Manually Insert a Song Available on the Forum

In this tutorial, we'll be using FF3us to import a song that is already in the FF6 music format. You do not need musical knowledge to complete the following steps. The only thing that will help is being familiar with the hexadecimal system, offsets, the difference between an absolute and HiROM offset, and hex editors.

1. Getting the file and tool

We'll be using the FFIV song "The Prologue" from our Song Database. The other thing you will need is a hex editor. There are many you can choose from, but I'd suggest one that has copy selection, paste-write, and paste-insert functionalities. One good all-purpose hex editor is HxD, and this is what has been used to take the screenshots below.

2. Files we will be importing

Extract the files from **FF4_prologue.7z**, and you will see the following files:
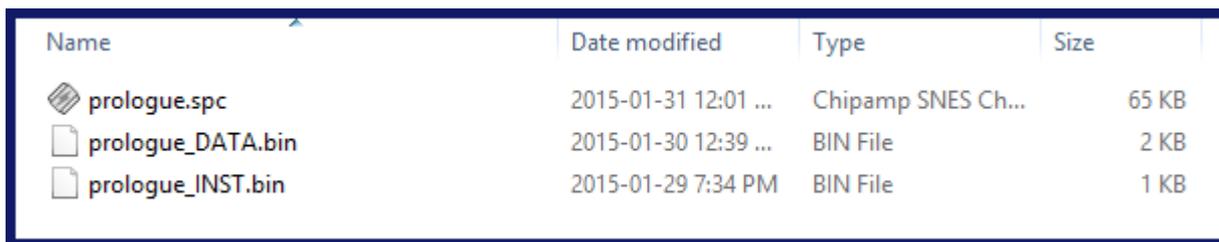
Name	Date modified	Type	Size
prologue.spc	2015-01-31 12:01 ...	Chipamp SNES Ch...	65 KB
prologue_DATA.bin	2015-01-30 12:39 ...	BIN File	2 KB
prologue_INST.bin	2015-01-29 7:34 PM	BIN File	1 KB

prologue.spc is the song in SNES format. These files can be played with an SPC player or by using winamp and a plug-in. For more info on SPC files and how to play them, use Google or check out the great extracted music tutorial at FantasyAnime.com. The DATA and INST files are both binary files; the former contains the music data in FF6 format, and the latter contains the instruments used in the song. The DATA and INST files do not contain instrument samples, as those are in the game already.

We provide the same three files for every song in our song database. Some songs (most were made by tsushiy and have a "ᵖ" next to their title) require the instrument patch, which is available in the same thread. This patch installs new BRR samples in the game, giving access to a wider range of instruments to use in songs. Some song instrument (INST) files use those added instruments. This is not the case with our prologue song, so we do not need to apply the patch in this tutorial.

3. Choosing the right spot

The first thing that you have to ask yourself is *"Do I want to replace an existing song or expand the number of existing songs?"* If you only want to replace a existing song, you can skip to section 4. You must follow several steps to expand the number of songs. First, you have to move the song pointers block at $C53E96-$C53F94 because there is no room to add an extra one. To find free space in a non-expanded ROM, you can look here. The offsets in this list take the ROM header into account, but my whole tutorial assumes that you have a headerless ROM, so you must subtract 0x200 from the offset you choose. You can also expand your ROM and put the pointers in the $FX banks.

For this example, I chose the spot at $EEAF01, which contains 767 free bytes. This is more than enough to contain the song pointers. As shown in the image below, I:

1. Select the pointer block with the mouse
2. *Ctrl+C*
3. *Ctrl+G* to jump directly to $EEAF01 (it would take a long time to reach it by scrolling)
4. *Right-Click → Paste write.*

