$PSDocId: Document Release 1.0 for Runtime Library Release 3.6$

96.Sep.

PlayStation(R) Programmer Tool
Run-time Library Release 3.6

Changes from ver.3.5, Enhancements,
and Known bugs

Known Bugs and Cautions

In libsnd, where a noise sound source such as SEQ is used , a noise may be produced at an unexpected place. Thus if you want to use a noise, please use libspu/libsnd functions directly on the programmer tool(PC) or use sampled noise and treat it as the same way as the waveform.

In libgs, dividing GsSortObject4J lower functions using GsA4div will not reflect the TMD translucent information set in "attribute". If you want to use the translucent attribute, the translucent flag of the TMD data must be set ON.

In libcd, CdlSetloc must be issued to determine the reading position before issuing CdRead(). Without setting the reading position, repeating the CdRead()may fail to read in the right amount of data.

In libspu SpuSetVoiceAttr()/SpuGetVoiceAttr(), the correct pitch value cannot be set/obtained by using sample_note or note. The value will contain considerably large errors. In order to set/obtain the correct pitch, use the "pitch" attribute.

In libgun, the vertical direction counter value stored into the buffer is the system clock value and is cleared at every H-BLANK. Thus correct the value to reflect the screen size before actual use.

Samples Modified and Added

CD

\PSX\SAMPLE\CD\TUTO\TUTO1.C now complies with the new CD swapping specification. Please refer to the document "Imperative Steps for Title Consisting of Multiple CDs", BBS released from SCEI and

psx\doc\jp\word\overview\cd.doc

SCEA\CNTRL
=========
SCEA\CNTRL: Demonstrates usage of the various controller
API's. This was shown at the autumn SCEA developer's conference in 1996.
The program automatically recognizes what controllers are attached to
the controller ports -- pads, light guns, steering wheels, and
mice.

Graphics
============
\PSX\SAMPLE\GRAPHICS\ROTATE\AXESMIME : Axis Interpolation MIME sample

\PSX\SAMPLE\GRAPHICS\CLUTFOG\TUTO3 : Texture Depth Queuing by DR_LOAD
sample

Sound
=========
\PSX\SAMPLE\SOUND\SIMPLE\JUMP: FSEQ Play by Jump Table sample

Document: Directory Structure, Modifications, and Changes
==========
(For Japanese only)
For this version, the new document in HTML format is added.
Please use Netscape 3.0 or later, Internet Explorer 3.0 or later to
browse it.

Note, with this change, document directory structure was changed as
follows;

```
PSX--DOC--JP---HTML-LIBREF
     |  |
     |  ---OVERVIEW
     |
     --TEXT-LIBREF
     |  |
     |  ---OVERVIEW
     |
     --WORD-LIBREF
        |
        ---OVERVIEW
```

The documents, "abstract.txt" & "appendix.txt" and "function.txt" & "struct.txt"
that were previously under "TEXT" were merged into "OVERVIEW" and "LIBREF"
respectively.

(e.g.)..TEXT\GPU\ABSTRACT.TXT, APPENDIX.TXT•¨..TEXT\OVERVIEW\GPU.TXT
..TEXT\GPU\FUNCTION.TXT, STRUCT.TXT•¨..TEXT\LIBREF\GPU.TXT

The size of the argument, "result"(8 bytes) was added to the CdReadSync () explanation.

An incorrect specification of CdGetSector() was corrected in the "Library Reference".

The bit explanation of GsSPRITE member, "attribute", was enhanced.

Type 2 instruction mapping table for C header files(inlin_c.h, inline_o.h)
and Assemble header file(inline_a.h) and explanation of Type 2 instruction with
and without "nop" was added to dmpsx abstract.txt

GsDefDispBuff () explanation was modified.
Wrong default values for the double buffer (0,0), (0, y_res) was
corrected to (0,0), (0,0).

Serial Input/Output Library (libsio)

This is a newly available library from this release 3.6.
This is a library to perform standard I/O between PC and PS using the communication
cable DTL-H3050.
Since the standard I/O of the debugging station is set to NULL normally, no debug
information can be obtained.
By using this library, libsio, standard I/O can be allocated to the PS communication port, and
by connecting the communication cable DTL-H3050, RS232C I/O is enabled.

Available Functions

AddSIO	Registers SIO Driver
DelSIO	Delete SIO Driver
_sio_control	BIOS Interface for SIO Driver

Kernel Library(libapi)

Renamed Function

In C++, the function "delete" releases memory that was allocated by
the "new" function. Unfortunately, this "delete" conflicts with the
libapi function, "delete()", and causes compiler errors. To avoid
these compiler errors, the libapi function "delete()" has been changed
to "erase()".

Malloc.obj deleted

The functions included in the "malloc.obj" file have been moved into
libapi.lib, and the names of those functions have been changed
to malloc2(), free2(), realloc2(), InitHeap2(), calloc2(), free2().
"malloc.obj" has been deleted from this release.

Functions Added

InitHeap2 Initializes Heap Area

Syntax
```
void InitHeap2( head, size )
void *head;
long size;
```

Arguments

head Heap start address

size Heap size (multiple of 4, in bytes)

Explanation

This function initializes a group of standard function library memory control functions. After using this function, malloc2(), etc. are usable.

There is an overhead so the entire "size" in bytes cannot be used.

This is the bug fix version of InitHeap() but causes a larger program size since this is a memory resident function.

Return Value

None

Remarks

If several executions of this function overlap, the memory control information previously held will be lost.

Related Functions

InitHeap(),malloc2(),realloc2(),calloc2(),free2()

malloc2 Allocates main memory

Syntax
```
#include <stdlib.h>
void *malloc2(size_t s)
```

Explanation

This function allocates "s" bytes of memory block from the heap memory.

Corresponds to InitHeap2().

Return Value

Returns a pointer to allocated memory block.
If failed, NULL is returned.
*Heap memory is defined as below;

Low Address Module Highest Address + 4
High Address On-board memory - 64KB

Related Function

calloc2(),realloc2(),free2()

realloc2 Changes the heap memory allocation

Syntax
```
#include <stdlib.h>
void *realloc2(void *block, size_t s)
```

Explanation

This function increases/decreases the size of the memory block previously allocated to "s" bytes.
Same as malloc2 when block is NULL.
Corresponds to InitHeap2().

Return Value

Returns a pointer to the reallocated memory block.
The new pointer may have different address from the original.
If reallocation fails, NULL will be returned, and the original block will not be released.

Related Function

calloc2(),malloc2(),free2()

calloc2

Allocates main memory

Syntax

```
#include <stdlib.h>
void *calloc2( size_t n, size_t s )
```

Argument		
	n	Number of partitions
	s	Size of one partition

Explanation

This function allocates a block of n*s bytes.
Corresponds to InitHeap2().

Return Value

Returns a pointer to the allocated memory block.
If the allocation fails, NULL will be returned.

Related Function

malloc2(),realloc2(),free2()

free2

Frees allocated memory blocks

Syntax

```
#include <stdlib.h>
void free2(void *block)
```

Explanation

This function releases a memory block that was allocated by calloc2,malloc2, and realloc2.
Corresponds to InitHeap2().

Return Value

None

Related Function

calloc2(),malloc2(),realloc2()

DMPSX

Library Version Update

DMPSX has been upgraded for GTE specification disclosure.

DMPSX version 2.06 -> version 3.01

Please read "psx\doc\jp\text\overview\dmpsx.txt" for detail.

Header File Changes

Type 2 instruction without "nop" have been added to "inline_c.h".
This instruction has the name "gte_rtps_b", "_b" added to the original
type 2 instruction name "gte_rtps", and "nop" is deleted.

(Example)

gte_rtps() -> gte_rtps_b()

DMPSX gives errors and will not convert when there are cop2 instructions
in 2 slots prior to the type 2 instructions. Thus when using type 2 instructions
without nop in the program, make sure to place instruction (e.g. CPU instruction)
other than cop2.

CD-ROM Library/Streaming Library (libcd)

Header File Changes

The prototype declaration for CdReset() has been added to "libcd.h" as follows;

```
int CdReset(int mode);
```

A duplication of the StFreeRing() prototype declaration has been fixed.

Function Deleted

The primitive command CdlReset has been deleted.

