

PlayStation Audio



PlayStation Audio

- ▶ Design issues
- ▶ PlayStation sound formats
- ▶ DRAM and SPU RAM management
- ▶ Common problems and questions
- ▶ New and cool in 3.6

Which Library Should You Use?

libspu or libsnd?

- ▶ Option 1- Use libspu:
 - VAGs only
 - VAG streaming
 - Less DRAM overhead for sound effects
 - libspu smaller than libsnd
 - No MIDI support

libspu or libsnd? (cont.)

- ▶ Option 2 - Use libsnd:
 - SEQs\SEPs
 - Markers to trigger callbacks
 - Multiple loop points
 - Easier to pause and play successively
 - Can be more easily interactive than VAGs
 - VABs

libspu or libsnd? (cont.)

- ▶ Option 3 - Use both:
 - Memory hit for both libraries
 - libsnd over 100K, but small modules
 - SEQ functions take over 30K
 - Vab functions take almost 50K
 - libspu - 70K
 - 2K for rarely used functions
 - 13K for VAG streaming
 - Cd audio available for either

libspu or libsnd? (cont.)

- ▶ Using both libraries
 - No need to call SpuInit(), SsInit() calls this function
 - Use SsSetReservedVoice() to reserve voices for libsnd

VAGs and VAG Streaming

VAGs

- ▶ Currently impossible to pause
- ▶ Detecting the end of a sample
 - SpuSetIRQAddr(), SpuSetIRQ()
 - Only 1 sample end can be detected at a time

VAG Streaming

- ▶ What is it? Why should I use it?
 - Stream VAGs from DRAM to SPU RAM
 - Uses less SPU RAM
 - VAGs larger than 512K
 - Multiple streams available
 - Uses less DRAM

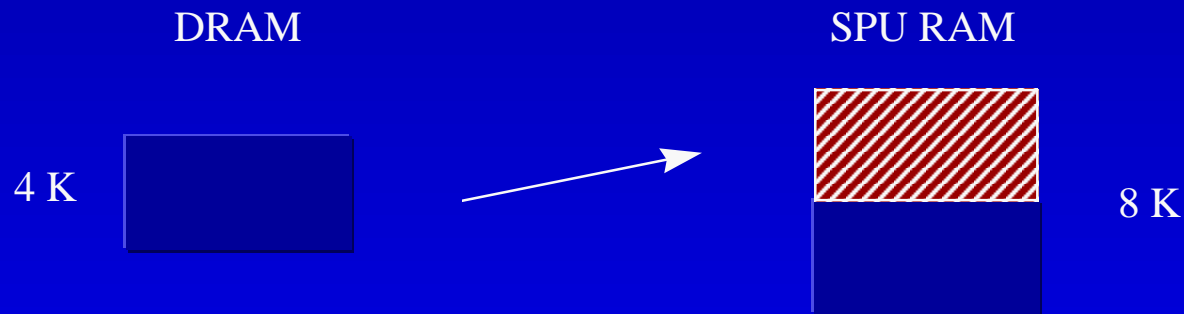
VAG Streaming Concepts

- ▶ SPU buffer system
- ▶ Both halves of SPU RAM buffer must be contiguous
- ▶ Data for each DRAM transfer need not be contiguous
- ▶ 1st half of SPU RAM buffer must be preloaded to avoid lagtime

VAG Streaming Stages

► Stage 1 - Preparation

- Setup DRAM addr, SPU RAM addr, buffer size



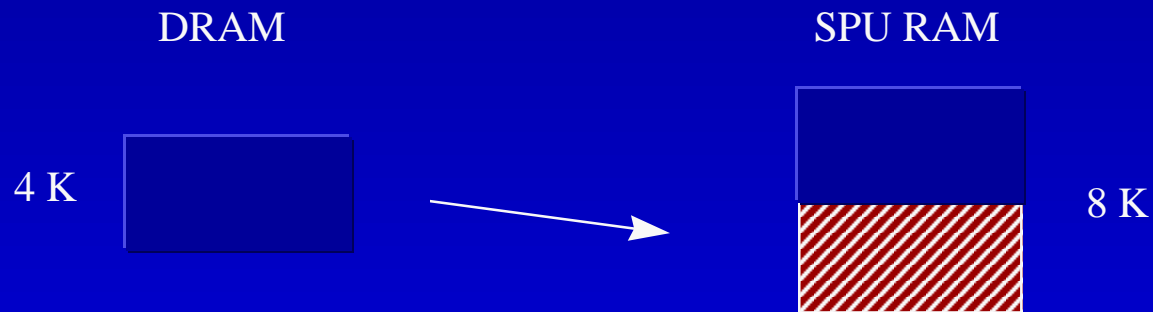
- Load 1st part of sample into 1st half of SPU RAM buffer using `SpuStTransfer()`

