Adding a new BRR Sample

In this tutorial we'll be using FF3us to add a new BRR sample that can be used in songs. You do not need extended musical knowledge to complete this tutorial. One thing that will help is being familiar with the hexadecimal system, the concept of offset, the difference between an absolute and HiROM offset and hex editors.

1/6

1. Getting the file and tool

We'll be using the FF5 bass drum sample from our BRR Sample Database. The only thing that 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. Finally make sure you have a 1.0 or 1.1 FF3us ROM that is expanded either to 28Mbit or 32Mbit.

2. File we will be importing

Select the second download on the FF5 page, the one labeled "BRR Samples with first two bytes as sample length. Pitch, loop and ADSR data are in a text file.". Extract the archive and you are done for now.

3. Changing the Code

Since each sample has data attached to it, we need to move this data first to make some room for the new sample data. What need to be moved is loop start positions, pitch multipliers and ADSR data. The pointers to the BRR data do not need to be relocated, for the simple reason that we will free the space right after them, thus leaving room to add more.

C53C5F	C53D1B	Pointers to Instrument BRR Data (3 bytes each, absolute)
C53D1C	C53D99	Instrument Loop Start Positions (63 items, 2 bytes each)
C53D9A	C53E17	Instrument Pitch Multipliers (63 items, 2 bytes each)
C53E18	C53E95	Instrument ADSR Data (63 items, 2 bytes each)

We will put the loop starting positions at \$F20000, the pitch mutipliers at \$F20200 and finally the ADSR data at \$F20400. This leave enough room for the maximum of 256 samples.

There are 3 ASM instruction to modify, more precisely the offset that these instruction carry. Below is the original and modified code. Open HxD and press Ctrl+G, that will open a window. Type 05041C and press "Ok". You are not at \$C5041C. You need to enter 0000F2 (\$F20000 inverted). We do not touch the 1st byte of the instruction, only bytes 2,3,4. Repeat a similar process for \$C5049C and \$C504DE.

Original code

C5/041B: 7F1C3DC5 ADC \$C53D1C,X C5/049B: BF9A3DC5 LDA \$C53D9A,X C5/04DD: BF183EC5 LDA \$C53E18,X	(pitch multipliers)
--	---------------------

Modified code

C5/041B:	7F0000F2	ADC	\$F20000,X	
C5/049B:	BF0002F2	LDA	\$F20200,X	
C5/04DD:	BF0004F2	LDA	\$F20400,X	

(loop starting positions) (pitch multipliers) (ADSR data)

4. Moving the Data

Let's move first the loop starting positions. Select the data from \$C53D1C to \$C53D99 as shown on the left below and press Ctrl+C. Press Ctrl+G and enter 320000 (\$F20000 in HiROM offset). Right click and press "*paste write*". The result should be like the right screenshot:

00053D00	C7	C8	CF	C7	C3	E5	C7	53	F4	C7	C5	05	C8	DF	11	C8
00053D10	64	1C	C8	92	2C	C8	66	3C	C8	43	43	C8	88	0B	B1	03
00053D20	59	07	39	0F	47	10	39	06	41	04	04	08	8D	03	D6	05
00053D30	зв	01	91	14	21	03	02	04	00	00	00	00	00	00	8C	0A
00053D40	00	00	00	00	00	00	00	00	00	00	78	06	CD	05	94	08
00053D50	9F	03	80	04	7B	03	FA	05	E7	03	18	0C	00	00	65	04
00053D60	00	00	41	0D	D9	02	CO	06	00	00	00	00	00	00	00	00
00053D70	00	00	77	04	00	00	DC	08	00	00	00	00	42	06	FD	0B
00053D80	E2	0B	77	04	17	0A	EC	04	F9	15	1B	00	77	04	13	02
00053D90	1B	00	2C	10	D2	OF	54	06	12	00	FD	AO	Α9	40	B0	80
00053DA0	84	00	B0	20	ΔF	80	E1	58	FD	A0	90	00	Δ9	16	BE	90
00000DA0	· · ·	~~	20				_					~~		20		
00000DA0			20	2.0					1.5	110	50			10		
0031FFF0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
			20			FF 07										_
0031FFF0	FF		20		FF	FF 07 01	FF	FF		FF	FF	FF				FF
0031FFF0 00320000	FF 88	FF 0B	20		FF 59	FF 07 01 00	FF	FF		FF 10	FF	FF				FF 08
0031FFF0 00320000 00320010	FF 88 8D	FF 0B	20	FF 03 05	FF 59 3B	FF 07 01 00 03	FF 39 91	FF		FF 10 03	FF 39 02	FF				FF 08 00
0031FFF0 00320000 00320010 00320020	FF 88 8D	FF 0B	20	FF 03 05	FF 59 3B 00	07 01 00	FF 39 91 00	FF		FF 10 03 00	FF 39 02 00	FF				FF 08 00 06
0031FFF0 00320000 00320010 00320020 00320030	FF 88 8D 00 CD	FF 0B	20	FF 03 05	FF 59 3B 00 9F	07 01 00 03	FF 39 91 00	FF	FF 47 21 00 7B	FF 10 03 00 03	FF 39 02 00	FF 06 04 00 05				FF 08 00 06 0C
0031FFF0 00320000 00320010 00320020 00320030 00320040	FF 88 8D 00 CD 00	FF 0B	20	FF 03 05	FF 59 3B 00 9F	07 01 00 03	FF 39 91 00	FF 0F 14 00 04 0D	FF 47 21 00 7B	FF 10 03 00 03	FF 39 02 00	FF 06 04 00 05 06				FF 08 00 06 0C 00
0031FFF0 00320000 00320010 00320020 00320030 00320040 00320050	FF 88 8D 00 CD 00 00	FF 0B	20	FF 03 05 0A 08 04 00	FF 59 3B 00 9F	07 01 00 03 00 00	FF 39 91 00 80 41 77 77	FF 0F 14 00 04 0D 04 04	FF 47 21 00 7B	FF 10 03 00 03 02 00 00 0A	FF 39 02 00 FA C0 DC	FF 06 04 00 05 06	FF 41 00 00 E7 00 00	FF 04 00 00 03 00 00	FF 04 00 78 18 00 00	FF 08 00 06 0C 00 00

