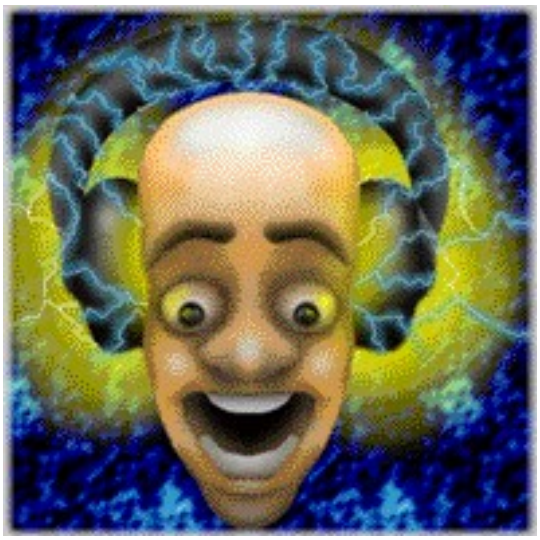


Business and Technical Contact

Steve Hales
14 Sunnyside Avenue
San Anselmo, CA 94960
office 415.258.9223
fax 415.258.9353
email hales@netcom.com

Last revision 2/8/94

SoundMusicSys
Technical Documentation

Introduction

This document describes how to orchestrate music for the Macintosh Computer using Revision 3 of the SoundMusicSys music & sound effects driver.

What software is included with the driver

- MakeMIDI - converts MIDI files to 'Midi' resources.
- AIFF to Resource - converts AIFF files (digital audio samples) to 'snd' resources.
- WAV to Resource - converts WAV files (Windows Sound files) to 'snd' resources.
- 8SXV to Resource - converts IFF-8SVX files (Amiga Sound files) to 'snd' resources.
- General MIDI Sample Library - a set of sampled instruments, so you can make music right away.
- CompressSnd - turns uncompressed 'snd' resources into compressed 'csnd' resources.
- MIDI Player - plays sound driver resource files.
- ResEdit templates, which are included for the INST and SONG resource types. The latest versions are always included in MIDI Player.

What software you need

- A music sequencer, for creating your music, and putting it in MIDI form. Vision or Studio Vision from Opcode Systems is recommended.
- A digital sample editor (if you want to create your own instruments). Sound Edit Pro or Sound Edit 16 from Macromedia.
- A resource editor (like ResEdit (version 2.1.x or later) or Resorcerer), for assembling a driver resource file.

What is the SoundMusicSys Sound Driver anyway?

The SoundMusicSys Driver plays music and sound effects simultaneously through the Macintosh speaker; it is a sequencer and sampler in software form, similar to the keyboard sequencers and hardware samplers used by musicians. For instrument patches it uses digital samples, and for music it uses MIDI files, both in the form of resources.

Resource Types

The basic resource formats are: 'snd' , 'INST', 'Midi', 'SMOD', and 'SONG.' Resource names are not utilized by the sound driver for identification, so you can name your resources as you wish.

snd/csnd Resource Types

‘snd’ resources are 8-bit digital samples used for instruments and sound effects. Type 1 and 2 ‘snd’ formats are supported. MACE data compression is not supportted instruments.

‘csnd’ resources are compress ‘snd’ resources. The CompressSnd utility asks for a source file and a destination file; it examines all ‘snd’ resources in the source file and tries to compress them. CompressSnd creates a ‘csnd’ resource for each corresponding ‘snd’ resource that is successfully compressed; when unsuccessul, it will copy the original ‘snd’ into the destination file.

You can reduce the size of your sampled instruments by looping them. A loop is a section of a sample which is repeated when a note’s duration exceeds the length of the

sample. You must use a digital sample editor (like Alchemy or SoundEditPro) to set the loop points for a sample. Save your samples in AIFF format, then convert them to resource format with the utility 'AIFF to Resource'; they must be in resource form to stick them into your driver resource with your resource editor.

Sample Loops

Loops must be at least 370 bytes long (Macintosh restriction). Larger loops obviously make for larger samples, but require less driver overhead to play long notes.

WARNING: When you play a sample above its sampled pitch, the loop shrinks (!) If you play the sample so that the loop is effectively smaller than 370 bytes, you will get clicks and pops. In general, do not play samples above their sampled frequency, unless they have large loops.