VAG Streaming Stages (cont.)

- ▶ Stage 1 - Preparation
 - Preparation finished callback
`SpuStSetPreparationFinishedCallback()`
 - DRAM addr of next transfer

VAG Streaming Stages (cont.)

► Stage 2 - Transfer



- Load 2nd part of sample into 2nd half of SPU RAM using `SpuStTransfer()`
- Sample automatically keyed on

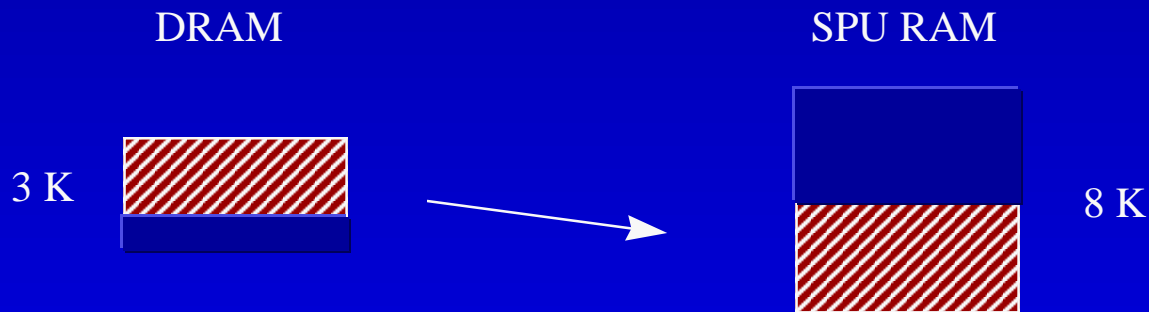
VAG Streaming Stages (cont.)

▶ Stage 2 - Transfer

- After each transfer of 1/2 buffer size, callback sets up next transfer
 - DRAM addr of next transfer
- Sample continuously loaded into SPU RAM
- Possible to prepare new streams during this stage
- Possible to terminate individual streams during this stage

VAG Streaming Stages (cont.)

► Stage 3 - Termination



- Final 1/2 buffer or smaller is to be transferred

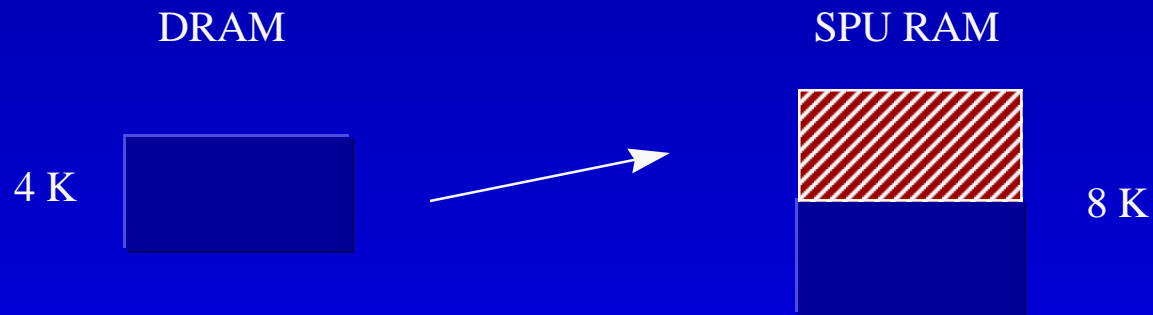
VAG Streaming Stages (cont.)