Redo this process for the pitch multipliers at \$C53D9A and the ADSR data at \$C53E18:

Pitch multipliers

2021/05/10 06:54

00053D80	E2	0B	77	04	17	0A	EC	04	F9	15	1B	00	77	04	13	02
00053D90	1B	00	2C	10	D2	OF	54	06	12	00	FD	A0	Α9	40	BО	80
00053DA0	84	00	BО	20	AF	80	E1	58	\mathbf{FD}	AO	90	00	Α9	16	BE	90
00053DB0	в0	60	AF	A0	Α9	00	00	00	9C	00	00	00	00	00	00	00
00053DC0	00	00	00	00	F9	00	00	00	Β7	50	70	00	FD	AO	Α9	40
00053DD0	FD	AO	\mathbf{FD}	AO	29	CO	В9	FF	A9	00	00	00	00	00	00	00
00053DE0	88	00	Α7	A8	00	00	43	DO	43	00	43	00	7F	FF	00	00
00053DF0	C5	00	00	00	00	00	00	00	00	00	00	00	68	FC	6E	ΕO
00053E00	FF	00	8D	00	Α9	60	00	00	80	00	88	00	29	E4	95	00
00053E10	00	00	00	00	Α9	60	00	00	FF	F1	FF	EE	FF	ΕO	FF	F3
00053E20	ਸਸ	EO	ਸਸ	EO	FF	EO	ਸਸ	EO	चच	EO	ਜੁਤ	EO	FF	EO	FF	EF
000000000000000000000000000000000000000					_	_							_	_		
00000120		20						20		20						
003201F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
	FF FD	_	FF A9	FF 40	FF B0	FF 80	FF 84	FF 00	FF B0	_	FF AF	FF 80	FF E1	FF 58	FF FD	FF A0
003201F0	FF FD 90	_	FF A9 A9	FF 40 16				FF 00 60		_					FF FD 9C	
003201F0 00320200	FD	_	A9	40	в0	80	84	00	в0	FF 20	AF	80	E1	58	FD	AO
003201F0 00320200 00320210	FD 90	_	A9	40	в0	80	84	00 60	B0 AF	FF 20	AF A9	80	E1	58	FD 9C	A0 00
003201F0 00320200 00320210 00320220	FD 90 00	_	A9	40	В0 ВЕ 00	80	84 B0 00	00 60 00	B0 AF 00	FF 20 A0 00	AF A9 F9	80	E1 00 00	58 00 00	FD 9C B7	A0 00 50
003201F0 00320200 00320210 00320220 00320230	FD 90 00 70	_	A9	40	В0 ВЕ 00	80	84 B0 00 FD	00 60 00 A0	B0 AF 00 FD	FF 20 A0 00 A0	AF A9 F9 29	80	E1 00 00 B9	58 00 00	FD 9C B7	A0 00 50 00
003201F0 00320200 00320210 00320220 00320230 00320240	FD 90 00 70 00	_	A9	40	В0 ВЕ 00	80	84 B0 00 FD 88	00 60 00 A0 00	B0 AF 00 FD	FF 20 A0 00 A0	AF A9 F9 29	80	E1 00 00 B9	58 00 00	FD 9C B7	A0 00 50 00 00
003201F0 00320200 00320210 00320220 00320230 00320240 00320250	FD 90 00 70 00 43	_	A9	40	B0 BE 00 A9 00 00	80 90 00 40 00 00	84 B0 00 FD 88 C5	00 60 00 A0 00 00	B0 AF 00 FD A7 00	FF 20 A0 00 A0	AF A9 F9 29 00	80 00 00 C0 00 00	E1 00 00 B9	58 00 00	FD 9C B7	A0 00 50 00 00 00

