 0 is at first determined by velFollowPitchCenter / velFollowAmpCenter. After this the slope of the curve is determined by velFollowPitch / velFollowAmp in type0 ~ type5. Finally, in type6 ~ type9, the slope of the curve is determined such that the effective amount for a velocity of 1/127 becomes 1/-1.

VAGInfo Chunk

A VAGInfo chunk is a data structure that receives a link from a sample chunk and associates it with the corresponding waveform data located in a body file. The contents and structure are shown below.

Table 31

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	Chunk type Vagi
chunkSize	U32	4bytes	Chunk size
maxVagInfoNumber	U32	4bytes	Maximum value (*n*) of VAGInfo number
vagInfoOffsetAddr[0]	COFST	4bytes	Location of parameter block of VAGInfo0 (Number of bytes from start of chunk)
...	...	...	...
vagInfoOffsetAddr[*n*]	COFST	4bytes	Location of parameter block of VAGInfon (Number of bytes from start of chunk)
vagInfoParam[0]	*	8bytes	Parameter block of VAGInfo0
...	...	...	...
vagInfoParam[*n*]	*	8bytes	Parameter block of VAGInfon

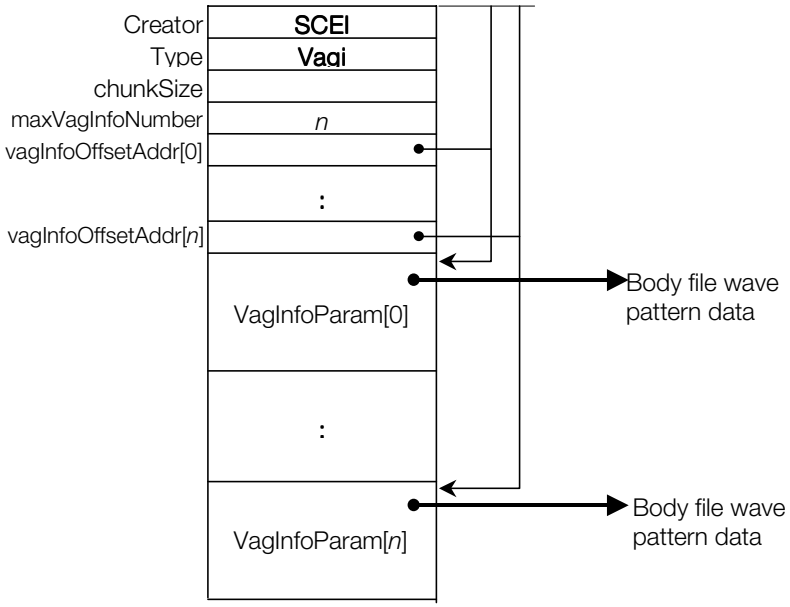
* sceHardSynthVagParam

The contents and structure of a VAGInfo parameter block (sceHardSynthVagParam) are shown below.

Table 32

Field	Type	Size	Contents
VagOffsetAddr	FOFST	4bytes	Location of waveform data (Number of bytes from start of body file)
VagSampleRate	U16	2bytes	Sampling rate
VagAttribute	U8	1byte	Loop flag VAG_ATTR_1SHOT(0x00) : 1 Shot VAG_ATTR_LOOP(0x01) : loop
Reserved	U8	1byte (reserved area)	

Figure 15: Structure of VAGInfo chunk



Sound Effect Timbre Chunk

A sound effect timbre chunk is a data structure that indicates parameters and waveform data for generating sound effect sequence data.

Complete Structure of Sound Effect Timbre Chunk

The contents and structure of a sound effect timbre chunk are shown below.