▶ Stage 3 - Termination

- Set up stream termination in transfer callback procedure
 - DRAM addr of final transfer
 - Voice status set to SPU_ST_STOP
 - Size of final transfer
- After stream termination, a new stream can begin on same voice

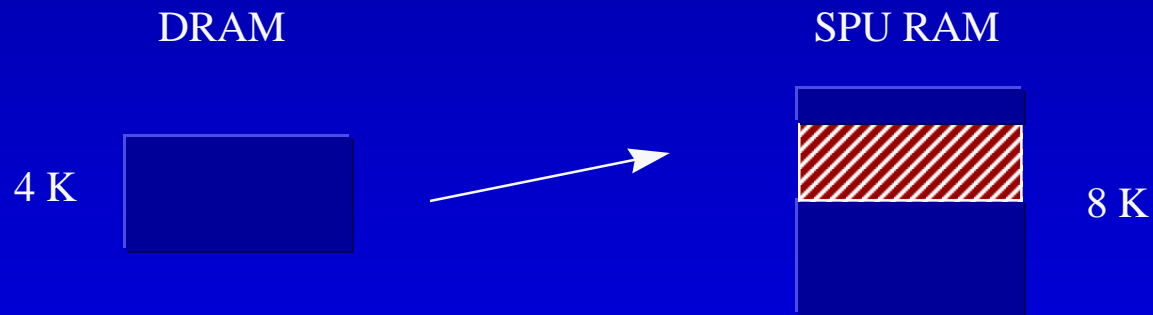
VAG Streaming Caveats

- ▶ Termination on 1st half of buffer



VAG Streaming Caveats (cont.)

- ▶ Preparation lagtime of new stream



VAG Streaming Caveats (cont.)

- ▶ 1 SPU IRQ
- ▶ VAGs cannot currently be concatenated
- ▶ Tailor buffer size
- ▶ Do NOT key off before stream termination

DRAM and SPU RAM Management

DRAM Management: .VH format

- ▶ What is a VAB exactly?
 - Waveforms
 - Tones
 - Programs
 - Header

DRAM Management: .VH format (cont.)

▶ .VH header info

Format ID ("VABp")	4 bytes
Version	4 bytes
VABID	4 bytes
File Size	4 bytes
Reserved	2 bytes
No. of Programs	2 bytes
Total No. of Tones	2 bytes
Total No. of VAGs	2 bytes
Master Volume	1 byte
Master Pan	1 byte
Reserved	6 bytes

DRAM Management: .VH format (cont.)

► .VH VAB control data

ProgAtr Table 16 bytes x 128 (Max # of programs)

 Number of tones in the program

 Volume

 Pan

VagAtr Table 512 bytes/program (32 bytes * 16 tones)

 Priority

 Volume

 Pan

 Center note and fine tuning

 Note range

 Pitch bend minimum and maximum

 Program which contains the tone

 VAG which the tone points to

VAG Offset Table 512 bytes

DRAM Management: .VH format (cont.)

▶ Overwriting unused data

- Program Attributes

- Can overwrite contiguous area from last used program through program #127
- Potential savings 2K

- VAG offset table

- Can overwrite contiguous area from last used VAG+1 through end of header
- Potential savings .5K

DRAM Management: .VH format (cont.)

▶ Overwriting unused data


- Tone Attributes

- Can overwrite unused tone area inside each program
 - Maximum contiguous area - 480 bytes
 - .5K - 60K, depending on size of .VH and setup
- Keep in mind that your safety net is gone



▶ Compacting .VH size

- Combine programs which contain a single tone where possible

DRAM Management .VH format (cont.)

70424156	07000000	00000000	00460000	 Header
EEEE0100	01000100	7F400000	FFFFFFFF	
017FA962	40000000	FFFFFFFF	FFFFFFFF	 Program #0
0000A962	00000000	FFFFFFFF	FFFFFFFF	 Program #1

{ Programs #2-#127 }

00007F40	4800007F	00000000	0000B1B2	 Tone #0
EE8CEBDC	00000100	C000C100	C200C300	
00000000	00000000	00000000	0000B1B2	 Tone #1
FF80C05F	00000000	C000C100	C200C300	

{ Other Tone Data }

00003C07	00000000	00000000	00000000	 Vag table
----------	----------	----------	----------	---

3104 bytes  84 bytes

DRAM Management: Partial Transfers

- ▶ Use `SsVabTransBodyPartly()\SpuWritePartly()`
- ▶ VAG streaming

DRAM Management: SEQ\SEP Tips

- ▶ SEQ size vs. SEP size
 - An SEP containing 1 SEQ data is only 4 bytes larger than 1 SEQ
 - Each additional SEQ data per SEP saves 2 bytes
 - So, for the 3rd-16th SEQ per SEP, you save 2 bytes
 - However...