Be sure to follow Apple's restrictions that 'snd' resource ID numbers should not be lower than 4100. (The driver will function even if you don't follow this guideline).

INST Resource Type

'INST' resources tell the driver how to use 'snd' resources. This description includes the ID numbers of the 'snd' resources associated with the INST. At initialization, the sound driver loads all 'INST' resources with ID numbers from 0 to 127, and all 'snd' resources attached to them. 'INST' resources outside this range won't be loaded.

All 'snd' resources specified by the 'INST's' will be loaded also. 'INST' resources can use identical 'snd' resources, and use different characteristics such as vibrato to utilize the sound sample.

If a sound modifier (q.v. **Sound Modifier**) is specified, the driver makes a modified copy of the 'snd' resource in memory. A single 'snd' can be used by multiple 'INST' resources; if necessary, multiple copies are made in memory.

SMOD Resource Type

SMODs are 'sound modifiers', algorithms which apply envelopes to samples. SMODs do not perform real-time changes, so they do not use processor time except when they are loaded. There are four types of SMODs. Each different

volume level is associated with a program change; each program change selects a specific INST resource, which contains the SMOD parameters.

You assign two parameters to a Sound Modifier, although both may not be used (depending on the SMOD). SMOD 0 is used to create copies of samples at predefined, discrete volume levels: the driver multiplies the amplitude of the sample by the ratio of Parameter 1 to Parameter 2. Let's assume you have a song with three different sections:

• Section 1 of the music requires a soft flute. You decide that program change 0 will represent a flute at half volume. In INST resource 0 you set the two parameters for SMOD 0 to 1 and 2 (1 divided by 2 equals one half). In the MIDI data, just prior to the notes of section 1, you put a program change set to 0. When the driver sees this program change, it will use the sample of the flute at half volume.

• Section 2 of the music follows immediately after section 1, and requires that the flute be at full volume. You decide that program change 1 will represent a flute at full volume. You create INST resource 1, and set the SMOD parameter to 100 and 100 (or 0 and 0); just before section 2 in the MIDI data you put a program change set to 1. When the driver sees this program change, it will use the sample of the flute at full volume.

• For section 3 you decide that an overdriven flute would be very cool, so you create an INST rsrc, makes its ID number 2, and set the SMOD parameters to 3 and 1; now the sample will be played at three times its recorded volume.

The result of these program changes is that you have created three flute samples in memory at run time: the original flute sample is copied into memory then scaled to the appropriate volume. The driver doesn't have to scale on the fly, a process which uses a lot of CPU time.

ALWAYS assign a value of 0 to unused parameters. A value of 0 guarantees identical results for future versions of the SMODs, which may use the parameters for extra features.

MIDI Resource Type

A 'Midi' resource contains a MIDI standard file. Most sequencers are capable of writing standard MIDI files. A MIDI resource is placed in your driver resource file using a resource editor (such as ResEdit or Resorcerer). Type 0 (single track) and Type 1 (multiple simultaneous playing tracks) MIDI files are supported, although Type 0 files require less processing. Type 2 files are not supported. Use your sequencer software to split any Type 2 files, or manually break up the "MTrk" resources into separate Type 0 files.

Use the application "Make MIDI" to create a MIDI resource from a standard MIDI 1.0 file (type 0 and 1), and creates a MIDI resource that you can paste into your music application. The application "UnMake MIDI" performs the reverse process of "MakeMIDI": it extracts a MIDI file from a MIDI resource.

Opcode's Vision (Version 1.2 and beyond) allows you to export MIDI data to the clipboard. You can then paste the clipboard contents in ResEdit. (The resulting resource type 'Midi' is considered to be the same as the type 'MIDI').

SONG Resource Type

The SONG resource ties everything together conveniently for the programmer.

'SONG' resources contain a detailed description of how the song should be performed, including the number of voices and tempo.

Many SONG resources can point to same MIDI resource; this makes it possible to have one copy of a MIDI score, but multiple ways of playing it. This arrangement is very memory efficient, because SONG resources are so small.