Let's say that I want to put my song at $F30540, which is in expanded ROM. I would type **40 05 F3** at $EEB0000 (see right picture above). In this example, you would use song ID $56 to play your new song from event code. As you may have guessed, pointers are always inverted regardless of whether they are two or three bytes long. As for the old pointer data that you copied, you can overwrite it with 00 or FF since you moved it, giving you room for other things if needed.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00053E80   FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0
00053E90   FF E0 FF EC FF E0 7A 5C C8 A0 5C C8 DB 83 C9 9D
00053EA0   B4 C8 82 C8 C8 1E 64 C8 33 67 C8 69 6D C8 C5 70
00053EB0   C8 BF 74 C8 F8 78 C8 AF 7C C8 28 80 C8 38 84 C8
00053EC0   9A 88 C8 ED 8B C8 56 8F C8 6F 95 C8 29 98 C8 62
00053ED0   9B C8 D4 A5 C8 36 AD C8 B8 B7 C8 E8 BF C8 4C C2
00053EE0   C8 C1 CE C8 30 D3 C8 56 DA C8 BF DD C8 6B E1 C8
00053EF0   57 E3 C8 E2 E3 C8 48 EA C8 A6 EF C8 72 F4 C8 15
00053F00   FA C8 43 FE C8 4B 05 C9 E9 05 C9 66 0A C9 B6 90
00053F10   C9 A2 93 C9 9C 14 C9 14 8E C9 5F 97 C9 4C 1A C9
00053F20   DD 1E C9 8F 26 C9 97 29 C9 0B 2E C9 58 32 C9 FF
00053F30   37 C9 AE 3F C9 65 44 C9 B3 4A C9 6F 4D C9 16 53
00053F40   C9 DB 53 C9 C5 54 C9 57 55 C9 C9 62 C9 CD 63 C9
00053F50   03 69 C9 6E 6A C9 19 6B C9 C2 6B C9 DA 70 C9 C9
00053F60   71 C9 06 7A C9 EB 7C C9 7C 7F C9 42 88 C9 99 8C
00053F70   C9 E8 8C C9 85 8D C9 DF 97 C9 BF 9D C9 4F A2 C9
00053F80   D8 A3 C9 51 AC C9 9F AE C9 7A 5C C8 B9 B9 C9 F9
00053F90   BA C9 3F DF C9 00 00 00 00 00 00 00 00 00 00 00
00053FA0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
002EAEF0   A9 D0 50 8F 5C B8 7E A9 90 E0 8F 5E B8 7E E2 20
002EAF00   60 7A 5C C8 A0 5C C8 DB 83 C9 9D B4 C8 82 C8 C8
002EAF10   1E 64 C8 33 67 C8 69 6D C8 C5 70 C8 BF 74 C8 F8
002EAF20   78 C8 AF 7C C8 28 80 C8 38 84 C8 9A 88 C8 ED 8B
002EAF30   C8 56 8F C8 6F 95 C8 29 98 C8 62 9B C8 D4 A5 C8
002EAF40   36 AD C8 B8 B7 C8 E8 BF C8 4C C2 C8 C1 CE C8 30
002EAF50   D3 C8 56 DA C8 BF DD C8 6B E1 C8 57 E3 C8 E2 E3
002EAF60   C8 48 EA C8 A6 EF C8 72 F4 C8 15 FA C8 43 FE C8
002EAF70   4B 05 C9 E9 05 C9 66 0A C9 B6 90 C9 A2 93 C9 9C
002EAF80   14 C9 14 8E C9 5F 97 C9 4C 1A C9 DD 1E C9 8F 26
002EAF90   C9 97 29 C9 0B 2E C9 58 32 C9 FF 37 C9 AE 3F C9
002EAFA0   65 44 C9 B3 4A C9 6F 4D C9 16 53 C9 DB 53 C9 C5
002EAFB0   54 C9 57 55 C9 C9 62 C9 CD 63 C9 03 69 C9 6E 6A
002EAFC0   C9 19 6B C9 C2 6B C9 DA 70 C9 C9 71 C9 06 7A C9
002EAFD0   EB 7C C9 7C 7F C9 42 88 C9 99 8C C9 E8 8C C9 85
002EAFE0   8D C9 DF 97 C9 BF 9D C9 4F A2 C9 D8 A3 C9 51 AC
002EAFF0   C9 9F AE C9 7A 5C C8 B9 B9 C9 F9 BA C9 3F DF C9
002EB000   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

The next thing to do is to change the following code, which is the only place where these pointers are read from. You have to modify the three LDA instructions to the new base offset, the new base offset + 1, and the new base offset + 2. This is not an ASM course, so I will not explain this further except for mentioning that you have a total of 9 bytes to change.

