Header formats

Mega Drive SMPS

All pointers in this section differ according to the game you are editing. Remember, relative pointers for 68k SMPS, and absolute little endian pointers for Z80 SMPS. The music header format listed is a universal format used in all incarnations of the SMPS engine except for *Knuckles' Chaotix*, which uses a slightly modified version to accommodate for the extra sound hardware (the PWM sound channels).

SMPS main header

Byte offset	Description
$00-$01	Voice table pointer.
$02	Number of FM+DAC channels. There must always be a DAC channel defined, so this is number of FM channels + 1, and the DAC channel MUST be the first one. If you don't want to use the DAC, you must put a $F2 right at the start of the track. Sonic 1 or 2 specific: There is enough track RAM for 6 FM+1 DAC. If you want to use 6 FM channels, you must put a $07 in this byte to initialize all tracks and disable the DAC as above. The DAC channel will be disabled in this case. For Sonic 2 only, DAC will be enabled during music initialization, Sonic 1 only disables it. Sonic 3 or K specific: There is enough track RAM on the z80 for 5 FM + 1 DAC. You can't use the 6th FM channel for music (you can for SFX). There is one song (the song that plays when you get a chaos emerald song) that initializes 6 FM+1 DAC as above — and it works only by luck, as the FM6 track is treated as a PSG channel and the PSG3 track is treated as an FM channel with its own voice pointer.
$03	Number of PSG tracks. If you want a noise channel, the 3rd PSG channel is the only one that can be safely converted to noise.
$04	Dividing timing: in all drivers I analyzed (Sonic 1, Sonic 2, Sonic 3, Sonic & Knuckles, Sonic 3D Blast, Ristar, Outrunners) this works by multiplying note duration by this value. This can lead to broken notes, as the final duration is stored in a single byte; thus, the maximum note duration you can use without problems is $FF/dividing timing (round down, maximum of $7F). Sonic 3 or K specific: a dividing timing of $00 multiplies the note duration by 256, making all notes have a duration of $00 and last for 256 frames.

$05	Main tempo modifier. This works as follows: Sonic 1 specific: if main tempo is nn, the song runs for nn-1 frames and is delayed by 1 frame. A main tempo of $01 runs for 0 frames and is delayed by 1 frame, hence is broken; a main tempo of $00 will overflow and run for $FF frames and be delayed by 1 frame. Sonic 2 specific: a main tempo of nn runs on nn out of 256 frames, as evenly spaced as possible. A main tempo of $00 does not run at all. Sonic 3 or K specific: a main tempo of nn runs on (256 - nn) out of 256 frames, as evenly spaced as possible. All tempo values are valid.

It is easy to convert between Sonic 2 and Sonic 3+ main tempo values:

```
S2 main tempo = (256 - S3 main tempo) mod 256
S3 main tempo = (256 - S2 main tempo) mod 256
```

The conversion works for all values of main tempo other than 0 — this value cannot be converted exactly between the two drivers. The conversion of main tempos from Sonic 1 to Sonic 2 can be done (approximately) with the following formula:

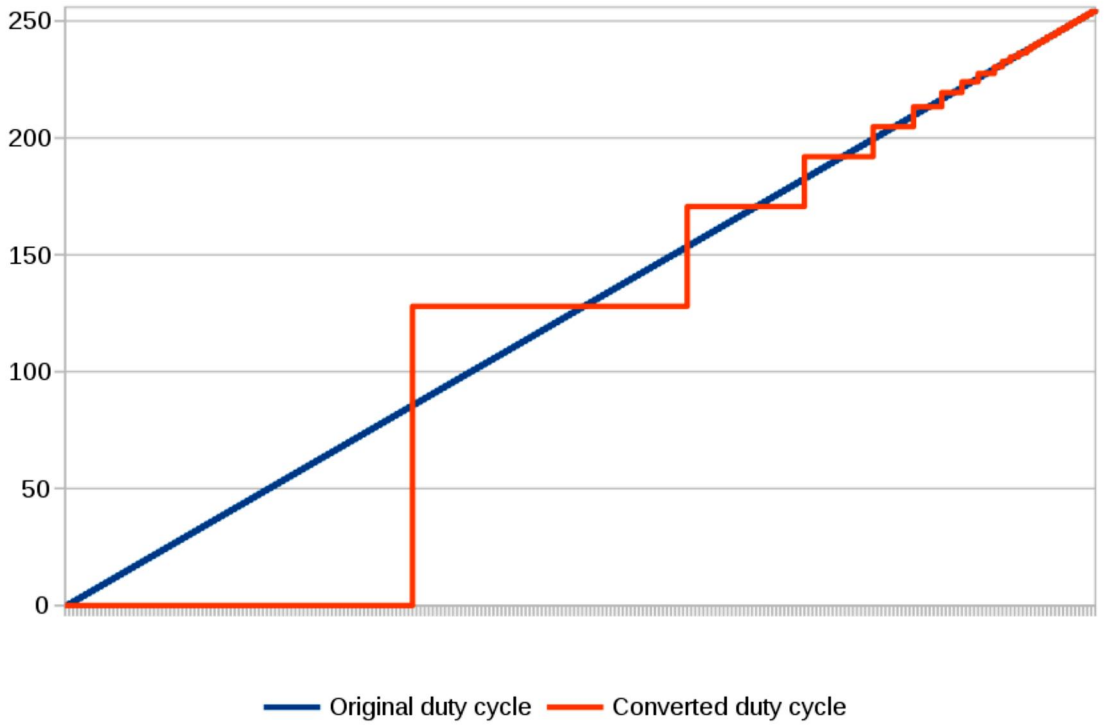
```
S2 main tempo = floor{ [ (S1 main tempo-1) * 256 + floor(S1 main tempo/2) ] /
```

Here, a S1 main tempo of 0 should be plugged as 256. There is an error intrinsic to this approximation; this error is always less than half a frame out of every 256 frames, and the average absolute error for all main tempo values is about 0.239 frames out of every 256. This means that all Sonic 1 main tempos can be converted quite accurately to Sonic 2, or to Sonic 3+ by composing the formulas.

The main tempo conversion from Sonic 2 to Sonic 1 can be done with the following formula:

```
S1 main tempo = floor{ floor[ (768 - S2 main tempo)/2 ] / (256 - S2 main tempo
```

This formula has much worse error bounds that the other formula: wast swatches of Sonic 2 main tempos are squashed to the same value because they can't be accurately represented in the Sonic 1 main tempo. This is shown in this plot:



(/File:Smps2asm-conv2s1.svg)

The "duty cycle" is how many frames the music is updated in out of every 256 frames; this illustrates how bad the formula can get for "low" S2 tempos ($C0 or less).

Then follows headers for FM and DAC channels, as many as are specified on byte $02 (offsets relative to header position in file).