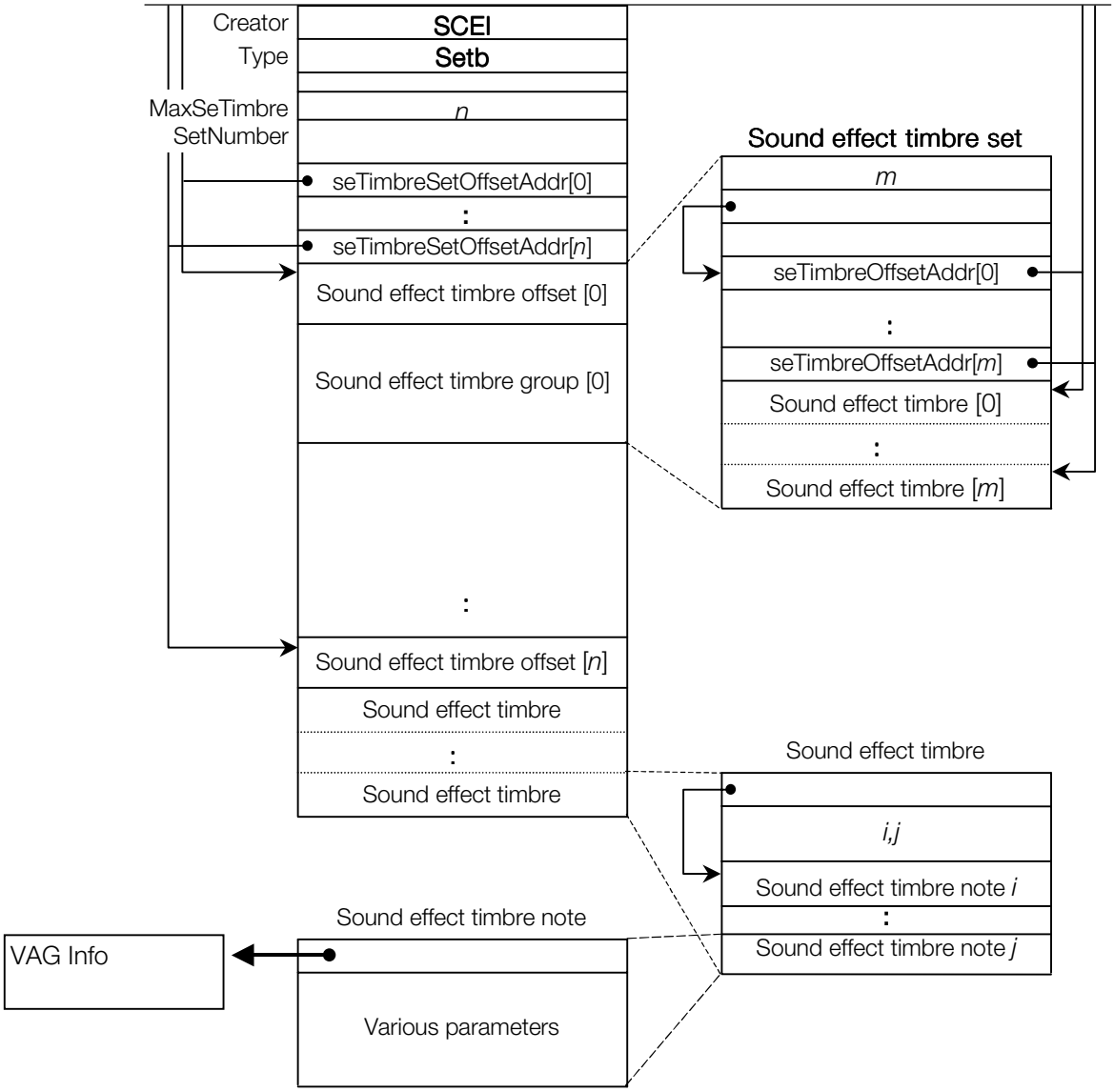
Table 33

Field	Type	Size	Contents
Creator	U32	4bytes	Creator code SCEI
Type	U32	4bytes	chunk type Setb
chunkSize	U32	4bytes	chunk size
maxSeTimbreSetNumber	U32	4bytes	Maximum value (*n*) of sound effect sequence set number
seTimbreSetOffsetAddr[0]	COFST	4bytes	Location of parameter block of sound effect timbre set 0 (Number of bytes from top of the chunk)
...	...	...	...
seTimbreSetOffsetAddr[*n*]	COFST	4bytes	Location of parameter block of sound effect timbre set *n* (Number of bytes from start of the chunk)
Sound effect timbre set 0	*	Not specified	Parameter block of sound effect timbre set 0
Sound effect timbre	**	Not specified	Sound effect timbre group that belongs to sound effect timbre set 0
...	...	...	...
Sound effect timbre set *n*	*	Not specified	Parameter block of sound effect timbre set *n*
Sound effect timbre	**	Not specified	Sound effect timbre group that belongs to sound effect timbre set *n*

*sceHardSynthSeTimbreSetBlock

**sceHardSynthSeTimbreBlock

Figure 16: Structure of sound effect timbre chunk



Structure of Sound Effect Timbre Set

The contents of a sound effect timbre set are shown below.

Table 34

Field	Type	Size	Contents
maxSeTimbreNumber	U32	4bytes	Maximum size (*m*) of sound effect timbre number that belongs to this sound effect timbre set
seTimbreOffsetAddr[0]	COFST	4bytes	Location of parameter block of sound effect timbre 0 (Number of bytes from start of chunk)
...	...	...	...
seTimbreOffsetAddr[*m*]	COFST	4bytes	Location of parameter block of sound effect timbre *m* (Number of bytes from start of chunk)

Structure of Sound Effect Timbre

The contents of a sound effect timbre are shown below.

Table 35

Field	Type	Size	Contents
SeTimbreNoteBlockAddr	POFST	4bytes	Offset to first sound effect timbre note block within this sound effect timbre
SeTimbreNoteBlockSize	U8	1byte	Size (*s*) of sound effect timbre note block
SeTimberNoteStart	U8	1byte	Note number (*i*) of first sound effect timbre note block
SeTimberNoteEnd	U8	1byte	Note number (*j*) of last sound effect timbre note block
Reserved	U8	1byte	reserved area
SeTimbreNoteBlock	*	*s*	Sound effect timbre note block of note number *i*
...	...	...	...
seTimbreNoteBlock	*	*s*	Sound effect timbre note block of note number *j*

*sceHardSynthSeTimbreNoteBlock

Structure of Sound Effect Timbre Note Block

The contents of a sound effect timbre note block are shown below.

Table 36

Field	Type	Size	Contents
VagIndex	CINDEX	2bytes	VAGInfo number (link information to VAGInfo chunk)
seTimbreNoteVelCurve	U8	1byte	Velocity curve type (0-5) (Note 1)
seTimbreNoteVolume	U8	1byte	Volume(0 - 128)
seTimbreNotePanpot	S8	1byte	Panpot (0~64~127 = left~center~right, minus = negative phase)
seTimbreNoteTranspose	S8	1byte	Transpose (units in notes)
seTimbreNoteDetune	S8	1byte	Detune (units in 1/128 half steps)
seTimbreNoteGroup	U8	1byte	Sound effect timbre note group number (0 - 127, 0 = no group)
SeTimbreNoteGroupLimit	U8	1byte	Limit of number of generated sounds within sound effect timbre note group (0 - 48, 0 = no limit)
seTimbreNotePriority	U8	1byte	Priority (Maximum priority = 127)
reserved	U8 X 2	2bytes	reserved area
seTimbreNoteAttr	U8	1byte	Sound effect timbre note attribute (note 2)
seTimbreNoteSpuAttr	U8	1byte	SPU attribute (note 3)
seTimbreAdsr1	U16	2bytes	Envelope parameter ADSR1
seTimbreAdsr2	U16	2bytes	Envelope parameter ADSR2
seTimbreNotePitchLfoWave	U8	1byte	Pitch LFO waveform (note 4)
seTimbreNoteAmpLfoWave	U8	1byte	Amp LFO waveform (note 4)
seTimbreNotePitchLfoStartPhase	U8	1byte	Pitch LFO start phase (0 - 255, corresponding to 0 ~ 359 degrees)
seTimbreNoteAmpLfoStartPhase	U8	1byte	Amp LFO start phase (0 - 255, corresponding to 0 ~ 359 degrees)

Field	Type	Size	Contents
seTimbreNotePitchLfoRandomPhase	U8	1byte	Pitch LFO random start phase (0 - 255, corresponding to 0 ~ 359 degrees)
seTimbreNoteAmpLfoRandomPhase	U8	1byte	Amp LFO random start phase (0 - 255, corresponding to 0 ~ 359 degrees)
seTimbreNotePitchLfoCycle	U16	2bytes	Pitch LFO period (units in msec)
seTimbreNoteAmpLfoCycle	U16	2bytes	Amp LFO period (units in msec)
seTimbreNotePitchLfoDepth	S16	2bytes	Pitch LFO amplitude+ (units in 1/128 half steps)
seTimbreNotePitchLfoDepth2	S16	2bytes	Pitch LFO amplitude- (units in 1/128 half steps)
seTimbreNoteAmpLfoDepth	S8	1byte	Amp LFO amplitude+
seTimbreNoteAmpLfoDepth2	S8	1byte	Amp LFO amplitude-
seTimbreNotePitchLfoDelay	U16	2bytes	Pitch LFO delay time (units in msec)
seTimbreNoteAmpLfoDelay	U16	2bytes	Amp LFO delay time (units in msec)
seTimbreNotePitchLfoFade	U16	2bytes	Pitch LFO fade time (units in msec)
seTimbreNoteAmpLfoFade	U16	2bytes	Amp LFO fade time (units in msec)
seTimbreNoteVelFollowPitch	S8	1byte	Width change of pitch LFO in response to velocity (-127 - 0 - 127: units in 1/128 half steps)
seTimbreNoteVelFollowAmp	S8	1byte	Width change of amp LFO in response to velocity (-127 - 0 - 127)
seTimbreNoteVelFollowPitchCenter	U8	1byte	Velocity which causes seTimbreNoteVelFollowPitch to become 0 (0 - 127)
seTimbreNoteVelFollowAmpCenter	U8	1byte	Velocity which causes seTimbreNoteVelFollowAmp to become 0 (0 - 127)
seTimbreNoteVelFollowPitchCurve	U8	1byte	Curve type that determines the amount of seTimbreNoteVelFollowPitch effect (0 - 9) (note 5)

Field	Type	Size	Contents
seTimbreNoteVelFollowAmpCurve	U8	1byte	Curve type that determines the amount of seTimbreNoteVelFollowAmp effect (0 - 9) (note 5)
reserved	U8	1byte	reserved area
seTimbreNoteLfoAttr	U8	1byte	LFO attribute (note 6)

(Note 1) For details on the velocity curve, refer to Fig. 11 Velocity curve type (page 24)

(Note 2) Sound effect timbre note attribute (seTimbreNoteAttr) is specified by the following bit flags.

