Music File Format

How to do it?

Proposal 1: Text file

- on a row: CWND CWND CWND CWND
- “Channel, Waveform, Note, Duration” ex 12G4Q is channel 1, waveform 2, note G4, quarter note
- Can have commands like TEMPO = 90 and ROW=HalfNote (meaning 180 rows/minute for Tempo=90)
    - This would require auto-shutoff of notes/a polling player (could be done using Int 11h music timers)

Proposal 2 (Data-File)

MUSIC FILE FORMAT v1.0

- Header (16 bytes):
    - +0: [2] Magic: “SD” ($53 $44)
    - +2: [1] Version (1)
    - +3: [1] Flags
    - +4: [2] Tempo (ms per row)
    - +6: [1] Number of instruments
    - +7: [1] Number of patterns
    - +8: [1] Song length (orders)
    - +9: [7] Reserved
- Instrument Table (10 bytes each):
    - +0: [1] Instrument ID
    - +1: [1] Instrument Type
- Waveforms:
    - +2: [1] Waveform (WAVE_xx constant)
    - +3: [1] Pulse width / duty cycle
    - +4: [1] Attack
    - +5: [1] Decay
    - +6: [1] Sustain
    - +7: [1] Release
    - +8: [1] Default volume
    - +9: [1] Reserved
- Samples:
    - +2: [1] address (bank)
    - +3: [1] address (low byte)
    - +4: [1] address (high byte)
    - +5: [1] Volume
    - +6: [1] Playback speed

    - +7: [1] reserved
    - +8: [1] reserved
    - +9: [1] reserved

NOTE: need to do more research on what data we should include with samples. How does SNES/etc. do it?

- Order List (1 byte each):
    - Pattern numbers in playback order
    - Terminated by $FF ($FFFF or $0000?)
- Patterns:
    - Each pattern is a sequence of rows
    - Each row is a sequence of channel commands
    - Row terminated by $00
    - Pattern terminated by $FF (see above)

Row Command Format

Command Byte:

- Bits 7-6: Channel (0-3)
- Bits 5-4: Command type
- Bits 3-0: Depends on command type

OR

- Command Byte
- Channel Byte
- Data Word + Data Word + Data Word (Reserved)

Command Types:

- 00 = Note On (3 bytes total)
    - [Cmd] [Note] [Instrument]
- 01 = Note Off (1 byte)
    - [Cmd]
- 10 = Set Volume (2 bytes)
    - [Cmd] [Volume]
- 11 = Effect (3 bytes)
    - [Cmd] [Effect Type] [Effect Param]

Note data

- $00 = Rest (no change)
- $01-$60 = Notes (C0 to B7)
- $61 = Note off
- $FF = End of pattern
- Note calculation:
    - Note = (Octave * 12) + Semitone + 1

- C4 = (4 * 12) + 0 + 1 = 49 = $31
- A4 = (4 * 12) + 9 + 1 = 58 = $3A

Example Data File

```
;
================================================================================
; Simple test song: C-E-G-C chord progression
;
================================================================================

music_test_song:

; Header (16 bytes)
music_header:
    .bytes "SD"             ; Magic
    .byte 1                 ; Version
    .byte 0                 ; Flags
    .word 250               ; Tempo: 250ms per row (240 BPM)
    .byte 2                 ; 2 instruments
    .byte 2                 ; 2 patterns
    .byte 4                 ; 4 orders (play pattern 0, 1, 0, 1)
    .fill 7, 0              ; Reserved

; Instruments (8 bytes each)
music_instruments:
    ; Instrument 0: Lead (pulse wave)
    .byte WAVE_PULSE        ; Waveform
    .byte $40               ; 50% duty
    .byte $10               ; Attack
    .byte $20               ; Decay
    .byte $80               ; Sustain
    .byte $40               ; Release
    .byte $C0               ; Volume
    .byte 0                 ; Reserved

    ; Instrument 1: Bass (triangle)
    .byte WAVE_TRIANGLE     ; Waveform
    .byte 0                 ; Duty (ignored for triangle)
    .byte $08               ; Attack
    .byte $10               ; Decay
    .byte $A0               ; Sustain
    .byte $30               ; Release
    .byte $FF               ; Volume
    .byte 0                 ; Reserved

; Order list
music_orders:
    .byte 0                 ; Play pattern 0
```

```
    .byte 1                 ; Play pattern 1
    .byte 0                 ; Play pattern 0 again
    .byte 1                 ; Play pattern 1 again
    .byte $FF               ; End of song

; Patterns
music_patterns:

; Pattern 0 offset (for seeking)
music_pattern_0:
    ; Row 0: Channel 0 plays C4 with instrument 0
    .byte $00, $31, $00     ; Ch0, Note On, C4, Inst 0
    .byte $40, $19, $01     ; Ch1, Note On, C3, Inst 1 (bass)
    .byte $FF               ; End of row

    ; Row 1: Channel 0 plays E4
    .byte $00, $35, $00     ; Ch0, Note On, E4, Inst 0
    .byte $FF               ; End of row

    ; Row 2: Channel 0 plays G4
    .byte $00, $38, $00     ; Ch0, Note On, G4, Inst 0
    .byte $FF               ; End of row

    ; Row 3: Channel 0 plays C5
    .byte $00, $3D, $00     ; Ch0, Note On, C5, Inst 0
    .byte $FF               ; End of row

    .byte $FF               ; End of pattern

music_pattern_1:
    ; Row 0: Note off on channel 0, bass plays G2
    .byte $10               ; Ch0, Note Off
    .byte $40, $14, $01     ; Ch1, Note On, G2, Inst 1
    .byte $FF               ; End of row

    ; Row 1: Channel 0 plays G4
    .byte $00, $38, $00     ; Ch0, Note On, G4, Inst 0
    .byte $FF               ; End of row

    ; Row 2: Channel 0 plays B4
    .byte $00, $3C, $00     ; Ch0, Note On, B4, Inst 0
    .byte $FF               ; End of row

    ; Row 3: Channel 0 plays D5
    .byte $00, $3F, $00     ; Ch0, Note On, D5, Inst 0
    .byte $FF               ; End of row

    .byte $FF               ; End of pattern

music_end:
```