Programming Sound for the PlayStation

libspu - VAG header format

- 48 byte header
  - Must be removed before transfer to SPU RAM
  - Leave header on before creating VABs from VAGs

libspu - VAG data format

- Body made up of 16 byte blocks
- First block is always all zero data
  - Avoids noise, do NOT remove
- One-shot VAGs have “SPU IRQ Clear Block”
  - Block reads “0007 0000…” or “00 07 7777…”
  - Remove this block if no IRQ functions used
  - Save 16 bytes SPU RAM per VAG

libspu - changing VAG loop start

- Looping VAG start point can be changed during playback only
- After initial loop start reached, use *loop_addr* member of SpuVoiceAttr
- Must align *loop_addr* on 16 byte boundaries

libspu - VAG loop start

- Challenge - VAG data format not public
  - VAGsize2 = VAGsize - 32 bytes (init and SPU IRQ clear blocks)
  - *loop_addr* = (VAGsize2 * Sample_new_loop_point / Sample_size) + 16[init block]
  - Need Sample_new_loop_point, Sample_size and Sample_orig_loop_point from musician

libspu - VAG loop start

- Setting *loop_addd*r<initial loop start
  - Each time initial loop start processed by SPU, the loop start address register will be reset to initial loop start address
- Setting *loop_add*dr>initial loop start
  - Allows for continual looping from the new point without continual resetting of *loop_addr*

libspu - VAG playback from midpoints

- Must be aligned on 16 byte boundaries
- Calculate *addr* member of SpuVoiceAttr using above method

libspu - pausing VAGs

- Step 1 - Save the VAG pitch via SpuGetVoiceAttr()
- Step 2 - Pitch = 0 via SpuSetVoiceAttr()
- Step 3 - Restore old pitch value via SpuSetVoiceAttr()

Encoding on the fly - EncSPU()

- Allows for conversion of AIFF2VAG at run-time
  - ENCSPUENV structure
  - AIFF data at *src* converted to VAG data at *dest*
  - Choices of
    - Endian format (*byte_swap*)
    - Looping (*loop_start* and *loop*)
    - Encoding in parts (*proceed*)
  - Return value is VAG size
  - No VAG header created
  - SPU IRQ Clear Block is removable

SPU decode data region

- 0x0-0x7FF CD left and right raw data
- 0x800-0x1000 Voice 1 and 3 raw data
- Can read back data and then re-encode that data
- SpuReadDecodeData()

libspu - note and pitch calculations

- Registers store pitch values
- Pitch = sampling_rate<<3 / 44100
- *Note* and *sample_note* converted to pitch
  - pitch = difference between *note* and *sample_note*
- Do not use *note* and *sample_note* in pre 4.0 libraries
- SpuGetVoiceAttr() vs. SpuNGetVoiceAttr()
  - SpuGetVoiceAttr() slower - calls SpuNGetVoiceAttr()

libspu - note and pitch calculations

- SpuSetVoiceAttr() vs. SpuNSetVoiceAttr() vs. SpuRSetVoiceAttr() vs. SpuSetVoice...()
  - SpuSetVoiceAttr() is the least useful - calls SpuNSetVoiceAttr() anyway
  - SpuNSetVoiceAttr() best for setting many components of 1 voice
  - SpuRSetVoiceAttr() best for setting many componenets of many voices
  - SpuSetVoice...() best for setting individual components of voices

libspu - DMA

- 2 choices for finish of DMA
  - Polling
    - SpuWrite()
    - SpuIsTransferCompleted()
  - IRQ
    - SpuSetTransferCallback()
    - SpuWrite()

libspu - noise generator

- Noise generator is white noise
- The noise generator waveform takes up no SPU RAM
- The noise generator sound may be changed by using SpuSetNoiseClock()
- Default value for noise clock is 0
- 0 value does generate some noise

libspu - noise generator

- Use SpuSetNoiseVoice() to apply noise generator to specific voices
- All noise generator voices will have same noise clock
- No SpuSetVoiceAttr() call necessary
- ADSR can be applied to noise generator, but pitch does not change

libspu - noise generator

- Pitch LFO may not be applied to noise generator
- Noise generator functions in libsnd not currently implemented

libspu - pitch LFO

- Applies the output of one voice on another as an oscillator
- Uses voice (N-1) to affect voice (N)
- Newpitch = (1+Output(N-1))*Pitch(N)