Table 37

Constant	Value	Meaning
ROUND_PAN	0x01	Negative phase can also be used in panpot changes of sound effect timbre note

(Note 3) SPU attribute (seTimbreNoteSpuAttr) is specified by the following bit flags.

Table 38

Constant	Value	Meaning
SPU_DIRECTSEND_L	0x01	Set VMIXL0 / 1 register to 1 (to DryL)
SPU_DIRECTSEND_R	0x02	Set VMIXR0 / 1 register to 1 (to DryR)
SPU_EFFECTSEND_L	0x04	Set VMIXEL0 / 1 register to 1 (to WetL)
SPU_EFFECTSEND_R	0x08	Set VMIXER0 / 1 register to 1 (to WetR)
SPU_CORE_0	0x10	Generate sound with core 0
SPU_CORE_1	0x20	Generate sound with core 1

(Note 4) LFO waveform is specified with the following constants.

Table 39

Macro constant	value	waveform
LFO_WAVEFORM_NON	0	No LFO
LFO_WAVEFORM_SAWUP	1	Rising sawtooth wave
LFO_WAVEFORM_SAWDOWN	2	Falling sawtooth wave
LFO_WAVEFORM_TRIANGLE	3	Triangle wave
LFO_WAVEFORM_SQUEARE	4	Rectangular wave
LFO_WAVEFORM_NOISE	5	Noise
LFO_WAVEFORM_SIN	6	Sine wave
LFO_WAVEFORM_USER	0x80	User defined (not supported)

(Note 5) For details on the velocity curve, refer to Fig. 13 LFO velocity curve type (page 28)

(Note 6) LFO attribute (seTimbreLfoAttr) is specified by the following bit flags.

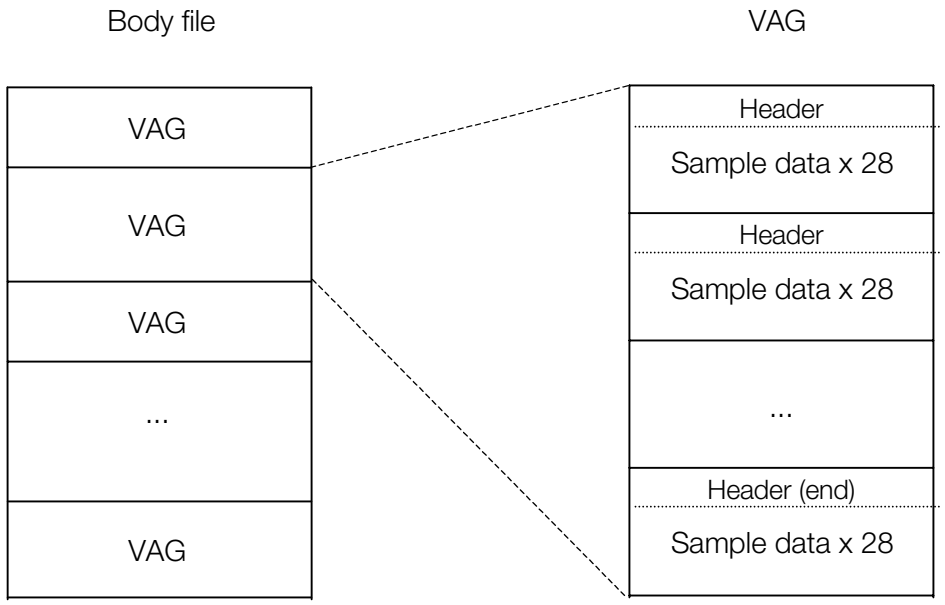
Table 40

Constant	Value	Meaning
LFO_KEY_ON_PITCH	0x01	Make Key On the Pitch LFO trigger
LFO_KEY_OFF_PITCH	0x02	Make Key Off the Pitch LFO trigger
LFO_BOTH_PITCH	0x04	Make both Key On / Key Off the Pitch LFO trigger
LFO_KEY_ON_AMP	0x10	Make Key On the Amp LFO trigger
LFO_KEY_OFF_AMP	0x20	Make Key Off the Amp LFO trigger
LFO_BOTH_AMP	0x40	Make both Key On / Key Off the Amp LFO trigger

Body File

A body file (bd) consists of a list of waveform data (VAG) that were compressed using ADPCM. VAG is comprised of 16-byte waveform data blocks, each of which has a 16-bit header and 28 sets of 4-bit sample data. An endpoint bit in the header signifies the last block.

Figure 17: Structure of body file (bd)

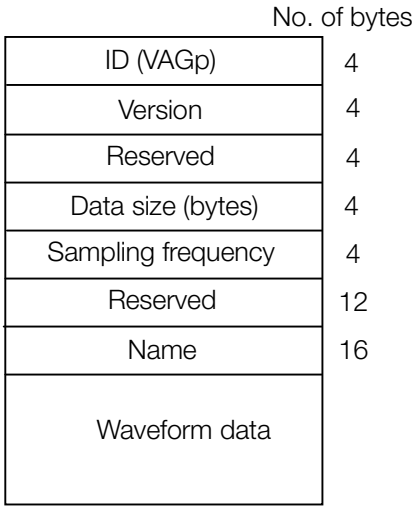


Refer to the System Manual "SPU2 Overview" for details on waveform data blocks.

Reference: Output format of AIFF2VAG

The VAG file format when converted by AIFF2VAG is shown below. In the figure, the waveform data portion corresponds to the VAG of the body file.

Figure 18



Software Synthesizer Waveform Data Format (SS)

Overview

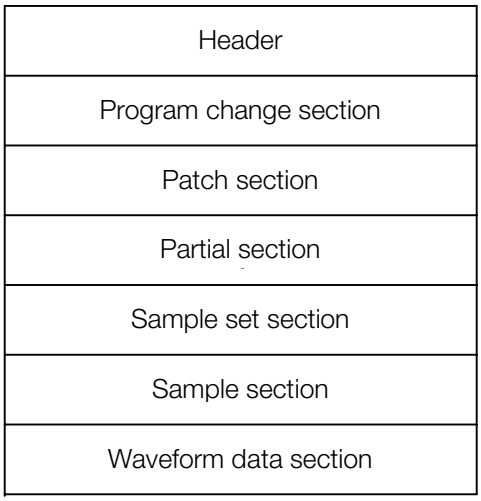
SS is a phoneme and parameter data format used by the CSL software synthesizer (libssyn).

A file having the format shown below is loaded into EE memory, sceSSyn_PrepareParameter() is called to resolve the address, and sceSSyn_Load() is called to register the file so that it can be used by the software synthesizer.

File Structure

Software synthesizer waveform data consists of a header followed by an arrangement of sections as shown below. Since the position of each section is indicated by position information within the header, the specification allows the order of sections to be arbitrary, and either empty space or other data may be inserted between sections without causing a problem.