When resetting CD subsystem, be sure to use the following function;

int CdReset(int mode)

When using the above function, internal variables used in the library
will not be updated. For this reason, any function that follows may not be
invoked correctly. Affected functions currently known are as below;

CdMode()
CdLastCom()

Although CdGetSector() was remarked as a NON-BLOCKING in the earlier version of library reference manual, it actually is a BLOCKING function. Since the data transfer completes upon returning from a function, you do not need to use CdDataSync() or CdDataCallback() to determine transfer of data has been completed.

Command Added
==========
The following commands are added to libcd;

CdlGetTN	Obtains number of TOC entries
CdlGetTD	Obtains TOC

Refer to the document "overview/cd.doc" for command details.

Functions Modified
==========
CdGetDiskType() now complies with the new CD swapping specification.

Please refer to the document "Imperative Steps for Title Consisting of Multiple CDs", BBS released from SCEI,
psx\doc\jp\word\overview\cd.doc,
and the sample program,
psx\sample\cd\tuto\tuto11.c.

Combat Cable Library(libcomb)
==========

Official Version
==========
"libcomb", which was located in the Runtime Library Release 3.5 beta version as \beta\comb.lzh, is now officially released in this version.

Basic Graphics library(libgpu)
==========

Bug Fixes in Library
==========
- The bug that all characters specified in KanjiFntOpen() are not displayed in KanjiFntPrint() when attempting to display many characters has been fixed.

- The bug that ResetGraph(3) did not operate has been fixed.
(Already fixed in Release 3.5)

- The bug that ClearImage() for texture pages (other than Drawing Area) only clears either odd or even lines has been fixed.

- Although the drawing environment was reset within the library upon

restarting drawing by ContinueDraw in the previous releases,
it now does not reset due to the following bug:
Upon execution of drawing setting related primitive(such as DR_ENV),
suspending the drawing by BreakDraw and then restart by ContinueDraw fails to
set the correct drawing settings.

If the drawing environment for the primitives to be inserted and the primitive
to be continued are different, add the primitive that resets the drawing
environment at the end of the primitive to be inserted.

Although the drawing is suspended by BreakDraw, GPU will not stop
until the drawing is completed.

The new function, IsIdleGPU was added to check if the drawing
suspended by BreakDraw has been completed or not.

- In MargePrim() for checking the maximum packet size, the tag that
the error return value -1 was not returned even when the packet size
exceeds its max, 16-word(including the tag), has been fixed.

Function Added
==========
IsIdleGPU Checks if the drawing once suspended by BreakDraw
was completed

Syntax int IsIdleGPU(int maxcount)

Argument
maxcount Count value

Explanation
Although drawing is suspended by BreakDraw, GPU will not stop
until the drawing is completed.

Thus this function checks if the drawing suspended by BreakDraw
has been completed or not. If GPU will not be an idle state within
the time given by maxcount, -1 will be returned.

Return Value
0: GPU is in idle state
-1: GPU is in drawing state

Functions Added
==========
- Mode for GPU initializing has been added to GsInitGraph() called "intl".
(Please see the description of the "GsInitGraph" in the "Header File Changes"
section below -- [Ed.])

Extended Graphics Library(libgs)
==========

Bug Fixes in Library
==========

- In GsLinkObject4 and GsSortObject4, the bug that the gradation square polygon with light source calculation was not displayed has been fixed. With this fix, two new members are added to "_GsFCALL" structure in "libgs.h". Please be careful when initializing a jump table.

- In GsLinkObject4 and GsSortObject4, the bug that the gradation square translucent polygon with light source calculation was displayed as opaque has been fixed.

Header File Changes
==========
In GsSortClear(), a prototype declaration duplication had been removed.

Functions Added
==========
GsInitGraph Graphics System Initialization

Syntax

```
void GsInitGraph(ix_res, y_res, intl, dither, vram)
int x_res;
int y_res;
int intl;
int dither;
int vram;
```

Argument

x_res Horizontal Resolution (256/320/384/512/640)

y_res Vertical Resolution (240/480)

intl Interlace Display Flag(bit 0)

0: Non-interlace	GsNONINTER
1: Interlace	GSINTER

Double buffer offset mode(bit 2)

0: GTE Offset	GsOFSGTE
1: GPU Offset	GsOFSGPU

GPU Initialize Parameter(bit 4-5)

0: ResetGraph(0)	GsRESET0
3: ResetGraph(3)	GsRESET3

dither Dithering processing flag
0: OFF
1: ON

vram: VRAM mode
0: 16bit
1: 24bit

Explanation

Resets "libgpu" and initializes "libgs" graphic system.
"libgpu" settings are accessed by a global variables, GsDISPENV, and GsDRAWENV. Thus the programmer can verify and/or modify "libgpu" by referencing these variables.

Vertical 480 line non-interlace mode is effective only when a VGA monitor is connected.

WARNING: The top and bottom 8 lines are almost invisible on typical home-use TV monitors when the vertical resolution line mode is set to 240.

For PAL mode, display position should be shifted down by 24 lines.

Double buffer offset mode determines whether GTE or GPU offset mode is used . When the GPU is used, the packet does not include offset values, and thus is easier to be process.

For 24 bit mode, only the memory image display is available ; therefore, polygon drawing cannot be performed.

Since initializing the graphic system involves initialization of GsIDMATRIX and GsIDMATRIX2 as well, GsInitGraph() must be called prior to all other Gs library functions for correct operation.

Return Value

None

Mip-mapping functions

==========

(Editor's note: The following mipmapping functions have been added. A description of the exact syntax follows. Please refer to the \samples\graphics\mipmap for an example implementation.)

GsTMDfastTF4LM	mip-map Flat Texture Square (Light Source Calc.)
GsTMDfastTF4LFGM	mip-map Flat Texture Square (Light Source Calc.) {FOG)
GsTMDfastTF4NLM	mip-map Flat Texture Square (Without Light Source Calc.)
GsTMDfastTNF4M	mip-map Flat Texture Square (Without Light Source Calc.)
GsTMDfastTG4LM	mip-map Gouraud Texture Square(Light Source Calc.)
GsTMDfastTG4LFGM	mip-map Gouraud Texture Square(Light Source Calc.){FOG)
GsTMDfastTG4NLM	mip-map Gouraud Texture Square(Without Light Source Calc.)
GsTMDfastTNG4M	mip-map Gouraud Texture Square(Without Light Source Calc.)
GsTMDdivTF4LM	mip-map Flat Texture Square(Fixed Division) {Light Source Calc.)
GsTMDdivTF4LFGM	mip-map Flat Texture Square (Fixed Division) {Light Source Calc.){FOG)
GsTMDdivTF4NLM	mip-map Flat Texture Square (Fixed Division){Without Light Source Calc.)
GsTMDdivTNF4M	mip-map Flat Texture Square (Fixed Division){Without Light Source Calc.)
GsTMDdivTG4LM	mip-map Gouraud Texture Square (Fixed Division){Light Source Calc.)
GsTMDdivTG4LFGM	mip-map Gouraud Texture Square (Fixed Division){Light Source Calc.){FOG)
GsTMDdivTG4NLM	mip-map Gouraud Texture Square (Fixed Division){Without Light Source Calc.)
GsTMDdivTNG4M	mip-map Gouraud Texture Square (Fixed Division){Without Light Source Calc.)
GsA4divTF4LM	mip-map Flat Texture Square (Automatic Division){Light Source Calc.)
GsA4divTF4LFGM	mip-map Flat Texture Square

(Automatic Division){Light Source Calc.){FOG)

GsA4divTF4NLM mip-map Flat Texture Square
(Automatic Division){Without Light Source Calc.)

GsA4divTNF4M mip-map Flat Texture Square
(Automatic Division){Without Light Source Calc.)

GsA4divTG4LM mip-map Gouraud Texture Square
(Automatic Division){Light Source Calc.)

GsA4divTG4LFGM mip-map Gouraud Texture Square
(Automatic Division){Light Source Calc.){FOG)

GsA4divTG4NLM mip-map Gouraud Texture Square
(Automatic Division){Without Light Source Calc.)

GsA4divTNG4M mip-map Gouraud Texture Square
(Automatic Division){Without Light Source Calc.)