There is overlap between the MIDI file and the SONG resource; for instance, you can set the tempo in the MIDI file, but you can also adjust it in the SONG resource. In many ways it's faster to adjust playback in the SONG resource, since you don't have to launch the sequencer, make the change, then copy and paste the MIDI resource in the resource editor. This is especially useful when you are tweaking a

song for the driver. Get in the habit of making minor adjustments to the music from the INST and SONG resources, then making the more extensive (and final) changes in your sequencing software.

If a program change is specified by the MIDI track, and the program change feature is enabled in the "SONG" resource, the instrument corresponding to the program change # will be performed, i.e. program change 43 will cause logical instrument #43 to be played.

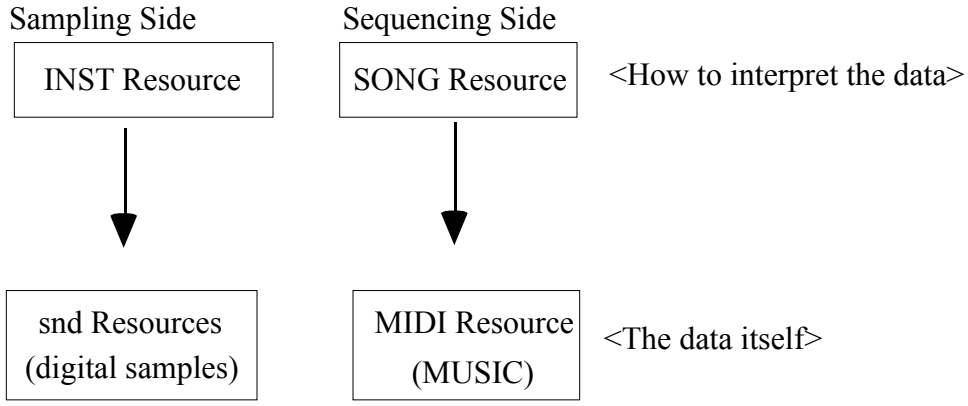
Instrument remapping should be used to assign logical instrument numbers to physical "INST" resource ID numbers. For example, you might have a MIDI program change to specify instrument #100. In this case, you could remap instrument #100 to a physical ID of, say, 5.

Instrument remapping also allows you to share instruments among several different songs. Each song could use a different mix of tracks and channels but still use the same instrument ID numbers.

PUTTING IT ALL TOGETHER

Diagram 1 denotes the relationship between 1) INST and snd resources and 2) SONG and MIDI resources.

Diagram 1



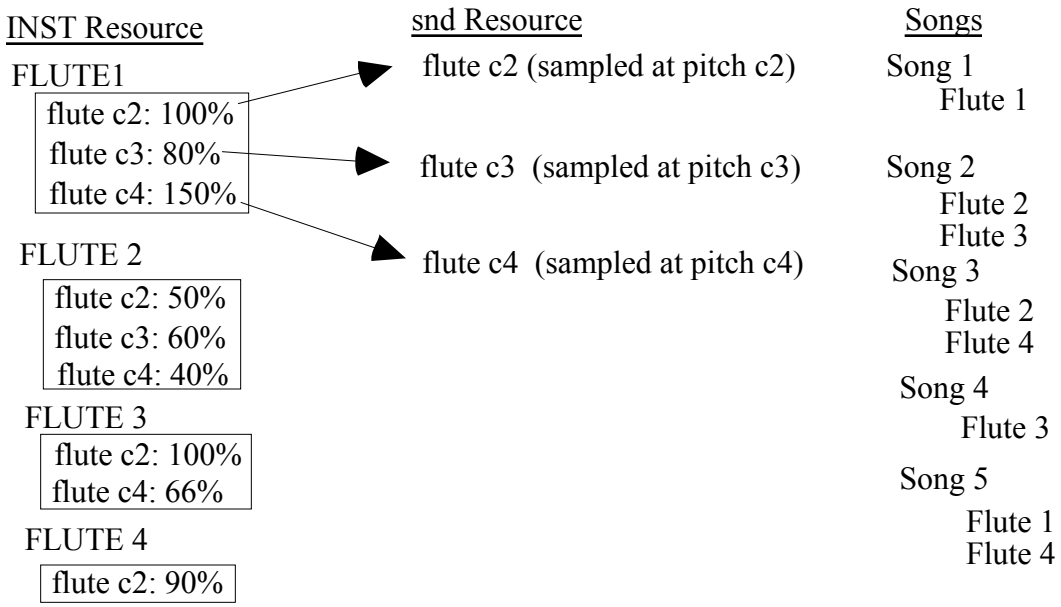
The driver has fundamentally two aspects: a sampling side, and a sequencing side. The sequencing side uses MIDI data, and interprets the MIDI data accordingly to the SONG resource. The MIDI data is similar to a piece of sheet music; it is not the sound itself, but tells the musician (in this case the driver)