Figure 19: Overall Structure of SS Data



Header

The header is defined as an sceSynthesizerParameterHeader structure having the following contents.

Table 41

Field	Type	Size	Contents
attrib	U32	4 bytes	Data attribute (always 0)
programChange_section	FOFST	4 bytes	Pointer to program change section
programChange_section_size	U32	4 bytes	Program change section size
patch_section	FOFST	4 bytes	Pointer to patch section
patch_section_size	U32	4 bytes	Patch section size
partial_section	FOFST	4 bytes	Pointer to partial section
partial_section_size	U32	4 bytes	Partial section size
sampleSet_section	FOFST	4 bytes	Pointer to sample set section
sampleSet_section_size	U32	4 bytes	Sample set section size
sample_section	FOFST	4 bytes	Pointer to sample section
sample_section_size	U32	4 bytes	Sample section size
wave_section	FOFST	4 bytes	Pointer to waveform data section
wave_section_size	U32	4 bytes	Waveform data section size
bank_map	U8 x 128	128 bytes	Mapping table from bank select to program change

The pointer to each section, which appears as an FOFST type in the above table, represents the offset in the file. However, address resolution is performed to rewrite it as a address in memory so that it becomes a pointer type to the appropriate structure. Similar processing is performed for each section explained below.

Bank Map

A bank map, which is a table for mapping from a bank select to a program change map, is embedded in the header.

Table 42

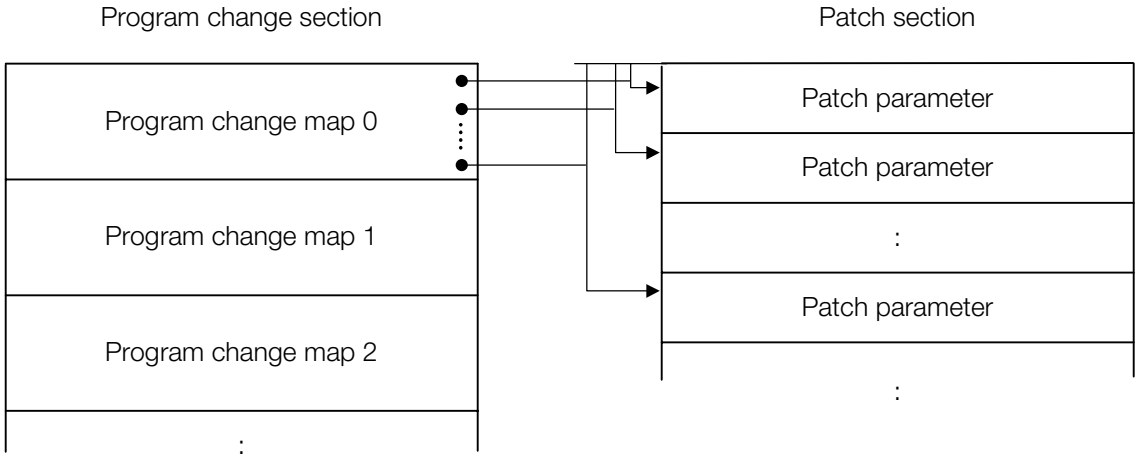
Field	Type	Size	Contents
pPgMap[0]	U8	1 byte	Program change map number corresponding to bank 0 (0 - 127)
pPgMap[1]	U8	1 byte	Program change map number corresponding to bank 1 (0 - 127)
...	...	...	...
pPgMap[127]	U8	1 byte	Program change map number corresponding to bank 127 (0 - 127)

When bank *n* does not exist, 128 is entered for pPgMap[n].

Program Change Section

The program change section, which contains data indicating a mapping from the program number to the patch section, consists of a list of program change maps corresponding to each bank.

Figure 20: Correspondence Between Program Change Section and Patch Section



The program change map contents and structure are as follows. The program change map is defined as a sceSynthsizerProgramChangeMap structure.

Table 43

Field	Type	Size	Contents
patch[0]	SOFST	4 bytes	Position of patch parameter corresponding to program0
patch[1]	SOFST	4 bytes	Position of patch parameter corresponding to program1
...	...	...	...
patch[127]	SOFST	4 bytes	Position of patch parameter corresponding to program127

When the corresponding program does not exist, NO_SOFST (-1) is entered for patch[n].

Patch Section

The patch section collects together the patch parameters. A patch parameter, which contains data corresponding to one program number (tone quality or timbre), has the following structure.

Figure 21: Patch Parameter Structure

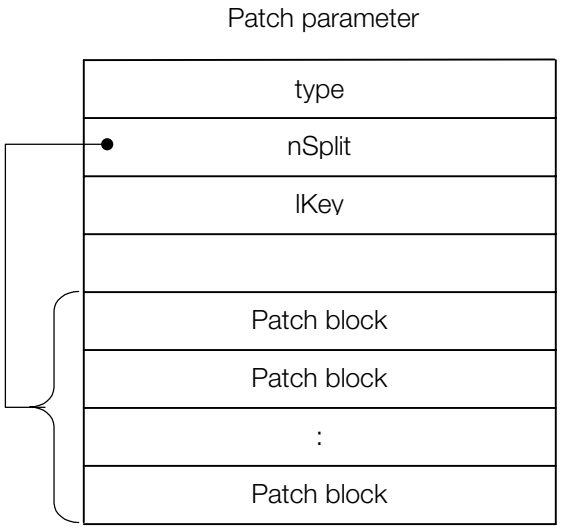


Table 44

Field	Type	Size	Contents
type	U8	1 byte	Timbre type (patch block type) 0 Melody instrument 1 Drum instrument
nSplit	U8	1 byte	Number of splits
lKey	U8	1 byte	Lower limit key number of register (0 - 127)
reserved	-	1 byte	Reserved area
split	(*)	8 bytes x nSplit	Patch block

(*) sceSynthesizerMeloPatchBlock or sceSynthesizerDrumPatchBlock, depending on type.

With a normal musical scale instrument, a patch block corresponds to key split information, and the following kind of sceSynthesizerMeloPatchBlock structure, which has an arbitrary key area (note number range) and partial parameter information corresponding to that key area, is used.

Table 45

Field	Type	Size	Contents
pPartial	SINDEX	2 bytes	Index of partial parameter corresponding to this split
tune	S16	2 bytes	Pitch correction (cent)
volume	U8	1 byte	Volume (0 - 255)
panpot	S8	1 byte	Panpot (-127=left, 0=center, 127=right)
uKey	U8	1 byte	Upper limit key number of register
efxSend	U8	1 byte	Effect send level (0 - 255)

With a drum instrument, a patch block corresponds to an individual instrument, and the following kind of sceSynthesizerMeloPatchBlock structure is assigned to a single note number.