Syntax

```
PACKET *GsTMDfastTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTNF4M(TMD_P_TF4 *op, VERT *vp,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDfastTNG4M(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot)
PACKET *GsTMDdivTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,
                DIVPOLYGON4 *divp)
PACKET *GsTMDdivTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTNF4M(TMD_P_TF4 *op, VERT *vp,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsTMDdivTNG4M(TMD_P_TG4 *op, VERT *vp,
        PACKET *pk,int n,int shift, GsOT *ot,       DIVPOLYGON4 *divp)
PACKET *GsA4divTF4LM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot,       u_long *scratch)
PACKET *GsA4divTF4LFGM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTF4NLM(TMD_P_TF4 *op, VERT *vp, VERT *np,
        PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTNF4M(TMD_P_TF4 *op, VERT *vp,
        PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
```

PACKET *GsA4divTG4LM(TMD_P_TG4 *op, VERT *vp, VERT *np,
PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTG4LFGM(TMD_P_TG4 *op, VERT *vp, VERT *np,
PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTG4NLM(TMD_P_TG4 *op, VERT *vp, VERT *np,
PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)
PACKET *GsA4divTNG4M(TMD_P_TG4 *op, VERT *vp,
PACKET *pk,int n,int shift, GsOT *ot, u_long *scratch)

Argument

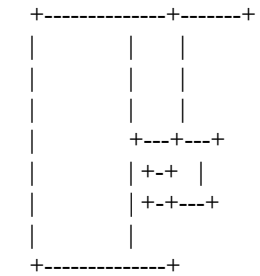
op	TMD PRIMITIVE Starting Address
vp	TMD VERTEXS Starting Address
np	TMD NORMAL Starting Address
pk	GPU Packet Buffer Starting Address
n	Number of PRIMITIVEs
shift	Number of bits to be shifted when sorting to OT
ot	Pointer to GsOT
scratch	Non-used scratch pad Starting Address

Explanation

Low level function group of GsSortObject4J()
Need to be registered into GsFCALL4 as a low level function
before using.

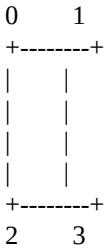
This function performs mip-map, texture switching based
on the polygon size , to the flat texture squares included in
the TMD data.

Locate the texture on the V-RAM as below;



There are four texture sizes, 1, 1/4, 1/16, and 1/64.
Which texture size should be used is determined by the polygon's
outer product.

Polygon vertices must be in the order below;



Return Value

Non-used Packet Area Starting Address

Basic Geometry Library (libgte)

Bug Fixes in Library

In the RotMeshPrimS_FCT3 function, setting the
"len" member of the TMESH structure to an odd number would cause a bug.
All sides of such a mesh would be displayed in reverse, except for the first side.
This bug has been fixed.

Header File Changes

An ApplyMatrix*() prototype declaration duplication had been removed.

Function Descriptions Added

(Documentation updates only)

SetMulRotMatrix Multiplies constant rotation matrix by a matrix and sets one constant rotation matrix.

Syntax
```
MATRIX* SetMulRotMatrix(m0)
MATRIX *m0;    /*Input: Matrix */
```

Explanation

This function multiplies constant rotation matrix and
a matrix and stores the value in one constant rotation matrix.

<Argument Format>
m0->m[i][j] : (1,3,12)

Return Value

m0

ApplyRotMatrixLV Multiplies a vector by a constant rotation matrix.

Syntax
```
VECTOR* ApplyRotMatrixLV(v0,v1)
VECTOR *v0;    /* Input: Vector */
VECTOR *v1;    /*Output: Vector */
```

Explanation

This function multiplies a constant rotation matrix by a vector
v0 beginning from the rightmost end and stores the result in
vector v1.

<Argument Format>
v0->vx,vy,vz :(1,31,0)
v1->vx,vy,vz :(1,31,0)

Return Value v1

MatrixNormal_0 Orthonormalizes a matrix.

Syntax void MatrixNormal_0(m,n)
MATRIX *m; /* Input: Matrix */
MATRIX *n; /* Output: Matrix*/

Explanation
This function orthonormalizes a distorted rotation matrix.
(Note: m[2][0],m[2][1], and m[2][2] will be ignored.)

<Argument Format>
m->m[i][j] : (1,3,12)
n->m[i][j] : (1,3,12)

Return Value None

CompMatrixLV Make a composite coordinate transformation matrix.

Syntax MATRIX* CompMatrix(m0,m1.m2)
MATRIX *m0; /* Input: Matrix */
MATRIX *m1; /* Input: Matrix */
MATRIX *m2; /* Output: Matrix */

Explanation
This function makes a composite coordinate transformation
matrix that includes parallel translation.
[m2->m] = [m0->m] * [m1->m]
(m2->t) = [m0->m] * (m1->t) + (m0->t)
<Argument Format>
m0->m[i][j] : (1,3,12)
m0->t[i] : (1,31,0)
m1->m[i][j] : (1,3,12)
m1->t[i] : (1,31,0)
m2->m[i][j] : (1,3,12)
m2->t[i] : (1,31,0)

<Remarks>
This function destroys a rotation matrix.
Return Value m2

Gun Library (libgun)
==========

Renamed Functions
==========
The following functions were renamed to avoid duplicate definition errors during
linking.

InitGUN <- InitGun,InitPAD
StartGUN <- StartPAD
StopGUN <- StopPAD
SelectGUN <- SelectGun

Previous InitGun and InitPAD functions were merged into one function, InitGUN, and the same arguments for InitGun and InitPAD are used. A new function RemoveGUN was added in order to completely remove the gun driver when a child process is used in LoadExec and the like.

Data Processing Library (libpress)

Bug Fixes in Library

We fixed the following bug...If you set non-zero value by DecDCTvlcSize(), DecDCTvlc() broke some part of memory area.(The NEWS version library is free from this bug.)

Functions Added

EncSPU Encodes 16-bit PCM data into PlayStation original waveform format

---

Syntax long EncSPU (ENCSPUENV *es_env)

Argument

es_env SPU encode environment attribute structure

Explanation

This function encodes the PCM data specified in a member "src" of the SPU encode environment attribute structure, "es_env" into the PlayStation original waveform data(VAG, without header information) and returns the encoded data in a member "dest".

Specify the user area address for both members "src" and "dest" of the SPU encode environment attribute structure, "es_env".

Divided encoding can be done by specifying an attribute to a member "proceed" of the SPU encode environment structure, "es_env".

Return Value

ENCSPU_ENCODE_ERROR is returned for a size error of encoded PlayStation original waveform data.

ENCSPUENV SPU encode environment attribute structure

Structrue

```
typedef struct {
    short *src;
    short *dest;
    long   size;
    long   loop_start;
    char   loop;
    char   byte_swap;
    char   proceed;
    char   pad4;
} ENCSPUENV;
```

Member

src	16-bit PCM data address
dest	PlayStation original waveform data
size	16-bit PCM data size(in bytes)
loop_start	PCM data loop start point(in bytes)
loop	Loop waveform generation specification
ENCSPU_ENCODE_LOOP:	Generate loop waveform data
ENCSPU_ENCODE_NO_LOOP:	Generate non-loop waveform data
byte_swap	PCM data endian specification
ENCSPU_ENCODE_ENDIAN_BIG:	16-bit big endian
ENCSPU_ENCODE_ENDIAN_LITTLE:	16-bit little endian
proceed	Whole/Divided encoding specification
ENCSPU_ENCODE_WHOLE	Whole encoding
ENCSPU_ENCODE_START	Start divided encoding
ENCSPU_ENCODE_CONTINUE	Continue divided encoding
ENCSPU_ENCODE_END	End divided encoding
pad4	System reserved

Explanation

This structure is used to specify the SPU encode environment attributes for EncSPU() function.

Remarks

When 0 is specified for "loop", "loop_start" will be ignored.

Extended Sound Library (libsnd)

Bug Fix in Library

In SsSeqSetVol, the bug that SEQ long note data volume could not be controlled has been fixed.

Specification Changes in Function
==========
The specification of SsVoKeyOn() has been changed in the way that
it now returns allocated voice in bit mask.

Functions Added
==========
dmy_Ss.... Jump Table Low Level Function Dummy

Syntax

void dmy_Ss...()

Argument

None

Explanation

When this function is called for the first time, it outputs the entry name of the jump table to the standard output device.

Use this as a dummy low level function and to determine which entry was called.

Return Value

None

Remarks

This function is provided for debugging.

SsSeqOpenJ Opens SEQ data.(Function Table version)

Syntax

short SsSeqOpenJ (unsigned long* addr, short vab_id)

Argument

addr Start address of SEQ data in the main memory
vab_id VAB id

Explanation

This function is equivalent to SsSeqOpen() if all the low level functions were registered. In addition to the SsSeqOpen() capability, this function enables a programmer to control functions to be registered to the table and thus improve code efficiency by not calling unnecessary low level functions.