what notes to play and how to play them. There is overlap between the SONG and MIDI resources in that the SONG resource has commands analogous to those in the MIDI file. The advantage to the SONG resource is that it allows us to interpret the same piece of MIDI data multiple ways. Consequently we save disk space because we don't have to make multiple copies of MIDI data (large), but multiple SONG resources (small).

The sampling side of the driver is similar to the sequencer side. The sampling side uses digital samples, and interprets those samples according to the INST resource. INST resources save space in the same way that SONG resources save space: they use

one data source, but interpret it many ways. Let's say that you had five different tunes, and three flute samples, but each tune used the flutes slightly differently. You make an INST resource for each way to play a flute sample.

DIAGRAM 2



In the above example, we have five songs, four INST's, and three flute samples of C notes, each an octave apart from each other (c3 is middle C, c2 an octave below, and c4 an octave above). Each SONG uses different flute INSTs, which in turn use different flutes samples. The percentages in the INST description are instances of SMOD 0, which tell the driver how much to scale the volume level of the sample when it is copied into memory. Rather than creating several samples on disk, we create several adjusted copies in memory at run-time.

It is easier to mix instrument volumes using INST resources than scaling the volume of the samples via a digital sample editor. For more precise volume control, you could set the note velocities in the MIDI file, but this requires that the driver use amplitude scaling (CPU drain).

INST Resource Overview (ResEdit 2.x Template parameters)

Open the SONG template and examine it as you read these descriptions.

Any template entries entitled **Reserved (Set to 0)** should be set to zero to ensure compatibility with future versions of the driver.

‘snd/csnd’ Rsrc ID # (Default)

snd (or csnd) Resource ID number. This is the default resource used when no keyboard split is specified. CSND stands for compressed sound.

Sample root key (Use 0 for default in 'snd')

Default: 0.

This is the MIDI note at which the sample has been recorded. 60 is middle C. The default (0) indicates tells the driver to use the value found in the 'snd' resource.

Interpolate if lead instrument does?

Default: 0.

1: this instrument interpolates if the lead instrument interpolates. CPU intensive.

Use note amplitude scaling?

Default: 0.

If 1, the volume of the instrument is scaled according to the attack velocity of each note in the MIDI data. This is CPU intensive.

Disable "snd " looping?

Default: 0.

If 1, then the instrument will not be looped.

Never interpolate?

Default: 0.

If 1, then this instrument will never be interpolated, despite other interpolation settings.

Play only at sampled frequency?

Default: 0.

If 1, the driver will play a sample at only its recorded pitch.

Transpose to fit key splits?

Default: 1.

If 1, then the instrument will be transposed with NotePitchShift when used in a keyboard split.

Apply Sound Modifier?

Default: 0.

If 1, then an SMOD will be applied to this instrument. See Sound Modifier Rsrc ID#. This is memory intensive. SMODs are applied when the sound is loaded, so the CPU drain happens only once.

Instrument not polyphonic?

Default: 0.

1: the instrument can be used for only one note at a time; **new notes will cut off old notes.**

Enable pitch randomness?

Default: 0.

If 1, then the driver will add a random amount of pitch to each note played with this instrument. This is used to create pitch fluctuations like those of violins and string sounds. This is CPU intensive.

Play from random keyboard splits?

Default: 0.

SMOD (Sound Modifier) Rsrc ID #

If Apply Sound Modifier? is 1, then this is the SMOD resource ID. Currently only 0-3 are supported.

SMOD parameter 1

Parameter for SMOD resources. See section about SMOD resource for details.

SMOD parameter 2

Parameter for SMOD resources. See section about SMOD resource for details.

Number of keyboard splits

Default: 0.