Table 46

Field	Type	Size	Contents
pPartial	SINDEX	2 bytes	Index of partial parameter corresponding to this instrument
tune	S16	2 bytes	Pitch correction (cent)
volume	U8	1 byte	Volume (0 - 255)
panpot	S8	1 byte	Panpot (-127=left, 0=center, 127=right)
monoGroup	U8	1 byte	Monogroup (1 - 127)
efxSend	U8	1 byte	Effect send level (0 - 255)

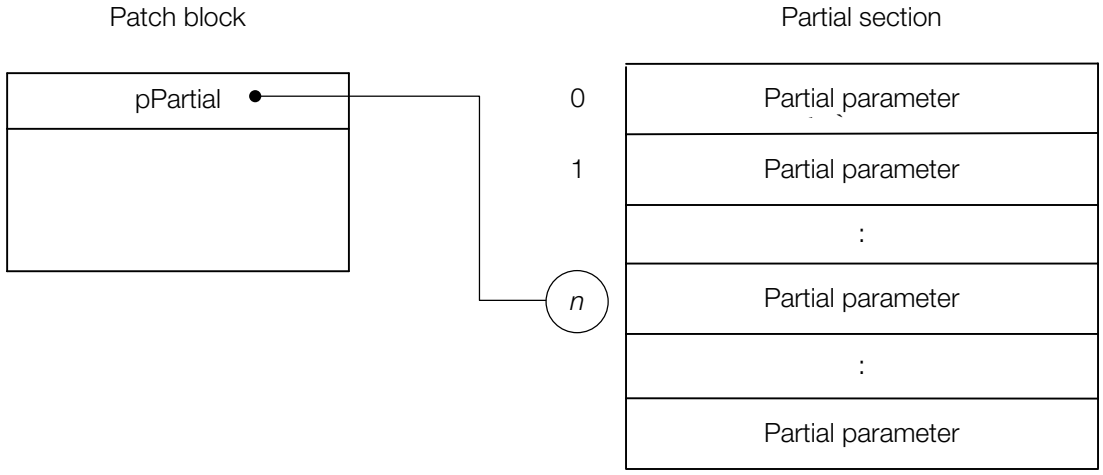
A mono group, which inhibits the simultaneous production of sound by grouping patch blocks, assigns the same number to the patch blocks for which monaural sound is to be produced. The value 255 is assigned when a patch block does not belong to a group. (In JAM, not belonging to a group is represented by 0, but in SS data, 255 is used.)

Although volume is represented in JAM by a number from 0 to 127, in SS data, it is represented as 128 levels in the range from 0 to 255.

Partial Section

The partial section lists partial parameters that are basic sets of timbres. A partial parameter is linked from a patch block by an index.

Figure 22: Correspondence Between Patch Blocks and Partial Parameters



A partial parameter is defined as an sceSynthesizerPartialParam structure consisting of five blocks as shown below.

Table 47

Field	Type	Size	Contents
osc	Described later	26 bytes	Oscillator parameter
tvf	Described later	48 bytes	TVF parameter
tva	Described later	30 bytes	TVA parameter
lfo[0]	Described later	26 bytes	LFO parameter 0
lfo[1]	Described later	26 bytes	LFO parameter 1

lfo[0] and lfo[1] are independent LFO parameters. They both affect the oscillator, TVA, and TVF parameters, but have separate envelope depths for each. Details are given later.

Oscillator Parameter

The oscillator parameter osc, which keeps waveform data position information and information related to noise, is defined as an sceSynthesizerOscParam structure having the following structure.

Table 48

Field	Type	Size	Contents
set[0]	Described later	24 bytes	Oscillator block 0
set[1]	Described later	24 bytes	Oscillator block 1
pitch_envelope	Described later	24 bytes	Pitch envelope parameter
vel_curve	Described later	4 bytes	Velocity curve

The oscillator blocks set[0] and set[1], which keep waveform data position information and information related to noise, are defined as sceSynthesizerOscBlock structures having the following structure.

Table 49

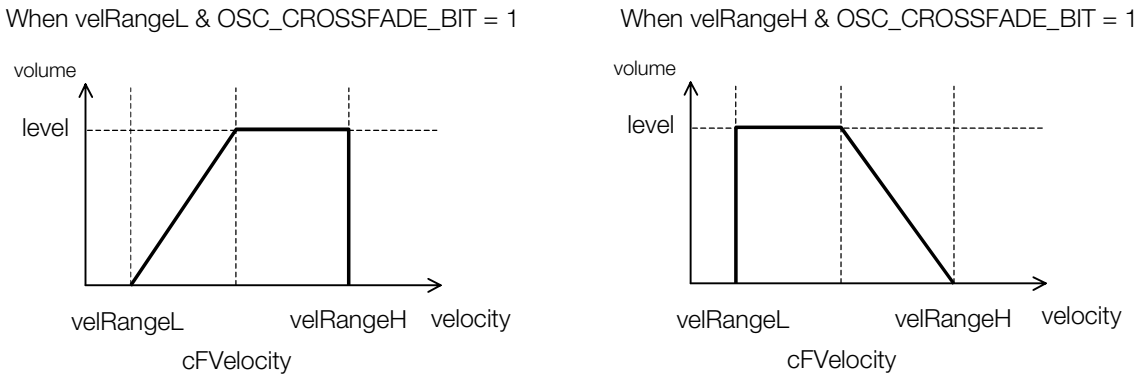
Field	Type	Size	Contents
pSampleSet	SOFST	2 bytes	Sample set parameter position information
vel_sample_start	SWADDR	4 bytes	Variation in read position due to velocity
vel_pitch_mod	S16	2 bytes	Pitch modulation due to velocity (cent: -1200 - 1200)
tune	S16	2 bytes	Tuning (cent)
keyFollow	S16	2 bytes	Variation width per octave (cent: -2400 - 2400)
pitch_env_depth	S16	2 bytes	Pitch envelope depth (cent: -2400 - 2400)
att	U8	1 byte	Attenuator (0 - 255, 128=1.0)
velRangeL	U8	1 byte	Velocity lower limit value
cFVelocity	U8	1 byte	Cross-fade starting velocity
velRangeH	U8	1 byte	Velocity upper limit value
level	U8	1 byte	Noise level (128=1.0)
reserved	-	3 bytes	Reserved area

If pSampleSet is set to -1 or the level field is set to 0, this oscillator block will be disabled.

pitch_env_depth indicates how much effect pitch_envelope within the oscillator parameter has on this oscillator block.

The following relationship must hold among velRangeL, cFVelocity, and velRangeH: velRangeL <= cFVelocity <= velRangeH. When the OSC_CROSSFADE_BIT of velRangeL or velRangeH is 1, a cross-fade occurs as follows.