libspu - pitch LFO step by step

- Step 1: Set up voice(N) via SpuSetPitchLFOVoice()
- Step 2: SpuSetKey(SPU_ON) both waveforms
- Step 3: As volume applied to voice(N-1), LFO effect triggered

libspu - pitch LFO limits

- Cannot be applied to voice #0
- Uses up 2 available voices
- Can use noise generator as oscillation voice
- Not available in libsnd

libspu - initialization

- SpuInit() timing is crucial
  - Reverb noise may continue if SPU is not initialized
  - SpuIsTransferCompleted() may fail
- Proper order is:
  - CdInit()
  - SpuInit() /* BEFORE logo and intro movie playback */

libspu - proper reverb setup

- Most common reverb problem - noise
- Step 1: SpuSetReverbModeParam() or SpuSetReverbModeType()
  - Set *depth.left* = *depth.right* = 0
- Step 2: SpuSetReverb(SPU_ON)
- Step 3: Wait approx. 2 seconds
  - Can load sound data during this time

libspu - proper reverb setup

- Step 4: SpuSetReverbDepth() or SpuSetReverbModeParam()
- Can set *feedback* and *delay* and reverb voices at any time after this point

libspu - designing a MIDI engine

- Problem - key off and key on registers can only be checked once per 1/44100 seconds
- Rapid key on\key off messages can be missed
- libsnd handles this problem through a queueing system
- New functions from lib 3.6
  - SpuSetEnv()
  - SpuFlush()

libspu - theory of 3D sound

- Volume mode SPU_VOICE_DIRECT allows a range of -0x4000 to 0x3fff
- SPU_VOICE_DIRECT is default volume mode
- Other volume modes do not allow for negative volumes
- Use sound pairs - one front, one back
- For back sound to be effective, use one negative volume and one positive volume

How libsnd works - volume calculations

- Range of volume 0-127
- Negative values for 3D sound NOT currently supported
- 2 volumes (*voll* and *volr*) input to SsVoKeyOn() or SsUtKeyOn()
- SsUtGetVVol() and SsUtGetDetVVol() return values vary greatly from input values - Why?

How libsnd works - volume calculations

- Maxvolume = greatest of *voll* and *volr*
- VAB volume reduction factors
  - maxvol = maxvolume*Master volume VAB/127*Program volume/127*Tone volume/127
- VAB pan and input pan reduction factors
  - Reduce maximum volume of panned away from side for:
    - Program pan, tone pan, input pan (calculated from ratio of *voll* and *volr)*

How libsnd works - volume calculations

- Lastly, exponential separation of the two volumes
  - voll = voll*voll / (127*127)
  - volr = volr*volr / (127*127)

How libsnd works - volume settings

- SsUtSetDetVVol() will set the volume.left =*detvoll* & volume.right =*detvolr*
  - Valid ranges 0-0x3fff
- SsUtSetVVol() will modify *voll* and *volr* as follows
  - lvol = *voll*voll* / 127
  - rvol = *volr*volr* / 127

How libsnd works - voice allocation

- SsSetReservedVoice() = voices available
- Default is 24
- If less than 24 voices sounding, first empty slot allocated
- Once voice limit reached, priority system goes into effect
- Oldest voice with lowest priority and smallest envelope allocated

How libsnd works - the tick callback

- SsSetTickMode() sets tick callback frequency
- If tick mode = SS_TICKVSYNC, SS_TICK50 (PAL), or SS_TICK60 (NTSC)
  - Use SsStart2()
  - Sound callback hooked into Vsync callback
- If tick mode != VSYNC of system
  - Use SsStart()
  - Sound callback hooked into Root Counter 2
  - Do not change Root Counter 2 callback frequency

How libsnd works - tick callback

- If tick mode = SS_NOTICK
  - SsSeqCalledTbyT() must be called by the program
- If tick mode = (SS_NOTICK | any resolution)
  - SsSeqCalledTbyT() must be called by the program
  - 60 < resolution < 240
  - No macro usage; use actual resolution
- What tick mode should you use?
  - SS_TICK60 may be too slow
  - SS_TICK240 probably much too fast
  - SS_TICK120?