DRAM Management: SEQ\SEP Tips (cont.)

▶ SsSetTableSize()

- Sets up a data attribute table as follows

$SS_SEQ_TABSIZ * S_MAX * T_MAX$

Where:

$SS_SEQ_TABSIZ = 172$ in lib 3.5

S_MAX = maximum SEQ+SEP data open

T_MAX = maximum SEQ data per SEP

- 88K for just the data attribute table, not counting SEQ\SEP data size!

DRAM Management: SEQ\SEP Tips (cont.)

- ▶ Reduce this size through design
 - How many samples do you need to have open at any given time?
 - How many samples do you need to have access to at any given time?
 - Reduce table slots which are blank. Make all SEPs that will be accessed at the same time contain the same number of SEP

Saving SPU RAM - A few tricks

- ▶ Looping sounds
- ▶ Pitch shifts
- ▶ Lower sample rates for low frequency sounds
- ▶ CD Audio (XA or DA)
- ▶ VAG streaming
- ▶ SsVabOpenHeadSticky()
- ▶ Reverb effect size

XA Audio

Non-traditional Uses of XA

- ▶ Don't relegate XA to FMV only
- ▶ Use the multiple channels to the best effect:
 - Create interactive sentences or phrases
 - Create a more interactive music environment

XA: Avoiding Noise

- ▶ Starting - StSetChannel() beforehand
- ▶ Stopping - Mute the CD input

XA: Detecting End of Sample

- ▶ Method 1 - Sector Location
 - Hard code location or
 - Use CdSearchFile() and file length to determine
 - Poll at regular intervals

XA: Detecting End of Sample

- ▶ Method 2 - CdData callback
 - Must have at least 1 channel of blank or data interleaved
 - CdData callback will occur every sector
 - Can determine how many callbacks occur per audio sector
 - However, callback rate will change if audio channels are varying lengths
 - CdData callback and CdRead2()

Common Problems

Common Problems

- ▶ Parameter differences - Volume & Pan
 - libspu 0-0x3fff, libsnd ranges from 0-127
 - $\text{libsnd vol} = 0x3fff * \text{master vol perct.} * \text{program vol perct.} * \text{tone vol perct.} * \text{requested vol perct.}$
 - libsnd pan then shifts the left and right volumes appropriately, with 63 being the center value

Common Problems (cont.)

▶ Voice assignment

- Some voices in key-off, envelope 0 state
- Some voices in key-off, envelope > 0 state
- All voices in key-on state

Common Problems (cont.)