Figure 23: Cross-fade



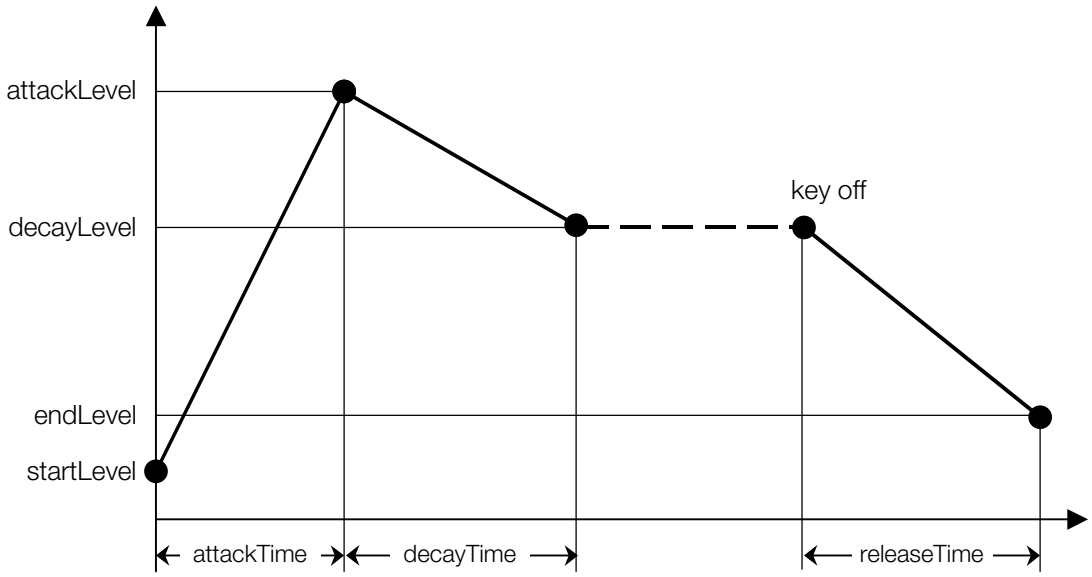
The envelope parameter is defined as the following kind of sceSynthesizerEnvelopeParam structure.

Table 50

Field	Type	Size	Contents
attackTime	U16	2 bytes	Attack time (msec)
decayTime	U16	2 bytes	Decay time (msec)
releaseTime	U16	2 bytes	Release time (msec)
startLevel	S16	2 bytes	Attack starting level
attackLevel	S16	2 bytes	Attack level
decayLevel	S16	2 bytes	Decay level
endLevel	S16	2 bytes	Release ending level
kf_point	U8	1 byte	Key flow point (key number)
reserved	-	1 byte	Reserved area
kf_time	S16	2 bytes	Variation width of envelope time due to key flow (per octave)
kf_level	S16	2 bytes	Variation width of envelope level due to key flow (per octave)
vel_time	S16	2 bytes	Variation width of envelope time due to velocity (per octave)
vel_level	S16	2 bytes	Variation width of envelope level due to velocity (per octave)

The parameters from attackTime to decayLevel represent the envelope curve as follows.

Figure 24: Envelope Curve



Note: For TVA, endLevel must be set to 0.

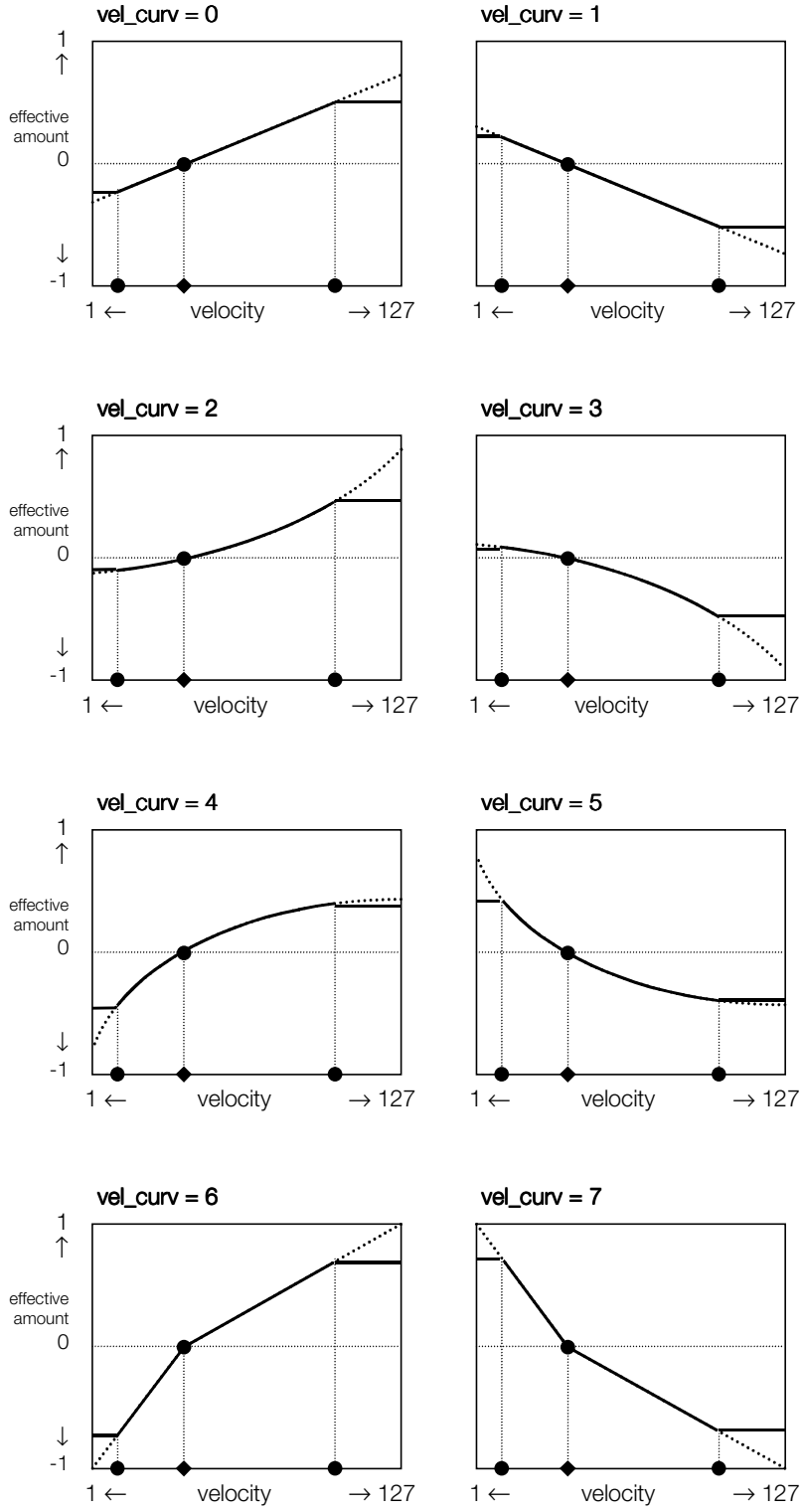
The velocity curve is defined as the following kind of sceSynthesizerVelocityCurve structure.

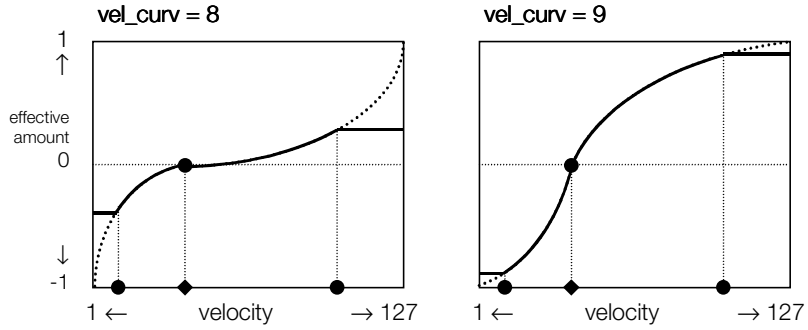
Table 51

Field	Type	Size	Contents
vel_curv	U8	1 byte	Velocity curve type
vel_center	U8	1 byte	Velocity value for which modulation is 0 due to velocity curve
vel_uLimit	U8	1 byte	Velocity upper limit value. A velocity value that exceeds this is clamped at vel_uLimit.
vel_lLimit	U8	1 byte	Velocity lower limit value. A velocity value below this is clamped at vel_lLimit.

vel_curv values, which range from 0 to 9 (VELOCITY_CURVE_MAX), represent the following kind of velocity curve. The three points shown on the horizontal axis in the figure represent vel_lLimit, vel_center, and vel_uLimit.