How libsnd works - SsVoKeyOn() and SsUtKeyOn()

- SsVoKeyOn(*vab_pro*, *pitch*, *voll*, *volr*)
  - Slower
  - Should be used for multiple tone sound efffects only
- Each tone of the specified program checks bits 8-15 of *pitch* vs. minimum and maximum note values as define by VAB header

How libsnd works - SsVoKeyOn() and SsUtKeyOn()

- SsUtKeyOn(*vabid*, *prog*, *tone*, *note*, *fine*, *voll*, *volr*)
  - Faster
  - Should be used for single tone sound effects
- Only one tone specified to check *note* vs. min. and max. note values as defined by VAB header

libsnd - SsVoKeyOn() problem

- SsUtKeyOn() and MIDI key on calls are wrappered
- Set up of a specific tone to be keyed on cannot be interrupted in these two functions
- SsVoKeyOn() is NOT wrappered and can be interrupted:
  - 1: SsVoKeyOn() sets up tone to be keyed on
  - 2: MIDI callback interrupts before key on
  - 3: MIDI key on sets up tone to be keyed on

libsnd - SsVoKeyOn() problem

- SsVoKeyOn() interruption cont.
  - 4: SsVoKeyOn() resumes with incorrect tone information
  - 5: Key On of MIDI tone (often looping) occurs
  - 6: Wrong noise played and sometimes loops forever

libsnd - SsVoKeyOn() workaround #1

- libsnd - SsVoKeyOn() workaround #1
- Use SsUtKeyOn() instead
- Define which tones make up which sound effects
- Call SsUtKeyOn() once for each tone

libsnd - SsVoKeyOn() workaround #2

- Create your own wrapper #1:
  - 1: void RobVoKeyOn(whatever...);
    - { vokeyon = 1;
    - SsVoKeyOn(whatever..);
    - vokeyon = 0; }
  - 2: SsSetTickMode(SS_NOTICK);
  - 3: void RobTickCallback(whatever2…);
  - { if (! vokeyon) SsSeqCalledTbyT();
    - else return; }

libsnd - SsVoKeyOn() workaround #3

- Create your own wrapper #2
- (Sw)EnterCriticalSection();
- SsVoKeyOn();
- (Sw)ExitCriticalSection();
- Be careful not to destroy interrupt context

libsnd - jump table functions

- 2 new functions SsSeqOpenJ() and SsSepOpenJ()
- libsnd provides excessive MIDI functionality in many cases
- These 2 new functions allow the user to remove unused MIDI functions

libsnd - jump table functions

- Step 1 - Determine which MIDI functions are being used
  - Method 1 - ask your musician
  - Method 2 - do it yourself
    - Use the _SsFCALL structure to set up dmy_Ss... functions
    - Call SsSeq(Sep)OpenJ()
    - Printfs will be output for each function used

libsnd - jump table functions

- Step 2 - hook in used calls only
  - Replace used dmy_Ss… functions with low-level MIDI functions
  - Remove unused dmy_Ss… functions
  - Savings of up to 7K

libsnd - seq management table

- SsSetTableSize()
- SS_TABSIZ*s_max*t_max
- SS_TABSIZ = 172
- s_max = SEQs+SEPs open at once
  - limit = 32
  - must use SsSeq(Sep)Close when limit reached
- t_max = max. SEQs in SEP - “limit” is 16

libsnd - seq management table and SEP design

- Don’t make SEPs as large as possible
  - Bloats main RAM size
    - One developer had SEP size set to 80+
    - 13K+ just for management table!
  - Slows sound callback
    - Each SEQ in each open SEP has 4 status checks even if not playing
  - Slows game init
    - SsSetTableSize() must initialize each table entry
- Shouldn’t need access to so many SEQs

libsnd - SsChannelMute() and SEQ design

- 16 MIDI channels available
- Any or all of these channels can be muted
- Muting only affects key on commands; pan, tempo, etc. unaffected

libsnd - SsChannelMute() and SEQ design

- Usage #1 - Increasing individual song variability
  - Map individual instruments to individual MIDI tracks
  - Always have main instruments unmuted
  - Use callback markers in SEQ as appropriate timing events
  - Mute/Unmute detail instruments adaptively to player performance/location etc.

libsnd - SsChannelMute() and SEQ design

- Usage #2 - Switching SEQs mid-song
  - SsSeqPlay() 2 SEQs simultaneously, one completely muted, one completely unmuted
  - Reverse muting for each SEQ at appropriate times

libsnd - SsChannelMute() and SEQ design

- Usage #3 - Switching SEQs mid-song
  - SsSeqPlay() 2 SEQs simultaneously, one detail instruments, one main instruments
  - Use callback markers in SEQ as appropriate timing events
  - Mute/Unmute detail instruments adaptively to player performance/location etc.

