

# Insert Manually a Song Available on the Forum




In this tutorial we'll be importing in FF3us a song that is already in FF6 music format. There is no need for musical knowledge to be able to complete the following steps. The only things that will help is being familiar with the hexadecimal system as well as some basic ROM hacking knowledge like what is an offset, the difference between an absolute and HiROM offset and how to use an hex editor.

## 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 a 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. Those can be played with a SPC player or with winamp and a plug-in. For more info on SPC file and how to play them, use Google or check the great [extracted music tutorial](#) at FantasyAnime.com. **prologue\_DATA.bin** and **prologue\_INST.bin** are two binary files, the first has the music data in FF6 format and the other the instruments used in the song. The DATA and INST files do not contain instrument samples, those are in the game.

All songs in our song database have the three same files. Some songs, mostly done by tsushiy and having a "P" next to their title need the instrument patch, available in the same thread. This patch install new BRR samples in the game, giving access to a wider range of instruments to use in songs. Some songs instrument file use those added instruments. This is not the case of our prologue song.

## 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. To expand the number of songs there are few thing to do. First you have to move the \$C53E96-\$C53F94 song pointers block because there is no room to add an extra one. To find a spot in non-expanded ROM, you can look [here](#). The offsets in this list take in account the ROM header though while my whole tutorial assume you have a headerless ROM, so substract 0x200 from the offset you choose. You can also expand your ROM and put the pointers in the \$FX banks.

I decided for the example to take the \$EEAF01 spot, which has 767 free bytes, which is more than enough. As shown in the image below, I select with the mouse the pointer block, I do *Ctrl+C*, *Ctrl+G* to enter my destination (you can scroll a long time for the same result) and then *Right-Click→Paste write*. Now let's say I would put my song at \$F30540 in expanded ROM, I would type **40 05 F3** at \$EEB0000 (see right picture). You would use song ID \$56 for your new song in game events as an example. As you have guessed pointers are always inverted whether they are two or three bytes long. As for the old pointer data copied, you can blank with 00 or FF all the data 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
002EAF00	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
002EAF00	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. It's the only place pointers are read and you have to modify each LDA instruction so new base offset, new base offset + 1, new base offset + 2. This is not an ASM course so I'm not explaining further except only mentioning 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
```

Last thing to do is changing the total number of songs. The offset is \$C53C5E. It's a single byte that should be \$55. Increase it of one each time you add a new song (replacing an existing one does not count). So congratulation you have not expanded pointer and can add as much songs as you want up to a limit of 255!

## 4. Verifying the song length

The next thing to do is verifying if your song fit in the spot you have chosen. If you expanded the song pointers and put your song in expanded ROM, you'll likely won't have to consider the following but some info is still important. Open the **prelude\_DATA.bin** file with HxD. The first two bytes (inverted) are the amount of bytes the song take. In our case it is 0x044C bytes, 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” the next song or the data after. If you locate your song in regular non-expanded ROM but outside the songs block, refer to the [ROM map](#) to verify. All you have to do is add to your song offset the song length and verify it is lower than next data. If you only replace a song by *The Prelude*, check the [song list](#). Song are in order that they appear in the ROM. You need to consider the song following yours. I decided we are replacing “*Another World of Beast*”.

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

**\$C8F472 - \$C8EFA6 = 0x04CC**

We are below the limit with our 0x044C bytes for *The Prelude* so we are ready to insert the song!

## 5. Copying the song data

Copy all the content of **prelude\_DATA.bin** (*Ctrl+C*) go to your ROM do *Ctrl+G* and enter "08EFA6". You'll land on the offset you need to paste the song data (*Right-click→Paste-write*). The below two screenshots are the beginning and ed 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

You can optionally put *00* or *FF* at the place of the rest of the old song. This 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 and you can

calculate its position will the below formula. It's simply pointers base offset + (song ID multiplied by 3). The other method is doing a hex search in HxD with the old offset of the song. In the case of as an example \$C8EFA6, search for A6EFC8.

$$\text{\$C53E96} + (0\text{x21} * 0\text{x03}) = \text{\$C53EF9}$$

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 to calculate the beginning of a song instruments, except you multiply your song ID by 32 (0x20). The instruments file is 32 bytes in size (16 channels times 2 bytes per instruments). First byte is instrument ID, second byte is always 00.

$$\text{\$C53F95} + (0\text{x21} * 0\text{x20}) = \text{\$C543B5}$$

Once the instruments 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 *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 instrument and replace what you think are the wrong ones. A bit trial and error but you'll develop your music skills. If you have instruments IDs in your file from \$40 and above it mean your song require the instrument patch available in the same thread as the songs. The songs requiring will have a "P" next to their name. The wrong assigned instruments are often those, because the author added more instruments in his hack without noticing the admins, resulting in instrument indexes above the limit 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

A thing that differs between songs available on the forum and original songs are the channel pointers in the song. Below is a channel of *The Prelude* and the song at the end loop toward the beginning of the blue section. The last command is F6 02 04 which mean continue by going back to 0x0402 byte after the start of the song. Every song in the original game have these command relative to their offset in the ROM meaning this F6 command would look like as F6 8E F3 (\$C8F38E). It would not be exactly this because a song don't redo the first channel commands twice but I talk about this because such a tutorial would not have been possible with an unmodified original song, because when you move it you need to change all the song loop commands and channel pointers at beginning of song data. By having the custom song channel pointers start at 00 00 and not the song offset last two bytes, we easy the process a lot.

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	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, a loop command of an original song pointing at \$C8F9B9. In the case of a custom song, we would see more or less F6 0X YZ, likely something 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 in the range of anyone having the time and patience to learn. As a conclusion I've made a small video showing the import and at the same time did a little tribute to those who contributed to the song database.

## Result in-game

[sd-result.mp4](#)

From:

<https://www.ff6hacking.com/wiki/> - **ff6hacking.com wiki**

Permanent link:

<https://www.ff6hacking.com/wiki/doku.php?id=ff3:ff3us:tutorial:music:songdata&rev=1519310915>

Last update: **2019/02/12 10:43**