Original code

```
C5/0538:  BF963EC5        LDA $C53E96,X  (SPC pointer low byte)
C5/053C:  8510            STA $10
C5/053E:  BF973EC5        LDA $C53E97,X  (SPC pointer middle byte)
C5/0542:  8511            STA $11
C5/0544:  BF983EC5        LDA $C53E98,X  (SPC pointer high byte)
C5/0548:  8512            STA $12
```

Modified code

```
C5/0538:  BF01AFEE        LDA $EEAF01,X  (SPC pointer low byte)
C5/053C:  8510            STA $10
C5/053E:  BF02AFEE        LDA $EEAF02,X  (SPC pointer middle byte)
C5/0542:  8511            STA $11
C5/0544:  BF03AFEE        LDA $EEAF03,X  (SPC pointer high byte)
C5/0548:  8512            STA $12
```

The last thing to do is to change the total number of songs. The offset where this value is stored is $C53C5E. It's a single byte that should be $55. Increase it by one each time you add a new song (replacing an existing one does not count). Congratulations! You have now expanded the pointers, so you can add up to 255 songs!

4. Verifying the song length

The next thing to do is to verify if your song fits in the spot that you have chosen. If you expanded the song pointers and put your song in expanded ROM, you likely won't have to consider the following, but some info is still important. Open the DATA file with HxD. The first two bytes (inverted) show the amount of space (in bytes) that the song takes up. In our case, the song is 0x044C bytes long, as seen below.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   4C 04 26 00 4C 04 26 00 E6 00 63 01 02 02 8B 02
```

You have to verify the song will not "hit" and overwrite the next song or the data after. If you want to locate your song in regular non-expanded ROM that is outside the song block, refer to the ROM map to verify this. All you have to do is add the song length to your song offset and verify that it is lower than the offset of the following data. If you are only replacing a song with *The Prelude*, check the song list. The songs are in the same order that they appear in the ROM. You need to consider the starting offset of the song following yours. I decided that we will replace *"Another World of Beasts"*.

| $21 | $C8EFA6 | Another World of Beasts |
| $22 | $C8F472 | Grand Finale #2 |

$C8F472 - $C8EFA6 = 0x04CC

We are below the $0x04CC byte limit because *The Prelude* is only $0x044C bytes long, so we are ready to insert the song!

5. Copying the song data

Now copy the song data to your ROM.

1. Select and copy the content of *prelude_DATA.bin* (*Ctrl+C*)
2. In your ROM file, press *Ctrl+G* and enter *"08EFA6"* to reach the offset where I decided to paste the song data.
3. *Right-click → Paste-write*. The two screenshots below are the beginning and end of the song pasted.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0008EF60   53 53 C4 32 4C B3 C4 1E 53 53 C4 32 4E 4E 4E 4C
0008EF70   B3 C4 1E 53 53 C4 32 47 E3 E2 01 C4 0A C5 60 46
0008EF80   E2 1F 53 E3 C5 60 0A E2 1F 53 E3 C4 32 4C B3 C4
0008EF90   1E 53 53 C4 32 4C B3 C4 1E 53 53 C4 32 47 E3 B6
0008EFA0   B6 B6 B6 F6 18 EF 4C 04 26 00 4C 04 26 00 E6 00
0008EFB0   63 01 02 02 8B 02 48 03 B9 03 E8 03 26 00 E6 00
0008EFC0   63 01 02 02 8B 02 48 03 B9 03 E8 03 F2 32 D4 D9
0008EFD0   01 F0 78 C4 3C C6 1E DC 20 D6 05 47 AC 4F B3 4F
0008EFE0   4F B3 4A 4A 4A 4A 71 AC 79 B3 79 79 B3 74 74 74
0008EFF0   74 D7 01 AC 09 B3 09 09 B3 04 04 04 04 01 AC 09
```