For those slots that SsFCALL will not register, use dummy functions, standard function names with the prefix "dmy" so that even when a lower function was called, no BUS ERROR would occur and the function names would be printed out. After checking the called function names, register the function names without "dmy".

Return Value

SEQ Access Number: Used in the SEQ data access function, being the inner SEQ data control table number.

Related External Variable

SsFCALL Function table that SsSeqOpenJ() refers.

SsSepOpenJ Opens SEP data.(Function Table version)

Syntax

short SsSepOpenJ (unsigned long* addr, short vab_id,
short num2)

Argument

addr Start address of SEQ data in the main memory
vab_id VAB id
num2 Number of SEQs contained in SEP

Explanation

This function is equivalent to SsSepOpen() if all the low level functions were registered. In addition to the SsSepOpen() capability, this function enables a programmer to control functions to be registered to the table and thus improve code efficiency by not calling unnecessary low level functions.

For those slots that SsFCALL will not register, use dummy functions, standard function names with the prefix "dmy" so that even when a lower function was called, no BUS ERROR would occur and the function names would be printed out. After checking the called function names, register the function names without "dmy".

Return Value

SEQ Access Number: Used in the SEQ data access function, being the inner SEQ data control table number.

Related External Variable

SsFCALL Function table that SsSepOpenJ() refers.

_SsFCALL Function table type referenced in SsSeqOpenJ() and SsSepOpenJ()

Structure

```
typedef struct {
void (*noteon) ();
    void (*programchange) ();
        void (*pitchbend) ();
        void (*metaevent) ();
    void (*control[13]) ();
    void (*ccentry[20]) ();
} _SsFCALL;
```

Member

All members hold pointers to the low level functions.

noteon, programchanghe, pitchbend,
metaevent, control, ccentry MIDI status data
control array Events of MIDI status data
control change
ccentry array Entry events for nrpn, rpn data

Explanation

Functions SsSeqPlay() and SsSepPlay() analyze the MIDI status data and call low level functions. Although there are many low level functions, an application would not usually use all the functions. These low level function groups will be set by calling either SsSeqOpen() or SsSepOpen().

In order to reduce the code size by not linking unnecessary
low level functions, new functions SsSeqOpenJ() and SsSepOpenJ(),
compatible with SsSeqOpen() and SsSepOpen() respectively.
In the new functions, low level functions are in the jump
table so that the user can set only desired function group.

_SsFCALL is a structure that defines this function table.

Necessary functions can be linked by assigning the pointer to
the low level function. In reverse, link can be eliminated by
not assigning pointer and not placing extern declaration.
Note that calling a function without setting a pointer will cause
BUS ERROR. To avoid BUS ERROR, set a dummy function by
prefixing the low level function name with "dmy".

_Ss... Low level functions of SsSeqOpen(SsSepOpen)

Syntax

```
void _SsNoteOn (short num1, short num2, unsigned char pitch,
                                    unsigned char volume);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
pitch Pitch
volume Volume

Explanation

This low level function must be called when the Note
On/Off data is contained in SEQ/SEP.

Syntax

```
void _SsSetProgramChange(short num1, short num2,
                                        unsigned char data);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called when the Program
Change data is contained in SEQ/SEP.

Syntax

```
void _SsGetMetaEvent(short num1, short num2,
                                        unsigned char data);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called when the Meta Event
data(Tempo Change) is contained in SEQ/SEP.

Syntax

```
void _SsSetPitchBend(short num1, short num2);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data

Explanation

This low level function must be called when the Pitch Bend data is contained in SEQ/SEP.

Syntax

```
void _SsSetControlChange(short num1, short num2,
                         unsigned char data);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called when the Control Change data is contained in SEQ/SEP.

Syntax `void _SsContBankChange(short num1, short num2);`

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 0th data is contained in SEQ/SEP.

Syntax

```
void _SsContDataEntry(short num1, short num2,
                      unsigned char data);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 6th data is contained in SEQ/SEP.

Syntax

```
void _SsContMainVol(short num1, short num2,
                    unsigned char data);
```

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 7th data is contained in SEQ/SEP.

Syntax void _SsContPanpot(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 10th data is contained in SEQ/SEP.

Syntax void _SsContExpression(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 11th data is contained in SEQ/SEP.

Syntax void _SsContDamper(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 64th data is contained in SEQ/SEP.

Syntax void _SsContExternal(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with
_SsSetControlChange function when the Control Change
91th data is contained in SEQ/SEP.

Syntax void _SsContNrpn1(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with
_SsSetControlChange function when the Control Change
100th data is contained in SEQ/SEP.

Syntax void _SsContNrpn2(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with
_SsSetControlChange function when the Control Change
101th data is contained in SEQ/SEP.

Syntax void _SsContRpn1(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with
_SsSetControlChange function when the Control Change
98th data is contained in SEQ/SEP.

Syntax void _SsContRpn2(short num1, short num2, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with
_SsSetControlChange function when the Control Change
99th data is contained in SEQ/SEP.

Syntax void _SsContResetAll(short num1, short num2);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
data Status message data

Explanation

This low level function must be called along with _SsSetControlChange function when the Control Change 121th data is contained in SEQ/SEP.

Syntax void _SsSetNrpnVabAttr0(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr1(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr2(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr3(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr4(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr5(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr6(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr7(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr8(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr9(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr10(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr11(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr12(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr13(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr14(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr15(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr16(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr17(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr18(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Syntax void _SsSetNrpnVabAttr19(short num1, short num2, short num3, VagAtr va, short nrpn, unsigned char data);

Argument

num1 SEQ/SEP access number
num2 SEQ number within SEP data
num3 Track number

va VAG Tone headear
nrpn NRPN number
data Status message data

Explanation

This function must be called along with _SsSetControlChange, _SsContDataEntry, _SsContNrpn1, and _SsContNrpn2 when NRPN data is contained (NRPN sets VAB attribute data and repeat within a music and marking) in SEQ/SEP. Note that each setting data corresponds to the different low level function. See below for detail.

_SsSetNrpnVabAttr0 ... Priority (CC98=0)
_SsSetNrpnVabAttr1 ... Mode (CC 98 = 1)
_SsSetNrpnVabAttr2 ... Limit Low (CC 98 = 2)
_SsSetNrpnVabAttr3 ... Limit high (CC 98 = 3)
_SsSetNrpnVabAttr4 ... ADSR (AR-L) (CC 98 = 4)
_SsSetNrpnVabAttr5 ... ADSR (AR-E) (CC 98 = 5)
_SsSetNrpnVabAttr6 ... ADSR (DR) (CC 98 = 6)
_SsSetNrpnVabAttr7 ... ADSR (SL) (CC 98 = 7)
_SsSetNrpnVabAttr8 ... ADSR (SR-L) (CC 98 = 8)
_SsSetNrpnVabAttr9 ... ADSR (SR-E) (CC 98 = 9)
_SsSetNrpnVabAttr10 ... ADSR (RR-L) (CC 98 = 10)
_SsSetNrpnVabAttr11 ... ADSR (RR-E) (CC 98 = 11)
_SsSetNrpnVabAttr12 ... ADSR (SR-•}) (CC 98 = 12)
_SsSetNrpnVabAttr13 ... Vibrate time (CC 98 = 13)
_SsSetNrpnVabAttr14 ... Portamento depth (CC 98 = 14)
_SsSetNrpnVabAttr15 ... Reverb type (CC 98 = 15)
_SsSetNrpnVabAttr16 ... Reverb depth (CC 98 = 16)
_SsSetNrpnVabAttr17 ... Echo feed back (CC 98 = 17)
_SsSetNrpnVabAttr18 ... Echo delay time (CC 98 = 18)
_SsSetNrpnVabAttr19 ... Delay delay time(CC 98 = 19)

SsGetCurrentPoint Obtain SEQ/SEP address currently read-in

Syntax

unsigned char*
SsGetCurrentPoint(short acn, short trn)

Argument

acn SEP access number
trn SEQ number within SEP
(0 when the music score data is SEQ)

Explanation

This function obtains the current read-in address for the SEQ/SEP data that is being played.