ADSR data

00053E00	FF	00	8D	00	Α9	60	00	00	80	00	88	00	29	E4	95	00
00053E10	00	00	00	00	A9	60	00	00	FF	F1	FF	EE	FF	ΕO	FF	F3
00053E20	FF	E0	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	EF
00053E30	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	EE	FF	ΕO	FF	ΕO
00053E40	FF	ΕO	FF	ΕO	FF	ΕO	FF	EC	FF	F5	FF	ΕO	FF	FO	FF	ΕO
00053E50	FF	FO	FF	ΕO	FF	ΕO	FF	EA	FF	ΕO	FF	ΕO	FF	ΕO	FF	EA
00053E60	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO
00053E70	FF	ΕO	FF	F3	FF	ΕO	FF	ΕO	FF	$\mathbf{E}\mathbf{D}$	FF	FO	FF	ΕO	FF	ΕO
00053E80	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO	FF	ΕO
00053E90	FF	ΕO	FF	EC	FF	ΕO	7A	5C	C8	AO	5C	C8	DB	83	C9	9D
00053EA0	В4	C8	82	C8	C8	1E	64	C8	33	67	C8	69	6D	C8	C5	70
003203F0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
003203F0 00320400	FF FF	FF F1	FF FF	FF EE	FF FF	FF E0	FF FF	FF F3	FF FF	FF E0	FF FF	FF E0	FF FF	FF E0	FF FF	FF E0
		FF F1 E0	FF FF FF													
00320400	FF	F1	FF FF FF FF	EE	FF	EO	FF	F3	FF	E0	FF	E0	FF	EO	FF	EO
00320400 00320410	FF FF	F1 E0	FF FF	EE E0	FF FF	E0 E0	FF FF	F3 EF	FF FF	E0 E0	FF FF	E0 E0	FF FF	E0 E0	FF FF	E0 E0
00320400 00320410 00320420	FF FF FF	F1 E0 E0	FF FF FF	EE E0 EE	FF FF FF	E0 E0 E0	FF FF FF	F3 EF E0	FF FF FF	E0 E0 E0	FF FF FF	E0 E0 E0	FF FF FF	E0 E0 E0	FF FF FF	E0 E0 EC
00320400 00320410 00320420 00320430	FF FF FF FF	F1 E0 E0 F5	FF FF FF FF	EE E0 EE E0	FF FF FF FF	E0 E0 E0 F0	FF FF FF FF	F3 EF E0 E0	FF FF FF FF	E0 E0 E0 F0	FF FF FF FF	E0 E0 E0 E0	FF FF FF FF	E0 E0 E0 E0	FF FF FF FF	E0 E0 EC EA
00320400 00320410 00320420 00320430 00320440	FF FF FF FF FF	F1 E0 E0 F5 E0	FF FF FF FF FF	EE EO EO EO	FF FF FF FF FF	E0 E0 E0 F0 E0	FF FF FF FF FF	F3 EF E0 E0 EA	FF FF FF FF FF	E0 E0 E0 F0 E0	FF FF FF FF FF	E0 E0 E0 E0	FF FF FF FF FF	E0 E0 E0 E0	FF FF FF FF FF	E0 E0 EC EA E0
00320400 00320410 00320420 00320430 00320440 00320450	FF FF FF FF FF	F1 E0 E0 F5 E0 E0	FF FF FF FF FF FF	EE E0 E0 E0 E0	FF FF FF FF FF	E0 E0 E0 F0 E0 E0	FF FF FF FF FF FF	F3 EF E0 E0 EA E0	FF FF FF FF FF FF	E0 E0 E0 F0 E0 E0	FF FF FF FF FF FF	E0 E0 E0 E0 E0 F3	FF FF FF FF FF	E0 E0 E0 E0 E0 E0	FF FF FF FF FF	E0 E0 EC EA E0 E0