The farther a sample is played from its original pitch, the less that sound resembles the original instrument; key splits are used to match these timbre variations so that the software instruments sound more like physical instruments. For instance, there is a vast difference between the timbre of the highest note on the piano and the lowest. A sample of the lowest note played at the pitch of the highest note will not sound like a piano, and vice versa. To replicate these differences in timbre, you must make multiple samples, and assign each one to a range of notes (you split the keys among several samples, hence "key splits").

Keyboard splits are useful for MIDI drum kits, in which each note is mapped to a different percussion instrument.

Each keyboard split has these four parameters:

Lowest MIDI note playable
Default: 0.
Value of 0 to 127.

Highest MIDI note playable
Default: 0.
Value of 0 to 127.

snd/csnd rsrc # for range
snd or INST resource ID for this key range.

SMOD Parameter 1 (0 for no SMOD)
Default: 0.
The driver will scale the volume of the sample by the ratio of SMOD Parameter 1 to SMOD Parameter 2.

SMOD Parameter 2 (0 for no SMOD)
Default: 0.
The driver will scale the volume of the sample by the ratio of SMOD Parameter 1 to SMOD Parameter 2.

Tremolo data

Tremolo is an continuous cycling of an instrument’s pitch, usually in a range close to the note's original pitch. The tremolo feature can be used to make complicated warbling or percussion effects. The driver extracts tremolo data from the "INST" resource at a rate of 60 values per second, and stops when a tremolo value of $8000 is encountered. When each value of tremolo data is read, it is multiplied by 1/1024 of the basic note pitch, and added to the current note pitch. To "loop" (repeat) a section of the tremolo data, insert a value of $8100 minus the number of values to repeat. For a simple tremolo that wavers around the basic pitch, the data might look like this:

```
$0003  ; +3/1024
$0003  ; +3
$FFFD  ; -3
$FFFD  ; -3
```

```
$FFFD  ; -3
$FFFD  ; -3
$FFFD  ; -3
$0003  ; +3
$0003  ; +3
$0003  ; +3
$80F6  ; $8100 - ten: repeat the last ten samples
```

Don’t let the pitch vary too greatly. The driver can't always properly handle notes at unusually high or low pitches.

Tremolo data
One value of vibrato data.

Terminate vibrato with $8000!
Make sure the last entry in your vibrato data is $8000. If not, the driver will happily go hunting through memory in search of an $8000. Your instrument could sound strange, performance could suffer, or the machine could simply lock up.

Copyright
Text description of the copyright. (Optional)

Author
Text description of the song’s author. (Optional)

SMOD Overview

SMOD 0: Volume scaling.

This SMOD is the one most frequently used. Its parameters scale the volume of a sample associated with an instrument. This Modifier doesn’t do the scaling in real-time; it creates a scaled copy of the sample when the song is loaded. The driver multiplies the amplitude of the sample by the ratio of Parameter 1 to Parameter 2. Each different volume level will be identified by a program change; each program change selects a different INST resource containing the SMOD parameters.

For example:
- To play a sound 25% louder than its sampled volume, set parameter 1 to 125, and parameter 2 to 100.
- To halve the volume, set parameter 1 to 1, and parameter 2 to 2.
- When both parameters are 0, the sample is played at its recorded level.

SMOD 1; Square'O'Matic.

This algorithm squares the waveform of the sound sample, but still uses its original amplitude envelope. Parameter 1 specifies an optional ‘spike’ amount which can be used to further distort the sound, and parameter 2 sets the frequency at which the ‘spike’ is applied. Parameter 1 should be between 0 and 127; Parameter 2 can range from 0 (never) to about 10 (higher values probably won't allow the spike to kick in.)

SMOD 2; Triangulator.

Makes waveforms more triangular. Parameter 1 specifies the slope of the side of the triangle

• Values of 1 to 20 are useful. Lower numbers make straighter lines, and higher numbers mean track the original waveform more faithfully.

• Parameter 2 is not used, but should be set to 0 for compatibility with future versions.

SMOD 3; Low Pass Filter.

This algorithm reduces the unpleasant edge of instruments played at very high frequencies.

SONG Overview (ResEdit 2.x Template parameters)

Open the SONG template and examine it as you read these descriptions.

Midi/cmid resource ID

Resource number of Midi resource attached to this song. Cmid stands for compressed midi.