Return Value

SEP/SEQ data address

Related Function

SsSeqPlay(), SsSepPlay()

SsChannelMute Select SEQ channel and play

Syntax

void SsChannelMute(short acn, short trn, long channels)

Argument

acn SEP access number

trn SEQ number within SEP
(0 when the music score data is SEQ)

channels MIDI channel

Explanation

This function specifies MIDI channel in SEQ with 16bit upon playing SEQ. The parts specified with the channel bits can be muted. This function must be called before SsSeqOpen() or SsSepOpen().

Return Value

None

Related Function

SsSeqPlay(), SsSepPlay()

Basic Sound Library (libspu)

Changes in Header File

libspu.h: Deleted not used structure, SpuVolume16.

Bug Fix in Library

- When libspu was used alone, reverb could not be specified unless some area was allocated by SpuInitMalloc().
  In this release, this bug has been fixed.

- In SpuGetVoiceAttr(), the bug that an incorrect value was returned depending on ADSR MODE, A, S, and R has been fixed.

- In SpuWrite0() now returns correct values as in the spec.

Changes in Structure

- A member, low_priority was added to the structure, SpuStEnv.
  Setting SPU_ON to low_priority lower the streaming process priority.
  The default value is SPU_OFF (do not lower the priority).

Function Added

SpuSetEnv Sets Basic Sound Library Environment

Syntax

void
SpuSetEnv (SpuEnv *env)

Argument

env: Basic Sound Library Environment Attribute

Explanation

This function sets the Basic Sound Library environment.
Attribute can be set by setting env.mask with bitwise inclusive ORed
desired attributes. Currently, there is only one available attribute;
SPU_ENV_EVENT_QUEUEING (queue an event)

When env.mask is set to 0, all the attributes will be set.

See below for various setting;

- Queue an event
env.queueing,
SPU_ON ... Queue an event
SPU_OFF ... Do not queue an event (default)
can set either to queue or not queue an event such as Key ON/OFF,
Pitch LFO Voice Set, Noise Voice Set, and Reverb voice Set.
Default is to set immediately without queuing.

Return Value

None

Related Function

SpuSetKey(), SpuSetKeyOnWithAttr(), SpuSetPitchLFOVoice(),
SpuSetNoiseVoice(), SpuSetReverbVoice(), SpuFlush()

SpuFlush Flushes queued events

Syntax

unsigned long
SpuFlush (unsigned long ev)

Argument

ev: Event to be flushed

Explanation

This function flushes a queued event.
Set ev with bitwise inclusive ORed events to be flushed;

SPU_EVENT_KEY	Key ON/OFF
SPU_EVENT_PITCHLFO	Pitch LFO Voice Set
SPU_EVENT_NOISE	Noise Voice Set
SPU_EVENT_REVERB	Reverb Voice Set

When ev is set to SPU_EVENT_ALL, all events will be flushed.

Return Value

Bitwise inclusive ORed value of the flushed event(s).

Related Function

SpuSetEnv(), SpuSetKey(), SpuSetKeyOnWithAttr(),
SpuSetPitchLFOVoice(), SpuSetNoiseVoice(), SpuSetReverbVoice()

SpuNSetVoiceAttr Sets Voice Attributes

Syntax

void

SpuNSetVoiceAttr (int voiceNum, SpuVoiceAttr *attr)

Argument

voiceNum: Voice Number (0 - 23)

attr: Voice Attribute

Explanation

This function sets the voice attribute.

Set voice number to be obtained explicitly into voiceNum.

Set attr.mask with bitwise inclusive ORed attributes;

SPU_VOICE_VOLL Volume(Left)
SPU_VOICE_VOLR Volume(Right)
SPU_VOICE_VOLMODEL VolumeMode(Left)
SPU_VOICE_VOLMODER VolumeMode(Right)
SPU_VOICE_PITCH Interval (Pitch specification)
SPU_VOICE_NOTE Interval (Note specification)
SPU_VOICE_SAMPLE_NOTE Waveform Data Sample Note
SPU_VOICE_WDSA Waveform Data Start Address
SPU_VOICE_ADSR_AMODE ADSR Attack rate Mode
SPU_VOICE_ADSR_SMODE ADSR Sustain rate Mode
SPU_VOICE_ADSR_RMODE ADSR Release rate Mode
SPU_VOICE_ADSR_AR ADSR Attack rate
SPU_VOICE_ADSR_DR ADSR Decay rate
SPU_VOICE_ADSR_SR ADSR Sustain rate
SPU_VOICE_ADSR_RR ADSR Release rate
SPU_VOICE_ADSR_SL ADSR Sustain level
SPU_VOICE_ADSR_ADSR1 ADSR adsr1 for `VagAtr'
SPU_VOICE_ADSR_ADSR2 ADSR adsr2 for `VagAtr'
SPU_VOICE_LSAX Loop Start Address

When attr.mask is set to 0, all attributes will be set.

Refer to SpuSetVoiceAttr for attribute detail.

Return Value

None

Related Functions

SpuSetVoiceAttr(), SpuNSetVoiceAttr(),SpuRSetVoiceAttr(),
SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuSetKey(),
SpuSetKeyOnWithAttr(), SpuSetVoiceVolume(),
SpuSetVoiceVolumeAttr(), SpuSetVoicePitch(),
SpuSetVoiceNote(), SpuSetVoiceSampleNote(),
SpuSetVoiceStartAddr(), SpuSetVoiceLoopStartAddr(),
SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(),
SpuSetVoiceRR(), SpuSetVoiceSL(), SpuSetVoiceARAttr(),
SpuSetVoiceSRAttr(), SpuSetVoiceRRAttr(),
SpuSetVoiceADSR(), SpuSetVoiceADSRAttr()

SpuSetVoiceVolume Voice volume set

Syntax

void SpuSetVoiceVolume (int voiceNum,
short volumeL, short volumeR)

Argument

voiceNum: Voice Number (0 - 23)

volumeL: Volume (Left)

volumeR: Volume (Right)

Explanation

This function sets the voice volume, equivalent process to SpuSetVoiceAttr

SPU_VOICE_VOLL
SPU_VOICE_VOLR.

Thus the Volume Mode will become "Direct Mode" and the range of value that can be specified to volumeL and volumeR is equivalent to "Direct Mode" of SpuSetVoiceAttr.
If you want to specify both volume and volume mode at the same time, use SpuSetVoiceVolumeAttr.
Refer SpuSetVoiceAttr for values that can be specified in volumeL and/or volumeR.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceVolumeAttr()

SpuSetVoiceVolumeAttr Voice volume/volume mode set

Syntax

```
void    SpuSetVoiceVolumeAttr ( int voiceNum,
                                short volumeL,  short volumeR,
                                short volModeL, short volModeR )
```

Argument

voiceNum: Voice Number (0 - 23)
volumeL: Volume (Left)
volumeR: Volume (Right)
volModeL: Volume mode (Left)
volModeR: Volume mode (Right)

Explanation

This function sets voice volume and/or volume mode, equivalent process to SpuSetVoiceAttr

SPU_VOICE_VOLL
SPU_VOICE_VOLR
SPU_VOICE_VOLMODEL
SPU_VOICE_VOLMODER.

Refer SpuSetVoiceAttr for values that can be specified in volModeL, volModeR, volumeL and/or volumeR.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceVolumeAttr()

SpuSetVoicePitch Sets Interval (Pitch specification)

Syntax

```
void    SpuSetVoicePitch ( int voiceNum, unsigned short pitch )
```

Argument

voiceNum: Voice Number (0 - 23)

pitch: Interval (Pitch specification)

Explanation

This function specifies voice interval by pitch,
equivalent to
SpuSetVoiceAttr
SPU_VOICE_PITCH.
Refer SpuSetVoiceAttr for values that can be specified in
the interval by pitch specification.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr()

SpuSetVoiceNote Sets Interval (Note specification)

Syntax

void SpuSetVoiceNote (int voiceNum, unsigned short note)

Argument

voiceNum: Voice Number (0 - 23)

note: Interval(Note specification)

Explanation

This function specifies voice interval by note.,
equivalent to
SpuSetVoiceAttr
SPU_VOICE_NOTE.
Thus prior to call SpuSetVoiceNote,
SpuSetVoiceAttr SPU_VOICE_SAMPLE_NOTE or
the waveform data sample note feature for voice must be set.
Refer SpuSetVoiceAttr for values that can be specified in
the interval by note specification.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetVoiceSampleNote()

SpuSetVoiceSampleNote Sets waveform data sample note

Syntax

void SpuSetVoiceSampleNote (int voiceNum,
unsigned short sampleNote)