Figure 25: Velocity Curve





TVF Parameter

The TVF parameter is defined as an sceSynthesizerTvfParam structure having the following structure.

Table 52

Field	Type	Size	Contents
cutoff	U16	2 bytes	Cut-off frequency (Hz)
resonance	U16	2 bytes	Resonance value. bits 0-3 are fractional part.
type	U8		Filter type
kf_point	U8	1 byte	Key flow point (key number)
kf_cutoff	S16	2 bytes	Variation width per octave of cut-off frequency (cent)
kf_resonance	S16	2 bytes	Variation width per octave of resonance value (0x4000 = 1.0)
vel_cutoff	S16	2 bytes	Variation width due to velocity of cut-off frequency
vel_resonance	S16	2 bytes	Variation width due to velocity of resonance value
tvf_env_depth	S16	2 bytes	Envelope depth (cent)
random_cutoff	S16	2 bytes	Random modulation depth of cut-off frequency (cent)
tvf_envelope	(*)	24 bytes	Envelope parameter
vel_curve	(*)(*)	4 bytes	Velocity curve

(*) sceSynthesizerEnvelopeParam

(*)(*) sceSynthesizerVelocityCurve

Any of the following constants can be entered for type, which indicates the filter type.

Table 53

Constant	Value	Meaning
SCESYNTHESIZER_TVF_NONE	0	No filter
SCESYNTHESIZER_TVF_LPF	1	Low-pass filter
SCESYNTHESIZER_TVF_BPF	2	Band-pass filter
SCESYNTHESIZER_TVF_HPF	3	High-pass filter

For information about the envelope parameter and velocity curve, see the description of the oscillator parameter.

TVA Parameter

The TVA parameter is defined as an sceSynthesizerTvaParam structure having the following structure.

Table 54

Field	Type	Size	Contents
kf_point	U8	1 byte	Key flow point (key number)
kf_panpot	S8	1 byte	Variation width of panpot due to key flow (per octave)
vel_panpot	S8	1 byte	Variation width of panpot due to velocity
random_panpot	U8	1 byte	Random panpot variation range (0 - 127)
panpot	S8	1 byte	Panpot 0=Center, -127=Left, 127=Right
vel_curve	(*)	4 bytes	Velocity curve type
reserved	-	1 byte	Reserved area
tva_envelope	(*)(*)	24 bytes	Envelope parameter

(*) sceSynthesizerVelocityCurve

(*)(*) sceSynthesizerEnvelopeParam

For information about the velocity curve type and envelope parameter, see the description of the oscillator parameter. For TVA, endLevel within the envelope parameter must be set to 0. Also, there is no field for setting the envelope depth, which is always 1.

LFO Parameter

The LFO parameter is defined as an sceSynthesizerLfoParam structure having the following structure.

Table 55

Field	Type	Size	Contents
freq	U16	2 bytes	Frequency (0.01Hz units)
delay	U16	2 bytes	Start Delay (msec)
fadeTime	U16	2 bytes	Fade-in time (msec)
pitch_depth[2]	S16	2 bytes x 2	Pitch modulation depth (cent: -2400 - 2400)
midi_pitch_depth[2]	S16	2 bytes x 2	Pitch modulation depth at MIDI mod. maximum (cent: -2400 - 2400)
tvf_depth	S16	2 bytes	TVF cut-off modulation depth (cent: -2400 - 2400)
midi_tvf_depth	S16	2 bytes	TVF cut-off modulation depth at MIDI mod. maximum (cent: -2400 - 2400)
reserved	-	2 bytes	Reserved area

Field	Type	Size	Contents
tva_depth	S8	1 byte	TVA modulation depth (-127=-1.0, +127=+1.0)
midi_tva_depth	S8	1 byte	TVA modulation depth at MIDI mod. maximum (-127=-1.0, +127=+1.0)
waveForm	U8	1 byte	LFO waveform
start_phase	U8	1 byte	LFO start phase (255=360 degrees)
start_phase_random_depth	U8	1 byte	Start phase random variation range (255=360 degrees)
reserved	-	1 byte	Reserved area

pitch_depth[0] and pitch_depth[1] represent the LFO depth for oscillator blocks 0 and 1, respectively. Similarly for midi_pitch_depth[0] and midi_pitch_depth[1].

The following constants are defined for specifying the LFO waveform in waveForm.

Table 56

Constant	Value	Meaning
SCESYNTHESIZER_LFO_NONE	0	No LFO
SCESYNTHESIZER_LFO_SAWUP	1	Right rising sawtooth wave
SCESYNTHESIZER_LFO_SAWDOWN	2	Right falling sawtooth wave
SCESYNTHESIZER_LFO_TRIANGLE	3	Triangle wave
SCESYNTHESIZER_LFO_SQUARE	4	Square wave
SCESYNTHESIZER_LFO_NOISE	5	Noise

Sample Set Section

The sample set section collects together sample set parameters. A sample set parameter, which contains data that indicates a key area (note number range) and samples corresponding to that area, has the following structure.

Figure 26: Sample set parameter structure

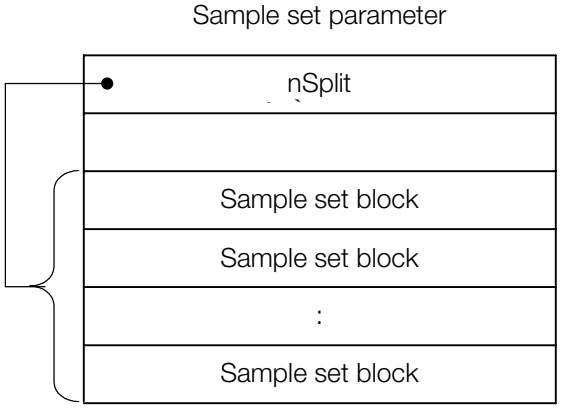


Figure 27: Sample Set Parameter Structure

Field	Type	Size	Contents
nSplit	U8	1 byte	Number of splits (= Number of sample set blocks)
reserved	-	1 byte	Reserved area
sampleBlock	(*)	4 bytes x nSplit	Sample set block

For the data specification, multiple sample parameters are assigned to one sample set parameter as described above. However, JAM, which does not support sample set editing, generates data in which one sample set parameter contains only one sample set block (nSplit is always 1). This is done so that two sample set parameters can be assigned for a partial parameter, which makes it easier for the creator to handle.