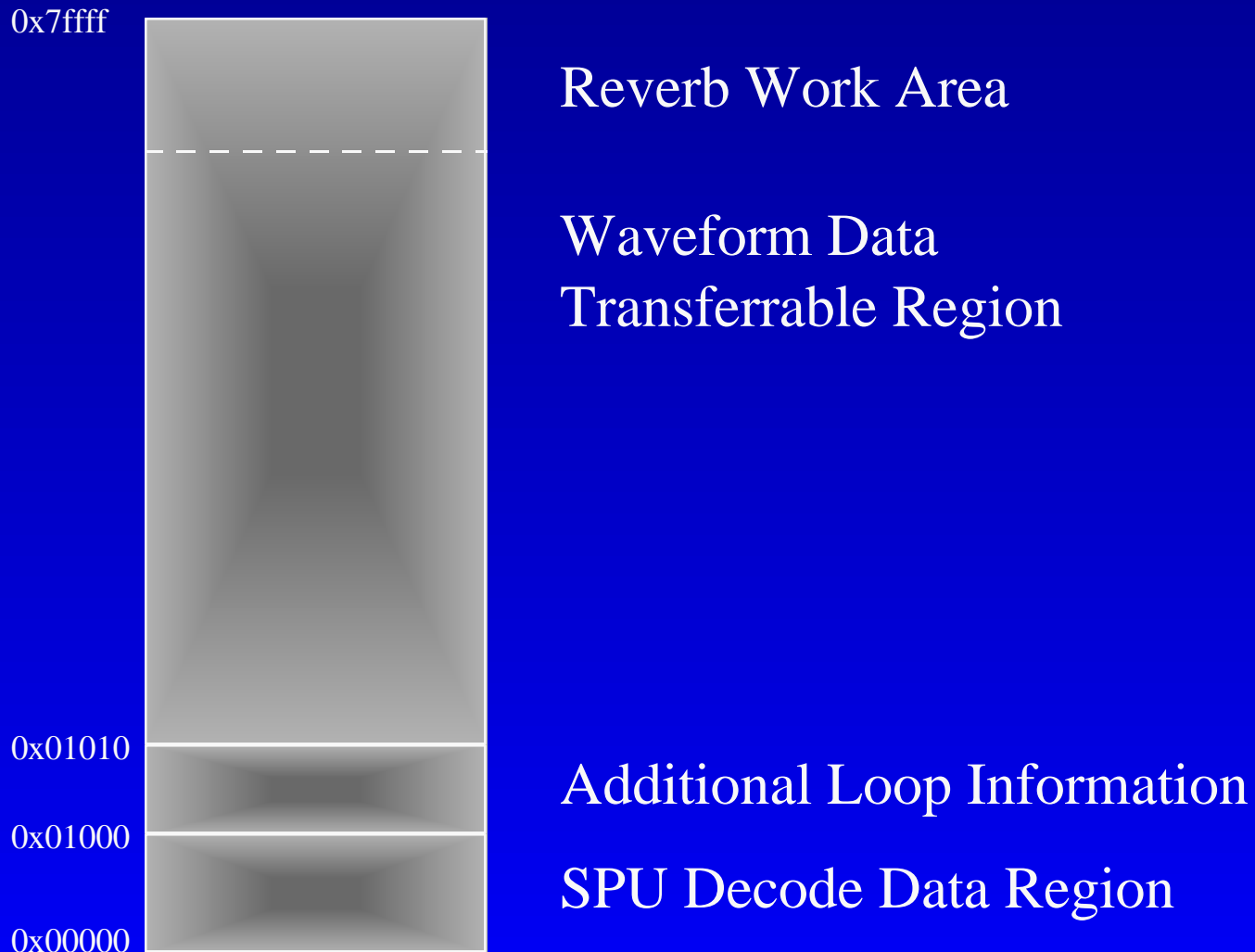
▶ Reverb loss

- `SpuSetReverbModeParam()` turns reverb off
- Reverb work area used

▶ Cd loss

- `SpuInit()` sets CD volume to 0

Common Problems (cont.)



Common Problems (cont.)

- ▶ SsVabOpen() is obsolete...so
- ▶ Instead use the trio
 - SsVabOpenHead()
 - SsVabTransBody()
 - SsVabTransCompleted()
- ▶ OR SsVabTransfer() new in 3.5

New and Cool

New and Cool in 3.6: EncSPU()

- ▶ What does it do?
 - Allows for real-time VAG creation
- ▶ Why is it cool?
 - Can combine multiple samples into 1 VAG
 - Sentence construction
 - Speech synthesis - word building
 - Can use SpuReadDecodeData() to capture CD input and encode
 - Uncompressed sample can reside in non-contiguous areas of DRAM

New and Cool in 3.6: EncSPU()

- ▶ Limitations

- Uncompressed samples will hog DRAM

New and Cool in 3.6: *SsChannelMute()*

- ▶ What does it do?
 - Selects which channels of a specific MIDI track play at any given time
- ▶ Why is it cool?
 - More interactive environment

New and Cool in 3.6: *SsSeqOpenJ() and SsSepOpenJ()*

- ▶ What does it do?
 - SsSeqPlay() analyzes MIDI status data and calls low level functions
 - Many low level functions are not usually used
 - Allows unused low level functions to be eliminated
- ▶ Why is it cool?
 - Reduces code size