Argument

voiceNum: Voice Number (0 - 23)

sampleNote: Waveform data sample note

Explanation

This function sets the waveform data sample note for voice,
equivalent to
SpuSetVoiceAttr

SPU_VOICE_SAMPLE_NOTE.
Refer SpuSetVoiceAttr for values that can be specified in
the sampleNote.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetVoiceNote()

SpuSetVoiceStartAddr Sets start address of waveform data in the sound buffer

---

Syntax
void SpuSetVoiceStartAddr (int voiceNum,
unsigned long startAddr)
Argument
voiceNum: Voice number (0 - 23)
startAddr: Waveform data start address
Explanation
This function sets start address of waveform data in
the sound buffer , equivalent to
SpuSetVoiceAttr
SPU_VOICE_WDSA.
Refer SpuSetTransferStartAddr for values that can be specified in
the startAddr, waveform data start address.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetTransferStartAddr()

SpuSetVoiceLoopStartAddr Sets loop start address of waveform data in the sound buffer

---

Syntax
void SpuSetVoiceLoopStartAddr (int voiceNum,
unsigned long loopStartAddr)
Argument
voiceNum: Voice number (0 - 23)
loopStartAddr: Loop start address
Explanation
This function sets start address of waveform data in
the sound buffer , equivalent to
SpuSetVoiceAttr
SPU_VOICE_LSAX.
Refer SpuSetTransferStartAddr for values that can be specified in
the loopStartAddr, loop start address.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetTransferStartAddr()

SpuSetVoiceAR Sets ADSR Attack Rate

Syntax

void SpuSetVoiceAR (int voiceNum, unsigned short AR)

Argument

voiceNum: Voice number (0 - 23)

AR: ADSR Attack Rate

Explanation

This function sets ADSR Attack Rate in voice equivalent to, SpuSetVoiceAttr

SPU_VOICE_ADSR_AR.

ADSR Attack Rate mode becomes SPU_VOICE_LINEARIncN (Linear Increase mode) . If you want to set ADSR Attack Rate and ADSR Attack Rate Mode at the same time, use SpuSetVoiceARAttr.

Refer SpuSetVoiceAttr for values that can be specified in ADSR Attack Rate, AR.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceARAttr()

SpuSetVoiceDR Sets ADSR Decay Rate

Syntax

void SpuSetVoiceDR (int voiceNum, unsigned short DR)

Argument

voiceNum: Voice number (0 - 23)

DR: ADSR Decay Rate

Explanation

This function sets DSR Decay Rate used in the voice, equivalent to

SpuSetVoiceAttr

SPU_VOICE_ADSR_DR.

Refer SpuSetVoiceAttr for values that can be specified in ADSR Decay Rate, DR.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr()

SpuSetVoiceSR Sets ADSR Sustain Rate

Syntax

void SpuSetVoiceSR (int voiceNum, unsigned short SR)

Argument

voiceNum: Voice number (0 - 23)

SR: ADSR Sustain Rate

Explanation

This function sets ADSR Sustain Rate in voice equivalent to, SpuSetVoiceAttr
SPU_VOICE_ADSR_SR.
ADSR Sustain Rate mode becomes SPU_VOICE_LINEARDecN (Linear Decrease mode) . If you want to set ADSR Sustain Rate and ADSR Sustain Rate mode at the same time, use SpuSetVoiceSRAttr.
Refer SpuSetVoiceAttr for values that can be specified in ADSR Sustain Rate, SR.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceSRAttr()

SpuSetVoiceRR Sets ADSR Release Rate

Syntax
void
SpuSetVoiceRR (int voiceNum, unsigned short RR)
Argument
voiceNum: Voice number (0 - 23)
RR: ADSR Release Rate
Explanation
This function sets ADSR Release Rate in voice equivalent to, SpuSetVoiceAttr
SPU_VOICE_ADSR_RR.
ADSR Sustain Rate mode becomes SPU_VOICE_LINEARDecN (Linear Decrease mode) . If you want to set ADSR Release Rate and ADSR Release Rate mode at the same time, use SpuSetVoiceRRAttr.
Refer SpuSetVoiceAttr for values that can be specified in ADSR Release Rate, RR.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceRRAttr()

SpuSetVoiceSL Sets ADSR Sustain Level

Syntax
void SpuSetVoiceSL (int voiceNum, unsigned short SL)
Argument
voiceNum: Voice number (0 - 23)
SL: ADSR Sustain Level
Explanation
This function sets ADSR Sustain Level in voice equivalent to, SpuSetVoiceAttr
SPU_VOICE_ADSR_SL.
ADSR Sustain Level mode becomes SPU_VOICE_LINEARDecN

(Linear Decrease mode) . If you want to set ADSR Sustain Level and ADSR Sustain Level mode at the same time, use SpuSetVoiceSLAttr.
Refer SpuSetVoiceAttr for values that can be specified in ADSR Sustain Level, SL.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceRRAttr()

SpuSetVoiceARAttr Sets ADSR Attack Rate / Attack Rate Mode

Syntax
void SpuSetVoiceARAttr (int voiceNum, unsigned short AR, long ARmode)
Argument
voiceNum: Voice number (0 - 23)
AR: ADSR Attack Rate
ARmode: ADSR Attack Rate Mode
Explanation
This function sets ADSR Attack Rate / ADSR Attack Rate Mode used in voice, equivalent to
SpuSetVoiceAttr
SPU_VOICE_ADSR_AR
SPU_VOICE_ADSR_AMODE.
Refer SpuSetVoiceAttr for values that can be specified in ADSR Attack Rate AR and ADSR Attack Rate Mode, ARmode.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceAR()

SpuSetVoiceSRAttr Sets ADSR Sustain Rate / Sustain Rate Mode

Syntax
void SpuSetVoiceSRAttr (int voiceNum, unsigned short SR, long SRmode)
Argument
voiceNum: Voice number (0 - 23)
SR: ADSR Sustain Rate
SRmode: ADSR Sustain Rate Mode
Explanation
This function sets ADSR Sustain Rate / ADSR Sustain Rate Mode used in voice, equivalent to
SpuSetVoiceAttr
SPU_VOICE_ADSR_SR
SPU_VOICE_ADSR_SAMODE.
Refer SpuSetVoiceAttr for values that can be specified in ADSR Sustain Rate SR and ADSR Sustain Rate Mode, SRmode.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetVoiceSR()

SpuSetVoiceRRAttr Sets ADSR Release Rate / Release Rate Mode

Syntax
void SpuSetVoiceRRAttr (int voiceNum,
unsigned short RR, long RRmode)
Argument
voiceNum: Voice number (0 - 23)
RR: ADSR Release Rate
RRmode: ADSR Release Rate Mode
Explanation
This function sets ADSR Releae Rate / ADSR Release Rate Mode
used in voice, equivalent to
SpuSetVoiceAttr
SPU_VOICE_ADSR_RR
SPU_VOICE_ADSR_RMODE.
Refer SpuSetVoiceAttr for values that can be specified in
ADSR Release Rate AR and ADSR Release Rate Mode, RRmode.

Return Value
None
Related Function
SpuSetVoiceAttr(), SpuNSetVoiceAttr(),
SpuSetVoiceRR()

SpuSetVoiceADSR Sets ADSR

Syntax
void SpuSetVoiceADSR (int voiceNum,
unsigned short AR, unsigned short DR,
unsigned short SR, unsigned short RR,
unsigned short SL)
Argument
voiceNum: Voice number (0 - 23)
AR: ADSR Attack Rate
DR: ADSR Decay Rate
SR: ADSR Sustain Rate
RR: ADSR Release Rate
SL: ADSR Sustain Level
Explanation
This function sets each ADSR attribute used in the S voice,
equivalent to
SpuSetVoiceAttr
SPU_VOICE_ADSR_AR
SPU_VOICE_ADSR_DR
SPU_VOICE_ADSR_SR
SPU_VOICE_ADSR_RR

SPU_VOICE_ADSR_SL.