libsnd - SsChannelMute() and SEQ design

- Usage #4 - “Packing” SEQs
  - Setup main instruments song 1 on MIDI channel 0
  - Setup detail instruments song 1 on MIDI channel 1
  - Setup main instruments song 2 on MIDI channel 2
  - …
  - Setup detail instruments song 8 on MIDI channel 15

libsnd - SsChannelMute() and SEQ design

- Benefits and drawbacksof “packing” SEQs
  - Reduces individual song variability
  - Increases song switching choices
  - Slight increase in tick callback processing (tempo and bank changes etc.)
  - Increased access to more SEQs for a smaller memory hit
    - 1 SEQ with 8 songs saves minimum of (172*7=1204) bytes in SEQ management table

libsnd - SEQ problems (pre 4.1)

- If s_max<32, programmer must manage # of open SEQs+SEPs
  - SsSeq/SepOpen/J functions will return success
  - Then, an area after the end of your SEQ management table will be overwritten
  - Use SsSeq/SepClose() when s_max reached

libsnd - SEQ problems (pre 4.1)

- Do not call SsSeq / SepOpen / J functions from within a callback
  - If function interrupted by a similar call and only one open slot remains, evil things will happen

libsnd - altering the VAB header

- See Vabhead3.doc in Sound Prog. on web
- Adding new instruments
  - Altering the *reserved2* memeber of ProgAtr allows for new waveforms to be added to the VAB
  - Align these waveforms on 32 byte boundaries
- Changing reverb
  - Altering the *mode* member of VagAtr allows reverb to be turned on / off for individual tones
  - However, this will NOT change the reverb of currently sounding voices; see reverb problem workaround #1 to accomplish this

libsnd - pausing sounds

- Pausing SEQs is easy - SsSeqPause()
- Lookup table in libsnd contains no “0” data
- Pausing individual tones must be accomplished via libspu
  - Method 1
    - Step 1: Save oldpitch via SpuGetVoicePitch() or SpuGetVoiceAttr() *pitch*
      - Cannot use SpuGetVoiceNote() or SpuGetVoiceAttr()*note*

libsnd - pausing sounds

- Method 1 cont.
  - Step 2: Pause
    - Set pitch = 0 via SpuSetVoicePitch() or SpuSetVoiceAttr() *pitch*
  - Step 3: Unpause
    - Set pitch = oldpitch via SpuSetVoicePitch() or SpuSetVoiceAttr() *pitch*

libsnd - pausing sounds

- Method 2:
  - Step 1: Pause
    - Set pitch = 0 via SpuSetVoicePitch() or SpuSetVoiceAttr() *pitch*
  - Step 2: Unpause
    - Set note = note_keyed_on [SpuSetVoiceNote() or SpuSetVoiceAttr() *note*]
    - AND sample_note = tone.center [SpuSetVoiceSampleNote() or SpuSetVoiceAttr() *sample_note*]

libsnd - reverb setup

- Most common reverb problem - noise
- Step 1: SsUtSetReverbType()
- Step 2: SsUtReverbOn()
- Step 3: Wait approx. 2 seconds
  - Can load sound data during this time
- Step 4: SsUtSetReverbDepth()

libsnd - reverb setup

- Call SsUtSetReverbFeedback() and SsUtReverbDelay() at any time after this point
- Reverb voices determined by *mode* member of VagAtr

libsnd - initialization

- SsInit() timing is crucial
  - Reverb noise may continue from Sony boot logos if SPU is not initialized
  - SsVabTransCompleted() may fail
  - Proper order is:
    - CdInit()
    - SsInit() /*BEFORE logo and intro movie playback*/

libsnd - initialization

- SsStart() / SsStart2() timing
  - Wait until just before first key on call or SsSeqPlay()
  - Unnecessary sound callback slows down level loading times

libsnd - initialization, loading speedup

- Use sound data load times to do other work
- Suggested function ordering
  - SsInit()
  - ReverbCalls 1-2
  - SsVabOpenHead()
  - SsVabTransBody()
  - SsSetReservedVoice()
  - SsSetTableSize()
  - Jump table functions

libsnd - loading speedup

- SsSeqOpenJ()
- SsSetTickMode()
- SsSetTickCallback()
- SsVabTransCompleted(SS_WAIT_COMPLETED)
- SsStart() / SsStart2()
- ReverbCall 4
- SsChannelMute()
- SsSeqPlay() or other key on call