5. Adding the new sample data

Now we will add the data for the new sample. For more info on these 3 sample data, refer to the last 3 sections of this tutorial. Open the FF5.txt file in the sample archive and look at the 1st line. You'll see the three values that we need to add. Simply append 8C0A to the loop starting positions, append C000 to the pitch multipliers and append FFE0 to the ADSR data. The 3 following screenshots show this:

Loop starting position

Last update: 2019/02/12 11:26

00320060																
00320070	77	04	13	02	1B	00	2C	10	D2	OF	54	06	12	00	8C	A0
00320080	FF															

Pitch multiplier

00320260	00	00	68	FC	6E	E0	FF	00	8D	00	Α9	60	00	00	80	00
00320270																
00320280	FF															

ADSR data

00320460	FF	ED	FF	FO	FF	E0										
00320470																
00320480	FF															

6. Importing the Sample

We need after this to import the actual sample. Open 01_bass_drum.brr with HxD, select all (Ctrl+A), copy (Ctrl+C) then "paste write" at \$F20600. The two screenshots below show the beginning and the end of the sample:

Beginning

Finally we will add our new BRR pointer. Go at \$C53D1C and add 00 06 F2 (\$F20600 inverted). Note that in the screenshot below I replaced all the (now) useless sample data with FF. You can now use your new sample (\$40) in the instrument data of a song (32 bytes each starting at \$C53F95).

00053CF0	89	C7	18	9A	C7	37	A 7	C7	90	B3	C7	79	C0	C7	BF	CA
00053D00																
00053D10																
00053D20	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF						
00053D30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF						

This cover the mechanical part of the import. Further down will be detailed how to change ADSR data, loop starting positions and pitch multipliers. Not all samples in the BRR Sample Database are "plug and play" like the FF5 samples. Some require data modifications.

A. Loop Starting Position

Those two bytes are a looping position. As an example, for a sample of size \$0900, he could have a loop position of \$0850, meaning once the read reach \$0850, \$0850 to \$08FF will loop. That loop position would be written 5008 in the ROM. A loop position is obviously always smaller than the sample length. The loop starting positions of the samples in the BRR Sample Database are in the most case good.

B. ADSR Data

The ASDR Data is a two bytes value that apply an Attack Rate, Decay Rate, Sustain Level and Sustain Rate envelope to the sample. The format is the following, high bit of 1st byte tells if ADSR is enabled, otherwise Gain is used. Attack is on 4 bits, Decay on 3 while the 2nd byte has Sustain (3 bits) and Release (5 bits). The ADSR data of the samples in the BRR Sample Database are in the most case good. As an example and ADSR value of FFE0 is Attack Rate of 15, Decay Rate of 7, Sustain Level of 7 and Sustain Rate of 0.

7 6	5	4	3	2	1	0	Ļ
ENABL ++	DR	 	+	AI	R +	+	+
7 6 +++ SL	5 + 	4	3 + SR	2	1 +	0	+

Note that the ADSR settings on SNES are a bit different than the usual ADSR. A quick overview:

Α	Attack Rate	0 to 15	higher is shorter	maximum 4100ms (at 0)
D	Decay Rate	0 to 7	higher is shorter	maximum 1200ms (at 0)
S	Sustain Level	0 to 7	higher is louder	0% to 100% of initial sound
R	Sustain Rate	0 to 31	higher is shorter	maximum 38,000ms (at 1); 0 is infinite

C. Pitch Mutiplier

The pitch multiplier is a two bytes value that is added to the note multiplier of a note to result in the pitch of the played note (VxPITCH). The game use the following table and formulas to get the correct pitch of a note:

Note modifiers

Α:	\$0879
A#:	\$08FA
в:	\$0983
с:	\$0A14
C#:	\$0AAD
D :	\$0B50
D#:	\$0BFC
Ε:	\$0CB2
F :	\$0D74
F#:	\$0E41
G :	\$0F1A
G#:	\$1000
A :	\$10F3

Formulas

VxPITCH = (note_multiplier * pitch_multiplier) >> 16 if(pitch_multiplier < 0x8000) VxPITCH += note_multiplier

Note that the pitch multipliers in the BRR Sample Database are good for all the Squaresoft games except FF4, Romacing Saga 1, Seiken Densetsu 3, Super Mario RPG, Bahamut Lagoon and Treasure of the Rudras. Those game will need modifications to their pitch values.

From: https://www.ff6hacking.com/wiki/ - **ff6hacking.com wiki**

Permanent link: https://www.ff6hacking.com/wiki/doku.php?id=ff3:ff3us:tutorial:music:brr



Last update: 2019/02/12 11:26