Lead instrument 'INST' ID

You can designate a particular instrument to be the lead instrument. The driver can give special consideration to this instrument For instance, you can turn on interpolation for just the lead INST, whether or not interpolation is enabled for the whole song.

Note: this field must contain a valid ID. You cannot leave this field as zero, unless you are using INST 0 in the song.

Tempo (or 0, default 16667) < slower, > faster

Default: 0.

A value of 0 tells the driver to use the tempo inside the MIDI resource. If you want to scale the tempo, choose numbers near 16667 (the default playback rate). Higher numbers accelerate the playing speed. (This setting doesn't replace the tempo commands in the MIDI file, just scales them).

Song pitch shift (12 is up an octave, -12 is down an octave)

Default: 0.

Transpose all notes by the indicated number of semitones.

Extra channels for sound effects

This is the number of sound tracks that are allocated for sound effects while music is playing. Sound Effects will never use Music Tracks and vice versa; in other words, if you don't allocate any channels for Sound Effects, then Sound Effects will not be played.

Max Notes

Maximum number of voices that can be played simultaneously, which is

nominally 16, but depends on the power of the target machine. A good conservative target is 4. The CPU usage by the driver should not exceed 50%, since a drain of that magnitude would disrupt operation of the computer.

Max Norm Notes

This setting is constant source of confusion, and the key to understanding its effect lies in empirical exploration, i.e. try a lot of different values and see what works best for a particular song. Small changes in MaxNormNotes can have a dramatic effect on the performance of a song.

• MaxNormNotes is the number of voices that the driver will need most of the time to play a song. MaxNormNotes should be set to an average; the number of voices used at any one time can fall below and above this average.

• MaxNormNotes represents an inverse gain level: the driver divides the dynamic range among the indicated number of Normalized voices by MaxNormNotes, so an increase in MaxNormNotes yields a decrease in the dynamic range of each voice. For playing only sound effects and no music, set Max normalized voices to 0.

• Set MaxNormNotes accordingly to maintain a consistent volume level. If MaxNormNotes is set too low, there will be frequent fluctutations in volume and in the signal-to-noise ratio. (You can prevent these drastic changes by telling the driver in the particular song resource to drop older notes when maxNormNotes is exceeded). If you set it too high, you will lose dynamic range for each instrument.

• The driver will run slightly better if MaxNotes and MaxNormNotes are the same.

Terminate decaying notes early when exceeding Max Norm Notes?

Default: 0.

When the number of notes being played exceed MaxNormNotes, the driver will stop playing the oldest notes to play new notes. If 0 then the driver will reallocate the dynamic range to accommodate the new notes. (The latter is a CPU drain).

Note interpolate whole song?

Default: 0.

If 1, then the driver will interpolate the lead INST resources and all other INST resource that allow interpolating. This is CPU intensive.

Note interpolate lead instrument?

Default: 0.

If 1, the driver interpolates the lead INST resource, which is identified in the field “Lead instrument INST ID # (see above).

Set Default programs: 0-program=channel, 1-program=track

Default: 0.

If 1, the driver assigns INST 0 to track 1, INST 1 to track 2, etc. Use this feature if you aren’t going to use program changes to choose your instruments. (This feature was originally created as a work-around for bugs in sequencing software packages).

Enable MIDI Program Change for INST settings?

Default: 0.

If 1, then the driver selects the INST resources that correspond to the value of the program changes in the MIDI resource.

Disable note click removal?

Default: 0.

If 0, the driver will remove clicks and pops from the instruments during playback. This is CPU intensive.

Global note release (in 1/60ths)

Default: 0.

This feature extends a note's duration beyond its MIDI duration. This extension is in the form of decay, and is measured in 60ths of a second. This decay is added to prevent abrupt cessations of sound; the music will sound more legato.

These extended notes obviously use up voices, so if you set Global note release too high, the number of voices will quickly exceed maxNormNotes.

Master enable: inst. pitch randomness

Default: 0.

If 1, then all INST that have "Enable pitch randomness?" enabled will have pitch randomness. This randomness is slight.

Scale lead INST when amplitude scaling enabled?

Default: 0.

When scaling for volume a 1 will allow the lead INST to be adjusted. This is CPU intensive.

Force all INSTs to use amplitude scaling if Master enable set?

Default: 0.

1: all INST resources will be adjusted for volume during playback. This is CPU intensive.