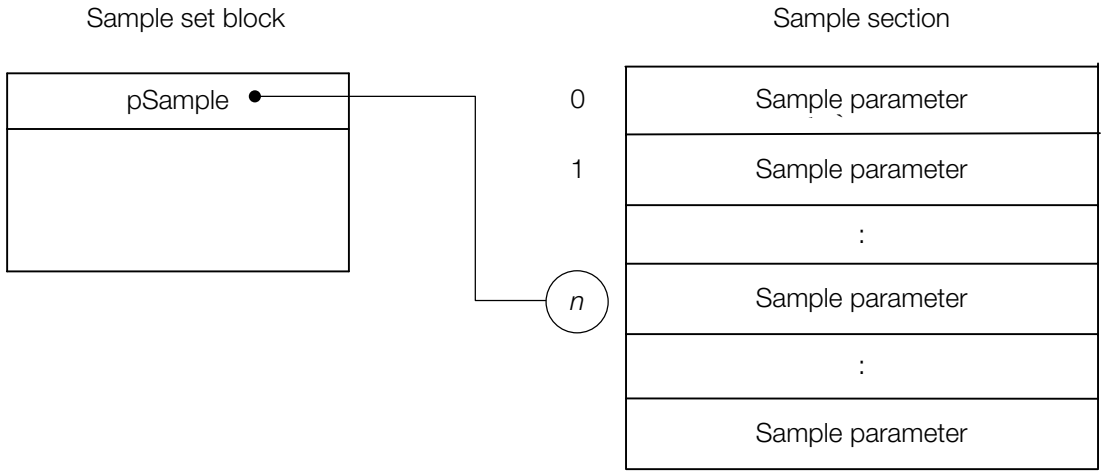
A sample set block is defined as an sceSynthesizerSampleSetBlock structure having the following structure.

Table 57

Field	Type	Size	Contents
pSample	SINDEX	2 bytes	Sample parameter index
uKey	U8	1 byte	Sound production upper limit key number
reserved	U8	1 byte	Reserved area

The correspondence with a sample parameter is created by pSample having sample section index information as follows.

Figure 28: Correspondence Between Sample Set Block and Sample Parameter



Sample Section

The sample section collects together sample parameters, which are basic information for reading waveforms. A sample parameter is defined as the following kind of sceSynthesizerSampleParam structure.

Table 58

Field	Type	Size	Contents
pWave	SOFST	4 bytes	Waveform data address
waveStart	WADDR	4 bytes	Read starting position
loopStart	WADDR	4 bytes	Loop starting position
loopEnd	WADDR	4 bytes	Loop ending position
tune	S16	2 bytes	Correction tuning due to sampling rate (cent)
rootKey	U8	1 byte	Sampling note
att	U8	1 byte	Attenuator (0 - 255, 127=1.0)
loopTune	S8	1 byte	Loop tuning (0.1 cent units)
atribute	U8	1 byte	Attribute
reserved	-	2 bytes	Reserved area

rootKey represents the sampled musical interval according to key number.

For attribute, bit 0 represents the loop type and bit 1 represents the data width (8-bit PCM or 16-bit PCM). These are defined by the following constants.

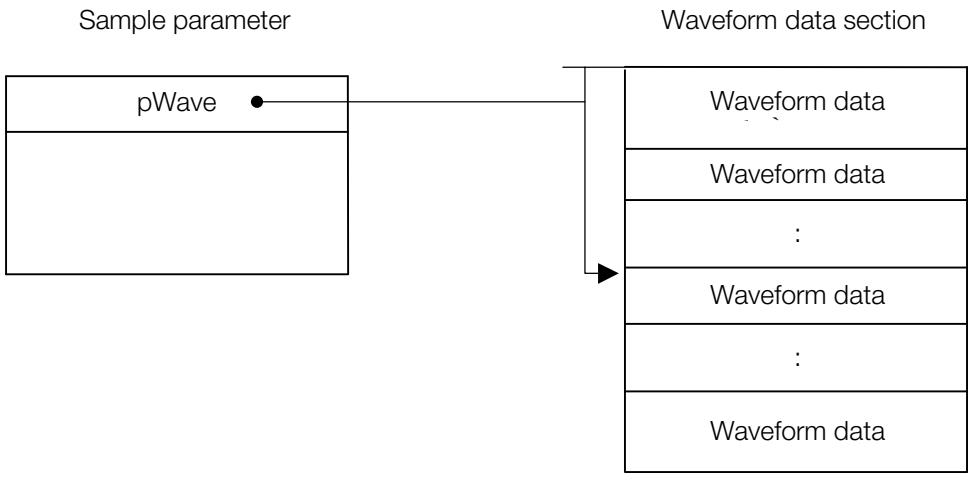
Table 59

Constant	Value	Meaning
OSC_PCM_WAVE_ATTRIB_FWD	0<<0	Loop exists
OSC_PCM_WAVE_ATTRIB_ONESHOT	1<<0	No loop
OSC_PCM_WAVE_ATTRIB_PCM16	0<<1	16-bit PCM
OSC_PCM_WAVE_ATTRIB_PCM8	1<<1	8-bit PCM

Waveform Data Section

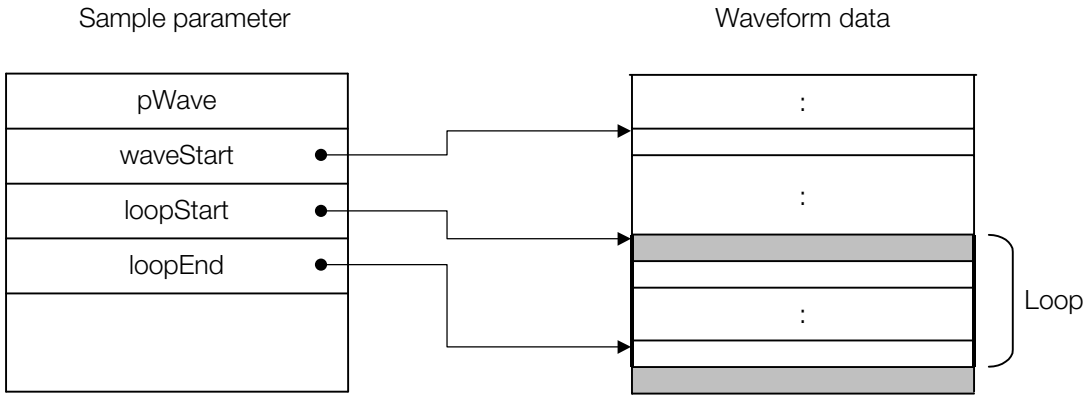
The waveform data section collects together waveform data. The waveform data is simply listed sequentially as shown in the figure, and sample parameters point to each waveform data value.

Figure 29: Correspondence Between Sample Parameters and Waveform Data



Waveform data is 8-bit or 16-bit signed monaural PCM data with 2-byte alignment. The sample immediately after the loop ending position must be the same data as the loop starting position data. Only one set of waveform data can be set per loop.

Figure 30: Waveform Data Loop



In addition, the upper limit of the musical interval for which sound can be produced by that waveform data is determined by the length of the loop. So, to produce sound up to x octaves above the original musical interval (rootKey), the loop length (number of samples) must be at least 2x.