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. 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

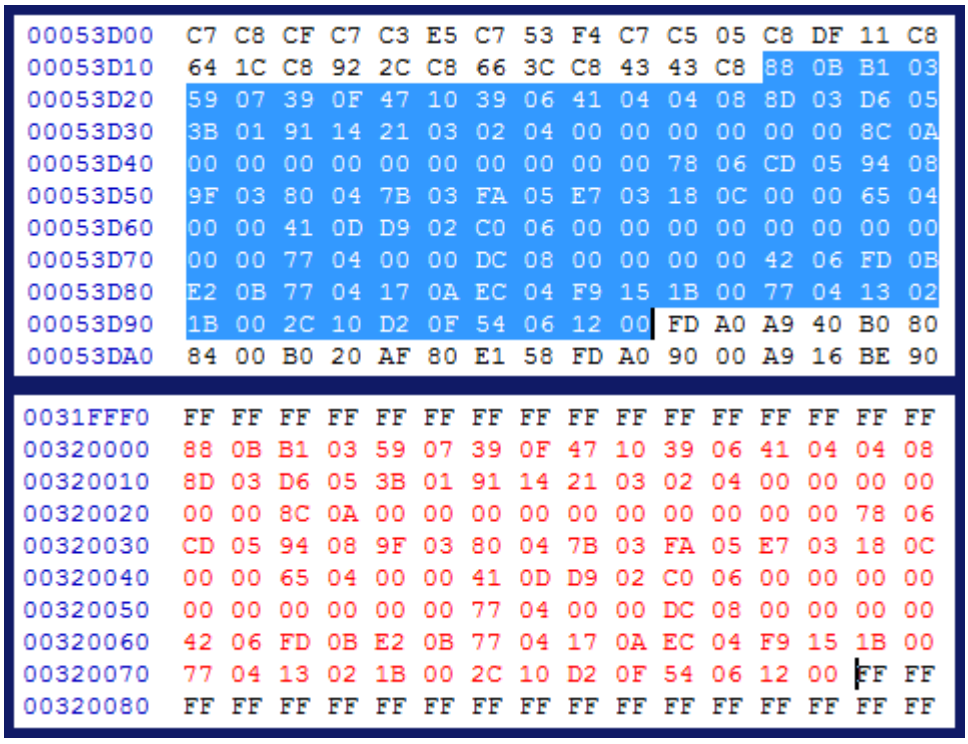
```
C5/041B:  7F1C3DC5 ADC $C53D1C,X  (loop starting positions)
C5/049B:  BF9A3DC5 LDA $C53D9A,X  (pitch multipliers)
C5/04DD:  BF183EC5 LDA $C53E18,X  (ADSR data)
```

Modified code

```
C5/041B:  7F0000F2 ADC $F20000,X  (loop starting positions)
C5/049B:  BF0002F2 LDA $F20200,X  (pitch multipliers)
C5/04DD:  BF0004F2 LDA $F20400,X  (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:



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

Pitch multipliers

```
00053D80  E2 0B 77 04 17 0A EC 04 F9 15 1B 00 77 04 13 02
00053D90  1B 00 2C 10 D2 0F 54 06 12 00 FD A0 A9 40 B0 80
00053DA0  84 00 B0 20 AF 80 E1 58 FD A0 90 00 A9 16 BE 90
00053DB0  B0 60 AF A0 A9 00 00 00 9C 00 00 00 00 00 00 00
00053DC0  00 00 00 00 F9 00 00 00 B7 50 70 00 FD A0 A9 40
00053DD0  FD A0 FD A0 29 C0 B9 FF A9 00 00 00 00 00 00 00
00053DE0  88 00 A7 A8 00 00 43 D0 43 00 43 00 7F FF 00 00
00053DF0  C5 00 00 00 00 00 00 00 00 00 00 00 68 FC 6E E0
00053E00  FF 00 8D 00 A9 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 E0 FF F3
00053E20  FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF EF