Master enable: allow note amplitude scaling?

Default: 0.

1: amplitude scaling is turned on, regardless of the settings in the individual INST resources. Amplitude scaling is CPU intensive.

"INST" Remaps:

This allows the driver to remap instrument to other instruments. Each instance of remapping specifies:

- Inst. # in song
  MIDI instrument number in the music data. 0-127.
- 'INST' res #
  Instrument number to remap to. 0-127.

Copyright

Text description of the copyright. (Optional)

Author

Text description of the song's author. (Optional)

Tuning the Driver's performance

There are several ways to tailor the driver's output to achieve the ideal CPU load for a target platform.

Here's a list of the driver features which are CPU intensive.

INST resource
- Interpolating if lead INST does.
- Using note amplitude scaling.

SONG resource
• Click Removal
• Not terminating decaying notes early when MaxNormNotes is exceeded.
• Note interpolating the whole song.
• Instrument pitch randomness.
• Amplitude scaling (Scaling lead INST when amplitude scaling is enabled, and forcing all INSTs to use amplitude scaling if Master enable is set).
• Setting MaxNormNotes too low. This can cause an enormous CPU drain, because the driver is optimized to handle the number of voices less than or equal to MaxNormNotes.

INTERPOLATION

Interpolation reduces noise by improving the resolution of sampled instruments. This enhancement costs a lot of CPU time. Set the "Do Linear" option in the "SONG" resource to 1 if you want linear interpolation for the song. Keep in mind that the same "MIDI" resource can be used by two different SONG resources: one version with interpolation turned on (for more powerful machines), and the other version without (for less powerful machines).

Interpolation can be disabled for specific instruments. Because interpolation isn't essential for noise-based (percussion) instruments, you can save CPU time by setting the "Never Interpolate?" option to 1 in its INST rsrc.

Some instruments sound noticeably worse when played without interpolation. They can be grouped along with a "lead" instrument, by setting the "Interpolate when lead instrument does?" option in the INST. Even when the SONG is set to "interpolate the lead voice only," INSTs with this option set will be interpolated also.

AMPLITUDE SCALING

Amplitude scaling is costly in CPU time (more so on 68000 machines than on 68020 or '030 based machines). Amplitude scaling can be enabled on a song-by-song basis, or by instrument, so that only instruments requiring scaling will actually use it.

Sound Modifier 0 can be used instead of amplitude scaling: this requires less CPU time, but increases the amount of RAM required at run time. Each different volume for level will be identified by a program change; each program change selects a different INST resource, which defines a volume level via SMOD 0.

INSTRUMENT USE

Play percussion instruments at their sampled pitch; a sample played at an integer multiple of its sampled frequency requires MUCH less CPU time. This is true of musical sounds as well, but pitch bend, vibrato, and pitch randomness reduce the time that the note is played at an integer multiple of its original rate.

NUMBER OF VOICES

If you choose a power of 2 (1, 2, 4, 8, or 16) for both MaxNotes AND MaxNormNotes, the driver will run slightly faster. If MaxNotes and MaxNormNotes are set to 1 the driver runs much more efficiently. One voice

may be adequate for some applications, especially if chords are recorded as samples. For instance, make a sample of a major triad; to play this chord would require a single voice, and a single MIDI note.

PLAYBACK RATE

The 11 kHz playback rate yields lower sound quality, but the CPU usage is considerably reduced, almost by half in most cases. The 11 KHz setting may be more than adequate if the song is orchestrated to avoid high pitches, which are more susceptible to aliasing and distortion at lower sampling rates.

TO DO LIST

Use this list of steps to ensure that you've set up your resource correctly.

• In the application Midi Player, create a new file by selecting 'New' from the File menu; you will put your samples and MIDI files in this new resource.

• In your sequencing software, save your music as a MIDI file (type 0 or type 1) (In Opcode's Vision you can export a MIDI file to the clipboard, then paste the clipboard contents in your resource editor, e.g ResEdit or Resorcerer).

• Convert the MIDI file to a 'Midi' resource using MakeMIDI, which comes with the driver software.

• Run your resource editor and open your new driver resource. Open your Midi resource, paste it into the Midi section, and assign it an ID #.