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0008F3B0   A9 51 4F 51 51 4F 51 4A B6 A8 A8 A8 A8 A8 A8 A9
0008F3C0   51 4F 51 51 4F 51 74 B7 4A 74 B7 51 4F 51 51 4F
0008F3D0   51 74 BA 4A BA 74 B7 51 4F 51 51 4F 51 74 B7 4A
0008F3E0   74 B7 4A 74 BA 74 BA 82 B7 51 4F 51 51 4F 51 F6
0008F3F0   02 04 B9 B7 B9 B7 B9 B7 B9 B7 B9 B7 B9 B7 B9 B7
0008F400   B9 B7 B9 B7 B9 B7 B9 B7 B9 B7 B9 B7 B9 D6 05 2B
0008F410   AB A9 AB A9 AB A9 AB 2B AB A9 AB A9 AB A9 AB F6
0008F420   E5 F3 C4 50 DC 24 C6 40 D4 C9 30 12 EF DD 0A E0
0008F430   08 B7 B9 B7 B9 D6 04 2B AB A9 AB 2B AB A9 AB 2B
0008F440   AB A9 AB 39 AB A9 AB 2B AB A9 AB 2B AB A9 AB 2B
```

Optionally, you can overwrite the rest of the old song with *00* or *FF*. That way, you can easily see where there is unused space.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0008F3E0   74 B7 4A 74 BA 74 BA 82 B7 51 4F 51 51 4F 51 F6
0008F3F0   02 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F400   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F410   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F420   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F430   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F440   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F450   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F460   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F470   00 00 A1 05 98 F4 15 FA 98 F4 A9 F5 8B F6 71 F7
```

6. Changing the song pointer

If you have to put your song elsewhere in the ROM, you need to modify the pointer. You can calculate its position with the formula below:

<Pointer Block Base Offset> + (<Song ID> * 3).

You can also do a hex search in HxD with the old offset of the song. For example, if the old offset of the song is $C8EFA6 (as with *"Another World of Beasts"*), search for *A6EFC8*.

After locating the pointer, change it to the new offset of the song (inverted).

$C53E96 + (0x21 * 0x03) = $C53EF9

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00053EE0   C8 C1 CE C8 30 D3 C8 56 DA C8 BF DD C8 6B E1 C8
00053EF0   57 E3 C8 E2 E3 C8 48 EA C8 A6 EF C8 72 F4 C8 15
00053F00   FA C8 43 FE C8 4B 05 C9 E9 05 C9 66 0A C9 B6 90
```

7. Changing the instruments

Now the other part of the job is changing the song channels' instruments. It's the same logic as calculating the offset of a song pointer, except that you multiply your song ID by 32 (0x20). The instruments file is 32 bytes in size (16 channels times 2 bytes per instrument). The first byte for each channel is the instrument ID, while the second byte is always 00.

$C53F95 + (0x21 * 0x20) = $C543B5

Once the instrument data is pasted, save your ROM. Note that you can save (*Ctrl+S*) anytime during the tutorial; it does not matter.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000543A0  00 07 00 14 00 0F 00 00 00 00 00 00 00 00 00 00
000543B0  00 00 00 00 00 0D 00 0E 00 0C 00 12 00 16 00 07
000543C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000543D0  00 00 00 00 00 08 00 0E 00 0D 00 2C 00 12 00 14
000543E0  00 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000543F0  00 00 00 00 00 0D 00 08 00 0B 00 1B 00 1E 00 02
00054400  00 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00054410  00 00 00 00 00 0D 00 0E 00 1E 00 02 00 0F 00 11
00054420  00 12 00 22 00 14 00 00 00 00 00 00 00 00 00 00
```