003201F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00320200  FD A0 A9 40 B0 80 84 00 B0 20 AF 80 E1 58 FD A0
00320210  90 00 A9 16 BE 90 B0 60 AF A0 A9 00 00 00 9C 00
00320220  00 00 00 00 00 00 00 00 00 00 F9 00 00 00 B7 50
00320230  70 00 FD A0 A9 40 FD A0 FD A0 29 C0 B9 FF A9 00
00320240  00 00 00 00 00 00 88 00 A7 A8 00 00 43 D0 43 00
00320250  43 00 7F FF 00 00 C5 00 00 00 00 00 00 00 00 00
00320260  00 00 68 FC 6E E0 FF 00 8D 00 A9 60 00 00 80 00
00320270  88 00 29 E4 95 00 00 00 00 00 A9 60 00 00 FF FF
00320280  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

ADSR data

```
00053E00  FF 00 8D 00 A9 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 E0 FF F3
00053E20  FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF EF
00053E30  FF E0 FF E0 FF E0 FF E0 FF E0 FF EE FF E0 FF E0
00053E40  FF E0 FF E0 FF E0 FF EC FF F5 FF E0 FF F0 FF E0
00053E50  FF F0 FF E0 FF E0 FF EA FF E0 FF E0 FF E0 FF EA
00053E60  FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0
00053E70  FF E0 FF F3 FF E0 FF E0 FF ED FF F0 FF E0 FF E0
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

003203F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00320400  FF F1 FF EE FF E0 FF F3 FF E0 FF E0 FF E0 FF E0
00320410  FF E0 FF E0 FF E0 FF EF FF E0 FF E0 FF E0 FF E0
00320420  FF E0 FF EE FF E0 FF E0 FF E0 FF E0 FF E0 FF EC
00320430  FF F5 FF E0 FF F0 FF E0 FF F0 FF E0 FF E0 FF EA
00320440  FF E0 FF E0 FF E0 FF EA FF E0 FF E0 FF E0 FF E0
00320450  FF E0 FF E0 FF E0 FF E0 FF E0 FF F3 FF E0 FF E0
00320460  FF ED FF F0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0
00320470  FF E0 FF E0 FF E0 FF E0 FF E0 FF EC FF E0 FF FF
00320480  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

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

```
00320060  42 06 FD 0B E2 0B 77 04 17 0A EC 04 F9 15 1B 00
00320070  77 04 13 02 1B 00 2C 10 D2 0F 54 06 12 00 8C 0A
00320080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Pitch multiplier

```
00320260  00 00 68 FC 6E E0 FF 00 8D 00 A9 60 00 00 80 00
00320270  88 00 29 E4 95 00 00 00 00 00 A9 60 00 00 C0 00
00320280  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

ADSR data

```
00320460  FF ED FF F0 FF E0 FF E0 FF E0 FF E0 FF E0 FF E0
00320470  FF E0 FF E0 FF E0 FF E0 FF E0 FF EC FF E0 FF E0
00320480  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 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

```
003205E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
003205F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00320600  8C 0A 00 00 00 00 00 00 00 00 00 94 FF 00 C1 2D
00320610  2B 2E 2E 12 94 A2 ED 10 BE 5A 1E 3A 1A A4 2B 3D
00320620  FF 2F C1 3F DE 3C B4 01 F2 BF 1F 4E 0D 12 F3 C0
```

End

```
00321060  03 04 4F 20 12 10 10 63 10 E3 04 00 10 11 00 31
00321070  00 02 00 04 00 11 10 00 00 00 00 10 04 00 F0 0F
00321080  00 F0 00 1F 00 01 00 00 01 00 00 00 00 00 FF FF
00321090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
003210A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

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 ($30) in the instrument data of a song (32 bytes each starting at $C53F95).

```
00053CF0  89 C7 18 9A C7 37 A7 C7 90 B3 C7 79 C0 C7 BF CA
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 00 06 F2 FF
00053D20  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00053D30  FF FF FF FF FF FF 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 [link|BRR 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.

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

A	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

```
A :    $0879
A#:    $08FA
B :    $0983
C :    $0A14
C#:    $0AAD
D :    $0B50
D#:    $0BFC
E :    $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.