For Attack, Sustain, and Release Rate, Rate mode becomes as below;

Attack Rate Mode	SPU_VOICE_LINEARIncN (Linear Increase)
Sustain Rate Mode	SPU_VOICE_LINEARDecN (Linear Decrease)
Release_Rate_Mode	SPU_VOICE_LINEARDecN_(Linear Decrease)

If you want to set multiple Rate Modes at the same time, use SpuSetVoiceADSRAttr.
Refer SpuSetVoiceAttr for values that can be specified in each Rate.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(), SpuSetVoiceRR(), SpuSetVoiceSL()

SpuSetVoiceADSRAttr Sets ADSR and each Mode

Syntax

```
void    SpuSetVoiceADSRAttr ( int voiceNum,
                              unsigned short AR,
                              unsigned short DR,
                              unsigned short SR,
                              unsigned short RR,
                              unsigned short SL,
                              long ARmode,
                              long SRmode,
                              long RRmode )
```

Argument

voiceNum:	Voice number (0 - 23)
AR:	ADSR Attack Rate
DR:	ADSR Decay Rate
SR:	ADSR Sustain Rate
RR:	ADSR Release Rate
SL:	ADSR Sustain Level
ARmode:	ADSR Attack Rate Mode
SRmode:	ADSR Sustain Rate Mode
RRmode:	ADSR Release Rate Mode

Explanation

This function sets ADSR attributes and Mode, equivalent to SpuSetVoiceAttr

SPU_VOICE_ADSR_AR SPU_VOICE_ADSR_AMODE
SPU_VOICE_ADSR_DR
SPU_VOICE_ADSR_SR SPU_VOICE_ADSR_SMODE
SPU_VOICE_ADSR_RR SPU_VOICE_ADSR_RMODE
SPU_VOICE_ADSR_SL.

Refer SpuSetVoiceAttr for values that can be specified in each Rate and Rate Mode.

Return Value

None

Related Function

SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuSetVoiceADSR(), SpuSetVoiceAR(), SpuSetVoiceDR(), SpuSetVoiceSR(), SpuSetVoiceRR(), SpuSetVoiceSL(), SpuSetVoiceARAttr(), SpuSetVoiceSRAttr(), SpuSetVoiceRRAttr()

SpuNGetVoiceAttr Gets Voice Attribute

Syntax

void SpuNGetVoiceAttr (int voiceNum, SpuVoiceAttr *attr)

Argument

voiceNum: Voice Number (0 - 23)

attr: Voice Attribute

Explanation

This function obtains voice attribute.
Set voice number to be obtained explicitly
into voiceNum.
All attributes except "mask" will be returned for "attr"
structrue members.
Refer SpuSetVoiceAttr for detail of each attribute.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuSetVoiceAttr(), SpuNSetVoiceAttr(), SpuRSetVoiceAttr(), SpuSetKey(), SpuSetKeyOnWithAttr(), SpuGetVoiceVolume(), SpuGetVoiceVolumeAttr(), SpuGetVoiceVolumeX(), SpuGetVoicePitch(), SpuGetVoiceNote(), SpuGetVoiceSampleNote(), SpuGetVoiceEnvelope(), SpuGetVoiceStartAddr(), SpuGetVoiceLoopStartAddr(), SpuGetVoiceAR(), SpuGetVoiceDR(), SpuGetVoiceSR(), SpuGetVoiceRR(), SpuGetVoiceSL(), SpuGetVoiceARAttr(), SpuGetVoiceSRAttr(), SpuGetVoiceRRAttr(), SpuGetVoiceADSR(), SpuGetVoiceADSRAttr(), SpuGetVoiceEnvelopeAttr()

SpuGetVoiceVolume Gets Voice Volume

Syntax

void SpuGetVoiceVolume (int voiceNum,
short *volumeL, short *volumeR)

Argument

voiceNum: Voice number (0 - 23)

volumeL: Volume (Left)

volumeR: Volume (Right)

Explanation

This function obtains Voice Volume, equivalent to
the process to obtain the value for SpuVoiceAttr member,
volume using SpuGetVoiceAttr function.
The value obtained is valid only when the Volume Mode is
set to "Direct Mode". For other Volume Mode, the value is
undefined.
When the Volume Mode is not "Direct Mode" or both Volume and

Volume Mode need to be obtained at the same time, use
SpuGetVoiceVolumeAttr.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceVolumeAttr()

SpuGetVoiceVolumeAttr Gets Voice Volume/Volume Mode

Syntax

```
void    SpuGetVoiceVolumeAttr (int voiceNum,
                               short *volumeL,  short *volumeR,
                               short *volModeL, short *volModeR )
```

Argument

voiceNum: Voice number (0 - 23)
volumeL: Volume (Left)
volumeR: Volume (Right)
volModeL: VolumeMode (Left)
volModeR: VolumeMode (Right)

Explanation

This function obtains Voice Volume/ Volume Mode,
equivalent to the process to obtain the value for
SpuVoiceAttr members, volume and volumed using
SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceVolume()

SpuGetVoiceVolumeX Gets Current Voice Volume

Syntax

```
void    SpuGetVoiceVolumeX ( int voiceNum,
                             short *volumeXL, short *volumeXR )
```

Argument

voiceNum: Voice number (0 - 23)
volumeXL: Current Volume (Left)
volumeXR: Current Volume (Right)

Explanation

This functions obtains current Voice Volume.
This function obtains Voice Volume, equivalent to
the process to obtain the value for SpuVoiceAttr member,
volumex using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceVolume(), SpuGetVoiceVolumeAttr()

SpuGetVoicePitch Gets Interval (Pitch Specification)

Syntax

```
void    SpuGetVoicePitch ( int voiceNum,
                           unsigned short *pitch )
```

Argument

voiceNum: Voice number (0 - 23)
pitch: Interval (Pitch Specification)

Explanation

This function obtains Voice Interval (Pitch Specification), equivalent to thehe process to obtain the value for SpuVoiceAttr member, pitch using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr()

SpuGetVoiceNote Gets Interval (Note Specification)

Syntax

```
void    SpuGetVoiceNote ( int voiceNum,
                          unsigned short *note )
```

Argument

voiceNum: Voice number (0 - 23)
note: Interval (Note Specification)

Explanation

This function obtains Voice Interval (Note Specification), equivalent to the process to obtain the value for SpuVoiceAttr member, note using SpuGetVoiceAttr function.
Thus prior to call SpuSetVoiceNote, SpuSetVoiceAttr SPU_VOICE_SAMPLE_NOTE or the waveform data sample note feature for voice must be set.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuSetVoiceSampleNote(), SpuGetVoiceSampleNote()

SpuGetVoiceSampleNote Gets Waveform Data Sample Note

Syntax

```
void    SpuGetVoiceSampleNote ( int voiceNum,
                                unsigned short *sampleNote )
```

Argument

voiceNum: Voice number (0 - 23)
sampleNote: Waveform Data Sample Note

Explanation

This function obtains waveform data sample note, equivalent to the process to obtain the value for SpuVoiceAttr member, sample_note using SpuGetVoiceAttr function.

Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceNote()

SpuGetVoiceEnvelope Gets Current Envelope Value

Syntax
void SpuGetVoiceEnvelope (int voiceNum, short *envx)
Argument
voiceNum: Voice number (0 - 23)
envx: Current Envelope Value
Explanation
This function obtains the current Voice Envelope Value,
equivalent to the process to obtain the value for SpuVoiceAttr
envx, using SpuGetVoiceAttr function.
Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr()

SpuGetVoiceStartAddr Gets Start Address of Waveform Data Sound Buffer

---

Syntax
void SpuGetVoiceStartAddr (int voiceNum,
unsigned long *startAddr)
Argument
voiceNum: Voice number (0 - 23)
startAddr: Waveform Data Start Address
Explanation
This function obtains start address of Waveform Data in
the Sound Buffer, equivalent to the process to obtain the
value for SpuVoiceAttr member, addr using SpuGetVoiceAttr
function.

Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetTransferStartAddr()

SpuGetVoiceLoopStartAddr Gets loop start address of waveform data sound buffer

---

Syntax
void SpuGetVoiceLoopStartAddr (int voiceNum,
unsigned long *loopStartAddr)
Argument
voiceNum: Voice number (0 - 23)
loopStartAddr: Loop start address
Explanation
This function obtains loop start address of waveform data in the

sound buffer, equivalent to the process to obtain the value for SpuVoiceAttr loop_addr, using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetTransferStartAddr()

SpuGetVoiceAR Obtains ADSR Attack Rate

Syntax

void SpuGetVoiceAR (int voiceNum, unsigned short *AR)

Argument

voiceNum: Voice number (0 - 23)

AR: ADSR Attack Rate

Explanation

This function obtains ADSR Attack Rate used in voice. This function obtains Voice Volume, equivalent to the process to obtain the value for SpuVoiceAttr member, ar using SpuGetVoiceAttr function. The value obtained is valid only when ADSR Attack Rate Mode is set to SPU_VOICE_LINEARIncN (Linear Increase). For other ADSR Attack Rate Mode the value is undefined. When both ADSR Attack Rate Volume and ADSR Attack Rate Mode need to be obtained at the same time, use SpuGetVoiceARAttr.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceARAttr()

SpuGetVoiceDR Gets ADSR Decay Rate

Syntax

void SpuGetVoiceDR (int voiceNum, unsigned short *DR)

Argument

voiceNum: Voice number (0 - 23)

DR: ADSR Decay Rate

Explanation

This function obtains ADSR Decay Rate used in voice, equivalent to the process to obtain the value for SpuVoiceAttr member, dr using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr()

SpuGetVoiceSR Gets ADSR Sustain Rate

Syntax

void SpuGetVoiceSR (int voiceNum, unsigned short *SR)

Argument

voiceNum: Voice number (0 - 23)

SR: ADSR Sustain Rate

Explanation

This function obtains ADSR Sustain Rate in voice equivalent to, equivalent to the process to obtain the value for SpuVoiceAttr member, sr using SpuGetVoiceAttr function.
The value obtained is valid only when ADSR Sustain Rate Mode is set to SPU_VOICE_LINEARDecN (Linear Decrease mode) .
For other ADSR Sustain Rate Mode, the value is undefined.
If you want to obtain both ADSR Sustain Rate and ADSR Sustain Rate mode at the same time, use SpuGetVoiceSRAttr.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceSRAttr()

SpuGetVoiceRR Gets ADSR Release Rate

Syntax

void SpuGetVoiceRR (int voiceNum, unsigned short *RR)

Argument

voiceNum: Voice number (0 - 23)

RR: ADSR Release Rate

Explanation

This function obtains ADSR Release Rate in voice equivalent to, equivalent to the process to obtain the value for SpuVoiceAttr member, rr using SpuGetVoiceAttr function.
The value obtained is valid only when ADSR Release Rate Mode is set to SPU_VOICE_LINEARDecN (Linear Decrease mode) .
For other ADSR Release Rate Mode, the value is undefined.
If you want to obtain both ADSR Release Rate and ADSR Release Rate mode at the same time, use SpuGetVoiceRRAttr.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceRRAttr()

SpuGetVoiceSL Gets ADSR Sustain Level

Syntax

void SpuGetVoiceSL (int voiceNum, unsigned short *SL)

Argument

voiceNum: Voice number (0 - 23)

SL: ADSR Sustain Level

Explanation

This function obtains ADSR Sustain Level. equivalent to the process to obtain the value for SpuVoiceAttr member,

sl using SpuGetVoiceAttr function.

Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr()

SpuGetVoiceARAttr ADSR Attack Rate / Attack Rate Mode,ÌŽæ“¾

Syntax
void
SpuGetVoiceARAttr (int voiceNum,
unsigned short *AR, long *ARmode)
Argument
voiceNum: Voice number (0 - 23)
AR: ADSR Attack Rate
ARmode: ADSR Attack Rate Mode
Explanation
This function obtains ADSR Attack Rate / ADSR Attack Rate Mode
used in voice,
equivalent to the process to obtain the value for SpuVoiceAttr
members, ar and a_mode using SpuGetVoiceAttr function.

Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceAR()

SpuGetVoiceSRAttr Gets ADSR Sustain Rate/Sustain Rate Mode

Syntax
void
SpuGetVoiceSRAttr (int voiceNum,
unsigned short *SR, long *SRmode)
Argument
voiceNum: Voice number (0 - 23)
SR: ADSR Sustain Rate
SRmode: ADSR Sustain Rate Mode
Explanation
This function obtains ADSR Sustain Rate / ADSR Sustain Rate Mode
used in voice,
equivalent to the process to obtain the value for SpuVoiceAttr
members, sr and s_mode using SpuGetVoiceAttr function.

Return Value
None
Related Function
SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceSR()

SpuGetVoiceRRAttr Obtains ADSR Release Rate/Release Rate Mode

---

Syntax

```
void
SpuGetVoiceRRAttr ( int voiceNum,
                    unsigned short *RR, long *RRmode )
```

Argument

voiceNum: Voice number (0 - 23)
RR: ADSR Release Rate
RRmode: ADSR Release Rate Mode

Explanation

This function obtains ADSR Release Rate / ADSR Release Rate Mode used in voice,
equivalent to the process to obtain the value for SpuVoiceAttr members, rr and r_mode using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceRR()

SpuGetVoiceADSR Gets ADSR

Syntax

```
void    SpuGetVoiceADSR ( int voiceNum,
                          unsigned short *AR, unsigned short *DR,
                          unsigned short *SR, unsigned short *RR,
                          unsigned short *SL )
```

Argument

voiceNum: Voice number (0 - 23)
AR: ADSR Attack Rate
DR: ADSR Decay Rate
SR: ADSR Sustain Rate
RR: ADSR Release Rate
SL: ADSR Sustain Level

Explanation

This function obtains each ADSR attribute used in the voice,
equivalent to the process to obtain the values for SpuVoiceAttr members, ar, dr, sr, rr, and sl using SpuGetVoiceAttr function.

The value obtained are valid only when the Attack, Sustain, and Release Rate are set to the mode as below;

```
--------------------+-----------------------------------
Attack Rate Mode | SPU_VOICE_LINEARIncN (Linear Increase)
Sustain Rate Mode | SPU_VOICE_LINEARDecN (Linear Decrease)
Release_Rate_Mode | SPU_VOICE_LINEARDecN_(Linear Decrease)
--------------------+-----------------------------------
```

For other mode, the obtained values are undefined.
If you want to obtain multiple Rate Mode at the same time, use SpuSetVoiceADSRAttr.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceAR(), SpuGetVoiceDR(),SpuGetVoiceSR(),
SpuGetVoiceRR(), SpuGetVoiceSL()

SpuGetVoiceADSRAttr Gets ADSR and each Mode

Syntax

```
void    SpuGetVoiceADSRAttr ( int voiceNum,
                              unsigned short *AR,
                              unsigned short *DR,
                              unsigned short *SR,
                              unsigned short *RR,
                              unsigned short *SL,
                              long *ARmode,
                              long *SRmode,
                              long *RRmode )
```

Argument

voiceNum:	Voice number (0 - 23)
AR:	ADSR Attack Rate
DR:	ADSR Decay Rate
SR:	ADSR Sustain Rate
RR:	ADSR Release Rate
SL:	ADSR Sustain Level
ARmode:	ADSR Attack Rate Mode
SRmode:	ADSR Sustain Rate Mode
RRmode:	ADSR Release Rate Mode

Explanation

This function obtains each ADSR attribute used in the voice, equivalent to the process to obtain the values for SpuVoiceAttr members, ar, dr, sr, rr, sl, a_mode, s_mode, and r_mode using SpuGetVoiceAttr function.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(), SpuGetVoiceADSR(),
SpuGetVoiceAR(), SpuGetVoiceDR(), SpuGetVoiceSR(),
SpuGetVoiceRR(), SpuGetVoiceSL(), SpuGetVoiceARAttr(),
SpuGetVoiceSRAttr(), SpuGetVoiceRRAttr()

SpuGetVoiceEnvelopeAttr Gets current Voice Envelope value and Key ON/OFF status.

Syntax

```
void    SpuGetVoiceEnvelopeAttr ( int voiceNum,
                                  long *keyStat, short *envx )
```

Argument

voiceNum:	Voice number (0 - 23)
keyStat:	Status of Voice Envelope and Key ON/OFF
envx:	Current Envelope value

Explanation

This function obtains the current Voice Envelope value and Voice Key ON/OFF and Envelope status.

Refer SpuGetVoiceAttr for values that can be specified in
keystat, the Key ON/OFF and Envelope status.

Return Value

None

Related Function

SpuGetVoiceAttr(), SpuNGetVoiceAttr(),
SpuGetVoiceEnvelope(), SpuSetKey(),
SpuGetAllKeysStatus(), SpuRGetAllKeysStatus()

Tap Library (libtap)

Renamed Functions

Note that following libtap functions were renamed to avoid
duplicate definition errors at linkage from Release 3.5.

InitTAP	<-	InitPAD
StartTAP	<-	StartPAD
StopTAP	<-	StopTAP