Using both sound libs - an overview

- Only one init call needed - SsInit()
- SsSetReservedVoice(voices)
  - Should ONLY be called pre SsStart() / SsStart2()
  - Reserves voices 0 through (voices-1) for libsnd voice management system
  - Other voices available for libspu OR
- SsUtKeyOnV()
  - Circumvent the voice allocation system and keep highest priority voices sounding

Using libsnd & libspu - reverb problem

- Most calls to SpuSetReverbVoice() will be ineffective
- libsnd sets reverb voices only when a keyon of tone with reverb occurs
- libsnd zeroes out libspu voices

Reverb problem - Workaround 1

- Create your own tick callback

```
void RobsTick()
    {
    SsSeqCalledTbyT()
    SpuSetReverbVoice()
    }
```

- Merits
  - Can add or subtract reverb at any time from spu voices
- Drawbacks
  - Extra CPU cycles
  - Creating your own tick callback

Reverb problem - Workaround 2

- Beforehand
  - Create 2 tones in VAB
  - One points to very small VAG and has mode = “04” (reverb on)
  - One points to same VAG and has mode = “00” (reverb off)

Reverb problem - Workaround 2

- Reverb On
  - Set up desired voice's SpuVoiceAttr
  - Don't call SpuSetVoiceAttr() or SpuKeyOnWithAttr() yet
  - Call SsUtKeyOnV(voice, vabid, prog, reverb_on_tone, note_in_range, fine_unimportant, 0, 0)
  - Call SsUtFlush()
  - Now call SpuSetVoiceAttr() or SpuKeyOnWithAttr()

Reverb problem - Workaround 2

- Reverb Off
  - Set up desired voice's SpuVoiceAttr
  - Don't call SpuSetVoiceAttr() or SpuKeyOnWithAttr() yet
  - Call SsUtKeyOnV(voice, vabid, prog, reverb_off_tone, note_in_range, fine_unimportant, 0, 0)
  - Call SsUtFlush()
  - Now call SpuSetVoiceAttr() or SpuKeyOnWithAttr()

Reverb problem - Workaround 2

- Merits
  - Low overhead, occasional keyon commands only
- Drawbacks
  - Cannot add or subtract reverb to currently sounding voices
  - Very small addition to VAB header and SPU RAM

Using libspu & libsnd - libspu keyon

- 2 libs use many different internal structures
- Sound keyed on via libspu - VALID libsnd functions:
  - SsUtAllKeyOff()
  - SsUtGetDetVVol()
  - SsUtGetVVol()
  - SsUtKeyOffV()
  - SsUtSetDetVVol()
  - SsUtSetVVol()

Using libspu & libsnd - libspu keyon

- Sound keyed on via libspu - INVALID libsnd functions:
  - SsUtAutoPan() [pan used in vol calcs not valid]
  - SsUtAutoVol() [as above]
  - SsUtChangeADSR() [voice verification function fails]
  - SsUtChangePitch() [as above]
  - SsUtKeyOff() [as above]
  - SsUtPitchBend() [as above]
  - SsVoKeyOff() [as above]

Using libspu & libsnd - libsnd keyon

- Sound keyed on via libsnd
  - Almost all libspu functions valid except the following
    - SpuGetVoiceAttr() [*sample_note* and *note* are not valid]
    - SpuGetVoiceNote()
    - SpuGetVoiceSampleNote()
    - SpuNGetVoiceAttr() [*sample_note* and *note* are not valid]
  - These functions ARE valid if *sample_note* and *note* members set up via SpuSetVoiceAttr(), SpuRSetVoiceAttr(), SpuSetVoiceSampleNote(), SpuSetVoiceNote() or SpuNSetAttr()

libspu & libsnd - SPU Transfers

- If using SPU transfer callback
  - Step 1: (void) SpuSetTransferCallback((SpuTransferCallbackProc)NULL)
  - Step 2: SsVabTransBody() or SsVabTransfer()

Converting sound for PAL - SEQ tempo

- Newtempo = oldtempo*60 / 50