You'll notice that *The Prelude* has 6 instruments. As mentioned previously, some songs downloaded from the forum (not a majority) might have incorrect instruments. In such an event, you'll need to assign the correct instruments. There is a complete list of original instruments here. You'll need to identify each channel's instrument and replace the ones you think are wrong. This requires a bit of trial and error, but you'll develop your music skills. If you have instruments IDs in your file that are $40 and above, it means that your song requires the instrument patch available in the same thread as the song. The songs requiring this will have a "ᵖ" next to their name. These songs often have incorrectly assigned instruments as well because the authors sometimes add more instruments to their hacks without notifying the admins, resulting in instrument indexes above the limits of the patch.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000543A0  00 07 00 14 00 0F 00 00 00 00 00 00 00 00 00 00
000543B0  00 00 00 00 00 0D 00 0E 00 0C 00 12 00 16 00 07
000543C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000543D0  00 00 00 00 00 08 00 0E 00 0D 00 2C 00 12 00 14
000543E0  00 16 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000543F0  00 00 00 00 00 0D 00 08 00 0B 00 1B 00 1E 00 02
00054400  00 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

8. Details on song channels

One thing that differs between songs available on the forum and original songs are the channel pointers in the song. The image below shows a channel of *The Prelude*. The song loops from the end of the highlighted section to its beginning. The last command is *F6 02 04*, which loops the song by going back to the offset that is 0x0402 bytes after the start of the song. For every song in the original game, these commands are relative to their offset in the ROM, so this F6 command would look similar to *F6 8E F3* ($C8F38E) instead. However, it would not be exactly this because a song don't redo the first channel commands. I wanted to talk about this because such a tutorial would not have been possible with an unmodified original song. If you moved or inserted a song, you would have needed to change every song loop command and channel pointer! By having the custom song channel pointers start at 00 00 and not the song offset's last two bytes, we make the process a lot easier.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0008F380   A8 A8 A8 7E B6 7E B6 7E 7E 7E B6 F6 CD 03 D4 D9
0008F390   01 C4 50 C6 40 DC 24 B6 A8 A8 A8 A8 A8 A8 A9 AC
0008F3A0   D6 03 51 4F 51 51 4F 51 4A B6 A8 A8 A8 A8 A8 A8
0008F3B0   A9 51 4F 51 51 4F 51 4A B6 A8 A8 A8 A8 A8 A8 A9
0008F3C0   51 4F 51 51 4F 51 74 B7 4A 74 B7 51 4F 51 51 4F
0008F3D0   51 74 BA 4A BA 74 B7 51 4F 51 51 4F 51 74 B7 4A
0008F3E0   74 B7 4A 74 BA 74 BA 82 B7 51 4F 51 51 4F 51 F6
0008F3F0   02 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0008F400   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Below, there is a loop command of an original song pointing to $C8F9B9. In the case of a custom song, this command would be something like F6 0X XX instead, likely a number below 0x0A00.

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0008F9E0   9E D7 3C D8 9E 4A B7 4A 58 B9 58 5B E2 07 74 D7
0008F9F0   2E D8 E3 E3 BA 9E BA 9E BA 9E BA 9E 7E 9A D7 0E
0008FA00   D8 7E BA 9E BA 9E BA 9E BA 9E D7 00 D8 8C 62 D7
0008FA10   00 D8 F6 B9 F9 2C 04 3B FA 43 FE 3B FA BC FA 35
0008FA20   FB CD FB 62 FC C3 FC 28 FD 99 FD 3B FA BC FA 35
0008FA30   FB CD FB 62 FC C3 FC 28 FD 99 FD F0 49 F4 F8 F7
```

9. Conclusion

This tutorial was aimed at beginners. The rest of music hacking is harder and require some practice. However, it is within the abilities of anyone who has the time and patience to learn. To serve as a conclusion, I've made a small video that shows the import and simultaneously pays a little tribute to those who contributed to the song database.

10. Result in-game

sd-result.mp4