• Duplicate the SONG template. Open the new SONG resource and set the "Midi ID #" field to the ID # you just assigned to the new Midi resource.

• In the SONG resource:
  • set MaxNotes and MaxNormNotes accordingly. (q.v. MaxNormNotes under the explanation of the SONG resource). Don't worry too much about the settings at this stage; you'll probably have to tweak them anyway.
  • set the Lead INST ID # to an existing INST.
  • set "Enable Midi Program Changes for INST settings?" if you plan to use program changes.

• Create the instruments you'll need to play your tune. (If you don't plan to use the General MIDI library that comes with the driver). Save them as AIFF files.

• Convert those samples to resource format by using "AIFF to Resource" then paste them into your music resource with ResEdit
  This utility has a drag-and-drop feature. It doesn't replace the source files, but creates converted versions in the same directory. After you paste these resource files into your driver resource, you will want to delete the duplicate resource files, since they needlessly occupy hard disk space.

• Make enough INST resources for the snd resources.
This doesn't mean create the same number of INST resources and snd

resources. If you were making a solo piano piece for the driver, you might have 8 paino samples at different pitches (snd resources), but one INST resource with key splits. The ID numbers of the INST resources must correspond to the program change numbers in the Midi data; otherwise the driver won't know what instrument to play.

• In the INST resource, set the snd ID # fields to the corresponding snd resource ID's.

• Test your creation with MIDI Player. You have to leave your resource editor to do this.

COMMON DIFFICULTIES

SONG DOESN'T PLAY

• MIDI data
  Make sure that the very first command in every MIDI track is a program change. (No other event should occur simultaneously with The driver expects that information first since that tells the driver what samples to load into memory.
• SONG resource
  Does the SONG resource point to the right set of MIDI data, i.e. in the SONG resource, is the field "MIDI ID #" set to the ID # of the MIDI resource.

THE SONG PLAYS

<u>One of my instruments won't play (the rest will).</u>
• INST resource
  Is the volume scaled to zero? (Volume is controlled by SMOD 0)
  Does the INST point to the right snd resources?
• MIDI data
  Did you mute that instrument's channel in the sequencer?

<u>An instrument is playing the right set of notes, but in the wrong key.</u>
• The sample is probably set to the wrong root key. For instance, if you had a sample of a piano at middle C, but the root key were set to the F above Middle C, the driver will treat the sample as the F. You can set the root key with your resource editor by changing the hex data of the 'snd' resource.

<u>Set the parameters for SMOD 0, but the volume of the instrument doesn't seem to change.</u>
• You must enable Sound Modifiers by checking "Apply Sound Modifier?" in the INST resource.

<u>The volume keeps changing inappropriately</u>
• Song Resource
  MaxNormNotes is set too low, and the driver has to adjust the dynamic range of each instrument frequently.

<u>Machine keeps locking up</u>.
• INST resource

Make sure the last entry of Tremolo Data is $8000. (not 8000, which is decimal, but $8000, which is hexadecimal).

•Using too much CPU time. Try:
- running the tune on a more powerful machine.
- reducing the playback rate. (from 22kHz to 11kHz).
- reducing or turning off interpolation.
- reducing the number of voices in the SONG resource.

This is the most drastic solution, since it usually entails a rearrangment of the music; you can temporarily set MaxNotes lower, but obviously a lot of notes are going to be cut off or ignored.

Notes are not held long enough (cut short).
• SONG resource

MaxNotes is too low, or the driver is terminating old notes when MaxNormNotes is exceeded.

PERCUSSION

Percussion notes keep getting cut off.

• MIDI file

Remember that MIDI events are not treated as triggers, but as durations. If your cymbal sample is 1 second long, and a MIDI event associated with that sample has a duration of 1/10 of a second, the cymbal sample will be played for the duration of the MIDI event, 1/10 of a second.

A percussion instrument has several attacks, even though there is only one MIDI event.

• INST resource

Turn off looping in the percussion INST resource. (See the option “Disable Snd Looping?”)

OBSCURE FEATURES

MIDI format songs will play instruments that, by default, use the MIDI channel number minus 1. Alternatively, the MIDI track number can be chosen as the default instrument number for each track. If using a type 1 MIDI file, you can set up a unique track for each instrument, and remap the instrument numbers to your choices using the Instrument remapping feature.